

HP-UX Memory File System (MemFS) 2.0

Administrator's Guide

Second Edition



Manufacturing Part Number: 5992-2360

August 2007

© Copyright 2007 Hewlett-Packard Development Company L.P.

Legal Notices

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Trademark Notices

UNIX is a registered trademark of The Open Group.

1. Introducing HP-UX Memory File System (MemFS)

Overview	7
Need for a Memory-Based File System	8
Architecture of HP-UX Memory File System	9
MemFS User Process Backing Store	10
Application Areas for HP-UX Memory File System	11
Design Limitations	12
Known Problems	12
Other Limitations	12

2. Installing, Configuring, Tuning, and Removing HP-UX Memory File System

Installing HP-UX Memory File System	15
Installing HP-UX MemFS in the Non-Interactive Mode	15
Installing MemFS in the Interactive Mode	15
Configuring Memory File System	16
Tuning Memory File System	17
Removing Memory File System	18

Contents

About This Document

The *HP-UX Memory File System (MemFS) 2.0 Administrator's Guide* describes how to install, configure, and tune the HP-UX MemFS 2.0.

Intended Audience

This document is for system administrators responsible for installing and configuring MemFS on HP-UX 11i v2. Readers are expected to have knowledge of:

- HP-UX System administration concepts
- UNIX operating system concepts
- UNIX File system concepts

Document Organization

The *HP-UX Memory File System (MemFS) 2.0 Administrator's Guide* is divided into the following chapters:

Table 1 **Document Organization**

Chapter	Description
Chapter 1 "Introducing HP-UX Memory File System (MemFS)," on page 7	Describes the architecture of HP-UX Memory File System.
Chapter 2 "Installing, Configuring, Tuning, and Removing HP-UX Memory File System," on page 15	Describes the steps for installing, configuring, tuning, and removing Memory File System.

Typographic Conventions

Table 2 describes the typographic conventions used in this document.

Table 2 **Typographic Conventions**

Typeface	Usage	Examples
monospace	Computer output, files, directories, software elements such as command options, function names, and parameters	Read tunables from the <code>/etc/vx/tunefstab</code> file. See the <code>ls(1)</code> manpage for more information.

Table 2 **Typographic Conventions (Continued)**

Typeface	Usage	Examples
<i>italic</i>	New terms, book titles, emphasis, variables replaced with a name or value	See the <i>HP-UX Memory File System (MemFS) 1.0 Administrator's Guide</i> for details.
%	C shell prompt	Not applicable
\$	Bourne/Korn shell prompt	Not applicable
#	Superuser prompt (all shells)	Not applicable
\	Continued input on the following line; you do not type this character	# mount -F vxfs \ /h/filesys
[]	In command synopsis, brackets indicates an optional argument.	ls [-a]
	In command synopsis, a vertical bar separates mutually exclusive arguments.	mount [suid nosuid]
blue text	An active hypertext link	In PDF and HTML files, click on links to move to the specified location.

Related Documentation

- *MemFS v2 -A Memory-based File System on HP-UX 11i v2* whitepaper.

1 Introducing HP-UX Memory File System (MemFS)

This chapter introduces you to the HP-UX implementation of Memory File System (MemFS).

The following topics are discussed:

- Overview
- Architecture of HP-UX Memory File System
- Application Areas for HP-UX Memory File System
- Design Limitations
- Known Problems
- Other Limitations

Overview

A Memory-based File System (MemFS) is a file system that resides in memory. It does not normally write data out to stable storage. A MemFS is created from a mount operation, and ceases to exist when it is un-mounted. The purpose of such a file system is to provide fast access for temporary files that do not need to be kept for an indeterminate time. Because it does not normally have to do I/O to stable storage, the MemFS is able to provide extremely high throughput.

Keeping data in memory comes at a cost. It consumes system physical memory. Even with today's large-memory systems, physical memory comes at a premium. A system or application that runs out of available memory at a critical time can cause irreparable loss to the user. This is the reason why most Virtual Memory (VM) management systems implement a paging policy, wherein less frequently used memory pages are paged out to a swap device. This policy has been extended to MemFS. Under memory pressure, the VM system can deallocate MemFS pages and re-assign them where needed.

