English

Fujitsu Software BS2000

POSIX
BS2000 file system bs2fs

User Guide

Valid for:
POSIX A49
BS2000 V21.0B

Edition August 2025

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: bs2000.info@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_bs2fs_en 5
1 Preface 6
1.1 Objectives and target groups of this manual 7
1.2 Summary of contents 8
1.3 Changes since the last edition of the manual 9
1.4 Notational conventions 10
2 Overview and embedding in POSIX 11
2.1 Concept and embedding of the bs2fs file system in POSIX 12
2.2 Overview of the use of the bs2fs files system 13
2.2.1 Making files available in the bs2fs file system 14
2.2.2 Representation of files in the bs2fs file system 15
2.2.3 Interfaces for managing bs2fs file systems 16
2.3 Security concept 17
2.4 Sample sessions 18
2.4.1 Introductory example for quick entry into the bs2fs file system 19
2.4.2 Application scenarios for bs2fs file systems 26
3 Use of bs2fs file systems 30
3.1 Managing bs2fs file systems 31
3.1.1 Creating and mounting the bs2fs container 32
3.1.2 Mounting bs2fs file systems 33
3.1.3 General aspects relating to mounting and unmounting operations 34
3.1.4 Automation 35
3.2 Working with the bs2fs file system 36
3.2.1 BS2000 files and library elements supported 37
3.2.1.1 BS2000 DMS files 38
3.2.1.2 PLAM libraries and elements 41
3.2.2 bs2fs naming conventions 43
3.2.3 Access attributes 44
3.2.4 Security 46
3.2.5 Various special features of bs2fs 48
4 Interfaces for supporting bs2fs file systems 51
4.1 Administration interfaces 52
4.1.1 Commands 53
4.1.2 mount - mount a file system 54
4.1.3 mountall - mount file systems 61
4.1.4 show_pubset_export - show file system affected by pubset export 63
4.1.5 start_bs2fsd - start copy daemons 65

4.1.6 umount - unmount a file system 66
4.1.7 umountall - unmount file systems 67
4.1.8 Copy daemon bs2fsd 68
4.1.9 Administration files 69
4.1.10 /etc/mnttab - table of the mounted file systems 70
4.1.11 /etc/vfstab - table of the defined file systems 72
4.2 Information on shell commands and POSIX tools 75
4.2.1 df - report free disk space 76
4.2.2 dumpfs - dump file system 77
4.2.3 fsck - file system check 78
4.2.4 fsexpand - expand existing file systems 79
4.2.5 pathchk - check pathnames 80
4.2.6 pdbl/posdbl - set up and manage user-specific/global program cache 81
4.3 Information for C program interfaces 83
4.3.1 errnos 84
4.3.2 fstat, stat - bs2fs file structure 85
4.3.3 fstatvfs, statvfs - file system information 87
4.3.4 sysfs - query information about the file system type 89
4.3.5 Further special aspects and restrictions 90
5 Access to bs2fs file systems via NFS 92
5.1 Sharing bs2fs files 93
5.2 Mounting shared bs2fs files 94
5.3 Security aspects and access rights 95
5.4 Converting file names 96
5.5 Further special features when working with bs2fs files on the NFS client . 97
5.6 Recommendations for mounting bs2fs file systems on NFS clients 98
5.7 Examples 99
6 Diagnosis and enhancing performance 101
6.1 Overview of the mount and unmount operations 102
6.2 Diagnosis 105
6.3 get_container_index assign bs2fs container index to bs2fs file system . . . 106
6.4 Correcting errors 107
6.5 Performing recovery for the bs2fs_lost+found area 108
6.6 Measures to enhance performance 113
7 Glossary 114
8 Related publications 116

POSIX_bs2fs_en

1 Preface

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.

The BS2000 file system *bs2fs* is supported since POSIX V7.0.

1.1 Objectives and target groups of this manual

The manual is intended for all bs2fs file system users. Use of this manual presupposes a knowledge of the UNIX and BS2000 operating systems and assumes that you have access to the manual "POSIX Basics".

1.2 Summary of contents

Chapters 2 and 3 provide an overview of bs2fs file systems, how they are integrated in BS2000, and how to use bs2fs file systems.

Chapter 4 contains a description of the commands, the daemons and the administration files.

Chapter 5 describes the accesss to bs2fs file systems via NFS.

Chapter 6 gives advice on troubleshooting and error recovery, along with measures you can undertake to enhance the performance of bs2fs file systems.

The manual also contains a glossary and a list of related publications.

1.3 Changes since the last edition of the manual

This edition of the manual contains the following changes with respect to its predecessor (order number: U41802-J-Z125-1-76):

Support of access to bs2fs files systems via NFS.

Recovery of bs2fs files which were relocated as a result of errors when writing back to the bs2fs_lost+found directory.

1.4 Notational conventions

The following notational conventions are used in this manual:

In body text

italics

All syntax elements and also other file names, path names and commands are shown in *italics*.

i ...

This symbol identifies important information and exceptional circumstances which you should be aware of.

In the syntax

plain type

Variables: These characters are replaced by other characters which you select and enter.

bold face

Constants: These characters must be entered exactly as they are printed.

`[ ]`

Optional: Everything which is enclosed in square brackets can be entered, but does not have to be. The square brackets themselves should not be entered unless this is expressly required.

`' '`

A blank which must be entered.

`...`

The preceding expression can be repeated. If blanks need to be entered between the repeated expressions, and the blank is not contained in the expression, a blank (`' '`) will precede the ellipsis symbol.

`|`

The vertical line separates alternative specifications.

In examples

`bold typewriter face`

Inputs: With character-oriented terminals input lines are concluded with the RETURN key, while with block-oriented terminals the same function is fulfilled by EM DUE; the key specifications are therefore omitted.

`normal typewriter face`

Outputs

2 Overview and embedding in POSIX

This chapter provides you with an overview of bs2fs file systems. You will learn how bs2fs file systems are embedded in POSIX and which security mechanisms you can use to protect files in bs2fs file systems.

The chapter also contains a collection of sample sessions.

2.1 Concept and embedding of the bs2fs file system in POSIX

The bs2fs file system enables you to access BS2000 files using POSIX interfaces (commands and program interfaces).

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 therefore 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.

2.2 Overview of the use of the bs2fs files system

The bs2fs file system enables you to make BS2000 files available for editing under POSIX and to edit these as though they were POSIX files.

2.2.1 Making files available in the bs2fs file system

The bs2fs file system is a hierarchical file system, such as the UNIX file system, which consists of directories and files.

The system administrator must take the following steps to permit BS2000 files to be edited in a bs2fs file system:

- Create the bs2fs container
- Mount the bs2fs container (*mount*)
- Mount the bs2fs file system (*mount*)

2.2.2 Representation of files in the bs2fs file system

How BS2000 files are represented in a bs2fs file system depends on whether "simple" DMS files or PLAM libraries are concerned:

- DMS files are represented as files in POSIX.
- PLAM libraries are represented as directories with a permanently predefined two-level hierarchy in POSIX.

The names of these files and directories are identical to the names of the corresponding DMS files, library elements or PLAM libraries. However, in BS2000 the file names are not case-sensitive.

The following applies for the notation for file names in the bs2fs file system:

- File names are always output in lowercase notation.
- Entry can generally take place in uppercase or lowercase notation or a mixture of both. When wildcards are used in the shell or in the case of the *find* command, however, **only lowercase** notation may be employed, otherwise no corresponding file names are
  found.
- As customary in the POSIX shell, special characters of the POSIX shell (e.g. '$' or '*') must be explicitly escaped during input.

  Examples of how to escape BS2000 file name BS2DAT$$1:

  ```
  ls /home/bach/bs2fs1/bs2dat\$\$1
  ls /home/bach/bs2fs1/'bs2dat$$1'
  ```

2.2.3 Interfaces for managing bs2fs file systems

Commands, daemons and administration files are available for managing bs2fs file systems.

Commands

The *mount*, *mountall* and *umount*, *umountall* commands are contained in POSIX-BC. These commands can also be used for mounting and unmounting bs2fs file systems.

Daemons

Daemons are system processes which run constantly and mainly in the background and which execute general tasks. The copy daemon *bs2fsd* transfer files from BS2000 to the bs2fs container and vice versa. When the bs2fs container is mounted, two *bs2fsd* daemons are started automatically.

Administration files

The administration files support the management of resources. They contain either information for the user which is output using commands or information for commands which either the user or commands have entered in these files. The administration files */etc/mnttab* and */etc/vfstab* are used for the application of bs2fs file systems.

2.3 Security concept

bs2fs is used to make BS2000 objects (DMS files, PLAM libraries, PLAM types, PLAM libray elements) visible for POSIX interfaces in the form of a file system.

Each POSIX user should be entitled to exactly the same rights as they are entitled to as a BS2000 user. Each BS2000 object should be provided with precisely the same protection vis-à-vis POSIX functions as vis-à-vis BS2000 functions.

Only users who are permitted to modify protection attributes with BS2000 means should also be permitted to do this in a bs2fs file system with POSIX means. Users who are permitted to read, write or modify a BS2000 object with BS2000 means should also be permitted to do this in a bs2fs file system with POSIX means. Users who may not do the former should also not be allowed to do the latter.

In the case of bs2fs file accesses, the POSIX user's rights may on no account exceed the rights they have as a BS2000 user.

This concept means that when bs2fs files are accessed with POSIX means, the same options are not always offered as in the UNIX world.

An example: write permission is required to delete BS2000 objects with BS2000 means. Consequently bs2fs files can therefore only be deleted with POSIX means (e.g. *rm*) if the write attribute is set. In the UNIX world files can also be deleted without the write attribute.

2.4 Sample sessions

This section contains examples of the application of bs2fs file systems. You are shown how you can configure and manage bs2fs file systems and how you can access the files. A few examples of how bs2fs file systems are used are also provided.

Two bs2fs file systems are used in these examples.

2.4.1 Introductory example for quick entry into the bs2fs file system

This section describes how a session with two bs2fs file systems typically runs.

First the system administrator configures the bs2fs container and mounts it. Then the system administrator mounts two bs2fs file systems for a user ID. Subsequently the user can work with the files of the bs2fs file systems. Finally the system administrator unmounts the bs2fs file systems again.

Configuring and mounting the bs2fs container using the POSIX installation program

The append function of the POSIX installation program (Administrate POSIX filesystems) is used to create a new ufs file system with the *bs2fscontainer* option and to mount it immediately using `Automount=Y`. Creating a new ufs file system or overwriting an existing ufs file system with the *bs2fscontainer* option causes this file system to be assigned the property bs2fs container, which is required to permit the ufs file system to be used as a bs2fs container.

```
                    Definition of BS2000 Container File

BS2000 filename:  :V70A:$SYSROOT.FS.BS2FSCONTAINER

BS2000 filesize:  1000000   PAM-Pages

POSIX filesystem? (y/n): Y
A new BS2000 container file has been created
================================================================================

                       Definition of POSIX filesystem

Size of filesystem: 100000    PAM-Pages     Journaling? (y/n): N

POSIX mountpoint: /bs2fscont
Automount? (y/n): Y          Mountoptions:  bs2fscontainer

Overwrite existing filesystem? (y/n):       POSIX filesystem marker (y/n): Y
No filesystem in BS2000 container yet
================================================================================
Save definitions:  ENTER
Help            :  F1                                        terminate: F2
```

After the container has been mounted successfsully and the installation program has been terminated, two *bs2fsd* copy daemons are started automatically.

```
# ps -efT|grep bs2fsd
    ROOT   163  74MR     1  0 15:29:55 ?         0:00 [bs2fsd]
    ROOT   162  74MQ     1  0 15:29:55 ?         0:00 [bs2fsd]
```

At this time the bs2fs container still has no content:

```
# ls -l /bs2fscont
Total 0
#
```

Alternative: mounting the bs2fs container using the mount command

Once a bs2fs container file system has been created with the POSIX installation program (property bs2fs container, see above), it can also be mounted using the *mount* command by specifying the *–o bs2fscontainer* option, e.g.

```
# mount –F ufs –o bs2fscontainer /bs2fscont
```

Two copy daemons are also started automatically in this case.

> **Supplementary information on mounting the bs2fs container:**
>
> The ufs file system which functions as a bs2fs container is expected to be an empty file system. If it is not empty, its contents are deleted by the *-o bs2fscontainer* option when it is mounted.
>
> The contents are deleted in all mount operations: *mount* command, automount in the case of POSIX startup, automount by the installation program.
>
> The property bs2fs container was introduced as a safety measure to prevent a ufs file system from being emptied inadvertently when it is mounted as a bs2fs container.

Mounting bs2fs file systems

After the bs2fs container has been mounted and the copy daemons have been stated, bs2fs file systems can be mounted. As with nfs file systems, this is not done using the POSIX installation program, but with the *mount* command.

The following files under the user ID **BACH** and with the catalog ID **:V70A:** should be made available in bs2fs file systems:

1. All files whose names match the pattern **ASS.*.S** should be made available under **/home/bach/bs2.1**
2. All files whose names match the pattern **PLAMLIB*** should be made available under **/home/bach/bs2.2**

Two bs2fs file systems are therefore required, and these can be set using the following commands:

```
# mount –F bs2fs –o ftyp=text ':v70a:$bach.ass.*.s'  /home/bach/bs2.1
# mount –F bs2fs –o ftyp=text ':v70a:$bach.plamlib*' /home/bach/bs2.2
```

The *mount* command automatically extends the */etc/mnttab* file by the following entries:

```
:V70A:$BACH.ASS.*.S   /home/bach/bs2.1  bs2fs  ftyp=text  1196152657
:V70A:$BACH.PLAMLIB*  /home/bach/bs2.2  bs2fs  ftyp=text  1196152666
```

The bs2fs container now contains directories (which are still empty) which are to accommodate copies of the two bs2fs file systems' BS2000 files. Pure information functions such as the *ls* command do not initiate copying. Only when an *open()* is applied to a bs2fs file does the bs2fsd daemon become active. Details on the structure of the bs2fs container are provided in "Overview of the mount and unmount operations".

```
# ls -l /bs2fscont
Total 16
drwx--x--x   2 SYSROOT  SYSROOT     2048 Nov 27 11:40 V70A.BACH.1
drwx--x--x   2 SYSROOT  SYSROOT     2048 Nov 27 11:41 V70A.BACH.2
#
```

The protection bit attributes of the bs2fs container are automatically set by the *mount* operation in such a manner that the container's contents are not visible to nonprivileged users (`drwx--x--x`).

Working with bs2fs file systems

After the two bs2fs file systems have been mounted, the result of the mount operations can be checked.

Display the content of the file system mounted under **`/home/bach/bs2.1`**:

```
# ls -l /home/bach/bs2.1
Total 3620
-rwx------   1 BACH     OS315      12288 Nov 22 14:54 ass.consio.s
-rwx------   1 BACH     OS315      18432 Nov 22 14:54 ass.lock.s
-rwx------   1 BACH     OS315       6144 Nov 22 14:54 ass.lockadm.s
-rwx------   1 BACH     OS315      18432 Nov 22 14:55 ass.lockwrk.s
-rwx------   1 BACH     OS315       2048 Nov 22 14:55 ass.posasto.s
-rwx------   1 BACH     OS315      18432 Nov 22 14:55 ass.posbs2fs.s
-rwx------   1 BACH     OS315     139264 Nov 22 14:56 ass.posbs2fx.s
...
-rwx------   1 BACH     OS315      69632 Nov 22 14:58 ass.posutil.s
-rwx------   1 BACH     OS315      51200 Nov 22 14:59 ass.posvert.s
-rwx------   1 BACH     OS315      49152 Nov 22 14:59 ass.posvmm.s
-rwx------   1 BACH     OS315       2048 Nov 22 14:59 ass.pprot.s
-rwx------   1 BACH     OS315      18432 Nov 22 15:00 ass.slpwkp.s
-rwx------   1 BACH     OS315      32768 Nov 22 15:00 ass.ttrap.s
-rwx------   1 BACH     OS315       2048 Nov 22 15:00 ass.uaddr.s
#
```

So a number of simple BS2000 files is mounted under **`/home/bach/bs2.1`**.

Supplementary information on the representation of the bs2fs files:

Protection bits

map either the BS2000 standard attributes (USER-ACCESS, ACCESS) or the BACL properties. In our case, for example, the standard attributes USER-ACCESS=*USER-ONLY and ACCESS=*WRITE.

Owner (USER)

is always the ID of the bs2fs mount.

Group (GROUP)

is always the POSIX group of the owner ID. However, this group assignment is not relevant for bs2fs accesses. The BS2000 group assignments with / without SECOS apply. See chapter 3 for details.

File size

of unopened files is a multiple of 2048 occupied PAM pages. The smallest file according to the *ls* command is therefore 2048 bytes in size.

File name:

Names are displayed in lowercase notation. This also applies for library and element names. The reference to bs2fs objects with shell commands or C program interfaces, on the other hand, can be in either uppercase or lowercase.

Exception: lowercase notation is required in wildcard constructs in the shell and with the *find* command.

