Gesellschaft für
Parallele Systemtechnik mbH

parsytec

H e l i o s   Series

Software Product

HELIOS FILE SYSTEM

Autor :
A. Ishan

Helios File System

Software Documentation

Version 1.1, September 1989

This software was written by A.Ishan. Additional work was done by M.Clauss and H.J.Ermen.

Printed in West-Germany

C o n t e n t

1. Introduction

The Helios File-System (HFS) is a powerful tool to handle mass-storage devices in multi-Transputer networks. Various applications which need highest data transfer rates can be realized - especially in combination with SCSI-devices which can be accessed by Parsytec's "Mass-Storage-Controller" (MSC) board. Typical fields for applications which can make use of this product are for example image-processing and real-time and distributed databases.

1.1 Main Features:

- The Helios-File-System (HFS) is based on the Berkeley Fast-File-System (FFS) up from version 4.2BSD. A lot of extensions have been made to achieve maximal performance and guarantee an optimal use of the disk-capacities.

- The basic block size is set to 4 KBytes. In dependancy on the client's request, the server is able to build "packets" of various sizes in it's buffer cache to fit the request in an optimal way. The packet-handling is done in a transparent manner to the client tasks.

o Directories and files are placed on the disk or partition by using different strategies to get a good compromise between spreading and clustering of information (minimal fragmentation). To implement such balancing procedures, the file-server divides the disk into a number of equal sized "cylinder-groups" which are controlled separately.

o In contrast to the BSD Fast-File-System the inode-handling was implemented in a more straight-forward way: Inodes and entry-names build "Directory-Entries" which are kept in special directory-blocks. There is no fixed upper limit of the number of directory-entries in the file-system.

o The HFS is designed as a device-independant standard Helios-server. This means especially, that the server can be placed in the Transputer-network on a node by the user's choice. The topology of the target network is therefore nearly irrelevant! More than one HFS can be installed and used in one Helios network (only the MSC-version!) which can be accessed by a nearly unlimited number of users tasks.

o The HFS also offers an inexpensive alternative to the MSC-board and SCSI-devices: If a PC or compatible machine is used as a host, it is possible to make use of a "rawdisk-interface". The rawdisk requires a separate partition on the PC's hard-disk which has to be formatted physically first and on which the logical file-system is created by a special Helios-utility. (see below) o To achieve maximal flexibility for the user, the server is full parametrizable. With the desired hardware and application environment in mind, it is possible to manipulate the disk- and the buffer-cache parameters in a very wide range.

o Various utilities are added to the software - package to ease handling of the server. Especially protection mechanisms based on capabilities are enabled.

o A file-system consistency checker and more tools for diagnostic work will be added in the next release of the Helios File System (HFS release 1.2).

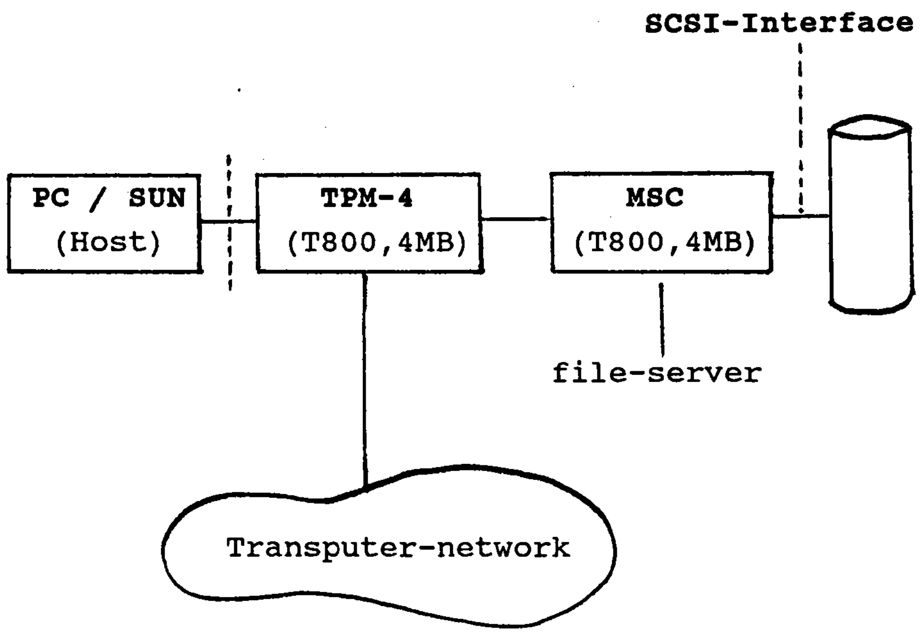
2. Hardware Requirements

The distribution includes three device drivers in source-code and binary format which build the interface between the device independant layers of the file-server and the physical storage device. The sources of these device drivers are part of the package to demonstrate how a device driver has to be written. The following devices are especially supported:

2.1 MSC - Version

The MSC-version makes use of Parsytec's MSC-board which offers 4 MBytes RAM and a T800 Transputer for processing. The HFS will run on this processor and builds up it's buffer-cache structures by making use of the MSC's memory. A suitable SCSI-device with a capacity of up to 2 GBytes can be easily added. At the moment only hard-disks are supported - future versions will also allow access to streamers and optical drives.

