English

Fujitsu Software BS2000

POSIX

Basics for Users and System Administrators

User Guide

Valid for:
POSIX A49
BS2000 V21.0B

Edition October 2023

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to: bs2000services@fujitsu.com.

Certified documentation according to DIN EN ISO 9001:2015

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2015.

Copyright and Trademarks

Table of Contents

POSIX Basics

1 Preface

POSIX (**p**ortable **o**pen **s**ystem **i**nterface for UNI**X)** is a range of UNIX-based standards which ensure the compatibility and interoperability of applications in a heterogeneous network. A heterogeneous network consists of computers from different manufacturers, as well as system and application software from different software suppliers.

The POSIX standard was defined as the national American standard by the Institute of Electrical and Electronics Engineers (IEEE) in 1989. It was then extended by the X/OPEN consortium, and in 1990 became adopted as the international standard.
(X/OPEN Portability Guide IV).

The X/OPEN Portability Guide IV, also known as XPG4 standard, comprises 7 volumes, including interface definitions for basic operating systems, programming languages, data management and networking. The BS2000 /OSD operating system as of V2.0 supports the XPG4 standards which are contained in the first two volumes:

- Volume 1: System Interfaces and Headers (approx. 350 program interfaces)
- Volume 2: Commands and Utilities (approx. 200 user interfaces)

In order to support these interfaces, the POSIX functionality was integrated into BS2000. POSIX designates both the IEEE standard and the BS2000 “POSIX” functionality. POSIX satisfies the requirements to allow its certification according to the XPG4 standard, which is carried out in two stages: at the end of 1995, BS2000 received the “XPG4 base branding” (XPG4) from “The Open Group” (previously X/OPEN), and around mid 1997 it received branding according to the “XPG4 UNIX profile” (also known as XPG4.2 or UNIX95). In addition, BS2000 with its POSIX subsystem has been certified as an internet server by “The Open Group” in 1999.

The kernel of the POSIX software product is implemented as a BS2000 subsystem. The library functions of the XPG4 standard are available to the user via a C library, and a defined set of commands is available via a shell (POSIX shell). The C library is a component of the product CRTE (Common RunTime Environment).

Application programs can be easily ported with POSIX, irrespective of the operating system being used. Programs consistent with XPG4 can therefore also run in BS2000 following recompilation.

POSIX program interfaces are offered together with BS2000 program interfaces. It is possible to use a combination of both BS2000 and POSIX program interfaces in the same program, albeit with certain restrictions.

1.1 Structure of the POSIX documentation

The following documentation is available to introduce you to POSIX and to support you in learning to use the POSIX subsystem and the POSIX shell in BS2000:

- The present manual, “POSIX - Basics for Users and System Administrators”, provides an introduction to working with the POSIX subsystem. The administrative tasks which arise in connection with the POSIX subsystem are also described. The BS2000 software products which can be used with the POSIX subsystem are also discussed.
- A complete description of the POSIX commands which you can use in the POSIX shell is given in the manual entitled "POSIX Commands" [1].
- All the information needed to use bs2fs file systems is provided in the manual "POSIX BS2000 filesystem bs2fs [2].

POSIX documentation in the BS2000 environment

In BS2000, the functions of the software products are extended so that you can also use the POSIX functionality with these products.

Access to the POSIX file system is possible through a range of utility routines. For example, you can process files in the POSIX file system using EDT.

By extending the CRTE in accordance with the XPG4 standard, you can write portable
C programs with V2.2A C library functions irrespective of the operating system being used.

The present manual, “POSIX - Basics for Users and System Administrators”, is required as a basis for accessing POSIX functionality from other software products.

1.2 Objectives and target groups of this manual

This manual is intended for:

- DP managers who would like an overview of POSIX.
- Nonprivileged BS2000 users who wish to work with POSIX. A basic knowledge of the UNIX operating system would be an advantage.
- Workstation users who in the past have worked mainly with UNIX systems, and who would now like to use POSIX. A basic knowledge of BS2000 is required.
- BS2000 system administrators and POSIX administrators. A good knowledge of the BS2000 and UNIX systems is required.

Basic terminology used in the UNIX operating system has been adopted in this manual for BS2000 users.

1.3 Summary of contents

This manual contains some chapters which are important for all users, and others which are only relevant for BS2000 system administrators, POSIX administrators or BS2000 group administrators.

Chapters 1 to 4 are intended for all users:

- Preface
- Introduction to POSIX
- Working with POSIX
- BS2000 software products in the POSIX environment

Chapters 5 to 8 are intended only for BS2000 system administrators, POSIX administrators and BS2000 group administrators:

- Installing POSIX
- POSIX subsystem and POSIX loader
- Administering and monitoring file systems
- Administering POSIX users

The reference section of the manual (chapters 9 to 10) contains:

- BS2000 commands for POSIX
- Privileges in POSIX
- Commands belonging to the POSIX shell
- Daemons in POSIX
- Directories created during an initial installation
- Special files created during an initial installation
- Administration files created during an initial installation
- Tuning measures
- POSIX logging files
- Local time in POSIX

The reference section is followed by an index designed to make it easier to use this manual.

Additional product information

Current information, version and hardware dependencies, and instructions for installing and using a product version are contained in the associated Release Notice. These Release Notices are available online at https://bs2manuals.ts.fujitsu.com.

1.4 Changes since the last edition of the manual

This edition of the manual contains the following changes with respect to the previous edition (July 2020):

- Recommendations regarding file system sizes for the initial installation have been adapted (see "Carrying out an initial installation with the POSIX installation program").
- Additional hints for upgrading to a new POSIX version have been added (see "Installing an upgrade for a new POSIX version").
- The description of the COPY-POSIX-FILE command's error handling has been modified (see "COPY-POSIX-FILE").
- The function "Journaling for file systems" is no longer supported and the related sections have been deleted or modified.
- The software list in section "Installing additional software" has been updated.
- Some text fields in the FHS masks of the POSIX installation program have been modified (see "POSIX installation program in interactive mode").
- The section "SNMP-Basic-Agent and SNMP-Standard-Collection" has been replaced by "NET-SNMP and SNMP-AGENTS".
- The *uuname* command is removed.
- Notes on "compactification" are included in the "Expand POSIX filesystems" section, and it is described that the *fsexpand* shell command has an option to suppress it if necessary.

1.5 Notational conventions

Certain notational conventions are used throughout this manual in the command / statement formats. These are explained further in the manual "BS2000 Commands [28].

The conventions are as follows:

- In continuous text, no distinction is made between constants and variables. All elements of the syntax and components of data structures, as well as file names, path names and commands appear in *italics*.
- For application examples, entries to the system are printed in **`bold fixed-pitch`** font. All lines input on character-oriented terminals have to be terminated with the enter key; inputs on block-oriented terminals are terminated with **EM DUE**. These symbols do not appear at the ends of the input lines shown in the manual. Some entries are terminal-specific, i.e. they differ for block-oriented and character-oriented terminals (for more information, see "Terminal support").

  Output from the operating system is printed in `fixed pitch`.
- References are given in quotation marks and appear in abbreviated titles in the manual. The complete title, together with a brief description, can be found under "Related publications" at the back of the manual.
- References in the text specify the corresponding page in the manual, together with the section or chapter number, as appropriate. References to subjects discussed in other manuals only use the abbreviated title of these manuals. You can use the index to find the corresponding section in the referenced manual.

! This symbol represents a warning which must be heeded in the interest of the system and operating reliability.

¡ This symbol is used to emphasize important information which must be taken into account.

2 Introduction to POSIX

This chapter is intended for all readers who want an overview of POSIX. It describes the role of POSIX in BS2000, the POSIX file system, the POSIX subsystem, and the POSIX security concept.

2.1 POSIX in BS2000

POSIX is directly in line with the BS2000 policy of open systems as defined in the “Open System Direction”.

This section provides a general overview of

- open systems
- the advantages of the POSIX standard
- the components of POSIX
- the hardware requirements for POSIX
- terminal support
- the BS2000 software products that have been modified for use with POSIX

New requirements facing information technology

For many decades, most companies had a strict hierarchical structure. Flat organizational structures were introduced in recent years. “Lean management” guarantees shorter routes and quicker, more flexible decisions.

This development also made new demands on information technology. Powerful, low-cost PCs and workstations form part of the basic layout of today’s working environment. The growing desire on the part of users for comprehensive solutions calls for the combination of these PCs and workstations with the present host systems to form an optimal overall system. All systems must be able to communicate with each other in order to use the common resources.

In this regard, networking heterogeneous systems is a basic requirement of information technology. The cost-effectiveness and user-friendliness of PCs and workstations must be linked with the high computer performance, storage capacity, availability, data consistency and security provided by host systems.

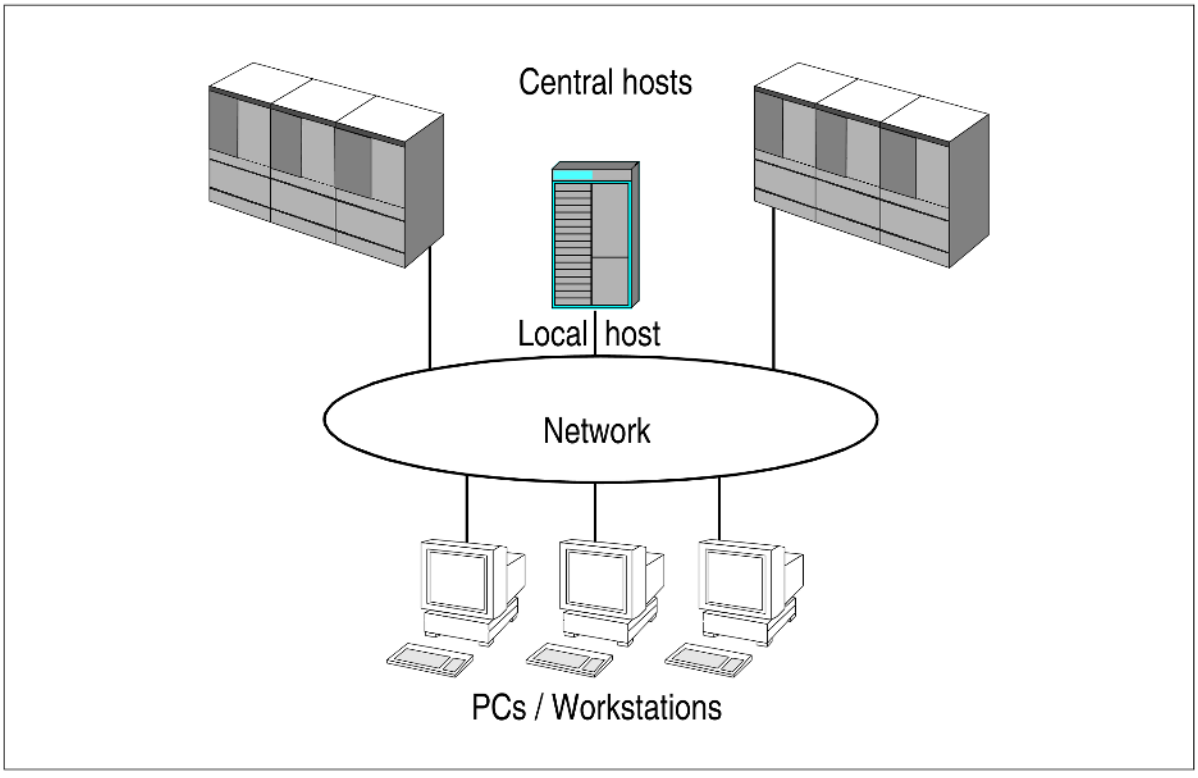

Figure 1: Networking heterogeneous systems

Communication between heterogeneous systems and the improved use of such systems are only possible if standards for operating systems interfaces are defined and adhered to.

2.1.1 A world of open systems

The task of the Institute of Electrical and Electronic Engineers (IEEE) was to develop extensive standards for portable operating system interfaces. These standards were combined under the “POSIX” concept. Through POSIX, proprietary systems become open systems. In open systems, applications can be transferred across system borders (portability) and can work with other applications (interoperability).

2.1.1.1 Openness through client/server architectures

Different computer worlds are integrated system-wide into client/server architectures. Decentralized, intelligent computers are linked with central mainframes, which makes distributed processing possible.

Within this homogeneous whole, the servers provide different services which are used by the clients. Server functions are mainly performed by mainframe and UNIX systems, while client functions are supplied mainly by PCs, workstations and UNIX systems. Client and server systems can be combined as desired.

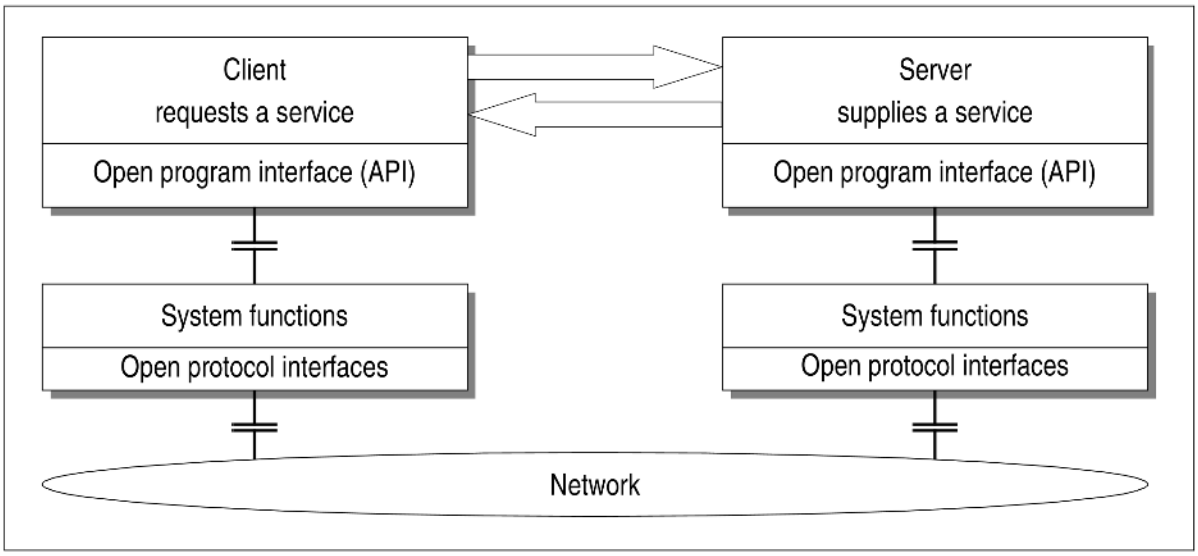

Figure 2: Role distribution in a client/server architecture

In a flexible architecture, the same computer can be used as a client for some services and as a server for others. This allows the best possible use to be made of the strengths of different computers:

- on PCs, standard applications run preferably under MS-Windows on PCs, especially in the areas of word processing, spreadsheet calculation and business graphics
- workstations are better suited for applications which have a high graphics performance, such as CAD (Computer-Aided Design)
- with their immense computer capacity, the use of large bulk storage areas, the outstanding security and the high degree of automation of the administration, BS2000 mainframes are especially suited as enterprise-wide servers.

BS2000 servers work in perfect cooperation with other servers in the network, irrespective of whether the server is a BS2000 or UNIX system. In this way for example, departmental data can be administered by different UNIX systems, while a central BS2000 server with high-performance peripherals is responsible for enterprise-wide data.

For the user, client/server architectures offer several advantages:

- The flexibility of procedural organization is increased.
- Information is more easily and more widely available.
- An optimum distribution of tasks among the different systems is achieved.
- The computer network can be adapted exactly to the size of the company using it.

2.1.1.2 BS2000 brings the UNIX systems and BS2000 worlds together

Workstation users can make use of the resources and performance of BS2000 via the POSIX interface, without having to know the BS2000-proprietary interfaces. In the network with BS2000, the workstation disk is extended to the host disk so that the workstation user can also process large volumes of data. For example, a developer can develop his/her applications on a workstation, and then compile, test, correct and execute the programs in BS2000.

The UNIX platforms and the BS2000 worlds can exist independently of each other, while sharing the same task and memory resources. The users can choose the best from both worlds: the standard interface and portability of UNIX systems, and a large number of the services provided by BS2000.

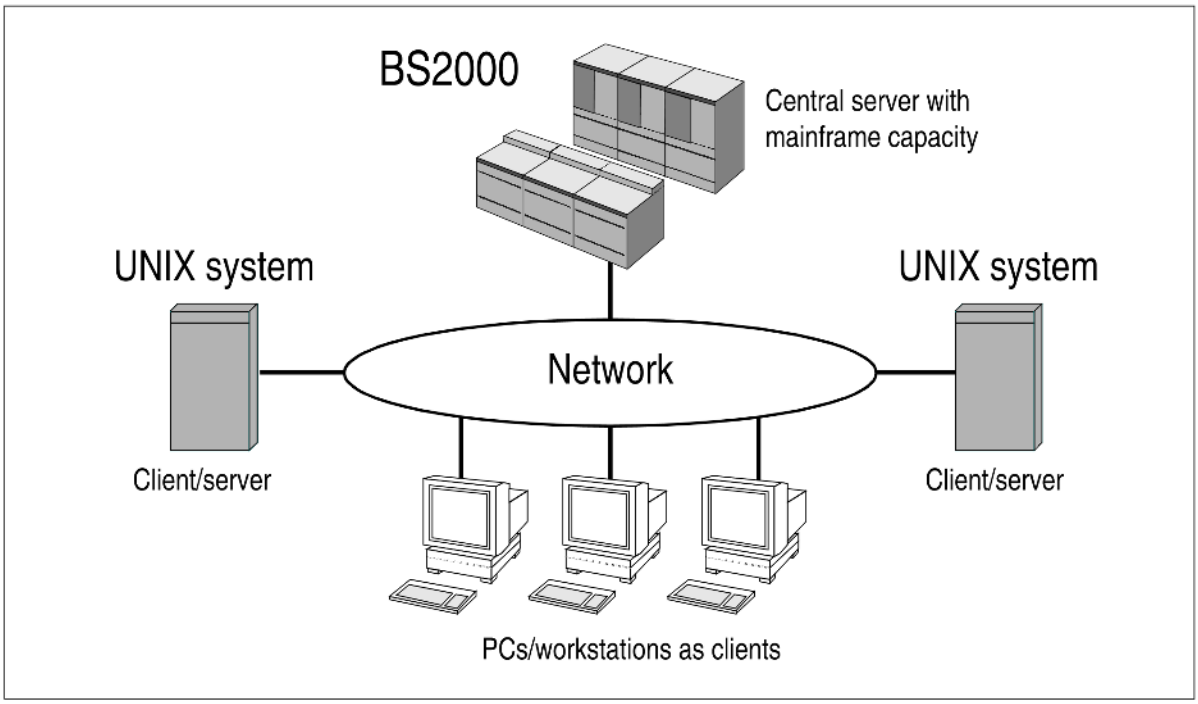

Figure 3: Position of BS2000 in the computer network

2.1.2 Advantages of the POSIX standard

The POSIX standard offers you the following advantages:

- Portability of application programs
- Interoperability of application programs
- Working with hierarchical UNIX file systems
- BS2000 as a server
- Distributed data storage
- Distributed processing
- Common development tools

These advantages are described in more detail below.

Portability of application programs

Application programs which are written in accordance with the POSIX interfaces can operate on all XPG4-compatible operating systems and hardware platforms. Portable application programs can operate as smoothly in BS2000 as, for example, on a UNIX platform.

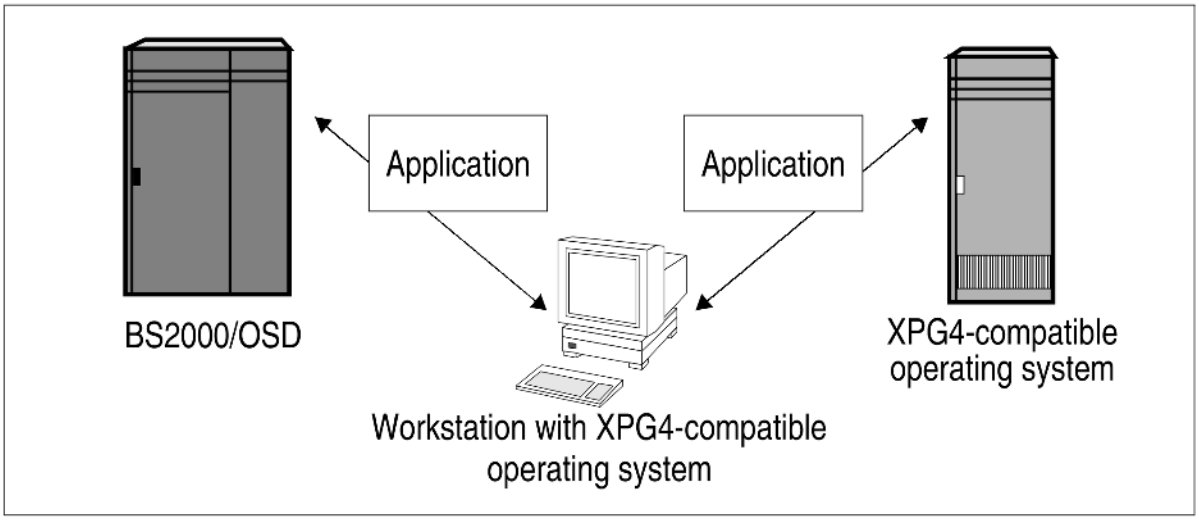

Figure 4: Porting an application to any XPG4-compatible system

Data and files from application programs in ASCII code must be converted to EBCDIC before being used with POSIX (see “Copying and converting files”).

Interoperability of application programs

Application programs which operate under different XPG4-compatible operating systems can exchange data with each other if the file formats match (see “Copying and converting files”).

Working with hierarchical UNIX file systems

The POSIX file system extends BS2000 to include a hierarchically structured file system. A POSIX file system is a container file in BS2000 with the structure of a UNIX file system (UFS). The POSIX file system consists of files (POSIX files) and directories (see “POSIX file system”). POSIX files can be created and processed by POSIX users.

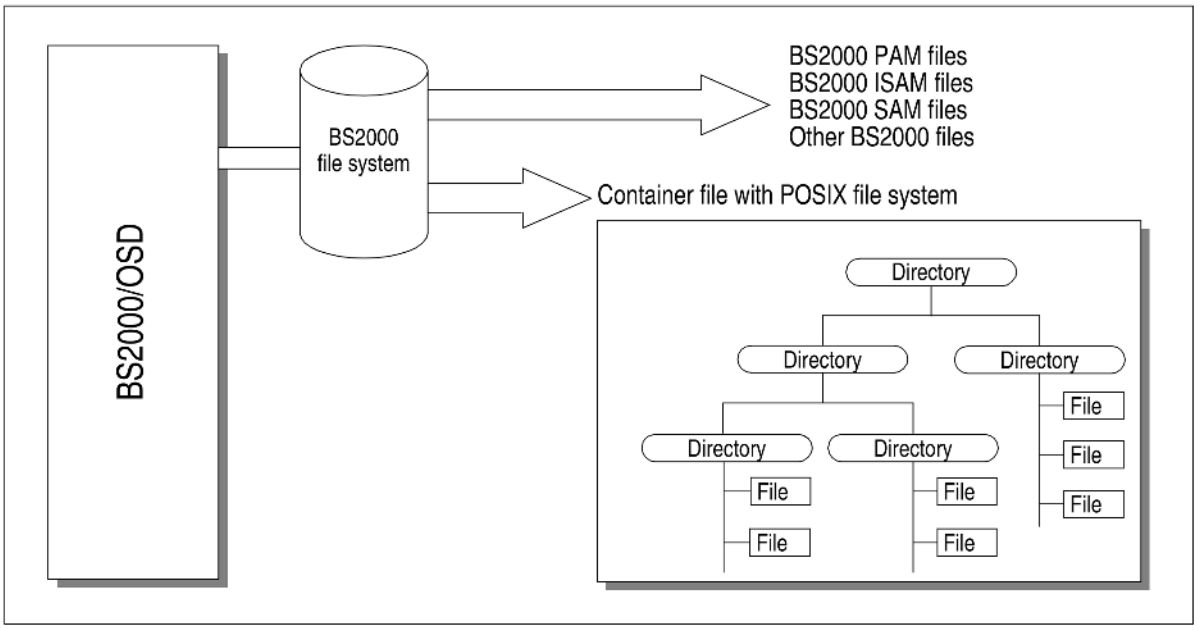

Figure 5: Supporting POSIX file systems with BS2000

With the aid of the software product NFS (**N**etwork **F**ile **S**ystem), local POSIX file systems can be mounted in remote computers, and remote UFS file systems in a local POSIX file system.

bs2fs file systems enable direct access to BS2000 files from POSIX (see the "POSIX BS2000 filesystem bs2fs" [2] manual.

BS2000 as a server

BS2000 can be used as a pure data server. In this case, the data (databases and files) is located on a BS2000 computer. The applications are stored on another computer. This is useful in the case of a low number of data accesses per transaction.

When used as a server for applications and data, the applications and data are located on the same BS2000 computer. This is useful if the data is accessed frequently per transaction.

As a file server, BS2000 offers the capacity, data access speed, and data access security of its memory subsystems.

As a backup server, BS2000 can store data resources from the network on its storage media, and use the HSMS protection mechanisms which are available there (see the HSMS manuals [21] and [22]).

As a print server, BS2000 makes its printers available via a distributed SPOOL and printing system (see the SPOOL manuals [32] and [33]). As a result, users of UNIX systems workstations can print their print jobs on BS2000 printers quickly and economically.

POSIX and the World Wide Web (WWW)

The following products are available for connecting BS2000 to the World Wide Web:

- APACHE (BS2000) is the name of the webserver on BS2000. APACHE (BS2000) is a port of the Apache World Wide Web server of the Apache Group.
- The communications software openNet Server provides the framework for the openNetworking in BS2000 and is further divided into products, such as DCAM and BCAM.
- interNet Services comprise FTP, TELNET, DNS, NTP, OPENSSH, SMTP Server and IMAP/POP3 Server. These products are ports of corresponding standard internet products which have been customized for the specific requirements of BS2000.
- JENV is the BS2000 environment for JAVA.

BS2000 applications can also present themselves with their data in the Internet. Direct access to the applications is implemented via a Common Gateway Interface (CGI), which is provided via the WWW server in BS2000. These applications may run with or without transaction control via openUTM and can access any stored BS2000 data.

The BS2000 security functions ensure that not all data is accessible to all users.

Bringing BS2000 applications into the WWW

Existing BS2000 applications can be made WWW-compatible with little effort. The product WebTransactions is thereby used to convert alphanumeric user interfaces (masks) into HTML format and pass them to the WWW browsers for output. Further information on this can be found in the WebTransactions manuals [38] and [39].

The number of offered host connections is being increased continuously. Special connections are currently offered by Fujitsu as project services with the necessary connection technology.

An overview of the documentation on WebTransactions can be found in the internet under https://bs2manuals.ts.fujitsu.com.
Select: *Software* > *openSEAS* > *WebTransactions* and then the product group or product you require.

Distributed data storage

With distributed data storage, you can work with both local and remote data. This means that you can position the data volumes at the most cost-effective location in a computer network.

You can access BS2000 files from a workstation once the BS2000 files have been copied to a POSIX file system. You can also copy BS2000 files to a POSIX file system and load them on a workstation.

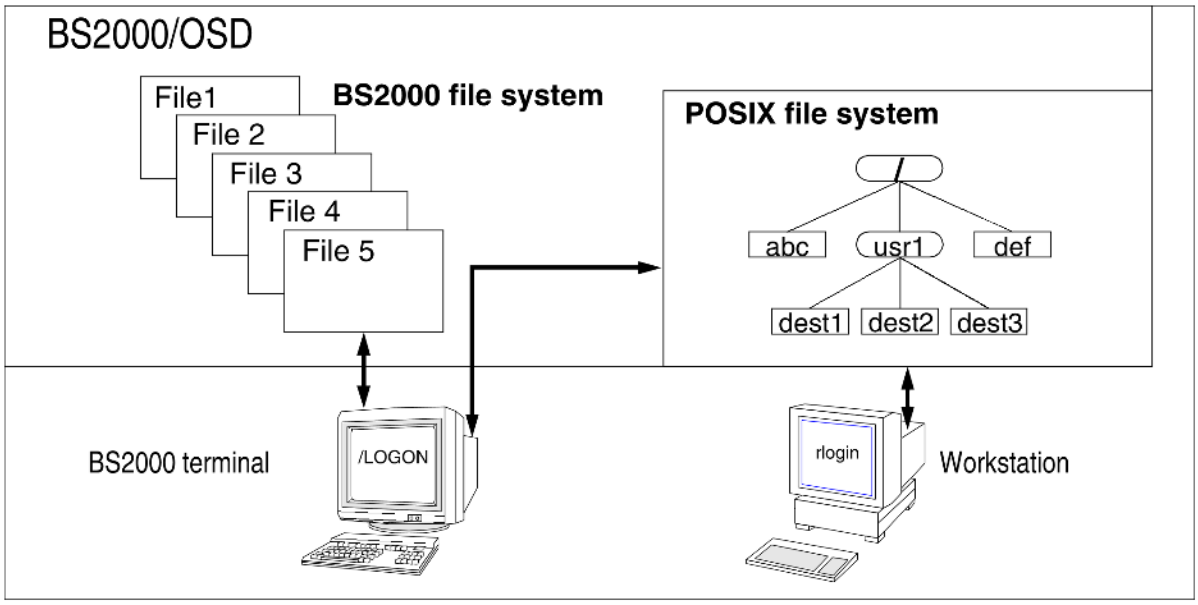

Figure 6: Distributed data storage

Distributed processing

With distributed processing, you can optimize use of the computing capacities. For this type of client/server architecture, processing takes place at the most suitable location. For example, certain processing operations such as input checks and calculations can run on the workstation (client), while database accesses and computer-bound evaluations run on the BS2000 computer (server).

Common development tools

UNIX system based development tools such as tools subject to the Gnu Public Licence (GPL) can be ported to POSIX without difficulty. This enables the creation of application programs that can run on UNIX systems and on POSIX.

2.1.3 POSIX components

POSIX comprises the following components:

- **POSIX subsystem**
  Incorporating a UNIX system kernel ported to BS2000.
- **POSIX shell**
  to establish the link between the system kernel and BS2000 using the shell commands which comply with the XPG4 standard.
- **Shared Libraries**
  Components for supporting shared objects are supplied with POSIX.
- **POSIX sockets and XTI**
  The program interfaces for the transport system and communications services are provided with POSIX sockets and XTI (X/Open Transport Interface), and are part of BS2000 basic configuration.

In addition, POSIX offers additive program interfaces which go beyond the XPG4 standard, network program interfaces (TLI, RPC and XDR) and header files for program development. It is also possible to log POSIX events in order to create SecureAuditTrails (can only be used with the BS2000 product SECOS).

2.1.4 Hardware requirements for POSIX

POSIX can operate on all mainframes on which BS2000 will run. POSIX is supplied as a component of BS2000 basic configuration as the selectable unit POSIX-BC. Please refer to the POSIX-BC Release Notice for information on the version dependencies between POSIX-BC and BS2000.

2.1.5 Terminal support

In addition to the block terminals used in BS2000, POSIX also supports the character terminals used in UNIX systems. These terminals connect to UNIX multiuser systems and are served by POSIX via network links. A character terminal is emulated when accessing POSIX via a workstation. In the case of UNIX workstations, this is a terminal such as the type 97801.

Block and character terminals differ in their modes of operation:

- BS2000 and POSIX commands can be entered from block terminals, however, the input of POSIX commands is subject to certain minor restrictions (see the "POSIX Commands" [1] manual).

  Character-oriented processing is not possible in the case of block terminals. Block terminals transfer the entire text input on the screen to the BS2000 computer as a data block. The block terminal itself performs control functions.
- In the case of character terminals, every character entered is immediately passed to the UNIX System, and from there transferred and mapped to the screen as a response to the input. The UNIX computer to which the terminal is connected executes control functions, such as cursor movement and uppercase and lowercase lettering. It also buffers the data transferred.

  In POSIX, character terminals are treated as files. They have a unique name and can be read from and written to. The same functions as for file access are used for this purpose.

  Screen-oriented applications - such as the vi editor in UNIX systems - require character-based operations. Therefore, they can only run if they are started on a character terminal.

Some inputs are terminal-specific, in other words they differ for block and character terminals:

Block terminal	**Character terminal**
@@d	**CTRL** + **D**
@@c	**CTRL** + **C**
@@/	**CTRL** + \
EM DUE	Enter key
@@z	**CTRL** + **Z**
-	**CTRL** + **S**, **CTRL** + **Q**, ...

Segmentation of the screen is not supported in BS2000. The screen always displays its contents from top to bottom. Input and output both occur in the lowest active line. When the screen is full, the contents are moved upwards a line, causing the uppermost line to disappear. Preceding lines can no longer be accessed.

2.1.6 BS2000 software products adapted to POSIX

POSIX-BC is a subsystem of BS2000. Different BS2000 software products have been adapted to the POSIX interfaces or ported to BS2000 based on POSIX:

- BLS (Binder Loader System)
- C/C++-Compiler
- COBOL85 / COBOL2000
- CRTE
- JENV (Java)
- AID
- EDT
- HSMS
- SBA-BS2
- SM2
- NFS
- SDF-A
- SECOS
- SPOOL
- Dprint
- SORT
- SOCKETS/XTI (POSIX-SOCKETS)
- TLI (POSIX-NSL)
- File transfer products (openFT, openFT-FTAM, openFT-AC)
- openUTM
- OMINS
- interNet Services
- openNet Server
- APACHE Webserver on BS2000
- WebTransactions

You can find further information on these BS2000 software products on the internet under https://bs2manuals.ts.fujitsu.com.

2.2 POSIX file system

A POSIX file system is a container file in BS2000 with the structure of a UNIX file system (UFS). Like in UNIX systems, it can consist of several file systems. It is hierarchically structured and consists of directories and files (POSIX files).

The *root* directory, which is marked by a slash (/), is at the top of the hierarchy and the directory structure branches from this point downwards. It is possible to branch from one directory to another directory or to a file, but it is no longer possible to branch from a file.There are no restrictions on either the number of directory levels or on the number of directories and files on a level. For this reason, a POSIX file system can be efficiently structured and organized.

Directories are also referred to as nodes of a POSIX file system. In these nodes, names of files or further directories are located. The user can assign the names for the directories and files, taking due account of the prevailing conventions.

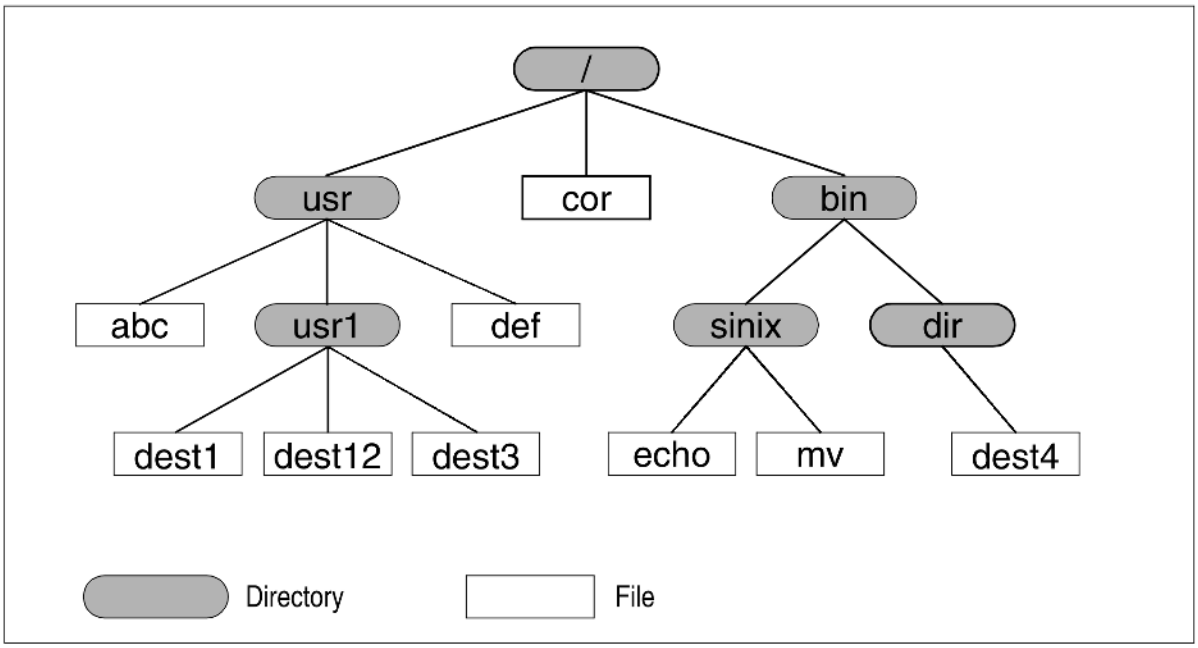

Figure 7: Hierarchical structure of a POSIX file system

2.2.1 Advantages of a hierarchical file system

A file system such as the hierarchically structured POSIX file system offers you several advantages:

- You can structure your data resources better.
- You can work with every file or directory in the entire file system, provided you have access permission for the corresponding file or directory (see “Access protection for files and directories”).
- Files can be easily transferred from the current directory to another. Three options are provided for this purpose:
  - You can physically copy a file to another directory with the POSIX command *cp* (copy); several physical copies of the file are then available.
  - You can alternatively transfer only the name of a file to another directory by means of the POSIX command *ln* (link). Several references to this file exist, but only one copy of the file exists physically.
  - You can rename a file or move it to another location in the directory tree by means of the POSIX command *mv* (move). *mv* creates no physical copy of the relocated or renamed file within a file system; it only modifies the entries in the superordinate directory.

  The POSIX commands *cp*, *ln* and *mv* are described in detail in the "POSIX Commands [1] manual.
- You can write your files to one or more directories. Consequently, files can be organized clearly and logically.
- Several files with the same name can be available in the same file system. However, the files must be stored in different directories.

2.2.2 Storing POSIX file systems in container files

POSIX file systems are stored in BS2000 in container files, which correspond to the partitions used to store file systems in UNIX systems. Container files are BS2000 PAM files which are located on a pubset. Container files must not be stored on private disks or on Net-Storage. Container files on Shared Public Volume Sets (SPVSs) can only be used at any given time by the POSIX of a BS2000 system. Container files and other BS2000 files may be located on the same pubset.

A container file is processed solely by the POSIX subsystem. Its content may not be modified using other access methods.

From the point of view of a POSIX file subsystem, a container file represents a file system that is administered by the ported UNIX system kernel.

POSIX administrators and BS2000 system administrators with root authorization can set up container files when installing new POSIX file systems with the POSIX installation program (see “Administer POSIX filesystems”). The size of the container file and thus of the POSIX file system is also determined during this process. Subsequent resizing is possible via *fsexpand*.

In order to limit the memory space for a user, a separate POSIX file system with the corresponding size can be created for this user. In this way, the present memory space in BS2000 is used more efficiently.

For the sake of performance, it is best not to have the container files of extensive, frequently used POSIX file systems on the same pubset as the container file of the root file system.

2.2.3 Information on file system coding (df)

The *-c* option enables the *df* command to output the coding of a file system (EBCDIC or ASCII).

This “coding” is defined using the parameter POSIX file system *marker* = *yes* (EBCDIC) or *no* (ASCII) when a ufs file system is configured. If the marker is not set, the file system is considered to be an ASCII file system under POSIX. This means that an ASCII-EBCDIC conversion takes place, depending on the *IO_CONVERSION* environment variable (see “Copying and converting files”).

The POSIX installation mask for file system administration also shows the type of coding.

2.2.4 Advantages of creating several POSIX file systems

Setting up several POSIX file systems has the following advantages:

- Greater data security:
  If one POSIX file system is destroyed, the remaining POSIX file systems are retained. Individual POSIX file systems can be selected for backup.
  Unmodified POSIX file systems can be excluded from a backup.
- Greater data protection
  Only currently required POSIX file systems are mounted and made available to the user.
- Improved clarity and structuring
  A POSIX file system can be set up specifically for a user or for a project.

2.2.5 Conventions for names of POSIX files and directories

Every file and directory in a POSIX file system has a unique path name. The path name specifies the position of a file or a directory within a POSIX file system and shows how it can be accessed. The path name consists of the names of all the preceding directories, starting with the *root* directory and the actual name of the file or the directory. In each case, the names of the directories are separated from each other by a slash. If the POSIX file system in figure 7 (POSIX file system) is the starting point, for example, then the path of the *root* directory to the *echo* file has the following name: */bin/sinix/echo*

If you create a file or a directory without specifying a path, the name is always entered automatically in the current directory in which you are working.

The POSIX file path names must be no more than 1023 byte long. The file name itself must be no more than 255 characters long.

2.2.6 Copying and converting files

POSIX files are byte-oriented and contain no data records. BS2000 files, on the other hand, contain record-oriented and/or PAM block-oriented data.

POSIX handles files in EBCDIC format by default, while UNIX systems, MS-DOS and Windows files are in ASCII format. ASCII files stored in the POSIX file system have to be converted before they can be processed in the POSIX shell.

Copy and conversion routines are available so that files in either format can be used in both types of file system. Conversion is bidirectional between EBCDIC.DF.03 and
ASCII-ISO 7-bit code. Conversion is only meaningful for text files.

In addition, conversion routines are available so that files can be converted and inverted from the EBCDIC.DF.04 character set to the 8-bit character set (ISO 8859).

Automatic conversion

The environment variable *IO_CONVERSION* is used to control whether files accessed with POSIX commands (e.g. *awk*, *cat*, *grep*...) on mounted ASCII file systems are converted automatically. The environment variable *IO_CONVERSION* is set to “NO” by default, i.e. no automatic conversion takes place. Automatic conversion is enabled with the following command:

```
export IO_CONVERSION=YES
```

The following are handled as ASCII file systems:

- Remote UNIX/Windows file systems mounted with NFS
- Local POSIX file systems with POSIX file system marker=N
- Remote POSIX file systems with POSIX file system marker=N which were mounted with NFS

To set automatic conversion as default for a POSIX user when the POSIX shell is started, the above *export* command must be entered in the *.profile* file in the home directory of the user concerned.

> ¡ Automatic conversion must be disabled if one of the following tools is used as the tools include conversion:
> *dd*, *iconv* or *bs2cp* with the *-k* switch.
>
> Handling archives and libraries:
> *ar* does not convert automatically since *ar* libraries often contain binary data.
>
> *pax* and *tar* convert automatically. However, a *pax* or *tar* archive may not be copied with *cp* if automatic conversion is enabled.

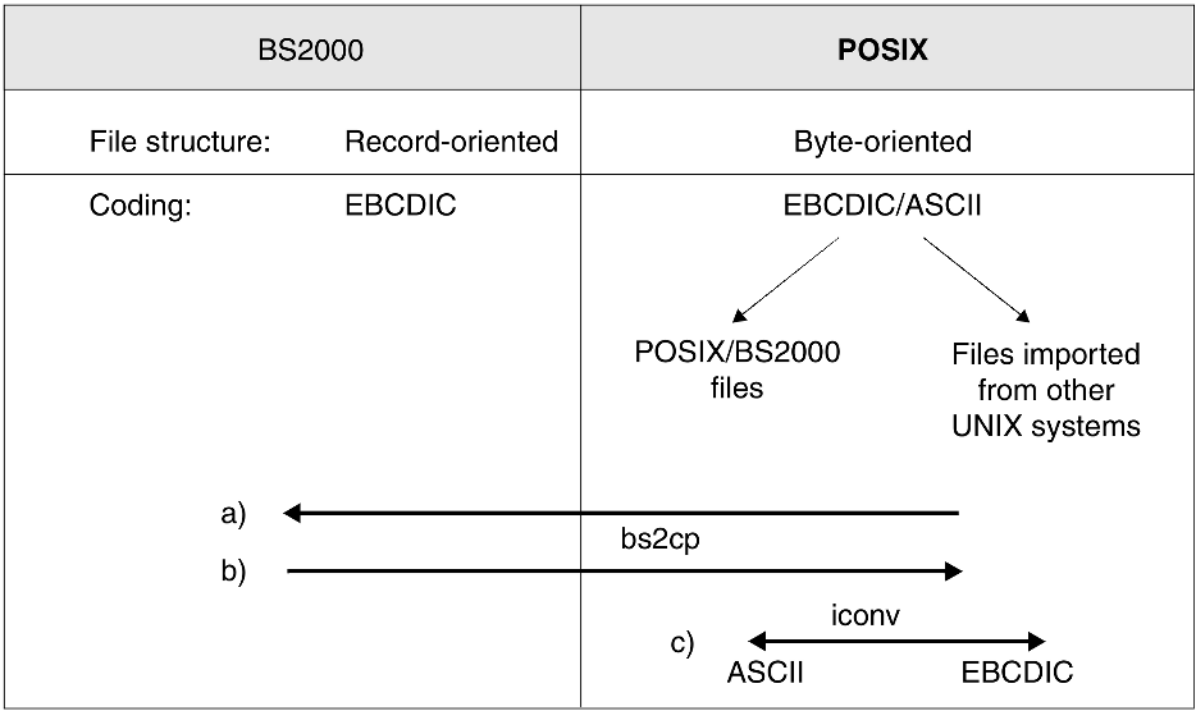

Figure 8: Copying and converting files

1. Transfer files from POSIX to BS2000 (from point of view of POSIX shell):

   Use the POSIX command *bs2cp* to transfer files from POSIX to BS2000. You can omit the *-k* option if the files are in EBCDIC format in both file systems.

   You can set the file attributes for the BS2000 file. Before issuing the copy command *bs2cp,* use the POSIX command *bs2file* to define the attributes of the BS2000 file. *bs2file* is mapped to the BS2000 command SET-FILE-LINK.

2. Transfer files from BS2000 to POSIX (from the point of view of the POSIX shell):

   Use the POSIX command *bs2cp* to transfer files from POSIX to BS2000. You can omit the *-k* option if the files are in EBCDIC format in both file systems.

   The following must be noted, depending on the type of BS2000 file (SAM, ISAM):

   - You have the choice of storing a SAM file as a text file, a binary file or a binary text file in the POSIX file system. You set this definition using the POSIX *ftype* command before issuing the *bs2cp* copy command.
   - ISAM files are always stored as text files in the POSIX file system.

3. The command used for conversion within the POSIX file system is *iconv*. This POSIX command converts the contents of the files.

2.2.7 Access to POSIX file systems in BS2000

When accessing POSIX file systems in BS2000 in **EBCDIC code,** the following points must be borne in mind:

- No measures are necessary when accessing from BS2000.
- When accessing from UNIX systems, files must first be converted, e.g. by means of the POSIX command *iconv* (see the "POSIX Commands" [1] manual).

When accessing POSIX file systems in BS2000 in **ASCII code,** the following points must be borne in mind:

- Access from BS2000:

  Versions of CRTE onwards offer *automatic* conversion to EBCDIC code, albeit with the following restrictions:

  - The associated file system must not be tagged as “generated by POSIX”
    (see “Definition of the POSIX file system” (Install POSIX subsystem)).
  - The file must be opened by means of the *fopen* call.
  - The file must not be opened in binary mode.
  - The *IO-CONVERSION* environment variable does not exist or has the value *YES*.

  In addition, versions of CRTE as of V2.0A offer *explicit* conversion by means of the *ascii_to_ebcdic* and *ebcdic_to_ascii* library functions.
- No measures are necessary when accessing from UNIX systems.

2.2.8 Accessing POSIX files

You can access POSIX files via POSIX software interfaces (see “POSIX program interfaces” (POSIX program interfaces)). Several BS2000 software products support access to POSIX files (see “BS2000 software products in the POSIX environment”).

2.2.9 Access to BS2000 files and PLAM library elements via the bs2fs file system

The BS2000 file system *bs2fs* permits direct and transparent access to BS2000 files under POSIX. Consequently both “simple” DMS files and PLAM library elements under POSIX can be edited as if they were POSIX files.

To do this, the user must specify the set of files with which they wish to work (using BS2000 wildcard syntax) and have these files mounted (by the system administrator) in POSIX as a bs2fs file system. This *mount* operation makes these BS2000 files accessible to the user in POSIX. These files can then be edited in the bs2fs file system using POSIX commands or from POSIX programs.

To enable these accesses, when the first access takes place in the bs2fs file system (first open), a background process (daemon) copies the files concerned from BS2000 to a special ufs fie system in POSIX which was mounted solely for this purpose (bs2fs container). Only the system may access this file which is stored temporarily in the bs2fs container. Access by a user take place only to the file mounted below the mount point in the bs2fs file system. The system redirects this access to the file stored in the bs2fs container.

In the case of write accesses, the file is locked for other users in BS2000, but the bs2fs file is not locked for other POSIX users. After processing in the bs2fs file system has been completed, a daemon transfers the file back to BS2000 again. After this has been done, it can then also be accessed there by other BS2000 users. As long as only information functions such as *ls* are executed, no copy function by a bs2fs daemon is initiated. The *ls* command merely outputs the files determined in BS2000 using FSTST as POSIX path names from the bs2fs mount point.

To summarize, use of the bs2fs file system offers the advantage that the user no longer needs to copy each individual file from BS2000 to the POSIX file system (e.g. with *bs2cp*) in order to be able to edit them with POSIX means. The user need only define the required BS2000 file set and have this mounted by the system administrator. The file set defined can consist either of files which already exist or ones which are to be created later. Transfer between BS2000 and POSIX and in the opposite direction is executed invisibly for the user of copy daemons as soon as a file is opened or when write processing has been completed.

The use of bs2fs file systems offers, for example, the following options:

- BS2000 files and PLAM library elements can be searched according to particular patterns using the POSIX command *grep*.
- *make* can be used to generate programs or program systems efficiently.
- Nested procedures in which multiple switches between the BS2000 command level and the shell take place can be replaced by pure POSIX shell scripts if the required BS2000 files are mounted beforehand in a bs2fs file system.

For further information, please refer to the "POSIX BS2000 filesystem bs2fs" [2] manual.

2.2.10 Accessing remote files

With POSIX, you can only access POSIX file systems which are located on the local computer. In order to be able to work with the file systems of a remote computer, the software product NFS (Network File System) must be installed on the remote and local computers. On the remote computer (NFS server), the file system to be mounted must be made available by means of the NFS command *share*; on the local computer (NFS client) it must be mounted by means of the NFS command *mount*. The remote file system can then be accessed from the local computer. NFS is available for BS2000, UNIX systems and Windows.
NFS is described in the manual "NFS" [8].

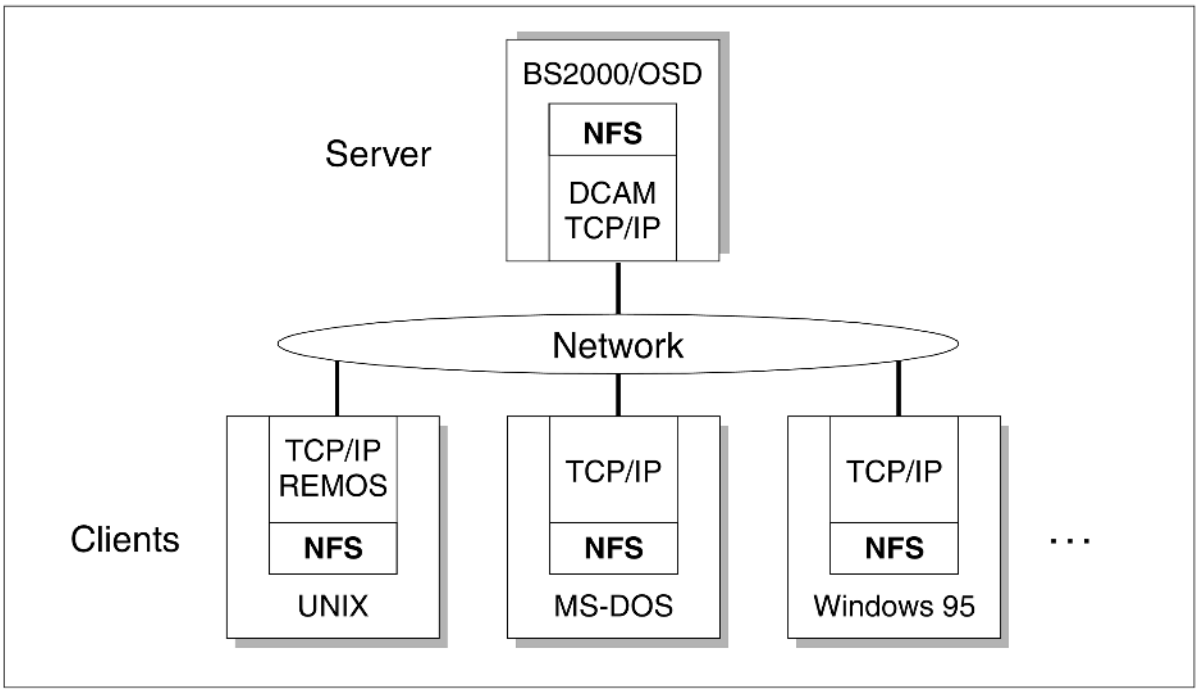

Figure 9: Distributed data storage in a heterogeneous computer network with NFS

r commands

Files or complete directory trees can be copied between POSIX-BS2000 and UNIX systems with the *rcp* (remote copy) command. Copying between two POSIX-BS2000 systems is also possible if a TCP/IP connection is established.
rcp carries out automatic ASCII/EBCDIC conversion. If this is not desired, *rcp* must be called with the *-b* (binary) switch.