Now display the content of the second file system mounted under `/home/bach/bs2.2`:

```
# ls -l /home/bach/bs2.2
Total 5224
drwx------   2 BACH     OS315    1562624 Nov 19 17:39 plamlib.1
drwxrwxrwx   2 BACH     OS315    1112064 Nov 19 17:30 plamlib.2
```

This file system consists of two PLAM libraries, which can be recognized from the representation as a directory.

Display the content of the library plamlib.1:

```
# ls -l /home/bach/bs2.2/plamlib.1
Total 32
drwx------   2 BACH     OS315         48 Nov 19 17:39 d
drwx------   2 BACH     OS315         48 Nov 19 17:39 j
drwx------   2 BACH     OS315         48 Nov 19 17:39 l
drwx------   2 BACH     OS315         48 Nov 19 17:39 m
drwx------   2 BACH     OS315         48 Nov 19 17:39 p
drwx------   2 BACH     OS315       4032 Nov 19 17:39 s
drwx------   2 BACH     OS315         48 Nov 19 17:39 x
```

The library directory contains subdirectories whose names correspond to the element types (D, J, L, M, P, S, X).

Supplementary information on the library and type directories:

The directory structure and the directory attributes cannot be modified. Actions such as deleting, renaming, creating and modifying attributes are always rejected when they apply to library and type directories.

Consequently, for example, no new PLAM library can be created using *mkdir*.

Now display the type S elements:

```
# ls -l /home/bach/bs2.2/plamlib.1/s
Total 6024
-rwx------   2 BACH     OS315       8192 Nov 19 17:33 acct.c
-rwx------   2 BACH     OS315       8192 Nov 19 17:33 acct.c+001
-rwx------   2 BACH     OS315      51200 Nov 19 17:33 bio.c
-rwx------   2 BACH     OS315      51200 Nov 19 17:33 bio.c+001
-rwx------   2 BACH     OS315       8192 Nov 19 17:33 bitmap.c
-rwx------   2 BACH     OS315       8192 Nov 19 17:33 bitmap.c+001
-rwx------   2 BACH     OS315       4096 Nov 19 17:33 bitmasks.c
-rwx------   2 BACH     OS315       4096 Nov 19 17:33 bitmasks.c+001
-rwx------   2 BACH     OS315      12288 Nov 19 17:34 bs.c
-rwx------   2 BACH     OS315      12288 Nov 19 17:34 bs.c+001
...
-rwx------   2 BACH     OS315      14336 Nov 19 17:39 vm_pageout.c
-rwx------   2 BACH     OS315      14336 Nov 19 17:39 vm_pageout.c+001
-rwx------   2 BACH     OS315      18432 Nov 19 17:39 vm_pgout.c
-rwx------   2 BACH     OS315      18432 Nov 19 17:39 vm_pgout.c+001
-rwx------   2 BACH     OS315       6144 Nov 19 17:39 vm_subr.c
-rwx------   2 BACH     OS315       6144 Nov 19 17:39 vm_subr.c+001
-rwx------   2 BACH     OS315       4096 Nov 19 17:39 xmmu.c
-rwx------   2 BACH     OS315       4096 Nov 19 17:39 xmmu.c+001
-rwx------   2 BACH     OS315       8192 Nov 19 17:39 xsys.c
-rwx------   2 BACH     OS315       8192 Nov 19 17:39 xsys.c+001
#
```

The library directory contains some type S elements. Each of these elements occurs in the example with just one version, '001'. As a result, two entries exist for each element: one entry for the element name without a version identifier and one with the (only) version identifier.

Explanation:

In the bs2fs file system the element with the highest version can also be addressed as an element name without a version (implemented internally as a hard link). The representation and the reference to elements with lower versions, on the other hand, always include the version. See also chapter 3.

Now you can, for example, apply POSIX commands to the files of the bs2fs file systems:

```
# grep ETPND /home/bach/bs2.1/*.s
/home/bach/bs2.1/ass.consio.s:    ETPND   &MODNAME,VER=&MODVERS,PATCH=200
/home/bach/bs2.1/ass.lock.s:      TITLE    '&MODNAME - ETPND/XREF'
/home/bach/bs2.1/ass.lock.s:      ETPND    &MODNAME,VER=&MODVERS,PATCH=200
# tail /home/bach/bs2.1/ass.posvert.s
         HDRCHECK UNIT=228,FUNCT=(187,1)
         @BEND
         @BEND
         @EXIT
         @END
         EJECT
***************************** ETPND *********************************
         ETPND &MODNAME,VER=&MODVERS,PATCH=200
*
         END
#
```

Unmounting the bs2fs file systems

If the files in the bs2fs file systems are no longer required in POSIX, the file systems concerned can be unmounted again.

Use the *umount* command to unmount individual bs2fs file systems, thus in this example:

`umount /home/bach/bs2.1`

and

`umount /home/bach/bs2.2`

You can use the *umountall* command to unmount all bs2fs file systems:

either

`umountall –F bs2fs`

or

`umountall –b`

After you have unmounted the file systems using *umount* or *umountall*, the corresponding entries in the */etc/mnttab* will also have been deleted.

Automating the bs2fs mount operations

The following options, among others, are available to automate bs2fs *mount* operations:

1. Enter the *mount* commands in shell scripts and execute these scripts at any time.
2. Create entries in the file system table */etc/vfstab* using `Automount=Y`.
   The bs2fs *mount* operations are then executed automatically when the POSIX subsystem is started. Furthermore, the */etc/vfstab* table is read out by the *mountall* command if no other file system table is specified.

   You can use the *mountall* command to mount all the bs2fs file systems on the basis of the entries in the */etc/vfstab* table:
   either

   `mountall –F bs2fs`

or

```
mountall –b
```

3. Create entries in a separate file system table (similarly to */etc/vfstab*). This table can then be used as an argument of the *mountall* command.

Example: create an entry in a mountall table (/etc/vfstab would be similar)

You can use the *mount -p* command to create a file which already contains a suitable line in vfstab format:

```
# mount -p |grep /home/bach/bs2.1 >> /etc/bs2fstab
# cat /etc/bs2fstab
:V70A:$BACH.ASS.*.S - /bs2test/bs2fs bs2fs - no  ftyp=text
```

You must change the automount entry from `no` to `yes` to permit automatic mounting with the *mountall* command, e.g. using

```
# edtu /etc/bs2fstab
:V70A:$BACH.ASS.*.S - /bs2test/bs2fs bs2fs – yes ftyp=text
```

You can use the *mountall* command to mount all bs2fs file systems on the basis of the entries in the */etc/bs2fstab* table:

either

```
mountall –F bs2fs /etc/bs2fstab
```

or

```
mountall –b /etc/bs2fstab
```

2.4.2 Application scenarios for bs2fs file systems

Various application options for bs2fs file systems are presented in this section.

Example 1: Searching CONSLOG files

The stored CONSLOG files for the month of June 2007 are to be searched for particular contents. This search can be executed very simply using the *grep* command if the files are available in a bs2fs file system.

This example requires that the bs2fs container must already have been created.

Proceed as follows:

Mount the BS2000 files which are to be processed:

```
# mount -F bs2fs ':V70a:$sysaudit.sys.conslog.2007-06*' /home/bs2.conslog
```

Check the result of the mount operation:

```
# mount | grep 'bs2.conslog'
/home/bs2.conslog on :V70A:$SYSAUDIT.SYS.CONSLOG.2007-06* ftyp=text/nosuid on Tue Nov 27 13:
52:23 2007
```

or

```
# df -k -F bs2fs | grep 'bs2.conslog'
:V70A:$SYSAUDIT.SYS.CONSLOG.2007-06*  2000000   331518  1668482    17% /home/bs2.conslog
```

Display the BS2000 files made available in the bs2fs file system:

```
# ls -l /home/bs2.conslog
Total 11836
-r-x------   1 100      OTHER     151552 Jun 12 12:32 sys.conslog.2007-06-11.007.001
-r-x------   1 100      OTHER     256000 Jun 13 13:17 sys.conslog.2007-06-12.007.001
-r-x------   1 100      OTHER      75776 Jun 13 16:26 sys.conslog.2007-06-13.007.001
-r-x------   1 100      OTHER      73728 Jun 13 17:25 sys.conslog.2007-06-13.007.002
-r-x------   1 100      OTHER      77824 Jun 14 12:36 sys.conslog.2007-06-13.007.003
-r-x------   1 100      OTHER      77824 Jun 14 14:42 sys.conslog.2007-06-14.007.001
-r-x------   1 100      OTHER    5347328 Nov 14 11:12 sys.conslog.2007-06-14.007.002
#
```

In the files provided, search for file names with the prefix *:V70A:$BACH.SEM:*

```
# grep ':V70A:$BACH.SEM' /home/bs2.conslog/*
...
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MBW-000.163259  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: couldbebs2fsname <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MB0-000.163310  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: analyseresource after toupper <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MB0-000.163310  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: couldbebs2fsname <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MCH-000.170925  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: analyseresource after toupper <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MCH-000.170925  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: couldbebs2fsname <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MCL-000.170939  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: analyseresource after toupper <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MCL-000.170939  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: couldbebs2fsname <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MER-000.144635  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: analyseresource after toupper <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MER-000.144635  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: couldbebs2fsname <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0ME1-000.165231  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: analyseresource after toupper <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0ME1-000.165231  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: couldbebs2fsname <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MKI-000.133902  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: analyseresource after toupper <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MKI-000.133902  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: couldbebs2fsname <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MNC-000.130039  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: analyseresource after toupper <:V70A:$BACH.SEM*.C>
/home/bs2.conslog/sys.conslog.2007-06-14.007.002:  <C %0MNC-000.130039  %  POS1020 Message
of the POSIX kernel:WARNING: bs2_subr.c: couldbebs2fsname <:V70A:$BACH.SEM*.C>
#
```

Example 2: Converting lowercase notation to uppercase notation

The lowercase notation in a BS2000 file is to be converted to uppercase notation. You can use the *tr* command for this conversion if the files are contained in a bs2fs file system.

In this example it is assumed that the file to be edited is made available under */home/bs2.conslog*, as in "Example 1: Searching CONSLOG files".

Display the BS2000 files provided in the bs2fs file system:

```
# ls -l /home/bs2.conslog
Total 11836
-r-x------   1 100      OTHER     151552 Jun 12 12:32 sys.conslog.2007-06-11.007.001
-r-x------   1 100      OTHER     256000 Jun 13 13:17 sys.conslog.2007-06-12.007.001
-r-x------   1 100      OTHER      75776 Jun 13 16:26 sys.conslog.2007-06-13.007.001
-r-x------   1 100      OTHER      73728 Jun 13 17:25 sys.conslog.2007-06-13.007.002
-r-x------   1 100      OTHER      77824 Jun 14 12:36 sys.conslog.2007-06-13.007.003
-r-x------   1 100      OTHER      77824 Jun 14 14:42 sys.conslog.2007-06-14.007.001
-r-x------   1 100      OTHER    5347328 Nov 14 11:12 sys.conslog.2007-06-14.007.002
#
```

Convert the lowercase notation in the *sys.conslog.2007-06-13.007.001* file to uppercase notation:

Write the result to the *conslog.out* file. (The name of the input file is specified unambiguously in abbreviated form using wildcards.)

```
# tr '[:lower:]' '[:upper:]' </home/bs2.conslog/*13.007.001 >conslog.out
```

Check the result by displaying the last records of the input and output files:

```
# tail  /home/bs2.conslog/*13.007.001
  <C %0CYT-000.162451  %  POS1020 Message of the POSIX kernel:WARNING: vfs.c: generic mount 1
  <C %0CYT-000.162451  %  POS1020 Message of the POSIX kernel:WARNING: vfs.c: generic mount 2
  <C %0CYT-000.162451  %  POS1020 Message of the POSIX kernel:WARNING: vfs.c: generic mount 3
  <C %0CYT-000.162451  %  POS1020 Message of the POSIX kernel:WARNING: bs2_vfsops.c:
entering bs2fs_mount
  <C %0CYT-000.162451  %  POS1020 Message of the POSIX kernel:WARNING: bs2_subr.c:
couldbebs2fsname
  <0 %0CYT-000.162451  %  EXC0420 /LOGOFF PROCESSED. CPU TIME USED: 2.8544 SEC, USER ID:
TSOS, TASK ID: 1DDF00D6
  <0 %0CYU-000.162528  %  JMS0154 'TSOS' LOGGED ON FOR 'MCP0212C/STATION9'. JOB NAME
'BACHMANN'. CALLER 'TSN 0BEP'. TID 1C7E0082
  <0 %0CYU-000.162529  %  EXC0420 /LOGOFF PROCESSED. CPU TIME USED: 0.3710 SEC, USER ID:
TSOS, TASK ID: 1C7E0082
     /0B1Q-000.162609 CHANGE-CONSLOG PROCESSOR NAME: MCP0212C STATION NAME: STATIO10 AUDIT-
ID: 00000000000000000000000000000000
     TCLOG    .162609  ***2007-06-13*** 000004 **** UTC+02:00
************************************************************
# tail conslog.out
  <C %0CYT-000.162451  %  POS1020 MESSAGE OF THE POSIX KERNEL:WARNING: VFS.C: GENERIC MOUNT 1
  <C %0CYT-000.162451  %  POS1020 MESSAGE OF THE POSIX KERNEL:WARNING: VFS.C: GENERIC MOUNT 2
  <C %0CYT-000.162451  %  POS1020 MESSAGE OF THE POSIX KERNEL:WARNING: VFS.C: GENERIC MOUNT 3
  <C %0CYT-000.162451  %  POS1020 MESSAGE OF THE POSIX KERNEL:WARNING: BS2_VFSOPS.C:
ENTERING BS2FS_MOUNT
  <C %0CYT-000.162451  %  POS1020 MESSAGE OF THE POSIX KERNEL:WARNING: BS2_SUBR.C:
COULDBEBS2FSNAME
  <0 %0CYT-000.162451  %  EXC0420 /LOGOFF PROCESSED. CPU TIME USED: 2.8544 SEC, USER ID:
TSOS, TASK ID: 1DDF00D6
  <0 %0CYU-000.162528  %  JMS0154 'TSOS' LOGGED ON FOR 'MCP0212C/STATION9'. JOB NAME
'BACHMANN'. CALLER 'TSN 0BEP'. TID 1C7E0082
  <0 %0CYU-000.162529  %  EXC0420 /LOGOFF PROCESSED. CPU TIME USED: 0.3710 SEC, USER ID:
TSOS, TASK ID: 1C7E0082
     /0B1Q-000.162609 CHANGE-CONSLOG PROCESSOR NAME: MCP0212C STATION NAME: STATIO10 AUDIT-
ID: 00000000000000000000000000000000
     TCLOG    .162609  ***2007-06-13*** 000004 **** UTC+02:00
************************************************************
#
```

3 Use of bs2fs file systems

This chapter shows you how to manage bs2fs file systems and what you need to take into consideration when working with bs2fs file systems.

3.1 Managing bs2fs file systems

Managing bs2fs file systems involves the following tasks:

- Creating the bs2fs container with the POSIX installation program
- Mounting and unmounting the bs2fs container
- Mounting and unmounting bs2fs file systems
- Modifying administration files if the lists of the resources which need to be provided or mounted need to be updated
- Locating and correcting problems when bs2fs file systems are used (see "Diagnosis and enhancing performance")

3.1.1 Creating and mounting the bs2fs container

To permit the use of bs2fs file systems, a special area is required in POSIX which temporarily stores the files of bs2fs file systems: the bs2fs container. This is a ufs file system which is explicitly identified as the bs2fs container. Only one single bs2fs container can exist in a POSIX system. This bs2fs container accommodates all the files of all mounted bs2fs file systems.

You can create a file system with the property bs2fs container using the POSIX installation program when installation takes place.

Its size should be defined in accordance with the size of the files actually used. As it can be assumed that not all BS2000 files will be opened at the same time and therefore copied into the container, you could, for example, estimate the space required in the container as follows: multiply the sum of the sizes of all BS2000 files which are mounted with bs2fs by 0.5.

Creating the bs2fs container using the POSIX installation program

When *append* is specified for a new ufs file system or an existing file system (with

overwrite=yes) together with the *bs2fscontainer* option, this file system is internally

assigned the property bs2fs container which is required to enable this file system to be

used as the bs2fs container.

Mounting the bs2fs container using the POSIX installation program

Like any other ufs file system, the bs2fs container (= ufs file system with the property bs2fscontainer) can be mounted in the file system hierarchy when *Automount=Y* is specified. The *Automount=Y* entry which is then created in the file system table */etc/vfstab* also causes the file system to be mounted automatically when the POSIX subsystem is started.

Mounting the bs2fs container during ongoing operation using the mount command

To mount the bs2fs container, enter a corresponding *mount* command with the *-o bs2fscontainer* option, e.g.

```
mount -F ufs -o bs2fscontainer /dev/dsk/23 /home/bach/mount2
```

A prerequisite for mounting a file system as the bs2fs container is that it has been created as described under "Creating the bs2fs container using the POSIX installation program", otherwise a *mount* command with the *-o bs2fscontainer* option is rejected.

