# Tunable Kernel Parameters

## HP-UX Release 11i Version 1.6

**Edition 2**

Customer Order Number: ONLINE ONLY


**Manufacturing Part Number : TKP-90203**
**E0203**

Printed in United States of America

© Copyright 2000-2003 Hewlett-Packard Company. All rights reserved.

# Legal Notices

The information in this document is subject to change without notice.

*Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.* Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Please read the enclosed Hewlett-Packard Software Product License Agreement and Limited Warranty before operating this product. Rights in the software are offered only on the condition that the customer accepts all terms and conditions of the License Agreement.

Operating the product indicates your acceptance of these terms and conditions. If you do not agree to the License Agreement, you may return the unused product for a full refund.

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

### Warranty

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

### Restricted Rights Legend

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY
3000 Hanover Street
Palo Alto, California 94304 U.S.A.

Use of this document and any supporting software media (CD-ROMs, flexible disk, and tape cartridges) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs, in their present form or with alterations, is expressly prohibited.

### Copyright Notices

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

### Trademark Notices

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through The Open Group.

# Revision History

This guide's printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The part number changes when extensive technical changes are incorporated.

New editions of this manual incorporate all material updated since the previous edition.

| | |
|---|---|
| Edition 2 | February 2003. Part number TKP-90302. Online only.<br>HP-UX Release 11i Version 1.6 |
| Update 1.1 | February 2002. Part number TKP-90202. Online only.<br>HP-UX Release 11i (Version 1.0). |
| Edition 1 | December 2001. Part number TKP-91201. Online only.<br>HP-UX Release 11i (Version 1.0). |

# Conventions

We use the following typographical conventions.

| | |
|---|---|
| *audit* (5) | An HP-UX manpage. On docs.hp.com and on the Instant Information CD, it may be a hot link to the manpage itself.<br><br>For example, *audit* is the name and *5* is the section in the *HP-UX Reference*. From the HP-UX command line, you can enter "`man audit`" or "`man 5 audit`" to view the manpage. See *man* (1). |
| *Book Title* | The title of a book. On the web and on the Instant Information CD, it may be a hot link to the book itself. |
| **KeyCap** | The name of a keyboard key. Note that **Return** and **Enter** both refer to the same key. |
| *Emphasis* | Text that is emphasized. |
| **Emphasis** | Text that is strongly emphasized or a heading. |
| **Term** | The defined use of an important word or phrase. |
| `ComputerOut` | Text displayed by the computer. |
| **`UserInput`** | Commands and other text that you type. |
| `Command` | A command name or qualified command phrase. |
| *`Variable`* | The name of a variable that you may replace in a command or function or information in a display that represents several possible values. |
| `[ ]` | The contents are optional in formats and command descriptions. If the contents are a list separated by `\|`, you *may* choose one of the items. |
| `{ }` | The contents are required in formats and command descriptions. If the contents are a list separated by `\|`, you *must* choose one of the items. |
| ... | The preceding element may be repeated an arbitrary number of times. |
| `\|` | Separates items in a list of choices. |

# Contents

# Contents

# 1 Overview

## Tunables by Subsystem

The tunable kernel parameters are divided into the following subsystems and groups.

# Tunable Kernel Parameters

Tunables (or configurable kernel parameters) are kernel variables that allow the operating system to be configured to fit specific system needs, resulting in better performance and/or more efficient allocation of resources. The ideal value for each parameter is often determined by the system's particular hardware configuration, the specific mix of applications the system runs, and the trustworthiness of system users; factors that vary widely from system to system.

HP attempts to provide reasonable default parameter settings, but it may be necessary or beneficial to modify these settings to better suit the needs of your particular system's users. Refer to the individual manpage to obtain detailed information about any configurable kernel parameter.

Individual parameters usually pertain to a specific subsystem; some are independent, but others are interrelated or interact with each other. The parameter descriptions are grouped according to subsystem in “Tunables by Subsystem” on page 10.

**IMPORTANT**

All HP-UX tunable kernel parameters are release-specific. They may be added, removed, or have their meanings changed in future releases of HP-UX.

The tunable parameters described in this manual are specific to HP-UX Release 11i Version 1.6. Though most of the names are the same as those in the previous releases, the behavior of specific tunables may differ significantly from previous releases.

As technological advances expand system size and capabilities, it is not uncommon for maximum and/or default values of certain parameters to change between releases or to vary between 32-bit and 64-bit processors. The information provided in this manual is believed to be accurate as of the time it was written. However, system changes may occasionally result in differences between this manual and what is actually present on your system.