MemFS requires a user process to be associated with each of its instances. The memory allocated in the process address space of this user process is used to store the data from the reallocated pages. These user memory pages can be paged out to the system swap device whenever there is system memory pressure.

Need for a Memory-Based File System

A memory-based file system is typically used as storage for temporary files. By keeping as much data as possible in memory, it avoids having to perform disk I/O and the associated overhead. Traditional file systems manage two types of data. One is the file content (called data), which an application accesses through the read, write and mmap mechanisms. The other, called metadata is the information related to file attributes and the file system structure.

Most file systems implement a buffering mechanism for data. This implies that a copy of the disk data blocks is maintained in memory (buffer cache or page cache, based on the implementation). This buffer is used to service the reads from a disk, and to implement a "delayed write" mechanism. Using a buffer helps the file systems avoid disk I/O to some extent and improve the file system performance. Structural and attribute changes are buffered to a lesser extent, and will usually cause disk I/O to be done.

A memory-based file system goes a bit further to avoid disk I/O. Metadata is always kept in memory, so the overhead of writing structural changes to the disk is totally avoided. MemFS data blocks are treated mostly at par with other file system blocks and can be swapped out of the buffer cache. There is a difference, though. Disk based file systems try to keep the disk copy up to date with the buffer copy. So, "dirty" buffers (those that have been modified) are periodically written back to the disk. This is important since these file systems are expected to preserve data across system shutdown and reboot. A memory-based file system has no such requirement, so data is written to the disk only if that buffer is being paged out.

Before memory-based file systems became popular, the concept of a "RAM disk" existed. A RAM disk reserves a range of memory and makes it available through a block device interface using a pseudo driver. The block device interface permits a file system to be created on it, in effect providing a memory-based file system. However, RAM disks have certain drawbacks. Since memory is reserved at the time of the disk creation, it is locked from shared system use, whether actually in use or not. Most RAM disks do not support paging of their memory, resulting in very poor system response in cases where the system is running low on free physical memory.

Since the system sees a RAM disk as a device rather than a file system, any access to a file stored on it results in a second copy of the data being kept in the file system buffer. Some of the implementations of a RAM disk may provide an improved interface and some of these shortcomings may not be present.

As attractive as the performance aspects sound, a memory-based file system only works efficiently under certain circumstances. The biggest disadvantage is that it does not preserve data across mounts. It cannot be used to replace file systems that store persistent data. Since it uses memory which is a shared resource, excessive use can adversely affect other memory consumers. A typical use of a memory file system is for storing temporary files that are

created, accessed frequently in a short span of time and then deleted. The advantage comes from not having to update large amounts of metadata on file creation - growing the file - and then deletion and cleanup.

The write-to-swap feature of memory-based file systems efficiency has direct correlation with the swap activity. As the swap activity increases, the file system performance decreases.

Architecture of HP-UX Memory File System

The memory file-based system which is available as a configurable product in HP-UX 11i v2 is based on the buffer cache approach.

HP-UX uses the Hierarchical File System (HFS) as the base for MemFS¹. HFS² provides a well-designed and tested file system implementation. It has all the features needed for a basic file system. Some of the more advanced file system features (e.g. journaling, un-buffered I/O) that some of the contemporary file systems provide are neither required nor relevant in the context of a memory file system.

A MemFS is created from a *mount* (1M) command. Initially the metadata for the file system is created in memory, and the necessary VFS³ and inode structures are initialized. A file system has the concept of disk layout, which refers to how information such as the superblock, inodes and the data blocks are organized on the disk. In a MemFS this disk information is simulated, and kept in memory. The mount results in a combination of an *mkfs*(1M) and a *mount*(1M) without the disk operations.

Subsequent to a mount, the MemFS can be used just like any other file system, with the same commands and system calls. From an application or user perspective there should be nothing that differentiates it from any other file system, except a better response time.