Commands can be executed on a UNIX system with the *rsh* (remote shell) command.

A detailed description of the *rcp* and *rsh* commands can be found in the "POSIX Commands" [1] manual.

2.3 Large files in the POSIX file system

Previously, only files smaller than 2 gigabytes were supported in POSIX. This was because the data within a file was addressed with a variable of the data type integer (signed). This could only address a maximum of 2^{31}–1 bytes, i.e. 2 gigabytes.

This limit caused problems with different applications, for instance, with print files with memory-intensive graphics. Consequently, more users called for the maximum file size to be increased. This was also supported by the standardization authorities and led to a new standard being defined for large files.

Standard for large files

The central point of this standard is that a “long long” variable is used to address data within a file. This data type consists of an integer pair, which enables an address to be 2^{63}–1 bytes long, including the sign.

This new class of files should of course be as compatible as possible with the existing ones, so that existing programs can also work with large files without any great difficulty. That is to say, it should be possible to process the large files where possible with the same interfaces as previous files, at least in terms of syntax and semantics.

2.3.1 Large POSIX file systems

The maximum size of a POSIX file is also limited by the size of the file system in which it is located, i.e by the size of the container file in BS2000, see "Storing POSIX file systems in container files". Previously, a container file could not be larger than 2 gigabytes, as it was addressed internally by an integer-type variable.

Since a long long-type variable can now be used to address a file internally, a much greater range can be addressed. Consequently, container files and thus POSIX file systems as well can be larger than 2 gigabytes.

The following table shows the limit values for BS2000 files and POSIX file systems:

Size of a BS2000 file	**Size of the POSIX file system**
max. 4096 gigabytes (= 4 terabytes)	max. 1024 gigabytes (= 1 terabyte)

The difference between the maximum size of BS2000 files and the maximum container size is determined by the nature of the implementation in POSIX.

The size of a container file and thus of a POSIX file system is defined when it is created with the administration tool POSINST, see “Administer POSIX filesystems”.

2.3.2 Large POSIX files

Large POSIX files are files of a POSIX file system which can be larger than 2 gigabytes. Large POSIX files can only be created in POSIX file systems which are based on a large container and can thus exceed the limit value of 2 gigabytes, see the previous section.

The maximum size of a POSIX file is limited by the size of the container file which contains it. You can also specify a maximum file size in POSIX, which applies to all files of the POSIX file system (command *ulimit* or parameter FILESIZE in the POSIX information file).

Program interfaces for large POSIX files

To work with POSIX files, there are a number of C-library functions, such as *open()*, *close()*, which are made available by CRTE. A subset of these functions is available in 64-bit form, so that they can process large POSIX files. These functions have the same name, with the additional suffix “64”, e.g. *open64()*. Some data structures and data types were also converted to 64-bit form. For further information, see section “Program interface for large POSIX files”.

Shell commands for large POSIX files

Most file processing commands of the POSIX shell can recognize and usually process large POSIX files. They fall into two categories:

large file aware	This command can process large POSIX files correctly. Some of the commands in this category can only process large files up to a certain file size, for instance *cpio* up to a maximum of 8 GB.
large file safe	This command recognizes large POSIX files but rejects processing them, e.g. with an appropriate message.

To determine which category a command belongs to, see section “Commands belonging to the POSIX shell” in the column LFS.

BS2000 programs which work with POSIX files (e.g. HSMS, SORT, SPOOL) have been adapted in the same way where it is necessary or advisable to process large files.

2.4 POSIX as a subsystem in BS2000

POSIX is a privileged BS2000 subsystem which processes the jobs of privileged and nonprivileged users. The POSIX subsystem basically consists of three components:

- A UNIX system kernel which was ported into BS2000.
- BS2000 interfaces and services which set up a connection between the ported UNIX system kernel and BS2000.
- Separate routines for the initialization and termination of the POSIX subsystem.

The POSIX subsystem supports the POSIX file system.

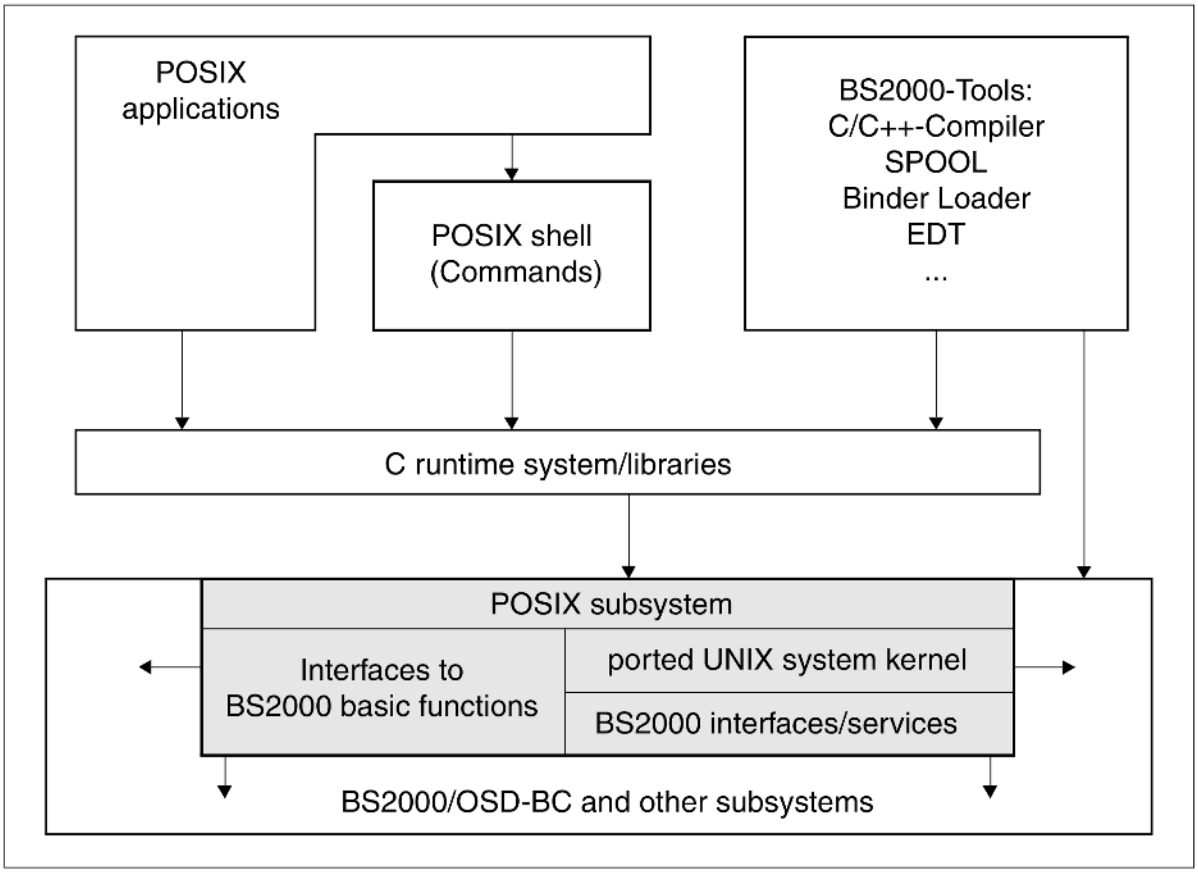

Figure 10: Embedding the POSIX subsystem in BS2000

The “"DSSM/SSCM" [29]” manual contains general information on subsystems in BS2000.

2.4.1 Administering the POSIX subsystem with DSSM

Dynamic subsystem management (DSSM) of BS2000 links the POSIX subsystem from link and load modules supplied in the following platform-specific program libraries:

- SYSLNK.POSIX-BC.<version> (/390)
- SKMLNK.POSIX-BC.<version> (X86)

The POSIX system kernel code - like the original UNIX system kernel code - contains various control parameters (tuning parameters) which a system administrator can set in the POSIX information file SYSSSI.POSIX-BC.010 in accordance with the specific application. These control parameters are used to configure the system kernel and to improve performance. In addition, the name of the root file system is entered in the POSIX information file.

In POSIX, the UNIX tuning mechanism is mapped to the parameter service of DSSM. The POSIX information file contains the name of the root file system in addition to the control parameters of the system kernel code, and other POSIX-specific control parameters. The contents of the POSIX information file are described in “POSIX information file”.

The POSIX information file is delivered to the customers together with other components. It is created as a SAM file and it already contains default values. The default values are selected so that the POSIX subsystem can operate in any environment without burdening the overall system by excessive use of resources. However, it may often be useful to adapt some control parameters, such as the maximum number of POSIX process, to the specific POSIX application and the resources available to the overall system.

If the system administrator enters an invalid parameter value in the POSIX information file, message POS1020 is returned. Instead of the invalid parameter value, the default value is entered in the internal subsystem parameter table.

If a POSIX information file is not available or if it cannot be opened when the POSIX subsystem starts, a corresponding message is returned and the subsystem is not started.

Some selected control parameters can also be modified using the priviledged POSIX command *usp* during an active POSIX session. The resources required are then available without the system needing to be rebooted.

2.4.2 POSIX process administration

In POSIX, the program executes in a process, the counterpart of the BS2000 task. POSIX processes are mapped to BS2000 tasks.

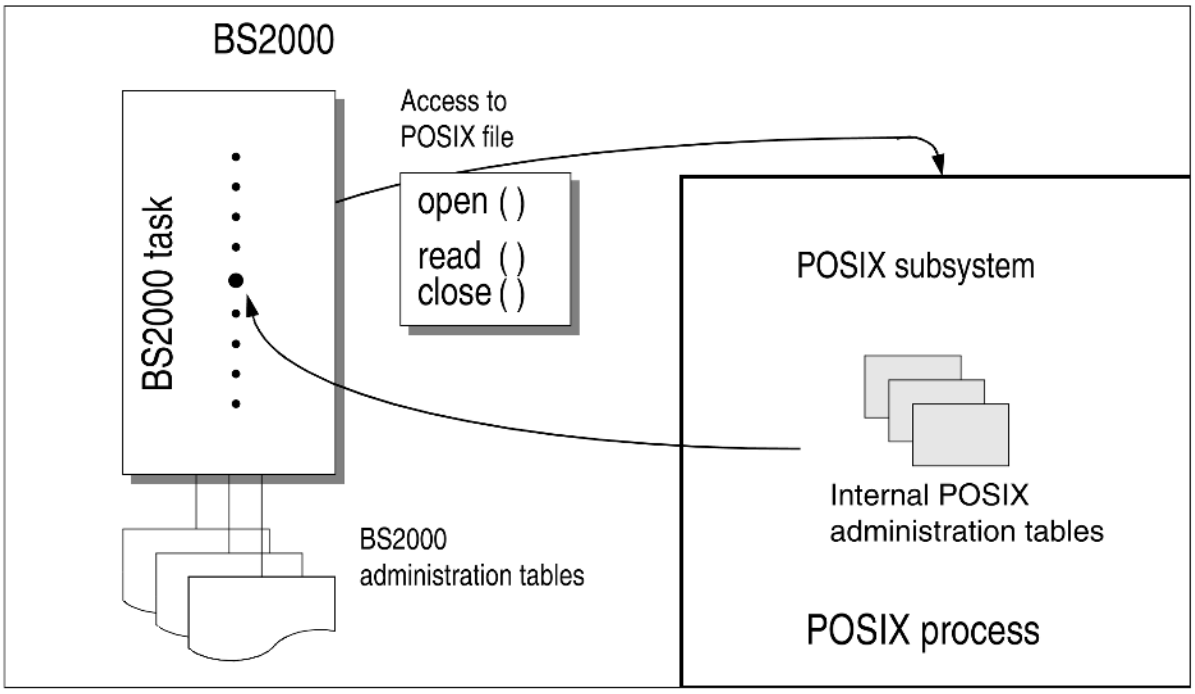

Figure 11: Accessing a BS2000 task on POSIX

In POSIX, all processes are structured hierarchically. The process hierarchy results from an initiating process (*init*) and further processes which are subordinate to this initiating process. This is commonly known as a parent-child relationship. The initiating process is the parent of all the processes. Processes directly subordinate to it are the child processes, who in turn can also have child processes. This order of priority can be continued up to a configurable maximum number of processes (*NPROC* control parameter in the POSIX information file).

The individual POSIX mechanisms and processes are described in more detail below.

fork

A new child process is created by calling the *fork* function of a parent process. The *fork* function creates a new process environment and copies selected information of the parent process for the child process. A separate address space, isolated from the parent process, is available to the child process. The child process can access all POSIX resources opened by the parent process.

The two process work independently of each other immediately after the function call. They can be differentiated by their respective return codes: the child process receives the
value 0, and the parent process receives the process identification (PID) of the child process.

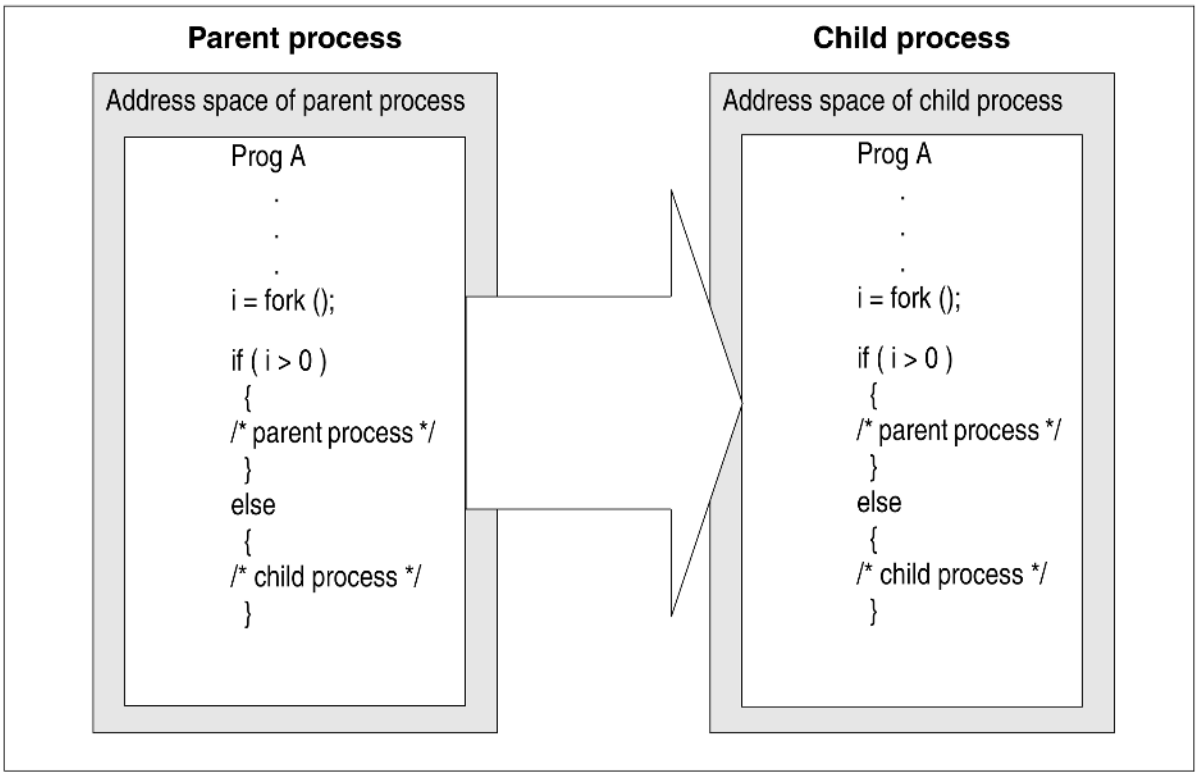

Figure 12: Mode of operation of the *fork* function

> DMS files and other BS2000 resources are not inherited when the *fork* function is called.

exec

If the *exec* function is invoked in a program, the current process environment is overlapped completely by a new one. As a result, another program can run in a child process than that running in the parent process, for example. The processes remain linked, however, by the parent-child relationship.

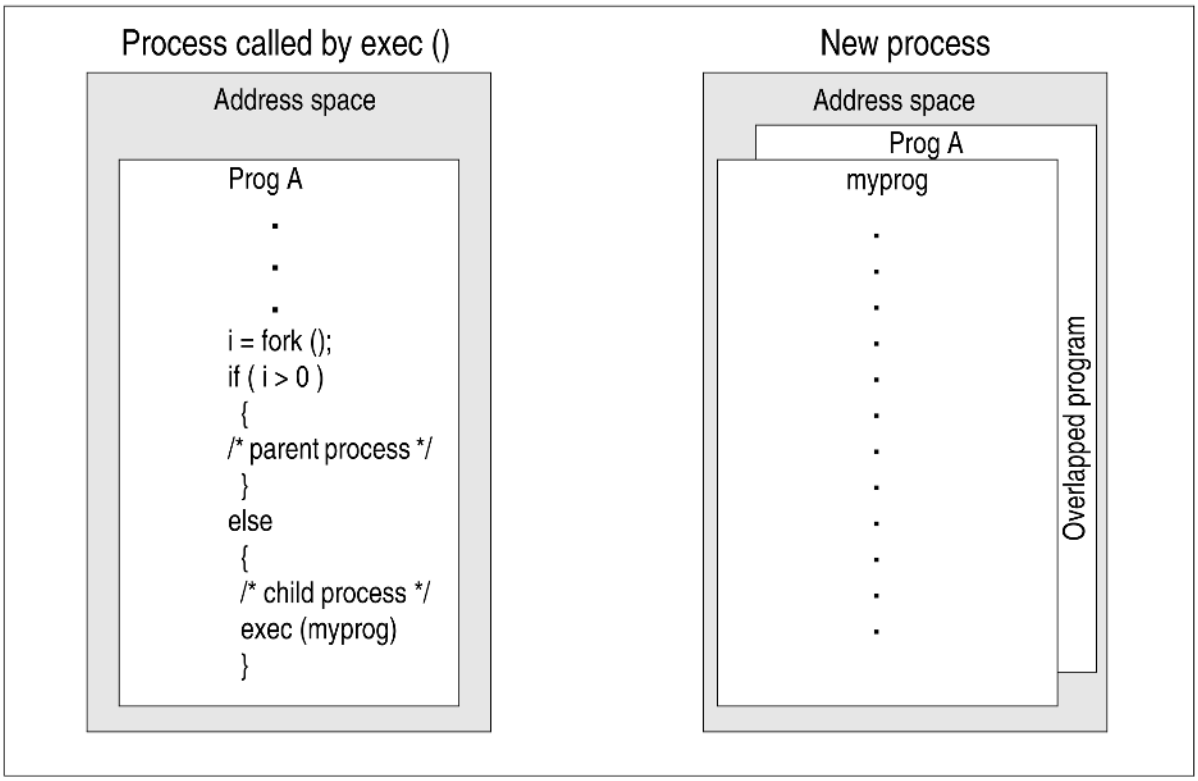

Figure 13: Mode of operation of the exec function

Combining fork and exec

The *fork* and *exec* functions can also be combined. The advantage of such a combination lies in the fact that partial tasks can be exported to another process. Once all partial tasks are exited, the process can be exited.

An example is the POSIX shell. For some commands, the POSIX shell starts a new process which overlays itself and starts the command execution program. The process is terminated and the POSIX shell continues when this program is exited.

pipe

The *pipe* function is available with POSIX for interprocess communication (application programming). This function creates a pipe function, i.e. a data repository of the type “first in - first out”. A process can use a pipe in order to transmit information to another process.

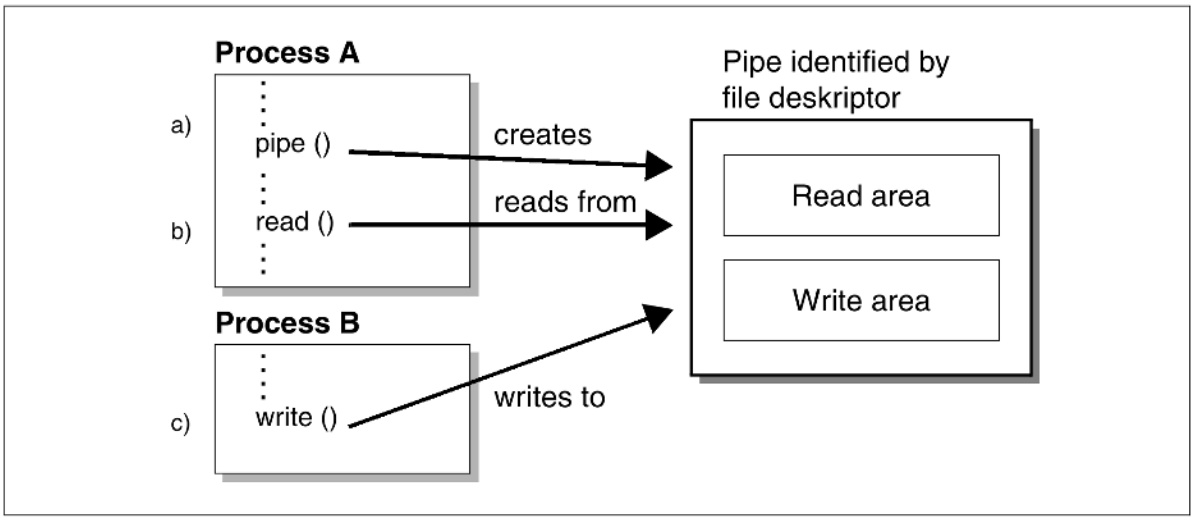

Figure 14: Mode of operation of the unnamed pipe

1. A pipe for reading and writing data is created when process A calls the *pipe* function. The pipe is identified by a file descriptor which marks a file as open.
2. Process B can write data to the pipe later and leave a message which process A can then read whenever it likes.
3. Process A calls the *read* function, specifies the file descriptor related to the pipe, and reads the message left by process B.

“copy-on-write” mechanism

When the *fork* function is called, the complete process environment of the parent is inherited by the child. This means that copying the complete address space can take a relatively long time. The POSIX-specific *copy-on-write* mechanism considerably improves performance. For one *fork* call, the address space is not copied, but merely marked. Consequently, the memory space is available to both the parent and child processes together. If either process accesses a page for the first time, the flag establishes that an action is to be performed. In this case, the appropriate memory page is copied for the child process so that both processes have their own version. In this way, only the memory pages which are actually required are copied. The *fork* call can thus essentially operate more effectively.

A further advantage of the *copy-on-write* mechanism is that a *fork* call is often followed by an *exec* call. In this case, there would be no point in copying the storage area, since it is overlapped by the *exec* call. Without *copy-on-write*, the entire storage area would have been copied and immediately overwritten again.

Holder task

POSIX uses a holder task. Initialization and termination operate under the control of this holder task. The holder task is not available to the POSIX subsystem during the subsystem session, i.e. between initialization and closure.

Caller task

The caller task is linked to the POSIX subsystem during the first SVC call, and is separated from it when the program is terminated. The calling task receives a POSIX process environment during the first POSIX system call.

Subsystem-private servers

There are two types of subsystem-private servers in the POSIX subsystem: system processes and daemons. System processes are implemented as BS2000 system tasks, daemons as nonprivileged BS2000 tasks.

While the command /START-SUBSYSTEM SUBSYSTEM-NAME=POSIX is being processed, only system processes are initialized as servers. Only once this has been done are daemons created.

2.5 Security concept

This section describes how POSIX was embedded into BS2000 in order to ensure the security of the overall system. The necessary functions were partly ported with the UNIX system kernel, and they are also partly a component of the BS2000 module SRPM (System Resources and Privileges Management). For more information on SRPM, please refer to chapter “Administering POSIX users”.

The security concept covers:

- user data administration
- group administration
- access protection for container files
- access protection for files and directories
- access protection for access via remote computer

2.5.1 User data administration

Interfaces for the security control of a user are defined in the POSIX standard. Specific information on a user is requested with these interfaces before this user may use an operating system. The following user data is available for authentication:

- User ID/login name of the user
- Password
- User number
- Group number
- Initial value for the working directory
- Program to be started

Further information can be added as required.

The user ID and the corresponding password entered by a user when signing on are checked against the administration information. If the input values are correct, the user is allowed to access to the operating system. For further information, please refer to section “Assigning POSIX user attributes”.

The following relationships exist between the POSIX user data and the BS2000 user data:

- The login name of the user and the BS2000 user ID are identical.
- The POSIX password and the BS2000 LOGON password are identical.

For the remaining POSIX user data, there is no equivalent on the BS2000 side.

In POSIX, only user names in uppercase letters are supported in login names.

POSIX user data is stored and administered by BS2000 user administration for this purpose, (for further information, please refer to chapter “Administering POSIX users”). The data is integrated in the BS2000 user data as POSIX user attributes. POSIX user data is accessed via BS200 user and system administrator commands.

2.5.2 Group administration

Group administration in POSIX corresponds to that in UNIX. It differs from group administration in BS2000 in the following ways:

- In POSIX, the sole function of groups is to distribute access permissions to files. In BS2000, groups also serve to control the use of resources, such as disk storage, computer power etc.
- In POSIX, a user can belong to up to 16 groups simultaneously, whereas in BS2000 a user may only belong to one group.
- BS2000 groups are arranged hierarchically, but this feature is not available in POSIX.
- A user can change the current group in POSIX, whereas this is not possible in BS2000.

Because of these substantial differences, POSIX and BS2000 groups exist side by side but are administered separately: the POSIX groups on the shell level, the BS2000 groups on the BS2000 level. This corresponds to the different protection mechanisms of POSIX and BS2000 files.

If the hierarchy is omitted, POSIX and BS2000 groups can be defined identically, i.e. they then contain the same users.

For further information on group administration, please refer to section “Administering BS2000 and POSIX groups”.

2.5.3 Access protection for container files

POSIX file systems are stored in container files. Container files are BS2000 PAM files in non-key format. They are protected against unauthorized access via standard access control of BS2000 (ACCESS/USER-ACCESS attribute).

The POSIX installation program specifies container files as non-sharable, and with ACCESS=*WRITE. These protection attributes must not be modified. Nor is it permissible to assign a file password.

The user of a POSIX file does not require an access right for the container file.

2.5.4 Access protection for files and directories

Access protection for files and directories is implemented with the following protection mechanisms in POSIX:

- User IDs
- Passwords for user IDs
- Combining user IDs for groups
- Permission bits for files and directories

These protection mechanisms prevent a user from reading and modifying the files and directories of another user without authorization.

Access protection by user ID, password and group number

Anyone who wishes to use POSIX must have a user ID created by the BS2000 system administrator on the corresponding BS2000 computer. Users themselves can define or change a password in order to protect their user IDs against unauthorized access.

See also section “Access protection for access via remote computer”.

Users can be combined to form groups. Consequently, files and directories can be made accessible to all members of a given group. For this purpose, the system administrator must allocate a group number to every user. Users with the same group number belong to the same group (see section “Administering BS2000 and POSIX groups”).

Access protection with permission bits

Each file and directory is automatically assigned permission bits and the user and group number of the generating process when it is created. These permission bits are preset as a default for specific accesses. Permission bits are available for the following three user classes:

- owner of the file
- group to which the owner belongs
- other

Each of these user classes has one permission bit for read permission (**r**ead), write permission (**w**rite) and execute permission (e**x**ecute).

Example

Owner: `r w x`

Group: `r w -`

Other: `r - -`

The permission bits apply exclusively to their user class. If, for example, only the owner has access permission for a file, neither the user class *group* nor the user class *others* may work with this file.

Access permissions have different meanings for files and directories:

Access permission	File	Directory
read	read	Read entries
write	write	Delete/create entries (files)
execute	execute	Execute/scan

Before the first permission bit for the owner, there is an identifier, which is assigned automatically. It has the following meaning:

- `-` file
- `b` block-oriented device
- `c` character-oriented device
- `d` directory
- `l` symbolic link

The permission bits can be modified by means of the POSIX command *chmod*. A user with the user number 0 can modify the permission bits of all files and directories, whereas the owner can only modify his/her own files and directories. Even if someone from the user class *group* or *other* has full access rights to a file or directory, he/she cannot change the permission bits.

The permission bits for the user class *group* are assigned in accordance with the group membership of the owner. When creating a new file, the group number and thus the group membership of the current directory is accepted.

The currently valid permission bit mask can be output or modified by means of the POSIX command *umask*. This permission bit mask determines which access rights the files and directories which you can now create in the current shell or in one of your subshells are to receive.

If you modify the permission bit mask using *umask*, this modification is valid either until you define a new value with *umask* or until you terminate the shell in which you called *umask*.

POSIX administrators can define the value of the permission bit mask by means of *umask* in the */etc/profile* file. Since the */etc/profile* file is executed by every login shell, the defined access rights are valid for every user logged onto the system.

For further information on the POSIX commands *chmod* and *umask*, please refer to the "POSIX Commands" [1] manual.

2.5.5 Access protection for access via remote computer

POSIX can also be used from remote computers (see section “Accessing the POSIX shell”). Users who signed on to POSIX using the *rlogin* command are entered as local users in the BS2000 user administration of the central computer. The BS2000 module SRPM checks the access rights during rlogin processing.

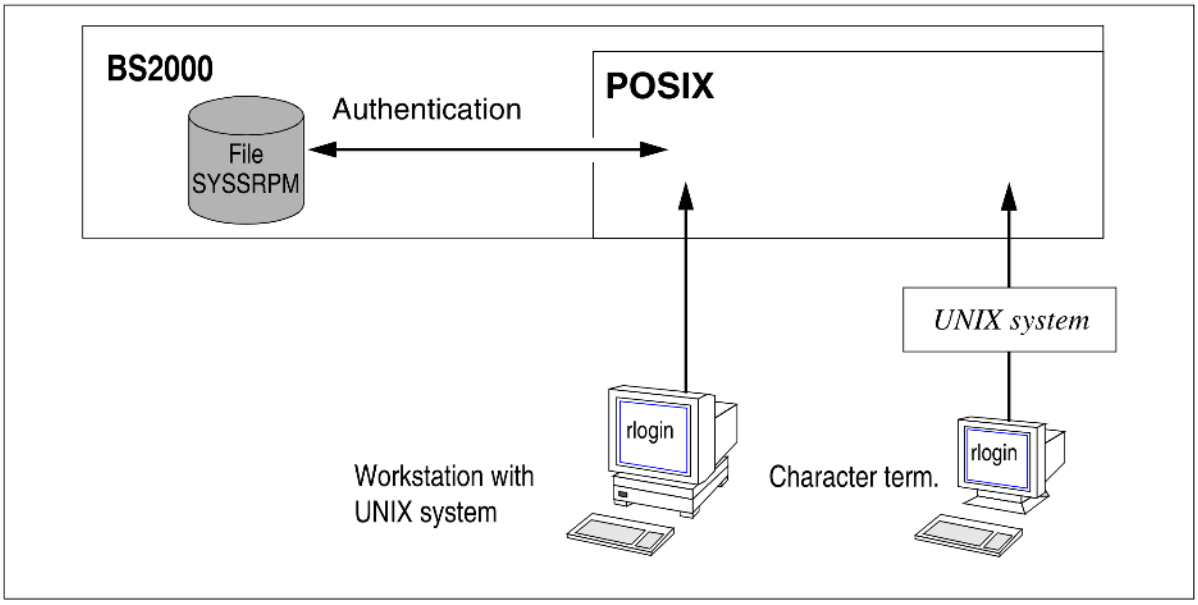

Figure 15: Access protection for access via rlogin

The following applies for the *rcp* and *rsh* commands:

Access rights are handled as in UNIX, i.e. authorized hosts and users are taken from the *$HOME/.rhosts* file. For SECOS, this can also be set using BS2000 systems, see section “Defining access rights for users of remote computers”.

> **ATTENTION!**
> Entries in the */.rhosts* file (in the Root directory) enable commands to be executed under TSOS and are thus relevant to security!

3 Working with POSIX

This chapter is aimed at all POSIX users. It contains information on the POSIX shell and program interfaces. In addition it contains a sample session.

3.1 The POSIX shell

The POSIX shell is the interface which links you to the POSIX subsystem via the C runtime system/libraries. The following diagram shows the structure of POSIX in BS2000 and the embedding of the POSIX shell.

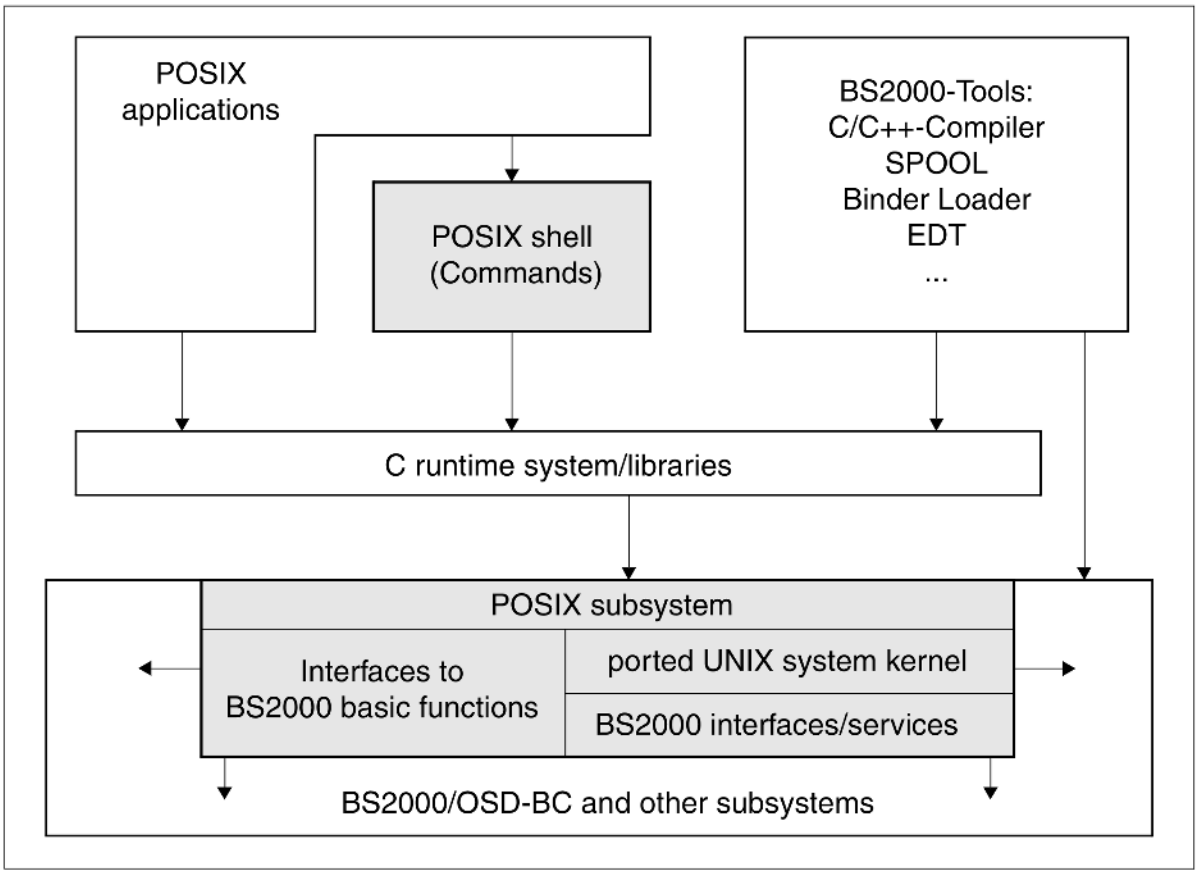

Figure 16: Structure of POSIX in BS2000 and the embedding of the POSIX shell

The POSIX shell is a command interface which you can use in addition to the BS2000 command interface (see figure 17).

All the POSIX shell commands are available to you once you have successfully accessed the POSIX shell (see "Accessing the POSIX shell "). You can begin to enter BS2000 commands again once you have exited the POSIX shell.

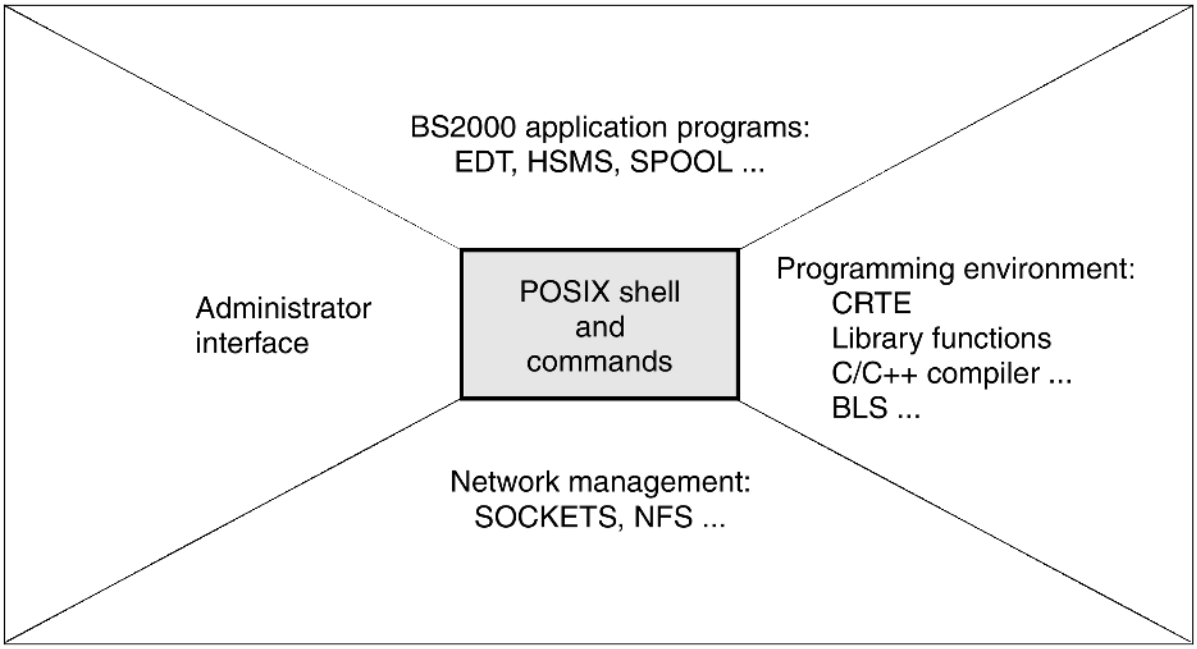

Figure 17: The command level POSIX shell in the POSIX subsystem

The POSIX shell reads commands from a terminal or from a POSIX file, interprets them according to specific rules and executes them. A file which contains commands for the POSIX shell is called a shell procedure (shell script).

The operation and performance of the POSIX shell depend on whether the terminal at which the user is working is a block terminal or a character terminal.

The POSIX shell offers you a comprehensive command language which can be applied as a programming language. You can use the available commands to create your own programs and execute them without compiling them beforehand.

3.1.1 Accessing the POSIX shell

The POSIX shell can be accessed in the following ways:

- via a BS2000 terminal (block terminal)
- from a terminal of a UNIX system (character terminal)
- via a terminal emulation

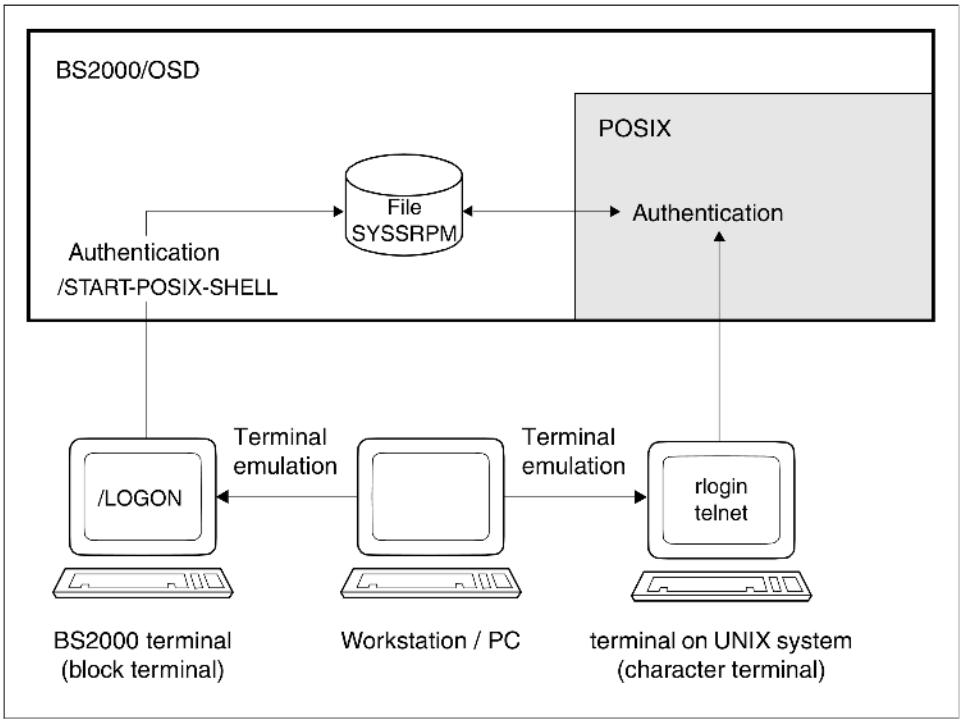

Figure 18: Accessing the POSIX shell

Access via a block terminal

Following successful BS2000 LOGON, every BS2000 user can request the POSIX shell by means of the BS2000 command START-POSIX-SHELL. This command has no operands since all it does is set up the POSIX environment and call the program which is entered in the SYSSRPM file for the appropriate user is invoked (see user attribute “Program” in section “Assigning POSIX user attributes”).

Users entering the POSIX shell as a default program in their user data can work interactively with the POSIX shell once the BS2000 command /START-POSIX-SHELL has been entered. All the commands and functions of the POSIX shell are then available to these users. The POSIX shell incorporates commands specially for interactions between BS200 and the POSIX subsystem (see section “Commands belonging to the POSIX shell”).

Return to BS2000 is only possible when a user exits the POSIX shell by means of the POSIX command *exit*.

Access from a character terminal

Access via rlogin

The user can log on to a BS2000 computer from a terminal on a UNIX system by means of the *rlogin* command, provided the BS2000 computer user has access rights. A user ID and the corresponding password are also required when using the BS2000 computer. After logon, POSIX can be used as if in local mode.

In order to connect to a BS2000 computer, the user must enter the following command in the POSIX shell:

```
rlogin <host> [-l <userid>]
```

If the user does not enter a user ID, the ID under which the user is logged on at the local computer is used. In an *rlogin,* the password is requested for the user ID. The password is verified via the BS2000 component SRPM (**S**ystem **R**esources and **P**rivileges
Management): the details of the BS2000 user ID and the password are checked against the access control attributes in the home pubset. If they correspond, the user is granted access to the POSIX subsystem. If the SECOS product is being used, access by way of LOGON protection can be controlled even more closely.

An account number must be used for accounting each remote login system run. This can be defined with the ADD-USER or the MODIFY-USER-ATTRIBUTES, the operand in question being POSIX-RLOGIN-DEFAULT.

If you use *rlogin* to access POSIX, then you can access BS2000 commands only to a limited extent (e.g. because of the missing SYSFILE environment).

Access via telnet

The *telnet* daemon *telnetd* implements direct access to BS2000 via the *telnet* protocol from the UNIX system or from the PC via the *telnet* application, which behaves as a character terminal for POSIX. Access control is the same as with *rlogin*, i.e. via BS2000 access mechanisms. Access without specifying a password, as is possible between UNIX systems (via an entry in the *.rhosts* file), is not supported.

Parallel operation of TELNET from the interNet Services (BS2000-TELNET) selectable unit and from POSIX is not possible without configuration changes as default port number 23 is used in both cases. The daemon *in.telnetd* in POSIX is thus only ever started if the comment character '#' in front of the entry *telnet* in the configuration file of the *inetd* daemon (*/etc/inetd.conf*) has been deleted.

If BS2000-TELNET is active in the system, the following steps are also required to use TELNET from POSIX:

- Deactivate the BS2000 batch job TELSR using /STOP-TELNET-DEMON
- wait some minutes then activate the *in.telnetd* daemon with the following command:

  ```
  kill -s HUP <pid von inetd-Daemon>
  ```

  (the *pid* can be calculated with the formula: `ps -ef|grep inetd`)

Parallel operation of the two TELNET servers can, for example, be enabled by operating the POSIX TELNET server on an alternative port. To do this, a new service must be entered in the */etc/services* file (e.g. "telnetx 1023/tcp"), the service name "telnet" must be changed to "telnetx" in the */etc/inet/inetd.conf* file, and the comment character at the start of this line must be removed. "kill -s HUP <pid>" must subsequently be used to start reconfiguration of the *inetd* . The TELNET client programs can then use the alternative port number (1023 in the example) to access the POSIX TELNET server and the default port number (23) to access the BS2000 TELNET server.

Access via a terminal emulation

The third means of accessing the system is via a terminal emulation. To do this, the user logs on at a workstation or PC and then starts a terminal emulation, which must emulate either a terminal on a UNIX system or a BS2000 block terminal:

BS2000 terminal emulation

A block-type terminal is available to the user in the case of system access via a BS2000 terminal emulation (e.g. EM9750 or MT9750). The user has to provide authentication as usual in BS2000 and can then input BS2000 commands and /START-POSIX-SHELL as with a BS2000 terminal (see "Accessing the POSIX shell").

Terminal emulation for UNIX systems

Terminal emulations for UNIX systems are available for workstations with UNIX systems with graphical, OSF/Motif-based user interfaces and for Windows PCs (e.g. EM97801, SINIX-TE). These emulate a UNIX systems character terminal and the user can input commands such as *rlogin* (see "Accessing the POSIX shell").

3.1.2 Points to remember when working with the POSIX shell

Defaults in the user environment

The POSIX shell is started following successful access to the POSIX subsystem. Before the POSIX shell prompt appears, the following defaults are set in the user environment:

- The POSIX shell initializes the standard shell variables. It assigns default values to the following shell variables: HOME, HZ, IO_CONVERSION, LANG, LOGNAME, MAIL, PATH, PROGRAM_ENVIRONMENT, PS1, PWD, SHELL, TERM, TTY, TZ and USER.
  If a variable <x> is already defined by the BS2000 S variable SYSPOSIX.<x>, this value is used. The shell variables HOME, HZ, LOGNAME, MAIL, TERM, TTY and USER must not be set by the user.
- The */etc/profile* file is executed.
- The *$HOME/.profile* file is executed, if it has been created.

Special functions (P keys, Ctrl)

You can assign functions to the P keys **P3**, **P4** and **P5** by calling the POSIX command *bs2pkey* as follows:

P3	with @@c (**CTRL** + **C**)
P4	with @@c (**CTRL** + **D**)
P5	with @@z (**CTRL** + **Z**)

The program can either be called in the POSIX shell (without options) or placed in the */etc/profile* file. In the latter case, the program is activated every time the shell is called.

3.1.3 POSIX loader

The POSIX loader is a component of the POSIX subsystem. It manages system global, user-specific or session-specific program caches of variable size where ready-to-run core images of POSIX programs are stored and copied to memory for execution. Load processes in POSIX can be speeded up considerably with the POSIX loader. An ample description of the POSIX loader is contained in the section “POSIX loader”.

Setting up program caches

When POSIX starts, no program caches are set up. The global program cache is set up implicitly in accordance with the entries in the POSIX information file. The global program cache can also be set up later by the super user with the *posdbl* command. The user-specific and session-specific program caches are set up by the current user with the aid of the *pdbl* command.

Storage in a program cache

After the program caches have been set up, POSIX programs are stored in a program cache in the following way:

Implicitly: At the first call of a (non-builtin) POSIX command (basic shell and extended shell), the program is loaded via BLS. Its core image is stored in the global program cache, loaded into memory and subsequently started.

- Explicitly: POSIX programs may be stored in the global program cache (by the super user with the *posdbl* command) or in a user-specific/session-specific program cache (by the user with the *pdbl* command).

Loading during a program call

Each time a (non-builtin) POSIX command is called via the *exec* system call, the following conditions are tested in the specified order:

- Was the task created by *fork*?
- Is debugging deactivated for the program?
- Is the corresponding core image stored in a program cache?

If one of the conditions does not apply, the program is loaded in the “classic" manner via BLS and is subsequently started. Otherwise, the program stored in a program cache is directly copied to memory and started. Bypassing BLS means the program is not completely embedded in the BS2000 program environment. The resulting restrictions are described in the section “Loader process”.

3.1.4 Entering commands from the POSIX shell

Once you have accessed the POSIX subsystem, the POSIX shell is started.

If you are using the POSIX shell interactively, the POSIX shell outputs the value of the PS1 environment variable as a prompt before it reads a command. For a privileged user this is normally either a dollar sign ($) or a hash sign (#) followed by a space character(`' '`).

Command input has the following format:

`Command``[ `**`options`**`][ `**`parameter`**`] ...`

For *command*, you must enter the name of a POSIX command or a shell script to be executed. For *options*, enter control statements to execute commands. For *parameter*, you must enter a call argument, which the POSIX shell transfers to *command*. You can also specify several call arguments, independently of the command.

If you are working on a character terminal, command names and call arguments must be separated either by tabulator characters or by spaces. The last call argument, and thus also command input, must be terminated by pressing the enter key, or **EM DUE** in the case of a block terminal.

You cannot start pure BS2000 programs from within the POSIX shell.

If the screen line for input is too short, you have two options:

- Continue writing to the end of the line without pressing the enter key. When the command is completely entered, terminate it by pressing the enter key.
- Continue the line with backslash and the enter key. The backslash character (\) counteracts the command termination function of the enter key. You can then continue entereing the command. When the enter key is pressed without backslash (\), the command is executed.

Every POSIX command returns a value to the POSIX shell in which it was invoked, namely its exit status. An error-free procedure returns a value of 0, an errored procedure a value other than 0.

If, when working with a character terminal, a command outputs information on the screen which is larger than the size of a screen page, you can stop the output by pressing the keys **CTRL** + **S** and then continue by pressing the keys **CTRL** + **Q**. Block terminals do not support this function.

For more detailed information on command input, please refer to the "POSIX Commands" [1] manual.

3.1.5 Commands for large POSIX files

Large POSIX files (> 2 GB) can only be processed using commands which are suited to this (= large file aware). A number of commands recognize large files but will reject processing them (= large file safe). File processing commands which are neither *large file safe* nor *large file aware* should not be used to process large files.

The following table indicates which commands are *large file aware* and which are *large file safe* (in alphabetical order):

Attribute	**Commands**
large file aware	awk, bs2cp, cat, cd, chgrp, chmod, chown, cksum, cmp, comm, compress, cp, cp, cpio, cut, df, diff, du, dumpfs, file, find, fsck, fsexpand, funzip, fuser, getconf, grep, hd, head, iconv, join, ln, ls, more, mv, pax, rcp, rmdir, sh, split, strings, sum, tail, tar, touch, tr, ulimit, uncompress, unzip, unzipsfx, wc, zcat, zip, zipcloak, zipgrep, zipinfo, zipnote, zipsplit
large file safe	ar, csplit, dd, edt, edtu, egrep, expand, fgrep, fold, logrotate, nl, od, paste, sort, unexpand

3.2 POSIX program interfaces

The POSIX program interfaces are available in addition to the BS2000 program interfaces. As a result, pure BS2000 programs, pure POSIX programs and merged programs can be executed. Merged programs contain both BS2000 and POSIX program interfaces, and are subject to certain restrictions (see "Restrictions for programs with merged functionality").