A bs2fs container is by default empty when it is mounted. If it is not empty, its contents are deleted when it is mounted.

3.1.2 Mounting bs2fs file systems

When you mount bs2fs file systems, you define which BS2000 files are made available at which position in the POSIX file system hierarchy.

You can only mount a bs2fs file system after the bs2fs container has been mounted.

To mount a bs2fs file system, enter a corresponding *mount* command with the *-F bs2fs* option, e.g.

```
mount -F bs2fs ':v70a:$bach.ass.*.s' /home/bach/bs2.1
```

3.1.3 General aspects relating to mounting and unmounting operations

Both bs2fs file systems and the bs2fs container can be mounted and unmounted during ongoing operation using the *mount* and *umount* commands respectively. However, the order specified below must be observed here. Command entries which deviate from this order result in errors and cause the command concerned to abort:

Order for mounting and unmounting operations

- Mounting the bs2fs container
  The ufs file system with the property bs2fs container must be mounted before the first bs2fs file system can be mounted.
- Mounting or unmounting bs2fs file systems
  bs2fs file systems can be mounted and unmounted again while the bs2fs container is mounted.
- Unmounting the bs2fs container
  The bs2fs file system with the property bs2fs container can be unmounted only when no bs2fs file system is mounted.

3.1.4 Automation

bs2fs container

The *mount*/*umount* operations are managed and thus also automated in the same way as other POSIX file systems of the type "ufs". When the file system is created using the POSIX installation program (*append* function), an entry is automatically generated in the */etc/vfstab* table. If `Automount=yes` was specified, the file system is mounted immediately. The `Automount=yes` entry in the */etc/vfstab* table also causes the file system to be mounted automatically the next time the POSIX subsystem starts. Modifications in the */etc/vfstab* table can also be made using the POSIX installation program (*modify* function).

bs2fs file systems

Similarly to nfs file systems, bs2fs file systems cannot be managed using the POSIX installation program. To automate *mount* operations, the administrator must edit the system tables or create scripts.

Overview of the various automation options:

1. Enter *mount* commands in shell scripts and execute these at any time.
2. Enter *mount* commands in shell scripts of the type "rc". These will then be executed automatically when the POSIX subsystem is started.
3. Enter the required bs2fs file systems in vfstab format (with Automount=yes) in a separate file system table. This table can then be used as an argument in the *mountall* command.
4. Enter the required bs2fs file systems (with Automount=yes) in the */etc/vfstab* table. This table is processed by the *mountall* command if no file system table is specified. When the POSIX subsystem starts, the *mount* operations will be executed automatically.

Recommendation: entries suitable for vfstab can be created with *mount –p* for a bs2fs file system which has already been successfully mounted.e

An example of this is provided in "Introductory example for quick entry into the bs2fs file system".

Starting and terminating POSIX

If the *mount*/*umount* operations take place automatically when POSIX is started and terminated, this has the advantage that the necessary order is observed.

Startup: ufs file systems, bs2fs file systems, nfs file systems

Termination: nfs file systems, bs2fs file systems, ufs file systems

If the administrator wants to mount and unmount the file systems explicitly, they must observe this order.

3.2 Working with the bs2fs file system

This section describes what types of files and library elements you can edit in a bs2fs file system, which access rights apply in a bs2fs file system, and what special aspects you must bear in mind when working with bs2fs file systems.

3.2.1 BS2000 files and library elements supported

In bs2fs file systems you can edit "simple" DMS files and PLAM libraries and their elements.

3.2.1.1 BS2000 DMS files

When BS2000 DMS files are supported in a bs2fs file system, a distinction must be made

between whether a new file is created or whether it already exists in BS2000.

Newly created DMS files

- FCBTYPE

  New DMS files created in a bs2fs file system are always of the type FCBTYPE SAM with a variable record length.

- BACL (see also "Access attributes")

  The BACL attributes are automatically enabled in BS2000 in accordance with the mode specified in *open()* and the *umask* setting (`mode & ~umask`).

  For example, a *umask* setting of `022` (default in the shell) and a *mode* pattern of `777` result in the protection bit rights `rwx r-x r-x`.

  When BACL is enabled, modifications are no longer evaluated with /MODIFY-FILE-ATTRIBUTES in USER-ACCESS or ACCESS.

- Transfer mode *ftyp*

  Whether a SAM file is transferred to the bs2fs container and back in *text*, *textbin* or *binary* mode depends on the *ftyp* option of the *mount* command for the bs2fs file system. The POSIX command *ftyp* is not evaluated and is reserved for *bs2cp*. See also *mount*
  command, *-o* option *ftyp*.

Existing DMS files

- FCBTYPE

  Existing DMS files can be of the FCBTYPE SAM, ISAM or PAM. The FCBTYPE is also retained if such a file is overwritten.

  The following is meant by overwriting: *creat()*, *open (O_CREAT |O_WRONLY |O_TRUNC)*, *cp* command, but not the *rename()* or *mv* command.

  No FCBTYPE (no open) is yet defined for existing DMS files which were created only by means of /CREATE-FILE or something similar. After being opened as bs2fs files, these files are assigned the FCBTYPE SAM.

- BACL (see also "Access attributes")

  When write or overwrite accesses take place, all the original protection attributes are retained. If the BACL attributes had not been enabled beforehand, they are also not enabled later. The *mode* specifications in *open()* or *creat()* are, as is customary in POSIX, ignored in the case of existing files.

  In the bs2fs file system the BACL properties of existing files without BACL can only be enabled by explicitly modifying the protection attributes with *chmod()* or the *chmod* command.

- Transfer mode ftyp

  ISAM files are always transferred in *text* mode, PAM files always in *binary* mode. In the case of SAM files - as with newly created SAM files - the *ftyp* option of the *mount* command is relevant for the bs2fs file system.

  > **Take care when copying within the bs2fs world**
  > If the file attributes of the source and the target are different with respect to the transfer mode (text, binary), this can lead to unexpected (data) results.
  >
  > There is no analogy to the /COPY-FILE command, where the target file is assigned the same FCBTYPE as the source file, when copying using POSIX bs2fs means.

BS2000 file types which are not supported

The following files/libraries are **not** displayed in a bs2fs file system under POSIX, in other words, for example, they are not output by the *ls* command:

- Exported (migrated) files/libraries (please also observe the notes on handling migrated files in "Various special features of bs2fs")
- Alias catalog entries
- File generation groups and file generations
- Files/libraries which are accessed via RFA
- Tape files
- Files on private disks

Files protected by a password are visible in the bs2fs file system under POSIX, but their contents can only be accessed by operations which are not protected by a password, e.g. read access to a file protected by a write password.

Files protected by guards are visible in the bs2fs file system under POSIX. Their contents can only be accessed if the protecting guard permits corresponding access for the caller's user ID.

Overview of the open modes of DMS files

DMS files in POSIX are interpreted differently according to their FCBTYPE, the record length and the *ftyp* option when the bs2fs file system is mounted (see "mount - mount a file system "). An overview of how these properties are mapped to the file type and the corresponding open modes in the bs2fs file system is provided by the table below:

FCBTYPE	ftyp	File type	Open mode for reading	Open mode for writing
SAM	text (default)	Text file	r	w,tabexp=yes
	binary	Binary file	rb,type=record	wb,type=record,forg=seq
	textbin	Binary text file	r	w,tabexp=no
ISAM	-	Text file	r	w,tabexp=yes
PAM	-	Binary file	r	wb,type=record

The record keys of ISAM files are ignored when reading; when writing, numeric record keys (in character representation) are generated.

3.2.1.2 PLAM libraries and elements

Element types supported

The bs2fs file system supports the same element types as the *bs2cp* command (and /COPY-POSIX-FILE), namely the predefined text types (*D*, *J*, *M*, *P*, *S*, *X*) and LLMs (*L*). Furthermore, types defined by the user (e.g. with openFT) and types derived from the standard types mentioned above are also supported in bs2fs file systems.

Like DMS files of the type SAM, the *text*, *textbin* or *binary* mode in which the text elements (all types other than LLM) are copied to and from the bs2fs container depends on the *ftyp* option of the *mount* command for the bs2fs file system.

Representation of PLAM libraries and library elements

A PLAM library is represented as a directory with the name of the PLAM library. By default this library directory contains the subdirectories of the standard types *D, J*, *L*, *M*, *P*, *S* and *X* in which the corresponding element types are stored.

in addition, the library directory contains further directories with the names of **derived element types**, but only if at least one element of the relevant type exists and the derived type has one of the standard types D, J, L, M, P, S or X as its basic type.

The standard types C, F, H, R, U and SYSJ and types derived from these are not supported. This also applies for the reserved types whose names begin with '$' or 'SYS'.

If the PLAM library contains **user-defined types** which do not have a standard type as their basic type, these are represented as corresponding subdirectories. Such types can, for instance, be created with openFT. New PLAM libraries cannot be created with bs2fs means (e.g. with *mkdir*). New PLAM libraries which were created with other means (e.g. with LMS) after a bs2fs *mount* can, however, be edited at any time in the bs2fs file system. To permit this, it is only necessary for their name to match the wildcard pattern of the bs2fs *mount* command.

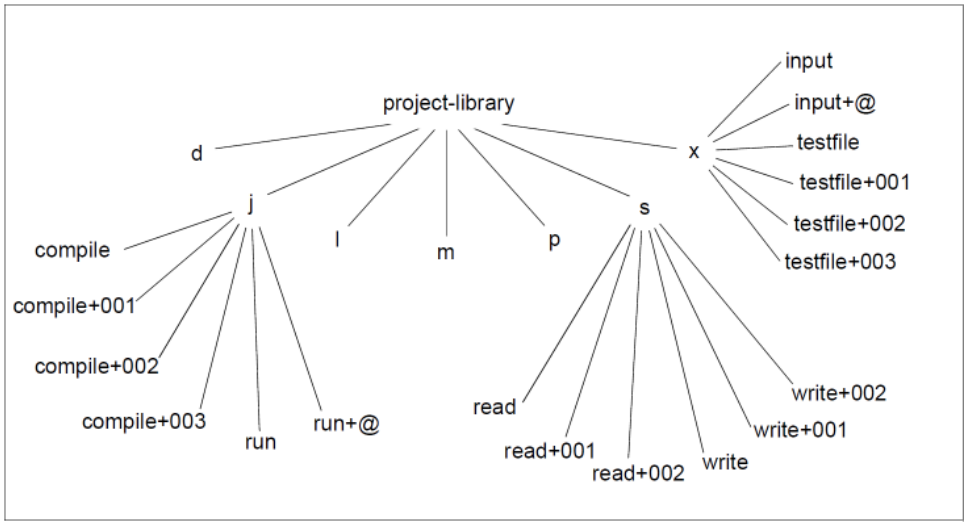
Library elements are stored as files in the directory whose name corresponds to their type. Their names have the following format: *elementname* **+** *version*. This name permits targeted access to a particular version of an element. The name *elementname* permits access to the highest version of an element. A hard link to the highest version of an element is involved here.

The following special features apply for this hard link:

- If a new element version with a higher version identifier is generated in the bs2fs file system, *elementname* automatically refers to the new highest version.
- Deleting *elementname* (e.g. with `rm` *elementname*) has the following effect:

  First the file containing the highest version of the element is deleted. If other versions of the element exist, *elementname* refers to the new highest version. Otherwise the hard link *elementname* is also deleted.
- If a particular version of an element is deleted (e.g. with `rm` *elementname*+*ver*), the hard link *elementname* is also deleted if no other versions of the element exist.
- All versions of an element can be deleted with `rm` *elementname* `+*`.

Example of the structure of a PLAM library



3.2.2 bs2fs naming conventions

When a new BS2000 DMS file or a library element is created with POSIX means (*open O_CREAT*, *creat*, *rename*), the name is specified as a POSIX file name, but it is subject to the rules for creating DMS file names, element names (name length, valid characters) and to further bs2fs-specific rules.

Uppercase/lowercase notation

The names can be specified in uppercase or lowercase or a mixture of both. However, the file names are always output in lowercase notation. When wildcards are used in the shell or in the *find* command, though, lowercase notation must be used, otherwise no corresponding file names are found. If uppercase notation were used in the file name pattern, file name replacement in the shell would fail as the pattern is transferred with wildcard syntax.

Uppercase/lowercase notation is possible for all objects below the bs2fs mount point, i.e in the case of PLAM libraries also for the library directories and the type directories

Wildcard patterns in the mount command for the bs2fs file system

When a new DMS file is created, the file name is checked to see whether it is contained in the range of the BS2000 wildcard pattern which was specified for the bs2fs file system in the *mount* command. If it is not, the *open* is rejected with *errno ENAME*. This check also takes place when a file (e.g. for a reading *open*, *access*, *stat*, etc.) or PLAM library is searched for; a non-match is rejected with *errno ENAME*.

3.2.3 Access attributes

Mapping the BS2000 standard attributes

The table below shows how the BS2000 standard attributes are mapped to the access rights in POSIX:

BS2000 attributes		POSIX standard ACL								
		user			group			others		
ACCESS	**USER-ACCESS**	**r**	**w**	**x**	**r**	**w**	**x**	**r**	**w**	**x**
***READ**	***OWNER-ONLY**	x	-	x	-	-	-	-	-	-
***READ**	***ALL-USERS/*SPECIAL**	x	-	x	x	-	x	x	-	x
***WRITE**	***OWNER-ONLY**	x	x	x	-	-	-	-	-	-
***WRITE**	***ALL-USERS/*SPECIAL**	x	x	x	x	x	x	x	x	x

The rights defined with ACCESS and USER-ACCESS are mapped to POSIX access rights for a file only if no Basic Access Control List (BACL) is defined for it.

The following attributes of files and library elements are not mapped to the bs2fs file system:

- EXEC-PASSWORD, READ-PASSWORD, WRITE-PASSWORD (file or library element)
  EXPIRATION-DATE (file)
  GUARD (file or library element)

  If one of these protection attributes is set for a file or an element, an access attempt can be rejected with EACCES even if the status information supplied seems to permit access.

- CODED-CHARACTER-SET (file or library element)

  If this attribute is not *NONE, *STD or EDF03IRV, the result of processing in a bs2fs file system can differ from the result of the same processing in BS2000 (for example with the same C program): e.g. different collating sequence, different representation on the screen, etc.

- STATE=*IN-HOLD (library element)

  If this attribute is set, an access attempt is rejected with EACCES, even though the status information supplied does not enable you to recognize this.

- ACCESS-METHOD (library element)

  In the case of elements with SOURCE-ATTRIBUTE=*KEEP, this attribute can be adopted by a corresponding file. However, it is ignored in the bs2fs file system. If such an element is copied or moved to a file, processing this file can yield different results from processing the original element.

- ISAM key

  ISAM keys from BS2000 files are not visible when access takes place via a bs2fs file system. If an ISAM file in a bs2fs file system is copied to another existing ISAM file which is overwritten in the process, the records of the two files then generally have different ISAM keys.

- ISAM key (library element)

  ISAM keys from an ISAM file which was transferred to a library element using SOURCE-ATTRIBUTE=*KEEP are a component part of the records concerned.

- STORAGE-FORM=*DELTA (library element)

  Library elements with this attribute (delta elements) can only be read in a bs2fs file system.

- Record format B (library element)

  Library elements with records with format B can only be edited if they are of type L or of a type derived from this type. Attempts to access elements of a different type are rejected with EMODE.

- Record type other than 1 (library element)

  Record types with format A and a record type other than 1 are not visible when reading takes place.

  In the case of an update (O_RDWR or O_APPEND) and when renaming within the same library and the same type or between types with the same basic type, they are retained provided the source is not open in write mode when renaming takes place.

  They are not transferred when a library element is copied or moved into a file or another library or when it is renamed as a type with a different basic type or when a source which is open in write mode is renamed.

BACL attributes

When new bs2fs files and library elements are created and when attributes are modified with *chmod*, the BACL attributes in BS2000 are enabled. When the BACL is enabled, modifications made using /MODIFY-FILE-ATTRIBUTES are no longer evaluated for USER-ACCESS or ACCESS.

- Systems without SECOS:

  The BACL group rights of the BS2000 file and the group rights mapped in the bs2fs file system can be modified on systems on which SECOS is not active, but they are not evaluated when files are accessed. Only the user rights and the rights for OTHER count. This behavior is identical in the case of both BS2000 and bs2fs accesses.

- Systems with SECOS, *UNIVERSAL group:

  On systems on which SECOS is active but no SECOS groups have been configured and assigned, all IDs are members of the *UNIVERSAL group. This has the following consequences:

  - In the case of file accesses in native BS2000, only the user rights and the rights for GROUP are evaluated, but not the rights for OTHER.
  - In the case of accesses in the bs2fs file system, the POSIX group and the rights for OTHER play a role:

    If the protection attribute `-rw-r----- (0640)` is set, read access by another ID is rejected if the ID does not also belong to the same POSIX group. If this is not the case, read permission for OTHER is required to permit access in read mode (0644).

    If the ID belongs to the same POSIX group, only read permission for GROUP is required.