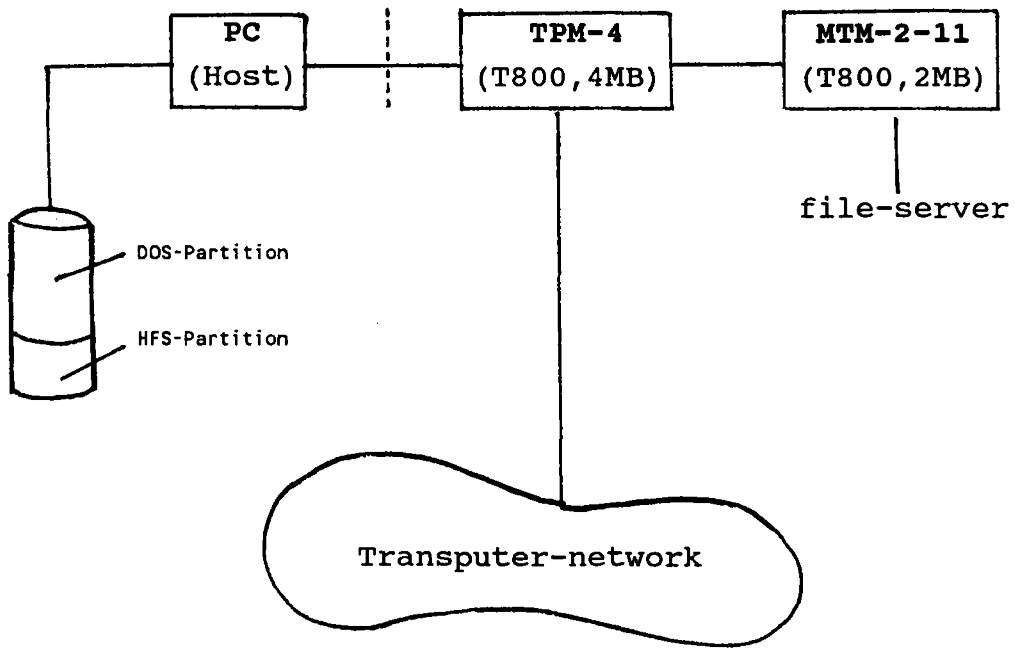
The following example demonstrates how the HFS is integrated into a system which makes use of the MSC and a SCSI-device.



2.2 Rawdisk - Version

As mentioned before, the rawdisk-version makes use of a separate partition of the PC's harddisk. To allow comfortable work, there should be at least 20 MBytes on this partition. The server can be placed in this case on a processor by the user's choice. The only thing to be kept in mind is that this node should offer enough memory to install a suitable buffer-cache.

An example for a Helios-network, working with the HFS in combination with the Rawdisk-device is shown below:



3. Software - Requirements

To work with the HFS properly, the "Helios distributed operating system" is required. The software is able to run with Helios version 1.1 (/s and /n) and upper.

4. Installation

The Helios File System is shipped on one 5,25" (DD) disk in IBM-PC compatible format. You should find the following directories and files on this disk:

The server program "fs" and various utilities :

```
a:\bin\chmod
a:\bin\fs
a:\bin\fsync
a:\bin\gdi
a:\bin\matrix
a:\bin\refine
a:\bin\sync
a:\bin\termfs
```

The device-info source files and binary objects :

```
a:\etc\devinfo.src
a:\etc\devinfo
```

Device driver programs :

```
a:\lib\raw.dev
a:\lib\msc.dev
a:\lib\msc02.gen
```

Example device drivers :

a:\device\raw	: The PC rawdisk device-driver
a:\device\msc	: The MSC (SCSI) device-driver
a:\device\m212	: The M212 device-driver

New and modified header-files :

```
a:\include\device.h
a:\include\module.h
```

i) The first step is to copy all files from the subdirectories of the distribution disk to the associated directories of the host's file-system. This can be done for example by :

```
xcopy  a:  c:\helios  /s   (under MS-DOS)

or

cp -r  /a/*  /helios      (under Helios)
```

ii) **Device dependant notes:**

- If you are using a rawdisk-device, you have to prepare the desired partition on your PC-harddisk under MS-DOS with the utility **MAKEDISK.EXE**, which is found in the subdirectory \DEVICE\RAW. One of the tasks of MAKEDISK is to erase the File Allocation Table (FAT) of this partition.