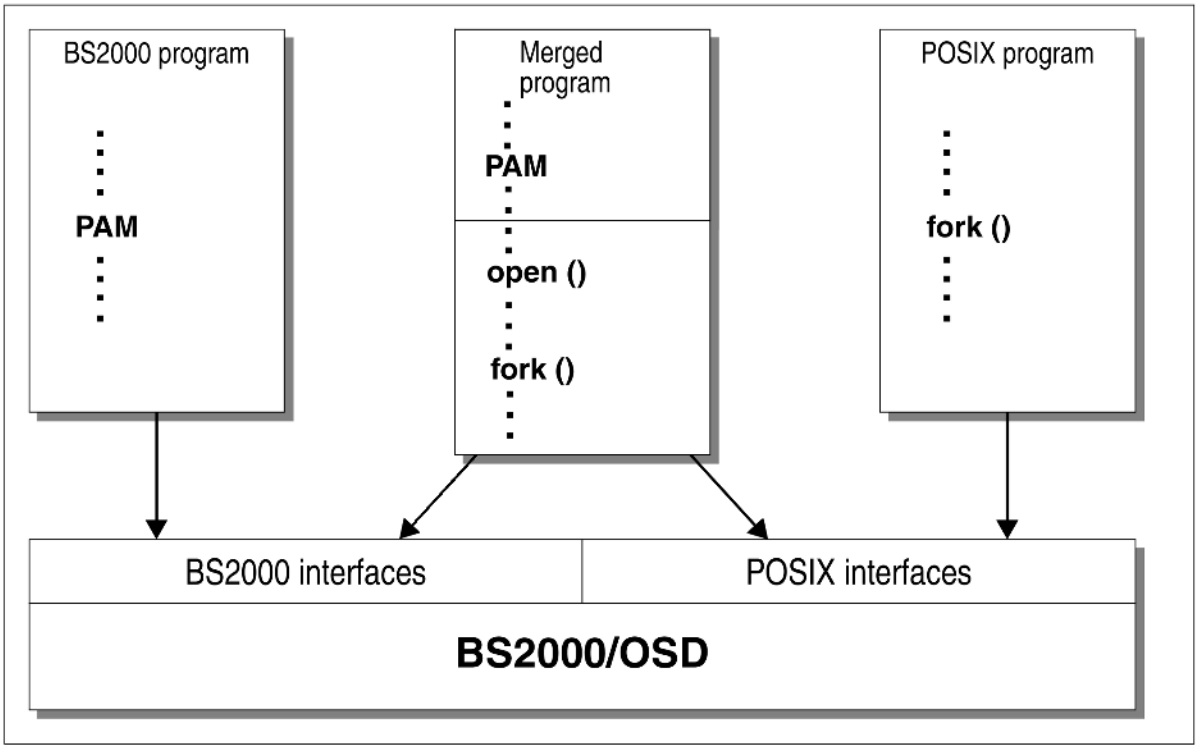

Figure 19: Pure and merged applications

POSIX program interfaces contain C library functions with BS2000 and POSIX functionality. Programs written on other platforms (UNIX, Windows, ...) in accordance with the XPG4 standard need only be recompiled in order to be executable in BS2000.

POSIX files can be accessed via pure or merged programs and can also be processed by some BS2000 software products, such as EDT and HSMS (see chapter “BS2000 software products in the POSIX environment”).

3.2.1 Restrictions for programs with merged functionality

Programs which use POSIX interfaces are treated as programs with pure BS2000 interfaces. However, restrictions apply whenever a process is created with a *fork* call and its BS2000 environment is not passed on with it.

A distinction must be made between the following calls of a merged program:

1. A logon process (dialog task) which was not created by a *fork* call

   The BS2000 and POSIX program interfaces can be merged as desired.

2. A process which was created from a logon process by a *fork* call

   The SYSFILE environment is not passed on, since BS2000 files opened by the parent process are not inherited by the child process.
   The SYSFILE environment is a customized system file for command input (SYSCMD), data input (SYSDTA), logging (SYSLST), and for message or data output (SYSOUT).

   A combination of BS2000 and POSIX program interfaces is permitted in the following situations:

   - Parallel output via POSIX mechanisms and WROUT
   - Input via RDATA is not possible
   - Checkpoint/restart is not possible
   - BKPT is not possible
   - *fork* is not possible, if DIV or FASTPAM areas exist

   Otherwise, BS2000 and POSIX program interfaces can be merged as desired.

3. A merged program which is started from the POSIX shell

   A merged program which is started from the POSIX shell has a SYSFILE environment other than the POSIX shell, since it was created by a *fork* call. The same situation applies as for number 2 above.

3.2.2 Restrictions for macros

Each process created by call for the *fork* function has a SYSFILE environment but no SYSCMD system file (with the exception of rlogin sessions). As a result any attempts to access the SYSCMD system file are rejected with a return code. In every other case, BS2000 commands can be entered via CMD macros. Note the following requirements:

- A CMD macro requires subsequent SDF initialization of the task generated by fork. This has a detrimental effect on performance.
- The following commands are not permitted in CMD macros:
  - All commands which are already prohibited in BS2000 (see the "Executive Macros" [30] manual); for example, the /HOLD-PROGRAM command is prohibited.
  - Macros from commands which access the SYSCMD system file, for example the AID %TRACE command

As in BS2000, the BS2000 commands /EXIT-JOB and /LOGOFF terminate the task and return you to the parent process.

3.2.3 Inheritance

In the case of a *fork* call, only POSIX resources are inherited. For this reason, POSIX files that the parent process opened are also available in the child processes. By contrast, BS2000 files which were opened in the parent process remain closed.

The class 6 memory of a program is inherited in its entirety. For class 5 memory, only those pages previously marked as inheritable are inherited.

Quasi inheritance of BS2000 resources can be incorporated in the program by means of opening resources (e.g. BS2000 files) as shareable in the parent process. The information on these resources can then be transmitted via a privately defined data structure to the child process, which is, after all, executing the same program as the parent process. The child process can then reconnect itself explicitly to these resources.

3.3 Sample session

This section contains an example showing how to work with the POSIX shell. Log on to BS2000, wait for the contents of your user ID to be output, and then start the POSIX shell. Firstly, create a *.profile* file in the POSIX shell, in which, in order to simplify your work, you define new alias variables and also a new prompt which outputs the current path. Following execution of the *.profile* file, the definitions made here become effective.
You then transfer a file from the BS2000 file system to the POSIX file system and process it there.

```
(1)  /set-logon-parameters user-id=user1,account=...
(2)  /show-file-attributes
     %     114 :1OSN:$USER1.ANHANG.V2
     %       3 :1OSN:$USER1.AVASQUER
     %      78 :1OSN:$USER1.BIB.EXAMPLES.SDF
     %       6 :1OSN:$USER1.DO.MSGCHECK
     %    5007 :1OSN:$USER1.FS.USER1
     %       3 :1OSN:$USER1.MSG.PROT
     %       3 :1OSN:$USER1.OUTPUT
     %       3 :1OSN:$USER1.PROG.C
     %       3 :1OSN:$USER1.SYS.SDF.LOGON.USERPROC
(3)  /start-posix-shell
     POSIX Basisshell 09.0A43 created Feb 20 2012
     POSIX Shell 08.0A43 created Aug 02 2011
     Copyright (C) Fujitsu Technology Solutions 2009
                   All Rights reserved
(4)  Last login: Sun Jul 22 19:42:55 2012 on term/001
(5)  $ edtu .profile
```

(1) Log on to BS2000 in the usual way.

(2) Wait for your user ID to be output by the BS2000 command /SHOW-FILE-ATTRIBUTES.

(3) Call the POSIX shell by means of the BS2000 command /START-POSIX-SHELL.

(4) You are accepted as a POSIX shell user.

(5) Create the *.profile* file with the POSIX editor *edtu*.Since the file does not exist, *edtu* creates a new file (see "Sample session").

```
   1.00 alias ll='ls -l'
   2.00 alias la='ls -al'
   3.00 PS1='$PWD> '
   4.00
   5.00
   6.00
   7.00
   8.00
   9.00
  10.00
  11.00
  12.00
  13.00
  14.00
  15.00
  16.00
  17.00
  18.00
  19.00
  20.00
  21.00
  22.00
                    POSIX edtu opened file: .profile (new)
return                                                    0000.00:00001(00)
```

```
(6)  $ . .profile
(7)  /home/user1> la
     total 84

     drwxr-xr-x   5 USER1    USROTHER    2048 Dec 22 14:03 .
     drwxr-xr-x  63 SYSROOT  POSSYS      2048 Dec 22 06:35 ..
     -rw-r--r--   1 USER1    USROTHER      48 Dec 22 14:02 .profile
     -rw-------   1 USER1    USROTHER    2576 Dec 22 14:06 .sh_history
     drwxr-xr-x   2 USER1    USROTHER    2048 Dec 15 17:18 c-source
     drwxr-xr-x   2 USER1    USROTHER    8192 Dec  5 13:47 lost+found
     -rw-r--r--   1 USER1    USROTHER      94 Dec 21 14:02 letter1
     drwxr-xr-x   2 USER1    USROTHER    2048 Dec 19 15:05 test
       ...
(8)  /home/user1> cd c-source
```

(6) After having created the *.profile* file with *edtu* and having terminated the editor by means of the *return* command, the *.profile* file shall be evaluated in the current shell. To do this, enter `. .profile`.

(7) The POSIX shell responds with the newly defined prompt, which outputs the current */home/user1* path.The list of all the files in the directory is displayed with the command defined by the alias *la*.

(8) Go to the *c-source* subdirectory where, for example, your C programs are stored.

```
(9)  /home/user1/c-source> bs2cp bs2:prog.c prog.c
     /home/user1/c-source> la
     total 60

     drwxr-xr-x  2  USER1      USER1GRP     2048   Jul 6 .
     drwxr-xr-x  2  USER1      USER1GRP     2048   Jul 6 ..
     -rw-r--r--  1  USER1      USER1GRP     2048   Jul 6 prog.c
(10) /home/user1/c-source> cat prog.c
     #include <stdio.h>
     main()
     {
       printf("hello world\n");
       return(0);
     }
(11) /home/user1/c-source> cc -o prog prog.c
(12) /home/user1/c-source> prog
     hello world

(13) /home/user1/c-source> exit
(14) ....

(15) /exit-job
```

(9) Copy the *prog.c* file in the BS2000 file system to the POSIX file system. The file is written to the current *c-source* directory.

(10) Output the contents of the *prog.c* file by entering *cat.*

(11) Compile the *prog.c* file with the C compiler. The result of the compilation run should be written to the *prog* file.

(12) Run the *prog* program. It outputs the string “hello world” on the screen.

(13) Terminate the POSIX shell by means of the *exit* command.

(14) Enter further BS2000 commands if desired.

(15) Log off from BS2000.

3.4 Program interface for large POSIX files

This section of the chapter describes how to create new programs to handle large files and how to modify existing programs in order to access large files.

3.4.1 Creating new programs

When creating new programs to access large POSIX files:

- Place the following Define statement in front of the first Include statement:

  ```
  #define _LARGEFILE64_SOURCE 1
  ```

- Specify the header *unistd.h* as the first Include statement.
  Then the program will have all interfaces and data types it needs available to it.
- Use the 64-bit functions to access large POSIX files, i.e. simply use *open64()*, *lseek64()*, ... instead of the normal functions *open()*, *lseek()*,... .
  The following list provides an overview of all 64-bit functions:

creat64()	fstat64()	lseek64()	stat64()
fgetpos64()	fstatvfs64()	lstat64()	statvf64()s
fopen64()	ftell64()	mmap64()	statvfs64()
freopen64()	ftruncate64()	open64()	truncate64()
fseek64()	getdents64()	readdir64()	
fsetpos64()	getrlimit64()	setrlimit64()	

- Use the 64-bit data types in the program instead of the 32-bit data types, e.g. *off64_t* (64-Bit) instead of *off_t* (32-Bit). These data types are defined in the Include file *sys/types.h*. This allows the 64-bit and 32-bit interfaces to be used in parallel in a program, e.g. *lseek()* and *lseek64()*. This facilitates migration of programs.

3.4.2 Adapting existing programs to large files

If you wish to adapt existing programs to access large files, you must follow the procedures that are described in the section "Creating new programs", i.e. add the appropriate Define statements and the Include file *unistd.h*, replace the 32-bit functions (*open()*, *lseek()*, ...) with the corresponding 64-Bit functions *open64()*, *lseek64()*, ... and if necessary use the 64-bit data types. This may mean internal data structures may have to be modified in an incompatible way.

To clarify the problems associated with accessing large files, a small program *prog32* is given below as an example. This program is subsequently converted so that it can access large files (Program example *prog64* in "Adapting existing programs to large files").

Program example prog32

prog32 opens a specified file and outputs a maximum of 32 Bytes of this file starting from a specified point. For reasons of legibility, the input parameters check has been omitted.

```
/*
** prog.c
**
** Parameter 1: Name of file to be read
** Parameter 2: Offset of data to be read in file
**
*/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/fcntl.h>
#define BUFFER_LENGTH   8192
#define READ_LENGTH     32
char buffer[BUFFER_LENGTH];
int
main(int argc, char *argv[]) {
int fd;
int len;
int i;
off_t filelen;
off_t offset_in_file;
   offset_in_file = atol(argv[2]);
   printf ("reading from file <%s> with offset %d and length %d\n", argv[1],
         offset_in_file, READ_LENGTH);
   /* open file */
   if ((fd = open (argv[1], O_RDONLY)) < 0) {
      printf ("open not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
   /* now get the length of the file */
   if ((filelen = lseek(fd, (off_t)0, SEEK_END)) == (off_t)(-1)) {
      printf ("lseek to end of file not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
   if (offset_in_file > filelen) {
      printf ("offset %d is greater than filelength %d, termination\n",
            offset_in_file, filelen);
```

```
      exit(EXIT_FAILURE);
   }
   /* now seek to the offset to be read */
   if (lseek(fd, offset_in_file - filelen, SEEK_CUR) == (off_t)(-1)) {
      printf ("lseek not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
   /* read the data */
   if ((len = read (fd, &buffer[0], READ_LENGTH)) <= 0 ) {
      printf ("read not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
   else {
      /* now print the data that were read in hexadecimal form */
      printf ("data of size %d (expected %d) were read\n", len, READ_LENGTH);
      for (i = 0; i < len; i++) {
         printf ("%02X ", buffer[i]);
      }
      printf ("\n");
   }
   if (close(fd) != 0) {
      printf ("close not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
}
```

Program example prog64

prog64 is derived from *prog32* and converted so that it can access large files.

```
/*
** prog.c
**
** 1. Parameter: Name of file to be read
** 2. Parameter: Offset of data to be read in file
**
*/
#define _LARGEFILE64_SOURCE   1
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/fcntl.h>
#define BUFFER_LENGTH        8192
#define READ_LENGTH          32
char buffer[BUFFER_LENGTH];
int
main(int argc, char *argv[]) {
int fd;
int len;
int i;
off64_t filelen;
off64_t offset_in_file;
```

```
   offset_in_file = atoll(argv[2]);
   printf ("reading from file <%s> with offset %lld and length %ld\n",
             argv[1], offset_in_file, READ_LENGTH);
   /* open file */
   if ((fd = open64 (argv[1], O_RDONLY)) < 0) {
      printf ("open not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
   /* now get the length of the file */
   if ((filelen = lseek64(fd, (off64_t)0, SEEK_END)) == (off64_t)(-1)) {
      printf ("lseek to end of file not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
   if (offset_in_file > filelen) {
      printf ("offset %lld is greater than filelength %lld, termination\n",
            offset_in_file, filelen);
      exit(EXIT_FAILURE);
   }
   /* now seek to the offset to be read */
   if (lseek64(fd, offset_in_file - filelen, SEEK_CUR) == (off64_t)(-1)) {
      printf ("lseek not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
   /* read the data */
   if ((len = read (fd, &buffer[0], READ_LENGTH)) <= 0 ) {
      printf ("read not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
   else {
      /* now print the data that were read in hexadecimal form */
      printf ("data of size %d (expected %d) were read\n", len, READ_LENGTH);
      for (i = 0; i < len; i++) {
         printf ("%02X ", buffer[i]);
      }
      printf ("\n");
   }
   if (close(fd) != 0) {
      printf ("close not successful, termination\n");
      perror("ERRNO SET");
      exit(EXIT_FAILURE);
   }
}
```

Applying prog32 and prog64

Both programs were applied to the following files in the */mnt33/dir1* directory:

```
$ ls -l /mnt33/dir1
total 10245088
-rw-r--r--   1 BACH     OS315    2621440000 Feb 27 18:26 bigfile1
-rw-r--r--   1 BACH     OS315    2621440000 Mar  6 15:06 bigfile2
-rw-r--r--   1 BACH     OS315         18 Mar 15 13:56 smallfile1
-rw-r--r--   1 BACH     OS315         26 Mar 15 13:57 smallfile2
drwxr-xr-x   2 BACH     OS315       2048 Mar 15 13:58 subdir1
drwxr-xr-x   2 BACH     OS315       2048 Mar 15 13:58 subdir2
```

Results of some program runs:

```
$ prog32 /mnt33/dir1/smallfile1  128
reading from file </mnt33/dir1/smallfile1> with offset 128 and length 32
offset 128 is greater than filelength 18, termination
$ prog32 /mnt33/dir1/bigfile1  128  28
reading from file </mnt33/dir1/bigfile1> with offset 128 and length 32
lseek to end of file not successful, termination
ERRNO SET: Value too large for defined data type
$ prog64 /mnt33/dir1/smallfile1  128
reading from file </mnt33/dir1/smallfile1> with offset 128 and length 32
offset 128 is greater than filelength 18, termination
```

```
$ prog64 /mnt33/dir1/smallfile1  10 28
reading from file </mnt33/dir1/smallfile1> with offset 10 and length 32
data of size 8 (expected 32) were read
91 A5 A2 A5 A2 A5 95 15
$ prog64 /mnt33/dir1/bigfile1  128  0
reading from file </mnt33/dir1/bigfile1> with offset 128 and length 32
data of size 32 (expected 32) were read
F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3 F3
F3 F3 F3 F3 F3 F3
$ prog64 /mnt33/dir1/bigfile1  2500000000
reading from file </mnt33/dir1/bigfile1> with offset 2500000000 and length 32
data of size 32 (expected 32) were read
F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1 F1
F1 F1 F1 F1 F1 F1
$ prog64 /mnt33/dir1/bigfile1  500000000000
reading from file </mnt33/dir1/bigfile1> with offset 5000000000 and length 32
offset 5000000000 is greater than filelength 2621440000, termination
```

4 BS2000 software products in the POSIX environment

This chapter is aimed at all POSIX users. It gives a brief overview of all software products in the POSIX environment. Sections “Binder Loader System” through “SORT” deal with BS2000 products which have been adapted to POSIX and can work with POSIX files. From Section “interNet Services”, products are introduced which have been ported to BS2000 with POSIX. The number of ported products is continuously increasing.

4.1 Binder Loader System

To link and load executable programs, POSIX uses the BS2000 Binder Loader System (BLS), the most important components of which are the linkage editor BINDER and the DBL (dynamic binder loader). DBL is part of BLSSERV. POSIX always calls BLS when a POSIX user calls a program or requests an executable program as a result when compiling a source program.

An external user interface such as BINDER in BS2000 is not available in the POSIX environment. To link and load programs in the POSIX environment, the conventions established by the compiler and the standard POSIX program calls apply.

If user programs which contain unresolved externals are loaded into the POSIX environment, the binder loader system generates messages which are displayed in the POSIX shell.

See the manual "BLSSERV" [11] for further information on the binder loader system of BS2000.

The commands for linking in the POSIX shell are *cc*, *c89, cobol* and *CC*. These commands are described in detail in the manual "C/C++ POSIX Commands" [4].

4.2 C and C++ compilers

The BS2000 compilers C/C++ can be called and controlled with options from both the BS2000 environment (with SDF) and from the POSIX environment (POSIX shell).

Compiler control via the SDF interface

All compiler I/O operations are possible both in the BS2000 file system (DMS/PLAM) and in the POSIX file system:

- input of source programs
- input of header files
- output of LLM
- output of recompilable source programs
- output of compiler listings
- output of message lists
- output of CIF information

Any desired combinations are possible, i.e. input and output of both BS2000 and of POSIX files in the same compilation run.

The SDF interface of the compiler is described in the User Guide "C/C++ Compiler" [5].

Compiler control via the POSIX shell interface

The following POSIX commands are available for controlling the C/C++ compiler from the POSIX environment:

cc, c89	Compiles C sources
CC	Compiles C++ sources
cclistgen	Calls the global list generator in C/C++

These commands are described in detail in the manual "C/C++ POSIX Commands" [4].

All compiler input or output occurs exclusively in the POSIX file system.

A linkage stage in which the compiled objects can be linked to form an executable unit is also integrated into the *cc*, *c89* and *CC* call commands.

The options and operands described in the above call commands largely cover the services and functions which are available with compiler control via the SDF interface. The syntax of the POSIX commands is based on the XPG4 standard definition and on the familiar shell commands in UNIX systems.

The compiler supports 64-bit arithmetic data types (long long). This is a prerequisite for programming access to large files in POSIX.

Notes on CRTE

Since CRTE (Common RunTime Environment) provides the runtime environment for C and C++ programs, it is required for using the C/C++ compiler.

The CRTE version required depends on the version of the compiler which is used. Further information can be found in the C/C++ compiler release notice.

Character sets for input/output files

Source programs and include files are available in EBCDIC or ASCII code, which means that it is also possible to compile source programs from file systems located on a remote UNIX system. All the files in a file system (local POSIX file system or linked remote file system) must be available in the same character set. i.e. all files in the POSIX file system must be available in EBCDIC code and all files in the remote UNIX file system must be available in ASCII code.

The environment variable IO_CONVERSION must be set to “YES” (see "Copying and converting files" for more information).

The output character set of text files (lists etc.) depends on the character set of the target file system.
In general, EBCDIC procedure code is generated.

4.3 COBOL85 / COBOL2000 compilers

The BS2000 compilers COBOL85 as of V2.3 and COBOL2000 can be called both from the BS2000 environment (using SDF) and the POSIX environment (shell), and controlled with options.

The following user manuals describe how to use the compilers:

- "COBOL85" [13]
- "COBOL2000" [12]

Controlling compilers using the SDF interface

All compiler input and output possible in both the BS2000 file system (DMS/PLAM) and the POSIX file system:

- Input of source programs
- Input of copy elements
- Output of LLMs
- Output of compiler lists
- Output of message lists

Combinations of these are also possible, i.e. inputting and outputting both BS2000 and POSIX files in a single compilation run.

Controlling compilers using the POSIX shell interface

COBOL compilers can be controlled from a POSIX environment using the POSIX command *cobol*.

All compiler input and output is carried out exclusively in the POSIX file system.

A link phase is also integrated into the *cobol* command. This enables the compiled objects to be linked into an executable unit.

The options and operands of the call commands above mainly cover the services and functions which are available with the compiler control over the SDF interface. The syntax of the POSIX command is oriented towards the shell commands used in UNIX systems.

Notes on CRTE

CRTE (Common RunTime Environment) provides the runtime environment for COBOL programs. CRTE is also a prerequisite for using the COBOL compiler.

Further information can be found in the COBOL compiler release notice.

Character sets for input and output files

The source programs and Include files can be available in EBCDIC code or ASCII code. This makes it possible to compile source programs from file systems which are located on a remote UNIX system. All files of a file system (local POSIX file system or mounted remote file system) must be available in the same character set, i.e. all files in the POSIX file system must be available in EBCDIC code and all files in the remote UNIX file system must be available in ASCII code.

The environment variable *IO_CONVERSION* must be assigned the value “YES” (see also "Copying and converting files").

The output character set of the text files (lists etc.) matches the character set of the target file system.
EBCDIC execution code is generally generated.

Executing COBOL applications

To access files in the POSIX file system using COBOL-IO, the COBOL program needs to be compiled using the option COMOPT ENABLE-UFS=YES, which is implicitly set by the POSIX command *cobol*. Programs compiled in this way can then process both BS2000 and POSIX files.

The issues described in the COBOL User Manual for COBOL programs executed under a POSIX shell also apply i. e. the following language elements are not supported:

- CALL identifier
- ENTRY
- READ PREVIOUS
- File processing with CODESET STANDARD.

Further differences in the functionality arise from differences in the system during file processing, especially tape processing with labels, concurrent processing of files and outputting checkpoints, and from assigning files. In exceptional cases, the PROSOS SIS code is made available over the file status data fields instead of the BS2000 DMS code.

No job variables such as job switches and user switches are available to control the application externally. Extended language elements are available to access the command line.

4.4 BS2000 Environment for Java (JENV)

With the BS2000 Environment for Java (JENV), Java programs created on any platform can be executed on BS2000 systems. This reflects the Java concept “write once, run everywhere”. In the same way, Java applications developed for BS2000 are also executable on other platforms.

The range of functions corresponds to the relevant basic Java2 Standard Edition SDK of JavaSoft. This is ensured by Sun validation and thus acquires the right to use the “Java Compatible Logo” for BS2000.

JENV is normally used inside the POSIX environment (POSIX shell), but it can also be controlled from the BS2000 environment using procedures.

When installing JENV with POSIX, it must be noted that a relatively large quantity of storage space is required in the POSIX file system. If possible, this should be taken into account when the POSIX file system is designed. For information on the required sizes, see the release notice.

JENV is a delivery component of the BS2000 basic configuration and is executable on all hardware platforms. Special high-performance variants exist for non-/390 platforms. These are then only executable on the appropriate hardware.

For further information, see the appropriate manual and the release notice.

4.5 EDT

You can create and process POSIX files with EDT. Therfore the POSIX subsystem must be started.

For operation in Unicode mode, EDT is called using START-EDTU in BS2000 or using the *edtu* command in the POSIX shell.

For operation in compatibility mode, EDT is called using START-EDT in BS2000 or using the *edt* command in the POSIX shell. The edt command uses the EDT version 16.6.

For more information, see the manuals "POSIX Commands" [1], "EDT Unicode Mode Statements" [15] and "EDT Statements" [14].

4.6 File transfer openFT for BS2000

The file transfer product openFT for BS2000 supports the transfer of POSIX files. Optional components are openFT-FTAM, to implement the FTAM functionality, openFT-AC, for the access control provided by the FTAC functionality, and if applicable openFTP, for the support of the FTP functionality.

The POSIX subsystem must be started, so that the POSIX functions of openFT can be used. POSPRRTS must also be started.

Specifying a POSIX file

A POSIX file must be specified with a special syntax in openFT commands. File names which begin with / or ./ are interpreted as fully or partially qualified POSIX file names. Files which do not begin with one of these characters are regarded as BS2000 files.

For more information, see the "openFT (UNIX/Windows)" [19]” manual.

4.7 HSMS

With HSMS, you can save, archive and restore files located on remote, networked computers. The file system to be processed may be either the local BS2000-UFS or a remote UNIX file system that the BS2000 system administrator must mount on the HSMS directory of the local BS2000-UFS. The HSMS directory is on the level immediately beneath the root directory “/”, which exists in every UNIX file system and acts as the point of entry.

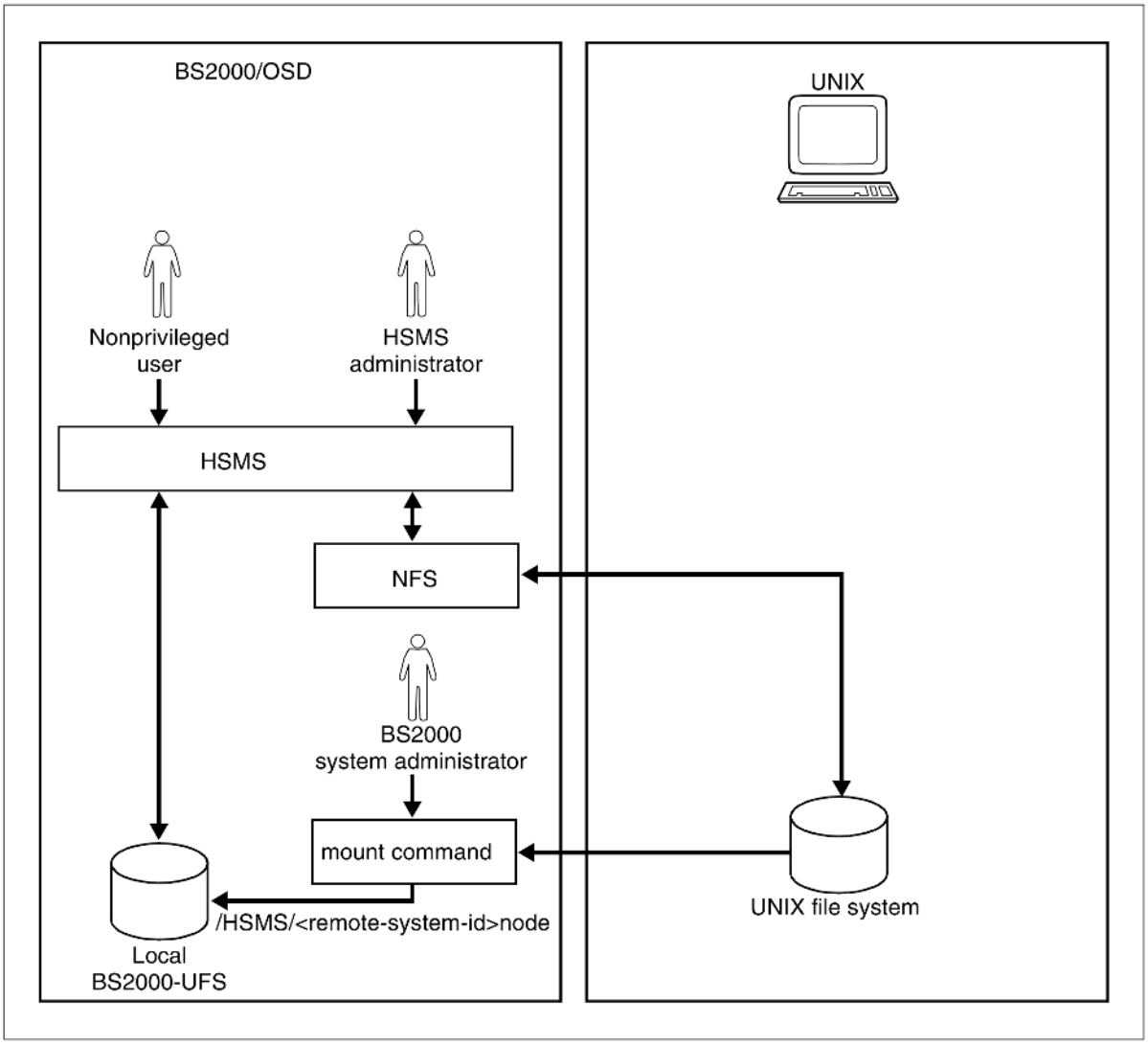

Figure 20: Remote UNIX file systems in relation to the /HSMS directory

The following HSMS statements are available to HSMS administrators and - with certain restrictions - to nonprivileged users for processing files belonging to a UNIX file system:

- ARCHIVE-NODE-FILES
  Archive node files
- BACKUP-NODE-FILES
  Save node files
- COPY-NODE-SAVE-FILE
  Copy node save files
- MODIFY-NODE-PARAMETERS
  Modify parameters for a node-S0

- RESTORE-NODE-FILES
  Restore node files
- SELECT-NODE-FILES
  Select filenames from a node lists
- SHOW-NODE-PARAMETERS
  Show parameters for node-S0

See the manual "HSMS" [21] for more information.

4.8 NFS

Before you can work with the file systems of a remote computer, the NFS (Network File System) software product must be installed on both the local and remote computers. The file system to be mounted must be exported with the NFS command *share* on the remote computer (NFS server) and mounted with the NFS command *mount* on the local computer (NFS client). The remote file system can then be accessed from the local computer. Conversely, the local computer can of course also be the NFS server and the remote computer the NFS client.

See the manual "NFS" [8] for more information.

4.9 SECOS

POSIX uses the SECOS component SRPM for the administration and access control of POSIX users.

If SECOS is not installed in your system, the relevant part of SRPM for POSIX is contained in the BS2000 basic configuration.

For more information on the BS2000 administration of POSIX users, please refer to chapter “Administering POSIX users”.
Access control for users who want to connect to a BS2000 computer from a UNIX system by means of the *rlogin* command is described in section “Access from a character terminal”.

If SECOS is being used, the following options are also available for POSIX:

- Use of the POSIX-ADMINISTRATION privilege for selected user IDs (SRPM).
- Logging and analysis of security-relevant events which affect POSIX with SAT.
  In addition to the general options for monitoring user IDs, DMS file objects, and events, the following events are defined specifically for POSIX:
  - JFK event: create POSIX task
  - UPA event: /MODIFY-POSIX-USER-ATTRIBUTES command
  - UPD event: / MODIFY-POSIX-USER-DEFAULTS command

  The security-relevant events of privilege administration - for example, assign the POSIX-ADMINISTRATION privilege - are always logged with SAT.
- Logging of approx. 50 security-relevant POSIX events, grouped according to:
  - File access (POSIX-FILE-and-Directory)
  - Process attributes (POSIX-Process)
  - Fork (POSIX-CHILD-Process)
  - Semaphore, shared memory (POSIX-SYSTEM-Resources)
- Individual system access classes for global POSIX services (*rlogin*, *rcp*, ...).
- POSIX batch processes are subject to a check by SECOS. Changing the user ID can be permitted or forbidden by SECOS (for each ID).

See the SECOS manuals [9] and [10] for more information.

4.10 SOCKETS/XTI (POSIX SOCKETS)

The delivery unit POSIX also incorporates the SOCKETS/XTI interfaces for programming network functions which facilitate access to the Internet via TCP/IP and UDP/IP, thus giving access to the open network world.

The SOCKETS/XTI interfaces are supplied with POSIX-SOCKETS and defined in a separate library. If this library is linked into a POSIX application, the SOCKETS/XTI interfaces generate the connection to the network via the POSIX subsystem and the BCAM transport system.

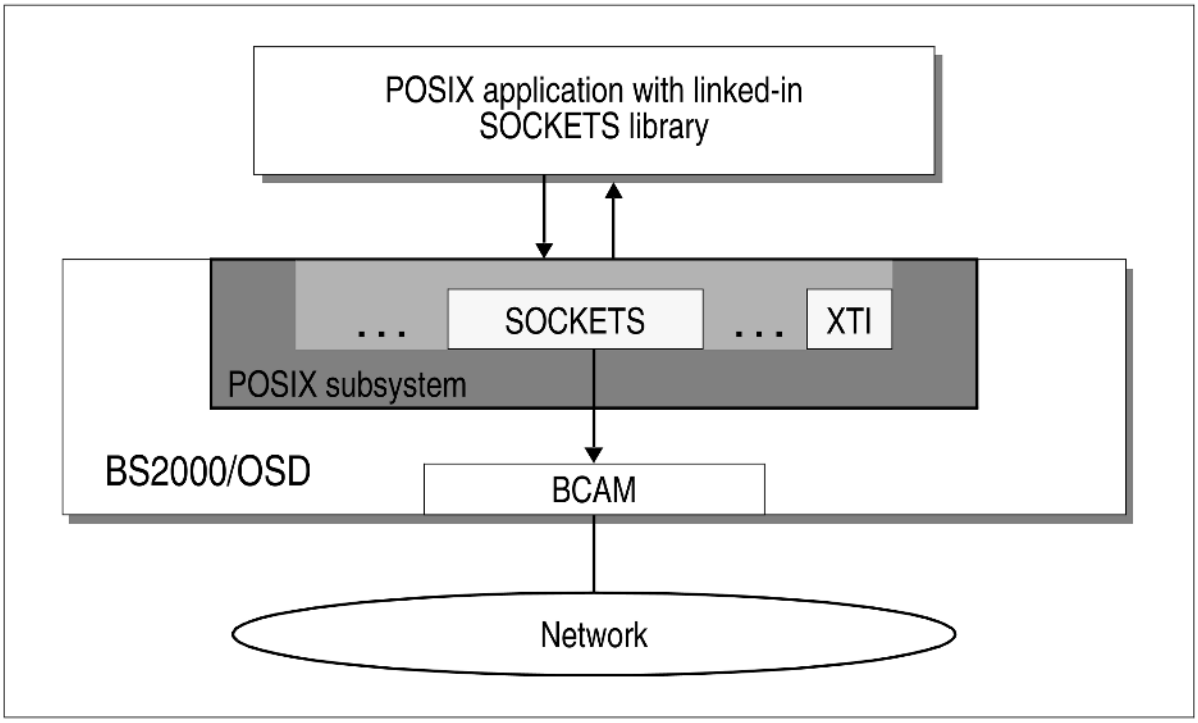

Figure 21: SOCKETS/XTI in BS2000 and in POSIX

See the manual "POSIX SOCKETS/XTI" [3] for more information.

4.11 SPOOL

You can print POSIX files with SPOOL, using BS2000 or POSIX commands.

Example

Print the POSIX file */home/psxroot/usr1* from BS2000:

```
/PRINT-DOCUMENT FROM-FILE='/home/psxroot/usr1',...
```

Print the same POSIX file from the POSIX shell:

```
/START-POSIX-SHELL
.
.
.
/home/psxroot> lp usr1
/home/psxroot> exit
```

You can print POSIX files on the connected BS2000 printer by means of the POSIX *lp* command. The *lp* command uses BS2000 SPOOL for printing. No ID is assigned for a print job.
Print jobs can be only managed via BS2000 SPOOL.

See the SPOOL manuals [32] and [33] for more information.

4.12 TLI (POSIX-NSL)

In addition to the SOCKETS interfaces (see "SOCKETS/XTI (POSIX SOCKETS)"), the TLI network interfaces are also optionally available. These likewise facilitate access to Internet on the basis of TCP/IP and UDP/IP. The TLI interfaces are included in the POSIX-NSL component; POSIX-NSL is supplied with POSIX.

Like the SOCKETS interfaces, the TLI interfaces consist of a number of library functions which provide the connection to the network via the BCAM transport system.

4.13 AID

AID (Advanced Interactive Debugger) provides you with the means to debug pure POSIX or BS2000 programs as well as mixed programs. Mixed programs use both BS2000 and POSIX program interfaces. Debugging is made possible by extensions to the AID commands %AID and %STOP, as well as by the POSIX *debug* command.

The AID command %AID has been extended to include two new operands, namely FORK={OFF | NEXT | ALL} and EXEC={OFF | ON}. These cause the program to be interrupted immediately after a *fork()* or *exec()* call, after which control switches to debugging mode and you can enter AID commands for debugging your program as usual.

The AID command %STOP has likewise been extended by two new operands, namely T=tsn (task sequence number) and PID=pid (process identification). These enable you to interrupt a task created by *fork()*.

AID reports with the process number (pid) of the interrupted task, and you can control further execution of this task by means of AID commands.

The POSIX *debug* command enables you to load a program in the POSIX shell with LSD or interrupt a running process and switch to debugging mode:

`debug progname`

> The program is loaded in a fork task with LSD and switched to debugging mode, i.e. you can enter AID commands. The 'debug progname' command in the POSIX shell thus corresponds to the following BS2000 command in the BS2000 environment:
> `LOAD-PROGRAM progname, ... TEST-OPTIONS=*YES`

`debug -p pid`

> The process with the specified pid is taken over by AID and interrupted. 'debug -p pid' in the POSIX shell corresponds to the AID command %STOP PID=pid mentioned above and which you can enter in BS2000 system mode or in debugging mode of a task.

Dumps of mixed or POSIX programs are stored as previously in BS2000, where they can also be processed. If AID is to dynamically load the LSD with the AID %SYMLIB command to dump a POSIX program, you should bear in mind that %SYMLIB cannot access POSIX files. The file concerned must first be copied to a PLAM library in BS2000 as an L element using the POSIX *bs2cp* command, after which it can be assigned with %SYMLIB.

A detailed description of debugging POSIX and mixed programs with AID can be found in the manuals concerning AID [35] and [36].

4.14 SORT

POSIX files can be assigned as input files (INPUT-FILES operand) or as output files (OUTPUT-FILES operand) in the SORT control statement ASSIGN-FILES or in the SORT-FILE command.

To distinguish them from BS2000 file names, POSIX file names must be specified in single quotes in the INPUT-FILES and OUTPUT-FILE operands.

Work files and auxiliary files must not be POSIX files.

The data in POSIX files is in text format, which cannot be processed directly by SORT. Before it is processed by the sort routine, this data is converted by SORT into variable-length records, each prefixed by a record length field.

After the sort process, SORT converts the sorted output file back into text format if it is to be stored in the POSIX file system.

The internal use of variable-length records causes the position of the user data in the record to be displaced by the record length field, but this does not normally have any consequence for POSIX file users. With records from POSIX files, SORT calculates the field positions by default relative to the beginning of the user data.

However, if the user wants to access the internal record length field, e.g. in order to sort the records according to their length, the IGNORE-LENGTH-FIELD operand is available in the SET-SORT-OPTIONS statement and the SORT-FILE command.

Specifying IGNORE-LENGTH-FIELD=*NO causes the positions within the record to be calculated from the start of the record, both with variable-length records in BS2000 files and with records in POSIX files. The user data thus begins at position 5 in the record.

The encoding of the end-of-record identifier is determined by the CODE operand in the ASSIGN-FILES statement and the SORT-FILE command. If CODE=EBCDIC is specified, the end-of-record identifier is encoded as X'0A' and if CODE=ASCII is specified, as X'15'.

If POSIX files are used as output files, you must make sure that the output records do not contain characters which might be interpreted as end-of-record identifiers. More specifically:

- No constant fields containing end-of-record identifiers may be specified in the SORT-RECORDS statement or the SORT-FILE command.
- The records of a BS2000 input file must not contain end-of-record identifiers if the output file is to be a POSIX file.
- The sort type “tag sort” must not be used because it cannot be guaranteed that the address fields do not contain characters which could be interpreted as end-of-record identifiers.

> ¡ Using the *sort* command, which can be called in a POSIX shell, is not the same as calling the product SORT.

More detailed information can be found in the "SORT" [34] manual.

4.15 interNet Services

The product interNet Services provides the following TCP/IP services:

- File Transfer Protocol (FTP)
- Terminal Service (TELNET)
- Domain Name Service (DNS) Resolver (DNSD)r
- Domain Name Service (DNS) Server (NAMED)
- Network Time Protocol (NTP)
- Secure Shell Server (SSHD) and Secure Shell Clients
- Mail Server (Postfix) and IMAP/POP3 Server

With interNet Services as of V2.5A, an FTP client and a telnet client are available. These can be called using the POSIX command *ftp* or *telnet*.

With interNet Services as of V3.0B, a number of tools are available in POSIX via OpenSSH which serve as replacements for the unsecure r-Utilities (*rlogin*, *rsh*, *rcp*).

With interNet Services as of V3.1A, a mail server based on Postfix porting basis is available in POSIX which permits a mail client access to the mailboxes.

The FTP component is described in detail below.

With FTP, you can access POSIX directories from remote systems (UNIX systems, Windows, BS2000). A prerequisite for this is that the FTP server task was started with the START-FTP-DEMON command on the BS2000 system.

You can change to the POSIX-UFS in an FTP session by specifying the %POSIX path with *cd* or *lcd*.

Example

The following example shows an extract from an FTP session:

```
(1)  FTP> open BS2SERVER
     Connected to BS2SERVER, port 21.
     220 BS2SERVER FTP server (Version ... ) ready.
(2)  Name (BS2SERVER:USR): user1
     331 Password required for user1
     Password (BS2SERVER:red):

     332 Account required.
     Account: m0815xyz

     230 User USER1 logged in.
(3)  Ftp> cd %POSIX
     250 "/home/user1" is current directory now
     Ftp> ...
(4)  Ftp> bye
     221 Goodbye.
```

(1) With the *ftp* command, enter the BS2000 server on which the POSIX file system is installed.

(2) Enter the user ID, password and account of your BS2000 ID on this server.

(3) Specifying %POSIX with *cd* causes you to be switched from the BS2000 file system to the POSIX file system. You are then in the HOME directory of the POSIX ID assigned to your BS2000 ID.

(4) You use the FTP command to leave the POSIX file system, logoff from BS2000 and close the FTP session.

Further information can be found in the manuals "interNet Services" [40] and [41].

4.16 APACHE webservers on BS2000

With the webserver product APACHE for BS2000, the most widely used APACHE webserver in the world is also available for BS2000.

This free product offers the full functionality of the original APACHE webserver, including support of the https protocol, and has been extended with a number of additional, integrated components designed for web programming. These include the script interpreters PHP and Perl and the document management system WebDAV. In addition, Perl and PHP interpreters are offered as standalone programs.

APACHE (BS2000) and interNetSecurity (BS2000) are executed in the POSIX subsystem. PHP can be used to access BS2000 SAM and ISAM files and SESAM/SQL and ORACLE databases, and to execute BS2000 commands.

4.17 NET-SNMP and SNMP-AGENTS

The delivery unit NET-SNMP of the BS2000 operating system and the product SNMP-AGENTS provide the basic functionality for linking BS2000 systems into SNMP-based management environments. NET-SNMP and SNMP-AGENTS provide network, system and application management capability via SNMP.

Further information on SNMP management is provided in the manual "SNMP Management (NET-SNMP) for BS2000" [42].

5 Installing POSIX

This chapter is intended for BS2000 and POSIX system administrators who want to install POSIX and additional POSIX program packages. It contains information on

- the POSIX scope of delivery
- the POSIX installation concept
- the steps required for installing POSIX as an initial installation and as an upgrade installation
- the POSIX installation program (interactive mode and batch mode)
- the POSIX log files
- the POSIX information file (SYSSSI)

5.1 Scope of delivery

The basic POSIX functions and commands are part of the BS2000 basic configuration and are supplied in a separate technical “POSIX” selectable unit. This selectable unit comprises the following product components:

- POSIX-BC (POSIX subsystem and basic shell)
- POSIX-SH (extended shell)
- POSPRRTS (runtime system for the privileged part of POSIX)
- POSIX-SOCKETS (SOCKETS/XTI network interfaces)
- POSIX-NSL (TLI, RPC and XDR functions)
- POSIX-ADDON-LIB (interfaces which do not belong to the XPG4 standard)

POSIX provides the complete command set in accordance with the XPG4 standard.

The programming interfaces for POSIX are released and installed as library functions for the C/C++ programming language within the framework of the software product CRTE.

Please refer to the Release Notice (SYSFGM) for POSIX-BC for the file names of the separate delivery components.

5.2 POSIX installation concept

The installation of POSIX and POSIX products involves two steps:

1. Installation in BS2000 using the IMON/SOLIS procedure.

   The SOLIS delivery is installed in BS2000 using IMON. This updates, among other things, the subsystem catalog and the Software Configuration Inventory (SCI). This manual does not go into any further detail on installation under BS2000.

   The programs are not yet executable when they have been installed in BS2000. They must be installed in the POSIX file system with the following second step.

2. Installation in the POSIX file system.

   A separate POSIX installation program is available for installation in POSIX. This is started with the command /START-POSIX-INSTALLATION. The POSIX installation program can be called in interactive or batch mode (i. e. controlled using a parameter file).

   When installing in POSIX, the SCI can optionally be evaluated (see the section “The installation program in conjunction with IMON”).

This section of the chapter goes into further detail on the following areas of POSIX installation:

- the most important features of the POSIX installation program
- the format of the program packages
- installing with IMON support
- multimode installation, i.e simultaneous installation for several platforms (/390, X86)
- installing without IMON support
- installing private program packages

5.2.1 Features of the POSIX installation program

With the POSIX installation program, you can carry out an initial installation or an upgrade installation of POSIX, manage POSIX file systems and add or delete software program packages.

The POSIX installation program is started using /START-POSIX-INSTALLATION and can be called in interactive or batch mode:

- In interactive mode, you enter the data required in screen forms. Incorrect entries and inconsistencies are reported in interactive mode, so that you can immediately correct your entries. For further information, see the section “POSIX installation program in interactive mode”.
- In batch mode, you record the data required for installation in parameter files in a precisely defined layout. Invalid parameter files will cause the POSIX installation program to abort. For further information, see the section “POSIX installation program in batch mode”.

The actions of the installation program are logged, see the section “Logging the installation”.

For detailed information on which steps are needed for an initial or an upgrade installation, see the section “Initial installation of POSIX” or the section “Upgrade installation of POSIX”.

5.2.2 Format of program packages

Program packages for installation under POSIX, for the software supplied by Fujitsu, are delivered as BS2000 PLAM libraries with the default name

```
<prefix>LIB.<product>.<version>.<package>
```

and read in under a freely selectable archive ID.

Program packages can be installed optionally from the Software Configuration Inventory (i.e. following the official delivery procedure with IMON support, see the section "The installation program in conjunction with IMON") or from any archive ID, see the section "Installing a product without IMON support". This also applies to private program packages, see the section "Preparing private program packages for installation", e.g. private POSIX applications, as these can also be registered in the Software Configuration Inventory. Multiple installation allows more than one version of a product or correction status to be installed on POSIX.

Prefixes for different platforms

PLAM libraries with different prefixes are supplied for different platforms.

- Prefix SIN or SYS(*) for the /390 platform
- Prefix SKU or SKM(*) for the X86 platform

(*) These prefixes only apply to SOCKETS and NSL.

Installation scripts

PLAM libraries contain installation scripts which usually carry out the following actions, during installation with the POSIX installation program:

- Creating directories in POSIX file systems
- Copying text files (parameters, scripts, ...) into POSIX file systems
- Creating links to executable objects (LLMs) in the PLAM library
- Copying executable objects (LLMs) into POSIX file systems
- Executing ksh scripts for more complex processing

The products supplied by Fujitsu contain appropriate installation scripts.

If you wish to install third-party products or your own program packages in POSIX, you must make further adjustments and if necessary create your own installation scripts, see the section "Preparing private program packages for installation".

5.2.3 The installation program in conjunction with IMON

There are three different kinds of use if you are installing POSIX components with IMON support (see POSIX installation program, IMON support = Y):

1. Installation of the correction state of the product preset by IMON

   In this case, the POSIX administrator does not specify the version or the correction state. Either the highest correction state of the product is installed on POSIX or, if a /SELECT-PRODUCT-VERSION command was entered before the POSIX installation program startup, the correction state selected by this command is installed on POSIX.

2. Installation with selectable product correction state

   In addition to the product name, the POSIX administrator specifies the product version in Vmm.n format (m,n: digits) as well as the correction state in aso format (a: letter; s,o: digits) in the notation required by IMON.

3. Installation with selectable version and the correction state preset by IMON

   In addition to the product name, the POSIX administrator specifies the product version in Vmm.n format (m,n: digits). Either the highest correction state of the specified version is installed on POSIX or, if a /SELECT-PRODUCT-VERSION command was entered before POSIX installation tool startup, the correction state selected by this command is installed on POSIX.

5.2.4 Multimode installation

Multimode installation means that a product is installed in such a way that it can run on different platforms (/390 and X86) or so that programs can be generated for different platforms (e.g. for POSIX-SOCKETS). Multimode installation is only possible with IMON support.

There are two variants of multimode installation:

Installation variant A

Only **one** installation script is executed, irrespective of which platforms the product supports. If several platforms are supported, all installation scripts will be identical. The following conventions apply to this variant:

IMON Target	Logical Id	Name of installation script
any	SINLIB	INSTALL.<productname>.<productversion>

This installation variant is suitable for all products whose objects are platform-independent in POSIX, e.g. NFS and CRTE.
Note that links in POSIX to executable objects (LLMs) in PLAM libraries are platform-independent, but an LLM which has been copied into a POSIX file system is not.

Installation variant B

An installation script is executed for every platform supported by the product. The installation scripts may, for instance, differ from each other in that the LLMs are installed under different paths in POSIX. Examples include: POSIX-NSL, POSIX-SOCKETS, POSIX-SH.

For installation variant B, platform-specific installation scripts must be available in every PLAM library. The following conventions apply to this variant:

IMON Target	Logical Id	Name of installation script
S	SINLIB	INSTALL.<productname>.<productversion>.390
K	SINLIB	INSTALL.<productname>.<productversion>.X86

For reasons of compatibility, the following exception applies to the products POSIX-NSL and POSIX-SOCKETS:

IMON Target	Logical Id
S, K	SYSLIB

Interaction with the POSIX installation program

When installing with the POSIX installation program, you only need to specify the product name, as for variant 1 in the section “The installation program in conjunction with IMON”. The IMON target determines which installation variant is to be executed:

- If target A (ANY) is set, installation variant A is always executed, even if other targets are present. Links to executable objects (LLMs) in PLAM libraries are set up in such a way that the LLMs defined for the current platform or the LLMs of target A are always started in the current execution environment. The product can be used in any environment.
- If target A (ANY) is not set, the following applies:
  - Installation variant A is executed if only one target is set or if no platform-specific installation scripts exist.
  - Installation variant B is executed if more than one target is set and if platform-specific installation scripts do exist. The PLAM library of the first set target in the sequence above determines the existence of platform-specific installation scripts.

Irrespective of the installation variant which has been executed, the product can only be used in environments which were defined in IMON. In other environments, the error number ENOEXEC is returned by POSIX if a program is called over a link to executable objects (LLMs) in PLAM libraries.

5.2.5 Installing a product without IMON support

If you wish to install POSIX products without IMON support (see POSIX installation program, IMON support = N), the POSIX installation program will request the product name, the product version and the BS2000 ID. It uses these to form the name of the PLAM library from which the product components are installed in POSIX.

Multimode installation is not possible for product installation without IMON support, i.e. the product is only ever installed for a single platform.