3.2.4 Security

bs2fs file systems permit access to files in BS2000. You can prevent unauthorized accesses using the various protection mechanisms of BS2000 and POSIX. The following protection mechanisms are available:

- User administration (BS2000)
- File access protection (BS2000 and POSIX)

Access rights in the bs2fs file system

Users always have the same access rights in the bs2fs file system as in BS2000. They can therefore only perform the same types of access (read, write, execute) to a file or library element which they would also be permitted to perform in native BS2000.

Access restrictions can occur in individual cases. If, for instance, a file is protected by a password or guard, it is possible that a POSIX user is forbidden from using an access which they would be permitted to use as a BS2000 user.

The table below provides an overview of the access rights required for the various accesses:

Access	**Object**	**Rights required for access**
Create	File	Owner (userid) of the file system (if the file already exists, write permission for the file is sufficient)
	Library element	Write and read permission for the library **and** also administration permission for the library or the type if this permission is defined (if the library element already exists, write permission for the library element is sufficient)
Open for reading	File	Read permission for the file
	Library element	Read permission for the library **and** read permission for the element
Open for writing	File	Write and read permission for the file
	Library element	Write and read permission for the library **and** write permission for the element If the element is not opened for overwriting (O_TRUNC), read permission for the element is also required.
Delete	File	Owner (userid) of the file system **and** write permission for the file
	Library element	Write and read permission for the library **and** also administration permission for the library or the type if this permission is defined **and** write permission for the element
Rename	File/ Library element	Deletion rights are required for the source and creation rights for the target

The rights defined with Basic Access Control Lists (BACLs) or ACCESS and USER-ACCESS apply for all other accesses.

> By default the TSOS ID has the same rights as the owner of a file or a PLAM library.
>
> **No** special rights are assigned to the SYSROOT ID for accessing objects in bs2fs file systems.

Application recommendations

The following recommendations apply to ensure that the files and library elements which are mounted in a bs2fs file system can be accessed by a POSIX user:

- If a user is to be permitted to execute a file or library element, at least one read permission must be set for them. The user must also have read permission in order to open a file in write mode (e.g. open with O_WRONLY) or a library element in write mode, but not in overwrite mode (e.g. open with O_WRONLY without O_TRUNC). Read permission may not be withdrawn from the user if the object is protected by a guard.
- If a user is to be permitted to read, write or execute a file or a library element, the relevant access may not be protected by a password.
- To permit write access to a file or library element or to permit it to be deleted, this object may not be protected by an expiration date (EXPIRATION-DATE) which is later than the current dates.
- For each access to library elements, to their properties or to the directory of a library a user requires at least read permission for the library.

3.2.5 Various special features of bs2fs

When you work with files in a bs2fs file system, you must bear in mind a few special features (e.g. compared to working with local POSIX files), and these are summarized again below:

- New files created in a bs2fs file system

  New files or library elements which are created in a bs2fs file system are assigned BACL attributes. The file attributes ACCESS and USER-ACCESS are consequently irrelevant with regard to protection.

- DMS wildcard in the case of the *mount* command for the bs2fs file system

  New DMS files (always of the type SAM) can be created only if their names comply with the pattern specified in the *mount* command and the syntax rules of BS2000. In particular, it is not possible to create files whose names begin with a period (.).

- PLAM library: BS2000 file lock of the type read

  When a PLAM library in the bs2fs file system is accessed (e.g. with *ls*), it is assigned a read lock ("Lock Type INPUT") in BS2000. You can then not rename, delete or modify the PLAM library's file attributes. Read locks are automatically released again by the bs2fs daemon if no further bs2fs accesses to the PLAM library have taken place within approx. 20 seconds.

  The bs2fsd daemon immediately releases locks of the type write ("Lock Type INOUT") for a PLAM library as soon as there is no longer any element open for the job submitting user in the library.

- Write permission is required to delete files in a bs2fs file system

  Deletion of files in a bs2fs file system is always rejected if no write permission exists. In POSIX files can be deleted even without write permission (cf. *rm* with query or with the *-f* option or *mv* with query).

- Access rights for the privileged user

  As is the case with the access rights in BS2000, only the TSOS ID has the same rights as the owner of a file or of a PLAM library by default when accessing files in the bs2fs file system. The SYSROOT ID, which is also assigned the privileged uid=0 in POSIX, is treated like any other nonprivileged ID when bs2fs file accesses occur.

  In the POSIX/UNIX world privileged IDs (uid=0) are allowed to perform accesses which go beyond the file owner's access options, e.g. reading a file, although the protection bits do not grant the file owner read access.

  However, in bs2fs file systems TSOS also only has the permissions which the owner of an object has in BS2000. When SECOS is used (e.g. GUARDS), the owner can also restrict TSOS rights.

- Updating files in text mode

  When existing records in SAM or ISAM files which are mounted with *ftyp=text* are updated, the record length may not be modified. Shortening or lengthening a record results in an *EIO* error. The *write()* is not executed, and after *close()* only those parts of the BS2000 file are modified which were written successfully with *write()*.

- Time stamp

  The time stamps of a file or library element determined with FSTAT are mapped in the bs2fs file system to the fields of the *stat* data structure in POSIX as follows:

BS2000	**POSIX**
CRE-DATE/TIME	*st_ctime*
ACC-DATE/TIME	*st_atime*
CHANG-DATE/TIME	*st_mtime*

  The following must be borne in mind here:

  - While a file is open, the *st_atime* and *st_mtime* fields of the ufs shadow file are accepted as the *st_atime* and *st_mtime of the* bs2fs in the bs2fs file system; after the file has been closed, the values which FSTAT supplies in BS2000 are output again. This can result in the modification times for a file displayed with the /SHOW-FILE-ATTRIBUTES command differing from the times ascertained with *ls -l* in the bs2fs file system as long as the file is open.
  - The CRE-DATE/TIME and *st_ctime* fields are interpreted differently in BS2000 and POSIX/UNIX. In BS2000, CRE-DATE/TIME specifies the time when a file was created; in POSIX/UNIX, *st_ctime* logs the time when the management structures were last changed (e.g. because of *chmod*)

- File sizes

  As long as a file or library element has not not been opened in bs2fs after mounting (*mount*), the size of a file specified in the output of the *ls -l* command is calculated from the number of PAM pages supplied by the BS2000-FSTAT macro multiplied by 2048. Only after a file has been opened in the bs2fs file system is the display precise to the byte. This value can be incorrect if the size of the file was modified outside the bs2fs.

- Handling migrated files

  Processing of migrated files is not supported. This has the following consequences:

  - *ls*, *stat()*, *fstat()* do not display the names of migrated files.
  - If you attempt to open a migrated file, errno 8 (`ENOENT`, "No such file or directory") is returned.
  - If you attempt create a new file with the name of a migrated file, e.g. with *creat()*, *rename()* or with the shell command *cp*, *mv*, errno 206 (`ENXIO`, "No such device or address") is returned. `ENXIO` is unambiguous for bs2fs accesses, i.e. an attempt to create a new file with the name of a migrated file is always involved.

- X bit for directories

  The x bit for a directory is interpreted differently in the bs2fs file system than is customary in the UNIX world. In the bs2fs file system you require read permission to search for an entry in the table of contents of a PLAM library, not execution permission.

- The directory structure of a bs2fs file system is defined by the *mount* operation and cannot be modified as it maps the structure of PLAM libraries. Directories in a bs2fs file system can therefore not be deleted or renamed, nor is it possible to create new directories. Consequently no new PLAM libraries can be generated using bs2fs means (e.g. *mkdir*), either. Nor can the access rights of directories be modified.

- Library elements can be created in the defined directories (types) with any names which comply with the syntax rules for PLAM libraries, except in directories which map a type which does not have a standard type as its basic type.

- Library elements which are assigned to a type which does not have a standard type as its basic type can only be read.

- Library elements of a type with the basic type L can only be renamed as or copied to library elements of the same type or of another type with the basic type L. Likewise, files or library elements of a type with a basic type other than L cannot be renamed as or copied to an element of a type with the basic type L.

- Library elements are stored in the bs2fs file system under a name which is composed of the element name and the version identifier:
  elementname+*ver*
  In addition, a hard link *elementname* exists to the highest current version of the element. This has the following consequences:
  - If a new element version with a version identifier which is higher than the existing highest version identifier is generated in the bs2fs file system, *elementname* automatically refers to the new highest version.
  - Deleting *elementname* (e.g. with `rm` *elementname*) has the following consequences:First the file containing the highest version of the element is deleted.
    If other versions of the element exist, *elementname* subsequently refers to the new highest version. Otherwise the hard link *elementname* is also deleted.
  - If a particular version of an element is deleted (e.g. with `rm` *elementname*+*ver*), the hard link *elementname* is also deleted if no other versions of the element exist.

4 Interfaces for supporting bs2fs file systems

This chapter contains descriptions of the interfaces for managing bs2fs file systems, as well as information on shell commands, POSIX tools and C program interfaces.

4.1 Administration interfaces

POSIX supports the management of bs2fs file systems with:

- commands
- daemons
- administration files

You require POSIX root authorization (BS2000 ID TSOS or SYSROOT) to manage bs2fs file systems.

4.1.1 Commands

This section describes commands which are relevant for the use of bs2fs file systems in alphabetical order. The notational conventions which are used in the command syntax are explained in "Notational conventions".

Command	Function
mount	mount a file system
mountall	mount file systems
show_pubset_export	show file systems affected by EXPORT-PUBSET
start_bs2fsd	start copy daemons
umount	unmount a file system
umountall	unmount file systems

Table 1: bs2fs commands

4.1.2 mount - mount a file system

mount (Format 2 and Format 3) mounts a ufs file system into the file system hierarchy at the path name position *mountpoint.* This position must already exist. If *mountpoint* possesses any contents prior to the mount operation these remain hidden until the file sysem is dismounted again.

mount (Format 4 and Format 5) mounts a bs2fs file system at a particular position in the POSIX file system. A bs2fs file system is understood to be a selectable set of files in BS2000 which are made available transparently in POSIX so that they can be accessed using POSIX means (commands, program interfaces). The files are selected via the user and catalog ID and wildcard symbols.

In addition, *mount* (Format 1) can be used to output a list of all the mounted file systems.

Syntax

```
Format 1:  mount[ -v | -p]

Format 2:  mount[ -F ufs][ -V][ -r]
               [ -o spec_options] { resource | mountpoint}

Format 3:  mount[ -F ufs][ -V][ -r]
               [ -o spec_options] resource mountpoint

Format 4:  mount[ -F bs2fs][ -V][ -r]
               [ -o spec_options] { resource | mountpoint }

Format 5:  mount[ -F bs2fs][ -V][ -r]
               [ -o spec_options] resource mountpoint
```

Display a list of mounted file systems

```
Format 1:  mount[ -v | -p]
```

No option specified

mount displays a list of all mounted file systems (see Example).

-v

Displays a new presentation of the output. The new output contains the file system type and options in addition to the information in the old output. The fields *mountpoint* and *resource* change places (see Example).

-p

Displays a list of the mounted file systems in the */etc/vfstab* format.

mount ufs file systems

```
Format 2:  mount[ -F ufs][ -V][ -r]
               [ -o spec_options] { resource | mountpoint}

Format 3:  mount[ -F ufs][ -V][ -r]
               [ -o spec_options] resource mountpoint
```

The descriptions of Format 2 and Format 3 have been combined, since they differ only in terms of the (optional) specifications *resource* and *mountpoint*.

Format 2 can be used only if an entry for the relevant file system already exists in the */etc/vfstab* file. The missing specification for *resource* or *mountpoint* is then added from this.

Formats 2 and 3 can be entered only by the POSIX administrator.

No option specified

mount displays a list of all mounted file systems.

-F ufs

Specifies *ufs* as the file system type.

-V

Displays the entire command line on screen but does not execute the command. The command line is displayed together with the options and arguments entered by the user and with the values derived from */etc /vfstab*. This option allows you to check the general validity of the command line.

-r

Mounts the file system with read access.

-o

Specifies *ufs* file system-specific options. Multiple options should be comma-separated. If invalid options are specified a warning is issued and the invalid options are ignored.

The following options may be selected:

f

Imitates an */etc/mntab* entry but does not mount a file system. The parameters are not checked.

n

Mounts the file system without making an entry in */etc/mnttab*.

journal

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

rw | ro

Read/write or read only access. The default value is *rw*.

`nosuid`

By default, the file system is mounted in such a way that the s-bit is set for users. If you specify *nosuid* then the default value is deactivated and the file system is mounted without the s-bit being set for users.

`remount`

Is used together with *rw*. A file system which has been mounted with read access only can be remounted with read/write access. This option fails if the file system is not currently mounted or has been mounted with *rw*.

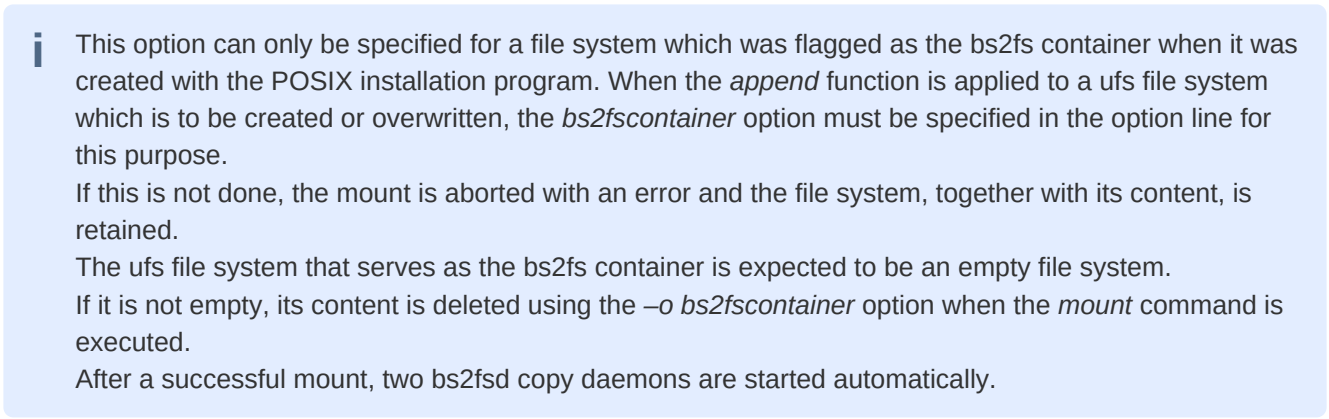
`bs2fscontainer`

Specifies the file system which is to be mounted as the bs2fs container, i.e. as a file system which temporarily accommodates files from bs2fs file systems.

This option may only be specified for a single *ufs* file system. Any other *mount* commando with this option is rejected.

The *-r*, *-o ro*, *-o journal* and *-o remount* options may not be specified together with the *bs2fscontainer* option.

When the POSIX installation program is used, this option can be entered via the option line.



This option can only be specified for a file system which was flagged as the bs2fs container when it was created with the POSIX installation program. When the *append* function is applied to a ufs file system which is to be created or overwritten, the *bs2fscontainer* option must be specified in the option line for this purpose.
If this is not done, the mount is aborted with an error and the file system, together with its content, is retained.
The ufs file system that serves as the bs2fs container is expected to be an empty file system.
If it is not empty, its content is deleted using the *–o bs2fscontainer* option when the *mount* command is executed.
After a successful mount, two bs2fsd copy daemons are started automatically.

`resource`

specifies the file system which is to be mounted.

`mountpoint`

Specifies the local position for mounting *resource*. You must specify an absolute path name.

mount bs2fs file sytems

```
Format 4:  mount[ -F bs2fs][ -V][ -r]
               [ -o spec_options] { resource | mountpoint }

Format 5:  mount[ -F bs2fs][ -V][ -r]
               [ -o spec_options] resource mountpoint
```

The descriptions of Format 4 and Format 5 have been combined, since they differ only in terms of the (optional) specifications *resource* and *mountpoint* .

Format 4 can be used only if an entry for the relevant file system already exists in the */etc/vfstab* file. The missing specification for *resource* or *mountpoint* is then added from this (see Hint).

Formats 4 and 5 can be entered only by the POSIX administrator.

A **prerequisite** for entering a *mount* command of Format 4 or 5 is that a bs2fs container is already mounted.

No option specified

mount displays a list of all mounted file systems.

-F bs2fs

Specifies *bs2fs* as the file system type.

-V

Displays the entire command line on screen but does not execute the command. The command line is displayed together with the options and arguments entered by the user and with the values derived from */etc /vfstab*. This option allows you to check the general validity of the command line.

-r

Mounts the file system with read access.

-o

Specifies *bs2fs* file system-specific options. Multiple options should be comma-separated. If invalid options are specified a warning is issued and the invalid options are ignored. The following options may be selected:

rw | ro

Read/write or read only access. The default value is *rw*.

nosuid

The file system is mounted without setting the s bit for users. This option is enabled by default for bs2fs file systems and cannot be disabled.

remount

Is used together with *rw*. A file system which has been mounted with read access only can be remounted with read/write access. This option fails if the file system is not currently mounted or has been mounted with *rw*.

ftyp={text|binary|textbin}

This option has the same effect as the *ftyp* command when copying files using the *bs2cp* command. It defines whether BS2000 SAM files and text type POSIX library elements (element type other than L) are interpreted in POSIX as text or binary files. PAM files are always interpreted as binary files, ISAM files always as text files.