  Note that this partition is - after formatting - for exclusive use of the HFS. No MS-DOS files can be placed there simultaneously !

- The rawdisk-device has to be declared in the HOST.CON configuration file. The following line has to be added:

Add for example

```
rawdisk_drive = d ,
```

if you want to make use of the DOS-partition "d" on your harddisk.

- If you are using the MSC as your mass-storage system, the file **MSC02.GEN** which contains some OCCAM low-level supporting routines has to be placed in the /lib - directory.

- You should declare the processor on which the HFS is installed as a SYSTEM-node in your resource-maps to prevent the Task-Force Manager from executing other tasks on this node.

iii) Booting the file-server

After installation and booting the Helios-network, an empty file system can be created on disk by executing "fs" with the option "-m" on the desired processor:

```
remote <processor> fs -m <device>
```

This causes the logical formatting routine to create an empty file-system depending on the informations kept in the device description file (see below!). If the server shall be rebooted another time <u>without</u> formatting, the following command line has to be executed:

```
remote <processor> fs <device> &
```

To ease handling at system startup-time, this line can be placed for example in the "loginrc" script-file.

Note that the user has to take care, that the desired processor really exists as a Helios-node and offers enough memory to install buffer-cache data structures. If the MSC-board is used it is recommend that the server is always executed on this processor!

5. Defining Disk Devices

All device-dependant informations associated with the physical disk device which shall be used by the HFS are kept in a "device information file" (devinfo.src). This file resides in the /etc - directory and can be easily modified by using a text editor. A special utility "gdi" generates a binary object - named "devinfo" - from the device information file. This file also has to be placed in /etc where it is accessed during the file-server's boot phase.

Some "advanced" utilities for optimization of the buffer-cache layout - especially the possibility to choose buffer-cache packet-sizes totally free - will be supplied in the next release of the HFS (version 1.2). This version will also contain a comprehensive Technical Manual which offers a deeper view into the mechanism and techniques which are used by the HFS.

The following listing is an example which shows, how informations are arranged in the device information file.

```
#       Example for a device information file
#    ********************************************
#

# A M212 - device

fileserver m212
{
        device          m212
        cachesize       100
        syncop          0
        volume {
                name            fs
                partition       0
        }
}

# The PC-rawdisk environment

fileserver raw
{
        device          raw     # Type of discdevice
#       cachesize       1000    # Cache size in kBytes (excluded)
        syncop          0       # Partly synchronous mode
        smallpkt        1       # Blocks per small cache packet (const.)
        mediumpkt       4       # Blocks per medium cache packet (const.)
        hugepkt         16      # Blocks per huge cache packet (const.)
        smallcount      10      # Number of small cache packets (<= 10)
        mediumcount     4       # Number of medium cache packets (<= 4)
        hugecount       14      # Number of huge cache packets (<= 14)
        volume
        {
                name            fs      # Name table entry
                partition       0       # currently not used
                cgsize          256     # Blocks per cylinder-group (<= 256)
                ncg             8       # Number of cylinder-group (<= 8)
                cgoffset        3       # Offset in cylinder-group (const.)
                minfree         0       # currently not used
        }
}

# The MSC (SCSI) environment

fileserver msc
{
        device          msc     # Type of discdevice
#       cachesize       2000    # Cache size in kBytes (excluded)
        syncop          0       # Partly synchronous mode
        smallpkt        1       # Blocks per small cache packet (const.)
        mediumpkt       4       # Blocks per medium cache packet (const.)
        hugepkt         16      # Blocks per huge cache packet (const.)
        smallcount      20      # Number of small cache packets (<= 40)
        mediumcount     8       # Number of medium cache packets (<= 15)
        hugecount       28      # Number of huge cache packets (<= 50)
        volume
        {
                name            fs      # Name table entry
                partition       0       # currently not used
                cgsize          2560    # Blocks per cylinder-group (<= 3072)
                ncg             4       # Number of cylinder-group (<= 300)
                cgoffset        3       # Offset in cylinder-group (const.)
                minfree         0       # currently not used
        }
}
```

```
discdevice m212
{
        name            m212.dev
        controller      3
        addressing      1
        mode            0x11
        partition {
                drive           0
                start           2
        }
        drive {
                id              1
                type            1
                sectorsize      512
                sectors         17
                tracks          4
                cylinders       612
        }
}

discdevice raw
{
        name            raw.dev
        controller      0
        addressing      1
        partition {
                drive           0
        }
#       drive {
#               id              0
#               type            0
#               sectorsize      512
#               sectors         17
#               tracks          6
#               cylinders       176
#       }
}

discdevice msc
{
        name            msc.dev
        controller      0
        addressing      1
        partition {
                drive           0
        }
#       drive {
#               id              0
#               type            0
#               sectorsize      512
#               sectors         17
#               tracks          6
#               cylinders       769
#       }
}

serialserver link3
{
        device          link.dev
        address         3
}
```

6. T u t o r i a l   I n s t a l l a t i o n :

The following example checklist should help you to setup a new file-system and work with it. For this example we use the configuration as described below:

- A PC acting as a host with the partition "d" for use by the HFS (rawdisk-version)

- A network with a node named "HFS" which offers a T800 and 4 MBytes RAM (a TPM-4 for example) to keep the file-server with all data structures.