5.2.6 Preparing private program packages for installation

Private program packages and third-party program packages must first be adapted to the form described below before they can be installed with the POSIX installation program:

- The product components must be available in a PLAM library.
- The PLAM library must have the product-specific name
  `<prefix>LIB.<product>.<version>.[<package>]`
- Executable programs must be available as L elements of the library.
- Header files, shell scripts and other components such as text files must be available as S elements of the library.
- The PLAM library must contain the following product-specific installation and deinstallation scripts as S elements:
  - INSTALL.<product>.<version>.[<package>]
  - DELETE.<product>.<version>.[<package>]

  These scripts describe the location of each product component in the POSIX file system. They also supply additional information relating to the storage area. The next section describes how the structure of these scripts.

Format of installation and deinstallation scripts

The individual lines of the installation and deinstallation scripts must have the following format:

```
element:selector:path:link name:access mode:user-number:group-number
```

The column width is variable. The colon delimiter (:) must be specified even if no value is specified. Comment lines start with “#”.

Entries for `selector` and `access mode` are explained in more detail below.

- The `selector` marks the installation subfunction (in alphabetical order):
  - `b` Create a binary file (PLAM element type X) under the specified name
  - `d` Create the directory indicated by the path name
  - `f` Create the command under the specified path name (link to element in PLAM library)
  - `i` Specify the installation path. The i entry must be the **first** statement line
  - `l` Create a hard link for the specified link name
  - `m` Create the command under the specified path name (as an LLM in UFS)
  - `o` Entry for files to be removed
  - `p` Create a procedure (`element`) under the specified path name
  - `r` Execute the script (procedure) with the specified path name
  - `s` Create a symbolic link for the specified link name
  - `u` Coded T files for *iconv*
  - `v` Entry for directories to be removed
- The `access mode` represents the access permission for owner, group and others (octal).

Environment variables in the installation and deinstallation scripts

You can use the following additional environment variables in installation scripts:

USER BS2000 user ID under which installation was started.

IPATH Installation path in the POSIX file system.
If IPATH is an empty string, the installation path is equal to “/”. If no installation path was defined with :i: (see below), $IPATH will not be interpreted, i.e. the string “$IPATH“ will then be part of a path name or link name.

IUID BS2000 installation ID
Either the *installation userid* from the dialog mask *BS2000 POSIX package installation* or the BS2000 user ID from the complete file name which is returned by IMON for the Logical-ID%SINLIB of the product to be installed. The leading dollar sign is a component of the string.

Installation path in the installation and deinstallation scripts

Using the code “i”, you can specify the installation path for the components of a product in the POSIX file system. The definition of an installation path must be the first statement line in the installation script, otherwise it has no effect.

An entry of this kind has the following syntax:

```
:i:installpath:access mode:user-number:group-number
```

This means:

`installpath`	Complete path name of a directory in the POSIX file system.
`access mode`	Access permissions assigned to `installpath`.
`user-number`	POSIX user ID of the owner of `installpath`
`group-number`	POSIX group ID of the owner of `installpath`

It is possible to install a product several times in the POSIX file system using this mechanism.

Specifying an installation path in the installation script results in the following:

- When installing in interactive mode, this installation path is displayed in the installation mask of the POSIX installation program. You can change the installation path there as you wish. The value which appears last in the dialog mask is effective.
- When installing in batch mode, this installation path applies if the parameter file does not contain an installation path. If an installation path was specified in the parameter file in the appropriate statement line, the installation path is taken from the parameter file, see the section “Administer POSIX filesystems”.
- The variable $IPATH in the installation script is replaced by the current value of the installation path. This applies to every specification of a path name or link name with the prefix $IPATH (except, of course, in the definition of the installation path).

Example

```
:i:/opt/C/022A10::0755:2:2      # default installpath
:d:$IPATH/bin::0755:2:2         # subdirectory
C89:f:$IPATH/bin/c89::0755:2:2  # command
```

It makes no sense to define an installation path in the deinstallation script. The specification :i: is ignored if it is syntactically correct.

The variable $IPATH in the deinstallation script is replaced:

- in interactive mode, by selecting the appropriate item in the deinstallation mask in which all currently installed products are listed.
- in batch mode, by the installation path which is specified in the statement line of the parameter file.

Messages, inputs and return codes in installation scripts

Shell scripts can be executed in installation scripts. Messages can be output from these shell scripts, and inputs can be made to them. Depending on the type of installation, the following applies for the inputs/output:

- In the event of installation in interactive mode, the inputs are read from the terminal and the outputs appear on the terminal, i.e. *stdin*, *stdout* and *stderr* are redirected to the terminal.
- In the event of installation in interactive mode, the inputs are read in from a response file. This file must be located in the directory from which the shell script is started. The name must consist of the name of the shell script plus *.response*.

  The outputs (*stdout* and *stderr*) are always directed to SYSOUT.

Control statements

A shell script can contain control statements which report the exit status if the return code is not 0 and determine how installation is to continue. These control statements always start with a lozenge and an exclamation point in the 1st line and can be contained at any position in the shell script.

The following control statements are possible:

```
#!REPORT_SHELLSCRIPT_ERROR={ON | OFF}
```

ON If the return code is not 0, a message is output (default). This message has the following format:

"shell script *script name* reports error *exit value*"

The meaning must be described in its context in the product description.

To prevent ambiguity, the return codes of the shell scripts should be in the range 128 – 255 as the POSIX shell also sends return codes.

OFF No message is output.

```
#!EXIT_ON_SHELLSCRIPT_ERROR={ON | OFF}
```

ON If the return code is not 0, product installation is aborted.

OFF Installation of the product is continued (default).

Example

The product *sample* (Version *123*) is installed from the product library *SINLIB.SAMPLE.123*. The installation script *INSTALL.SAMPLE.123* contains the following lines:

```
:i:/tmp/sample.install::0755:2:2
sample.sh:p:SIPATH/script::0555:2:2
sample.rs:p:SIPATH/script.response::0555:2:2
#!REPORT_SHELLSCRIPT_ERROR=ON
#!EXIT_ON_SHELLSCRIPT_ERROR=OFF
:r:SIPATH/script::::
```

5.3 Initial installation of POSIX

During an initial installation, a new POSIX root and var file system is created.

The procedure for an initial installation involves:

1. Reading in the POSIX SOLIS delivery with IMON

   When the product has been installed in BS2000 using the SOLIS/IMON procedure, the subsystem catalog, Software Configuration Inventory (SCI), message files and SDF syntax files etc. are updated.

2. Making preparations

   When POSIX is installed for the first time on a system which has **never** had POSIX run on it, certain measures must be taken after the product has been installed in BS2000, see section “Preparing for initial installation”.

3. Carrying out an initial installation of POSIX using the POSIX installation program, see section “Carrying out an initial installation with the POSIX installation program”.

5.3.1 Preparing for initial installation

If POSIX has not previously been run on the system, the following steps must be taken:

- Before you start the POSIX subsystem, remove the CPU limit of the system ID SYSROOT.

  This is necessary to permit the POSIX daemons started as batch tasks under the SYSROOT ID to run without CPU limit (NTL) and thus during the entire service life of the POSIX subsystem.

  Two options are available:

  1. Set the account privilege of the SYSROOT ID to NO-CPU-LIMIT= *YES if the default batch job class (or optionally the default POSIX job class) of the SYSROOT ID is defined with NO-CPU-LIMIT=*NO.

     Example

     ```
     /MODIFY-USER-ATTRIBUTES USER-IDENTIFICATION=SYSROOT, -
     /       ACCOUNT-ATTRIBUTES=*MODIFY(ACCOUNT=SYSACC, -
     /           PRIVILEGE=*PARAMETERS(NO-CPU-LIMIT=*YES))
     ```

  2. Alternatively you can define the default job classes assigned to the SYSROOT ID (batch and POSIX) with NO-CPU-LIMIT=*YES.

- Unlock the user ID SYSROOT:

  ```
  /UNLOCK-USER USER-ID=SYSROOT
  ```

5.3.2 Carrying out an initial installation with the POSIX installation program

After installing POSIX in BS2000, you must install POSIX and the requisite software in the POSIX file system under the TSOS user ID. You do this with the aid of the POSIX installation program. You may also have to create and process other file systems. During initial installation of POSIX, root authorization (user number 0, group number 0) is automatically assigned to TSOS.

The following steps must be performed under the TSOS user ID for initial installation:

1. Ensure that the POSIX subsystem has **not** been started and that the POSIX information file SYSSSI.POSIX-BC.<version> is writable (ACCESS=*WRITE).
2. Call the POSIX installation program:

   ```
   /START-POSIX-INSTALLATION
   ```

   You can also call the program in batch mode. For a precise description of masks and parameters, see "POSIX installation program in interactive mode " (interactive mode) or "POSIX installation program in batch mode " (batch mode).
3. Create and mount a new root and var file system

   Call the option “Install POSIX subsystem” in the POSIX installation program and enter the specifications for the container file and root file system in the mask. The size of the root file system must be at least 50,000 PAM pages and must be created under SYSROOT. Make the root file system larger if you are installing multiple products (recommended size: 100,000 PAM pages or more).

   The same mask will then be displayed again, so that you can enter the specifications for the var file system. The var file system must be at least 20,000 PAM pages in size and must be created under SYSROOT of the HOME pubset.

   For further information about space requirements on the root and var file systems for the products to be installed, refer to the relevant product manuals and the release notice on POSIX-BC.

Important directories and files are then copied automatically from the generic root file system to the newly created root and var file system. The directories, device and administrative files created in the installation process are logged. Once all directories and files have been copied, it is then possible to work with POSIX using the basic shell. Once the POSIX subsystem has automatically been restarted, the BS2000 command START-POSIX-SHELL can be entered.

5.3.3 Installing additional software

When POSIX has been installed, you can install additional products in POSIX.

The table below provides an overview of all the selectable units containing release units which can be installed in POSIX. The release units contain libraries for installation under POSIX which have the prefix <xxx>LIB, <xxx> being platform-dependent, with the possible values SIN, SKU or (only in the case of POSIX-SOCKETS and POSIX-NSL) SYS, SKM.

There are now a number of products which can be installed automatically in POSIX as part of SOLIS/IMON product installation. The selectable units of these products contain POSIX items of the type *PS or *NP.

Selectable Unit	Release Unit	Brief Description/ Functionality in POSIX	*PS/*NP Item?
POSIX (GA)	POSIX-BC	Basic shell	Y
	POSIX-ADDON-LIB	UNIX-/BS2000-specific extensions to the POSIX library functions	Y
	POSIX-NSL	TLI, RPC and XDR functions	Y
	POSIX-SH	Extended shell	Y
	POSIX-SOKKETS	Socket and XTI functions	Y
INETSERV	MAIL		Y
	IMAP	Internet Message Access Protocol	
	POSTFIX	SMTP server (Simple Mail Transfer Protocol)	
	TCP-IP-AP	File Transfer Protocol ftp	Y
	TCP-IP-SV		Y
	DNS	DNS Resolver	
	NAMED	DNS server	
	NTP	Client and server (Network Time Protocol)	
	OPENSSH	Secure Shell	
IMON (GA)	IMON-BAS	Installation Monitor, rc script for automatic POSIX package installation	Y
NFS	NFS	Network File System	Y
OPENFT	OPENFT	ft client for POSIX interfaces ncopy, ft, etc.	Y
BS2OSD (GA), BS2000 (GA)	NET-SNMP	SNMP management for BS2000	Y
SNMP-AGENTS	SNMP-AGENTS	SNMP agents for BS2000	Y

APACHE (GA)	APACHE	Apache web server	Y
	PERL	Script language perl	Y
CRTE-BAS (GA)	POSIX-HEADER	Include header for POSIX library functions	Y
CRTE	CRTE	Common RunTime Environment for C, C++ and Cobol, include header	Y
CPP	CPP	C/C++ compiler (cc, c89, CC)	Y
COBOL2000	COBOL2000	COBOL2000 compiler (cobol2000)	Y
COBOL85	COBOL85	COBO85L compiler (cobol85)	N
JENV (GA)	JENV	Java environment	Y
ONETSERV	BCAM	"netstat" command	Y
SHC-OSD	SHC-OSD	SHC-OSD components in POSIX	Y
BS2OSD (GA)	SANCHECK	Check of the SAN (Storage Area Network) configuration	Y
WEBTRANS-OSD (GA)	WEBTRANS-OSD	Link BS2000 interactive applications into the WWW	N
WEBTRANS-UTM	WEBTRANS-UTM	Link OpenUTM applications into the WWW	N

Installing the extended POSIX shell

Following initial installation of POSIX, you are recommended always to install the package POSIX-SH (extended shell). Only then is the complete functionality of the POSIX shell commands available.

Installing the C/C++ programming environment

If you wish to develop C/C++ programs in the POSIX shell (POSIX commands *cc*, *c89* or *CC*), you must also install CRTE, CPP and the POSIX-HEADER.

5.3.4 Notes on automatic POSIX package installation using IMON

Prerequisites and general concept

A prerequisite for automatic package installation is that

- IMON-BAS is installed in POSIX,
- "POSIX processing" is activated for the selectable unit when the IMON installation procedure is generated with INSTALL-UNITS or in menu mode and
- the selectable unit of the product contains POSIX items with type *PS.

When a new product version is supplied, the following actions are always performed:

1. Uninstallation of the previous version installed in POSIX
2. Installation of the new product version in POSIX

By default products are also installed in POSIX which were previously not yet installed in POSIX. In this case uninstallation is not required.

Automatic POSIX package installation can be disabled by reading in the SOLIS package (INSTALL-UNITS).

IMON installation procedure in BS2000

The IMON installation procedure executes the following steps when installing the products in BS2000:

1. The $SYSROOT.POSIX.CONFIGURATION file is generated or updated. This contains a list of all packages which are installed in POSIX.
2. Files are created which are used as input files for automatic POSIX package installation with the POSIX installation program (in batch mode):

   $SYSROOT.IMON.ACTIONS.REM (delete packages)
   $SYSROOT.IMON.ACTIONS.ADD (install packages)

   The content of the CONFIGURATION files is relevant for the structure of the REM file. All the named files are created on the home pubset in the SYSROOT ID.

Executing automatic package installation in POSIX

Package installation itself (uninstallation of the old packages and installation of the new packages) is performed automatically by IMON either the next time POSIX is started or as part of dynamic activation using ACTIVATE-UNITS. If problems occur in package installation, these are logged in the */var/sadm/pkg/insterr* file (see "START-POSIX-INSTALLATION").

- Package installation in a POSIX startup:

  Uninstallation/Installation takes place using the */etc/rc2.d/S05imon* script. The input files for the POSIX installation program are the IMON.ACTIONS files set up with the installation procedure. All the products contained there must be activated. They are activated following a restart of BS2000.

  After installation has been completed, the IMON.ACTIONS files are always assigned the suffix ".SAVE".

- Package installation with ACTIVATE-UNITS:

  ACTIVATE-UNITS for the selectable unit BS2GA.POSIX causes POSIX to restart automatically (if the POSIX subsystem is included in the delivery). Package installation then takes place subsequently when POSIX is restarted (see above).

  For all other selectable units package installation is executed as follows: When the activation procedure is performed, IMON moves the entries for products which are to be activated to temporary IMON.ACTIONS files (with the file name suffix #) and performs uninstallation/installation of the packages using these files.

  When the operation is successful, the entries for the installed products are removed from the IMON.ACTIONS files set up by the IMON installation procedure. The original status of these files is then contained in a backup copy with the suffix ".SAV". If an error occurs, the IMON.ACTIONS files remain unchanged.

  > **ATTENTION!**:
  >
  > Dynamic activation of the POSIX selectable unit should always only be performed if it has been performed for the other products, otherwise all products following the POSIX selectable unit will not yet have been activated when the automatic POSIX startup takes place and package installation for them will therefore fail.

Procedure for dynamic activation during ongoing operation

A fundamental distinction must be made according to whether or not the delivery contains the installation unit POSIX-BC (selectable unit BS2GA.POSIX) or CRTE-BASYS (selectable unit BS2GA.CRTE-BAS). Depending on this, the following variants are possible:

1. When deliveries do not include POSIX-BC or CRTE-BASYS, POSIX does not need to be restarted. Package installation takes place automatically as part of ACTIVATE-UNITS during ongoing POSIX operation.
2. In the case of deliveries with POSIX-BC or CRTE-BASYS, POSIX must be restarted. Two procedures are possible here:
   a. Joint installation of all products
      - Terminate POSIX
      - Install all selectable units in BS2000 and activate them with ACTIVATE-UNITS
      - Restart POSIX (automatic package installation takes place as part of the POSIX restart)
   b. Separate installation of POSIX-BC/CRTE-BASYS and other products
      - While POSIX is running, install all selectable units in BS2000 except for BS2GA.POSIX and BS2GA.CRTE-BAS
      - Activate these selectable units with ACTIVATE-UNITS (automatic package installation takes place as part of ACTIVATE-UNITS)
      - Terminate POSIX
      - Install BS2GA.POSIX and/or BS2GA.CRTE-BAS in BS2000 and activate them with ACTIVATE-UNITS
      - Restart POSIX (automatic package installation when POSIX restarts).

5.4 Upgrade installation of POSIX

The POSIX upgrade installation is required if you have already installed POSIX and wish to retain the existing root and var file systems.

The following procedure steps describe both the upgrade to a new POSIX correction state and a POSIX version change (see section “Installing an upgrade for a new POSIX correction status”), which may, for example, be necessary after changing the BS2000 version (see section “Installing an upgrade for a new POSIX version”).

5.4.1 Installing an upgrade for a new POSIX correction status

This involves the following steps:

1. Read in the SOLIS delivery of the new POSIX correction status.

   The newly supplied POSIX parameter file is stored under the name SYSSSI.POSIX-BC.<version>.NEW, i.e. the old SYSSSI parameter file, in which the name of the root file system etc. has already been entered under ROOTFSNAME, is used by default (without the suffix NEW).

2. Start the POSIX subsystem with the command /START-SUBSYSTEM POSIX.

   The new POSIX version is started up with the old file systems.

3. After 'POSIX ready' (message at the console):
   - Start the POSIX installation program (/START-POSIX-INSTALLATION) and select 'Install packages on POSIX'.
   - Install the product 'POSIX-BC'.
   - Then stop the POSIX installation program.

5.4.2 Installing an upgrade for a new POSIX version

To install an upgrade, you need simultaneous access to the POSIX product files <oldversion> and <newversion>.

If additional products (e.g. JENV) are installed in POSIX then these products should be uninstalled before starting with the following steps or the associated product files have to be copied on the new pubset, too.

The following steps must be taken:

1. Create a new HOME pubset for continued use of the POSIX data.

   Copy the old root and var file systems and the old POSIX product files (file names S*.POSIX-BC.<oldver>*) onto the new HOME pubset.
2. Read in the SOLIS delivery of the new POSIX version and proceed as follows:
   - Enter the file name of the root file system in the parameter file SYSSSI.POSIX-BC.<newversion> in the line for the ROOTFSNAME parameter.
   - Activate the version of the POSIX subsystem which has been changed in the subsystem by shutting down the BS2000 system and then restarting it.
3. Start the POSIX subsystem with the command /START-SUBSYSTEM POSIX.

   The new POSIX version is started up with the old file systems.
4. After 'POSIX ready' (message at the console):
   - Start the POSIX installation program (/START-POSIX-INSTALLATION) and select 'Install packages on POSIX'.
   - Install the product 'POSIX-BC' with the new version number <newvers>.
   - Then stop the POSIX installation program.

   Subsequently the product files of the old POSIX version can be deleted.

5.5 POSIX installation program in interactive mode

You can call the POSIX installation program offline (POSIX not started) and online (POSIX started). The main mask differs accordingly.

Calling the installation program offline

```
/START-POSIX-INSTALLATION
```

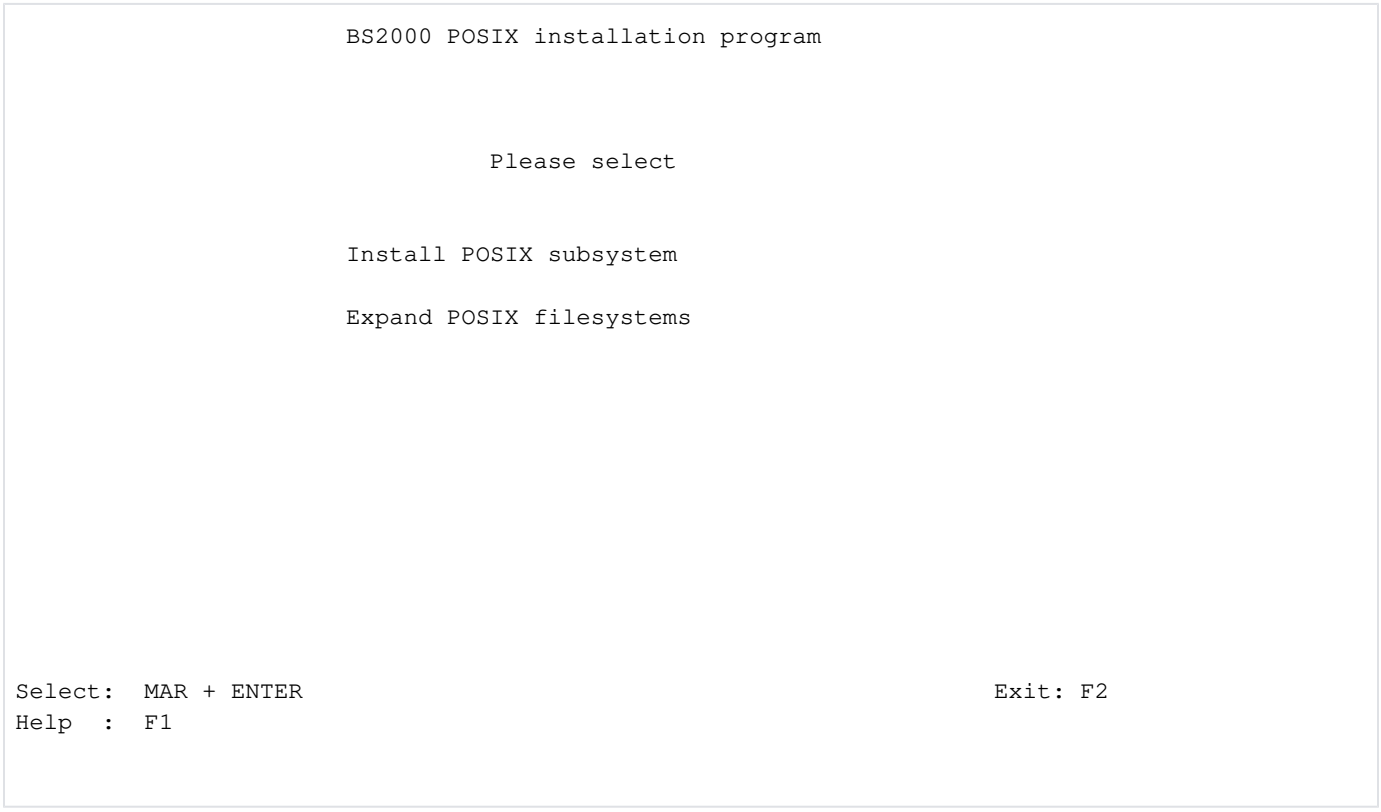

Figure 22: Main mask of the POSIX installation program - offline

The POSIX installation program then provides the following options:

- Install POSIX subsystem (`i` initial installation)
- Expand POSIX filesystems

Calling the installation program online

```
/START-POSIX-INSTALLATION
```

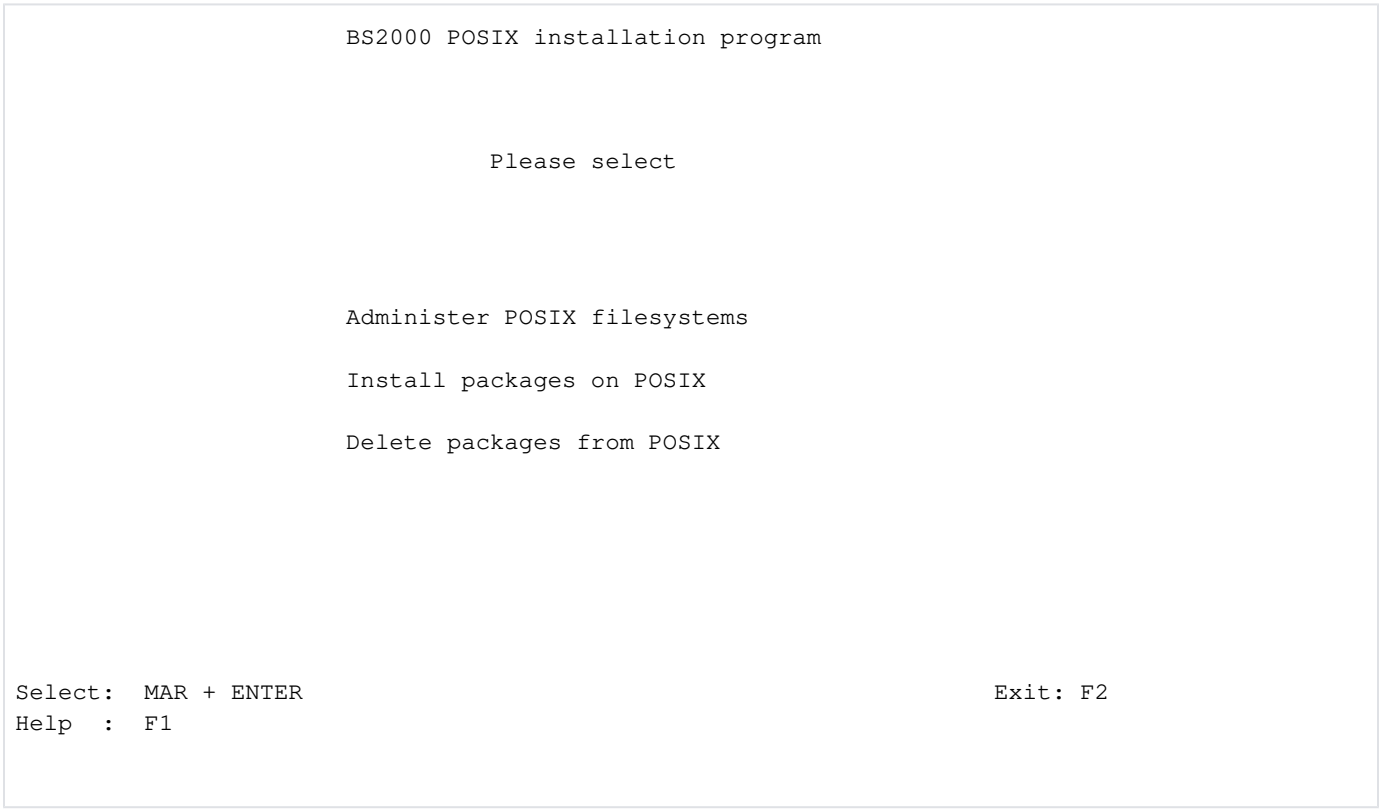

Figure 23: Main mask of the POSIX installation program - online

If the POSIX subsystem is active, the following options are available:

- Administrate POSIX file systems
- Install packages on POSIX
- Delete packages from POSIX

Select an option by highlighting it with the cursor, marking it by typing a character or by pressing the **MAR** key. Confirm your choice by pressing **ENTER**.
Press **F1** for context-sensitive help. Press **F2** to exit the installation program. The bottom line on the screen contains messages and information issued by the installation program.

5.5.1 Install POSIX subsystem

You can use this option to install a new POSIX subsystem.

The root and var file systems are installed as specified, directories are installed in the POSIX file system and the POSIX subsystem is started with the root file system just created.

```
                    Definition of BS2000 Container File

BS2000 filename:  $SYSROOT.FS.ROOT

BS2000 filesize:  150000    PAM-Pages

POSIX filesystem? (y/n): Y

================================================================================

                        Definition of POSIX filesystem

Size of filesystem:           PAM-Pages     Journaling? (y/n): N

POSIX mountpoint: /

Automount? (y/n): Y          Mountoptions:

Overwrite existing filesystem? (y/n):       POSIX filesystem marker (y/n): Y

================================================================================
Save definitions:  ENTER
Help            :  F1                                        terminate: F2
Indicate name of BS2000 container for the root filesystem
```

Figure 24: Follow-uo mask for “Install POSIX subsystem”

Definition of BS2000 container file:

BS2000 filename

Name of the PAM file to be used as a container file for the root or var file system.
The file name must contain the user ID SYSROOT. If the file does not yet exist, it is created with the specified size.

BS2000 filesize
Size of the container file in PAM pages (unit: 2Kb). The minimum size is 4096 PAM pages.The desired size must be entered for newly created container files.
If the container already exists, the actual size is incorporated into the field. You cannot modify the value in this case.

POSIX filesystem? (y/n)
Answer the query with y (yes) or n (no). The container file should normally contain a POSIX file system. However, you can bypass access via the POSIX file system in special cases and access the file directly (*raw* access).
The fields for the root and var file systems default to y (yes).

Definition of the POSIX file system

Size of filesystem

If a file system already exists in the BS2000 container file, its size is shown here in PAM pages (2 Kb). If no file system exists as yet, the field contains the size of the BS2000 container file.
You cannot overwrite the value in this field, because the size of a file system in a BS2000 container file is always the same as that of the BS2000 container file.

Journaling? (y/n)
In this field you can define whether the file system is to be defined with (y) or without (n) a journal. If you make no specification, the default value is = n.

> The function "Journaling" is no longer supported, i.e. the parameter is ignored.

POSIX mountpoint
Directory in which the POSIX file system is to be mounted.
You must enter the absolute name of the directory and the name must begin with a slash (/). If the directory does not yet exist, the program creates it.
The fields for the root and var file systems default to /.

Automount? (y/n)
If the file system is to be mounted immediately and automatically for every subsystem start, you must enter y (yes).
If you are only setting up the file system but do not yet wish to use it, you must enter n (no).
The fields for the root and var file systems default to y (yes).

Mountoptions
You can parameterize the mounting of the file system. You will find the appropriate options for the *mount* command in the "POSIX Commands" [1] manual. Multiple options must be separated by commas.

Overwrite existing filesystem? (y/n)
Answer the query with y (yes) or n (no).
This field is only activated if the container file already contains a POSIX file system. You must decide whether the file system is to be transferred unaltered or whether a new file system is to be created. When the root or var file system is created, this field is not activated, as these file systems are always overwritten.

POSIX filesystem marker (y/n)
Meaning: file system created in POSIX/BS2000.
The fields for the root and var file systems default to y (yes).

If the marker is not set, the file system is considered to be an ASCII file system under POSIX. This means that an ASCII-EBCDIC conversion takes place, depending on the *IO_CONVERSION* environment variable (see “Copying and converting files”).

5.5.2 Expand POSIX filesystems

This option enables you to expand any POSIX file system, including *root* and *var*, in offline mode.

> You can also expand all POSIX file system except *root* and *var* in online mode (see "Administer POSIX filesystems ", *expand* command).

```
                          Expand of POSIX filesystem

BS2000 filename:

   Characteristics         before expand            after expand

   BS2000 filesize                PAM-Pages                PAM-Pages
size of filesystem                PAM-Pages                PAM-Pages
            inodes
       free inodes
        datablocks                (4 kB)                   (4 kB)
   free datablocks                (4 kB)                   (4 kB)

Best value for expand is           PAM pages + N *           PAM pages (N >= 0)



                  Expand value:           PAM pages

   NOTE:    Filesystem compactification is always enabled, if applicable.
            For more information refer to the online help (F1).
================================================================================
Execute expand: ENTER             Help: F1                    Terminate: F2
```

Figure 25: Follow-up mask for “Expand POSIX filesystems”

BS2000 filename
In this field you enter the name of the container file for the file system that is to be expanded.

Characteristics before expand / after expand
The current characteristics of the file system before and after the expansion are displayed in these columns.

Best value for expand is ...
The optimum value for expansion is displayed in this line (to prevent unused or only partially used PAM pages).

Expand value
In this field you specify the number of PAM pages by which the file system is to be expanded. Repeating ENTER after a successful expansion does not trigger another expansion but is ignored.

A file system is expanded in two steps:

1. Physical expansion
2. Combining administrative units of the file system (known as cylinder groups) to form larger units (compactification).

Compactification is performed only if the expanded file system is greater than 2 GB and at least double the size of the original file system.

Compactification increases the runtime of fsexpand considerably. However, it has the advantage that performance gains can be achieved when using the file system, since the memory and CPU time requirements are lower due to the combination of the administrative units.

If compactification is not desired, the shell command "fsexpand -N ..." can be used instead of the installation program.

After successful expansion, a new file system can be specified or you can return to the start mask by pressing *F2*.

5.5.3 Administer POSIX filesystems

This option enables you to change, extend and delete existing POSIX file system entries and to generate new POSIX file system entries.

```
                      BS2000 POSIX filesystem table
                                                                    F J A P
   BS2000 filename                                          size     S O M F
.  :PUB1:$SYSROOT.FS.ROOT                                   200001  Y N Y Y
.  $SYSROOT.FS.VAR                                           50001  Y N Y Y
.  $SYSROOT.FS.OPT                                          500001  Y N Y Y
.  $SYSROOT.FS.BS2FSCONTAINER                                20001  Y N Y Y









edit commands:  'a'=append  'm'=modify  'd'=delete  'e'=expand
function keys:  F1=help  F2=terminate             scroll commands: '+'/'-'
command  ====>                                                      more:
```

Figure 26: Follow-up mask for “Administrate POSIX filesystems”

The follow-up mask lists all the BS2000 files that are registered in the POSIX subsystem as local container files. Use “+” and “-” to browse through the list.
Each entry contains:

- the BS2000 file name of the container file
- the size of the container file in PAM pages (unit: 2 Kb)
- the current setting of the following parameters:
  `FS` = File system inside (y/n),
  `JO` = Journaling (y/n),

  > The function "Journaling" is no longer supported, i.e. the column JO is meaningless.

  `AM` = Automount (y/n) and
  `PF` = POSIX file system marker (y/n)

The bottom line contains a selection field (“command”) where you can enter any of the following administrative commands.

‘a‘=append
Create a new entry
Add a container file and, if necessary, a file system to the list. Beforehand, mark the entry after which you want to add the new entry; otherwise the new entry will be appended to the end of the list.

When a file system is created with the append function, the system administrator has the choice of marking the file system as “created by POSIX“ (i.e. as an EBCDIC file system) or not (ASCII file system). The root and var file system are automatically tagged “created by POSIX” on initial installation.

‘m‘=modify
Modify marked entry
You can subsequently modify specific fields of the file system.

‘d‘=delete
Delete marked entry
Only the marked entry is removed from the list. The container file and the file system remain unchanged and can be entered again or deleted later.

‘e‘=expand
Expand marked file system
This command enables you to expand the selected file system, provided it can be unmounted. When this command is entered the same mask is displayed as with the option “Expand POSIX filesystems”.

In contrast to offline mode, the following applies here:

- The *root* and *var* file systems cannot be expanded here as they are always occupied and cannot be unmounted.
- The *BS2000 filename* field already contains the name marked in the *BS2000 POSIX filesystem table* mask and cannot be modified.
- After successful expansion and after you have pressed *F2*, the file system is remounted if it was not mounted before the expansion. This is the case regardless of the Automount setting.
- After you have pressed *F2* the program returns to the previous mask *BS2000 POSIX filesystem table*.

5.5.4 Install Packages on POSIX

Use this option to install POSIX user programs and program packages in the POSIX file system, see also "Installing additional software ".

```
                      BS2000 POSIX package installation


IMON support ?     : Y  (y) mandatory for official package
                        (n) private package (SINLIB...)

name of product    :
package of product :                      (optional for certain products)

version of product :          (format Vmm.n or mmn)

correction state   :          (format aso, optional for IMON support)

installation userid:          (mandatory for no IMON support)









install: ENTER    help: F1     terminate: F2
```

Figure 27: Follow-up mask for “Install packages on POSIX”

IMON support? (y/n)
Defines whether installation is carried out from within the SCI (IMON support: y) or from a private filing ID (IMON support: n).
The default is IMON support: y.

name of product
Product name (release unit).

package of product
Package name if the product is split into packages.

version of product (Vmm.n or mmn format)
Product version:

- with 'IMON support: y' in Vmm.n or mmn format (m,n: digits) or empty
- with 'IMON support: n' in mmn format (m,n: digits)

correction state (aso format)
Only with 'IMON support: y' and only together with 'version of product':
Specifies the correction state in aso format (a: letter; s,o: digits)
The field must remain empty if 'version of product' is empty (see case 1 in section “The installation program in conjunction with IMON”).

installation userid (no IMON support)
Only with 'IMON support: n' (otherwise empty):
User ID of the private filing ID

If incorrect entries are made (e.g. entry of characters in an “empty” field), an error message is output and the mask is presented again for modification.

> ¡ Before installing a new version, use “Delete packages from POSIX” to remove the old version of the program package. Note that this cannot delete the extended shell (package name POSIX-SH).

5.5.5 Delete packages from POSIX

Use this option to delete POSIX user programs and program packages.

```
                      BS2000 POSIX package delete


   Product                  Version Package             Date of installation
   POSIX-SH                 100                         Jan 17 14:27:29 2020
   /
   POSIX-SOCKETS            100                         Jan 17 14:31:31 2020
   /
   POSIX-NSL                100                         Jan 17 14:32:05 2020
   /
   NFS                      030                         Jan 17 14:32:24 2020
   /
   IMON-BAS                 033                         Jan 17 14:32:35 2020
   /
   POSIX-ADDON-LIB          021                         Jan 17 14:38:26 2020
   /
   JENV                     090                         Jan 17 14:39:28 2020
   /opt/java/jdk-9.0.4
   CPP                      040                         Jan 17 14:40:10 2020
   /opt/C

scroll: + (%/+/-/$)
delete: mark product with 'x' and ENTER               help: F1  terminate: F2
```

Figure 28: Follow-up mask for “Delete packages from POSIX”

You must select the components that you wish to delete. The POSIX shell itself cannot be deleted.

5.6 POSIX installation program in batch mode

Call the POSIX installation program for a batch run with:

```
/START-POSIX-INSTALLATION -
/  INPUT-INTERFACE=*FILE( -
/     FILE-NAME=<parameter file>, -
/     ERROR-HANDLING=*PARAMETERS(...) -
/  )
```

For the FILE-NAME operand, enter the name of a parameter file which contains the installation information in the form described below.

The behavior when an error occurs can be controlled using the ERROR-HANDLING operand (for detailed information on this, see the description of the START-POSIX-INSTALLATION command).

5.6.1 Format of the parameter files

A parameter file consists of an identification line, one or more statement lines, and (optional) comment lines.

Comments

Comments and comment lines are optional. They must always begin with the hash character “#”.

Identification line

The first line in a parameter file that is not a comment line must be the identification line, which selects one of the branches of installation:

[FirstInstallation]	Create the POSIX subsystem
oder	
[ExpandFileSystem]	Expand POSIX file system
oder	
[FileSystemAdministration]	Administrate the POSIX file system
oder	
[PackageInstallation]	Install POSIX program packages
oder	
[DeletePackage]	Delete POSIX program packages

You must always specify the square brackets. You can abbreviate the character strings between the brackets, provided there is no risk of confusion with another string. You can use uppercase and lowercase letters as desired.

Statement lines

The identification line is followed by one or more statement lines which contain the necessary parameters for the selected branch. The semicolon delimiter “;” must be specified even if you do not specify a value for a parameter.

5.6.2 Install POSIX subsystem

Identification line: [**FirstInstallation]**

Statement line: **<file>;<size><jo>**

Meanings of the terms:

<file>	BS2000 file name of the container file
<size>	BS2000 file size of the container file (= size of the file system)
<jo>	Journaling ? (Y/N) default : N

> The function "Journaling" is no longer supported, i.e. the parameter is ignored.

Exactly two statement lines are mandatory.

- The first statement line defines the root file system,
- The second statement line defines the var file system.

If more than two statement lines are specified, the excess lines are ignored. If statement lines are omitted, installation is aborted.

The two statement lines contain the specifications for the BS2000 container of a POSIX file system. In this case, additional specifications necessary for the complete description of a file system are preset in accordance with the following table:

Parameter	Root file system	Var file system
File system flag	Y	Y
POSIX file system marker	Y	Y
POSIX mountpoint	/	/var
Automount	Y	Y
Mounting options	-	-
Overwrite	N	N

Example

```
#
# Batch installation file
#
[FirstInstallation]          # POSIX initial installation
# <file>;<size>
$SYSROOT.FS.ROOT;20000  # Installing the root file system
$SYSROOT.FS.VAR;50000   # Installing the var file system
```

5.6.3 Expand POSIX filesystems

Identification line: **[ExpandFileSystem]**

Statement line: **<file>;<size>**

Meanings of the terms:

<file> BS2000 file name of the container file

<size> Size of the expansion

This statement is only executed for file systems which can be unmounted or are not mounted. In order to expand *root* and *var* file systems, for example, the POSIX subsystem must be terminated as these file systems cannot be unmounted during ongoing operation.

Example

```
#
# Batch installation file
#
[ExpandFileSystem]                  # Expand POSIX file systems
# <file>;<size>
# Expand root file system by 10,000 PAM pages
$SYSROOT.FS.ROOT;10000
# Expand var file system by 20,000 PAM pages
$SYSROOT.FS.VAR;20000
```

5.6.4 Administrate POSIX file systems

Identification line: [**FileSystemAdministration]**

Statement line:
<op>;<file>;<size>;<flag>;<mark>;<mdir>;<auto>;<mopt>;<ov><jo>

Meanings of the terms:

<op>	Editing command: a(ppend), m(odify) or d(elete)
<file>	BS2000 file name of the container file
<size>	BS2000 file size of the container file (= file system size)
<flag>	file system flag? (Y/N)
<mark>	POSIX file system marker? (Y/N)
<mdir>	POSIX mountpoint
<auto>	Automount? (Y/N)
<mopt>	Mounting options
<ov>	Overwrite POSIX file system? (Y/N)
<jo>	Journaling ? (Y/N)

> The function "Journaling" is no longer supported, i.e. the parameter is ignored.

Each statement line contains the editing command and the specifications for the BS2000 container file and for the POSIX file system. You do not have to specify all the parameters for every editing command. Therefore, when the editing command m(odify) is used, for example, the BS2000 file size does not have to be modified. The following table shows which parameter must be specified with which editing command:

Parameter	a(ppend)	m(odify)	d(elete)
BS2000 file name	x	x	x
BS2000 file size	x	–	–
POSIX file system flag	dy	–	–
POSIX file system marker	xy	–	–
POSIX mountpoint	xm	o	–
Automount	dy	o	–
Mounting options	dl	o	–
Overwrite POSIX file system	xo	–	–

Journaling	dn	o	–

Explanations:

- – Is ignored or has no effect if the syntax is correct
- dy Y is the default value
- dn N is the default value
- dl The empty character string is the default value
- o Optional; if there is no value, the current setting is valid
- x Mandatory entry
- xy Mandatory entry if y is specified for the POSIX file system flag; otherwise ignored
- xm Mandatory entry if y is specified for Automount; otherwise ignored
- xo Mandatory entry for an Overwrite situation; otherwise ignored

Example

```
#
# Batch installations file
#
[FileSystemAdministration]            # Administer POSIX file systems
# <op>;<file>;<size>;<flag>;<mark>;<mdir>;<auto>;<mopt>;<ov>
# Create new BS2000 POSIX file system
# An existing file system will be overwritten
append;$SYSROOT.FS.USR;50000;;y;/usr;y;;y
# Delete file system
delete;$SYSROOT.FS.USR;
# Create new BS2000 POSIX file system
# An existing file system will not be overwritten
append;:PUB:$SYSROOT.FS.USR;50000;;y;/HIPLEX/PUB/usr;y;;n
# Link an existing file system to /usr/home
modify;$SYSROOT.FS.HOME;;;;/usr/home;y;;y
```

5.6.5 Install packages on POSIX

Identification line: [**PackageInstallation]**

Statement line: **<prod>;<imon>;<vers>;<corr>;<uid>;<ipath>**

Meanings of the terms in angle brackets:

<prod> Name of the software package, comprising the product name and optional package names, if the product is split into packages.
Syntax: <productname>[:<packagename>]

<imon> IMON flag, defines whether installation is carried out from within the SCI (Y)or not (N)

<vers> Product version of the software package:

- with IMON flag= 'Y' in Vmm.n or mmn format (m,n: digits) or empty
- with IMON flag= 'N' in mmn format (m,n: digits)

<corr> Correction status in aso format (a: letter; s,o: digits)
The field must remain empty if 'vers' is empty, see Example 1 in the section "The installation program in conjunction with IMON"

<uid> BS2000 user ID of the archive ID, mandatory for IMON flag= 'N', ignored for IMON flag = 'Y'.

<ipath> Optional installation path, if the software package supports this.
Default value is '/'.

If the IMON flag = 'N', the name of the BS2000 PLAM library which contains the software package is formed as described in the section "Preparing private program packages for installation".

Example

```
#
# Batch installation file
[PackageInstallation]   # install program packages
# <product[:package]>;<imon>;<version>;<corr>;<uid>;<ipath>
#Installation of extended shell with IMON
POSIX-SH;Y
#Installation of NFS without IMON
NFS;N;030;;TSOS
# Installation of C89 with IMON
CPP;Y;;;;/opt/C
# Installation of OPENSSH with IMON
TCP-IP-SV:OPENSSH;Y
```

5.6.6 Delete packages from POSIX

Identification line: **[DeletePackage]**

Statement line: **<prod>;<vers> ;<ipath>**

Meanings of the terms in angle brackets:

<prod> Name of the software package, comprising the product name and optional package names, if the product is split into packages.
Syntax: <productname>[:<packagename>]

<vers> Product version of the software package

<ipath> Optional installation path, if the software package supports this.
Default value is '/'.

Example

```
#
# Batch installation file
#
[DeletePackage]              # Delete program packages
# <product[:package]>;<version>;<ipath>
# Delete NFS
NFS;030
# Delete C89 on specific installation path
CPP;032;/opt/C
```

5.7 Logging the installation

POSIX logs the package installation in the logging file */var/sadm/pkg/instlog*.

An entry containing the following information is written for each process:

- Flag for installation or deletion (in the 1st column: I (install) or D (delete))
- Name and version of the product
- Date and time of the installation or deletion
- Name of the package (optional)
- Installation path

Example

```
I JENV.050                Wed Sep 19 13:15:46 2007   ...
I BCAM.190                Mon Jun  9 11:35:10 2008   ...
I NFS.030                 Thu Nov 20 12:23:41 2008   ...
I POSIX-BC.070            Tue Jan 27 11:33:27 2009   ...
I POSIX-SH.070            Tue Jan 27 11:35:13 2009   ...
I POSIX-SOCKETS.070       Tue Jan 27 12:53:51 2009   ...
I POSIX-NSL.070           Tue Jan 27 12:58:30 2009   ...
```

Information on the packages installed can be output using the shell command *pkginfo*.

If errors occur during installation in batch mode, these are logged in the */var/sadm/pkg/insterr* file (see “START-POSIX-INSTALLATION").

Information on installed POSIX packages (pkginfo)

The *pkginfo* command shows information on software packages which are installed in POSIX. A software package installed in POSIX is defined by:

- Name of the software product
- Package from the software product (optional)
- Version of the software product
- Path where the software product is installed (default: /)
- BS2000 library (SINLIB) from which the software product was installed
- Date of the (last) installation

This command is also available to every nonprivileged POSIX user.

5.8 POSIX information file

Control parameters in the POSIX information file SYSSSI.POSIX-BC.<version> determine the size of the system tables. This is used to control the resources which the system and the user can request.

Each control parameter is assigned a default value, a minimum value and a maximum value. The default values are selected so that the POSIX subsystem can operate in any environment without burdening the overall system by making excessive use of resources. Sie können diese Steuerparameter jedoch bei Bedarf an die Gegebenheiten Ihres Systems anpassen.

The POSIX information file is only evaluated when the POSIX subsystem is started up; changes therefore only become effective after the next startup. However, some control parameters can also be modified during ongoing operation using the *usp* command, see the manual "POSIX Commands" [1].

5.8.1 Contents of the POSIX information file

The following table lists all parameters in alphabetical order. Parameters displayed in **bold print** can be modified dynamically using the *usp* command. The column *Category* indicates the category the parameter belongs to:

General	General system parameter
File system	File system parameter
IPC	Control parameter for interprocess communication
POSIX	Special parameter for POSIX

In the as-delivered state of the information file, no value is entered for the ROOTFSNAME control parameter. When POSIX is installed for the first time, the name of the root file system is automatically entered by the POSIX installation program. The system ID SYSROOT is mandatory since it is owner of the root filesystem and needs not to be specified. You can enter numerical values in the units of measure kilo (K, equivalent to 1024) and mega (M, equivalent to 1048576).

Parameter	Description	Default	Min. value	Max. value	Category
BINDANY	BCAM / mode flag	0	0	1	POSIX
BUFHWM	high-water-mark of buffer cache	2000	200	2000	file system
DBLPOOL	memory pool in class 6 memory	0	0	1024	POSIX
DBLSTATE	state of the loader	0	0	1	POSIX
FDFLUSHR	fsflush time interval	5	1	5	file system
FILESIZE	max. size of file	UNLIMITED	64	UNLIMITED	general
FLCKREC	max. # of active file records locks	1000	100	2000	general
FORCEDTERM	controlling termination	0	0	1	POSIX
HDPTNI	# of partition table entries	200	16	300	general
HDSTNI	# of hard disk server tasks	4	1	16	general
HEAPSZ	size of heap-segment	4M	2M	64M	general
KMAHWM	kma daemonkmem high water mark	2M	1M	2M	general
MAXTIMERC	max. wait time for rc term procs	660	120	1200	POSIX
MAXUP	max. # of processes per user	50	15	500	general
MINPAGEFREE	pageout daemon / min. # of free pages	0	0	0	general
MSGMAP	# of entries in msg map	200	10	200	IPC
MSGMAX	max. message size	2048	512	2048	IPC

MSGMNB	max. # bytes on queue	16384	4096	16384	IPC
MSGMNI	# of message queue identifiers	150	50	150	IPC
MSGSEG	# of msg segments (MUST BE < 32768)	2048	1024	32768	IPC
MSGSSZ	msg segment size	8	8	8	IPC
MSGTQL	# of system message headers	160	40	160	IPC
NAUTOUP	age of a delayed-write buffer	60	10	120	file system
NBUF	# of I/O buffers	200	100	2000	file system
NHBUF	buffer cache size for metadata	256	32	1024	file system
NOFILES	max. # of file descriptors	2048	2048	4096	general
NOPTY	max. # of ptys	64	4	1024	POSIX
NOSTTY	max. # of sttys	64	4	2000	POSIX
NOTTY	max. # of ttys	64	4	1024	POSIX
NPBUF	number of physical I/O buffers	20	20	40	general
NPROC	max. # of processes	200	50	2000	general
NRNODE	max. # of incore remote nodes (nfs)	600	400	600	file system
PGOVERFLOW	overflow buffers for pageout	32	32	32	general
PORTMON	port monitoring (nfs)	1	0	1	POSIX
ROOTFSNAME	name of root-file system	---	---	---	---
SEGMAPSZ	# of buffer cache entries	256	256	22500	file system
SEMAEM	adjust on exit max value	16384	16384	16384	IPC
SEMMAP	# of entries in semaphore map	150	10	150	IPC
SEMMNI	# of semaphore identifiers	150	10	150	IPC
SEMMNS	# of semaphores in system	200	60	200	IPC
SEMMNU	# of undo structures in system	200	30	200	IPC
SEMMSL	max. # of semaphores per id	25	25	25	IPC
SEMOPM	max. # of operations per semop call	20	10	20	IPC
SEMUME	max. # of undo entries per process	20	10	20	IPC
SEMVMX	semaphore maximum value	32767	32767	32767	IPC

SHMMAX	max. size of a shared memory segment	16M	131072	16M	IPC
SHMMIN	min. size of a shared memory segment	1	1	1	IPC
SHMMNI	# of shared memory headers	100	100	100	IPC
SHMSEG	max. # of segments per process	16	6	16	IPC
UFSNINODE	# of inodes	1000	600	1000	file system

5.8.2 Description of control parameters

In many cases, the default values of the control parameters are sufficient. However, it may occasionally be useful for the BS2000 system administrator to adapt the control parameter to suit the specific POSIX application and the resources of the overall system. The control parameters for which modification can be useful are listed below. The meaning of every control parameter is also specified.