**CAUTION**

Changing kernel parameters to improper or inappropriate values or combinations of values can cause data loss, system panics, and other (possibly very obscure and/or difficult to diagnose) operating anomalies.

- *Read the documentation*. Before altering the value of any configurable kernel parameter, be sure you know the implications of making the change.
- *Keep records*. Write down the parameter name and its old value as well as the new value in case you need to restore the old value.
- *Check ranges*. Never set any system parameter to a value outside the allowable range for the parameter. (SAM refuses to store values outside the allowable range.)
- *Be aware of side effects*. Many parameters interact, and their values must be selected in a balanced way.

## Manpage Layout

The tunable manpages described in this manual are available on HP-UX 11i Version 1.6 systems with the `man` command as well as on **`docs.hp.com`** and on Instant Information and through the *kcweb* (1M) interface.

The manpages for tunable kernel parameters have a layout that is different from other manpages. The special headings and subheadings are described below.

*NAME* This section contains the name or names of the tunable and a short description. Some manpages describe more than one tunable. The names are the names of the tunables as they appear in a *kcweb* (1M) or *kmtune* (1M) display.

*VALUES* This section contains subsections that describe the values that can be applied.

*Default Value* The value that is set on most systems when they are shipped. For most tunables, there should be no need to change the value from the default.

*Allowed Values* The range of values that can be used for the tunable. Using a value that is outside the range specified can produce unexpected results.

*Fail-safe Value* The value that is recommended for use when using the system in maintenance mode. Set this value when you are trying to recover from errors that have been caused by erroneous tuning of the system. If you set the tunable to this value, the system will boot at least in single-user mode without trouble caused by tunables set to improper values.

*Recommended Values* Recommended settings for the tunable.

*DESCRIPTION* This section describes what the tunable does and the information that you need to set the value of the tunable in a manner that is customized to the environment.

Under this section several questions that are relevant to the tunable are answered. The questions include:

- *Who Is Expected to Change This Tunable?*
- *Restrictions on Changing*
- *When Should the Value of This Tunable Be Raised?*
- *What are the Side Effects of Raising the Value?*
- *When Should the Value of This Tunable Be Lowered?*
- *What are the Side Effects of Lowering the Value?*
- *What Other Tunable Values Should Be Changed at the Same Time?*

# How to Specify Configurable Parameter Values

All configurable kernel parameters must be specified using an integer value or a formula consisting of a valid integer expression. All can be specified as an integer, and most can be specified using a formula; but only a minority are usually specified using formula values.

## Entering Values

Use the `kcweb` web interface or the `kmtune` command to view and change values. `kcweb` is described in the *kcweb* (1M) manpage and in the program's help topics. You can run `kcweb` from the command line or from the System Administration Manager (SAM); see *sam* (1M). You run `kmtune` from the command line; see *kmtune* (1M) for details.

## Integer Values

You can use any (optionally signed) integer value.

## Formula Values

When using formulas to specify a parameter value, the formula must be an integer expression. In other words, every element in the expression must be an integer value as defined for the C programming language.

Configurable parameter names are usually lowercase but the values assigned to them are specified in formulas as the same name in capitals. For example, the value assigned to `npty` (by a C-language `#define` statement) is `NPTY`. Thus, a typical formula for `nfile` is:

```
((16*(NPROC+16+MAXUSERS)/10)+32+2*NPTY)
```

where `NPROC` represents the defined value of `nproc`, `MAXUSERS` represents the defined value of `maxusers`, and `NPTY` represents the defined value for `npty`. There are a few isolated exceptions to the uppercase/lowercase rule on certain systems, but they are few.

Whitespace (spaces and tabs) are not allowed.

## Undocumented Kernel Parameters

Some of the configurable parameters that appear in the kernel master file are not documented, and some are not known to or are not supported by kcweb and kmtune for any of several possible reasons:

- The parameter is obsolete. It is no longer used in current HP-UX releases, but might appear in an existing kernel configuration. If kcweb and kmtune encounter an obsolete parameter in the current kernel configuration, they do not display it in the list of configurable parameters that can be changed. They also remove that parameter when creating the new configuration file used to build the pending kernel.
- The parameter is not supported by kcweb and kmtune. As with obsolete parameters, it is not displayed in the list of configurable parameters, but it is retained in the new kernel configuration file to ensure that no malfunctions are introduced due to a missing parameter.
- The parameter is assigned a value by kernel configuration software, which is frequently based on external factors. The assigned value might be used when calculating values for one or more other parameters.
- The parameter is for HP factory or support use only. No change from the default value should be made unless specifically directed otherwise by official HP support personnel.
- The parameter supports obsolete or obsolescent software. For information about how to select a nondefault value, consult the documentation furnished with the software that the parameter supports.
- The parameter supports independent software. For information about how to select a nondefault value, consult the documentation furnished with the software that the parameter supports.