-
1. The term MemFS is used to refer to the HP-UX implementation of a Memory File System.
 2. HFS – Hierarchical File System, the file system traditionally used by Unix based Operating Systems
 3. VFS- Virtual File System – an abstraction layer on top of a more concrete file system.

Internally a MemFS behaves differently from other file systems. The *syncer* (1M)¹ daemon, which periodically flushes "dirty" file system pages to disk, does not have an effect on MemFS buffers, as MemFS buffers are never added to the syncer dirty list - which is processed periodically for flushing. Periodically the VM paging algorithm reallocates the less used pages from the buffer cache and user address spaces. MemFS data pages, too, can be recycled, and written to the user memory allocated in the user address space of a process associated with each mount instance. When accessed, the data is retrieved and copied to the buffer cache.

A *umount* (1M) on the MemFS removes the file system and all its data. As per the specifications, data cannot be recovered once a *umount* (1M) happens.

MemFS User Process Backing Store

MemFS requires a user process to be created for every MemFS mount instance. The address space of the user process is used to store less frequently used MemFS data pages that have been moved out of the buffer cache. The user process continues to exist as a daemon, sleeping in the kernel, as long as the MemFS instance is mounted.

The decision to use a user process as the backing store was driven by the fact that the HP-UX 11i v2 kernel does not have support for pageable kernel memory.

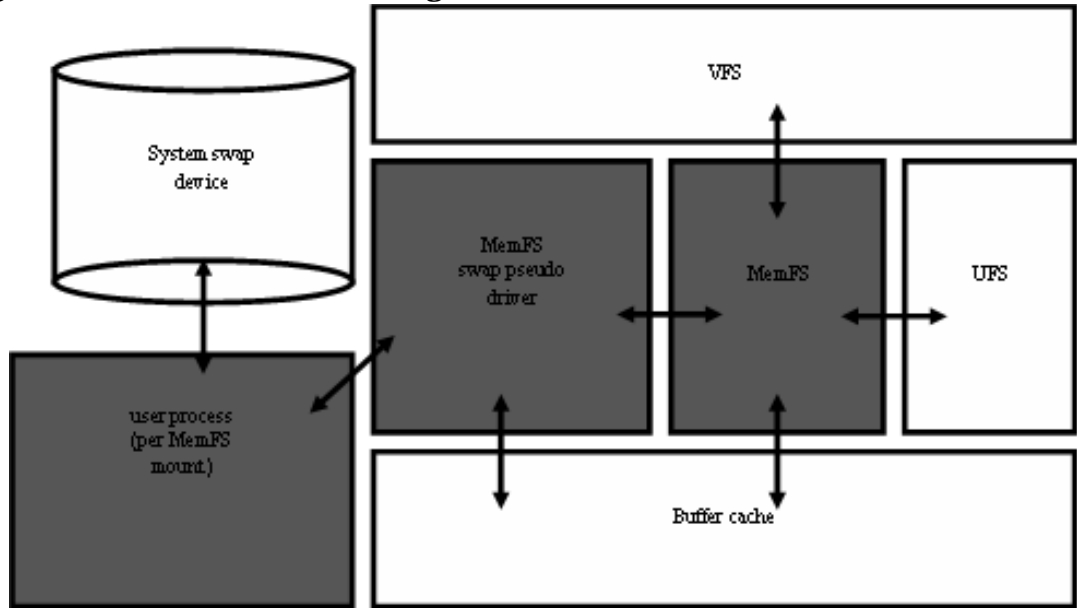
Currently the HP-UX 11i v2 file systems use the buffer cache approach for managing the data buffers. HP-UX 11i v3 has a unified file cache (UFC) model.

MemFS stores all data and metadata in the buffer cache. MemFS buffers are treated differently from other buffers in that they are not normally chosen for replacement through LRU policies. Since MemFS file data is stored in the buffer cache, rather than cached like other file systems, performance is improved. However, when the number of MemFS buffer pages exceeds the number computed by the tunable `memfs_bufcache_swappct`, data from least recently used MemFS pages will be copied to the user address space of the associated user process. These MemFS buffers are then released back to the buffer cache pool. Pages in the user pageable memory can be chosen by the paging daemon to be paged out to the system swap device and brought back to the buffer cache when they are accessed.