General system parameters

FILESIZE	Maximum size of a file for creating and writing. The maximum value preset is 1024 Gbyte. For reasons of compatibility, UNLIMITED64 can also be specified instead of UNLIMITED.
FLCKREC	Number of locking structures used by the system for data records (record locks).
HDPTNI	Maximum number of mounted local file systems.
HDSTNI	Number of server tasks for performing asynchronous I/Os.
HEAPSZ	Maximum value possible for *brk()* system call.
KMAHWM	If the dynamic Cl-4 memory map in POSIX exceeds the specified value, the kernel memory daemon will be activated to reorganize and release the memory.
MAXUP	Maximum number of processes which a nonprivileged user can start simultaneously (not for each terminal but all together).
MINPAGEFREE	No meaning, as it cannot be set. MINPAGEFREE is implicitly set to 128 K, i.e. if less than 128 K is free in the buffer cache, pageout will be activated.
NOFILES	Maximum number of open files in the system.
NPBUF	Maximum number of I/O buffers for physical I/Os. This value should be at least 4 * HDSTNI.
NPROC	Maximum number of user processes allowed in the system.
PGOVERFLOW	Number of reserved I/O buffers for pageout even with a memory bottleneck.

File system parameters

BUFHWM	Size of memory (in kilobytes) which can be occupied by the I/O buffer.
FDFLUSHR	Time interval (in seconds) between two activations of a process. *fsflush* writes data from the cache to the hard disk, and thereby ensures the consistency of the data on the hard disk. A low value for FDFLUSHR offers greater security against data loss in the event of a system crash, albeit at the expense of system performance.
NAUTOUP	Specification (in seconds) of how long a buffer must “age” in memory before it is reloaded by *fsflush.* This value only affects the contents of the cache buffer.
NBUF	Number of I/O buffers of the cache buffer which are assigned by the system kernel if no more are free.
NHBUF	Number of hash anchors for rapid access to cache buffers via device and block numbers.

NRNODE — Maximum number of NFS-rnode structures.
These are specific descriptors for open files from NFS file systems, i.e. these files are located on remote computers.

SEGMAPSZ — Maximum size of the cache buffer in class-4 memory (in units of 8KB).
This parameter only has an effect on the I/O throughput under certain conditions.
Generally, i.e. on hardware with data space support, the cache buffer of the POSIX kernel is kept in data spaces and not in class-4 memory, and this parameter has no effect.

UFSNINODE — Maximum number of UFS index entries in the system kernel.

Control parameters for interprocess communication

Message queues and semaphores are administered via resource maps. Resource maps keep a record of how much memory space is used by messages and semaphores. The number of occupied entries of a resource map at a given time serves as a measure for the current partitioning of the memory area available for messages or the available semaphores.
If control parameters such as MSGSEG or SEMMNS are increased, the size of the corresponding resource map should also be increased accordingly.

MSGMAP — Number of entries in the resource map for message queues.

MSGMAX — Maximum size of a message (in bytes).

MSGMNB — Maximum total size of all messages in a message queue (in bytes).

MSGMNI — Maximum number of message queues throughout the system.

MSGSEG — Number of message segments in the system. If the value of MSGSSZ is multiplied by the value of MSGSEG, the result is the total memory space which is available for message data.

MSGSSZ — Minimum allocation size for message memory (segment size in bytes).

MSGTQL — Number of message headers in the system. This number corresponds to the number of outstanding messages.

SEMAEM — Maximum undo value for one semaphore.

SEMMAP — Number of entries in the resource map for semaphore records.

SEMMNI — Maximum number of semaphore records.

SEMMSL — Maximum number of semaphores per record.

SEMMNS — Maximum number of semaphores in the system.

SEMMNU — Maximum number of processes with outstanding undo operations.
Processes can determine whether their semaphore actions are to be automatically undone at the end of the process.

SEMOPM — Maximum number of semaphore operations that can be executed per *semop*(2) system call.

SEMUME — Maximum number of undo operations per process.

SEMVMX — Maximum value for one semaphore.

SHMMAX — Maximum size of a shareable memory area (in bytes).

SHMMIN	Minimum size of a shareable memory area (in bytes).
SHMMNI	Maximum number of shareable memory areas.
SHMSEG	Maximum number of shareable memory areas which a process uses simultaneously.

Special parameters for POSIX

BINDANY	The BINDANY parameter is meaningless as of BCAM Version 13.
DBLPOOL	To speed up the loading process for Posix shell commands and other POSIX programs with the POSIX loader, a value greater than zero may be entered here (in MB). See also section “POSIX loader”.
DBLSTATE	Specifies whether the POSIX loader is automatically activated when the POSIX subsystem is started up: 0=no (default); 1=yes.
FORCEDTERM	Forced termination of the POSIX subsystem. If connections exist, this parameter controls whether a second STOP-SUBSYSTEM command with the parameter SUB-PARAMETER='FORCED-BY-SUBSYSTEM' must be issued, or whether the subsystem is to be stopped immediately without a second STOP-SUBSYSTEM command. FORCEDTERM=0 (previous behavior) FORCEDTERM=1 (forced termination)
MAXTIMERC	Maximum wait time for completing the rc termination procedures when terminating POSIX.
NOPTY	Maximum number of physical terminals (device dev/pts). This corresponds to the permissible number of *rlogin* and *telnet* accesses.
NOSTTY	Maximum number of system file terminals (device dev/sf) supported by POSIX. This corresponds to the permissible number of POSIX accesses via BS2000 procedures and programs.
NOTTY	Maximum number of block terminals (device dev/term) supported by POSIX. This corresponds to the number of POSIX accesses via BS2000 dialog tasks (START-POSIX-SHELL command).
PORTMON	Switch on/off port monitoring for NFS (0=off, 1=on).

6 POSIX subsystem and POSIX loader

This chapter is intended for BS2000 and POSIX file administrators. It contains useful information on

- Controlling the POSIX subsystem (starting, terminating and monitoring)
- the POSIX loader (overview, initialization, linking, loading, administering)

6.1 Controlling the POSIX subsystem

This section describes how to start, terminate and monitor the POSIX subsystem. It contains information on BCAM dependencies on startup and termination.

6.1.1 Starting the POSIX subsystem

The following conditions must be satisfied before POSIX can be started by a user with the SUBSYSTEM-MANAGEMENT or OPERATING privilege:

- The POSIX subsystem must be installed (see “Installing POSIX”).
- The POSIX information file may have to be modified (see "POSIX information file").

  The name of the container file where the root file system is located must match the corresponding control parameter ROOTFSNAME in the POSIX information file.

  During initial installation, the name of the new root file system is entered in the POSIX information file. Therefore, it is not necessary in this case to check for a match.
- Write access to the container file of the root file system and all other file systems which are to be linked during POSIX startup must be possible (attribute ACCESS=*WRITE in the file catalog).
- POSIX must be entered in the subsystem catalog.
  When installing with IMON, it is entered automatically.

POSIX is started either automatically following successful initial installation, or explicitly by means of the BS2000 command

```
/START-SUBSYSTEM SUBSYSTEM-NAME=POSIX.
```

If the POSIX subsystem startup was successful, the following message appears on the screen:

```
POS4100: INIT: THE POSIX SUBSYSTEM IS READY.
```

If startup was unsuccessful, e.g. because the started initialization process could not be terminated, the cause is recorded in the log file of the init process $SYSROOT.SYSLOG.POSIX-BC.<version>.INIT.

Subsystem parameters

Two parameters are supported in the START-SUBSYSTEM command in order to force a consistency check and cleansing of the file system when POSIX starts up. The entries for these parameters are not case-sensitive, but they may not be abbreviated.

- `/START-SUBSYSTEM POSIX,SUBSYSTEM-PARAMETER='CHECK-SYSTEM-FS'`

  The file systems below are checked with *fsck* before they are mounted and cleansed if required:

  / (root file system)

  /var

  /opt (only if present)
- `/START-SUBSYSTEM POSIX,SUBSYSTEM-PARAMETER='CHECK-ALL-FS'`

  All file systems are checked with *fsck* before they are mounted and cleansed if required.

Support of rc procedures

POSIX does not support the run level mechanism of native UNIX. However, similarly to UNIX, it is possible to define rc procedures which run automatically when POSIX is started and terminated. In the same way as for native UNIX, any rc procedures that are to be activated during startup must be stored in the */etc/rc2.d* directory and procedures that are activated during termination must be stored in the */etc/rc0.d* directory. The rc procedures are called one after the other in alphabetical order by the shell script */etc/rc2* or */etc/rc0* when POSIX is started or terminated. If the */etc/.trace.rc* file exists, these calls are logged on the BS2000 console.

rc procedures for starting and terminating the following daemons are configured with POSIX-BC and POSIX-SH: *shmd* (shared memory daemon), *syslogd* (syslog daemon), *fsmond* (daemon for monitoring file system occupancy), *rpcbind* (daemon for RPC services), *inetd* (internet super daemon for network services), *cron* (daemon for the *cron* and *at* commands). Additional software products such as NFS install their own rc procedures for starting or terminating further daemons.

The maximum wait time for execution of the rc termination procedures during POSIX termination can be set by means of the new MAXTIMERC parameter in the information file (see "Description of control parameters"). If execution of the rc termination procedures has not been completed by the time this period has elapsed, depending on the value of the FORCEDTERM parameter POSIX termination will either be canceled or continued in abnormal mode.

6.1.2 Terminating POSIX

POSIX is terminated explicitly by a user with the SUBSYSTEM-MANAGEMENT or OPERATING privilege, or automatically during shutdown of the BS2000 operating system. If a fatal error occurs, POSIX is terminated abnormally.

Explicit POSIX subsystem termination

The POSIX subsystem is terminated by the BS2000 system support or the operating with the following command:

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=POSIX
```

If users are still linked to the POSIX subsystem and FORCEDTERM=0 is set in the POSIX information file when POSIX is terminated, DSSM cancels termination. You can then force termination as follows:

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=POSIX, -
/ SUBSYSTEM-PARAMETER='FORCED-BY-SUBSYSTEM'
```

The entry for the parameter is not case-sensitive but may not be abbreviated.

If FORCEDTERM=1 is set in the POSIX information file, POSIX is always terminated immediately. No second STOP-SUBSYSTEM command is then required.

The POSIX subsystem cannot be halted with the command /HOLD-SUBSYSTEM POSIX. The command is rejected and the current POSIX session is not interrupted.

When the POSIX subsystem is terminated, the following message appears on the screen:

```
POS3010: SUBSYSTEM POSIX HAS BEEN TERMINATED.
```

POSIX subsystem termination during shutdown of the BS2000 operating system

DSSM terminates the POSIX subsystem implicitly during shutdown so that the POSIX files are kept consistent.

Abnormal termination of POSIX

If a fatal error occurs, POSIX is terminated abnormally. In this case, BS2000 subsystem administration and POSIX cooperate closely. All programs using POSIX are terminated abnormally and the BS2000 resources used by POSIX are released.

If the initialization process terminates itself during a POSIX session, an abnormal POSIX termination is initiated, since the initialization process has a central control function in POSIX and is therefore essential for error-free operation.

6.1.3 Monitoring the POSIX subsystem using a monitoring job variable

The POSIX subsystem can be monitored using the monitoring job variable $.SYS.POSIXSTATUS. To do this, start the POSIX subsystem with:

```
/START-SUBSYSTEM POSIX,MONJV=$.SYS.POSIXSTATUS
```

This monitoring job variable can be used to query both the statuses set by DSSM and the status of the POSIX subsystem. The following POSIX subsystem statuses are possible:

Time	Column 89 in MONJV:
before /START-SUBSYSTEM	NOT CREATED
before “Subsystem ready”	IN CREATE
before “POSIX ready”	CREATED
after “POSIX ready”	*AVAILABLE
after /STOP-SUBSYSTEM	IN DELETE
after /STOP rejected	CREATED
subsystem unloaded	NOT CREATED

If a BS2000 session is not terminated properly with SHUTDOWN, the monitoring job variable $.SYS. POSIXSTATUS remains locked. Before it can be used again to monitor the POSIX subsystem, it must be unlocked with the following command:

```
/MODIFY-JV-ATTRIBUTES JV-NAME=$.SYS.POSIXSTATUS, PROTECTION=*PARAMETERS(MONJV-
PROTECTION=*NO)
```

6.1.4 BCAM dependencies on starting and terminating POSIX

The POSIX subsystem can only be started following the command ’BCAM READY’.

When BCAM has been restarted, the POSIX subsystem must also be shut down and restarted. Until the POSIX subsystem has been terminated, this is indicated by console message POS1040, which can be displayed using /SHOW-PENDING-MSG (/STATUS MSG).

6.2 POSIX loader

This section describes the purpose, the components and the functional scope of the POSIX loader. A short introduction is found in the chapter “Working with POSIX” (POSIX loader).

The POSIX loader administration commands *posdbl* und *pdbl* are described by way of examples. Detailed descriptions are contained in the "POSIX Commands" [1] manual.

6.2.1 Overview

The time to load a POSIX program depends mainly on the size of the program. Loading usually accounts for at least 35% (approx.) of the total processing time. For the most part this is due to the library accesses in DMS/PLAM which are run at the request of BLS. Program loading times can be considerably reduced by using DAB but only when all the libraries involved in the loading process are actually included in the process.

The POSIX loader was developed as an alternative to the DAB method. It does not use libraries but instead ready-to-run core images in the memory once loading has been completed. The core images are saved in program caches and copied to the memory for any further processing. In comparison with BLS without DAB, POSIX decreases loading times by up to 80%; in comparison with BLS with DAB, loading times are decreased by up to 65%.

The figure below shows the standardized loading times of the example program *snet* in POSIX in comparison with the other methods.

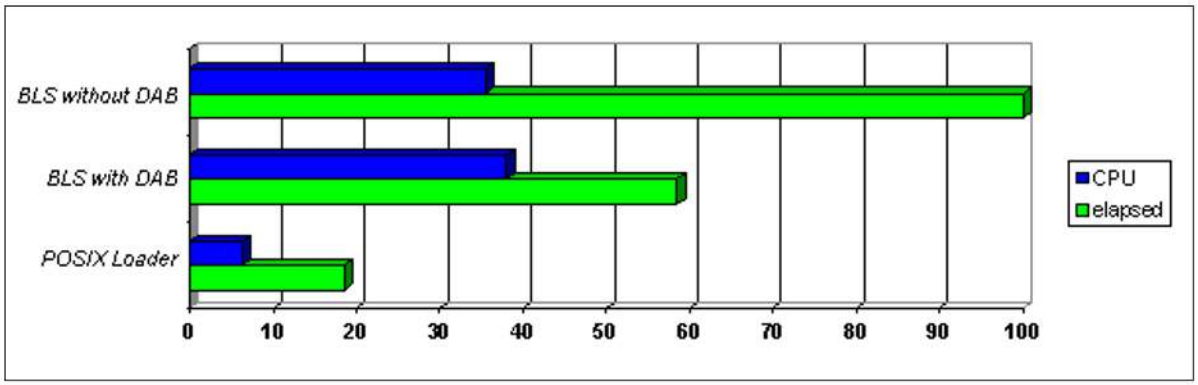

Figure 29: Loading times of the example program *snet*

The POSIX loader offers the additional advantage of being able to buffer programs of the UFS file system to specific needs, i.e. they can be temporarily stored as system global, user-specific or session-specific.

The POSIX loader is a component of the POSIX subsystem and consists of two parts:

- Privileged part

  This part can be administered by the super user only. This is a global program cache, scaled and designed to store ready-to-run core images of POSIX programs. These core images are either stored automatically into the program cache during the first call of a POSIX program from one of the defined libraries („implicit linker process“) or can be stored explicitly by a super user using the *posdbl -b* command. The global program cache is available to all users for loading a stored program.

- Non-privileged part

  This part can be used by any user. It can be used to create user-specific program caches administered by users themselves. The scope of a user-specific program cache is either

USERWIDE	all processes of an user ID are connected, or
SESSIONWIDE	all processes of a session are connected.

  Core images are saved in user-specific program cache with the aid of the *pdbl* command. The user-specific program caches are given priority in the loading of saved programs. If none of the caches contains a core image corresponding to a program, the program will be loaded in the usual way via BLS.

Use of the POSIX loader consists of the following steps:

- Initialization (setting up a program cache)

  The global program cache can be set up in two ways:

  - When the value of the DBLPOOL parameter in the POSIX information file is greater than 0, the global program cache is set up automatically with the relevant size (in MB) when the subsystem is started.
  - When DBLPOOL value is 0, the global program cache can be set up using the *posdbl* command and with the size defined beforehand in the *usp* command.

  The user-specific program caches are set up by the current user with the aid of the *pdbl* command.

- Linker process (creating a core image)

  The program is loaded and the core image is copied to the program cache with the aid of the *posdbl* command or the *pdbl* command. For the global program cache, an implicit linker process is triggered by the POSIX kernel via the system call *exec()*.

- Loader process (loading and starting a core image)

  Program caches are searched for a core image. The core image is copied and stored to memory when the system call *exec()* is run by the POSIX kernel.

- Administration (administering program caches)

  Program caches are activated and deactivated with the *posdbl* and *pdbl* commands. These commands can also be used to query the status, to list and delete core images and to delete complete program caches. User-specific program caches are resolved with the *pdbl* command.

> **i** A core image loaded from the program cache will not necessarily function correctly if it makes an attempt during the loading to dynamically reload a program segment.

6.2.2 Initialization

Setting up and activating the global program cache

The global program cache can be set up and activated in two ways:

- Automatically when the POSIX subsystem starts up, controlled by specific parameters in the POSIX information file
- Explicitly using the *posdbl* command during the ongoing POSIX session

The POSIX information file defines the following two parameters for the privileged part of the POSIX loader:

`DBLSTATE`	`initial state of POSIX loader`	*status*
`DBLPOOL`	`size of pool (MB) for POSIX loader`	*size*

A global program cache will not be set up if the *size* of the program cache is equal to zero megabytes. In this case, the starting *status* of the global program cache is ignored.

If the *size* of the program cache is greater than zero megabytes, the global program cache will be set up with the size specified. The global program cache is activated by setting the initial *status* to “1”. The global program cache is deactivated by setting the initial *status* to “0”.

The scope *Global* is used to set up a memory pool of the size specified. The upper limit value is not defined by *posdbl*, but is defined by the system-specific settings.

If the size defined using the DBLPOOL parameter when the POSIX subsystem was started is 0, the global parameter cache can also be set up again later in the following manner using the *posdbl* and *usp* commands:

- Define the size in MB using *usp* (usp –p dblpool –v *value*)
- Set up the program cache again using *posdbl* (posdbl –n)

The global parameter cache has not yet been activated when these actions have been performed.

The implicit linker and loader processes must be activated using the *-e* option of the *posdbl* command.

See also the section “Administration”.

Setting up and activating user-specific program caches

USERWIDE

In any process of a user ID *ruid* (real POSIX user identification), a program cache can be set up and activated using the command call

`pdbl -u -e` *size*

for all existing and subsequent processes of the user ID. A background process with the program name `dblu` *ruid* is created to hold the program cache.

size is the size of the program cache in megabytes. The scope *Group* sets up a memory pool of the size entered. The upper limit value is not defined by *pdbl* but is defined instead by the system-specific and task-specific settings.

SESSIONWIDE

In any process, the command call

`pdbl -s [`*sid*`] -e` *size*

sets up and automatically activates a program cache for all existing and subsequent processes in the session *sid*. If *sid* is not indicated, the current session will automatically be used.

The scope *Group* sets up a memory pool of the size entered. The upper limit value is not defined by *pdbl,* but is defined instead by the system-specific and task-specific settings. The size of the pool cannot exceed the ADDRESS-SPACE-LIMIT of the user ID.

A background process with the program name `dbls`*sid* is created to hold the program cache. If *sid* is different from the current process, the session must already exist and be active for the same user ID as that of the current process. This means that a user can only refer to his own sessions.

Example

```
$ pdbl -u -e 20     # set up program cache for all processes of the user ID

$ ps -ef
     UID   PID  PPID  C    STIME TTY      TIME CMD
   GUEST   206     1  0 16:04:01 ?        0:00 dblu101
   GUEST   204   203  0 15:59:59 pts/0    0:03 [sh]
   GUEST   207   204  0 16:04:04 pts/0    0:05 [ps]

$ pdbl -s -e 20     # set up program cache for the current session

$ ps -ef
     UID   PID  PPID  C    STIME TTY      TIME CMD
   GUEST   210     1  0 16:11:59 pts/0    0:00 dbls204
   GUEST   206     1  0 16:04:01 ?        0:00 dblu101
   GUEST   211   204  0 16:12:01 pts/0    0:01 [ps]
   GUEST   204   203  0 15:59:59 pts/0    0:03 [sh]
```

6.2.3 Linker process

Implicit linking to the global program cache

The linker process is started when the first call of a POSIX program contained in a library for which implicit linking is activated is made. The program is loaded via the BLS *BIND* interface. The program core image is analysed in the Cl.6 memory prior to the start of the program. All necessary information about the loaded program (for each slice: address, length and attributes) is returned by BLS. All slices of the loaded program are copied to the global program cache and the program is then started.

There is a number of commands which are used only relatively rarely and cannot therefore be loaded automatically into the global program cache. These are, for example, daemons or commands from the *mount*/*umount* complex.

Explicit linking to the global program cache

This is different from the implicit linker process in that the super user can copy any POSIX program into the global program cache. The command call

`posdbl -b` *path*

starts the linker process. The ready-to-run core image of the program with the path name *path* is copied into the global program cache.

Explicit linking to user-specific program cache

USERWIDE

With the command call

`pdbl -u -b` *path*

the linker process is initiated. The ready-to-run core image of the program with the path name *path* is copied into the user-specific program cache.

SESSIONWIDE

The command call

`pdbl -s [`*sid*`] -b` *path*

starts the linker process. The ready-to-run core image of the program with the path name *path* is copied into the user-specific program cache of the session *sid*. If *sid* is not indicated, then the current session will automatically be used.

The user must have execute permission for the program indicated in *path*.

6.2.4 Loader process

At each call of a POSIX program via the system call *exec(),* the following conditions will be tested in the order of priority indicated here:

- Was the task created by *fork*?
- Is debugging deactivated for the program?
- Does a program cache exist for the session or for the user, or does a global program cache already exist? Is the corresponding program core image stored in the cache?

If one of these conditions is not fulfilled, the program will be loaded and started via BLS.

If all the conditions are fulfilled, the program core image stored in the program cache selected will be copied to memory and started directly, thus bypassing BLS.

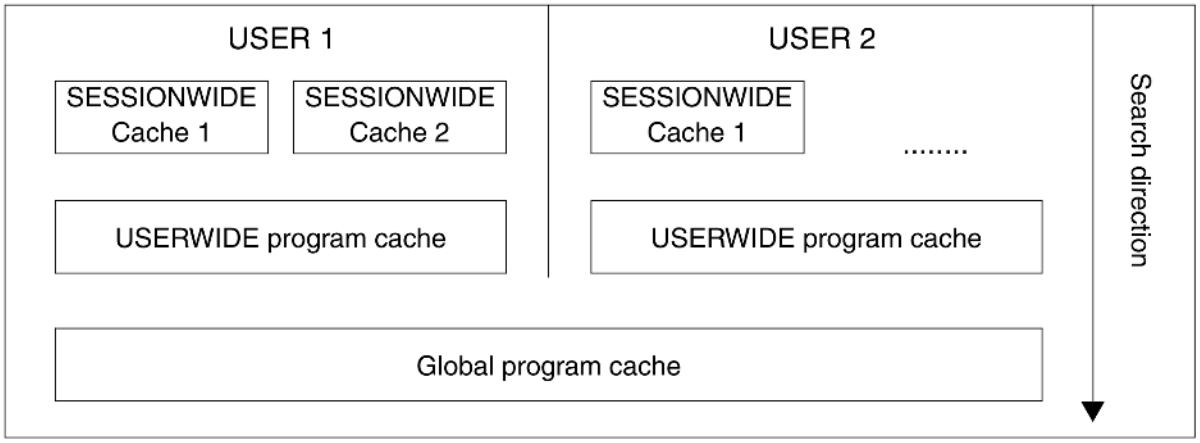

Figure 30: Program cache available for a process

Bypassing BLS will result in the incomplete integration of a POSIX-started program in the BS2000 program environment. This will impose some restrictions on the programs loaded from the program cache. These restrictions include:

- Entries defined in the program are not visible from the outside. This means that the program can dynamically load objects, but these objects cannot satisfy external links to the program.
- No loading message is generated.
- Testing with AID is not possible.
- The program name in the */STATUS* command output is missing
- The *debug* command looses its function when it is loaded from the program cache. Thus, this command should not be loaded into the program cache.

6.2.5 Administration

Activating/deactivating the global program cache

The initial status of the global program cache and the implicit linker process are defined in the POSIX information file as follows

```
DBLSTATE | initial state of POSIX loader | status
```

If *status* is set to “1”, both the implicit linker process and the loader process will be activated. If *status* is set to “0”, both processs are deactivated.

Later status changes to the implicit linker process and to the loader process can be performed together or separately, using the *posdbl* command.

- To activate/deactivate the implicit linker and loader processes together:

  ```
  posdbl {-e|-d} both
  ```

- To activate/deactivate the implicit linker process:

  ```
  posdbl {-e|-d} linker
  ```

- To activate/deactivate the implicit loader process:

  ```
  posdbl {-e|-d} loader
  ```

Explicit linker processes are always executed disregarding their status.

Activating/deactivating user-specific program caches

After setup, a user-specific program cache is activated. This means that during loader processes the program cache will be searched for appropriate core images. The user-specific program cache can be deactivated with the command *pdbl*.

- USERWIDE (deactivating user-specific program cache of the user ID):

  ```
  pdbl -u -d
  ```

- SESSIONWIDE (deactivating user-specific program cache of a session):

  ```
  pdbl -s[sid] -d
  ```

  If *sid* is not indicated, the current session will automatically be used.

If an empty program cache is deactivated, it will be deleted and must be set up again to be activated. If an occupied program cache is deactivated, it will not be taken into consideration until it is activated again during a loader process. Linker processes are always executed irrespective of their status.

Activation with the *pdbl* command is performed as follows:

- USERWIDE (activating user-specific program cache of the user ID):

  ```
  pdbl -u -a
  ```

- SESSIONWIDE (activating user-specific program cache of a session):

  ```
  pdbl -s[sid] -a
  ```

  If *sid* is not indicated, the current session will automatically be used.

Once it has been activated, the program cache will be taken into consideration during loader processes.

Querying the global program cache status

The command call

```
posdbl -s
```

outputs the status of the global program cache and the implicit linker process. It will also output statistical data about size and allocation as shown in the following example:

```
$ posdbl -s              # status output
POSIX-DBL:               linker ON       loader ON
Cache POSIX@DBL          CREATED: 01/22/09 13:06:11
                         SIZE: 24 MB     ENTRIES: 9
                         FREE PAGES: 2688
```

Querying the status of user-specific program caches

The *pdbl* command outputs information about user-specific program caches. In the following example, a 10 MB size program cache with the scope *USERWIDE* is first created. Then, the current program cache information is requested. After this, the program cache is deactivated and the updated information is requested again:

```
$ pdbl -u -e 10

$ pdbl -u -i
Cache DBLU2001           CREATED: 01/22/09 09:44:51       STATE: active
                         SIZE: 10 MB     ENTRIES: 0
                         FREE PAGES: 2559

$ pdbl -u -d

$ pdbl -u -i
pdbl: cache DBLU2001 not found
```

For a program cache with the scope *SESSIONWIDE*, current information can be queried with the command call

```
pdbl -s[sid] -i
```

If *sid* is not indicated, the current session will automatically be used.

Listing core images in the global program cache

A list of all core images stored in the global program cache is requested with the *posdbl* command. The following example shows zwei core images which were saved from the shell library to the global program cache by the implicit linker process. The third core image was explicitly saved with the command call

```
posdbl -b /opt/C/bin/snet
```

in the global program cache.

```
$ posdbl -l
 PS                 53 Jan 23 13:45:08 $TSOS.SINLIB.POSIX-BC.090.SHELL
 SH                243 Jan 28 13:15:23 $TSOS.SINLIB.POSIX-BC.090.SHELL
+IN@RLOGIND        113 Jan 28 13:15:17 $TSOS.SINLIB.POSIX-BC.090.ROOT
```

Detailed information about a core image can be requested with the command call

`posdbl -l` *element*

Element is the name of the core image as indicated in the listing.

```
$ posdbl -l IN@RLOGIND
IN@RLOGIND      CREATED : 01/22/09 11:56:00     ACCESS: 01/28/09 13:15:17
                START AT: 0x01003CA0            CACHESIZE: 452 kB
                                                USECOUNT: 12
                ------------------------------------------------------
                SLICES  : 1     LOADADDR:       SIZE:
                                0x01000000      452 kB
                ------------------------------------------------------
                Loaded by command from:
                $TSOS.SINLIB.POSIX-BC.090.ROOT
$
```

Listing core images in user-specific program caches

A list of all core images stored in user-specific program cache is requested with the *pdbl* command.

USERWIDE `pdbl -u -l`

SESSIONWIDE `pdbl -s[`*sid*`] -l`

If *sid* is not indicated, the current session is used automatically.

In the following example, for session *541* a 10 MB size program cache is set up. The core image of the program *snet* is stored to memory, the status information and subsequently, a list of stored core images is requested:

```
$ pdbl -s 541 -e 10
$ pdbl -s 541 -b /opt/C/bin/snet
$ pdbl -s 541 -i
Cache     DBLS90        CREATED: 01/23/09 13:12:32      STATE: active
                        SIZE: 10 MB     ENTRIES: 1
                        FREE PAGES: 153
$ pdbl -s 541 -l
SNET          2406 Jul 26 13:13:26 $TSOS.SINLIB.SNET.010
```

In order to request detailed information about a core image in a user-specific program cache, use the *pdbl* command:

USERWIDE `pdbl -u -l` *element*

SESSIONWIDE `pdbl -s[`*sid*`] -l` *element*

If *sid* is not indicated, the current session is used automatically.

Element is the name of the core image as indicated in the listing.

```
$ pdbl -s 541 -l SNET
SNET            CREATED : 01/23/09 16:43:41     ACCESS: 01/23/09 17:29:12
                START AT: 0x01000048            CACHESIZE: 9624 kB
                                                USECOUNT: 2
-------------------------------------------------------------------------

                SLICES  : 2     LOADADDR:       SIZE:
                                0x01000000      4272 kB
                                0x01500000      5352 kB
-----------------------------------------------------------

Loaded from:
$TSOS.SINLIB.SNET.010
```

Listing defined libraries for the implicit link operation in the global program cache

The libraries for which the implicit link operation is activated are listed using the *posdbl* command:

```
# posdbl -L
$TSOS.SINLIB.POSIX-SH.080
$TSOS.SINLIB.POSIX-BC.090.SHELL
```

Activating the implicit link operation for a library in the global program cache

If the implicit link operation has been activated for a library, all the programs contained in this library will be loaded automatically into the program cache when they are executed.

The implicit link operation for an additional library is activated using the *posdbl* command:

```
# posdbl -A \$TSOS.SINLIB.POSIX-BC.090.ROOT

$TSOS.SINLIB.POSIX-BC.090.ROOT: Successfully added.
```

Deactivating the implicit link operation for a library in the global program cache

The implicit link operation for a library is deactivated using the *posdbl* command:

```
# posdbl -R \$TSOS.SINLIB.POSIX-BC.090.ROOT

$TSOS.SINLIB.POSIX-BC.090.ROOT: Successfully deleted.
```

Deleting core images in the global program cache

Core images in global program cache are either singly or collectively deleted with the *posdbl* command:

`posdbl -r` *element*

During a single deletion, *element* is the name of the core image as given in the list output with the *posdbl -l* command. All core images are deleted if an asterisk (*) is indicated as *element*.

Deleting core images in user-specific program caches

Core images in user-specific program caches can be deleted either singly or collectively with the *pdbl* command:

USERWIDE	`pdbl -u -r` *element*
SESSIONWIDE	`pdbl -s[`*sid*`] -r` *element*

If *sid* is not indicated, the current session is used automatically.

When deleting single images, *element* is the name of the core image as indicated in the list of core images. If *element* is given as an asterisk (*), all core images will be deleted.

Modifying the size of the global program cache

When required, the size of the program cache can be modified during ongoing operation.

To do this, execute the following statements:

usp -p DBLPOOL -v *wert*

or

usp -P DBLPOOL -v *vale* (1)

posdbl -S >*scriptname* (2)

posdbl -D (3)

posdbl -n (4)

scriptname (5)

posdbl -e linker | **-e loader** | **both** (6)

(1) The new size of the program cache is defined. When the *-p* option is specified, the old size is set again after the POSIX subsystem has restarted. When *-P* is specified, the the new size applies event after a restart.

(2) The *scriptname* script is generated and can be used to restore the current content of the program cache.

(3) The current program cache is deleted.

(4) A new program cache is created with the size specified in step (1).

(5) The *scriptname* script is called in order to restore the previous content of the program cache.

(6) When the program cache is created with posdbl -n, the linker and loader are deactivated and must be reactivated if required.

> ¡ If the program cache is **reduced** using the *usp* statement in step (1), it may no longer be possible to restore the original content fully by calling the script in step (5).

Deleting the global program cache

The global program cache is maintained throughout the runtime of the POSIX subsystem. The global program cache is deleted when the POSIX subsystem terminates.

Deleting user-specific program caches

To delete user-specific program caches, use the *pdbl* command:

USERWIDE `pdbl -u -d`

If the user-specific program cache of the user ID is empty, it will be deleted. If it is not empty, it will be deactivated.

`pdbl -u -D`

Unconditionally deletes the user-specific program cache of the user ID.

SESSIONWIDE `pdbl -s[`*sid*`] -d`

If the user-specific program cache of the session *sid* is empty, it will be deleted. If it is not empty, it will be deactivated.

`pdbl -s[`*sid*`] -D`

Unconditionally deletes the user-specific program cache of the user ID. If *sid* is not indicated, the current session is used automatically.

7 Administering and monitoring file systems

This chapter is intended for BS2000 and POSIX file administrators. It contains useful information on

- administering file systems (creating, deleting, mounting, unmounting)
- monitoring file systems with *fsmond* (file system monitor daemon)

7.1 Administering file systems

The following table shows the privileges necessary for the POSIX administration tasks and the dedicated programs..

Task	Privilege	Command etc.	Input in
mounting and unmounting POSIX file systems	root privilege	mount, mountall; umount, umountall	POSIX shell
creating, changing and deleting POSIX file systems	TSOS with root privilege	POSIX installation tool	BS2000

Two or more POSIX file systems together can form a file tree. During a POSIX session, at least two file systems are always mounted - the root file system and the */var* file system (see section “Initial installation of POSIX”).

The root file system has the highest hierarchy in the file tree. During installation, the BS2000 system administrator must specify which POSIX file system is to be the root file system. The root file system is opened automatically when the POSIX subsystem is started.

7.1.1 Mounting and unmounting file systems

A file tree can be expanded by connecting additional file systems to the root file system. This operation is referred to as “mounting”. Every directory in the file tree, with the exception of the root directory, can be selected as the mount point. Mounting takes place

- automatically when POSIX starts if the file system is defined in the */etc/vfstab* file or in “Administrate POSIX file systems” with the *automount=yes* attribute or
- explicitly with the *mount* command. Only POSIX administrators can mount POSIX file systems.

When you mount file systems, please take the following into consideration:

- While you are mounting file systems using the POSIX installation tool, no *mount* commands should be issued in the shell.
- The original contents of the directories which are used as the mount point are “hidden” (files, subdirectories). Consequently the */usr* directory should not be used as the mount point.

POSIX administrators can unmount mounted file systems again using the *umount* command.

7.1.2 Administering local POSIX file systems

Local POSIX file systems can be installed, modified and deleted with the POSIX installation program. Users with the TSOS privilege plus the root authorization can perform these operations with the *Administrate POSIX filesystems* subfunction (see "Administer POSIX filesystems").

Local POSIX file systems registered with POSIX in the POSIX installation program can be mounted and unmounted by users with root authorization. The POSIX commands for mounting are *mount* and *mountall*. The POSIX commands for unmounting are *umount* and *umountall.* These commands are described in the "POSIX Commands" [1] manual.

7.1.3 Administering bs2fs file systems

The BS2000 file system *bs2fs* permits direct and transparent access to BS2000 files under POSIX. Consequently both “simple” DMS files and PLAM library elements under POSIX can be edited as if they were POSIX files.

Administration of bs2fs file systems comprises the following tasks:

- Creating the bs2fs container using the POSIX installation program
- Mounting and unmounting the bs2fs container
- Mounting and unmounting bs2fs file systems
- Modifying administrative files when the lists of the resources which are to be provided automatically or mounted are to be updated

You can use the relevant versions of the *mount*, *mountall*, *umount* and *umountall* commands or the */etc/vfstab* file to mount and unmount the bs2fs container and bs2fs file systems. The *show_pubset_export* and *start_bs2fsd* commands also support administration of bs2fs file systems.

For further information, please refer to the manual "POSIX BS2000 filesystem bs2fs" [2].

7.1.4 Administering distributed file systems

Distributed file systems can be administered in a heterogeneous network with the NFS software product. The term “distributed file systems” means:

- You can make local data shareable for processing on remote computers. You can select any parts of the hierarchy of the POSIX file system and make them shareable for remote users. Note, however, that the parts made available in this way must not overlap. The commands for making data shareable and undoing this attribute are *share*, *shareall*, *unshare* and *unshareall.* The file */etc/dfstab* can also be used for this purpose.
- When remote users make data shareable, you can mount the files in the POSIX file system on the local computer for processing. The user is not aware that the file system mounted in this way is physically located on a remote computer. The user can work with the files in this file system as if they were located in the local POSIX file system.

  You can use the NFS-specific versions of the *mount*, *mountall*, *umount* and *umountall* commands or the file */etc/vfstab* to mount and unmount files made shareable on remote computers.

See the "NFS" [8] manual for more information.

7.1.5 Checking the consistency of the file system

Use the POSIX command *fsck* to check the consistency of a file system. Inconsistencies can be corrected in an interactive dialog.

The POSIX command *fsck* is described in detail in the "POSIX Commands" [1] manual. The consistency check of particular or all file systems can also be forced when starting the POSIX subsystem, see section “Starting the POSIX subsystem”.

7.1.6 Expanding the file system

POSIX offers a space-saving direct method in which the file system is expanded at BS2000 level (BS2000 containers) without copying it beforehand. The internal structures of the POSIX file system are subsequently adapted to the new size by POSIX. This has the advantage that only the additional disk storage space is required for the new file system.

This file system expansion can take place online and offline in interactive or batch mode. If a file system is expanded online, the file system is mounted regardless of the automount setting provided it was already mounted beforehand .

The following options for expanding file systems are avaialble to you:

- POSIX installation program in interactive mode, offline (POSIX not started)

  This variant enables all POSIX file systems to be expanded, also the *root* and *var* file systems. Proceed as follows:

  - Call the POSIX installation program with `/START-POSIX-INSTALLATION`.
  - In the start mask, select the function *Expand of POSIX filesystem*.
  - In the follow-up mask enter the required file system and the new values, see section “Expand POSIX filesystems”.
- POSIX installation program in interactive mode, online (POSIX started)

  This variant enables you to expand all POSIX file systems except *root*and *var*, provided the file system can be unmounted. Proceed as follows:

  - Call the POSIX installation program with `/START-POSIX-INSTALLATION`.
  - In the start mask, select the function *Administrate POSIX filesystems*.
  - In the *Administrate POSIX filesystems* mask select the required file system, and in the input field select the E (expand) option, see section “Administer POSIX filesystems”.
  - In the follow-up mask enter the new values, see section “Expand POSIX Filesystems”.

- POSIX installation program in batch mode, offline or online

  With the offline call all POSIX file systems can be expanded, with the online call only the unmountable file systems; *root* and *var* cannot be expanded online as they cannot be unmounted when POSIX is running.

  Proceed as follows:

  - Create a parameter file with the identification line *[ExpandFileSystem],* see section “Expand POSIX file systems”.
  - Call the POSIX installation program with `/START-POSIX-INSTALLATION INPUT-INTERFACE=*FILE (<parameter-file>`

  Example

```
#
# Batch installation file
#
[ExpandFileSystem]
# <file>;<size>
$SYSROOT.FS.ROOT;10000    #  root file system 10 000 PAM pages more
$SYSROOT.FS.VAR;20000     #  var file system 20 000 PAM pages more
```

- *fsexpand* command

  This enables all file systems except *root* and *var* to be expanded.

  - Log onto the POSIX shell.
  - Enter the *fsexpand* command with the required options. The following syntax applies, see the POSIX “ Commands (Related publications)” manual [1]:

    fsexpand[**-i**][**-p** pam-pages| **-c** cylinder-groups] device

    For *device* enter the file system which is to be expanded.

7.2 File system monitoring with fsmond (file system monitor daemon)

During boot up of the POSIX subsystem, the file system monitor daemon *fsmond* is automatically started via rc scripts and automatically ended at the shut down of the POSIX subsystem (see below). It monitors the critical file systems */* (root) and */var*.

The allocation percentage of these files systems is checked regularly every *interval* seconds by the daemon. If the threshold *warnlimit* value is exceeded, the warning *POS4030* will be output on the console. If the threshold *errorlimit* value is exceeded, the acknowledge message *POS4031* will be output to the console.

Syntax

```
fsmond[ -t interval][ -w warnlimit][ -e errorlimit]
```

-t interval

This option is used to set the length of the file system test period in seconds. The permitted value is between 1 and 3600. Default value is 120.

-w warnlimit

This option indicates the file system allocation percentage at which the warning message is to be output to the console. A warning message is output to the console if this value worsens during the next run of *fsmond*. The permitted value is between 1 and 99. Default value is 80.

-e errorlimit

This option indicates the file system allocation percentage at which an acknowledge message is output to the console. The permitted value is between 1 and 99. Default value is 90.

> The value of *errorlimit* must exceed the value of *warnlimit* by at least 5.

Note

Allocation percentage

The allocation percentage for *warnlimit* and *errorlimit* (given in percentages) corresponds to the filling degree as output by the command call `df -v` (ratio of available to allocated blocks for privileged users).

rc scripts

The rc scripts for the automatic start/end of the daemon are as follows:
/etc/rc2.d/S14fsmond (starting)
/etc/rc0.d/K14fsmond (terminating)

Editing default values

If you want to run the daemon with values other than the default values, you must edit the daemon call in the rc script */etc/rc2.d/S14fsmond.* Once you have edited the values, the POSIX subsystem must subsequently be terminated and restarted. If the values entered are inconsistent, the daemon will be terminated with an error message which will be output to the syslog logging file.

Test status

Examples showing tests to check if the daemon was started properly:

```
/ # /etc/rc2.d/S14fsmond status
fsmond is running (pid=25)

/ # /etc/rc2.d/S14fsmond status
fsmond is not running

/ # ps -ef |grep fsmond
ROOT    25     1  0 10:24:32 ?        00:00 [fsmond]
```

In case the daemon is not started, the error reason is normally given in the file */var/adm/syslog*.

Advice on running the daemon

Generally speaking, the daemon should be started and ended automatically via the POSIX subsystem. In this case, the daemon runs as a background process under *$SYSROOT*.

It is possible to start and end a daemon using an rc script call but you should only do this exceptionally, e.g. when running a test:

```
/etc/rc2.d/S14fsmond stop
/etc/rc2.d/S14fsmond start
```

Console messages

```
%POS4030 FSMOND: WARNING! (&00)% ALLOCATION EXCEEDS WARNLIMIT ((&01)%) ON FILE
SYSTEM "(&02)"

?POS4031 FSMOND: ATTENTION! (&00)% ALLOCATION EXCEEDS errorlimit ((&01)%) ON FILE
SYSTEM "(&02)"
```

The *POS4031* message can be answered with any entry, e.g. with *tsn*.

> The daemon will remain in the waiting position until the *POS4031* message is answered. Regular testing of other file systems will be suppressed.

8 Administering POSIX users

This chapter is intended for BS2000 system administrators, BS2000 group administrators and POSIX administrators.

Every BS2000 user is also simultaneously a POSIX user. With the exception of a BS2000 user ID with valid individual POSIX user attributes (see "Assigning POSIX user attributes") no further conditions must be fulfilled to obtain user administration access to POSIX and its interfaces.

POSIX user administration is integrated in BS2000 user administration. This chapter describes the interfaces for managing the POSIX user attributes of a BS2000 user ID. These interfaces are part of the component SRPM (**S**ystem **R**esources and **P**rivileges **M**anagement), which is implemented in the BS2000 basic configuration and in the software product SECOS. Further details on SRPM are available in the manuals "BS2000 System Adminisration" [16] and "SECOS Security Control System - Access Control" [9]. It is not necessary for the SECOS software product to be installed in order to work with POSIX.

8.1 Privileges and functions

A new privilege, the POSIX-ADMINISTRATION privilege, has been introduced for POSIX. Owners of this privilege are referred to as POSIX administrators in this manual, and they have the following tasks and rights:

- administration of the POSIX user attributes of all BS2000 user IDs on all pubsets (see "Assigning POSIX user attributes")
- administration of default values for the POSIX user attributes on all pubsets (see "Defining default values for POSIX user attributes")

The POSIX-ADMINISTRATION privilege is automatically linked to the SYSROOT system ID. This privilege cannot be withdrawn by SYSROOT.

The security administrator (SECURITY-ADMINISTRATION privilege) can also grant the POSIX-ADMINISTRATION privilege to other BS2000 user IDs, and likewise withdraw it. The SECOS software product is required for this process.

SYSROOT is the POSIX counterpart to the system administrator ID *root* in UNIX systems. SYSROOT is set up following BS2000 system startup and automatically receives the user number 0. No other user ID can be assigned to SYSROOT.

Holders of the USER-ADMINISTRATION privilege also receive authorization to administer the POSIX user attributes. In this instance, they are treated as if they were POSIX administrators.

The authorization of the group administrator of the *UNIVERSAL group is extended to include the POSIX user attributes. When administering the POSIX user attributes on the pubset managed by the user, the user is treated as if he/she has the USER-ADMINISTRATION privilege. In this case, the restrictions for group administrators within the user's hierarchy described below do not apply to the user.

Group administrators may also administer POSIX user attributes. However, the following restrictions apply:

- They cannot administer the default values for the POSIX user attributes.
- The type of POSIX user attributes which they can use depends on their authorization (ADM-AUTHORITY).
- The value range of the POSIX user attributes is restricted for group administrators.
- They can only administer the group and subgroup members for whom they are responsible.

The following table gives an overview of the responsibilities and activities associated with POSIX user administration. Note that the administrators require certain privileges. Some functions are performed on the BS2000 level, the shell level, or both.

Function/activity	**Privilege**	**Command, etc.**	**Enter in**	**See**
Show POSIX status	SUBSYSTEM-MANAGEMENT	/SHOW-POSIX-STATUS	BS2000	"SHOW-POSIX-STATUS"
Grant/withdraw the POSIX-ADMINISTRATION privilege to BS2000 user IDs	SECURITY-ADMIN.	/SET-PRIVILEGE /RESET-PRIVILEGE	BS2000	"SECOS" [9] manual

Assign POSIX user attributes	USER-ADMIN. or POSIX-ADMIN. or BS2000 group administrator (with restrictions)	/MODIFY-POSIX-USER-ATTRIBUTES /SHOW-POSIX-USER-ATTRIBUTES	BS2000	"Assigning POSIX user attributes"
Assign an individual user number to a BS200 user ID	USER-ADMIN.	/MODIFY-POSIX-USER-ATTRIBUTES	BS2000	"Allocating an individual user number to a BS2000 user ID"
Administer POSIX groups in BS2000	USER-ADMIN. or POSIX-ADMIN. or group administrator	/MODIFY-POSIX-USER-ATTRIBUTES: User attribute GROUP-NUMBER	BS2000	"Administering BS2000 and POSIX groups", "Entering new POSIX users"
Administer POSIX groups in POSIX	Root authorization	File /etc/group	POSIX shell	"Administering BS2000 and POSIX groups", "Entering new POSIX users"
Add new POSIX users	USER-ADMIN., TSOS necessary for /ADD-POSIX-USER	/ADD-USER and /ADD-POSIX-USER	BS2000	"Entering new POSIX users"
Define defaults for POSIX user attributes	USER-ADMIN. or POSIX-ADMIN. or BS2000 group administrator (with restrictions)	/MODIFY-POSIX-USER-DEFAULTS /SHOW-POSIX-USER-DEFAULTS	BS2000	"Defining default values for POSIX user attributes"
Assign access permission for users on remote computers	USER-ADMIN. or BS2000 group administrator (with restrictions)	/SET-LOGON-PROTECTION /MODIFY-LOGON-PROTECTION /SHOW-LOGON-PROTECTION	BS2000	"Defining access rights for users of remote computers"

Enter account number for system access via a remote computer	USER-ADMIN. or BS2000 group administrator (with restrictions)	/ADD-USER /MODIFY-USER-ATTRIBUTES /SHOW-USER-ATTRIBUTES	BS2000	"Entering account numbers for system access via a remote computer"
Remove POSIX users	POSIX-ADMIN.	/MODIFY-POSIX-USER-ATTRIBUTES	BS2000	"Removing POSIX users"
Remove POSIX users	Root authorization	rmdir	POSIX shell	"Removing POSIX users"
Show information on entries in the user catalog for the own user IDs Read user information in a program	STD-PROCESSING	/SHOW-USER-ATTRIBUTES /SHOW-POSIX-USER-ATTRIBUTES SRMUINF macro	BS2000	"SHOW-POSIX-USER-DEFAULTS Display default values for POSIX user attributes" "SHOW-POSIX-USER-ATTRIBUTES Display POSIX user attributes" "Reading user information by program"

8.2 Assigning POSIX user attributes

The POSIX user attributes characterize the user, set the defaults and determine the authorizations. POSIX user attributes are *user number, group number*, *comment*, *login directory* and *program*. They correspond to the entries in user catalog */etc/passwd* of a UNIX system. However, the */etc/passwd* file does not exist in POSIX.

When a BS2000 user ID is created, the POSIX user attributes are assigned default values (see "Defining default values for POSIX user attributes"). The POSIX user attributes of a BS2000 user ID can be modified by means of the /MODIFY-POSIX-USER-ATTRIBUTES command (see "MODIFY-POSIX-USER-ATTRIBUTES").

BS2000 user IDs which already exist are automatically assigned the default user number for first startup or for a version upgrade (see "Assigning POSIX user attributes"). The default user ID of an existing BS2000 user ID can be modified by means of the /MODIFY-POSIX-USER-ATTRIBUTES command (see "MODIFY-POSIX-USER-ATTRIBUTES").

The POSIX user attributes are a component of the BS2000 user entry in the BS2000 user catalog SYSSRPM.

User number

The user number determines who owns the files and directories under POSIX. In contrast to BS2000, the BS2000 user ID - or more appropriately, the login name - is of secondary importance here. For this reason, an individual user number must be allocated to every BS2000 user ID that wants to use POSIX (see section “Allocating an individual user number to a BS2000 user ID”).

The user number 0 plays a special role: in conjunction with group number 0, it grants its owner POSIX administrator authorization, hereafter referred to as root authorization in this manual. The system ID SYSROOT has root authorization by default. The TSOS system ID automatically receives root authorization during initial installation.

Group number

The group number determines the membership of a POSIX group. This POSIX group receives the access rights of the “Group” user class for all POSIX files which this user creates.
The group number can be assigned by an entry in the POSIX group directory */etc/group* (see "Administering BS2000 and POSIX groups").

Comment

A comment on the owner of the BS2000 user ID can be entered here.

Login directory

The login directory determines the absolute path name of the directory where the user is automatically placed on linking in to POSIX. This is:

- for a merged program which is called from BS2000: the first call of a POSIX interface (POSIX SVC)
- for a user of the POSIX shell: processing of the /START-POSIX-SHELL command
- for an *rlogin* call: *rlogin* processing.

Program

This POSIX user attribute refers to the name of the program which is started after the user has called the "START-POSIX-SHELL" command.

8.3 Allocating an individual user number to a BS2000 user ID

A BS2000 user ID is identified under POSIX via the user number. For this reason, a user number must be allocated to every BS2000 user ID who wants to use POSIX (see "Defining default values for POSIX user attributes"):

- Every existing BS2000 user ID is automatically assigned the default user number for a first start or for version upgrade.
- Every new BS2000 user ID receives the default user number when it is defined.

As a result, there is a large number of BS2000 user IDs which all have the same default user number.

POSIX administrators and BS2000 system administrators can determine the value of the default user number with the BS2000 command /MODIFY-POSIX-USER-DEFAULTS. They must also assign an individual user number to each BS2000 user ID in place of the default user number before POSIX can be used under this BS2000 user ID. The command for this purpose is /MODIFY-POSIX-USER-ATTRIBUTES. A warning is displayed if a user number is repeatedly assigned, except in the case of the default user number.

User numbers from 0 to 99 are reserved for privileged users (system IDs). User numbers from 100 are kept for nonprivileged users.

Different BS2000 user IDs with the same user number are mapped to the same POSIX user ID. However, the BS2000 user ID and the user number are independent of each other.