This option should only be specified once. If it is specified more than once, the specification with the highest priority applies, where *ftyp=textbin* has the highest priority, *ftyp=text* the next highest priority and *ftyp=binary* the lowest priority.

The default is *ftyp=text*.

ftyp=text

SAM files and text-type library elements are interpreted as text files. When writing to a bs2fs file, end-of-line characters (X'15') are converted to a record change and tabulator characters (X'05') to the corresponding number of blanks.

ftyp=binary

SAM files and text-type library elements are interpreted as binary files. A 1:1 transfer takes place without interpreting and converting data (record change/end-of-line characters, tabulator/blanks, etc.).

ftyp=textbin

SAM files and text-type library elements are interpreted as binary text files. When writing to a bs2fs file, only end-of-line characters (X'15') are converted to a record change. Tabulator characters (X'05') are not converted to blanks.

`ressource`

Defines which BS2000 files are to be mounted. The following syntax applies for the option:

```
:cat:$user.filename-with-wild
```

The option can be entered in upper- or lowercase or in a mixture of both. Special characters of the POSIX shell such as '$' or '*' must be escaped explicitly.

`cat`

Catalog ID

`user`

BS2000 user ID

`filename-with-wild`

BS2000 file name with wildcard symbols:

`*`

Replaces an arbitrary (even empty) string.

`/`

Replaces precisely one arbitrary character.

`.` Terminating period

Partially -qualified entry of a name.
Corresponds implicitly to the string "./*", i.e. at least one arbitrary character follows the period.

`<s_x:s_y>`

Replaces a string that meets the following conditions:

- It is at least as long as the shortest string (s_x or s_y)
- It is not longer than the longest string (s_x or s_y)
- It lies between s_x and s_y in the alphabetic collating sequence; numbers are sorted after letters (A...Z, 0...9)
- s_x can also be an empty string which is in the first position in the alphabetic collating sequence
- s_y can also be an empty string which in this position stands for the string with the highest possible code (contains only the characters X'FF')

`<s_1,...>`

Replaces all strings that match any of the character combinations specified by s. s can also be an empty string.

Any such string can also be a range specification "s_x:s_y".

`-s`

Replaces all strings that do not match the specified s. The minus sign may only appear at the beginning of the string.

The file set defined with *resource* can consist of both existing files and files which are to be created. When a new file is to be created, the required file name must match the wildcard pattern of the corresponding *mount* command.

`mountpoint`

Specifies the local position for mounting *resource*. You must specify an absolute path name. If *mountpoint* is a symbolic reference then the file system is mounted in the directory to which the symbolic reference points rather than being mounted alongside the symbolic reference.

Hint

If an entry for the file system concerned exists in the */etc/vfstab* file, one of the options *resource* or *mountpoint* can be omitted (Format 2 or 4). When a bs2fs file system is used, the following must be observed in this case:

- If the *mountpoint* option is specified, an entry in */etc/vfstab* can then be identified unambiguously and the corresponding file system is mounted.
- If only the *resource* option is specified, multiple suitable entries for a bs2fs file system can be contained in the */etc/vfstab* file as this file system can be mounted in parallel at multiple locations. In this case only the **first** bs2fs file system entered in the */etc/vfstab* file is mounted. Only entries with an identical wildcard string are recognized as suitable entries. Entries with a different wildcard string are also not taken into account even if they define the same set of files.

File

/etc/mnttab

Table of mounted file systems.

/etc/dfs/fstypes

The default type of distributed file system.

/etc/vfstab

Table of automatically mounted file systems.

Example

Mounting the bs2fs container and a bs2fs file system. The example executes under the POSIX administrator ID.

```
# mount –F ufs –o bs2fscontainer /bs2fscont
# mount -F bs2fs ':V70a:$sysaudit.sys.conslog.2007-06*'  /home/bs2.conslog
# mount
/ on /dev/root read/write/setuid on Tue Nov 27 11:31:04 2007
...
/bs2fscont on /dev/dsk/23 bs2fscontainer/setuid/read/write/noquota on Tue Nov 27
11:35:23 2007
/home/bs2.conslog on :V70A:$SYSAUDIT.SYS.CONSLOG.2007-06* ftyp=text/nosuid on Tue
Nov 27 13:52:23 2007
#
```

4.1.3 mountall - mount file systems

mountall is used to mount file systems on the basis of a *file_system_table* (*/etc/vfstab* is the default file system table). The special file name "-" reads from the standard input. If you specify the hyphen then the standard input must possess the same format as */etc/vfstab*.

Before the individual file systems are mounted, *fsck* performs a plausibility test to determine whether the system appears to be viable for mounting (not in the case of file systems of the type *bs2fs* or *nfs)*. If the system is not viable for mounting, *fsck* corrects it before an attempt is made to mount it.
If **only** *file_system_type* is specified then *mountall* applies only to file systems of the specified type.
The file systems are mounted in the order *ufs* - *bs2fs* - *nfs*. This ensures that when the bs2fs file systems are mounted, the bs2fscontainer file system required for this purpose is already mounted in ufs.

Syntax

```
mountall[ -F file_system_type] [ -l |  -r | -b][file_system_table]
```

No option specified

mountall mounts all file systems for which the field *automnt* in the *file system table* is set to *yes*.

options

-F `file_system_type`

Specifies the type of file system to be mounted.

-l

Limits the process to local file systems (*ufs* and *bs2fs*).

-r

Limits the process to remote file system types (*nfs*).

-b

Limits the process to *bs2fs* file systems.

`file_system_table`

If *file_system_table* is not specified, *mountall* refers to */etc/vfstab*.

Hint

If the *-F* option is specified together with one or more of the options *-l*, *-r* and *-b* and the options are mutually compatible, the *-l*, *-r* and *-b* options have priority. For example, *mountall -F bs2fs -l* and *mountall -F ufs -l* have the same effect as *mountall -l*: all local file systems (i.e. all ufs and bs2fs file systems) are mounted. The entries *mountall -F bs2fs* and *mountall -b* also lead to the same result: all bs2fs file systems are mounted.

Error

If the file systems are viable for mounting and error-free, no message is output. Error and warning messages are issued by *fsck* and *mount* or by *mountall* in the case of incorrect syntax.

File

/etc/vfstab

Default file system table.

4.1.4 show_pubset_export - show file system affected by pubset export

This command supplies the system administrator with information on which file systems in POSIX are affected by the export of a pubset and must therefore be unmounted before the EXPORT-PUBSET command is executed.

This information is particularly helpful when bs2fs file systems are used. Unlike with the ufs and NFS file systems, it is not sufficient in this case to check whether the mount point of a file system is located on the pubset concerned.

With bs2fs file systems the location of the BS2000 files mounted by means of bs2fs is also relevant. The mount point of the bs2fs container also plays a role. If it is located on the pubset to be exported, all mounted bs2fs file systems are affected by the export, irrespective of their location.

Depending on the file system type, a file system is affected by EXPORT-PUBSET, if the objects shown in the following list are located on the pubset to be exported:

ufs:

Mount point or container file

nfs:

Mount point

bs2fs:

Mount point or mounted BS2000 files or mount point/container file of the bs2fs container (cf. ufs)

Syntax

```
show_pubset_export cat-id
```

cat-id

Catalog ID of the pubset which is to be checked (without enclosing colons ":"). The entry can be made in upper- or lowercase notation or a mixture of both; the check is performed with the *cat-id* converted to uppercase notation.

File

The following files are searched for the specified catalog ID in order to determine the file systems affected:

/etc/mnttab

Table of all mounted file systems

/etc/partitions

Table of all possible partitions
If the catalog ID entry is missing in this file, the default ID is determined via BS200 and used for the check.

SYSSSI.POSIX-BC.<version>

SYSSSI file of POSIX from BS2000
The BS2000 file name of the container file of the root file system is determined from this file (ROOTFSNAME parameter) as this name is not entered in the */etc/partitions* file.

Example

The file systems affected by the export of the DATA pubset are determined. The container file of the ufs file system mounted under the home directory */home/bach* resides on the DATA pubset.

```
# show_export DaTa
the nfs filesystems on pubset DATA
nfs filesystem PGTR0157:/home4 mounted on /home/bach/nfs4
nfs filesystem PGTR0157:/home5 mounted on /home/bach/nfs5
the bs2fs filesystems on pubset DATA
bs2fs filesystem :DATA:$BACH.ASS.* mounted on /home/bach/bs2/mount.ass
bs2fs filesystem :V70A:$BACH.CCC.* mounted on /home/bach/bs2/mount.ccc
bs2fs filesystem :DATA:$BACH.PLAM* mounted on /home/bach/bs2/mount.plam
the ufs filesystems on pubset DATA
ufs filesystem /dev/dsk/4 mounted on /home/froede
ufs filesystem /dev/dsk/5 mounted on /home/bach
#
```

4.1.5 start_bs2fsd - start copy daemons

start_bs2fsd enables the system administrator to start additional copy daemons *bs2fsd*.

Syntax

```
start_bs2fsd [number]
```

No option specified

start_bs2fsd provides information about the number of copy daemons currently running.

`number`

Specifies how many copy daemons are to be started in addition to the copy daemons already started. Up to 8 copy daemons can execute.

Example

Check how many daemons are currently running

```
# start_bs2fsd
/sbin/start_bs2fsd: 2 bs2fs daemons are running
```

Start an additional daemon and repeat the check

```
# start_bs2fsd 1
/sbin/start_bs2fsd: start additional daemon 1 of 1
# start_bs2fsd
/sbin/start_bs2fsd: 3 bs2fs daemons are running
```

4.1.6 umount - unmount a file system

The command *umount* unmounts a file system which was mounted using *mount*. The file system entry is deleted from the table */etc/mnttab*.

Syntax

```
umount[ -V] {resource | mountopint}
```

-V

Outputs the entire command line on screen but does not execute the command. The command line is generated with all the options and arguments specified by the user as well as the values taken from */etc /vfstab*. You should select this option in order to subject a command line to a general check and validity check.

`resource`

Specifies the resource which is to be unmounted. Format as for *mount*.

For bs2fs file systems the option must be specified in UPPERCASE in accordance with the entry in the internal tables. Special characters of the POSIX shell such as '$' or '*' must be escaped explicitly.

In the case of an NFS resource this is replaced by the name of the source server. This must be followed by a colon and the path name of the resource.

`mountpoint`

Specifies the local position at which *resource* must be unmounted. You must specify an absolute path name.

Hint

Specification of the *mountpoint* option is always recommended for unmounting bs2fs file systems. If a bs2fs file system is mounted in multiple positions in the POSIX file system (i.e. identical *resource* specification in *mount*) and only *resource* is specified for *umount*, only the file system mounted last is unmounted. However, if *mountpoint* is specified, the corresponding file system is always unmounted.

The *umount* command is rejected if it refers to a file system mounted using the *bs2fscontainer* option and at least one bs2fs file system is still mounted. In the case of a successful *umount* for the bs2f container, the bs2fsd copy daemons are terminated automatically.

File

/etc/mnttab

Table of mounted file systems.

/etc/vfstab

Table of automatically mounted file systems.

4.1.7 umountall - unmount file systems

umountall unmounts all mounted file systems with the exception of *root*, */proc*, */var* and */usr*. If **only** *file_system_type* is specified, *umountall* relates only to file systems of the specified type. The file systems are unmounted in the order *nfs - bs2fs - ufs*. This ensures that the bs2fscontainer file system required for bs2fs file systems is only unmounted in the ufs when it is no longer needed, i.e. when no further bs2fs file systems are mounted.

Syntax

```
umountall[ -F file_system_type][ -k][ -l | -r| -b]
```

-F

Specifies the file system type to be unmounted.

-k

Sends the SIGKILL signal to processes which have opened files in the file system.

-l

Limits the operation to local file systems (*ufs* and *bs2fs*).

-r

Limits the operation to remote file systems (*nfs*).

-b

Limits the operation to bs2fs file systems.

Error

If the file systems can be unmounted no message is output. Error and warning messages are issued by *fsck* and *mount*.

Hint

If the *-F* option is specified together with one or more of the options *-l*, *-r* and *-b* and the options are mutually compatible, the *-l*, *-r* and *-b* options have priority. For example, *mountall -F bs2fs -l* and *mountall -F ufs -l* have the same effect as *mountall -l*: all local file systems (i.e. all ufs and bs2fs file systems) are unmounted. The entries *mountall -F bs2fs* and *mountall -b* also lead to the same result: all bs2fs file systems are unmounted.

File

/etc/mnttab

Table of mounted file systems.

/etc/vfstab

Table of automatically mounted file systems.

4.1.8 Copy daemon bs2fsd

This section describes the copy daemon *bs2fsd*.

The notational conventions which are used in the command syntax are explained in "Notational conventions".

The copy daemon *bs2fsd* transfers BS2000 files and elements of PLAM libraries from BS2000 to the corresponding area of the bs2fs container and vice versa.

A file or library element is transferred from BS2000 to the bs2fs container automatically when the first access to the file takes place in the bs2fs file system (first *open*); in the case of write access, transfer back to BS2000 takes place after editing in POSIX has been completed (last *close*).

If the file or library element has been opened for writing, it remains open in BS2000 from the start of transfer from BS2000 until its transfer back to BS2000 has been completed. It can therefore not be modified by other users while it is being edited in POSIX. This guarantees that the file has a consistent status.

By default 2 *bs2fsd* daemons are started when the bs2fs container is mounted. However, the system administrator has the option of starting further copy daemons when required (e.g. a large number of bs2fs file systems, a large number of files, large files) using the *start_bs2fsd* command.

The daemons run as POSIX background processes mainly in the TU under the TSOS user ID. After they have been started, they "sleep" until they are "woken up" by a request.

You can monitor the daemons by, for instance, entering the following POSIX command:

```
ps -ef | grep bs2fsd
```

In addition, you can use the *start_bs2fsd* command to determine how many copy daemons are running at the moment.

When the bs2fs container is unmounted, all copy daemons are terminated automatically.

4.1.9 Administration files

This section describes the administration files.

The administration files */etc/mnttab* and */etc/vfstab* are employed for the application of bs2fs file systems, as is customary for file system management in POSIX.

The */etc/mnttab* file is automatically created by POSIX at startup and updated with each *mount*/*umount* operation.

The entries for ufs file systems and native file systems in the */etc/vfstab* file are created by the POSIX installation program. Entries for nfs and bs2fs file systems can be processed by the administrator (root authorization) using an editor.

The administration files are listed in the table below:

File	Function
/etc/mnttab	Table of the mounted files systems
/etc/vfstab	Table of the defined files systems

Table 2: Administration files

4.1.10 /etc/mnttab - table of the mounted file systems

The */etc/mnttab* file contains details of all the file systems mounted on the local computer. This file contains information which is generated by the *mount* command.

Each line contains the following information, which is separated by an arbitrary number of blanks and/or tabulators:

Format

resource	mountp	fstype	spec-options	time

`resource`

Absolute path name of the mounted file system or, in the case of bs2fs file systems, mounted BS2000 files in wildcard syntax.

For bs2fs file systems the entry differs from the specification in the *mount* command as follows:

- It is converted completely into uppercase notation
- Characters to be escaped are displayed without the associated escape character

Example:

`mount –F bs2fs –o ftyp=text :v70a:\$bach.sys\* /home/bach/bs2.1`

generates the following entry in */etc/mnttab*:

`:V70A:$BACH.SYS*` `/home/bach/bs2.1 bs2fs ftyp=text ...`

`mountp`

Absolute path name of the mount point.

`fstype`

File system type.

`spec-options`

Options as specified in the *mount* command.

`time`

Mount time, specified in seconds since 1/1/1970

Entries in the */etc/mnttab* file are deleted again if the *umount* or *umountall* command is executed for corresponding file systems or file system types.

Example

Enter the following in the POSIX shell: `cat /etc/mnttab`

```
/dev/root            /                  ufs    rw,suid                          1196069614
/proc                /proc              proc   rw                               1196069614
/dev/fd              /dev/fd            fdfs   rw                               1196069614
/dev/dsk/3           /var               ufs    suid,rw,noquota                  1196069614
/dev/dsk/4           /home/froede       ufs    suid,rw,noquota                  1196069615
/dev/dsk/10          /home/gast         ufs    suid,rw,noquota                  1196069615
/dev/dsk/5           /home/bach         ufs    suid,rw,noquota                  1196069621
/dev/dsk/2           /home/bach/mount99 ufs    suid,rw,noquota                  1196069621
/dev/dsk/23          /bs2fscont         ufs    bs2fscontainer,suid,rw,noquota 1196070061
:V70A:$BACH.ASS.*.S  /home/bach/bs2.1   bs2fs  ftyp=binary                      1196084245
:V70A:$BACH.CCC.*.C  /home/bs2.2        bs2fs  ftyp=text                        1196084250
:V70A:$BACH.PLAMLIB* /home/bach/bs2.2   bs2fs  ftyp=textbin                     1196084255
:V70A:$BACH.SEM*.C   /home/bs2000       bs2fs  ftyp=text                        1196084261
```

4.1.11 /etc/vfstab - table of the defined file systems

The */etc/vfstab* file describes every file system defined on the local computer. You can process the file using an editor.