# 2 Accounting Subsystem

## Accounting Parameter Summary

Two configurable kernel parameters are related to accounting:

| | | |
|---|---|---|
| Accounting | `acctresume` | Resume accounting when free space on the file system where accounting log files reside rises above `acctresume` plus `minfree` percent of total usable file system size. Manpage: *acctsuspend* (5). |
| Accounting | `acctsuspend` | Suspend accounting when free space on the file system where accounting log files reside drops below `acctsuspend` plus `minfree` percent of total usable file system size. Manpage: *acctsuspend* (5). |
| Accounting | `max_acct_file_size` | Maximum size of the accounting file. Manpage: *max_acct_file_size* (5). |

### Additional Information


- “Overview of Accounting Parameters” on page 17
- “Specifying a Threshold Value” on page 18
- “How to Specify Configurable Parameter Values” on page 13

# Overview of Accounting Parameters

When activated by the system administrator, the process accounting subsystem maintains log files for storing information about system and user processes. These files can become large, reducing available space for other files on the file system where they reside.

If process accounting is running, the system suspends process accounting whenever the available space on the file system where the accounting files reside falls below a certain threshold. The threshold is defined as the signed, arithmetic sum of the `acctsuspend` kernel parameter value at system boot time and the `minfree` file system parameter value that was defined during file system creation. (See "How to Determine Value of minfree" on page 19.) When process accounting is suspended, the system issues the message:

```
Accounting suspended
```

and accounting remains suspended until enough free space becomes available to resume. The resume threshold value is defined by the signed, arithmetic sum of the `acctresume` kernel parameter and the `minfree` file system parameter. To prevent suspend-resume conflicts, `acctresume` must have a value greater than `acctsuspend`.

When sufficient file system space becomes available and accounting resumes, the system issues the message:

```
Accounting resumed
```

For more information about selecting values for `acctsuspend` and `acctresume`, see "Specifying a Threshold Value" on page 18.

## Suspend Accounting Threshold

The suspend accounting threshold can be set to any value from zero to 100 percent of the usable file system size. Any value for `acctsuspend` which, when added to `minfree`, produces a zero or negative result, sets the suspend threshold at 0 percent, allowing accounting files to overwrite the entire reserved minimum free space without being suspended.

## Resume Accounting Threshold

The resume accounting threshold also can be set to any value from zero to 100 percent of the usable file system size. If the signed, arithmetic sum of the `acctresume` value at system boot time plus `minfree` is 100 or greater, accounting cannot resume until the system is rebooted, or a startup command is executed from the keyboard (see *acctsh* (1M)). In a more practical sense, any value that results in a resume threshold higher than maximum normal free space on the file system will prevent process accounting from resuming.

# Specifying a Threshold Value

acctsuspend and acctresume are integers that each specify a percentage of total usable space on the file system where the accounting log files reside. This percentage is added to the current value of minfree to determine what percentage of total file system space must be available for process accounting to continue operating.

Threshold values are determined by adding the specified value for acctsuspend or acctresume to the current value of minfree. If the value of either parameter is less than zero, that value reduces the free space requirement, allowing the accounting files to invade the protected minimum free space area reserved by minfree. If the sum of acctsuspend and minfree is zero or negative, accounting can continue until the entire file system is full, rendering the value of acctresume meaningless.

## Calculating Threshold Values

The value chosen for acctsuspend or acctresume represents the difference between the corresponding suspend-accounting or resume-accounting threshold value and the current value of minfree for the file system where accounting log files reside.

- Nonnegative values for acctsuspend and acctresume are added to the current value of minfree to determine the suspend or resume threshold, respectively, as a percentage of total usable file system size. Accounting cannot invade free-space area.
- Negative values for acctsuspend and acctresume are subtracted from the current value of minfree to determine the suspend or resume threshold as a percentage of total usable file system size. Accounting suspension or resumption, respectively, occurs inside the reserved minimum free-space area.
- A suspend-accounting threshold value that is zero or negative allows accounting to continue until the file system overflows.
- A resume threshold value that is 100 or greater, or that is larger than the maximum expectable free space on the file system at any time, prevents accounting from resuming until the system is rebooted or accounting is manually restarted using the startup command (see *acctsh* (1M)).