Unequivocal allocation of BS2000 user IDs and user numbers is particularly important in a computer network with UNIX systems, since consistent user identification based on user numbers is required for all computers and systems in the network.

8.4 Administering BS2000 and POSIX groups

Since group administration in POSIX and in BS2000 differs in certain fundamental features (see “Group administration”), POSIX and BS2000 groups exist independently of each other and are therefore also administered separately.

The POSIX group directory is not a component of SRPM/SECOS. Consequently, the root administrator must define and administer the POSIX groups separately in the POSIX group directory */etc/group*. The administrator is also responsible for making modifications to a BS2000 user ID (create, change group, delete) separately in the POSIX group directory */etc/group* (see "Administering BS2000 and POSIX groups").

The group number is taken from the POSIX user attributes without further checking when the user connects to POSIX. It is therefore up to the POSIX administrator and the root administrator to decide whether to match the GROUP-NUMBER attribute and the corresponding POSIX group entry in a separate action.

A BS2000 group administrator can assume the role of POSIX administrator for the members of his/her group. In order to map the BS2000 group structure to the POSIX group structure, the administrator applies the following convention:

“The group number of the POSIX group which corresponds to the BS2000 group is the same as the group number of the BS2000 group administrator.”

A BS2000 group administrator has the following permissions:

- The administrator may forward the group number to the BS2000 group members. If a higher-level group administrator takes over the group of the original administrator, he/she can only be assigned this group number.
- The administrator can exclude a BS2000 group member from the POSIX group by assigning this member the default group number.

Further administration of the POSIX groups must be performed centrally by a POSIX administrator.

Example

The BS2000 group with the group name `A5` contains the following users:
`POSIXTST, POSIX001` and `POSIX002`
The BS2000 group with the group name `A7` contains the following users:
`MANUAL01` and `MANUAL02`.

When using POSIX, groups with both group number 5 `(POSIXTST, POSIX001` and `POSIX002`) and group number 7 `(MANUAL01` and `MANUAL02`) can be defined. However, double membership of both groups - e.g. if `MANUAL01` also wishes to become a member of the group with number 5 - is only possible if the definition of the BS2000 group is altered.

Administering the POSIX group catalog */etc/group* of the POSIX shell

Every user is allocated to a user group once the BS2000 system administrator has assigned him/her a numeric group number. In the POSIX group catalog */etc/group*, the POSIX administrator or a user with root authorization can assign this group number a group name or define a new user group.
There is no equivalent to the POSIX group catalog */etc/group* in BS2000.

The POSIX group catalog */etc/group* system file is set up during initial installation. It consists of lines with the following format:

```
groupname: : groupnumber : userid[,...]
```

groupname
Name to be assigned for this group.

groupnumber
Numeric group number which was defined in the BS2000 user catalog SYSSRPM. A group name can be assigned to this group number via <groupname>.

userid
One or more user IDs which are to be included in this user group. If two or more user IDs are specified, you must separate them with a comma.
The same user ID can be used in several user groups.

The entries must be separated from each other by a colon. If you omit the entry for the password, you still have to specify the following colon. In each case, entries for every user group must begin in a new line.

The POSIX group catalog */etc/group* file contains the following user groups after the initial installation:

```
SYSROOT       (Groupnumber: 0, Member : SYSROOT)
OTHER         (Groupnumber: 1)
SYSBIN        (Groupnumber: 2)
SYSSYS        (Groupnumber: 3, Members: SYSROOT, SYSBIN)
MAIL          (Groupnumber: 6, Member : SYSROOT)
TTY           (Groupnumber: 7)
LP            (Groupnumber: 8)
USROTHER      (Groupnumber: 100)
DFS_STARTGID  (Groupnumber: 2000)
```

8.5 Entering new POSIX users

Following the creation of a new BS2000 user by means of the "ADD-USER" command, the POSIX user attributes *user number, group number*, *login directory* and *program* are assigned pubset-specific default values (see "Defining default values for POSIX user attributes"). Either the POSIX administrator or the BS2000 system administrator must modify the default values so that the new BS2000 user can use POSIX.

For this purpose a procedure is offered which is called using "ADD-POSIX-USER" command. This procedure can only run under the TSOS system ID. Before the procedure is started the POSIX subsystem must be started.

This procedure creates a home directory for the user and enters the name in the relevant POSIX user attribute.

8.6 Defining default values for POSIX user attributes

When a BS2000 user ID is created, the POSIX user attributes are initialized with the default values of the specified pubset. The operand value *BY-POSIX-USER-DEFAULTS always refers to the default values of the specified pubset.

The POSIX user attributes are automatically created during first startup or version changeover; they are initialized with predefined values:

USER-NUMBER	=	100
GROUP-NUMBER	=	1
COMMENT	=	*NONE
DIRECTORY	=	*ROOT
PROGRAM	=	*SHELL

With the exception of TSOS and SYSROOT, all BS2000 user IDs are first assigned the default user number and the default group number.

During first startup, the TSOS and SYSROOT system IDs are created with the predefined user number 0 and group number 0. The user number cannot be modified, but there are no restrictions for the group number.

The defaults for the user attributes can be changed with the command "MODIFY-POSIX-USER-DEFAULTS". You can use the command "SHOW-POSIX-USER-DEFAULTS" to view the attributes.

8.7 Defining access rights for users of remote computers

If the SECOS software product is used, it is possible to find out for existing BS2000 user IDs whether the user of a remote terminal may obtain access to the system via the *rlogin* command (see section “Accessing the POSIX shell”). The POSIX-RLOGIN-ACCESS operand is available in the BS2000 commands "SET-LOGON-PROTECTION" and "MODIFY-LOGON-PROTECTION" for this purpose.
The "SHOW-LOGON-PROTECTION" command can be used to display the protection attributes.

Classifying system access control with SECOS

With SECOS, you can classify access from a remote computer more accurately using the commands /SET-LOGON-PROTECTION and /MODIFY-LOGON-PROTECTION.

You can set the following for a user ID:

- whether and from which stations *rlogin* access is permitted (POSIX-RLOGIN-ACCESS operand).
- whether and from which stations access using remote commands (*rsh*, *rcp*, ...) is permitted (POSIX-REMOTE-ACCESS operand).
- whether access using *rlogin* or remote commands is protected with a guard. Guards can be assigned separately for *rlogin* and remote command access.
- whether tasks under this user ID may change their user ID with *ufork* .

8.8 Entering account numbers for system access via a remote computer

The commands "ADD-USER", "MODIFY-USER-ATTRIBUTES" and "SHOW-POSIX-USER-DEFAULTS" enable you to administer account numbers for system access via remote computers.

8.9 Removing POSIX users

The procedure for removing a POSIX user is as follows:

- POSIX administrator:
  Reset the individual user number to the default with the BS200 command "MODIFY-POSIX-USER-ATTRIBUTES".
- User with root authorization:
  If necessary, remove the home directory from the POSIX shell with the POSIX *rmdir* command.

  If applicable, delete the user's files or assign them to other users beforehand.

  The POSIX command *rmdir* is described in the "POSIX Commands" [1] manual.

8.10 Reading user information by program

The BS2000 system administrator creates an entry in the user catalog for every BS2000 user ID. The entry contains:

- BS2000 user ID, password authorization
- specifications concerning the system resources which the user can use (CPU time, memory space, ...)
- special user rights (privileged access, etc.)
- accounting data

Data in the user catalog can be read using the macro SRMUINF, and transferred to a previously defined area. Depending on the specification, accounting data, user-specific data, or the entire entry of a BS2000 user ID can be output from the user catalog.

Using POSIX does not cause any changes to the operands and operand values of the SRMUINF macro. However, the POSIX account number is marked. The accounting-specific part of the output contains an indicator for each individual account number which determines the number for the accounting of the remote login session.
Further information on the SRMUINF macro can be found in the "Executive Macros" [30] manual.

You can also use the CRTE macros *getlogin*, *getpwent*, *putpwent* etc. to read the data from the user catalog (see the "CRTE" [7] manual).

8.11 POSIX default job classes

Many POSIX tasks can be created using the *fork* mechanism of the POSIX subsystem. Such POSIX tasks can be controlled independently of the other batch and dialog jobs.

The JMU statement SET-POSIX-JOB-CLASS-DEFAULT (see "Utility Routines" [17] manual) enables individual or all user IDs to be assigned a default job class for POSIX tasks. This job class must have been defined beforehand using DEFINE-JOB-CLASS. If no POSIX default job class is defined, the behavior in the case of *fork* remains unchanged.

Example:

```
/START-JMU
//SET-MODIFICATION-MODE ...
//SET-POSIX-JOB-CLASS-DEFAULT -
//    NAME=JCBPSX1, ACTION=*ADD, USER=*ALL ————————————————————————  (1)
//SET-POSIX-JOB-CLASS-DEFAULT -
//    NAME=JCBPSX0, ACTION=*ADD, USER=(TSOS,SYSROOT) ——————————————  (2)
//END
```

(1) A POSIX default job class which is valid system-wide for all user IDs is defined.

(2) A different POSIX default job class is defined for the user IDs TSOS and SYSROOT.

With the conventional way of defining default job classes for BATCH or DIALOG using the JMU statement SET-JOB-CLASS-DEFAULT, a job class becomes the default job class either for batch or dialog tasks according to its definition (JOB-TYPE=). This distinction is not made for POSIX default job classes; the type of job class is arbitrary. A POSIX task generated by *fork* is assigned the task type of the parent task and the category of the job class in which it runs, i.e. the POSIX default job class.

If the parent task is not running in a default job class (POSIX, DIALOG or BATCH) when the *fork* takes place and the child task has the same user ID as the parent task, the child task is not assigned the POSIX default job class, but the job class of the parent task.

The POSIX default job classes apply only for the POSIX tasks generated by *fork*. POSIX programs which are started in batch or dialog tasks, e.g. with START-PROGRAM, run in the job class of this batch or dialog task. This applies in particular for POSIX access using the START-POSIX-SHELL command.

9 BS2000 commands for POSIX

All BS2000 commands for POSIX are listed in this chapter and the job variable $.SYS.POSIXSTATUS is described.

The command syntax and description of return codes used below corresponds to the conventional representation for SDF commands, see the "BS2000 Commands" [28] manual.

9.1 ADD-POSIX-USER

Define POSIX attributes for user ID

Domain:	SYSTEM-MANAGEMENT
Privileges:	TSOS USER-ADMINISTRATION

This command defines the attributes which a user ID must have to be used with POSIX.

Format

ADD-POSIX-USER
USER-NAME = <name 1..8>
,USER-NUMBER = ***DEFAULT** / <integer 0..60002>
,GROUP-**NUM**BER = ***DEFAULT** / <integer 0..60002>
,**PROG**RAM = ***DEFAULT** / <posix-pathname 1..1023 without-wild>
,**HOME-DIR**ECTORY = ***DEFAULT** / <posix-pathname 1..1023 without-wild>
,**RLOGIN-ACCOUNT** = ***NONE** / <alphanum-name 1..8>

Operands

USER-NAME = <name 1..8>

BS2000 user ID whose POSIX user attributes are to be defined.

USER-NUMBER =
User number to be defined for the user ID.

USER-NUMBER = *DEFAULT
The user number is given the currently set default value (see the "SHOW-POSIX-USER-DEFAULTS" command).

USER-NUMBER = <integer 0..60002>
The user number is given the specified value.

GROUP-NUMBER =
Group number to be defined for the BS2000 user ID.

GROUP-NUMBER = *DEFAULT
The group number is given the currently set default value (see the "SHOW-POSIX-USER-DEFAULTS" command).

GROUP-NUMBER = <integer 1..60002>
The group number is given the specified value .

PROGRAM =
Program that is started following the command *rlogin* or after the command START-POSIX-SHELL has been called.

PROGRAM = *DEFAULT
The program to be started is determined on the basis of the currently set default value (see the "SHOW-POSIX-USER-DEFAULTS" command).

PROGRAM = <posix-pathname 1..1023 without-wild>
The specified program is started.

HOME-DIRECTORY =
Specifies the absolute path name of the directory to which a user is automatically taken when they are connected to POSIX.

If the directory does not yet exist it is created and the owner is set to the user number and the group number of the POSIS user ID.

If the directory already exists its attributes remain unchanged and a respective message is output (POS2907 THE HOME DIRECTORY EXISTS; ITS ATTRIBUTES ARE NOT CHANGED).

HOME-DIRECTORY = *DEFAULT
The directory determined on the basis of the currently set default value (see the "SHOW-POSIX-USER-DEFAULTS" command).

HOME-DIRECTORY = <posix-pathname 1..1023 without-wild>
Specifies the directory.

RLOGIN-ACCOUNT =
Account number for using POSIX over a remote login and over NFS.

RLOGIN-ACCOUNT = *NONE
The account number defined with "ADD-USER" or "MODIFY-USER-ATTRIBUTES" for access over a remote login remains unchanged.

RLOGIN-ACCOUNT = <alphanum-name 1..8>
The account number specified is used for access over a remote login.

The account number is relevant for user IDs which wish to have remote access to POSIX (*rlogin* or Telnet access, *rsh* and *rcp* commands) or to use *at*, *crontab* or *batch*.

9.2 ADD-USER

Create user entry in user catalog

Domain:	USER-ADMINISTRATION
Privileges:	USER-ADMINISTRATION STD-PROCESSING

This command creates an entry in the user catalog of a pubset.

The POSIX user attributes are implicitly initialized with default values. Furthermore, the BS2000 system administrator must also define an account number for a remote login session.

The following users may execute this command:

- Owners of the USER-ADMINISTRATION privilege for all BS2000 user IDs.
- Group administrators who have at least the MANAGE-MEMBERS attribute for the BS2000 user IDs assigned to and subordinate to their groups.

Only the part of the command that is relevant to POSIX is shown in the syntax diagram on the next page. The command is described in full in the manuals "SECOS - Access Control" [9] and "BS2000 Commands" [28].

Format

ADD-USER

```
. . .
,ACCOUNT-ATTRIBUTES = *PARAMETERS(...)
    *PARAMETERS(...)
        | ACCOUNT = <alphanum-name 1..8>
        | . . .
        | ,POSIX-RLOGIN-DEFAULT = *NO / *YES
```

Operands

ACCOUNT-ATTRIBUTES = *PARAMETERS(...)

Determines the accounting data of a BS2000 user ID.

. . .

POSIX-RLOGIN-DEFAULT =
Determines whether the specified account number is used for accounting for a remote login session.

POSIX-RLOGIN-DEFAULT = *NO
The account number is not used for accounting.

POSIX-RLOGIN-DEFAULT = *YES
The specified account number is used for accounting.

This entry is required for user IDs which wish to have remote access to POSIX (*rlogin* or Telnet access, *rsh* and *rcp* commands) or to use *at*, *crontab* or *batch*.

Example

The PSXROOT user ID is created with the account number PSXACC.

```
/ADD-USER USER-ID=PSXROOT, -

/ ACCOUNT-ATTR=*PAR(ACCOUNT=PSXACC,POSIX-RLOGIN-DEFAULT=*YES)
```

9.3 COPY-POSIX-FILE

Copy files from BS2000 to the POSIX file system (and vice versa)

Domain: FILE

Privileges: all privileges

The BS2000 command /COPY-POSIX-FILE has the same functional scope as the *bs2cp* shell command plus some enhancements. The software required for the processing of this command is an installed version of SDF-P-BASYS or SDF-P version V2.2 or higher.

Hints for technical realization: The /COPY-POSIX-FILE command parameters are mapped to the parameters of the *bs2cp* shell command. After the POSIX shell is accessed (with /START-SHELL), the *bs2cp* command is called. This means that the calling user must have a HOME directory in POSIX in order to use the command /COPY-POSIX-FILE and that the copying process is affected by the settings in the *.profile* of this HOME directory (see “Control of the copying process via the .profile file”).

The command makes it possible to copy files or PLAM library elements from BS2000 into the POSIX file system (as plain files, not as elements of ar libraries) and also to copy POSIX files into BS2000. The source files can be entered explicitly or with wildcard syntax. With POSIX files, the POSIX wildcard syntax is supported. With BS2000 files, however, only a limited BS2000 wildcard syntax (only “*”) is supported. With PLAM elements, the PLAM wildcard syntax is supported.

If only a single file is copied, you can enter the name of the target file explicitly. If you are copying several files, the names of the target files are explicitly derived from the names of the source files (see parameter *BY-SOURCE). If several source files are entered, then the target files are given the same names as the source files and the target files may take a suffix and a prefix. You should exercise caution when copying from POSIX to BS2000 if the POSIX file has an exotic name which is permitted in POSIX but is not supported in BS2000.

You can prevent the accidental overwriting of existing BS2000 files or PLAM library elements by using the WRITE-MODE parameter.

The conversion of file contents can be controlled through the CHARACTER-CONVERSION parameter (analogous to the options *-k* and *-t* of the shell command *bs2cp*).

Control of the message output is also possible.

Two other commands are closely related to the shell command *bs2cp*: these are *bs2file* and *ftyp*.

The *bs2file* command generates a FILE macro in BS2000 from the shell, thereby creating a file in BS2000 which has the required properties. Here, the parameter FILE-ATTRIBUTES has been introduced which is supposed to carry the parameters of the shell command *bs2file*.

The command *ftyp* is used during copying of text or binary files from BS2000 to POSIX (and vice versa) to indicate whether the files should be treated as text files or binary files. Hence, the parameter RECORD-CONVERSION is introduced which can take the corresponding values of the shell command *ftyp* (see description in "POSIX Commands" [1] manual). The parameter is only valid for the current SDF command call.

BS2CP and CPXF are alias names.

bs2file command support via the FILE-ATTRIBUTES parameter

The *bs2file* command can be issued in the shell before the *bs2cp* command. This makes it possible to define the type of BS2000 file because the string given with *bs2file* implicitly issues a FILE macro when *bs2cp* is called. To enable this, the parameter values are written to the *.bs2cp* file. The FILE-ATTRIBUTES parameter controls if (and with which parameters) the *bs2file* command is to be issued before the *bs2cp* command.

This parameter only needs to be entered during copying to BS2000 (*FROM-POSIX) and is only required if plain files and not PLAM libraries are to be processed.

Handling of redirections

If a redirection of SYSOUT to a file is performed before the COPY-POSIX-FILE call, you should note the following points to ensure meaningful processing:

- If the command is running in dialog mode, the redirection of SYSOUT to a file should be done with the following parameters of the ASSIGN-SYSOUT command:

  ```
  /ASSIGN-SYSOUT TO = <file >, TERMINAL-DISPLAY = *YES
  ```

  This will ensure that all messages described in the file will also be output to a terminal. This is especially important for messages which require an acknowledgement.
- If the default value

  ```
  TERMINAL DISPLAY = *NO
  ```

  is used instead, no prompt, but an asterisk (*) will appear on the screen which requires an input. This should be avoided for obvious reasons.

  This does not affect the WRITE-MODE parameter which defines if the file is to be overwritten as a new file or if the data is to be appended at the end of the file. To achieve the desired effect, the parameter must be set accordingly.

For the call of this command in a batch procedure, these precautions are not necessary.

EXIT value during incorrect processing of a COPY-POSIX-FILE

The processing of the /COPY-POSIX-FILE command is performed in three steps :

1. First, SDF syntax check is activated. This checks for any violations of syntax rules. If a violation is found, SDF outputs an error message and command processing is aborted.
2. Next, an SDF procedure with the syntactically correct parameters is called. This step consists of a semantic check. If the check is successful, the *bs2cp* shell command with the currently specified parameters is generated. In the event of an error (the syntax allows the explicit entry of two POSIX files which can be copied to three explicitly entered BS2000 files), the procedure will be terminated with an error code. This also applies if the loaded version of SDF does not match or if *bs2cp* cannot be started (see the error messages POS6010 through POS6019).
3. When the shell command *bs2cp* ... is executed, errors during the processing of /COPY-POSIX-FILE may still occur. If an error does occur, *bs2cp* will be ended with an exit value not equal to zero which is output in the message POS6020. If several files are being copied during the *bs2cp* call and an error occurs during the copying of one file (exit value not equal to zero), the copying continues with the remaining files.

If a *bs2file* command was issued, it is treated as follows:

- If a *bs2file* command is issued, the parameters entered previously will be checked for correctness by generating a FILE macro with *DUMMY. In the event of an error, a message is output and processing is terminated.

- If, in certain cases, a *bs2file* command is issued before the *bs2cp* call, the command's EXIT status will be saved and output later.

Control of the copying process via the .profile file

During the use of the SDF command COPY-POSIX-FILE, the actual copying is done in the shell with *bs2cp*. Hence, the settings in the *.profile* file of the calling user will also affect the copying process. Examples of relevant settings are:

- Changing the current directory (*cd* command). By default, the current directory is the home directory of the calling user.
- Defining the *bs2cp*-specific environment variables *IO_CONVERSION* and *BS2CPTABS*.

Format

COPY-POSIX-FILE

Alias: **CPXF, BS2CP**

COPY-DIRECTION = ***FROM-POSIX** / ***TO-POSIX**

,POSIX-FILE = ***BY-SOURCE**(...) / <list-poss(2000): posix-pathname 1..1023>

***BY-SOURCE**(...)

| POSIX-DIRECTORY = **.** / <posix-pathname 1..1023 without-wild>

| ,PREFIX = ***NONE** / <c-string 0..80 with-low>

| ,SUFFIX = ***NONE** / <c-string 0..80 with-low>

,BS2000-FILE = ***BY-SOURCE**(...) / ***LIB**RARY**-ELEM**ENT(...) / list-poss(2000): <filename 1..80 with-wild>

***BY-SOURCE**(...)

| PREFIX = ***NONE** / <c-string 0..53>

| ,SUFFIX = ***NONE** / <c-string 0..40>

***LIB**RARY**-ELE**MENT(...)

| LIBRARY = <filename 1..54>

| ,ELEMENT = ***BY-SOURCE**(...)/ list-poss(2000): <composed-name 1..64 with-under-wild>(...)

| ***BY-SOURCE**(...)

| | VERSION = ***HIGH**EST-**EXIST**ING / ***UPPER-LIMIT** / <composed-name 1..24 with-under>

| | ,PREFIX = ***NONE** / <c-string 0..63>

| | ,SUFFIX = ***NONE** / <c-string 0..63>

| <composed-name 1..64 with-under-wild>(...)

| | VERSION = ***HIGH**EST-**EXIST**ING / ***UPPER-LIMIT** / <composed-name 1..24 with-under>

```
        |   ,TYPE = *S / *D / *J / *M / *P / *X / *L
,WRITE-MODE = *BY-DIALOG / *REPLACE / *CREATE
,CHARACTER-CONVERSION = *NO / *YES(...)
    *YES(...)
        |   TABLE = *STD / <posix-pathname 1..1023 without-wild>
,OUTPUT = *NONE / *SYSOUT
,RECORD-CONVERSION = *TEXT(...) / *BINARY
    *TEXT(...)
        |   SUBSTITUTE-TABULATOR = *YES / *NO
,FILE-ATTRIBUTES = *STD / *PARAMETER(...)
    *PARAMETER(...)
        |   FILE-NAME = *ALL / <filename 1..54>
        |   ,ATTRIBUTES = *STD / <c-string 0..1000>
```

Operands

COPY-DIRECTION =

Direction of copying process.

COPY-DIRECTION = *FROM-POSIX
POSIX files are copied to BS2000.

COPY-DIRECTION = *TO-POSIX
BS2000 files or PLAM elements are copied to POSIX.

POSIX-FILE =
Specifies the POSIX files to be used during copying.

POSIX-FILE = *BY-SOURCE
The names of the POSIX files are derived from the BS2000 names.

This is a mandatory entry when the copying direction is *TO-POSIX and more than one BS2000 file is to be copied.

This is an alternative entry to *posix-pathname* when the copying direction is *TO-POSIX, only one BS2000 file is to be copied and the name of the target file is to be derived from the name of the BS2000 file.

POSIX-DIRECTORY =
Specifies the directory to which the BS2000 files or the PLAM elements are to be copied.

POSIX-DIRECTORY = .
The current directory is used.

By default, this is the home directory of the calling BS2000 user. A different current directory can be set using the change directory (*cd*) command in the *.profile* file.

POSIX-DIRECTORY = <posix-pathname 1..1023 without-wild>
The explicitly entered directory is used.

PREFIX =
Specifies the prefix which precedes the POSIX file name.

PREFIX = *NONE
No prefix is used.

PREFIX = c-string 0..80 with-low>
The string entered is used as prefix.

SUFFIX =
Specifies the suffix which is attached to the POSIX file name.

SUFFIX = *NONE
No suffix is used.

SUFFIX = <c-string 0..80 with-low>
The string entered is used as suffix.

POSIX-FILE = list-poss(2000): <posix-pathname 1..1023>
The names of POSIX files are entered explicitly. Plaese note the following:

- With the copying direction *FROM-POSIX, enter one or more absolute or relative path names of POSIX files. To specify the POSIX file name, the wildcard syntax (shell special characters for file name replacement) is supported.
- With the copying direction *TO-POSIX, when only a single BS2000 file is to be copied and the name of the target file is to be explicitly specified, enter the absolute or relative path name of a POSIX file. Wildcard syntax is not allowed.
- By default, relative path names refer to the home directory of the calling BS2000 user. A different directory can be set using the change directory (*cd*) command in the *.profile* file.

BS2000-FILE =
Specifies the BS2000 files or the PLAM elements to be used during copying.

BS2000-FILE = *BY-SOURCE(...)
The names of the BS2000 files are derived from the names of the POSIX files.

This is a mandatory entry when the copying direction is *FROM-POSIX and more than one POSIX file is to be copied.

This is an alternative entry to *filename* when the copying direction is *FROM-POSIX, only one POSIX file is to be copied and the name of the target file is to be derived from the name of the POSIX file.

PREFIX =
Specifies the prefix which precedes the BS2000 file names.

PREFIX = *NONE
No prefix is used.

PREFIX = <c-string 0..53 with-low>
The string entered is used as prefix.

SUFFIX =
Specifies the suffix to be attached to the BS2000 files names.

SUFFIX = *NONE
No suffix is used.

SUFFIX = <c-string 0..40 with-low>
The indicated string is used as suffix.

BS2000-FILE = *LIBRARY-ELEMENT(...)
PLAM elements are used instead of BS2000 files during copying.

LIBRARY = <filename_1..54>
Explicit entry of the PLAM library to be used during copying.

ELEMENT =
Specifies the PLAM elements to be used during copying.

ELEMENT = *BY-SOURCE(...)
The names of the elements are derived from the POSIX files.

This is a mandatory entry when the copying direction is *FROM-POSIX and more than one POSIX file is to be copied.

This is an alternative entry to *composed-name* when the copying direction is *FROM-POSIX, only one POSIX file is to be copied and the name of the target element is to be derived from the name of the POSIX file.

VERSION =
Specifies which version of an element is to be used.

VERSION = *HIGHEST-EXISTING
The element with the highest version is used. Please note the following:

- If an element does not yet exist, it is assigned version *001*.
- If existing elements are copied, the element with the highest version is overwritten.

VERSION = *UPPER-LIMIT
The copied element is to be assigned the highest possible version (X'FF'; this corresponds to the tilde character in the *bs2cp* command).

VERSION = <composed-name 1..24 with-under>
The version is explicitly entered.

PREFIX =
Specifies the prefix which precedes the PLAM element names.

PREFIX = *NONE
A prefix is not used.

PREFIX = <c-string 0..63 with-low>
The string entered is used as prefix.

SUFFIX =
Specifies the suffix to be attached to the PLAM element name.

SUFFIX = *NONE
No suffix is used.

SUFFIX = <c-string 0..63 with-low>
The string entered is used as suffix.

ELEMENT = list-poss (2000): <composed-name 1..64 with-under-wild>(...)
The names of the elements are explicitly entered. Please note the following:

- With the copying direction *TO-POSIX, enter one or more element names. To specify element names, the LMS wildcard syntax (“*”, “<”, “:”, “>”) is supported. The specification of a list of LMS file names (list-poss) is an extension to the *bs2cp* command where only one PLAM element operand (**bs2:**) is possible. When a file name list is specified, the *bs2cp* command is called for each file name (with or without wildcards). In this case, the specifications in the remaining SDF parameters are valid for all *bs2cp* calls.
- With the copying direction *FROM-POSIX, when only one POSIX file is to be copied and the name of the target file is to be explicitly specified, enter the explicit name of an element. Wildcard syntax is not allowed.

  VERSION =
  Specifies the element version used.

  VERSION = *HIGHEST-EXISTING
  The element with the highest version is used.

  VERSION = *UPPER-LIMIT
  The copied element is to be assigned the highest possible version (X’FF’).

  VERSION = <composed-name 1..24 with-under>
  The version is entered explicitly.

TYPE = *S / *D / *J / *M / *P / *X / *L
Specifies the type of PLAM elements handled. By default, the type *S* (Source) is used.

BS2000-FILE = list-poss (2000): <filename 1..80 with-wild>
The names of the BS2000 files are entered explicitly. Please note the following:

- With the copying direction *TO-POSIX, enter one or more DVS file names. To specify the file name, the wildcard syntax for DVS files (“*”) is supported. The specification of a list of DVS file names (list-poss) is an extension to the *bs2cp* command where only one file name operand (**bs2:**) is possible. When a file name list is specified, the *bs2cp* command is called for each file name (with or without wildcards). In this case, the specifications in the remaining SDF parameters are valid for all *bs2cp* calls.
- With the copying direction *FROM-POSIX, when only one POSIX file is to be copied and the name of the target file is to be explicitly specified, enter the explicit name of a DVS file. Wildcard syntax is not allowed.

WRITE-MODE =
Specifies if BS2000 files are overwritten during the copying process. The specification of WRITE-MODE = is only relevant when the copying direction is *FROM-POSIX. It specifies whether or not existing BS2000 files (DVS files or PLAM elements) can be overwritten (this is analogous to the *-f* option in the *bs2cp* command).

WRITE-MODE = *BY-DIALOG
The user will be prompted if an existing file should be overwritten.

WRITE-MODE = *REPLACE
The dialog prompt is suppressed and existing files are always overwritten.

WRITE-MODE = *CREATE
The dialog prompt is suppressed. Already existing files are not overwritten but new files are created.

CHARACTER-CONVERSION =
Specifies if conversion should be performed during the copying process (this is analogous to the *-k* and *-t* options in the *bs2cp* command, see the POSIX “Commands (Related publications)“ manual [1]).

CHARACTER-CONVERSION = *NO
Conversion is not performed.

CHARACTER-CONVERSION = *YES(...)
Conversion is performed.

TABLE =
Specifies the conversion table.

TABLE = *STD
POSIX-internal default tables are used (this is analogous to the *-k* option in the *bs2cp* command, see the POSIX “Commands (Related publications)“ manual [1]).

TABLE = <posix-pathname 1..1023 without-wild>
The conversion table is entered explicitly (this is analogous to the *-t* option in the *bs2cp* command, see the POSIX “Commands (Related publications)“ manual [1]).

Note

The value of the *BS2CPTABS* shell variable (see *bs2cp*) is not controlled via the SDF parameter. If necessary, it can be supplied in the *.profile* file.

OUTPUT =
Specifies if the *bs2cp* extended logging is to be output (this is analogous to the *-l* option of the *bs2cp* command, see the POSIX “Commands (Related publications)“ manual [1]).

OUTPUT = *NONE
Extended logging will not be output.

OUTPUT = *SYSOUT
Extended logging will be output to SYSOUT.

RECORD-CONVERSION =
Specifies how the contents of BS2000 files are to be treated during copying.

This parameter generates the *ftyp* shell command with corresponding parameters. If the parameter RECORD-CONVERSION is not explicitly specified, *ftyp text* is set by default.

RECORD-CONVERSION = *TEXT(...)
SAM files are treated as text files. The “newline” character is converted to “new record”.

SUBSTITUTE-TABULATOR =
Specifies how tabulator characters are to be treated.

SUBSTITUTE-TABULATOR = *YES
Tabulator characters are to be filled accordingly (*ftyp text*).

SUBSTITUTE-TABULATOR = *NO
Tabulator characters are to be preserved (*ftyp textbin*).

RECORD-CONVERSION = *BINARY
SAM files are treated as binary files.

FILE-ATTRIBUTES =
When copying POSIX files to BS2000 as DVS files (not as PLAM elements), the file attributes are specified according to the shell command *bs2file*. Depending on the parameter entered, a *bs2file* command is issued in the shell before the actual *bs2cp* command call.

FILE-ATTRIBUTES = *STD
No *bs2file* shell command is issued during copying.

DVS files not yet existing are assigned the standard attributes FCBTYPE=SAM, RECFORM=V and BLKSIZE=STD. If already existing DVS files are overwritten, their file properties are kept. If only one catalog entry exists for a DVS file (without FCBTYPE, RECFORM, BLKSIZE), the file is assigned the standard attributes of the operating system (FCBTYPE=ISAM).

FILE-ATTRIBUTES = *PARAMETER(...)
During copying, the file attributes are specified in the same way as for the shell command *bs2file*.

FILE-NAME =
Specifies the files for which the attributes are to be set.

FILE-NAME = *ALL
The attributes entered are valid for all files to be copied with an arbitrary name (corresponds to the “*” entry in the *bs2file* command).

FILE-NAME = <filename 1..54>
The attributes indicated are valid for the file with the specified name.

ATTRIBUTES =
Specifies the attributes in the (ISP-)FILE command format.

ATTRIBUTES = *STD
Standard attribute with the values: FCBTYPE = SAM, RECFORM = V, BLKSIZE = STD.

ATTRIBUTES = <c-string_1..1000>
Explicitly entry of attributes.

The supported file attributes are listed in the *bs2cp* shell command description (see the POSIX “Commands (Related publications)“ manual [1], section “Attributes supported by DVS files”).

Example

```
ATTRIBUTES='FCBTYPE=SAM,RECFORM=F,BLKSIZE=80'
```

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No errors
	64	POS6010	Invalid combination of the specifications BS2000-FILE=*BY-SOURCE and COPY-DIRECTION=*TO-POSIX
	64	POS6011	Invalid combination of the specifications BS2000-FILE=*LIBRARY-ELEMENT(..., ELEMENT=*BY-SOURCE,...) and COPY-DIRECTION=*TO-POSIX
	64	POS6012	Invalid combination of the specifications POSIX-FILE=*BY-SOURCE and COPY-DIRECTION=*FROM-POSIX
	64	POS6013	More than one BS2000 file specified as target.
	64	POS6014	More than one PLAM library element specified as target.
	64	POS6015	More than one POSIX file specified as target.
	64	POS6016	More than one BS2000 file or PLAM library element specified as source and one POSIX file specified as target; however, this is not a directory.
	64	POS6017	More than one POSIX file specified as source, but BS2000-FILE=*BY-SOURCE not specified.
	64	POS6018	The required version of SDF or SDF-P-BASYS is not installed.
	64	POS6019	Error when starting the POSIX shell.
	64	POS6020	Error when executing the POSIX command bs2cp.
	64	POS6021	Invalid specification of FILE-ATTRIBUTES.
	64	POS6022	Error when executing the POSIX command bs2file.
	64	POS6023	Invalid specification of wildcards.
	64	POS6024	Invalid specification in the ATTRIBUTES operand.

9.4 EXECUTE-POSIX-CMD

Call POSIX commands from BS2000

Domain: PROCEDURE

Privileges: STD-PROCESSING

The BS2000 command EXECUTE-POSIX-CMD provides a way of calling POSIX shell commands from BS2000. This means that the shell no longer needs to be called explicitly to execute commands, and you can start individual commands, entire command sequences or shell scripts in BS2000.

For example, you can use EXECUTE-POSIX-CMD to configure directories in POSIX before a copy process with COPY-POSIX-FILE or to process the copied files further after a copy process.

Knowledge of the syntax of shell commands is a prerequisite for using EXECUTE-POSIX-CMD.

The commands or command sequences can either be entered explicitly or be read and executed from a BS2000 file. When multiple commands/command sequences are entered explicitly these are entered as individual list elements separated by commas. However, no separate check is performed; each list element is forwarded to the shell just as it is.
When entry is via a BS2000 file, it is possible to start commands and shell scripts after they have been copied to BS2000. No parameterization is possible in this case.

The commands/command sequences called explicitly or implicitly can be stored in a log file when EXECUTE-POSIX-CMD is executed. As this file is a BS2000 file, it can, among other things, be used as an input file for a subsequent call of EXECUTE-POSIX-CMD. If, for example, a comprehensive command sequence is to be performed several times, this means that explicit specification is not required.

The command outputs are either displayed on the screen (SYSOUT) or written to a BS2000 file.

The following environment variable is set in the shell which is started with the EXECUTE-POSIX-CMD command:

```
EXECUTE_POSIX_CMD="YES"
```

Querying this variable, e.g. in */etc/profile* or *.profile*, enables outputs to be suppressed which are not required for the EXECUTE-POSIX-CMD command.

Format

EXECUTE**-POSIX-CMD**

Alias: **ECXCMD**

CMD = <filename 1..54> / list-poss(50): <c-string 1..1500 with-low>

,**INPUT-LOG-FILE** = ***NONE** / <filename 1..54 without-generation-version>(...)

<filename 1..54 without-gen-vers>(...)

| **WR**ITE**-MODE** = ***REPLACE** / ***EXTEND**

,**OUTPUT** = ***SYSOUT** / <filename 1..54>

Operands

CMD=

Specifies the commands or scripts to be executed.

CMD = <filename 1..54>
The commands/command sequences are read from a BS2000 file.

CMD = list-poss (50): <c-string 1..1500 with-low>
The commands/command sequences are specified explicitly.

> If a list of strings is specified, their total length must not exceed approximately 1500, otherwise the SDF error CMD0065 may occur.

INPUT-LOG-FILE =
Specifies whether or not a log file should be written.

INPUT-LOG-FILE =*NONE
No log file is written.

INPUT-LOG-FILE = <filename 1..54 without-generation-version>(...)
Specifies the BS2000 file which is to be the log file.

WRITE-MODE = *REPLACE / *EXTEND
Specifies whether the log file is to be created or extended whenever EXECUTE-POSIX-CMD is called. WRITE-MODE is only relevant when a BS2000 file is specified.

OUTPUT =
Specifies where the command outputs are to appear.

OUTPUT = *SYSOUT
The command outputs are displayed on the screen.

OUTPUT = <filename 1..54>
The command outputs are written to a BS2000 file.

Restrictions

- Commands/scripts which are executed with EXECUTE-POSIX-CMD cannot be read from the default input as this is terminated before the command/script is executed. Consequently such commands/scripts receive EOF when they attempt to read from the standard input.

  POSIX commands which under some circumstances may read from the standard input:

  - rm

    Query when deleting write-protected files if the *-f* option was not specified; EOF when reading *stdin* is treated like *yes*, i.e. the file is deleted.

  - mv

    Query when overwriting write-protected files if the *-f* option was not specified; the file is not overwritten and an error is generated.

  - bs2cp

    Query when overwriting BS2000 files if the *-f* option was not specified; EOF when reading *stdin* is treated like *no*, i.e. the BS2000 file is not overwritten.

  - mailx

    Inputs to *mailx* are not possible, i.e. only the query as to whether messages are present makes sense.

- In the case of EXECUTE-POSIX-CMD *stdout* and *stderr* are not connected to a terminal but to a pipe. Commands/scripts which require *stdout* and *stderr* to be connected to a terminal therefore do not function or do not function correctly. These commands/scripts use the CRTE functions *isatty()* and *ttyname()* to ascertain the terminal with which *stdout* and *stderr* are connected.

  POSIX commands may not function or not function correctly for this reason are:

  - tty

    Returns *not a tty* with exit status *1* instead of */dev/term/nnnn*.

  - tabs

    Does not function on block terminals.

  - mesg, write, talk

    These commands for exchanging messages between terminals only function to a very limited degree on block terminals, and virtually not at all under EXECUTE-POSIX-CMD.

  - more

    The *more* command under EXECUTE-POSIX-CMD behaves like the *cat* command.

  - patch

    In the event of queries an empty answer is generated, which can result in endless loops.

  - pax

    Interactive mode (*-i* option) is not possible.

  - nohup

    The *nohup* command does not work because *stdout* is not a terminal.

  - ls

    The *ls* command outputs the files in several columns only if this is explicitly requested with *ls -C*.

  - fg

    Returns *No Job Control*.

  - bg

    Returns *No Job Control*.

- If the shell command *exec* is executed with EXECUTE-POSIX-CMD, the current shell is unloaded and the mechanisms for forwarding outputs and/or the exit value of forked processes may be disabled.

  Example

```
/begin-block
%BEGIN-BLOCK/ecxcmd ('exec who am i')
%BEGIN-BLOCK/if-cmd-error
%IF-CMD-ERROR/wrtx 'cmd 1 failed'
%IF-CMD-ERROR/else
%ELSE/wrtx 'cmd 1 ok'
%ELSE/end-if
%BEGIN-BLOCK/ecxcmd ('exec who ar u')
%BEGIN-BLOCK/if-cmd-error
%IF-CMD-ERROR/wrtx 'cmd 2 failed'
%IF-CMD-ERROR/else
%ELSE/wrtx 'cmd 2 ok'
%ELSE/end-if
%BEGIN-BLOCK/end-block
FROEDE     sf/002       Nov 12 09:07
cmd 1 ok
who: Syntax:
    who [-mu] -s [-bHlprt] [datei]
    who [-mTu] [-abdHlprt] [datei]
    who -q [-n #] [datei]
    who [am i|am I]
cmd 2 ok
/
```

- The *fc* command has no effect on inputs outside the script. It is therefore not suitable for use under /EXECUTE-POSIX-CMD.
- The shell commands executed with EXECUTE-POSIX-CMD are not logged in the customary command memory (*$HOME/.sh_history*) but in a separate, relatively short-term (*HISTSIZE=100*) command memory under */tmp/.ecxcmd_sh_history_<user-name>*.
- Command substitutions by *'command'* or *$(command)* under EXECUTE-POSIX-CMD are always executed in a subshell. The POSIX shell, on the other hand, provides a range of commands which can be substituted within the shell.

  The result of this is that individual commands can behave differently from in the POSIX shell if the results are process-specific. Familiar cases here are *ftyp* and *bs2file*, and also accesses to variables or functions which have not been exported.

- Aliases:
  The command sequence to be executed by EXECUTE-POSIX-CMD is executed in the POSIX shell with the dot command. Consequently the *alias* command is available for defining aliases, but it is of no relevance for command execution in the command sequence. If aliases are to be defined and used in a command sequence, the command sequence must be copied to a (temporary) POSIX file. This file must be assigned the Execute right and be executed (not with the dot command).

  The following procedures make sense:

  - The command sequence is created as a BS2000 file and is copied to a temporary POSIX file using COPY-POSIX-FILE. It is then executed.

    Example

    ```
    /EXEC-POSIX-CMD CMD=('chmod +x scriptfile', 'scriptfile', 'rm -f scriptfile')
    ```

  - The command sequence itself generates a temporary script file in POSIX from a Here document and executes this.

    Example

    ```
    cat <<***END_OF_SCRIPT >/tmp/my_scriptfile_$$
    command1
    command2
    ...
    ***END_OF_SCRIPT
    /tmp/my_scriptfile_$$
    rm -f /tmp/my_scriptfile_$$
    ```

- EXECUTE-POSIX-CMD terminates only when all the background processes started from the command sequence have been terminated. The *nohup* command cannot be used to force asynchronous processing.

- The command sequence called with EXECUTE-POSIX-CMD is executed in a subshell which is generated internally by means of *fork*. The SYSFILE environment of the calling procedure is not available in this subshell. This can have an effect on the BS2000 commands which are called with *bs2cmd*, and on the POSIX commands *lp*, *lpstat* and *cancel*.

  Example

  ```
  /begin-block
  /  ecxcmd 'bs2cmd sh-sys-file-ass \*syscmd'
  /end-block
  PROCEDURE LEVEL NUMBER 0
  SYSCMD  : (PRIMARY)
  /
  ```

  but cf.:

  ```
  /begin-block
  /  start-posix-shell
  /end-block
  POSIX basic shell ...
  *bs2cmd sh-sys-file-ass \*syscmd
  PROCEDURE LEVEL NUMBER 1
  SYSCMD  : *PRIMARY (DIALOG-BLOCK)
  *
  ```

- You must press the **K2** key twice in order to abort EXECUTE-POSIX-CMD.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No errors
	1	CMD0202	Error when calling procedure ECXCMD (see remark on SDF error CMD0065 above)
x	64	CCM0999	The shell command, command sequence or script supplies an exit status with the value x (not equal 0) which can be obtained from the SC2

9.5 MODIFY-LOGON-PROTECTION

Modify protection attributes

Domain:	USER-ADMINISTRATION
Privileges:	STD-PROCESSING
	USER-ADMINISTRATION

Modifies existing protection attributes for user IDs.

The following persons are authorized to execute the command:

- Global user administrators (with the USER-ADMINISTRATION privilege) for all user IDs
- Group administrators who at least have the MANAGE-MEMBERS attribute for the user IDs assigned to and subordinated to their user group.

Operands which are not specified remain unchanged (default value *UNCHANGED or *NONE).

The command /MODIFY-LOGON-PROTECTION is the means to reactivate user IDs which have been locked by the system either because they have exceeded their expiration date or because it has been too long since their password was last changed. In the first case, a new, future expiration date (EXPIRATION-DATE) must be entered, in the latter case a new password must be defined.

Only the part of the command that is relevant to POSIX is shown in the syntax diagram below. The operand BATCH-ACCESS can also be significant (e.g. for at, batch, crontab).

The command is described in full in the "SECOS Access Control" [9] manual.

Format

MODIFY**-LOGON-PROT**ECTION

```
. . .
,POSIX-RLOGIN-ACCESS = *UNCHANGED / *YES(...) / *NO
    *YES(...)
        |  PASSWORD-CHECK = *UNCHANGED / *YES / *NO
        |  ,TERMINAL-SET = *UNCHANGED / *NO-PROTECTION / *NONE /
        |                              *EXCEPTION-LIST(...) / *MODIFY-LIST(...) /
        |                              list-poss(48): <name 1..8> (...)
        |      *EXCEPTION-LIST (...)
        |          |  TERMINAL-SET = *NONE / list-poss(48): <name 1..8>(...)
        |          |       <name 1..8>(...)
```

```
        |           |           |   SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |       *MODIFY-LIST(...)
        |           |   REMOVE-TERMINAL-SETS = *NONE / *ALL / list-poss(48): <name 1..8>(...)
        |           |       <name 1..8>(...)
        |           |           |   SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |           |   ,ADD-TERMINAL-SETS = *NONE / *ALL / list-poss(48): <name 1..8>(...)
        |           |       <name 1..8>(...)
        |           |           |   SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |       <name 1..8> (...)
        |           |       SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |   ,GUARD-NAME = *UNCHANGED / *NONE / <filename 1..18 without-cat-gen-vers>
,POSIX-REMOTE-ACCESS = *UNCHANGED / *YES(...) / *NO
    YES(...)
        |   TERMINAL-SET = *UNCHANGED / *NO-PROTECTION / *NONE /
        |                              *EXCEPTION-LIST(...) / *MODIFY-LIST(...)
        |                              list-poss(48): <name 1..8> (...)
        |       *EXCEPTION-LIST (...)
        |           |   TERMINAL-SET = *NONE / list-poss(48): <name 1..8>(...)
        |           |       <name 1..8>(...)
        |           |           |   SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |       *MODIFY-LIST(...)
        |           |   REMOVE-TERMINAL-SETS = *NONE / *ALL / list-poss(48): <name 1..8>(...)
        |           |       <name 1..8>(...)
        |           |           |   SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |           |   ,ADD-TERMINAL-SETS = *NONE / *ALL / list-poss(48): <name 1..8>(...)
        |           |       <name 1..8>(...)
        |           |           |   SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |       <name 1..8>(...)
        |           |   SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |   ,GUARD-NAME = *UNCHANGED / *NONE / <filename 1..18 without-cat-gen-vers>
```

Operands

POSIX-RLOGIN-ACCESS = *UNCHANGED/ *YES(...) / *NO

The access class attributes for system access via a remote terminal can be modified.

POSIX-RLOGIN-ACCESS = *YES(...)
The BS2000 user ID is open for system access via a remote terminal.

PASSWORD-CHECK = *UNCHANGED / *YES / *NO
Determines whether the password is checked following system access via a remote terminal.

TERMINAL-SET = *UNCHANGED / *NO-PROTECTION / *NONE / *EXCEPTION-LIST(...) / *MODIFY-LIST (...) / list-poss(48): <name 1..8>(...)
Specifies whether the ID used for access over a POSIX remote login is protected by terminal sets.

TERMINAL-SET = *NO-PROTECTION
The ID is not protected by terminal sets.

TERMINAL-SET = *NONE
An empty terminal set list is assigned to the ID used for access over a POSIX remote login, i.e. no POSIX remote login is permitted.

TERMINAL-SET = *EXCEPTION-LIST(...)
A negative list of terminal sets is assigned.

TERMINAL-SET = *NONE
The negative list is empty, i.e. a POSIX remote login is permitted without restriction.

TERMINAL-SET = list-poss(48): <name 1..8>(...)
Access over a POSIX remote login is forbidden for terminals whose names match the terminal names in the specified terminal sets.

The significance of the subordinate operands is the same as for the following operand: TERMINAL-SET=list-poss(48): <name 1..8>(...).

TERMINAL-SET = *MODIFY-LIST(...)
Modifications are made to a terminal set list which has already been defined. The list property (negative list or positive list) remains unaffected by the modification.

REMOVE-TERMINAL-SETS =
Specifies terminal sets to be removed from the terminal set list for POSIX remote login access by the user ID.

If no terminal set list has been defined yet for POSIX remote login access for the user ID, a warning is output and the command processing continues. The same applies if one or more of the terminal sets to be removed is not included in the list.

REMOVE-TERMINAL-SETS = *NONE
No terminal sets are removed from the terminal set list.

REMOVE-TERMINAL-SETS = *ALL
All terminal sets are removed from the terminal set list.

REMOVE-TERMINAL-SETS = list-poss(48): <name 1..8>(...)
The terminal sets whose names are specified are removed from the terminal set list.

The significance of the subordinate operands is the same as for the following operand: TERMINAL-SET=list-poss(48): <name 1..8>(...).

ADD-TERMINAL-SETS =
Specifies terminal sets to be added to the defined terminal set list for POSIX remote login access for the user ID.

If no terminal set list has been defined yet for POSIX remote login access for the user ID, a positive list is implicitly created. If one or more of the terminal sets to be added is already included in the list, a warning is output.

ADD-TERMINAL-SETS = *NONE
No terminal sets are added to the defined terminal set list.

ADD-TERMINAL-SETS = list-poss(48): <name 1..8>(...)
The terminal sets whose names are specified are added to the defined terminal set list.

The significance of the subordinate operands is the same as for the following operand: TERMINAL-SET=list-poss(48): <name 1..8>(...).

TERMINAL-SET = list-poss(48): <name 1..8>(...)
A positive list of terminal sets is assigned. Access over a POSIX remote login is permitted for terminals whose names match the terminal names in the specified terminal sets.

SCOPE =
Class of the terminal set name.

SCOPE = *STD
By default, a global user administrator assigns global terminal sets and a group administrator assigns local terminal sets.

SCOPE = *USER
A terminal set owned by the user ID is assigned.

SCOPE = *GROUP
A terminal set owned by the user ID group is assigned.

SCOPE = *SYSTEM
A jointly owned terminal set is assigned.

GUARD-NAME = *UNCHANGED / *NONE / <filename 1..18 without-cat-gen-vers>Specifies whether access via the POSIX remote login is protected with a guard.

GUARD-NAME = *NONE
Access via the POSIX remote login is not protected with a guard.

GUARD-NAME = <filename 1..18 without-cat-gen-vers>
Access via the POSIX remote login is only permitted if the access conditions in the specified guard have been sastisfied. The protected user ID must be be an authorized user of the specified guard. When evaluating the guard, only the time conditions Date, Time and Weekday are taken into account. The protected user ID is the subject of the access conditions.

POSIX-RLOGIN-ACCESS = NO
The BS2000 user ID is locked for system access via a remote terminal.

POSIX-REMOTE-ACCESS = *UNCHANGED / *YES(...) / *NO
The BS2000 user ID is opened or locked for system access via a POSIX remote command (e.g. rsh, rcp).

TERMINAL-SET = *UNCHANGED / *NO-PROTECTION / *NONE / *EXCEPTION-LIST(...) / *MODIFY-LIST(...) / list-poss(48): <name 1..8>(...)
Specifies whether the ID is protected by terminal sets for access via a POSIX remote command.

TERMINAL-SET = *NO-PROTECTION
The ID is not protected by terminal sets.

TERMINAL-SET = *NONE
An empty terminal set list is assigned to the ID for access via a POSIX remote command, i.e. no access is permitted via a POSIX remote command.

TERMINAL-SET = *EXCEPTION-LIST(...)
A negative list of terminal sets is assigned.