The file systems for which *yes* is entered in the *automnt* column in the */etc/vfstab* file are mounted automatically when POSIX starts up or when the *mountall* command is executed.

The entries in the file are also used to add any missing details for *resource* or *mountpoint* and *mount* options when a *mount* command is executed.

Corresponding entries are generated automatically in the */etc/vfstab* file for ufs file systems which are defined using the POSIX installation program. The entries must be created manually for all other file systems (e.g. bs2fs or nfs file systems) when required.

In contrast to the */etc/mnttab* file, execution of the *mount* and *umount* commands for the */etc/vfstab* file has no effect. Corresponding entries are retained.

The fields in the table are separated by tabulators and/or blanks. A hyphen (–) indicates an empty entry in the field. The table contains the following fields:

Format

special	fsckdev	mountp	fstype	fsckpass	automnt	mntopts

`special`

Defines the resource to be mounted.

The following must be borne in mind in the case of (manual) entries for bs2fs file systems:

- Letters may only be specified in uppercase.
- Special characters may not be escaped, nor is it permissible to enclose the string in quotes.

`fsckdev`

Name of the block-oriented device or of the resource of the character-oriented device.

`mountp`

Mount point: absolute path name of the directory in which the resource is to be mounted.

`fstype`

File system type.

`fsckpass`

The sequence number to be used for multiple *fsck* commands.

`automnt`

Specifies whether the resource is to be mounted automatically when POSIX starts up or by means of the *mountall* command (*yes*) or not (*no*).

`mntopts`

List of options for mounting the file system, separated by commas. The options correspond to the *spec_options* of the *mount* command.

Example

Enter the following in the POSIX shell: `cat /etc/vfstab`

```
/dev/root              /dev/rroot     /                       ufs    1   yes  -
/proc                  -              /proc                   proc   -   no   -
/dev/fd                -              /dev/fd                 fdfs   -   no   -
/dev/dsk/3             /dev/rdsk/3    /var                    ufs    1   yes  -
172.25.86.64:/home2/froede/SHARE  -   /home/froede/RETSINA    nfs    -   no   soft
PGOB0004:/home2/froede/SHARE      -   /home/froede/PGOB0004   nfs    -   no   soft
/dev/dsk/4             /dev/rdsk/4    /home/froede            ufs    1   yes  -
/dev/dsk/10            /dev/rdsk/10   /home/gast              ufs    1   yes  -
/dev/dsk/13            /dev/rdsk/13   /mnt/ascii              ufs    1   no   -
/dev/dsk/8             /dev/rdsk/8    /mnt/dat1               ufs    1   no   -
/dev/dsk/23            /dev/rdsk/23   /bs2fscont              ufs    1   no   -
/dev/dsk/24            /dev/rdsk/24   /home/bach/mount3       ufs    1   no   -
/dev/dsk/25            /dev/rdsk/25   /home/bach/mountxxx     ufs    1   no   -
/dev/dsk/26            /dev/rdsk/26   /home/bach/mountyyy     ufs    1   no   -
/dev/dsk/5             /dev/rdsk/5    /home/bach              ufs    1   yes  -
/dev/dsk/2             /dev/rdsk/2    /home/bach/mount99      ufs    1   yes  -o
/dev/dsk/6             /dev/rdsk/6    /suderlan               ufs    1   no   -
:V70A:$BACH.ASS.*.S    -              /home/bach/bs2.1        bs2fs  -   yes  ftyp=binary
:V70A:$BACH.CCC.*.C    -              /home/bs2.2             bs2fs  -   yes  ftyp=text
:V70A:$BACH.PLAMLIB*   -              /home/bach/bs2.2        bs2fs  -   yes  ftyp=textbin
:V70A:$BACH.SEM*.C     -              /home/bs2000            bs2fs  -   yes  -
```

4.2 Information on shell commands and POSIX tools

This section describes special aspects when accessing files of bs2fs file systems using shell commands and POSIX tools.

4.2.1 df - report free disk space

The *statvfs()* interface (64-bit variant) is used implicitly in the *df* command. In the case of a bs2fs file system, the data output must then be interpreted accordingly (see also *statvfs()* in "fstatvfs, statvfs - file system information"). All 3 formats of the *df* command are supported for the file system type bs2fs.

Syntax

```
Format 1:  df[ -F FSType] -P[ -lV][ -k | -h | -H][ file]
Format 2:  df[ -F FSType][ -beglntVv][ -k | -h | -H][ -o ufs_options]...[ file]
Format 3:  df -c[ file]...
```

bs2fs-specific options and rules of the df command:

`-F bs2fs`

Identifies a bs2fs file system.

`file`

A path name or the resource can be specified.

Path name:

bs2fs mount point, simple bs2fs file, PLAM library index, type directory, library element

Resource:

Resource of the *mount* command for the bs2fs file system in accordance with the *mnttab* entry in uppercase notation (e.g. 'V70A:$BACH.ASS.*.S').

4.2.2 dumpfs - dump file system

The *dumpfs* command is not supported for bs2fs file systems.

4.2.3 fsck - file system check

The *fsck* command is not supported for bs2fs file systems.

4.2.4 fsexpand - expand existing file systems

The *fsexpand* command is not supported for bs2fs file systems.

4.2.5 pathchk - check pathnames

In the case of files or directories in a bs2fs file system, an error message is output if the name length does not comply with the rules of a bs2fs file system.

4.2.6 pdbl/posdbl - set up and manage user-specific/global program cache

Preloading of executables (LMS element, type L or simple "binary" DMS file) residing in the bs2fs file system with posdbl or pdbl (-b option) is supported.

Syntax (relevant options only)

```
pdbl{ -s[ sid]| -u} -b path

posdbl -b path
```

Example for an LLM element

```
# posdbl -s
POSIX-DBL:              linker ON       loader ON
POSIX-DBL:              holder of global cache: TSN=031F PID=19
Cache  POSIX@DBL        CREATED: 04/10/08 12:24:46
                        SIZE: 32 MB     ENTRIES: 16
                        FREE PAGES: 7112


# posdbl -l
 RM                 39 Apr 10 12:25:30 $TSOS.SINLIB.POSIX-BC.070.SHELL
+IN@RLOGIND        113 Apr 11 11:16:47 $TSOS.SINLIB.POSIX-BC.070.ROOT
+HD                 35 Apr 10 12:25:08 $TSOS.SINLIB.POSIX-BC.070.ROOT
 PS                 49 Apr 11 11:33:22 $TSOS.SINLIB.POSIX-BC.070.SHELL
 SH                243 Apr 11 12:20:29 $TSOS.SINLIB.POSIX-BC.070.SHELL
 MAN                37 Apr 11 12:14:39 $TSOS.SINLIB.POSIX-SH.070
 WHO                46 Apr 10 15:25:43 $TSOS.SINLIB.POSIX-SH.070
 MORE              107 Apr 11 12:14:39 $TSOS.SINLIB.POSIX-SH.070
 SORT               53 Apr 11 12:20:32 $TSOS.SINLIB.POSIX-BC.070.SHELL
 CRON               59 Apr 10 12:25:31 $TSOS.SINLIB.POSIX-SH.070
 GREP               41 Apr 11 12:02:40 $TSOS.SINLIB.POSIX-BC.070.SHELL
+STTY               39 Apr 11 11:17:06 $TSOS.SINLIB.POSIX-BC.070.ROOT
 EXPR               40 Apr 10 12:24:57 $TSOS.SINLIB.POSIX-BC.070.SHELL
 UNAME              35 Apr 11 11:17:00 $TSOS.SINLIB.POSIX-BC.070.SHELL
+POSDBL             44 Apr 11 12:38:53 $TSOS.SINLIB.POSIX-BC.070.ROOT
+IN@RSHD            97 Apr 10 12:25:11 $TSOS.SINLIB.POSIX-BC.070.INET

# posdbl -b /home/bach/bs2/plamlib.4/l/usp
```

```
# posdbl -l
 RM                 39 Apr 10 12:25:30 $TSOS.SINLIB.POSIX-BC.070.SHELL
+IN@RLOGIND        113 Apr 11 11:16:47 $TSOS.SINLIB.POSIX-BC.070.ROOT
+HD                 35 Apr 10 12:25:08 $TSOS.SINLIB.POSIX-BC.070.ROOT
 PS                 49 Apr 11 11:33:22 $TSOS.SINLIB.POSIX-BC.070.SHELL
 SH                243 Apr 11 12:20:29 $TSOS.SINLIB.POSIX-BC.070.SHELL
+usp                59 Apr 11 12:41:02 /home/bach/bs2/plamlib.4/l/usp
 MAN                37 Apr 11 12:14:39 $TSOS.SINLIB.POSIX-SH.070
 WHO                46 Apr 10 15:25:43 $TSOS.SINLIB.POSIX-SH.070
 MORE              107 Apr 11 12:14:39 $TSOS.SINLIB.POSIX-SH.070
 SORT               53 Apr 11 12:20:32 $TSOS.SINLIB.POSIX-BC.070.SHELL
 CRON               59 Apr 10 12:25:31 $TSOS.SINLIB.POSIX-SH.070
 GREP               41 Apr 11 12:02:40 $TSOS.SINLIB.POSIX-BC.070.SHELL
+STTY               39 Apr 11 11:17:06 $TSOS.SINLIB.POSIX-BC.070.ROOT
 EXPR               40 Apr 10 12:24:57 $TSOS.SINLIB.POSIX-BC.070.SHELL
 UNAME              35 Apr 11 11:17:00 $TSOS.SINLIB.POSIX-BC.070.SHELL
+POSDBL             44 Apr 11 12:41:18 $TSOS.SINLIB.POSIX-BC.070.ROOT
+IN@RSHD            97 Apr 10 12:25:11 $TSOS.SINLIB.POSIX-BC.070.INET
```

4.3 Information for C program interfaces

This section describes special aspects when accessing files of bs2fs file systems via program interfaces.

You can use the *sysfs* interface (see the "C Library Functions for POSIX Applications" manual and "sysfs - query information about the file system type") to determine whether bs2fs file systems are supported in an existing POSIX installation.

You can check whether a bs2fs file system is mounted using the *mount* commandor by searching the *mount* table */etc/mnttab* (see "/etc/mnttab - table of the mounted file systems"). For each mounted file system this table contains an entry with the data of this file system.

4.3.1 errnos

You can access files of bs2fs file systems using the same program interfaces as for when you access files of ufs file systems. However, in the event of an error the *errno* variable can contain additional values or values with a different meaning. The meaning of the various error numbers when you access bs2fs file systems is provided in the table below:

Error number	**Meaning**
EAGAIN	A BS2000 file can temporarily not be accessed, e.g. because the file was opened by another process or because the connection to the pubset has been interrupted.
EDOM	Internal interface error
EFAULT	Internal processing error or invalid address (e.g. buffer address)
EIO	DMS or LMSUP or PLAM reported I/O error
EMACRO	DMS or LMSUP or PLAM reported processing error
ENAME	Part of the path name is not a valid BS2000 file name, PLAM type name or element name (the latter possibly including a valid version identifier)
ENODEV	The volume or the device cannot be accessed
ENOEXEC	An LLM element which is to be edited does not have an LLM structure.
ENONET	Pubset is not available, possibly because of a problem in the MSCF network
ENOSTR	An attempt is being made to access a BS2000 file which does not have a file type which is supported
ENOSUBSYS	No connection could be established to LMSUP or PLAM
ENOSYS	Operation not supported by the bs2fs file system
ENXIO	File was displaced (migrated)
EOPR	Operation is not supported for the specified file
ESRCH	No bs2fs daemon exists which could perform the accesses to the BS2000 object. Response: notify the system administrator

Error values such as *EIO*, *ENOEXEC* or *ENOSPC* can also be set for *close* if the file was opened in write mode and could not be written back properly to the BS2000 file or the library element. In these cases the copy is retained in the bs2fs container.

4.3.2 fstat, stat - bs2fs file structure

The stat structure of the system calls *stat()* and *fstat()* supplies all the important information about a POSIX file. This information is, among other things, also evaluated and edited by the POSIX command *ls*.

In the case of bs2fs files there are a few special features, and these are listed in the table below.

In some cases this information differs depending on whether the file is currently open or closed. When it is open, the file resides in the bs2fs container as a "shadow file".

stat() enables the information to be called for both closed and open objects, *fstat()* only for closed objects.

	Information for closed bs2fs object (stat)	**Information for opened bs2fs object (stat or fstat)**
st_ino	Unique number between a bs2fs *mount* and *umount*	
st_dev	Unique number of the bs2fs file system in which the file identified with *st_ino* resides.	
st_nlink	In the case of DMS files: "normally" 1 In the case of library elements with highest version: 2 In the case of other library elements:1	
st_size	DMS file: last written PAM page in bytes Library element: size of PAM pages in bytes	Size of the shadow file in the container
st_uid	bs2-POSIX-user	
st_gid	bs2-POSIX-group	
st_atime **st_mtime** **st_ctime**	Attributes of the BS2000 file or library element In the case of library elements *st_atime* has the same value as in the library or an element-specific value if LMS is activated	Attributes of the shadow file in the container
st_mode	Attributes of the BS2000 file or library element	

Explanation of the time stamps

st_atime

When an object is opened in read mode, the BS2000 file's time stamp is updated in the context of the BS2000 close operation, which takes place immediately after the shadow file has been copied into the container, which may be necessary. During read accesses in the container, the time stamp is updated following every read access to the shadow file. The time stamps of the BS2000 and shadow files then differ, i.e. the BS2000 file always has an "older" atime status than the shadow file. (While the file is open, the application only ever sees the shadow file's time stamp. After the *close()*, however, it sees the "older" time stamp of the BS2000 file again.)

No access time stamp is normally maintained for library elements; the *st_atime* field then contains the library's access time stamp. Only if the maintenance of access time stamps was defined for the library using the LMS statement //MODIFY-LIBRARY-ATTRIBUTES with the ACCES-DATE=*KEEP operand or with the corresponding LMSUP function will the access time stamps for its elements be updated from this point, and they can then be transferred to the *st_atime* field.

st_ctime

The *st_ctime* field for the file status is not changed after *chmod()*; as for BS2000 files, *st_ctime* represents the creation time (CREATION TIME in FSTAT).

4.3.3 fstatvfs, statvfs - file system information

The program interfaces *fstatvfs* and *statvfs* supply information about the file system which contains a file. This file is specified by a file descriptor in the case of *fstatvfs* and by its name in the case of *statvfs*.

In both cases the information is output in the format which is defined in the *sys/statvfs.h* header in the *statvfs_t* structure.

```
typedef struct statvfs {
  unsigned long f_bsize;         /* Block size of the file system */
  unsigned long f_frsize;        /* Fragment size */
  unsigned long f_blocks;        /* # Blocks on file system with size f_frsize */
  unsigned long f_bfree;         /* # Free blocks with size f_frsize */
  unsigned long f_bavail;        /* # Available free blocks for non-superusers */
  unsigned long f_files;         /* # File nodes (inodes) */
  unsigned long f_ffree;         /* # Free file nodes (inodes) */
  unsigned long f_favail;        /* # Available free inodes for non-superusers */
  unsigned long f_fsid;          /* File system ID (number) */
  char f_basetype[FSTYPSZ];      /* Name of the target file system, Null-terminated */
  unsigned long f_flag;          /* Bit mask for the options of f_flag */
  unsigned long f_namemax;       /* Maximum length of the file names */
  char f_fstr[32];               /* File-system-specific string */
  unsigned long f_filler[16];    /* Reserved for future versions */
} statvfs_t;
```

This format is, however, tailored to ufs file systems. For a bs2fs file system the user must interpret this information as shown in the table below. The data is generated using the BS2000 interface $SJINFO.

`unsigned long f_bsize`

Set to 2048 (PAMBLOCKSIZE).

`unsigned long f_frsize`

Set to 2048 (PAMBLOCKSIZE).

`unsigned long f_blocks` *(*)*

This data is obtained from the corresponding values of the BS2000 interface $SJINFO. However, not all BS2000 values can be mapped to an exact equivalent which is used for ufs file systems.

`unsigned long f_bfree` *(*)*

`unsigned long f_bavail` *(*)*

`unsigned long f_files` *(*)*

`unsigned long f_ffree` *(*)*

`unsigned long f_favail` *(*)*

`unsigned long f_fsid`

Contains the index which is used by the *sysfs* interface.

`char f_basetype[FSTYPSZ]`

Contains the string "bs2fs" for identification.

`unsigned long f_flag`

Only the ST_RDONLY flag is relevant.

`unsigned long f_namemax`

Is set to 38 - 41 in accordance with the catalog and user IDs. This value applies for DMS files and PLAM libraries only. In the case of library elements or types, the variable is assigned the value for the higher-ranking PLAM library. The restrictions for names of elements and element types can only be ascertained with *pathconf*. Within the restriction of "Elementname+Version" to 89 characters it must be borne in mind that the Elementname component is limited to 64 characters and the Version component to 24 characters.