## Examples

Suspend when less than 15% of total file system space is available, resume when available space reaches 18%:

```
minfree = 10
acctsuspend = 5
acctresume = 8
```

Suspend when less than 2% of total file system space is available, resume when available space reaches minfree:

```
minfree = 5
acctsuspend = -3 (minus 3)
acctresume = 0
```

Suspend when available file-system space shrinks to minfree; resume when available space increases to minfree plus 3% of total file system:

minfree = 5
acctsuspend = 0
acctresume = 3

Suspend when file system is completely full, do not resume (resume when available space reaches 105% of total file system which cannot happen):

minfree = 5
acctsuspend = -5 (minus 5)
acctresume = 100

## How to Determine Value of minfree

`acctsuspend` and `acctresume` specify accounting thresholds based on the current value of the `minfree` parameter for the file system where the accounting files reside. `minfree` is specified when the file system is being created, and determines what percentage of the total file system space is reserved and cannot be used by ordinary users. The value can range from zero to 100.

The default value for `minfree` on HP-UX file systems is 10 percent. To determine the current value of `minfree`, use the `df` command with the `-t` option (see *df* (1M)).

For example, to find the current value of `minfree` on the mounted file system `/users`, use the command:

```
df -t /users
```

## Additional Information

- “Overview of Accounting Parameters” on page 17
- “Accounting Parameter Summary” on page 16
- “How to Specify Configurable Parameter Values” on page 13

# 3 Asynchronous I/O Subsystem

## Asynchronous I/O Parameter Summary

The following kernel parameters are used for managing asynchronous I/O operations. The first four (`aio_*`) are related to POSIX asynchronous I/O operations; the last (`max_async_ports`) pertains to open ports between processes and the asynchronous disk I/O driver.

See also “How to Specify Configurable Parameter Values” on page 13.

| | | |
|---|---|---|
| Asynchronous I/O | `aio_listio_max` | Maximum number of POSIX asynchronous I/O operations allowed in a single `lio_listio()` call. Manpage: *aio_listio_max* (5). |
| Asynchronous I/O | `aio_max_ops` | System-wide maximum number of POSIX asynchronous I/O operations allowed at one time. Manpage: *aio_max_ops* (5). |
| Asynchronous I/O | `aio_physmem_pct` | Maximum percentage of total system memory that can be locked for use in POSIX asynchronous I/O operations. Manpage: *aio_physmem_pct* (5). |
| Asynchronous I/O | `aio_prio_delta_max` | Maximum priority offset (slowdown factor) allowed in a POSIX asynchronous I/O control block (`aiocb`). Manpage: *aio_prio_delta_max* (5). |
| Asynchronous I/O | `max_async_ports` | System-wide maximum number of ports to the asynchronous disk I/O driver that processes can have open at any given time. Manpage: *max_async_ports* (5). |

# 4 File System Subsystem

# File System Parameter Summary

Configurable parameters related to file system performance fall into the following categories.

See also “How to Specify Configurable Parameter Values” on page 13.

## File System Buffer Parameter Summary

Allocate system physical memory resources for static and dynamic file system buffer cache space. For more information, see “File System Buffer Parameters” on page 27.

| | | |
|---|---|---|
| File System: Buffer | bufpages | Number of 4 KB pages in file system static buffer cache. Manpage: *bufpages* (5). |
| File System: Buffer | dbc_max_pct | Maximum percentage of memory for dynamic buffer cache. Manpage: *dbc_max_pct* (5). |
| File System: Buffer | dbc_min_pct | Minimum percentage of memory for dynamic buffer cache. Manpage: *dbc_min_pct* (5). |
| File System: Buffer | disksort_seconds | Maximum wait time for disk requests. NO MANPAGE. |
| File System: Buffer | nbuf | System-wide number of static file system buffer and cache buffer headers. Manpage: *nbuf* (5). |

## Journaled File Systems (VxFS) Parameter Summary

Allocate space for the Directory Name Lookup Cache (DNLC) associated with VxFS file system inodes.