TERMINAL-SET = *NONE / list-poss(48): <name 1..8>(...)
The negative list is empty, i.e. access via a POSIX remote command is permitted without restriction.

TERMINAL-SET = list-poss(48): <name 1..8>(...)
Access via a POSIX remote command is forbidden for terminals whose names match the terminal names in the specified terminal sets.

The significance of the subordinate operands is the same as for the following operand: TERMINAL-SET=list-poss(48): <name 1..8>(...).

TERMINAL-SET = *MODIFY-LIST(...)
Modifications are made to a terminal set list which has already been defined. The list property (negative list or positive list) remains unaffected by the modification.

REMOVE-TERMINAL-SETS =
Specifies terminal sets to be removed from the terminal set list for POSIX remote command access by the user ID.

If no terminal set list has been defined yet for POSIX remote command access by the user ID, a warning is output and command processing continues. The same applies if one or more of the terminal sets to be removed is not included in the list.

REMOVE-TERMINAL-SETS = *NONE
No terminal sets are removed from the terminal set list.

REMOVE-TERMINAL-SETS = *ALL
All terminal sets are removed from the terminal set list.

REMOVE-TERMINAL-SETS = list-poss(48): <name 1..8>(...)
The terminal sets whose names are specified are removed from the terminal set list.

The significance of the subordinate operands is the same as for the following operand: TERMINAL-SET=list-poss(48): <name 1..8>(...).

ADD-TERMINAL-SETS =
Specifies terminal sets to be added to the terminal set list defined for POSIX remote command access by the user ID.

If no terminal set list has been defined yet for POSIX remote command access by the user ID, a positive list is implicitly created. If one or more of the terminal sets to be added is already included in the list, a warning is output.

ADD-TERMINAL-SETS = *NONE
No terminal sets are added to the defined terminal set list.

ADD-TERMINAL-SETS = list-poss(48): <name 1..8>(...)
The terminal sets whose names are specified are added to the defined terminal set list.

The significance of the subordinate operands is the same as for the following operand: TERMINAL-SET=list-poss(48): <name 1..8>(...).

TERMINAL-SET = list-poss(48): <name 1..8>(...)
A positive list of terminal sets is assigned. Access via a POSIX remote command is permitted for terminals whose names match the terminal names in the specified terminal sets.

SCOPE =
Class of the terminal set name.

SCOPE = *STD
By default, a global user administrator assigns global terminal sets and a group administrator assigns local terminal sets.

SCOPE = *USER
A terminal set owned by the user ID is assigned.

SCOPE = *GROUP
A terminal set owned by the user ID group is assigned.

SCOPE = *SYSTEM
A jointly owned terminal set is assigned.

GUARD-NAME = *UNCHANGED / *NONE / <filename 1..18 without-cat-gen-vers>Specifies whether access via a POSIX remote command is protected by a guard.

GUARD-NAME = *NONE
Access via a POSIX remote command is not protected by a guard.

GUARD-NAME = <filename 1..18 without-cat-gen-vers>

Access via a POSIX remote command is only permitted if the access conditions in the specified guard have been sastisfied. The protected user ID must be be an authorized user of the specified guard. When evaluating the guard, only the time conditions Date, Time and Weekday are taken into account. The POSIX user ID under which the commands *rsh* or *rcp* were entered is the subject of the access conditions. It is not necessary for this user ID to exist in BS2000.

POSIX-REMOTE-ACCESS = *NO

The BS2000 user ID is locked for system access via a POSIX remote command.

9.6 MODIFY-POSIX-USER-ATTRIBUTES

Modify POSIX user attributes.

Domain:	USER-ADMINISTRATION
Privileges:	POSIX-ADMINISTRATION USER-ADMINISTRATION STD-PROCESSING

This command modifies the POSIX user attributes of a BS2000 user ID in the user catalog of the specified pubset.

Whenever a new BS2000 user ID is created, the POSIX user attributes with the default values are automatically assigned (see "Assigning POSIX user attributes"). These POSIX user attributes can be modified as required. The following users are authorized to do this:

- Owners of the POSIX-ADMINISTRATION or USER-ADMINISTRATION privilege for all BS2000 user IDs on all pubsets.
- Group administrators for their assigned group and subgroup members on the pubset which they are responsible for administering. However, the following restrictions apply to a group administrator:
  - The administrator's ADM-AUTHORITY authorization determines the POSIX user attributes which he/she is authorized to administer.
  - The value range of the POSIX user attributes is restricted for the administrator.

  Further details on the above are available in the description of the corresponding operands.

> ¡ Additionally the EDIT-POSIX-USER-ATTRIBUTES command is available which can be used to show and modify the current settings.

Format

MODIFY**-POSIX-USER-ATTR**IBUTES

USER-IDENTIFICATION = <name 1..8>

,**PUBSET** = ***HOME** / <catid 1..4>

,**USER-NUM**BER = ***UNCHA**NGED / ***BY-POSIX-USER-DEF**AULTS / ***HOME** / <integer 0..60002>

,**GR**OUP**-NUM**BER = ***UNCHA**NGED / ***BY-POSIX-USER-DEF**AULTS / ***GR**OUP**-ADM**INISTRATOR /
<integer 0..60002>

,**COMMENT** = ***UNCHA**NGED / ***BY-POSIX-USER-DEF**AULTS / ***NONE** / <c-string 1..255 with-low>

,**DIR**ECTORY = ***UNCHA**NGED / ***BY-POSIX-USER-DEF**AULTS / ***ROOT** /
<posix-pathname 1..1023 without-wild>

,**PROG**RAM = ***UNCHA**NGED / ***BY-POSIX-USER-DEF**AULTS / ***SHELL** /
<posix-pathname 1..1023 without-wild>

Operands

USER-IDENTIFICATION = <name 1..8>

BS2000 user ID whose POSIX user attributes are to be modified.

PUBSET =
Pubset in whose user directory the POSIX user attributes are to be modified.

PUBSET = *HOME
The home pubset is modified.

PUBSET = <catid 1..4>
The pubset with the specified catalog ID is modified.

USER-NUMBER =
The user number which is assigned automatically during creation of a BS2000 user ID can be modified.

The USER-NUMBER attribute is relevant for security since the user number confers privileges and determines the owner of a file.

The group administrator can only modify the user number if he/she holds at least the MANAGE-MEMBERS group administrator privilege. Even then, he/she cannot use the full range of values:

- The administrator cannot assign the user number 0, i.e. root authorization.
- The administrator can only modify the default user number.
- The administrator can only assign user numbers which are greater than the default user number.
- The administrator cannot assign user numbers more than once.
- On a data pubset the administrator can only assign the user number of the BS2000 user ID of the same name on the home pubset.

USER-NUMBER = *UNCHANGED
The user number is not modified.

USER-NUMBER = *BY-POSIX-USER-DEFAULTS
The user number receives the corresponding default value, which is entered in the user directory of the specified pubset.

USER-NUMBER = *HOME
The user number of the BS2000 user ID of the same name on the home pubset is accepted.This value is only significant if the user number is modified on a data pubset. This specification is redundant on the home pubset.

USER-NUMBER = <integer 0..60002>
The user number receives the specified value.

GROUP-NUMBER =
The group number which is automatically assigned during creation of a BS2000 user ID can be modified.

The GROUP-NUMBER attribute is relevant for security since, at login, POSIX does not check the validity of the combination BS2000 user ID and group against the POSIX group directory.

The group administrator can only modify the group number if he/she holds the MANAGE-MEMBERS group administrator privilege. Even then, he/she does not have access to the full range of values:

- The administrator can only assign the group number owned by the group administrator of the BS2000 user group of which the BS2000 user ID is a member, or the default group number.
- The administrator can assign no other group number for his/her own BS2000 user ID.

GROUP-NUMBER = *UNCHANGED
The group number is not modified.

GROUP-NUMBER = *BY-POSIX-USER-DEFAULTS
The group number receives the appropriate default value which, is entered in the user catalog of the specified pubset.

GROUP-NUMBER = *GROUP-ADMINISTRATOR
The group number owned by the group administrator of the BS2000 user group of which the BS2000 user ID is a member is assigned.

GROUP-NUMBER = <integer 0..60002>
The group number receives the specified value.

COMMENT =
The comment can be modified. Further information on the owner of the BS2000 user ID can be specified at the administrator’s discretion.

Note

This comment is, for example, used by mail programs to describe the sender.

COMMENT = *UNCHANGED
The comment is not modified.

COMMENT = *BY-POSIX-USER-DEFAULTS
The value of the appropriate POSIX default attribute, which is entered in the user catalog of the specified pubset, is assigned.

COMMENT = *NONE
No comment is entered.

COMMENT = <c-string 1..255 with-low>
The specified comment is entered.

DIRECTORY =
The absolute path name to the login directory of the user can be modified.

This attribute is not relevant for security since it only determines the contents of the HOME shell variables and the initial value of the working directory. It does not offer a way of bypassing the protection attributes of files and directories.

DIRECTORY = *UNCHANGED
The absolute path name is not modified.

DIRECTORY = *BY-POSIX-USER-DEFAULTS
The value of the appropriate POSIX default attribute, which is entered in the user catalog of the specified pubset, is assigned.

DIRECTORY = *ROOT
The root directory “/” is allocated.

DIRECTORY = <posix-pathname 1..1023 without-wild>
The specified path name is accepted.

PROGRAM =
The program which is started either by the *rlogin* command or by calling the /START-POSIX-SHELL command can be modified.

This attribute is not relevant for security since only those programs which the user is authorized to execute can be started.

PROGRAM = *UNCHANGED
The program is not modified.

PROGRAM = *BY-POSIX-USER-DEFAULTS
The value of the appropriate POSIX default attribute, which is entered in the user catalog of the specified pubset, is assigned.

PROGRAM = *SHELL
The POSIX shell is started.

PROGRAM = <posix-pathname 1..1023 without-wild>
The specified program is started.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
2	0	SRM6001	Command executed with warning
	32	SRM6020	Command rejected due to system error
	130	SRM6030	Command rejected because of unavailable resources
	64	SRM6040	Command rejected with error message

Examples

The POSIXTST user ID is to be assigned user number 107 and group number 66. The login directory (home directory) is to be named */home/posixtst* and the Bourne shell is to be called after logging in to POSIX. The comment is to read “posix-user@posix-server.com”.

```
/MODIFY-POSIX-USER-ATTRIBUTES USER-ID=POSIXTST, -
/                             USER-NUMBER=107, -
/                             GROUP-NUMBER=66, -
/                             DIRECTORY=/home/posixtst, -
/                             COMMENT='posix-user@posix-server.com'
```

The PSXROOT user ID is to be assigned the root authorization. */home/psxroot* is to be entered as the home directory.

```
/MODIFY-POSIX-USER-ATTRIBUTES USER-ID=PSXROOT, -
/                             USER-NUMBER=0, -
/                             GROUP-NUMBER=0, -
/                             DIRECTORY=/home/psxroot
```

9.7 MODIFY-POSIX-USER-DEFAULTS

Modify default values for POSIX user attributes

Domain:	USER-ADMINISTRATION
Privileges:	POSIX-ADMINISTRATION USER-ADMINISTRATION STD-PROCESSING

This command modifies the values for the POSIX default attributes in the user catalog of the specified pubset. The following users may execute the command:

- Owners of the POSIX-ADMINISTRATION or USER-ADMINISTRATION privilege for all pubsets.
- Group administrators of the *UNIVERSAL group on the pubset administered by them.

The POSIX default attributes are used when creating a new user (with ADD-USER).

> ¡ Additionally the EDIT-POSIX-USER-DEFAULTS command is available which can be used to show and modify the current settings.

Format

MODIFY**-POSIX-USER-DEF**AULTS
PUBSET = ***HOME** / <catid 1..4>
,USER-NUMBER = ***UNCHA**NGED / <integer 0..60002>
,GROUP**-NUM**BER = ***UNCHA**NGED / <integer 0..60002>
,COMMENT = ***UNCHA**NGED / ***NONE** / <c-string 1..255 with-low>
,DIRECTORY = ***UNCHA**NGED / ***ROOT** / <posix-pathname 1..1023 without-wild>
,**PROG**RAM = ***UNCHA**NGED / ***SHELL** / <posix-pathname 1..1023 without-wild>

Operands

PUBSET =

Pubset in whose user catalog the POSIX default attributes are to be modified.

PUBSET = *HOME
The POSIX default attributes are modified in the user catalog of the home pubset.

PUBSET = <catid 1..4>
The POSIX default attributes are modified in the user catalog of the specified pubset.

USER-NUMBER =
The user number can be modified.

USER-NUMBER = *UNCHANGED
The user number is not modified.

USER-NUMBER = <integer 0..60002>
The user number receives the specified value.

GROUP-NUMBER =
The group number can be modified.

GROUP-NUMBER = *UNCHANGED
The group number is not modified.

GROUP-NUMBER = <integer 0..60002>
The group number receives the specified value.

COMMENT =
The comment can be modified.

Note

This comment is, for example, used by mail programs to describe the sender.

COMMENT = *UNCHANGED
The comment is not modified.

COMMENT = *NONE
No comment is entered.

COMMENT = <c-string 1..255 with-low>
The specified comment is entered.

DIRECTORY =
The absolute path name to the login directory of the user can be modified.

DIRECTORY = *UNCHANGED
The absolute path name is not modified.

DIRECTORY = *ROOT
Control switches to the root directory.

DIRECTORY = <posix path name 1..1023 without-wild>
Control switches to the specified path name.

PROGRAM =
The program which is started after the user has logged on can be modified.

PROGRAM = *UNCHANGED
The program is not modified.

PROGRAM = *SHELL
The default POSIX shell is started.

PROGRAM = <posix path name 1..1023 without-wild>
The specified program is started.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
2	0	SRM6001	Command executed with warning
	32	SRM6020	Command rejected due to system error
	64	SRM6040	Command rejected with error message
	130	SRM6030	Command rejected because of unavailable resources

9.8 MODIFY-USER-ATTRIBUTES

Modify user catalog entry

Domain:	ACCOUNTING USER-ADMINISTRATION
Privileges:	USER-ADMINISTRATION STD-PROCESSING

This command modifies the attributes of a BS2000 user ID in the user catalog. The following users may execute the command:

- Owners of the USER-ADMINISTRATION privilege for all BS2000 user IDs.
- Group administrators for BS2000 user IDs which are assigned to and subordinate to their groups.

Only the part of the command that is relevant to POSIX is shown in the syntax diagram below. The command is described in full in the manuals "SECOS - Access Control" [9] and "BS2000 Commands" [28].

> ¡ Additionally the EDIT-USER-ATTRIBUTES command is available which can be used to show and modify the current settings.

Format

MODIFY**-USER**-ATTRIBUTES

```
. . .
,ACCOUNT-ATTRIBUTES = *UNCHANGED / *ADD(...) / *MODIFY(...) / *REMOVE(...)
    *ADD(...)
        | ACCOUNT = <alphanum-name 1..8>
        | . . .
    ,POSIX-RLOGIN-DEFAULT = *NO / *YES
    *MODIFY(...)
        | ACCOUNT = <alphanum-name 1..8>
        | . . .
        | ,POSIX-RLOGIN-DEFAULT = *UNCHANGED / *NO / *YES
    *REMOVE(...)
```

Operands

ACCOUNT-ATTRIBUTES = ... / *ADD(...) / *MODIFY(...) / ...

Determines the accounting data of a BS2000 user ID.

ACCOUNT-ATTRIBUTES = *ADD(...)
A new account number and specific attributes are entered for the BS2000 user ID.

ACCOUNT = <alphanum name 1..8>
Account number of the BS2000 user ID which is added to the user catalog and to which the following specifications refer.

. . .

POSIX-RLOGIN-DEFAULT =
Determines whether the specified account number is used for accounting for the remote login session.

POSIX-RLOGIN-DEFAULT = *NO
The specified account number is not used for accounting.

POSIX-RLOGIN-DEFAULT = *YES
The specified account number is used for accounting.

ACCOUNT-ATTRIBUTES = *MODIFY(...)
The attributes of an entered account number of the BS2000 user ID are modified.

ACCOUNT = <alphanum name 1..8>
Account number of the BS2000 user ID for which the subsequent values are modified in the user catalog.

. . .

POSIX-RLOGIN-DEFAULT =
Determines whether the account number to be modified is used for accounting for the remote login session.

POSIX-RLOGIN-DEFAULT = *UNCHANGED
The value set up to now is retained.

POSIX-RLOGIN-DEFAULT = *NO
The account number is not used for accounting.

POSIX-RLOGIN-DEFAULT = *YES
The account number is used for accounting.

Within a BS2000 user ID, the account number for remote login is unique. User administration automatically compares the number with the existing account numbers.

The account number for remote login can also be specified for the accounting for a BS2000 session and, as a result, a BS2000 user ID can manage with one single account number.

POSIX-RLOGIN-DEFAULT=*YES is required for user IDs which wish to have remote access to POSIX (*rlogin* or Telnet access, *rsh* and *rcp* commands) or to use *at*, *crontab* or *batch*.

If no account number for remote login is specified, access via remote login is refused, unless the user ID requesting access is TSOS.

9.9 SET-LOGON-PROTECTION

Define protection attributes

Domain:	USER-ADMINISTRATION
Privileges:	USER-ADMINISTRATION, STD-PROCESSING

This command defines access protection attributes for existing user IDs.

The following users may execute this command:

- Owners of the USER-ADMINISTRATION privilege for all user IDs.
- Group administrators who hold at least the MANAGE-MEMBERS attribute for the user IDs assigned to and subordinate to their groups.

The user interface of the command SHOW-LOGON-PROTECTION is not changed by POSIX. Only the part of the command that is relevant to POSIX is shown in the syntax diagram below. The operand BATCH-ACCESS can also be of significance (e.g. for at, batch, crontab).

The command is described in full in the "SECOS - Access Control" [9] manual.

Format

SET-LOGON-PROTECTION

```
. . .
,POSIX-RLOGIN-ACCESS = *YES (...) / *NO
    *YES(...)
        | PASSWORD-CHECK = *YES / *NO
        | ,TERMINAL-SET = *NO-PROTECTION / *NONE / *EXCEPTION-LIST(...) /
        |                 list-poss(48): <name 1..8>(...)
        |     *EXCEPTION-LIST(TERMINAL-SET = *NONE / list-poss(48): <name 1..8>(...)
        |         | TERMINAL-SET = *NONE / list-poss(48): <name 1..8>(...)
        |         |     <name 1..8>(...)
        |         |         | SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |     <name 1..8>(...)
        |         | SCOPE = *STD / *USER / *GROUP / *SYSTEM
        | ,GUARD-NAME = *NONE / <filename 1..18 without-cat-gen-vers>
,POSIX-REMOTE-ACCESS = *YES (...) / *NO
    *YES(...)
        | ,TERMINAL-SET = *NO-PROTECTION / *NONE / *EXCEPTION-LIST(...) /
        |                 list-poss(48): <name 1..8>(...)
        |     *EXCEPTION-LIST(TERMINAL-SET = *NONE / list-poss(48): <name 1..8>(...)
        |         |     <name 1..8>(...)
        |         |         | SCOPE = *STD / *USER / *GROUP / *SYSTEM
        |     <name 1..8>(...)
        |         | SCOPE = *STD / *USER / *GROUP / *SYSTEM
        | ,GUARD-NAME = *NONE / <filename 1..18 without-cat-gen-vers>
. . .
```

Operands

POSIX-RLOGIN-ACCESS =

The access class attributes for system access via a remote terminal can be defined.

POSIX-RLOGIN-ACCESS = *YES(...)
The BS2000 user ID is open for system access via a remote terminal.

PASSWORD-CHECK =*YES / *NO
Determines whether the password is checked following system access via a remote terminal.

TERMINAL-SET =
Specifies whether the ID used for access via a POSIX remote login is protected by terminal sets.

TERMINAL-SET = *NO-PROTECTION
The ID is not protected by terminal sets.

TERMINAL-SET = *NONE
An empty terminal set list is assigned to the ID, i.e. no POSIX remote login is permitted.

TERMINAL-SET = *EXCEPTION-LIST(...)
A negative list of terminal sets is assigned.

TERMINAL-SET = *NONE / list-poss(48): <name 1..8>(...)
The negative list is empty, i.e. the POSIX remote login is permitted without restriction.

TERMINAL-SET = list-poss(48): <name 1..8>(...)
Access via a POSIX remote login is forbidden for terminals whose names match the terminal names in the specified terminal sets.

The significance of the subordinate operands is the same as for the following operand: TERMINAL-SET.

TERMINAL-SET = list-poss(48): <name 1..8>(...)
A positive list of terminal sets is assigned. Access over a POSIX remote login is permitted for terminals whose names match the terminal names in the specified terminal sets.

SCOPE =
Class of the terminal set name.

SCOPE = *STD
By default, a global user administrator assigns global terminal sets and a group administrator assigns local terminal sets.

SCOPE = *USER
A terminal set owned by the user ID is assigned.

SCOPE = *GROUP
A terminal set owned by the user ID group is assigned.

SCOPE = *SYSTEM
A jointly owned terminal set is assigned.

GUARD-NAME =
Specifies whether access via a POSIX remote login is protected by a guard.

GUARD-NAME = *NONE
Access via a POSIX remote login is not protected by a guard.

GUARD-NAME = <filename 1..18 without-cat-gen-vers>
Access via the POSIX remote login is only permitted if the access conditions in the specified guard have been sastisfied. The protected user ID must be be an authorized user of the specified guard. When evaluating the guard, only the time conditions Date, Time and Weekday are taken into account. The protected user ID is the subject of the access conditions.

POSIX-RLOGIN-ACCESS = *NO
The BS2000 user ID is locked for system access via a POSIX remote login.

POSIX-REMOTE-ACCESS = *YES(...) / *NO
The BS2000 user ID is opened or locked for system access via a POSIX remote command.

TERMINAL-SET =
Specifies whether the ID is protected by terminal sets for access via a POSIX remote command.

TERMINAL-SET = *NO-PROTECTION
The ID is not protected by terminal sets.

TERMINAL-SET = *NONE
An empty terminal set list is assigned to the ID, i.e. no access is permitted via a POSIX remote command.

TERMINAL-SET = *EXCEPTION-LIST(...)
A negative list of terminal sets is assigned.

TERMINAL-SET = *NONE / list-poss(48): <name 1..8>(...)
The negative list is empty, i.e. access via a POSIX remote command is permitted without restriction.

TERMINAL-SET = list-poss(48): <name 1..8>(...)
Access via a POSIX remote command is forbidden for terminals whose names match the terminal names in the specified terminal sets.

The significance of the subordinate operands is the same as for the following operand: TERMINAL-SET.

TERMINAL-SET = list-poss(48): <name 1..8>(...)
A positive list of terminal sets is assigned. Access via POSIX remote command is permitted for terminals whose names match the terminal names in the specified terminal sets.

SCOPE =
Class of the terminal set name.

SCOPE = *STD
By default, a global user administrator assigns global terminal sets and a group administrator assigns local terminal sets.

SCOPE = *USER
A terminal set owned by the user ID is assigned.

SCOPE = *GROUP
A terminal set owned by the user ID group is assigned.

SCOPE = *SYSTEM
A jointly owned terminal set is assigned.

GUARD-NAME =
Specifies whether access via a POSIX remote command is protected by a guard.

GUARD-NAME = *NONE
Access via a POSIX remote command is not protected by a guard.

GUARD-NAME = <filename 1..18 without-cat-gen-vers>
Access via a POSIX remote command is only permitted if the access conditions in the specified guard have been sastisfied. The protected user ID must be be an authorized user of the specified guard. When evaluating the guard, only the time conditions Date, Time and Weekday are taken into account. The POSIX user ID under which the commands *rsh* or *rcp* were entered is the subject of the access conditions. It is not necessary for this user ID to exist in BS2000.

POSIX-REMOTE-ACCESS = *NO
The BS2000 user ID is locked for system access via a POSIX remote command.

Example

The PSXROOT user ID is permitted for system access via a remote terminal:

```
/SET-LOGON-PROTECTION USER-ID=PSXROOT,POSIX-RLOGIN-ACCESS=*YES
```

9.10 SHOW-LOGON-PROTECTION

Display protection attributes

Using POSIX does not change anything at the user interface of the SHOW-LOGON-PROTECTION command. The complete command is described in the "SECOS - Access Control" [9] manual.

The following table shows the POSIX-specific contents possible in the Type field of the access history and the corresponding meaning:

Type	Meaning
POS-BATCH	POSIX batch commands *at*, *cron* or *batch*
POS-REMOTE	POSIX remote commands *rcp* or *rsh*
RLOGIN	POSIX remote login

POSIX-specific S variables

Output information	Name of the S variable	T	Contents	Condition
Access check active with POSIX remote access?	Var(*LIST).POSIX-REM.ACCESS	S	*YES *NO	1
Name of guard with which POSIX remote access is protected	Var(*LIST).POSIX-REM.GUARD	S	*NONE <filename 1..18>	1
POSIX remote: password check	Var(*LIST).POSIX-REM.PASS-CHECK	S	*YES *NO	1
Terminal sets of GROUP class	Var(*LIST).POSIX-REM.TER-SET.GROUP(*LIST)	S	<name 1..8>	1
Group name	Var(*LIST).POSIX-REM.TER-SET.GROUP-ID	S	<name 1..8> *UNIV	1
Terminal sets of SYSTEM class	Var(*LIST).POSIX-REM.TER-SET.SYSTEM(*LIST)	S	<name 1..8>	1
Terminal sets of USER class	Var(*LIST).POSIX-REM.TER-SET.USER(*LIST)	S	<name 1..8>	1
User ID	Var(*LIST).POSIX-REM.TER-SET.USER-ID	S	<name 1..8>	1
POSIX remote access protected by terminal sets	Var(*LIST).POSIX-REM.TER-SET-DEFI	S	*NO-PROT *LIST *EXCEPT	1

Access check active with POSIX access via rlogin?	var(*LIST).POSIX-RLOG.ACCESS	S	*NO *YES	1
Name of guard with which POSIX rlogin access is protected	Var(*LIST).POSIX-RLOG.GUARD	S	*NONE <filename 1..18>	1
Password check active during POSIX access via rlogin?	var(*LIST).POSIX-RLOG.PASS-CHECK	S	*NO *YES	1
Terminal sets of GROUP class	Var(*LIST).POSIX-RLOG.TER-SET.GROUP(*LIST)	S	<name 1..8>	1
Group name	Var(*LIST).POSIX-RLOG.TER-SET.GROUP-ID	S	<name 1..8> *UNIV	1
Terminal sets of SYSTEM class	Var(*LIST).POSIX-RLOG.TER-SET.SYSTEM(*LIST)	S	<name 1..8>	1
Terminal sets of USER class	Var(*LIST).POSIX-RLOG.TER-SET.USER(*LIST)	S	<name 1..8>	1
User ID	Var(*LIST).POSIX-RLOG.TER-SET.USER-ID	S	<name 1..8>	1
POSIX rlogin access protected by terminal sets?	Var(*LIST).POSIX-RLOG.TER-SET-DEFI	S	*NO-PROT *LIST *EXCEPT	1
POSIX server: access	Var(*LIST).POSIX-SERVER.ACCESS	S	*YES *NO	1

9.11 SHOW-POSIX-STATUS

Show POSIX status

Domain: SYSTEM-MANAGEMENT

Privileges: SUBSYSTEM-MANAGEMENT

This command shows the status of POSIX.

Format

SHOW-POSIX-STATUS

This command outputs the current status of the POSIX subsystem to SYSOUT in the following format:

```
%POSSTAT POSIX-STATUS=<status>
```

<status> can assume the following values:

POSIX STATUS	**Meaning**
*AVAILABLE	POSIX is available for applications.
*IN-CREATE	The POSIX subsystem is currently starting.
*IN-DELETE	The POSIX subsystem is currently terminating.
*NOT-ACCESSIBLE	The POSIX subsystem is started, but is not available for applications.
*NOT-AVAILABLE	The POSIX subsystem is not loaded.
*UNKNOWN	The status is not recognizable.

Command return codes

(SC2)	**SC1**	**Maincode**	**Meaning**
	0	CMD0001	Command executed without errors

9.12 SHOW-POSIX-USER-ATTRIBUTES

Display POSIX user attributes

Domain: USER-ADMINISTRATION

Privileges: POSIX-ADMINISTRATION
USER-ADMINISTRATION
STD-PROCESSING

This command displays the POSIX user attributes of a BS2000 user ID which are entered in the user catalog of the specified pubset. The following users may execute the command:

- Owners of the POSIX-ADMINISTRATION or USER-ADMINISTRATION privilege for all BS2000 user IDs on all pubsets.
- Group administrators for the group and subgroup members under his/her responsibility on the pubset administered by them.
- Every user for his/her own BS2000 user ID.

Format

SHOW-POSIX-USER-ATTRIBUTES

USER-IDENTIFICATION = ***OWN** / ***ALL** / list-poss(20): <name 1..8>

,**PUBSET** = ***HOME** / ***ALL** / list-poss(20): <catid 1..4>

,**SEL**ECT = ***ALL** / ***BY-ATTR**IBUTES(...)

***BY-ATTR**IBUTES(...)

| **USER-NUM**BER = ***ANY** / ***BY-POSIX-USER-DEF**AULTS / ***OWN** / <integer 0..60002>

| **,GR**OUP-**NUM**BER = ***ANY** / ***BY-POSIX-USER-DEF**AULTS / ***OWN** / <integer 0..60002>

| **,COMMENT** = ***ANY** / ***BY-POSIX-USER-DEF**AULTS / ***NONE** / <c-string 1..255 with-low>

| **,DIR**ECTORY = ***ANY** / ***BY-POSIX-USER-DEF**AULTS / ***ROOT** /

| <posix-pathname 1..1023 without-wild>

| ,**PROG**RAM = ***ANY** / ***BY-POSIX-USER-DEF**AULTS / ***SHELL** /

| <posix-pathname 1..1023 without-wild>

,**INF**ORMATION = ***ALL** / ***USER-LIST**

,**OUTPUT** = list-poss(2): ***SYSOUT** / ***SYSLST**(...)

***SYSLST**(...)

| **SYSLST-NUM**BER = ***STD** / <integer 1..99>

| ,**LINE**S-PER-**PAGE** = **64** / <integer 20..255>

Operands

USER-IDENTIFICATION =

Determines the BS2000 user IDs whose POSIX user attributes are to be displayed.

USER-IDENTIFICATION = *OWN
The POSIX user attributes of the user's own BS2000 user ID, as entered in the user catalog of the specified pubset, are displayed.

USER-IDENTIFICATION = *ALL
The POSIX user attributes of all BS2000 user IDs which the caller is authorized to know are displayed.

USER-IDENTIFICATION = list-poss(20): <name 1..8>
The POSIX user attributes of the specified ID are displayed.

PUBSET =
Determines the pubset from whose user catalog the POSIX user attributes are to be displayed.

PUBSET = *HOME
The POSIX user attributes of the home pubset are displayed.

PUBSET = *ALL
The POSIX user attributes of all pubsets which are available at command input are displayed.

PUBSET = list-poss(20): <catid 1..4>
The POSIX user attributes of the specified pubset are displayed.

SELECT =
The BS2000 user IDs are selected by means of their POSIX user attributes.

SELECT = *ALL
The BS2000 user IDs are selected independently of their POSIX user attributes.

SELECT = *BY-ATTRIBUTES(...)
The BS2000 user IDs are selected depending on their POSIX user attributes. If more than one POSIX user attribute is specified, selection is made by logical ANDing.

USER-NUMBER =
The BS2000 user IDs are selected by means of their user number.

USER-NUMBER = *ANY
The BS2000 user IDs are selected independently of their user number.

USER-NUMBER = *BY-POSIX-USER-DEFAULTS
Only those BS2000 user IDs for which the appropriate default value is entered as the user number in the user catalog of the specified pubset are selected.

USER-NUMBER = *OWN
Only those BS2000 user IDs which entered the same user number as the caller in the user catalog of the specified pubset are selected.

USER-NUMBER = <integer 0..60002>
Only those BS2000 user IDs which entered the specified user number in the user catalog of the specified pubset are selected.

GROUP-NUMBER =
The BS2000 user IDs are selected on the basis of their group number.

GROUP-NUMBER = *ANY
The BS2000 user IDs are selected independently of their group number.

GROUP-NUMBER = *BY-POSIX-USER-DEFAULTS
Only those BS2000 user IDs for which the appropriate default value is entered as a group number in the user catalog of the specified pubset are selected.

GROUP-NUMBER = *OWN
Only those BS2000 user IDs which entered the same group number as the caller in the user catalog of the specified pubset are selected.

GROUP-NUMBER = <integer 0..60002>
Only those BS2000 user IDs which entered the specified group number in the user catalog of the specified pubset are selected.

COMMENT =
The BS2000 user IDs are selected by means of their comment.

COMMENT = *ANY
The BS2000 user IDs are selected independently of their comment.

COMMENT = *BY-POSIX-USER-DEFAULTS
Only the BS2000 user IDs for which the appropriate default value is entered as a comment in the user catalog of the specified pubset are selected.

COMMENT = *NONE
Only those BS2000 user IDs for which no comment has been entered are selected.

COMMENT = <c-string 1..255 with-low>
Only the BS2000 user IDs with the specified comment are selected.

DIRECTORY =
The BS2000 user IDs are selected by means of their login directory.

DIRECTORY = *ANY
The BS2000 user IDs are selected independently of their login directory.

DIRECTORY= *BY-POSIX-USER-DEFAULTS
Only those BS2000 user IDs for which the appropriate default value is entered as a login directory in the user catalog of the specified pubset are selected.

DIRECTORY = *ROOT
Only the BS2000 user IDs which entered the root directory as a login directory are selected.

DIRECTORY = <posix-pathname 1..1023 without-wild>
Only the BS2000 user IDs with the specified login directory are selected.

PROGRAM =
The BS2000 user IDs are selected by means of their program name.

PROGRAM = *ANY
The BS2000 user IDs are selected independently of the program name.

PROGRAM = *BY-POSIX-USER-DEFAULTS
Only those BS2000 user IDs for which the appropriate default value is entered as the program name in the user catalog of the specified pubset are selected.

PROGRAM = *SHELL
Only the BS2000 user IDs which entered *SHELL as the program name are selected.

PROGRAM = <posix-pathname 1..1023 without-wild>
Only the BS2000 user IDs with the specified program name are selected.

INFORMATION =
Determines the amount of information output.

INFORMATION = *ALL
All POSIX user attributes of a BS2000 user ID are displayed (see example 1).

INFORMATION = *USER-LIST
A list of the BS2000 user IDs without POSIX user attributes is displayed (see example 2).

OUTPUT =
Determines the system file for the output of information.

OUTPUT = *SYSOUT
The information is output to the SYSOUT system file.

OUTPUT = *SYSLST(...)
The information is output to the SYSLST system file.

SYSLST-NUMBER =
Determines the SYSLST number.

SYSLST-NUMBER = *STD
Determines default SYSLST output.

SYSLST-NUMBER = <integer 1..99>
Determines the specified SYSLST number.

LINES-PER-PAGE =
Specifies the line count per page.

LINES-PER-PAGE = 64
64 lines per page are printed by default.

LINES-PER-PAGE = <integer 20..255>
The specified number of lines per page is printed.

A user without an administrator function only obtains information about his/her own BS2000 user ID, with two possible exceptions:

```
INFORMATION=*USER-LIST,SELECT=*BY-ATTRIBUTES(USER-NUMBER=*OWN)
```

Where this is specified, the user also learns the identity of the users who share his/her user number, provided this user number is not the default user number.

```
INFORMATION=*USER-LIST,SELECT=*BY-ATTRIBUTES(GROUP-NUMBER=*OWN)
```

Where this is specified, the user also learns the identity of the members of his/her POSIX group, provided this POSIX group is not the default user group.

In the case of INFORMATION=*ALL, the user number and group number are marked if the appropriate default value, which is entered in the user catalog of the specified pubset, is assigned (SHOW output with “(DEFAULT)” or S variables with the suffix “-DEF”).

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
2	0	SRM6001	Command executed with warning
	32	CMD2009	Error while creating the output variable
	32	SRM6020	Command rejected due to system error
	64	OPS0002	K2 interruption during output in S variable
	64	SRM6040	Command rejected with error message
	130	OPS0001	Lack of resources during output in S variable
	130	SRM6030	Command rejected because of unavailable resources

Example 1

```
/SHOW-POSIX-USER-ATTRIBUTES USER-IDENTIFICATION=EXAMPLES,PUBSET=A
POSIX-USER-ATTRIBUTES --- PUBSET A                        2009-03-09 17:19:48
------------------------------------------------------------------------------
USER-ID          EXAMPLES                    PUBSET    A
USER-NUMBER           632
GROUP-NUMBER          632
COMMENT      S. Nobody, Mch-P, Tel.: 12345
DIRECTORY    /home/examples
PROGRAM      *SHELL
------------------------------------------------------------------------------
POSIX-USER-ATTRIBUTES                                          END OF DISPLAY
```

Example 2

```
/SHOW-POSIX-USER-ATTRIBUTES USER-ID=*ALL,PUBSET=A,INFORMATION=*USER-LIST
POSIX-USER-LIST       --- PUBSET A                        2009-03-09 17:23:12
------------------------------------------------------------------------------
EXAMPLES  SERVICE   SYSAUDIT  SYSDUMP   SYSGEN    SYSHSMS   SYSNAC   SYSPRIV
SYSROOT   SYSSNAP   SYSSPOOL  SYSUSER   TSOS
------------------------------------------------------------------------------
POSIX-USER-LIST                                          END OF DISPLAY
```

S variables

The INFORMATION operand of the command determines which S variables are supplied with values. The following specifications are possible for INFORMATION:

Notation in the command	Abbreviated notation in table
INFORMATION = *ALL	INF=ALL
INFORMATION = *USER-LIST	INF=U-LIST

Please note that S variables are only created if the appropriate conditions are valid (see under column “Condition”).

Output information	**Name of the S variable**	**T**	**Contents**	**Condition**
Comment used as a selection criterion for the BS2000 user ID	var(*LIST).COMMENT	S	*NONE <c-string 1..255>	INF=*ALL
Login directory	var(*LIST).DIR	S	<posix-path-name 1..1023>	INF=*ALL
POSIX group number	var(*LIST).GROUP-NUM	I	<integer 0..60002>	INF=*ALL
Default POSIX group number	var(*LIST).GROUP-NUM-DEF	B	FALSE TRUE	INF=*ALL
Name of the program used as a selection criterion for the BS2000 user ID	var(*LIST).PROG	S	*SHELL <posix-path-name 1..1023>	INF=*ALL
Catalog ID of the pubset	var(*LIST).PUBSET	S	<catid 1..4>	INF=*ALL/ *USER-LIST
BS2000 user ID whose POSIX user attributes are displayed	var(*LIST).USER-ID	S	<name 1..8>	INF=*ALL
	var(*LIST).USER-ID (*LIST)	S	<name 1..8>	INF= *USER-LIST
POSIX user number	var(*LIST).USER-NUM	I	<integer 0..60002>	INF=*ALL
Default POSIX user number	var(*LIST).USER-NUM-DEF	B	FALSE TRUE	INF=*ALL

See the "BS2000 Commands" [28] manual for more information on S variables.

Example 1

```
/declare-var var-name=pos-user-att(type=*struct),multi-elem=*list
/exec-cmd (show-posix-user-attr inf=*all),text-output=*none,structure-output=pos-user-att
/show-var pos-user-att,inf=*par(value=*c-literal)
POS-USER-ATT(*LIST).PUBSET = '1SBZ'
POS-USER-ATT(*LIST).USER-ID = 'TSOS'
POS-USER-ATT(*LIST).USER-NUM = 0
POS-USER-ATT(*LIST).USER-NUM-DEF = FALSE
POS-USER-ATT(*LIST).GROUP-NUM = 0
POS-USER-ATT(*LIST).GROUP-NUM-DEF = FALSE
POS-USER-ATT(*LIST).COMMENT = '*NONE'
POS-USER-ATT(*LIST).DIR = '/'
POS-USER-ATT(*LIST).PROG = '*SHELL'
*END-OF-VAR
/
```

Example 2

```
/exec-cmd (show-posix-user-attr inf=*user-list),text-outp=*none,struct-outp=pos-user-att
/show-var pos-user-att,inf=*par(value=*c-literal)
POS-USER-ATT(*LIST).PUBSET = '1SBZ'
POS-USER-ATT(*LIST).USER-ID(*LIST) = 'TSOS'
*END-OF-VAR
/
```

9.13 SHOW-POSIX-USER-DEFAULTS

Display default values for POSIX user attributes

Domain:	USER-ADMINISTRATION
Privileges:	POSIX-ADMINISTRATION USER-ADMINISTRATION STD-PROCESSING

This command displays the POSIX default attributes in the user catalog of the specified pubset. The following users may execute the command:

- Owners of the POSIX-ADMINISTRATION or USER-ADMINISTRATION privilege for all pubsets.
- Group administrators of the *UNIVERSAL group on the pubset administered by them.

Format

SHOW-POSIX-USER-DEFAULTS

PUBSET = ***HOME** / ***ALL** / list-poss(20): <catid 1..4>

,**OUTPUT** = list-poss(2): ***SYSOUT** / ***SYSLST**(...)

***SYSLST**(...)

| **SYSLST-NUM**BER = ***STD** / <integer 1..99>

| ,**LINE**S-P**ER-PAGE** =**64** / <integer 20..255>

Operands

PUBSET =

Pubset from whose user catalog the POSIX default attributes are to be displayed.

PUBSET = *HOME
The POSIX default attributes are displayed from the user catalog of the home pubset.

PUBSET = *ALL
The POSIX default attributes are displayed from the user directories of all pubsets which are available when the command is entered.

PUBSET = list-poss(20): <catid 1..4>
The POSIX default attributes are displayed from the user catalog of the specified pubset.

OUTPUT =
Determines the system file for the output of information.

OUTPUT = *SYSOUT
The information is output to the SYSOUT system file.

OUTPUT = *SYSLST(...)
The information is output to the SYSLST system file.

SYSLST-NUMBER =
Determines the SYSLST number.

SYSLST-NUMBER = *STD
Determines standard SYSLST output.

SYSLST-NUMBER = <integer 1..99>
Determines the specified SYSLST number.

LINES-PER-PAGE =
Specifies the line count per page.

LINES-PER-PAGE = 64
64 lines per page are printed by default.

LINES-PER-PAGE = <integer 20..255>
The specified number of lines per page is printed.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed without errors
2	0	SRM6001	Command executed with warning
	32	CMD2009	Error while creating the output variable
	32	SRM6020	Command rejected due to system error
	64	OPS0002	K2 interruption during output in S variable
	64	SRM6040	Command rejected with error message
	130	OPS0001	Lack of resources during output in S variable
	130	SRM6030	Command rejected because of unavailable resources

Example

```
/SHOW-POSIX-USER-DEFAULTS PUBSET=A
```

Output:

```
/SHOW-POSIX-USER-DEFAULTS PUBSET=A
POSIX-USER-DEFAULTS   --- PUBSET A                        2009-03-10 14:14:05
------------------------------------------------------------------------------
USER-NUMBER           200
GROUP-NUMBER            8
COMMENT      POSIX public userID
DIRECTORY    /home/usr0/guest
PROGRAM      *SHELL
------------------------------------------------------------------------------
POSIX-USER-DEFAULTS                                          END OF DISPLAY
```

S variables

Output information	Name of the S variable	T	Contents	Condition
Comment	var(*LIST).COMMENT	S	*NONE <c-string 1..255>	
Login directory	var(*LIST).DIR	S	<posix-path-name 1..1023>	
POSIX group number	var(*LIST).GROUP-NUM	I	<integer 0..60002>	
Name of the program	var(*LIST).PROG	S	*SHELL <posix-path-name 1..1023>	
Catalog ID of the pubset	var(*LIST).PUBSET	S	<catid 1..4>	
POSIX user number	var(*LIST).USER-NUM	I	<integer 0..60002>	

See the "BS2000 Commands" [28] manual for more information on S variables.

Example

```
/declare-var var-name=pos-user-def(type=*struct),multi-elem=*list
/exec-cmd cmd=(show-posix-user-defaults pubset=a), -
/         structure-output=pos-user-def,text-output=*none
/show-var var-name=pos-user-def
POS-USER-DEF(*LIST).PUBSET = A
POS-USER-DEF(*LIST).USER-NUM = 100
POS-USER-DEF(*LIST).GROUP-NUM = 1
POS-USER-DEF(*LIST).COMMENT =  System administration
POS-USER-DEF(*LIST).DIR = /home/bs2000
POS-USER-DEF(*LIST).PROG = *SHELL
```

9.14 SHOW-USER-ATTRIBUTES

Output information on user catalog entries

Domain: USER-ADMINISTRATION

Privileges: USER-ADMINISTRATION
SECURITY-ADMINISTRATION
SAT-FILE-MANAGEMENT
SAT-FILE-EVALUATION
STD-PROCESSING
HARDWARE-MAINTENANCE

This command displays the attributes of a BS2000 user ID which were defined by means of the commands /ADD-USER and /MODIFY-USER-ATTRIBUTES.

The POSIX user attributes can be displayed separately by the /SHOW-POSIX-USER-ATTRIBUTES command (see "SHOW-POSIX-USER-ATTRIBUTES").

The following users may execute the /SHOW-USER-ATTRIBUTES command:

- Owners of the USER-ADMINISTRATION privilege for all BS2000 user IDs.
- Group administrators for the BS2000 user IDs which are assigned to and subordinate to their groups.
- Every user for his/her own BS2000 user ID.

Using POSIX does not change anything at the user interface of the /SHOW-USER-ATTRIBUTES command. The command is described in full in the "BS2000 Commands" [28] manual.

> ¡ The /SHOW-USER-ATTRIBUTES command displays the POSIX account number if it was determined. Otherwise, *NONE is output.
>
> The POSIX account number is ignored for groups in the group commands.

S variables

The information on the rlogin account number is stored in the following S variables:

Output information	Name of the S variable	T	Contents	Condition
Account number for POSIX access via rlogin	var(*LIST).ACCOUNT(*LIST).POSIX-RLOG-DEF	S	*NO *YES	INF=*ATTR

See the “Commands (Related publications)” manual [28] for more information on S variables.

9.15 START-POSIX-INSTALLATION

Start the POSIX installation program

Domain:	SYSTEM-MANAGEMENT
Privileges:	TSOS POSIX-ADMINISTRATION

This command starts the POSIX installation program.

Format

START-POSIX-INSTALLATION

INPUT-INTERFACE = ***STD** / ***FHS** / ***FILE**(...)

***FILE**(...)

| **FILE-NAME** = <filename 1..54>

| ,**ERROR-HANDLING** = ***PAR**AMETERS(...)

| ***PAR**AMETERS(...)

| | **RETURNCODE** = ***NO** / ***Y**ES

| | ,**ABORT-ON-WARNING** = ***NO** / ***Y**ES

Operands

INPUT-INTERFACE = *STD / *FHS / * FILE(...)

Specifies whether the installation is to run in interactive mode or in automated mode.

INPUT-INTERFACE = *STD / *FHS
The installation is to run in interactive mode (using FHS masks). See section “POSIX installation program in interactive mode” for details on the installation procedure.

INPUT-INTERFACE = *FILE(...)
The installation is to run in automated mode using the specified parameter file. For details on the structure of the parameter file, see the section “POSIX installation program in batch mode”.

FILE-NAME= <filename 1..54>
Name of the parameter file.

ERROR-HANDLING = *PARAMETERS(...)
Defines the response when errors occur.

RETURNCODE = *NO / *YES
Defines whether, when errors occur, the command is to supply a POSIX-specific command return code (with the maincode POS295x) and trigger the spin-off mechanism.

RETURNCODE = *NO
When errors occur, the spin-off mechanism is not triggered and command return codes are not supplied.

RETURNCODE = *YES
When errors occur, the spin-off mechanism is triggered in procedures and the command supplies return codes (with the maincode POS295x).

ABORT-ON-WARNING = *NO / *YES
Controls the behavior when errors of the class 'warning' occur in the parameter file (with the maincode POS2956).

> When errors of the class 'error' occur, processing of the parameter file is always aborted; when errors of the class 'note' occur, processing is always continued in the next line.

ABORT-ON-WARNING = *NO
Processing of the parameter file is continued in the next line when errors of the class 'warning' occur.

ABORT-ON-WARNING = *YES
Processing of the parameter file is aborted when errors of the class 'warning' occur.

Command return codes

POSIX-specific command return codes (i.e. with maincode POS295x) are returned only if RETURNCODE=*YES was specified.

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No errors
	64	POS2950	Invalid procedure parameter file.
	64	POS2951	Parameter file was not found or is not accessible.
	64	POS2952	The user ID is not authorized to execute the POSIX installation program.
	64	POS2953	Another instance of the POSIX installation program is currently being executed.
	64	POS2954	Installation program cannot be loaded.
	64	POS2955	Serious error in the installation program.
`lnr`	64	POS2956	Error in the parameter file. The number of the line (`lnr`) in or after which the error occurred can be taken from the SC2. In the case of batch installation in online mode, detailed information can be found in the POSIX file `/var/sadm/pkg/insterr`.
	64	POS2957	Timeout when waiting for a POSIX restart. File is locked.

Logging errors in the parameter file (maincode POS2956)

In the case of batch installation in online mode, i.e. not in the case of initial installation or file system extension in offline mode, the following information is written to the POSIX file `/var/sadm/pkg/insterr:`

- Name of the parameter file
- Date and time of installation

In the event of notes / warnings / errors, also:

- Errored line in the parameter file
- Error class (note, warning, error) and error text

Examples:

```
input file : :FR01:$TSOS.POSIX-INSTALL.FS.TMP.TEST
time       : Wed Jan 28 13:17:40 2009
- line   3 : a;$SYSROOT.FS.TMP.TEST;8192;Y;Y;/tmp/test;N;;y;N
  note     : Line 3: BS2000 file already existing, file size may not be changed
  note     : Line 3: file system size of existing filesystem will be used
input file : :V70A:$TSOS.INSTALL.POSIX-SOCKENS
time       : Wed Jan 28 13:27:25 2009
- line   4 : POSIX-SOCKENS;Y
  warning  : IMON-GPN: installation unit not found in SCI
input file : :FR01:$TSOS.POSIX-INSTALL.FIRST
time       : Wed Jan 28 13:38:51 2009
- line 1   : [FirstInstallation]
  error    : POSIX subsystem is available
```

9.16 START-POSIX-SHELL

Start POSIX shell

Domain:	PROCEDURE UTILITIES
Privileges:	STD-PROCESSING

This command starts the POSIX shell. After successfully accessing the POSIX shell the user can enter POSIX commands (see section “Commands belonging to the POSIX shell” and the "POSIX Commands" [1] manual).

The POSIX command for terminating POSIX is *exit*.

Format

START-POSIX-SHELL Alias: **SHELL, POSIX-SH**
VERSION = ***STD** / <product-version without-man-corr>

Operands

VERSION = *STD / <product-version without-man-corr>

Version number of the program to be called (in this case the POSIX shell).
The default value is *STD, i.e. the currently available version is called.

Command return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Without error

Example

See section “Sample session”.