Startup :

The first step is to prepare the MS-DOS partition for working with the HFS. Execute the following command from MS-DOS level. Make sure that you have saved all files from the desired partition, because the whole MS-DOS filing-system on this logical drive will be erased!

```
makedisk  d
```

Add the following line to your HOST.CON - file:

```
rawdisk_drive  =  d
```

Then the Helios network can be booted as usual. (Make also sure that the processor "HFS" is declared as a SYSTEM-node in the resource-map.):

```
server
```

After logging in, you can prepare an empty file-system, based on the informations which are kept in the **devinfo** - file. Take a look at the declarations for the "raw" file-system and the "raw"-device in the example above. If you want to change some parameters, you have to recompile the devinfo-file:

```
pushd  /helios /etc

emacs  devinfo.src

<editing>
```

Compilation of a modified device information file:

```
gdi  devinfo.src  devinfo

popd
```

An empty Helios file-system will be created on the rawdisk-partition by executing:

```
remote  HFS  fs  -m  raw
```

Note that the file-server is not booted after formatting the disk. To perform the initial server boot you should type in:

```
remote  HFS  fs  raw  &
```

Now the server is booted and a name-table entry is created on the processor "HFS". To access the root of the file-system you can use for example an absolute pathname:

```
cp  xyz  /raw
```

This command will copy the file "xyz" into the root-directory of the file-system.

Finishing :

To terminate the file-server properly before shutting down the whole system, you should generate a "last sync" to write all blocks to disk which are marked as "delayed write". Note that this step is **always** recommended, if you are working in the "**partly** synchronous" mode (syncop = 0 in the device information file)

```
sync  /raw
```

The server itself is terminated by performing

```
termfs  /raw
```

7. Commands

The following commands can be executed from the command line and are supplied to make better use of the Helios File System.

c h m o d

Purpose: Alter the protection bits of a file

Format: chmod [vxyz][+-:] [rwefghvxyzda] <file>

Description:

chmod is used to enable and disable the protection bits of a file. For more details see: Technical Information, chapter 13 , "Protection and Authentication".

f s

Purpose: The binary object of the Helios File System

Format:

```
fs -m <device>    , to create a new file -
                    system
fs <device> &     , to boot the file-server
```

Description:

fs contains the binary code of the file server. If it is called with the option -m, the formatting routine is executed to generate a new file-system on the disk. After completion of this step the program terminates. A call of fs without the option -m causes the file-server to be booted. To place the server on a processor of your choice make use of the utility remote.

f s y n c

Purpose: Toggle between partly and fully synchronous mode

Format: fs <file server> [-as]

Description:

fsync allows the selection between two operation modes: At server startup-time the default mode is the "partly synchronous mode" (-a) which means that all data-blocks are written with a certain delay (of max. 20 seconds) to disk, when the "sync-process" - which is part of the server - becomes active and detects some of them. To guarantee that <u>all</u> blocks are written directly to disk ("write-through-cache"), the user has the alternative to switch to fully synchronous mode (-s) , which eliminates all delyed-write operations.

g d i

Purpose: Compile a "device information file"

Format: gdi <input> <output>

Description:

gdi is a simple compiler which generates a binary object from the given device information file. The default filename which is searched by the server is "devinfo". This file has to be placed in the /etc - directory.

m a t r i x

Purpose: Display the access matrix of a file

Format: matrix <file>

Description:

The utility matrix displays the access matrix of the given file. For more details see: Technical Information, chapter 13 , "Protection and Authentication".

r e f i n e

Purpose: Refine or restrict a capability

Format: refine <file>

Description:

refine allows refining and restricting of capabilities associated to a file. For more details see: Technical Information, chapter 13 , "Protection and Authentication".

s y n c

Purpose: Force a sync-operation immediately

Format: sync <file server>

Description:

The utility sync forces an "extra" sync-operation which guarantees that all data-blocks in the buffer-cache with the "delayed-write" flag set are written immediately to disk. sync is especially useful to guarantee consistency, if the file-server or the whole system shall be shut-down.

t e r m f s

Purpose: Terminate an active file-server

Format: termfs <file server>

Description:

termfs closes all ports to a file-server, deallocates the memory in use and finally terminates the file-server program. The user has to take care, that all client programs are in a proper state and that the last sync-process has been succesfully executed.