| | | |
|---|---|---|
| File System: Journaled | vxfs_max_ra_kbytes | Maximum amount of read-ahead data, in KB, that the kernel may have outstanding for a single VxFS file system. Manpage: *vxfs_max_ra_kbytes* (5). |
| File System: Journaled | vxfs_ra_per_disk | Maximum amount of VxFS file system read-ahead per disk, in KB. Manpage: *vxfs_ra_per_disk* (5). |
| File System: Journaled | vx_fancyra_enable | Enable or disable VxFS file system read-ahead. NO MANPAGE. |
| File System: Journaled | vx_maxlink | Number of subdirectories created within a directory. NO MANPAGE. |
| File System: Journaled | vx_ncsize | Memory space reserved for VxFS directory path name cache. Manpage: *vx_ncsize* (5). |
| File System: Journaled | vx_ninode | Number of entries in the VxFS inode table. NO MANPAGE. |

## Logical Volume Manager (LVM) Parameter Summary

Control kernel interaction with the Logical Volume Manager. For more information, see “Logical Volume Manager (LVM) Operation” on page 30.

| | | |
|---|---|---|
| File System: LVM | maxvgs | Maximum number of volume groups configured by the Logical Volume Manager on the system. Manpage: *maxvgs* (5). |

## Open or Locked Files Parameter Summary

Set number of files that can be open or locked simultaneously. For more information, see “Open or Locked Files Parameters” on page 29.

| | | |
|---|---|---|
| File System: Open/Lock | maxfiles | Soft limit on how many files a single process can have opened or locked at any given time. Manpage: *maxfiles* (5). |
| File System: Open/Lock | maxfiles_lim | Hard limit on how many files a single process can have opened or locked at any given time. Manpage: *maxfiles_lim* (5). |
| File System: Open/Lock | ncsize | Inode space needed for directory name lookup cache (DNLC). NO MANPAGE. |
| File System: Open/Lock | nfile | Maximum number of files that can be open simultaneously on the system at any given time. Manpage: *nfile* (5). |
| File System: Open/Lock | nflocks | Maximum combined number of file locks that are available system-wide to all processes at one time. Manpage: *nflocks* (5). |
| File System: Open/Lock | ninode | Maximum number of open inodes that can be in memory. Manpage: *ninode* (5). |

## Disk Read-Ahead Parameter Summary

Control the amount of memory available for disk read-ahead operations.

| | | |
|---|---|---|
| File System: Read | hfs_max_ra_blocks | The maximum number of read-ahead blocks that the kernel may have outstanding for a single HFS file system. Manpage: *hfs_max_ra_blocks* (5). |
| File System: Read | hfs_max_revra_blocks | The maximum number of reverse read-ahead blocks that the kernel may have outstanding for a single HFS file system. Manpage: *hfs_max_revra_blocks* (5). |
| File System: Read | hfs_ra_per_disk | The amount of HFS file system read-ahead per disk drive, in KB. Manpage: *hfs_ra_per_disk* (5). |
| File System: Read | hfs_revra_per_disk | The amount of memory (in KB) for HFS reverse read-ahead operations, per disk drive. Manpage: *hfs_revra_per_disk* (5). |
| File System: Read | hp_hfs_mtra_enabled | Enable or disable HFS multithreaded read-ahead. NO MANPAGE. |

| | | |
|---|---|---|
| File System: Read | vxfs_max_ra_kbytes | Maximum amount of read-ahead data, in KB, that the kernel may have outstanding for a single VxFS file system. Manpage: *vxfs_max_ra_kbytes* (5). |
| File System: Read | vxfs_ra_per_disk | Maximum amount of VxFS file system read-ahead per disk, in KB. Manpage: *vxfs_ra_per_disk* (5). |

### SCSI Devices Parameter Summary

Control SCSI device access.

| | | |
|---|---|---|
| File System: SCSI | scsi_maxphys | Maximum record size for the SCSI I/O subsystem, in bytes. Manpage: *scsi_maxphys* (5). |
| File System: SCSI | scsi_max_qdepth | Maximum number of SCSI commands queued up for SCSI devices. Manpage: *scsi_max_qdepth* (5). |
| File System: SCSI | st_ats_enabled | Flag whether to reserve a tape device on open. Manpage: *st_ats_enabled* (5). |
| File System: SCSI | st_fail_overruns | SCSI tape read resulting in data overrun causes failure. Manpage: *st_fail_overruns* (5). |
| File System: SCSI | st_large_recs | Enable large record support for SCSI tape. Manpage: *st_large_recs* (5). |

### Asynchronous Writes Parameter Summary

Control asynchronous writes to file system.

| | | |
|---|---|---|
| File System: Write | fs_async | Enable/disable asynchronous writes of file system data structures to disk. Manpage: *fs_async* (5). |

# File System Buffer Parameters

The system allocates a portion of system memory for use in block-mode file operations (such as `exec()` and `mount()` system calls and inode reading). Buffer space is reserved in increments of 4096-byte pages, but buffers can be much larger than 4096 bytes, requiring as many as 16 or more pages per buffer, depending on hardware device and configuration characteristics.