`char f_fstr[32]`

(*) unsigned long long in the case of *fstatvfs64()* or *statvfs64()*

4.3.4 sysfs - query information about the file system type

The *sysfs* program interface supplies information about the file system types configured in a system.

Depending on the function argument, *sysfs* supplies

- the index which corresponds to the file system type with the name specified in the system
- the name which corresponds to the file system type with the specified index
- the maximum number of file system types configured in the system

This interface enables you to check whether a system is capable of processing bs2fs files. The argument for the file system type bs2fs is "bs2fs".

Example

```
if (sysfs(GETFSIND, "bs2fs") < 0) {
    /* bs2fs NOT SUPPORTED */
}
```

4.3.5 Further special aspects and restrictions

Restrictions for accessing files in bs2fs file systems exist for the following program interfaces:

chmod()

This function is accepted only at file and element level. In all other cases it is rejected with EISDIR. The S_ISUID ("set user id on execution") and S_ISGID ("set group id on execution") bits are not defined for BS2000 files and library elements and are ignored (if specified in chmod). The *st_ctime* field for the file status is not changed; as for BS2000 files, it represents the creation time (cf. *fstat*).

chown(), **fchown()**

The functions are rejected with ENOSYS if an attempt is made to modify the user and/or group number.

close()

After errors in the case of *close()* (e.g. with the errno values EIO, ENOEXEC, ENOSPC), the shadow file is retained. With the support of the POSIX administrator it can be used to save the data.

creat(), open()

These functions are accepted only at file and element level. In all other cases they are rejected with EISDIR. A file or library element cannot be opened in write mode simultaneously in BS2000 and in a bs2fs file system. If a file or a library element in a bs2fs file system is opened in write mode, it cannot be opened in another bs2fs file system. If a file is not generated by the owner (user ID of the file system) but by a process with TSOS privilege, it is nevertheless assigned the uid and gid of the file system's owner. If a file generation group (which is not visible in the bs2fs file system) with the specified file name exists, the call is rejected with ENOSTR.

A library element may be generated not only by the owner, but also by any user who has write permission and, if necessary, administration permission for the library or type. However, if the user who generates a library element is not the owner, the protection attributes specified in the mode argument are ignored and standard protection attributes are set.

creat(), open() with O_EXCL

If a file which is not visible in the bs2fs file system exists, in other words one with properties which are not supported (STORAGE-LEVEL != S0, SUPPORT != PUBLIC), the call is rejected with ENXIO. Nevertheless the file remains invisible (e.g. for the *ls* comman*d*), but "bs2cmd fstat ..." can be used to check whether it exists.

link()

This function is not supported. The call is rejected with ENOSYS.

mkdir(), rmdir()

These functions are always rejected with ENOSYS.

open() with O_RDWR under ftyp=text or ftyp=textbin

In the case of text files (*ftyp=text* or *ftyp=textbin*) of the type SAM, CRTE in conjunction with SAM forces the existing record structure to be retained because of *fopen(...,"r+")* in the bs2fs daemon, i.e. in the file area which exists after *open()* each new line must remain unchanged in the position it occupies and no new line may be inserted. New records can be appended and modified at the end of the existing file.

readlink()

This function is always rejected with ENOSYS.

remove(), unlink()

These functions are accepted only at file and element level. In all other cases they are rejected with EISDIR.

rename()

This function is accepted only at file and element level. In all other cases it is rejected with EISDIR. However, it is possible to convert files to library elements and vice versa, e.g. using the *cp* or *mv* command. Library elements of the basic type L cannot, however, be renamed as files or text elements, and files and library elements of the basic types D, J, M, P, S, X cannot be renamed as elements of the basic type L. Renaming an element of the basic type X as a text element (basic type D, J, M, P or S) can corrupt the contents. If the target is simultaneously opened in write mode in BS2000 or another bs2fs file system, *rename()* is rejected with EAGAIN.

rmdir()

This function is always rejected with ENOSYS.

fstat(), stat()

If *stat()* or *fstat()* refers to a file in a bs2fs file system, the information supplied depends on whether or not the file is open. If the file is open, the file size precise to the byte is supplied, and the time stamps are supplied precise to the microsecond, as is customary for POSIX files. If the file is not open, *stat()* supplies the size of the file as specified in BS2000 (i.e. in multiples of 2K blocks), and the time stamps are only precise to the second; the fields with the microsecond details then always
contain 0.

symlink()

This function is always rejected with ENOSYS.

utime(), utimes()

These functions are accepted only at file and element level. In all other cases they are rejected with EISDIR. The st_ctime field for the file status is not changed (cf. *chmod*). Maintaining the access time stamps (access time st_atime) of elements is a special library property which can be set with LMS or LMSUP. By default no access time stamps are recorded for elements; *stat()* then outputs the library's access time stamp. The time at which the time stamps (access time st_atime, modification time st_mtime) for directories (root directory of a bs2fs file system, directories at library and type level) are updated is undefined. Only the current time stamp can be set. If the time stamp specified differs from the current time stamp by more than +/- 3600 seconds, the call is ignored.

5 Access to bs2fs file systems via NFS

This chapter describes the measures required to enable bs2fs file systems to be accessed via NFS and what special features need to be taken into account for this access.

5.1 Sharing bs2fs files

bs2fs files are shared with the NFS command *share.*

Syntax

```
share -F nfs [-o spec_options] [-d description]
path_name
```

The following special features compared to the sharing of ufs files apply here:

The *path name* specified must be the root directory of the bs2fs file system. It may not point to a subdirectory in this file system (PLAM library or member type in a PLAM library).

All client computers which are to be assigned access to the shared bs2fs file system must be specified explicitly with one of the options **rw=** *client*[:*client*]... (for read/write access) or **ro=** *client*[:*client*]... (for read access).

The following spec_options have also been introduced for sharing bs2fs files:

bs2anon=`bs2000_uid`

Specifies the BS2000 user ID under which "anonymous" accesses are to take place. Client processes with a UID other than the POSIX UID of the bs2fs owner then have access to the bs2fs file system only if the *bs2anon* option is used. This option enables the POSIX administrator to define precisely one BS2000 ID with whose rights these client processes access the files of the bs2fs file system concerned.

bs2conv

Specifies that character set conversion is to take place. This option has no arguments. If the option is specified, file contents are converted from EBCDIC.DF.04-1 to ISO 8859-1 when reading and in the other direction when writing.

> **i** This option should always be specified when bs2fs file systems which are mounted with ftyp=text or ftyp=textbin are shared except for clients which process EBCDIC files. If a client is to process ASCII files on a bs2fs file system without converting them, the file system must be mounted with ftyp=bin, otherwise conflicts will arise between ASCII new ilnes (0x0A) and EBCDIC new lines (0x15).

bs2nameconv

Specifies that file name conversion is to take place. This option has no arguments.

When this option is specified, specific file names are converted (see "Converting file names").

5.2 Mounting shared bs2fs files

Shared bs2fs file systems are mounted on the NFS client with the means made available by the client operating system. As a rule the *mount* command is used. Details can be found in the NFS documentation for the relevant client operating system. For information on BS2000 clients, see the manual "NFS".

Here there are no special features compared with the procedure for mounting ufs files. However, please note the information in "Recommendations for mounting bs2fs file systems on NFS clients".

5.3 Security aspects and access rights

To ensure that the NFS clients are not inadvertently granted access rights which undermine the BS2000 security strategies, the following restrictions for the sharing of bs2fs files apply in comparison with the sharing of ufs files:

- Sharing bs2fs files using the *share* command is permissible only under the BS2000 user ID TSOS.
- The administration must define all client computers which are to be granted access to the shared bs2fs file system explicitly with one of the options **rw=**... (for read/write access) or **ro=**... (for read access). Only these clients will be granted access. If a client name is contained in both lists, it will only be granted read access. If it is contained in neither, it is granted no access to the shared bs2fs file system.
- Client processes which run under the UID which matches the POSIX UID of the owner (BS2000 user ID) of the bs2fs files are granted access to the shared bs2fs files with the rights of their owner.

  For client processes which run under a different UID, the *bs2anon* option of the *share* command can be used to define a BS2000 user ID under which these processes can access the shared bs2fs file system. If this option is not specified, these processes are not granted access to the bs2fs file system.

  NFS clients can thus either perform access under the BS2000 user ID of the owner of the shared bs2fs files or – if specified in the *share* command – under the BS2000 user ID assigned for "anonymous" accesses.

  The administrator of the NFS server (BS2000) must verify the trustworthiness of the admitted clients – as is also the case with other NFS shares. In this case the client administrator (root) must be trusted, because this administrator assigns a user his/her UID. Client users with a UID which corresponds to the POSIX UID of the owner of the bs2fs files must also be trusted in the context of the access rights granted to them.

5.4 Converting file names

The namespace of BS2000 files and PLAM library members is as a rule smaller than that of the client system. On the client system applications can therefore exist which cannot or can only with difficulty cope with the restrictions to the namespace which apply on the server. For example, editors create backup copies or buffer information under file names which are not permitted on the bs2fs file system on the server.

When files which have not yet been opened are deleted, NFS clients temporarily create files whose names consist of ".nfs" followed by varying numbers of hexadecimal digits. Consequently these file names are always converted as follows:

File name on the NFS client	File name in the bs2fs file system
`.nfs[:xdigit:][:xdigit:]*`	`@nfs[:xdigit:][:xdigit:]*`

When a bs2fs file system is shared, the *bs2nameconv* option of the *share* command can be used to specify that additional conversions should take place:

File name on the NFS client	File name in the bs2fs file system	Description of the file name on the NFS client and of the conversion rule
`\.[^.].*`	`@[^.].*`	File names containing at least two characters, the first character being a dot and the second character not a dot: the leading dot (.) is converted to an at character (@).
`..*~.*`	`..*#.*`	Filenames that contain the tilde except as the first character: each tilde (~) is converted to a hash (#).

5.5 Further special features when working with bs2fs files on the NFS client

The following special features must be borne in mind when processing bs2fs files systems which are mounted on NFS clients:

- When files or library members are created, copied or renamed, the relevant naming regulations of BS2000 must be observed, otherwise the error number EINVAL ("invalid argument") will be returned to the application when *open()* takes place. This can, for instance, lead to problems in the case of:
  - editors which automatically generate copies of the original file or save intermediate results (.*filename*.swp, *filename*~, etc.)
  - file browsers which generate thumbnails of files and store these temporarily in the relevant directory

  As a rule this can be configured in the editors or browsers either by disabling the function concerned or by selecting a storage location outside the bs2fs file system.
- In order to optimize internal performance, the closing of a bs2fs file on the NFS server is slightly delayed (by 2 seconds) if there is no shortage of space in the bs2fs container. As a result, a file on the NFS server remains open for 2 seconds and from the viewpoint of BS2000 remains locked for this time although, from the viewpoint of the NFS client, it already appears to have been closed.
- The first *open()* of the bs2fs file as a rule takes place when the client application calls the *access()* function. This can result in the response time of the NFS function *ACCESS* being unexpectedly long. The response time in particular also depends on the size of the bs2fs file (see also "Recommendations for mounting bs2fs file systems on NFS clients").
- On UNIX file systems, the hard link count of directories is the same as the number of subdirectories plus two (because of the pseudo directories "." and ".."). In bs2fs file systems the mountpoint and also PLAM libraries can contain subdirectories. For them, the hard link count is set to this value only when the directory contents are read (*readdir* function; used, for example, by the ls command). Before this the count is 2 or the value which the last call of the *readdir* function supplied.
  In some UNIX applications this leads to problems. The *find* command on LINUX clients (GNU variant) can, for example, under some circumstances issue a warning that the hard link count of a directory is incorrect and therefore internal optimization algorithms do not function correctly and suggests that this optimization should be disabled.
- The file type detection on UNIX systems as a rule takes place by means of a heuristic examination of a small part of the file contents. For this purpose the file is opened, a specific quantity of data is read and analyzed, and the file is closed again. In the case of bs2fs files the entire BS2000 file must first be copied into the bs2fs container on the NFS server to permit this, and the larger the file is the longer this process takes.
  This can result in very long wait times on the NFS client which depend on the total size of the files in this directory.

5.6 Recommendations for mounting bs2fs file systems on NFS clients

The following measures may be required on the NFS clients on which bs2fs file systems are mounted:

- Disable the attribute cache

  As the size of a bs2fs file can change when the first *open()* takes place, the file attributes on the NHS client should not be stored in the attribute cache. On NFS clients the *mount* option *noac*, for example, is available to prevent this.

- Permit increased response times

  Because of the unexpectedly long response times, especially with the NFS function *ACCESS*, the timeout values specified on the NFS client must be sufficiently high. On NFS clients the *mount* option *timeo=n*, for example, is provided for this purpose. Depending on the size of the bs2fs files, ten times the default value or even more must be set here.

  If relatively large bs2fs files are to be processed, it may be necessary to perform a socalled "hard mount" (as a rule by omitting the *mount* option *soft*). As a result of this file accesses are always repeated after timeouts.

- Disable the *READDIRPLUS* function

  Using the NFS function *READDIRPLUS* instead of *READDIR* is designed to reduce the number of network I/Os and consequently to enhance the performance. However, a loss of performance may under some circumstances be recognized on some NFS clients. These offer, for example, a *mount* option *nordirplus* in order to disable the use of *READDIRPLUS*.

- Disable input/output buffering

  Some NFS clients use local buffering of the input/output data to reduce the number of network I/Os. This can, however, lead to possible input/output errors in bs2fs files being concealed, as a result of which the application on the NFS client incorrectly assumes that an input/output was successful. Input/output buffering on the NFS client should therefore be disabled. On NFS clients the *mount* option *forcedirectio*, for example, is available for this purpose.

5.7 Examples

The following prerequisite applies for the examples in this section: The BS2000 IDs MORITZ (POSIX UID=110), GAST (POSIX UID=100) and TSOS (POSIX UID=0)exist on the NFS server.

Example 1

The `:DATA:$MORITZ.*.LIB` files are mounted as a bs2fs file system. This file system is shared for reading and writing for the clients *client01* and *client02* and only for reading for the client *client03*. On the clients a user with the UID 110 is granted access with the rights of the BS2000 ID *MORITZ*. Every other user is denied access because the *bs2anon* option is not specified:

```
# mount -F bs2fs -o rw,ftyp=binary ':DATA:$MORITZ.*.LIB' /bs2mnt/moritz_lib
# share -F nfs -o rw=client01:client02,ro=client03       /bs2mnt/moritz_lib
```

Example 2

The `:DATA:$ MORITZ.*.SRC` files are mounted as a BS2000 file system. This file system is shared for reading and writing for the client *client04* and only for reading for the client *client05*. On the clients the user with UID=110 is granted access with the rights of the BS2000 ID *MORITZ* and all others are granted access with the rights of the BS2000 ID *GAST*:

```
# mount -F bs2fs -o rw,ftyp=text ':DATA:$MORITZ.*.SRC' /bs2mnt/moritz_src
# share -F nfs -o rw=client04,ro=client05,bs2anon=GAST /bs2mnt/moritz_src
```

Example 3

The `:HOME:$TSOS.SYSDAT.BCAM.*` files are shared for reading and writing for the super user (UID=0) on the client *client06*; the other users connected to this client are not granted access:

```
# mount -F bs2fs -o rw,ftyp=text ':HOME:$TSOS.SYSDAT.BCAM.*' /bs2mnt/bcam.dat
# share -F nfs -o rw=client06 /bs2mnt/bcam.dat
```

Example 4

The `:HOME:$SYSAUDIT.*CONSLOG*` files are shared for reading for any user connected to client *client07*, the file contents being converted from EBCDIC.DF.04-1 to ISO 8859-1. In other words the files contain EBCDIC texts and are output on the NFS clients as ASCII texts:

```
# mount -F bs2fs -o ro,ftyp=text ':HOME:$SYSAUDIT.*CONSLOG*' /bs2mnt/conslog
# share -F nfs -o ro=client07,bs2anon=SYSAUDIT,bs2conv       /bs2mnt/conslog
```

Example 5

The `:HOME:$MAX.*` files are shared for reading for every user connected to client *client08*, provided the access rights permit this, the file contents being converted from EBCDIC.DF.04-1 to ISO 8859-1. In other words the files contain EBCDIC texts and are output on the NFS clients as ASCII texts. In addition, file names with specific special characters are converted:

```
# mount -F bs2fs -o ro,ftyp=text ':HOME:$MAX.*'   /bs2mnt/max
# share -F nfs -o ro=client08,bs2conv,bs2nameconv /bs2mnt/max
```

6 Diagnosis and enhancing performance

This chapter provides an insight into the internal procedures and describes a diagnostic aid measures for enhancing performance.

The technical details described in this chapter are designed to facilitate diagnosis for an experience system administrator.

6.1 Overview of the mount and unmount operations

This section describes the internal procedures for mounting and unmounting the bs2fs container and bs2fs file systems.