10 Appendix

The appendix contains:

- Privileges in POSIX
- Commands belonging to the POSIX shell
- Daemons in POSIX
- Directories created during an initial installation
- Special files created during an initial installation
- Administration files created during an initial installation
- Tuning measures
- POSIX logging files
- Local time in POSIX

10.1 Privileges in POSIX

This table shows who is authorized to do what in POSIX:

Privilege	Authorization to
Operator (**OPERATING** privilege)	Start POSIX: /START-SUBSYSTEM SUBSYSTEM-NAME=POSIX
	Stop POSIX: /STOP-SUBSYSTEM SUBSYSTEM-NAME=POSIX
Subsystem administrator (**SUBSYSTEM-MANAGEMENT** privilege)	Show POSIX status: /SHOW-POSIX-STATUS
	Start POSIX: /START-SUBSYSTEM SUBSYSTEM-NAME=POSIX
	Stop POSIX: /STOP-SUBSYSTEM SUBSYSTEM-NAME=POSIX
Security administrator (**SECURITY-ADMINISTRATION** privilege)	Grant or withdraw the POSIX-ADMINISTRATION privilege to or from BS2000 IDs (except SYSROOT): /SET-PRIVILEGE /RESET-PRIVILEGE
BS2000 system administrator (**USER-ADMINISTRATION** privilege, but **no** root administrator)	Change values of the tuning parameters in the POSIX information file SYSSSI.POSIX-BC.*version*
	Administer new POSIX users: /ADD-USER and other measures
	Assign individual POSIX user attributes to BS2000 user IDs: /MODIFY-POSIX-USER-ATTRIBUTES
	Administer POSIX user attributes: /MODIFY-POSIX-USER-ATTRIBUTES /SHOW-POSIX-USER-ATTRIBUTES
	Administer default values for POSIX user attributes: /MODIFY-POSIX-USER-DEFAULTS /SHOW-POSIX-USER-DEFAULTS
	Administer access rights for users of remote terminals: /SET-LOGON-PROTECTION /MODIFY-LOGON-PROTECTION /SHOW-LOGON-PROTECTION

	Administer account numbers for system access via remote terminals: /ADD-USER /MODIFY-USER-ATTRIBUTES /SHOW-USER-ATTRIBUTES
	Administer POSIX groups/users: GROUP-NUMBER user attribute
BS2000 system administrator (**USER-ADMINISTRATION** privilege and **also** root administrator)	Perform all tasks of a BS2000 system administrator without root authorization, and also:
	Install POSIX: POSIX installation program
	Create, modify and delete POSIX file systems: POSIX installation program
POSIX administrator (**POSIX-ADMINISTRATION** privilege and also **root** authorization)	Administer new POSIX users /ADD-USER and other measures
	Remove POSIX users /MODIFY-POSIX-USER-ATTRIBUTES
	Create, modify and delete POSIX file systems: POSIX installation program
	Mount and unmount POSIX file systems: *mount* and *mountall; umount* and *umountall*
	Administer POSIX user attributes: /MODIFY-POSIX-USER-ATTRIBUTES /SHOW-POSIX-USER-ATTRIBUTES
	Administer default values for POSIX user attributes: /MODIFY-POSIX-USER-DEFAULT /SHOW-POSIX-USER-DEFAULTS
	Administer BS2000 and POSIX groups: GROUP-NUMBER user attribute, file */etc/group*
Root authorization (**user number 0, group number 0**)	Administer POSIX groups in POSIX: file */etc/group*
	Remove POSIX users: *rmdir*
BS2000 group administrator (group ***UNIVERSAL**)	Administer POSIX user attributes (with restrictions): /MODIFY-POSIX-USER-ATTRIBUTES /SHOW-POSIX-USER-ATTRIBUTES

	Administer default values for POSIX user attributes (with restrictions): /MODIFY-POSIX-USER-DEFAULTS /SHOW-POSIX-USER-DEFAULTS
	Administer access rights for the user of a remote terminal (with restrictions): /SET-LOGON-PROTECTION /MODIFY-LOGON-PROTECTION /SHOW-LOGON-PROTECTION
	Administer account numbers for system access from remote terminals (with restrictions): /ADD-USER /MODIFY-USER-ATTRIBUTES /SHOW-USER-ATTRIBUTES
	Administer POSIX groups/users user attribute GROUP-NUMBER
Nonprivileged POSIX users (privilege **STD-PROCESSING**)	Output information on the entries in the user catalog for their own user IDs /SHOW-USER-ATTRIBUTES /SHOW-POSIX-USER-ATTRIBUTES

10.2 Commands belonging to the POSIX shell

The POSIX shell comprises the basic shell (POSIX-BC) and the extended shell (POSIX-SH). It contains the POSIX commands found in the following table.

Entries in the *Type* column describe the command type:

bin separate module
blt built-in in the shell
scr script

The column *LFS* describes whether the commands can process large POSIX files:

A (large file **a**ware): uses large files correctly
S (large file **s**afe): recognizes large files but rejects processing in a defined manner

The column *L* describes in which product library the command is delivered:

1 SINLIB.POSIX-BC.vvv.SHELL
2 SINLIB.POSIX-BC.vvv.ROOT
3 SINLIB.POSIX-BC.vvv.INET
4 SINLIB.POSIX.BC.vvv.OSSW
5 SINLIB.POSIX-SH.vvv

Name	Location	Type	L	Description	LFS
alias	/usr/bin	blt+scr	1	Define or display alias	
ar	/usr/bin	bin	1	Administer libraries	S
asa	/usr/bin	bin	5	Convert control characters for positioning	
at	/usr/bin	bin	5	Execute commands at a later date	
awk	/usr/bin	bin	5	Programmable processing of text files	A
basename	/usr/bin	scr	1	Separate file name from path	
batch	/usr/bin	scr	1	Execute commands at a later date	
bc	/usr/bin	bin	5	Arithmetic language	
bg	-	blt	1	Process jobs in background	
bs2cmd	-	blt	1	Execute BS2000 command	
bs2cp	/usr/bin	blt+bin	1	Copy BS2000 files	A
bs2do	/usr/bin	bin	1	Calling BS2000 procedures from POSIX	
bs2file	/usr/bin	blt+bin	1	Define file attributes for BS2000 files	
bs2lp	/usr/bin	bin	1	Print files	

bs2pkey	/usr/bin	bin	1	Assign P keys	
cal	/usr/bin	bin	5	Display calendar	
cancel	/usr/bin	bin	5	Delete print jobs	
cat	/usr/bin	blt+bin	1	Concatenate and output files	A
cd	/usr/bin	blt+scr	1	Change current directory	A
chgrp	/usr/bin	blt+bin	1	Change group number of file	A
chmod	/usr/bin	blt+bin	1	Change access rights	A
chown	/usr/bin	blt+bin	1	Change owner of file	A
cksum	/usr/bin	bin	5	Write checksums and file sizes	A
cmp	/usr/bin	bin	5	Compare files character by character	A
comm	/usr/bin	bin	5	Search for identical lines in two sorted files	A
command	/usr/bin	blt+scr	1	Execute simple command	
compress	/usr/bin	bin	5	Compress files	A
cp	/sbin	blt+bin	1	Copy files	A
cp	/usr/bin	blt+bin	1	Copy files	A
cpio	/usr/bin	bin	1	Swap files and directories in and out	A
crontab	/usr/bin	bin	5	Execute commands at regular intervals	
csplit	/usr/bin	bin	5	Split file according to specific criteria	S
cut	/usr/bin	bin	5	Cut bytes, characters or fields from the lines of a file	A
date	/usr/bin	blt+bin	1	Display time and date	
dd	/sbin	bin	5	Copy and convert files	S
debug	/usr/bin	bin	2	Test POSIX programs	
df	/sbin	bin	1	Display number of free and occupied disk blocks	A
diff	/usr/bin	bin	5	Compare files line by line	A
dirname	/usr/bin	scr	1	Separate path prefix from file name	
du	/usr/bin	bin	1	Display occupied memory space	A
dumpfs	/sbin	bin	2	Display internal file system information	A
echo	/usr/bin	blt+bin	1	Output call arguments	
ed	/sbin	bin	5	Line editor in interactive mode	

edt	-	blt	1	Call BS2000 file editor EDT	S
edtu	-	blt	1	Call BS2000 file editor EDT	S
egrep	/usr/bin	bin	5	Find pattern	S
env	/usr/bin	bin	5	Change environment when executing commands	
eval	-	blt	1	Process call arguments and execute them as commands	
ex	/usr/bin	bin	5	Line editor	
exec	-	blt	1	Overlay current shell	
exit	-	blt	1	Terminate shell procedure	
expand	/usr/bin	bin	5	Convert tab character to blanks	S
export	-	blt	1	Export shell variables	
expr	/sbin	blt+bin	1	Evaluate expressions	
expr	/usr/bin	blt+bin	1	Evaluate expressions	
false	/usr/bin	alias+scr	1	Return end status not equal to 0	
fc	/usr/bin	blt+scr	1	Access to history file	
fg	-	blt	1	Bring jobs into foreground	
fgrep	/usr/bin	bin	5	Find strings	S
file	/usr/bin	bin	5	Define file type	A
find	/usr/bin	bin	5	Search directories	A
fold	/usr/bin	bin	5	Split up long lines	S
fsck	/sbin	bin	2	Check consistency of file system and correct interactively with user	A
fsexpand	/sbin	bin	2	Expand existing file systems	A
ftyp	/usr/bin	blt+bin	1	Define types of file processing (BS2000)	
funzip	/usr/local /bin	bin	4	Filter for extracting from a ZIP archive in a pipe	A
fuser	/usr/sbin	bin	2	Display file users	A
gencat	/usr/bin	bin	5	Generate binary coded message catalog	
genso	/usr/bin	bin	2	Generate shared object	
getconf	/usr/bin	bin	5	Call up configuration values	A

getopts	/usr/bin	blt+scr	1	Search procedure arguments for options	
grep	/sbin	bin	1	Get pattern	A
hash	/usr/bin	alias+scr	1	Process shell hash table	
hd	/usr/bin	bin	2	Hex dump	A
head	/usr/bin	bin	5	Output start of a file	A
iconv	/usr/bin	bin	1	Convert code	A
id	/usr/bin	blt+bin	1	Output user identification	
inetd	/usr/sbin	bin	2	Daemon for internet services	
info	/sbin	bin	2	Online diagnostic tool	
ipcrm	/usr/bin	bin	2	Remove setup for interprocess communications	
ipcs	/usr/bin	bin	2	Output state of interprocess communications setup	
jobs	-	blt	1	Output job information	
join	/usr/bin	bin	5	Merge two files according to matching fields	A
kill	/usr/bin	blt+scr	1	Send signals to processes	
last	/usr/bin	bin	2	Display last logged in users	
let	-	blt	1	Integer arithmetic	
lex	/usr/bin	bin	5	Create scanner	
ln	/sbin	blt+bin	1	Enter reference to file	A
locale	/usr/bin	bin	5	Call up information on the locale	
localedef	/usr/bin	bin	5	Define locale	
logger	/usr/bin	bin	5	Log messages	
logname	/usr/bin	bin	5	Query login name	
logrotate	/usr/sbin	scr	2	Change logging files of the syslog daemon	S
lp	/usr/bin	bin	5	Print out files	
lpstat	/usr/bin	bin	5	Output information on print jobs	
ls	/usr/bin	blt+bin	1	Output information on directories and files	A
mailx	/usr/bin	bin	5	Interactively process messages	
make	/usr/bin	bin	5	Manage groups of files	
man	/usr/bin	scr	5	Use online documentation	

mesg	/usr/bin	bin	5	Forbid or permit receipt of messages	
mkdir	/usr/bin	blt+bin	1	Create directory	
mkfifo	/usr/bin	bin	5	Create FIFO	
mkfs	/sbin	bin	2	Create file system	
mknod	/sbin	bin	2	Create device file	
more	/usr/bin	bin	5	Control screen output	A
mount	/sbin	bin	2	Mount file systems and remote resources	
mountall	/sbin	scr	2	Mount two or more file systems	
mv	/sbin	blt+bin	1	Move or rename files	A
newgrp	/usr/bin	blt+bin	1	Modify group membership	
nice	/usr/bin	bin	5	Change priority of commands	
nl	/usr/bin	bin	5	Number text lines	S
nm	/usr/bin	bin	5	Output an object file symbol table	
nohup	/usr/bin	bin	5	Execute command and ignore signals	
od	/usr/bin	bin	5	Output file contents in octal format	S
paste	/usr/bin	bin	5	Merge lines	S
patch	/usr/bin	bin	5	Use difference report	
pathchk	/usr/bin	bin	5	Check path names	
pax	/usr/bin	bin	1	Process portable archives	A
pdbl	/usr/bin	bin	2	Administer private POSIX loader	
ping	/usr/bin	bin	2	Send echo requests to network hosts	
pkginfo	/usr/bin	bin	2	Display information about software packages	
posdbl	/usr/sbin	bin	2	Administer POSIX loader	
pr	/usr/bin	bin	5	Format files and output to standard output	
print	-	blt	1	Output mechanism similar to echo	
printf	/usr/bin	blt+bin	5	Formatted output	
ps	/sbin	bin	1	Query process data	
pwd	/usr/bin	blt+bin	1	Output path name of current working directory	
rcp	/usr/bin	bin	3	Copy files from or to a remote computer	A

read	/usr/bin	blt+scr	1	Read arguments of standard input and assign shell variables	
readonly	-	blt	1	Protect shell variables	
renice	/usr/bin	bin	5	Change priority of current processes	
rm	/sbin	blt+bin	1	Delete files	
rmdir	/sbin	blt+bin	1	Delete directories	A
rmpart	/sbin	bin	2	Remove partition	
rsh	/usr/bin	bin	3	Execute commands on remote computer	
sed	/usr/bin	bin	5	Editor in procedure mode	
set	-	blt	1	Set options or parameters, output variables	
sh	/sbin	bin	1	POSIX shell command interpreter and programming language	A
shift	-	blt	1	Move values of positional parameters to the left	
sleep	/usr/bin	blt+bin	1	Temporarily halt processes	
show_pubset_export	/sbin	scr	2	show file systems affected by EXPORT-PUBSET	
sort	/usr/bin	bin	1	Sort and/or merge files	S
split	/usr/bin	bin	5	Distribute file across several files	A
start_bs2fsd	/sbin	scr	2	start copy daemons	
strings	/usr/bin	bin	5	Find printable strings in object or binary files	A
stty	/usr/bin	bin	2	Output or modify attributes of a data display station	
su	/sbin	bin	2	Substitute user ID	
sum	/usr/bin	bin	5	Calculate checksum of a file	A
sync	/sbin	bin	2	Write back system cache	
tabs	/usr/bin	bin	5	Set tabulator stops	
tail	/usr/bin	bin	5	Output the last part of a file	A
talk	/usr/bin	bin	5	Talk with another user	
tar	/usr/bin	bin	1	Archive files	A
tee	/usr/bin	bin	5	Join pipes together and copy input	
test	/usr/bin	blt-scr	1	Check conditions	
time	/usr/bin	bin	1	Measure command runtime	

times	-	blt	1	Output total runtime of processes started up to now	
touch	/usr/bin	blt+bin	1	Update modification and access times	A
tput	/usr/bin	bin	5	Initialize terminal or query terminfo database	
tr	/usr/bin	bin	1	Replace or delete character	A
trap	-	blt	1	Modify signal handling	
true	/usr/bin	alias+scr	1	Return end status 0	
tsort	/usr/bin	bin	5	Sort topologically	
tty	/usr/bin	bin	5	Output path name of current terminal	
type	/usr/bin	alias+scr	1	Query command type	
typeset	-	blt	1	Set attributes for shell variable	
ulimit	/usr/bin	blt+scr	1	Limit file size for writing or query current limit	A
umask	/usr/bin	blt-scr	1	Output or modify default allocation of access rights	
umount	/sbin	bin	2	Unmount file systems and remote resources	
umountall	/sbin	scr	2	Unmount two or more file systems	
unalias	/usr/bin	blt-scr	1	Delete variables from alias tables	
uname	/sbin	bin	1	Output basic data on the current operating system	
uncompress	/usr/bin	bin	5	Uncompress compressed files	A
unexpand	/usr/bin	bin	5	Convert blanks to tab characters	S
uniq	/usr/bin	bin	1	Search for repeated lines	
unzip	/usr/local/bin	bin	4	List, test and extract compressed files in a ZIP archive	A
unzipsfx	/usr/local/bin	bin	4	Self-extracting stub for prepending to ZIP archives	A
uudecode	/usr/bin	bin	5	Decode file after mailx transfer	
uuencode	/usr/bin	bin	5	Encode file for mailx transfer	
usp	/usr/bin	bin	2	Dynamisc setting of POSIX control parameters	
vi	/usr/bin	bin	5	Screen editor	
wait	/usr/bin	blt+scr	1	Wait for termination of background processes	
wc	/usr/bin	bin	5	Count words, characters and lines	A
whence	-	blt	1	Command localization	

who	/sbin	bin	5	Show active user IDs	
write	/usr/bin	bin	5	Send message to a user	
xargs	/usr/bin	bin	5	Create argument list(s) and execute command	
yacc	/usr/bin	bin	5	Create parser	
zcat	/usr/bin	bin	5	Output compressed files	A
zip	/usr/local/bin	bin	4	Package and compress (archive) files	A
zipcloak	/usr/local/bin	bin	4	Encrypt entries in a zipfile	A
zipgrep	/usr/local/bin	bin	4	Search files in a ZIP archive for lines matching a pattern	A
zipinfo	/usr/local/bin	bin	4	List detailed information about a ZIP archive	A
zipnote	/usr/local/bin	bin	4	Write the comments in zipfile to stdout, edit comments and rename files in zipfile	A
zipsplit	/usr/local/bin	bin	4	Split a zipfile into smaller zipfiles	A

10.3 Daemons in POSIX

The following table contains a list of all daemons supplied with POSIX.

Name	Location	Type	Delivery	Description
bs2fsd	/usr/sbin	bin	SINLIB.POSIX-BC.ROOT	Copy daemon for bs2fs file systems
cron	/usr/sbin	bin	SINLIB.POSIX-SH	Daemon to execute commands at specific times
fsmond	/usr/sbin	bin	SINLIB.POSIX-BC.ROOT	Daemon for monitoring file systems
in.rlogind	/usr/sbin	bin	SINLIB.POSIX-BC.ROOT	Server for rlogin (remote login)
in.rshd	/usr/sbin	bin	SINLIB.POSIX-BC.INET	Server for rsh (remote shell)
in.talkd	/usr/sbin	bin	SINLIB.POSIX-SH	Server for talk (interaction with another user)
in.telnetd	/usr/sbin	bin	SINLIB.POSIX-BC.INET	Server for telnet (psuedo terminal protocol)
inetd	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Daemon for internet services
rpcbind	/usr/sbin	bin	SINLIB.POSIX-BC.ROOT	Daemon for RPC protocol
shmd	/usr/sbin	bin	SINLIB.POSIX-BC.ROOT	Shared memory daemon
syslogd	/usr/sbin	bin	SINLIB.POSIX.BC.ROOT	syslog daemon

10.4 Directories created during an initial installation

The following directories are created in the course of initial installation:

/**dev** with the subdirectories:

> /dsk, /fd, /pts /rdsk, /sad /sf and /term

/**etc** with the subdirectories:

> /default, /dfs, /fs, /inet, /init.d, /net, /net/ticlts, /net/ticots, /net/ticotsord, /sm, /sm.d, /.products and /.products/.legit

/**home**

/**lost+found**

/**proc**

/**sbin**

/**tmp**

/**usr** with the subdirectories:

> /bin, /lib, /lib/iconv, /lib/lex, /lib/nfs, /sbin and /ucb

/**var** with the subdirectories:

> /adm, /lp, /mail, /preserve, /spool, /spool/lp/temp, /sadm/pkg and /tmp

These directories are created by the POSIX installation program and must not be edited or modified by the POSIX administrator.

10.5 Special files created during an initial installation

The following special files are created in the course of initial installation:

/dev/console

/dev/kmem

/dev/log

/dev/loop

/dev/null

/dev/dsk/0s0

/dev/rdsk/0s0

/dev/root

/dev/rroot

/dev/ptmx

/dev/sf/mmm 000 <= mmm < 2000

/dev/ticlts

/dev/ticots

/dev/ticotsord

/dev/tty

/dev/zero

/dev/sad/admin

/dev/sad/user

/dev/ptmx

/dev/pts/mmm 000 <= mmm < 1024

/dev/term/mmm 000 <= mmm < 1024

These special files are created by the POSIX installation program and must not be edited or modified by the POSIX administrator.

10.6 Administration files created during an initial installation

The following administration files are created in the course of initial installation:

/etc/dfs/dfstab

/etc/dfs/fstypes

/etc/dfs/sharetab

/etc/group

/etc/inetd.conf

/etc/mnttab

/etc/net/ticlts/hosts

/etc/net/ticlts/services

/etc/net/ticots/hosts

/etc/net/ticots/services

/etc/net/ticotsord/hosts

/etc/net/ticotsord/services

/etc/netconfig

/etc/partitions

/etc/print

/etc/protocols

/etc/profile

/etc/services

/etc/syslog.conf

/etc/termcap

/etc/TIMEZONE

/etc/vfstab

As regards content and meaning, the tables created in /etc are the counterparts of those in Reliant UNIX V5.45.

10.7 Tuning measures

The following table contains measures that are recommended for enhancing the POSIX performance.

Measure	Comments
Using the POSIX loader *posdbl*	A 30 MB memory pool is recommended (DBLPOOL parameter in the information file) Note: The PAGING AREA must be capable of accepting this additional address space. For program production: Compiler call is explicitly included in posdbl using `posdbl -b /usr/bin/c89` or `posdbl -b /usr/bin/cc`
Using current subsystems	The following subsystems should be started for the shell commands and for executing user programs: For CRTE: CRTEC, CRTEPART For CRTE-BASYS: CRTEBASY For using EDT: EDT, EDTU, EDTCON For program production: BINDER, PMLOG, CPP
Using DAB	The following files should be held in the cache: SYS.AIDIT0 (approx. 30 PAM pages): cache completely
Using the private POSIX loader *pdbl*	see "POSIX Commands" [1] manual
Increasing control parameters in the POSIX information file (see "POSIX information file")	In event of high I/O load: HDSTNI Number of server tasks for performing asynchronous I/Os NPBUF Maximum number of I/O buffers for physical I/Os. This value should be at least 4 * HDSTNI.

10.8 POSIX logging files

Messages of the POSIX programs and daemons are logged by calling the syslog daemon (*syslogd*) via the CRTE interface *syslog()*. The syslog daemon logs the messages in one or more logging files. The behavior of the syslog daemon can be controlled using the */etc/syslog.conf* file. By default the syslog daemon log all messages in the */var /adm/syslog* file.

The syslog daemon corresponds to Version 1.167 of the FreeBSD implementation (see http://www.freebsd.org). In addition to adjustments to the general conditions of POSIX/BS2000, the following functions have been removed:

- Reception of messages from remote computers
- Sending messages to remote computers
- Writing messages on terminals
- Writing messages on command pipes

The licencsing conditions are contained in section “Licensing conditions for the FreeBSD implementation”.

If POSIX programs with an obsolete runtime system (CRTE) are also executed on the system, their messages are logged in the */var/adm/messages* file. In this case the */var/adm/messages* file should not be assigned as the logging file for the syslog daemon because the *logrotate* command will then not function correctly.

Each time the POSIX subsystem is restarted new logging files are created after the previous files have been renamed (suffix ".0" through ".3"). The logging files can also be changed during ongoing operation using the *logrotate* command (see the manual "POSIX Commands" [1]).

10.8.1 Licensing conditions for the FreeBSD implementation

```
Copyright (c) 1983, 1988, 1993, 1994
     The Regents of the University of California.  All rights reserved
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
4. Neither the name of the University nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

10.9 syslogd - syslog daemon for logging system messages

The *syslog* daemon reads and logs messages in logging files in accordance with the definitions in its configuration file (*/etc/syslog.conf*).

Syntax

```
syslogd [-Cduv][-f config_file][-l [mode:]path][-m mark_interval]
```

-C

Logging files which do not yet exist are created (with the rights 0600).

> If this option is not specified, the logging files must be created before the syslog daemon is started as the syslog daemon will otherwise not log any messages.

-d

Starts the syslog daemon in debug mode, e.g. to diagnose problems.

-f `config_file`

Specifies an alternative configuration file; the default configuration file is */etc/syslog.conf*.

-m `mark_interval`

Interval (in minutes) at which the syslog daemon writes so-called *MARK* messages to the logging file.

These are used to document in the logging file that the syslog daemon is active.

The default setting is 20 minutes, but the output of *MARK* messages must also be activated in the configuration file.

-l `[mode:]path`

Path name under which the syslog daemon is to create an additional socket. The path must be specified in absolute terms.

The main task of this option is to create additional log sockets to*/var/run/log* for different *chroot* environments. Access rights of the sockets can be specified separately ahead of the socket name in the form of octal numbers separated by a colon (e.g. `0640:/rootjail/var/run/log`).

-u

Unique priority logging

This option changes the default value for the relational operator of the *priority* specification in the configuration file from **=>** to **=** (see "syslog.conf - syslogd configuration file ").

–v

Verbose logging

This option controls the logging of the origin (*facility*) and/or weight (*priority*) of the message. If this option is not specified, neither of the two attributes is logged.

When the option is specified **once**, the message is prefixed with numerical values of both attributes (e.g. <1.4>).

When the option is specified **twice**, the message is prefixed with the symbolic values of both attributes (e.g. <user.warn>).

When the option is specified **three times**, the message is prefixed with the symbolic value of the priority in compatible form (e.g. LOG_WARNING).

Hint

The *syslog* daemon is started automatically with the **-Cvv** options by the */etc/rc2.d/S001syslog* script when POSIX starts and by default logs all messages in the */var/adm/syslog* file. This corresponds to the `*.* /var/adm/syslog` entry in the configuration file */etc/syslog.conf*. Additional or alternative logging files can be defined in this file.

To enable changes to the syslog configuration file to take effect immediately, a hangup signal must be sent to the syslog daemon (e.g. by calling `/etc/rc2.d/S001syslog restart`). It will then close all logging files, read in the configuration file, and open the logging files contained in the configuration file.

The *syslog* daemon reads messages from the UNIX domain socket */var/run/log* and, if required, from the alternative sockets specified with the *–l* option.

The *syslog* daemon creates a */var/run/syslog.pid* file which contains its process ID.

The current status of the syslog daemon, its process ID and the logging files used can be determined by calling `/etc/rc2.d/S001syslog status`.

File

/etc/syslog.conf

Default configuration file

/var/run/syslog.pid

File for the process ID

/var/run/log

Name of the UNIX domain socket

/var/run/logpriv

UNIX domain socket for privileged applications

/var/adm/syslog

Default logging file

/etc/rc2.d/S001syslog

Start script

/etc/rc0.d/K999syslog

Stop script

BS2000

Kernel logging is not supported in the syslog daemon. This is done directly on the BS2000 console.

The following features in the configuration file are not supported and are rejected with a console message:

- Logging from and to remote hosts (remote syslog daemons)
- Logging to command pipes or user terminals

Logging to the system console by specifying the LOG_CONS option when the *syslog()* function is called is not supported. This option is ignored.

Only the system administrator can enable messages to be logged to the BS2000 console or BS2000 CONSLOG file by specifying output file */dev/console* or */dev/conslog* for the messages concerned.

10.10 syslog.conf - syslogd configuration file

syslog.conf is the configuration file for the syslog daemon *syslogd*. It controls which messages the syslog daemon writes to which logging files.

The file consists of blocks which are separated by *program* specifications. Each line defined in a block contains two specifications:

- The *selector* specification selects messages on the basis of their type and priority.
- The *action* specification defines the action which the syslog daemon is to execute for the messages selected with the *selector* specification.

The *selector* specification is separated from the *action* specification by at least one tabulator character or blank.

> **i** When blanks are used, the compatibility of your configuration file with other Unix-type systems is not guaranteed if the latter only permit tabulator characters.

program specification

A *program* specification introduces a block which controls the logging of messages only for programs which are defined in this *program* specification. A block at the start of the *syslog.conf* file without a preceding *program* specification controls logging for any programs.

Syntax

```
[#]![+|-]program[,program] ...
```

!

Indicates the *program* specification. The optional specification of **#** is supported for compatibility reasons.

+|-

Positive or negative selection (default: positive)

In the case of positive selection (+) the specifications contained in the current block are evaluated only for messages of programs which are included in the *program* specification,
and in the case of negative selection (-) only for messages of programs which are not included in this specification.

`program`

Program name or ***** (asterisk for all programs)

When *syslog()* is called, a check is made to see whether the specified name (or one of the specified names) matches the *ident* specification of the *openlog()* function of the program from which the *syslog()* call is made.

selector specification

The *selector* specification is used to select the messages for which the actions defined in the *action* specification are to be executed. The *selector* specification is structured as follows (without blanks):

- *facility* specification
- Dot (.)
- Optional negation
- Optional relational operands
- *priority* specification

Syntax

```
facility.[[!] [=>|>=|>|=|<|=<|<=]]priority
```

`facility`

Only messages which are sent by the specified part of the system are selected.

The following specifications are possible: **auth**, **cron**, **daemon**, **lpr**, **mail**, **mark**, **news**, **syslog**, **user**, **uucp**, **local0** through **local7** and *****.

These keywords (with the exception of **mark**) match the LOG_ values from */usr/include/sys/syslog.h* which can be specified in the *openlog()* function.

The special *facility* specification **mark** causes so-called *MARK* messages to be written periodically (default: every 20 minutes, see the *-m* option of the syslog daemon on "syslogd - syslog daemon for logging system messages ") to the logging file.

The specification * (asterisk) stands for any *facility* specifications except for **mark**.

> The *facility* specification is case-sensitive.

!

Negation. The following specification (relational operator and *priority*) is logically inverted.

=>|>=|>|=|<|=<|<=

The relational operators together with the subsequent *priority* specification select the messages on the basis of their priority.
If no relational operator is specified, the behavior depends on the *-u* option when the syslog daemon is started (see "syslogd - syslog daemon for logging system messages "). By default (without *-u* option) messages are selected whose priority is greater than or equal to the specified priority (corresponding to **>=** or **=>**). When the *-u* option is specified, only messages with the specified priority are selected (corresponding to **=**).
Relational operators with a preceding "!" are logically inverted. For example, "!=info" means that all messages with a priority other than "info" are selected.

`priority`

Only messages whose priority is the same as the specified priority (or, depending on the relational operator, higher or lower) are selected.

The following specifications are possible (beginning with the highest and ending with the lowest error weight: **alert**, **crit**, **err**, **warning**, **notice**, **info** and **debug**.
These keywords match the LOG_ values from */usr/include/sys/syslog.h* which can be specified in the *syslog()* function.

The special *priority* specification **none** causes the *facility* specification concerned to be excluded from the action defined.

The *priority* specification ***** (asterisk) selects all priorities.

> The *priority* specification is case-sensitive.

A *selector* specification can contain more than one *facility* specification and also more than one *priority* specification. Each of these is separated by a comma (",").

Multiple *selector* specifications can also be combined with one *action* specification. Each of these is then separated by a semicolon (";"). Here it is important to note that each *selector* specification can modify the definition which was specified beforehand.

action specification

The *action* specification defines which actions are executed for the messages which were selected with the *selector* specification and which match the *program* specification of the current block. The *action* specification has the following structure:

Syntax

```
[-]pathname
```

-

After each message has been written, *fsync()* is called, i.e. messages are synchronized explicitly. By default messages are not synchronized explicitly.

`pathname`

The messages are logged (attached) to the file with the specified path name. The path name must be specified with a / (slash) as the first character.

> Output of the messages to other destinations such as command pipes, terminals or remote computers is, in contrast to other syslogd implementations, not supported.

Comments

Comments are introduced by a hash mark (**#**). The characters in the line, including the **#**, are ignored, except in the case of *program* specifications. Empty lines and lines which contain only blanks or tabulator characters ahead of the first hash mark are totally ignored. The comment function of the **#** character can be canceled with \ (backslash).

File

/etc/syslog.conf

Configuration file for syslogd

BS2000

The following features are not supported and are rejected with a console message:

- Logging from and to remote hosts (remote syslog daemons)
- Logging to command pipes or user terminals

When logging to a BS2000 file (bs2fs file) it must be noted that the bs2fs file system in which this file resides must alredy be mounted when the syslog daemon starts. The automatic *mount* can be used for this purpose by means of an entry in */etc/vfstab*, e.g.:

```
/etc/vfstab
:FR07:$SYSROOT.POSIX.SYSLOG*  -  /var/adm/SYSROOT  bs2fs  1  yes  ftyp=text
/etc/syslog.conf
*.*    /var/adm/SYSROOT/POSIX.SYSLOG
```

The following special aspects must be taken into account here:

- The syslog daemon is started under the BS2000 user ID SYSROOT. The bs2fs files must therefore reside in this user ID.
- From the BS2000 viewpoint the current logging file remains locked until the syslog daemon has been terminated or until *logrotate*. Until this has happened it can only be read from POSIX.

Example 1

All messages are output to the */var /adm/syslog* file. With the exception of mail messages, messages with the weight *err* or higher and all authentication messages with the weight *notice* or higher are also output to the BS2000 CONSLOG file.

```
*.*                             /var/adm/syslog
*.err;auth.notice;mail.none     /dev/conslog
```

Example 2

With the exception of mail messages and authentication messages, all messages with the weight *info* or higher are output to the file.

```
*.info;mail.none;auth.none      /var/adm/syslog
```

Example 3

Daemon messages (only) with the weight *debug* are output to the file.

```
daemon.=debug                   /var/adm/daemon.debug
```

Example 4

Mail messages and authentication messages are output to specific files. All other messages are output to the syslog file.

```
*.*;auth.none;mail.none         /var/adm/syslog
mail.*                          /var/adm/maillog
auth.*                          /var/adm/authlog
```

Example 5

Mail and news messages with the weight *err* and higher are output to the file.

```
mail,news.err                   /var/adm/mailerr
```

Example 6

Messages of the *postfix* program are output to the mail logging file.

```
!postfix
*.*                                /var/adm/maillog
```

Example 7

Messages of the *named* and *dnsd* programs are output to a separate file, and this is synchronized with *fcync()* after each individual message.

```
!named,dnsd
*.*                               -/var/adm/dnslog
```

Example 8

MARK messages are logged.

```
mark.*                             /var/adm/syslog
```

10.11 Local time in POSIX

The information on the **local** time supplied by POSIX interfaces can, under certain circumstances, differ from the information supplied by the corresponding BS2000 interfaces. This does not, however, affect the information on the Coordinated **Universal** Time (UTC).

The reason for this difference is that BS2000 interfaces and POSIX interfaces use separate mechanisms for "localizing" the time.

In **BS2000** the settings which are defined in the PARAMS.GTIME file at system startup apply for the local time.

Example:

```
/BS2000 PARAMS
/BEGIN GTIME
ZONE=+01:00
DIFF=1:00
SEASON=S
CHDATE=1900-01-01/00:00
CHDATE=1980-04-06/02:00
CHDATE=1980-09-28/03:00
...
CHDATE=2020-03-29/02:00
CHDATE=2020-10-25/03:00
CHDATE=2021-03-28/02:00
CHDATE=2021-10-31/03:00
CHDATE=2022-03-27/02:00
CHDATE=2022-10-30/03:00
CHDATE=2023-03-26/02:00
CHDATE=2023-10-29/03:00
CHDATE=2024-03-31/02:00
CHDATE=2024-10-27/03:00
NEXTZONE
...
/EOF
/END-PARAMS
```

In **POSIX** the settings which are defined in the environmental variable *TZ* when the corresponding CRTE interfaces are called apply for the local time.

Example (default setting = Central Europe):

```
$ echo $TZ
MEZ-1MSZ-2,M3.5.0/02:00:00,M10.5.0/03:00:00
$
```

The content of the *TZ* variable in this example must be interpreted as follows:

`MEZ-1` Default time zone

Name	MEZ
Difference	MEZ - 01:00:00 = UTC

`MSZ-2` Alternative time zone

Name	MSZ
Difference	MSZ - 02:00:00 = UTC

`M3.5.0/02:00:00` Time for switching to the alternative time zone

Month	3 = March
Week	5 (or 4 if the month does not have 5 weeks)
Weekday	0 = Sunday
Time	02:00:00

`M10.5.0/03:00:00` Time for switching back to the default time zone

Month	10 = October
Week	5 (or 4 if the month does not have 5 weeks)
Weekday	0 = Sunday
Time	03:00:00

Supplying values for the *TZ* variable in POSIX

When POSIX-BC is installed, the */etc/TIMEZONE* file is intalled with the following content:

```
TZ=MEZ-1MSZ-2,M3.5.0/02:00:00,M10.5.0/03:00:00
```

A shell script is concerned here which sets the *TZ* variable in the calling shell if it is called using a dot command (`. /etc/TIMEZONE`). The *TZ* variable is automatically exported, i.e. it is automatically propagated to child processes.

The */etc/TIMEZONE* script is executed with every type of POSIX login, in other words also when opening a dialog or batch session with /START-POSIX-SHELL, and in the case of *rlogin*, *telnet*, *rsh* and *ssh*. It is also executed by the RC scripts before daemons are started.

The *TZ* variable is thus supplied with a value in all daemons and login scripts and also in all processes these generate with *fork()*. It thus controls the CRTE functions for determining the local time.

> The */etc/TIMEZONE* file is also reinstalled in the event of an upgrade installation of POSIX-BC. As a result, any changes the administrator may have made are overwritten.

Restrictions for the POSIX mechanism with the *TZ* variable

The POSIX mechanism permits only one fixed rule for changeover points between the default and alternative time zones and no variations from year to year. This complies with the currently applicable EU regulations.

Consequently timestamps which are earlier than 1996, i.e. which were specified before a fixed rule was introduced for changeover times, can under certain circumstances be incorrectly “localized”.

11 Glossary

All important terms used in this manual are listed and defined in alphabetical order in this glossary. For terms specific to operating systems, the environment from which they are taken is specified (BS2000, POSIX or UNIX). Cross-references to other terms are given in *italics*.

64-bit file interfaces

Set of new file interfaces with which *large POSIX files* can be processed. These file interfaces have been derived from existing file interfaces and have the suffix 64, e.g. open64().

64-bit integer arithmetic

64-bit integer arithmetic uses integers which are 64 bits in length (without sign) or 63 bits and a sign bit. It is implemented using a pair of 32-bit integers (data type long long).

absolute path name

POSIX/UNIX: *Path name* which begins at the *root directory (/)* of the *POSIX file system* and leads through all superordinate directories to a specific *file* or *directory.* Every file and directory has a unique absolute path name.

access permission

POSIX/UNIX: Property of a *file* which controls access to this file. Access permissions are assigned separately to the owner (see *file owner class*), the owner group (see *user class group*), and all other users (see *file group class*). There are three basic access permissions: read, write and execute permission.

account number

BS2000: Refers to an account for the associated *user ID*. The same account number can be assigned to two or more user IDs. A user ID can have a maximum of 60 account numbers. The account number is evaluated for SET-LOGON-PARAMETERS and ENTER-JOB.

Application Programming Interface (API)

Interface between the applications and the system and subsystem functions they use.

ASCII

Abbreviation for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.
Standardized code for the conversion of uppercase and lowercase letters, digits, special and control characters into digital numbers which can be processed in the computer. *UNIX* and *SINIX* work with the ASCII code. POSIX can process ASCII data after it has been converted to EBCDIC format.

authentication

BS2000: Check of the *user* specifications during system access. The user attributes “*User ID*” and “*Password*” are checked against the entries in the *user ID catalog.*

background process

Process which does not completely exhaust the resources of the computer, but which facilitates the simultaneous execution of other processes. A background process normally utilizes the time slots during which the processor is less active.

block-mode terminal

Terminal which does not support character-by-character input and output operations.

bs2fs file system

Selectable set of files in BS2000 which are made available in POSIX as a file system, thus enabling them to be accessed using POSIX means (commands, program interfaces). The files are selected using the user and catalog IDs and wildcard symbols.

catalog identification, catid

BS2000: Identifier of a *pubset*.
The catalog identifier consists of a maximum of 4 characters, which within file names or *path names* must be enclosed in inverted commas. The catalog identifier is a component of the *path name* of a *file*.

child process

See *fork*

container file

POSIX: BS2000 PAM file in which a *POSIX file system* is stored. A container file is stored on a pubset. Container files and BS2000 files may be located on the same pubset.

current directory

POSIX/UNIX: *Directory* in which the *user* is currently working. It can be displayed by means of the POSIX command pwd. In the current directory, the user can access all *files* and *subdirectories* directly.

daemon

POSIX/UNIX: System process which runs permanently and usually in the background and performs general functions. The best-known example is the printer daemon, which ensures that a file is printed while the user continues to work.

directory

POSIX/UNIX: A directory is used to group and organize *files* or directories.

EBCDIC

Abbreviation for **E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode.
EBCDI code is an 8-bit extended BCD code which is used on BS2000 computers, TRANSDATA communication computers and industrial standard machines.

file

UNIX: A file is identified in *UNIX* via an index entry. This entry contains information as to whether the file is a *normal file*, a *special file* or a *directory.* A normal file contains text, data, programs or other information. A special file refers to a device or part of a device, such as a drive or a hard disk partition for example. A directory contains other files.

BS2000: Records related to each other are combined in a named unit, the file.
Examples of files are conventional I/O program data, load modules, and text information which is created and processed with a file editor.

file group class

POSIX/UNIX: A *process* belongs to the file group class if it does not belong to the *file owner class*, and if the effective *group number,* or one of the additional group numbers of the process matches the group number of the files.

file other class

POSIX/UNIX: Property of a *file* which displays the *access permissions* of a *process* which is associated with the user ID of a process. A process belongs to the file other class of a file if it does not belong to the *file owner class* or the *file group class*.

file owner class

POSIX/UNIX: Property of a *file* which displays the *access permissions* of a *process* which is associated with the user and group identification of a process. A process belongs to the file owner class of a file, if the effective *user number* of the process matches the user number of the file.

file system

POSIX/UNIX: Collection of *directories* and *files* and specific file attributes.

first start

BS2000: The system files are created following first start of BS2000. The system assigns a range of *user IDs*, e.g. TSOS, SYSPRIV and SYSHSMS. The *user ID catalog* is always created in the course of a first start.

foreground process

Process which completely monopolizes the computer capacity, so that it cannot be used by other processes.

fork

POSIX/UNIX: System call which divides a *process* into two parts, the *parent process* and the *child process.*

group member

BS2000: *User ID* assigned to a *user group*. The group administrator can assign resources to a group member.

group ID

Non-negative whole number for identifying a group of users. Each user is a member of at least one group.

Large POSIX files

POSIX files which may be larger than 2 gigabytes (2 gigabyte = 2^{31} -1 bytes). *64-bit arithmetic* is required for addressing within a file of this size. Large POSIX files can only be created in *large POSIX file systems* and must be processed with *64-bit file interfaces*.

Large POSIX file systems

POSIX file systems which can be larger than 2 gigabytes. *64-bit arithmetic* is required for addressing within a file system of this size. The maximum size of a large POSIX file system is 1024 Gbyte. Large POSIX file systems are a prerequisite for *large POSIX files*.

home directory

POSIX: *Directory* into which the *user* is automatically placed when he/she connects with POSIX.

host

Central computer of a *computer network*. Programs are run, files are stored, and I/O is controlled on the host. A powerful computer network often contains several hosts.

job variable

BS2000: Job variables are storage areas for the exchange of information between jobs, as well as between the *operating system* and jobs. They have a name, and also contents (value). The contents can be used for controlling jobs and programs.
The *user* can create, modify, query and delete job variables. He/she can also instruct the operating system to change the setting of a monitoring job variable if the status of a job or a program is modified.

join file

BS2000: *File* which contains the *user attributes* of all the *user IDs* of a *pubset.*

kernel

POSIX/UNIX: Code of the POSIX */UNIX operating system*.

large file aware

A program is "large file aware" if it processes *large POSIX files* correctly.

large file safe

A program is "large file safe" if it recognizes *large POSIX files* and rejects processing in a defined manner, e.g. with an appropriate message.

login directory

See *home directory*

mount point

POSIX/UNIX: Name of a directory under which a remote resource such as a file tree is mounted.

mounting file systems

POSIX/UNIX: *File systems* can be mounted in a local file system by means of the command “mount”. The *mount point* must first be defined as a *directory.* This directory is no longer visible once the file system has been mounted.

Network File System (NFS)

BS2000: Software product which facilitates distributed data storage in a heterogeneous *computer network* . Thanks to this facility, the *user* can access remote *files* as if they were available on her/his *local computer.*

operating system

All software and firmware programs which make it possible to operate a computer without requiring tailoring to a specific application. As a rule, the operating system is supplied by the computer manufacturer.

parent process

See *fork*

password

BS2000: String of characters which the *user* must enter in order to receive access to a *user ID*, a *file*, a *job variable*, a network node or an application.

path name

POSIX/UNIX: Every *file* and every *directory* has a unique path name. The path name specifies the position of the file and directory within the *file system* and shows how it can be accessed. The path name consists of the names of all superordinate directories, starting from the top of the file system, and the name of the file or directory itself. In each case, the names of the directories are separated from each other by a slash.
(Example: /dir1/dir2/protocol)
A distinction is drawn between *absolute* and *relative path names.*

BS2000: Every file cataloged in BS2000 is also clearly identifiable by a path name. The path name consists of the *catalog identifier* (catid), the *user ID* (userid) and a fully qualified file name assigned by the user: :catid:$userid.filename

permission bits

POSIX/UNIX: Information on read, write or execute rights for a given file. The bits are divided into three sections: owner, group and other users.

pipe

POSIX/UNIX: Concatenation of two *POSIX/UNIX* commands. A pipe causes the output of one program to become the input of the next program, with the result that the programs are executed in sequence. A pipe is created when the pipe icon | is specified after the first command. The output of the process to the left of the pipe icon is directed to the *process* to the right of the pipe icon.

Portable Open System Interface for UNIX (POSIX)

Interface standards for open systems which were defined by the IEEE and are based on UNIX. POSIX contains standards for a wide spectrum of operating system components, beginning with the C programming language, right up to system administration. Among other things, POSIX specifications include:

1003.00 Guide to POSIX Open System Environment

1003.01 System Application Program Interface (API)

1003.02 Shell and Utilities

1003.07 System Administration

POSIX administrator

Holder of the POSIX-ADMINISTRATION privilege. This privilege is linked automatically to the SYSROOT system ID and cannot be withdrawn from SYSROOT. The security administrator can also grant this privilege to and withdraw it from other *user IDs*.

POSIX file system

BS2000/POSIX: *File system* in BS2000 with the UNIX file structure system (UFS). Like UNIX, it can consist of several file systems. It is structured hierarchically and consists of *directories* and *files* (POSIX files). The *root* directory, which is marked by a slash (/), is located at the top of the hierarchy. The directory structure branches downwards from here. Other files and directories can be branched to from a directory. However, branching from a file is not possible. Every file of a file system can be accessed via precisely one path of the file system.
The difference between a POSIX file system and a UNIX file system is the storage location. In the case of a UNIX file system, the storage location is a physical device, in the case of a POSIX file system, it is a PAM *container file*.

POSIX shell

BS2000/POSIX: Ported SINIX *system program* which handles communication between the *user* and the system. The POSIX shell is a command interpreter. It compiles the input POSIX commands into a language which the system recognizes.
If the POSIX shell is entered for the user attribute “Program”, the POSIX shell is started as soon as the user is connected to POSIX via remote login to POSIX.

public volume set

BS2000: Set of hard disks marked as belonging to the same group. MPVS systems work with several mutually independent pubsets.

pubset

BS2000: Abbreviated form of *public volume set*

regular file

File which is a freely accessible sequence of bytes without any further structure defined by the system.

relative path name

POSIX: Access path for a *file* or a *directory* which starts from the position of the *current directory* within the *file system*. Relative path names do not begin with a slash (/).

Reliant UNIX

Successor to *SINIX*, version 5.43 of which was renamed to Reliant UNIX as a result of the fusion between the former Siemens Nixdorf and Pyramid Technology UNIX versions.

root

POSIX/UNIX: User name (system administrator ID) with the most privileges.

root authorization

ID assigned the *user number* 0 and the *group number* 0. The system ID SYSROOT has the root authorization by default.

root directory

POSIX/UNIX: Main directory in a hierarchically structured *file system* from which all other *directories* branch.

security attributes

BS2000: Attributes of an object relevant to security *(file*, *job variable* etc.) which determine the type and possibility of data access to this object.
The following security attributes exist for files, for example: ACCESS/USER-ACCESS, SERVICE bit, AUDIT attribute, RDPASS, WRPASS, EXPASS, RETPD, BACL and GUARD.

shareable file

BS2000: A *file* which the *user* cataloged with the operand USER-ACCESS=*ALL-USERS. Files which are marked as shareable in this way can be invoked by all users. However, the user is required to know the *user ID* of the generator of the file and, if necessary, to specify the *password* if the file is password-protected.

shutdown

BS2000: Procedure for an orderly system termination sequence, including the saving of specific system files.

Sockets

Interface for network access via TCP/IP

special file

File, also referred to as a device driver, that is used as an interface to an I/O device (e.g. terminal, disk drive).

startup

BS2000: Loading the BS2000 *operating system* software. There are three variants:

AUTOMATIC-STARTUP

DIALOG-STARTUP

FAST-STARTUP

These variants differ in degrees of automation.

subdirectory

POSIX/UNIX: *Directory* subordinate to a directory on the next highest level of the *file system*.

system privileges

BS2000: All privileges which can be assigned by means of the /SET-PRIVILEGE command, as well as the privilege of the security administrator and the TSOS system ID.

UNIX File System (UFS)

File system variant of the Virtual File System for the administration of local file systems.

UNIX operating system

An interactive *operating system*, developed in 1969 by Bell Laboratories. Since only a central *kernel* of UNIX is hardware-dependent, UNIX is used on many different systems from different manufacturers.

user

BS2000: Represented by a *user ID.* The term “user” is a synonym for persons, applications and procedures etc. which can obtain access to the *operating system* via a user ID.

user administration

BS2000: Involves administration of *user IDs* and *user groups* in relation to resources and user rights, as well as the creation, modification and deletion of user IDs and user groups.

user attributes

BS2000/POSIX: All the features of a *user ID* which are stored in the *user ID catalog*.

user catalog

See *join file*

user group

BS2000: Collection of individual *users* under one name.

user ID

POSIX/UNIX: Positive integer which serves to identify a system user.

user privileges

BS2000: All attributes assigned to a *user ID* and stored in the *join file* which displays privileges.

12 Abbreviations

AID	Advanced Interactive Debugger
ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BOOTP	BOOtsTrap Protocol
BSD	Berkeley Software Distribution
CRTE	Common RunTime Environment
DFS	Distributed File System
DMS	Data Management System
DNS	Domain Name Service
DSSM	Dynamic Subsystem Management
EBCDIC	Extended Binary Coded Decimal Interchange Code
EDT	EDiTor
HSMS	Hierachical Storage Management System
IEEE	Institute of Electrical and Electronics Engineers
ISAM	Indexed Sequential Access Method
ISO	International Organization for Standardization
LAN	Local Area Network
NFS	Network File System
OPS	Output Presentation Service
PAM	Primary Access Method
PLAM	Program Library Access Method
POSIX	Portable Open System Interface for UNIX
SAM	Sequential Access Method
SCI	Software Configuration Inventory
SNMP	Simple Network Management Protocol
SPOOL	Simultaneous Peripheral Operation On Line
SRPM	System Resources and Privileges Management
TFTP	Trivial File Transfer Protocol
TIAM	Terminal Interactive Access Method

TLI	Transport Layer Interface
TPR	Task Privileged
TU	Task Unprivileged
UFS	UNIX File System
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
XTI	X/Open Transport Interface

13 Related publications

You will find the manuals on the internet at https://bs2manuals.ts.fujitsu.com.

[1] **POSIX**
Commands
User Guide

[2] **POSIX**
BS2000 filesystem bs2fs
User Guide

[3] **POSIX**
SOCKETS/XTI for POSIX
User Guide

[4] **C/C++**
POSIX Commands
User Guide

[5] **C/C++**
C/C++ Compiler
User Guide

[6] **C/C++**

C Library Functions for POSIX Applications
Reference Manual

[7] **CRTE**
Common RunTime Environment
User Guide

[8] **NFS**
Network File System
User Guide

[9] **SECOS**
Security Control System - Access Control
User Guide

[10] **SECOS**
Security Control System - Audit
User Guide

[11] **BLSSERV**
Dynamic Binder Loader / Starter in BS2000
User Guide

[12] **COBOL2000**
COBOL Compiler
User Guide

[13] **COBOL85**
COBOL Compiler
User Guide

[14] **EDT**
Statements
User Guide

[15] **EDT**
Unicode Mode Statements
User Guide

[16] **BS2000**
System Administration
User Guide

[17] **BS2000**
Utility Routines
User Guide

[18] **openFT (BS2000)**
Command Interface
User Guide

[19] **openFT (UNIX/Windows)**
Command Interface
User Guide

[20] **openFT (UNIX/Windows)**
Installation and Operation
System Administrator Guide

[21] **HSMS**
Volume 1: Functions, Management and Installation
User Guide

[22] **HSMS**
Volume 2: Statements
User Guide

[23] **IMON**
User Guide

[24] **JV**
User Guide

[25] **SDF**
SDF Dialog Interface
User Guide

[26] **SDF-A**
User Guide

[27] **BS2000**
Systems Administration
User Guide

[28] **BS2000**
Commands
User Guide

[29] **DSSM/SSCM**
User Guide

[30] **BS2000**
Executive Macros
User Guide

[31] **SDF-P**
Programming in the Command Language
User Guide

[32] **SPOOL**
User Guide

[33] **SPOOL&Print Commands**
User Guide

[34] **SORT**
User Guide

[35] **AID**
Debugging under POSIX
Supplement

[36] **AID**
Debugging of C/C++ Programs
User Guide

[37] **APACHE**
WWW-Server on BS2000
User Guide

[38] **WebTransactions**
Connection to openUTM Applications via UPIC
User Guide

[39] **WebTransactions**
Connection to OSD Applications
User Guide

[40] **interNet Services**
User Guide

[41] **interNet Services**
Administrator Guide

[42] **SNMP Management (NET-SNMP) for BS2000**
User Guide