A pseudo driver called `memfsswap` acts as a pseudo disk and maintains a dynamic mapping of MemFS memory pages and their virtual disk location on the user process.

1. The terms used here refer to those used with the HP-UX. Other implementations may use a different terminology.

Figure 1-1 MemFS Backing Store



Application Areas for HP-UX Memory File System

Applications that will benefit most from HP-UX MemFS are those that perform mostly metadata intensive operations like creating and deleting small files and directories. For example compilers, editors and sorting applications. MemFS never writes metadata for files to disk, so directory and file manipulation is always a performance gain. Applications can also interact with MemFS files using the mmap interface.

Some other usage guidelines are:

- Use MemFS only for temporary files
- Avoid too many files in a directory. Distribute the files evenly across a larger number of directories. If all files are accessed simultaneously limit the number of files to about 100 per directory or so.

Design Limitations

MemFS uses the Hierarchical File System (HFS) as its base, and, as a result, inherits several HFS functions that increase usage overhead. This includes the overhead of a disk based file system structure, static allocation, and pre-initialization of inodes. Also, lack of kernel pageable memory in HP-UX 11i v2 creates a requirement to pre-allocate user memory at mount time, increasing memory requirements for MemFS. These limitations will be addressed in a future MemFS version.

Known Problems

- MemFS stores all metadata and data in the buffer cache. If a large number of MemFS files and file systems are created, the buffer cache gets filled by the MemFS buffers causing the other file systems to under perform as the buffer cache used by all the file systems is common.
 - On large memory usage of MemFS file system instances, such as when MemFS pages occupy memory equal to the total RAM, the system may experience slow responses.
-

Other Limitations

- Since many HP-UX applications store critical data in `/tmp`, not being able to recover `/tmp` can cause serious damage to the consistency of such applications. HP recommends that you must not use MemFS to mount `/tmp` and any file systems which contain critical data that require consistency.

A limitation with MemFS and Ignite-UX (and other products) is that there is no support for `/tmp` or for `/var/tmp` as MemFS file systems. MemFS file systems should be used only to hold application transient files. The file system and all its contents will be lost with an unmount or reboot. Ignite-UX uses the information stored in `/tmp` during the install process and therefore the `/tmp` file systems cannot be MemFS based.

Since MemFS uses the buffer cache, it has some of the same limitations with regards to many processes accessing a single buffer. This may have performance implications when many processes access the same file at the same time.

These limitations will be the focus of future versions of MemFS.

2 Installing, Configuring, Tuning, and Removing HP-UX Memory File System

This chapter describes how to install, configure, tune and remove HP-UX Memory File System (MemFS).

The following topics are discussed:

- Installing HP-UX Memory File System
- Configuring Memory File System
- Tuning Memory File System
- Removing Memory File System

Installing HP-UX Memory File System

You can install HP-UX Memory File System (MemFS) either non-interactively or interactively.

Installing HP-UX MemFS in the Non-Interactive Mode

To install MemFS in non-interactive mode, enter the following command:

```
# swinstall -x reinstall=false -x autoreboot=true -s <depot-path> MemFS
```

Installing MemFS in the Interactive Mode

To install MemFS in interactive mode, enter the following command:

```
# swinstall -x reinstall=false -x autoreboot=true -s <depot-path>
```

NOTE If VxFS 4.1 is installed on the system, the new EnableVxFS bundle (which delivers `/sbin/mount_enh`) command must be installed on the same system where MemFS v2 is installed. If VxFS 4.1 has been installed previously and the new EnableVxFS bundle, that supports MemFS v2, has not been installed, MemFS mounts fail.