Mounting the bs2fs container

Mounting the ufs file system with the property bs2fs container (*mount -o bs2fscontainer...*) is the prerequisite for mounting bs2fs file systems. The bs2fs container is the physical medium in which the files of bs2fs file systems are stored temporarily during the time they are edited. When the bs2fs container is mounted, it is expected to be empty. If this is not the case because of an error in a previous session, the content is deleted when the bs2fs container is mounted.

A ufs file system can be mounted as these bs2fs container only if it was flagged appropriately when it was created (POSIX installation program). This prevents a ufs file system which was not intended for use as the bs2fs container from being mounted inadvertently as the bs2fs container, which would delete its contents. As additional protection you are recommended as far as possible to mount the bs2fs container automatically.

The bs2fs container should be invisible for the users. Only indirect access to the contents of the bs2fs container should be possible for them, i.e. by accessing the files in the mounted bs2fs file systems. These are redirected internally to the files in the bs2fs container.

Direct access to the bs2fs container should only be possible for the system administrator and solely for diagnostic purposes.

Unmounting the bs2fs container

The bs2fs container can only be unmounted if no bs2fs file systems are still mounted.

Mounting a bs2fs file system

When a bs2fs file system is mounted, a directory with the name *catid*.*userid*.*number* is created in the bs2fs container, where:

catid

catalog ID from the resource operand of the mount command

userid

user ID from the resource operand of the mount command

number

number with which the mount operations for bs2fs file systems are numbered in chronological order within a session

This directory is used to contain maps of objects (file, library, element type or library element) of a bs2fs file system if these have been opened for editing with *open()* or *opendir()*. These maps in the bs2fs container are always single files, even if the corresponding object in the bs2fs file system is presented as a directory. The mounted BS2000 files are therefore not stored physically at the mount point of the bs2fs file system but are located ("invisibly" for the user) in a directory of the bs2fs container reserved for this bs2fs file system.

In addition, directories also exist on the uppermost level of a bs2fs file system. Their map files which are created with *open* or *opendir* are views of these directories which are local to the process and are called %*userid*%*pid*, where *userid* is the BS2000 user ID (8 characters with blanks) of the caller of *open* or *opendir* and *pid* is their process ID, specifically, for example, "`%BACH  %00123`".

The table below shows how the names of the objects in the bs2fs file system are mapped to file names in the bs2fs container:

Type in BS2000	**Name in the bs2fs file system**	**Type in the bs2fs file system**	**Name of the file in the bs2fs container**
DVS file	<file>	File	<file>
PLAM library	<lib>	Directory	<lib>%
Element type	<lib>/<type>	Directory	<lib>%<type>%
Library element	<lib>/<type>/<memb>	File	<lib>%<type>%<memb>+<ver>

where

<file> File name

<lib> Library name

<type> Element type

<memb> Element name

<ver> Element version

The table below shows this mapping again for specific examples:

Object that is opened with open() or opendir()	**Name of the bs2fs file system**	**File name in the bs2fs-Container**
DVS file SOURCEFILE	sourcefile	sourcefile
PLAM library MYPLAMLIB	myplamlib	myplamlib%
Type S subdirectory of the PLAM library MYPLAMLIB	myplamlib/s	myplamlib%s%
Type S element SOURCE1 of the PLAM library MYPLAMLIB (where the highest element version is 001)	myplamlib/s/source1	myplamlib%s%source1+001

The map files in the bs2fs container are created with first *open()* or *opendir()* for the relevant object and are deleted again with the last *close()* or *closedir()*. No map files exist in the bs2fs container for files or directories which are accessed in a different manner.

Unmounting a bs2fs file system

When a bs2fs file system is unmounted, the corresponding directory in the bs2fs container is deleted.

6.2 Diagnosis

In the event of an error the information regarding which files stored in the bs2fs container correspond to which file system may be required. The *get_container_index* command is provided to ascertain this assignment.

6.3 get_container_index assign bs2fs container index to bs2fs file system

This command provides the administrator with the assignment of the index of a directory in the bs2fs container to the corresponding bs2fs file system.

Syntax

```
get_container_index [mountpoint]
```

No option specified

Output takes place for all mounted bs2fs file systems.

mountpoint

Output takes place only for the file system to which the specified file belongs.

Example 1

Listing the information about all the mounted bs2fs file systems

```
# get_container_index
index = 3 for /home/bach/bs2.2 (:V70A:$BACH.PLAMLIB*)
index = 2 for /home/bs2.2 (:V70A:$BACH.CCC.*.C)
index = 1 for /home/bach/bs2.1 (:V70A:$BACH.ASS.*.S)
```

Example 2

Listing the information about a particular bs2fs file system

```
# get_container_index /home/bs2.2
index = 2 for /home/bs2.2
```

6.4 Correcting errors

When a file in a bs2fs file system has been modified and cannot be copied back to BS2000 again after processing has been completed, the ufs file concerned is moved to a special bs2fs_lost+found directory in the bs2fs container. If a file which has the same name and stems from an earlier failed attempt to write it back exists there, it is overwritten without an inquiry.

The user can issue the *bs2fs_recover* command to obtain a list of files which have been relocated in this way and then (for example with a modified name) copy these to BS2000 and/or delete them in the bs2fs container.

Alternatively this task can also be performed by the system administrator in the role of a deputy.

To prevent files which have been relocated to the bs2fs_lost+found area from remaining unnoticed, when the bs2fs container is mounted a message is output on the console which provides information on the user IDs for which files exist in this area. The bs2fs_lost+found area is of course not deleted like the rest of the bs2fs container when the latter is mounted.

The *bs2fs_recover* command provides a simple way to check whether files of a specific ID exist in the bs2fs_lost+found area. When intensive use is made of the bs2fs file system, it is recommendable to configure an automatic check with this command, e.g. in the logon script (*$HOME/.profile* or */etc/profile*).

In order to get to know how the command works and the mechanisms associated with it, you can use the POSX file */etc/.bs2fsdSimulateCloseError* to simulate errors which occur when the bs2fs files are written back. If this file (which requires no content) exists, the ufs file is relocated to the bs2fs_lost+found area irrespective of the success of its being written back to BS2000. If the */etc/.bs2fsdSimulateCloseError* file is removed or renamed, the simulation will be canceled.

6.5 Performing recovery for the bs2fs_lost+found area

This command offers the following functions:

- Listing the files in the bs2fs_lost+found area
- Outputting the number of these files
- Generating a shell script which copies the files from the bs2fs_lost+found area to BS2000 and/or deletes them from this area
- Direct copying and/or deletion of files in the bs2fs_lost+found area, either in interactive mode with an inquiry or in an automatic mode with default settings.

Syntax

Format 1:

```
bs2fs_recover [-l] [-m level]
              [-a after] [-b before] [-u user|*ALL] [file selection]
```

Format 2:

```
bs2fs_recover {-g|-x} [-d] [-w] [-c] [-f n|y] [-p prefix] [-s suufix]
              [-a after] [-b before] [-u user|*ALL] [file selection]
```

Main options

The main options define which function is to be executed. Only one main option may be specified.

No main option specified

All selected files are listed, which corresponds to main option *-l*.

-l

The number or a list of selected files is output to the default output. The *-m* option defines the scope of the output.

-g

A shell script is generated which performs the actions specified with the *-d* und *-w* options when it is executed. The generated shell script is output to the default output.

-x

The actions specified with the *-d* und *-w* options are performed immediately. With these actions an output is by default only made in the event of an error. If you want to see the progress of processing, you should specify the -v option at least once so that the file name concerned is listed before the actions for this file are executed.

Additional options

The additional options are used to modify the action concerned or for file selection.

-m level

This option controls the scope of the output when the *-l* option is specified. If the *-m* option is not specified, *-m 1* is assumed.

It is obligatory to specify *level*. The following options can be specified:

0

The number of files found is output to the default output in a line with the format "*Number* file(s)". in this case the end status also supplies information on whether at least one file was found (end status = 0) or not (end status = 1). This provides a simple way of checking whether the bs2fs_lost+found area contains any of your own files.

1

A list of files is output to the default output. This contains the date and time of the last modification, the file size and the name of the associated BS2000 file or the complete specification of the corresponding library member (*library name*(*member name*,*type*,*version*)). The name is prefixed by the serial number of the bs2fs mount, separated by a "/".

2

Output as for *-m 1* with the following additional information:

File processing mode of the bs2fs mount T, TB, B (for ftyp text, textbin or binary) and absolute ufs file name in the bs2fs container (bs2fs_lost+found area).

3

Output as for *-m 2* with the following additional information:

Access method (PAM, SAM, ISAM or PLAM), block format, block length, record format and record length.

If certain information, such as record foramt, does not exist for PAM files, "-" is output instead. You can use this information to write your own scripts for copying and deleting the files.

Example of an output with -m 3:

```
2012-03-20 09:39:06 8 3/:90GB:$TSOS.TSOSDAT T /grossercont
/bs2fs_lost+found/TSOS/:90GB:3:TD01SV00000:tsosdat SAM DATA (STD,1) V -
```

-d

In conjunction with the main options *-g* and *-x*: The selected files are deleted from the bs2fs container area bs2fs_lost+found. This option can also be used together with the *-w* option.

-w

In conjunction with the main options *-g* and *-x*: The selected files are copied to BS2000. This option can also be used together with the *-d* option.

-c

Suppresses the use of the catalog ID and user ID employed to date when copying to BS2000 (*-w* option). This option can be used (if necessary together with the *-p* option) to perform backup under a different catalog ID/user ID from the one specified in bs2fs-mount.

-f n|y

Determines whether existing BS2000 files and LMS members are overwritten when copying to BS2000 (*-w* option).

Depending on the main option, the *-f* option has the following effects:

Main option *-g* (Generating a shell script)

When the *-w* option is specified, the *OV* variable is always set in the script. This controls the behavior when copying to BS2000 while the generated script is executed.

When the *-f* option is not specified or *-f n* is specified, the *OV* variable is assigned the value "NR", as a result of which BS2000 files are not overwritten when copying takes place. When *-f y* (*OV*="Y") is specified, they are overwritten.

When the *-w* and *-d* options are specified simultaneously, files are deleted only if the preceding copy operation was successful either because the file did not yet exist in BS2000 or because it was possible to overwrite it successfully.

If the script is generated only for the purpose of deletion (*-d* option), the *OV* variable is not set. The files in the Lost+Found area are then deleted without an inquiry irrespective of the *-f* option.

Main option *-x* (Immediate deletion or copying)

The use of the *-f* option decides whether an inquiry should be made before each action takes place (interactive mode) or not (automatic mode).

When the *-f* option is not specified, the following inquiries are made:

1. Should the file be copied (with the *-w* option)
2. Should the file be overwritten (with the *-w* option if the target file already exists)
3. Should the file be deleted from the Lost+Found area (with the *-d* option).

If one of these inquiries is answered with "no", no further inquiries or actions take place for the same file. In particular, a file is not deleted if copying or overwriting was rejected beforehand.

When the *-f y* or *-f n* option is specified, no inquiries are made either when copying (*-w*) or when deleting (*-d*). When copying, depending on the *-f* option existing files are either overwritten (*-f y*) or not (*-f n*).

-p `prefix`

Specifies a string with which the BS2000 file name is prefixed when copying to BS2000 (*-w* option). If *prefix* contains a catalog ID and/or a user ID, the *-c* option must also be specified to permit a valid BS2000 file name to be created.

-s `suffix`

Specifies a string which is appended to the BS2000 file name when copying to BS2000 (*-w* option).

-a `after`

The file selection is restricted to files which were last modified after the time specified with *after* .

The time is specified as follows: `[[YY]yy]MMDDhhmm[.SS]`

`[YY]yy`

Year (two or four characters)
A two-character year specification is complemented as following: *yy* > 68 means 19*yy*, *yy* <= 68 means 20*yy*.
If no year is specified, the current year is assumed.

`MM`

Month, two characters (01 through 12)

`DD`

Day, two characters (01 through 31)

`hh`

Hour, two characters (00 through 23)

`mm`

Minute, two characters (00 through 59)

`SS`

Second, two characters (00 through 61)
The values 60 and 61 are provided for leap seconds.
If second is not specified, 0 is assumed.

`-b` `before`

The file selection is restricted to files which were last modified before the time specified with *before*. The format of the time specification is described under the *-a* option.

`-u` `user`

The file selection is restricted to the files of the specified user ID. The entry for the user ID is not case-sensitive. As a rule, because of the default attributes of the subdirectories in the Lost+Found area (`drwx --- ---`) owners of nonprivileged IDs can only view and edit their own files.

-u *user* not specified:

The BS2000 user ID of the caller is used.

-u *ALL (not case-sensitive):

All BS2000 IDs are processed for which files are contained in the Lost+Found area. As a rule this specification is only relevant for the system administration TSOS ID) as only the system administration has the necessary rights to copy and delete files in foreign IDs.

`-v`

Additional information on the actions of the tool is output. This option can also be specified more than once (up to three times) in order to increase the degree of detail of the information.

`file selection`

file selection is used to control the selection of the files located in the Lost+Found area (in addition to the selection with the *-u* option).

Sample printouts can be used like those which are offered by the shell for creating file names. These printouts are then used for the BS2000 file names and LMS member specifications, as the tool outputs them when the *-l* option is used.

Example:

In the following output with *-l -m1* the file name printed in bold is relevant for thesample printout.

```
2012-03-20 09:39:06 8 3/:90GB:$TSOS.TSOSDAT
```

In this case the complete file name (without sample printout) would also have to include the prefix 3/.

file selection not specified:

All existing files of the calling user.

6.6 Measures to enhance performance

Check whether the following points were taken into account for the selection of BS2000 files which are to be made available in a bs2fs file system:

- The wildcard in the *resource* option should be selected in such a manner that if possible only the BS2000 files which are actually needed are mounted as the number of mounted files influences the performance.
- If it is syntactically not possible or only extremely difficult to separate the required files from those which are not required, one of the following options for bypassing this problem should be considered before mounting files for the first time: export the files required to another pubset (under the same or a different user ID) or rename these BS2000 files with meaningful names.
- You should avoid the same BS2000 files being mounted several times at different positions. This occurs when multiple *mount* operations are executed with identical or overlapping *resource* specifications. If such a file is open for write accesses in one of the bs2fs file systems involved, it cannot be opened simultaneously in another bs2fs file system.
- If required, start additional copy daemons with the *start_bs2fsd* command.

7 Glossary

bs2fs container

A file system of the type ufs which is used solely to accommodate files of bs2fs file systems temporarily. These files are then stored in directories (one directory for each mounted bs2fs file system) in the bs2fs container.

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.

copy daemon

System process which copies from BS2000 to the bs2fs container and back again.

daemon

Daemons are system processes which run permanently and normally in background mode, and which perform general tasks.

directory

A directory is used to group and organize files and subordinate directories of a hierarchical file systems.

file system

A file system is a hierarchical group of directories and files which are located physically on the same storage medium, e.g. in a partition or in a container file.

The term is used for organizational structures of files, such as UNIX file system, POSIX file system, hierarchical file system, other BS2000 file systems (DMS and LMS), for example.

file system type

Type of a file system in the file tree of POSIX or of a UNIX computer. The most familiar types are as follows:

Type *bs2fs*: BS2000 files which are made available under POSIX.
Type *ufs*: local file systems containing user data.
Type *nfs*: file systems which are located physically on remote computers.

Examples of further types: *fdfs*, *proc*, *bfs*, *rfs* and *s5*.

All types of file systems used in a system must be known when the system is configured; no new file system type can be added later. Information about the file system types which can be used in a system is provided by the sysfs() interface.

file tree

Overall hierarchy of the files on a UNIX computer or in POSIX. The UNIX or POSIX file hierarchy is organized on the basis of a tree structure. The root of the file tree is the root directory (/). All other directories are branches which emanate from the root. The files are the leaves of the tree. Each file is accessible via precisely one path of the file tree.

mounting

Local resources are made accessible in a local file system using the *mount* command; this is also referred to as mounting.

POSIX file system

File system on a BS2000 computer running POSIX. The POSIX file system corresponds to a UNIX file system.

root directory

The file system at which the file tree begins. The root directory is represented by the slash (/).

transparent file access

The user can access bs2fs files like ufs files (with the few restrictions described) using the same access functions or commands.

ufs (UNIX file system)

The ufs file system is the local standard file system in POSIX. It is implemented by a BS2000 file (of the type PLAM) in which the files of the file system are stored as on a UNIX disk.

8 Related publications

You will find the manuals on the internet at http://manuals.ts.fujitsu.com. You can order manuals which are also available in printed form at http://manualshop.ts.fujitsu.com.

[1] **POSIX**
Basics for Users and System Administrators
User Guide

[2] **POSIX**
Commands
User Guide

[3] **C/C++**
C Library Functions for POSIX Applications
Reference Manual

[4] **NFS**
NFS users and NFS administrators
User Guide

[5] **BS2000**
Introduction to System Administration
User Guide

[6] **BS2000**
Commands
User Guide

[7] **EDT**
Statements
User Guide

[8] **EDT**
Unicode Mode Statements
User Guide