Configuring Memory File System

A MemFS instance requires allocation of a memory area to be associated with it, whose size is equal to the size of the MemFS file system instance. The mount command will fail if it cannot allocate enough memory. Sufficient disk space must be configured to the system swap device for mount to succeed. If the MemFS file system size to be created is large (greater than 2 GB), the tunable `maxdsiz_64bit` must be tuned appropriately. This memory area is allocated in the user address space of a process that is forked off the mount command for every mount.

The maximum number of MemFS file system instances supported in HP-UX 11i v2 is 64.

A MemFS instance can be created either through a `mount(1)` command, or automatically through an entry in the `/etc/fstab` file. Creation of a MemFS instance through the `mount(2)` system call is not supported.

The syntax for mounting a MemFS instance using the `mount(1)` command is:

```
# /usr/sbin/mount -F memfs [-eQV] [-o size=<size>] directory
```

For example, to create a MemFS instance of size 100MB and mounting it on `/memfs`:

```
# /usr/sbin/mount -F memfs -o size=100MB /memfs
```

The optional size argument can be used to specify the size of a MemFS instance. Append to size, kb or KB to indicate the value is in kilobytes, mb or MB to indicate megabytes, or gb or GB to indicate gigabytes. If the size option is not specified, the default size is computed as the percentage of maximum buffer cache size that can be occupied by MemFS, as specified by the `memfs_bufcache_swappct` tunable.

A MemFS instance can be destroyed through a `umount(1)` command. Only the directory mount point is required as an argument:

```
# /usr/sbin/umount /memfs
```

Tuning Memory File System

Tuning the MemFS buffer cache size is very important because MemFS stores data and metadata in the buffer cache. The amount of memory in the buffer cache used for MemFS can be tuned using the following dynamic tunables on HP-UX 11i v2. These tunables can be modified without a system reboot.

`memfs_bufcache_swappct`

`memfs_bufcache_swappct` is the percentage of buffer cache at which MemFS buffers start swapping. The default value is 50. The `memfs_bufcache_swappct` tunable can be as low as 0 and can be as large as 100. If `memfs_bufcache_swappct` is set to 0, MemFS data buffers would start swapping immediately to the user process. If `memfs_bufcache_swappct` is set to 100, MemFS can occupy the entire buffer cache before it starts swapping. The larger `memfs_bufcache_swappct` is, the greater the amount of buffer cache that will be reserved for MemFS pages and the better the performance of MemFS. But this may adversely impact the performance of other filesystems that use the buffer cache.

`memfs_bufcache_metapct`

`memfs_bufcache_metapct` is the maximum percentage of buffer cache which can accommodate MemFS metadata of all MemFS instances. The default value is 10. The `memfs_bufcache_metapct` tunable cannot be less than 10 and cannot be greater than 50. Tuning `memfs_bufcache_metapct` to a value as high as 50 may be necessary if the system primarily uses large MemFS filesystems. The larger `memfs_bufcache_metapct` is, more of the buffer cache will be reserved for MemFS metadata pages and hence will allow creation of more and larger MemFS filesystem instances, but impact the performance of other filesystems that use the buffer cache. Since MemFS metadata pages can never be swapped out to the user process, exercise caution when tuning `memfs_bufcache_metapct` as this may severely impact normal performance of other filesystems.

WARNING **Shrinking the buffer cache (through `kctune` or `sam`) when any MemFS file systems are mounted should NOT be done. Doing so may cause the system to be unstable. If the buffer cache needs to be down-sized manually, all MemFS file systems should first be unmounted.**

NOTE The `memfs_bufcache_swappct` and `memfs_bufcache_metapct` tunables are dynamic on HP-UX 11i v2 and can be modified without a system reboot.

Removing Memory File System

If the MemFS usage is expected to be high on the system, then it is desirable to tune the buffer cache size to greater than 75% of system memory size. This can be tuned using the command `kctune dbc_max_pct=<pct value>`.

If the MemFS usage is expected to be low on the system, then the buffer cache size need not be resized.

Removing Memory File System

To remove MemFS, enter the following command:

```
# swremove -x autoreboot=true MemFS
```