Two methods for allocating buffer space are supported: static and dynamic. The obsolescent static method allocates buffer space and buffer header structures at system boot time. The preferred dynamic buffer cache method allocates buffer space and supporting data structures as they are needed, using predefined minimum and maximum values to establish overall buffer cache space allocation limits.

## Dynamic Buffer Cache

Most system administrators prefer to specify what percentage or range of percentages of available system memory can be allocated for buffer use, letting the system allocate memory for buffers as needed within the specified limits. Two kernel parameters, `dbc_min_pct` and `dbc_max_pct`, control the lower and upper limit, respectively, as a percentage of system memory. How many pages of memory are allocated for buffer cache use at any given time is determined by system needs, but the two parameters ensure that allocated memory never drops below `dbc_min_pct` and cannot exceed `dbc_max_pct` percent of total system memory. Administrators of multiple systems usually prefer this method because it provides an easy way to choose buffer space limits that are directly related to how much memory is actually installed in each machine, allowing common or similar kernel configurations throughout the network.

**NOTE** To enable dynamic buffer caching, the kernel parameters `nbuf` and `bufpages` must both be set to zero.

## Static Buffer Allocation

For administrators who choose not to use dynamic buffer caching, the two kernel parameters, `nbuf` and `bufpages`, control static buffer allocation. If `bufpages` is nonzero, it specifies the fixed number of 4096-byte pages that are to be allocated for the file system buffer cache. `nbuf` is provided for backward compatibility purposes. If set to a nonzero value, `nbuf` specifies the maximum number of buffer headers that can exist in the buffer header array. However, the preferred approach is to set `nbuf` to zero, in which case, one header is created for each two `bufpages` allocated.

## Recommended Procedure

Set `bufpages` and `nbuf` to zero, and set `dbc_min_pct` and `dbc_max_pct` to the desired upper and lower limits as a percentage of total system memory. This activates dynamic buffer cache allocation.

## Alternate Procedure

To allocate a fixed amount of memory for static buffer cache, set `bufpages` to the desired number of 4 KB pages, and set `nbuf` to zero, which allocates space for `bufpages/2` buffer headers in the header array. `dbc_min_pct` and `dbc_max_pct` are ignored.

## Additional Information

- “File System Buffer Parameter Summary” on page 24

# Open or Locked Files Parameters

## Open and Locked Files

Individual processes can open one or more files for reading or writing, and it is not uncommon for a process to have many files open at the same time; particularly in large database applications, for example.

Furthermore, mail, database, and other applications often require simultaneous access to a given file by two or more processes. To prevent file or data corruption, a process that is altering the contents of a file must be able to lock the file against conflicting uses until it is safe to release control of the file.

Open and locked files require memory and other system resources. These resources must be balanced against other system needs to maintain optimum overall system performance. Use `nflocks` to limit the combined total number of file locks that are available system-wide to all processes at any given time.

## Limits

Two parameters, `maxfiles` and `maxfiles_lim`, respectively, govern the soft and hard limits on the number of files a process can open simultaneously. `nfile` governs the maximum number of files that can be open on the entire system at any given time.

## Additional Information

- “Open or Locked Files Parameter Summary” on page 25

# Logical Volume Manager (LVM) Operation

Logical Volume Manager (LVM) is a subsystem for managing file systems and disk storage space that are structured into logical volumes rather than being restricted to the beginning and end points of a physical disk. Logical volumes can be smaller than the disk or disk array on which they reside, or they can include all or part of several disks or disk arrays. Logical volume boundaries are not required to coincide with the boundaries of physical disks when multiple disks or arrays are used.

Managing logical volumes is done by the Logical Volume Manager, not the kernel. However, the kernel contains data structures for each volume group on the system, and the space reserved for LVM data structures must be sufficient to support the number of volume groups that exist on the system. This is done with the `maxvgs` kernel configuration parameter.

## Additional Information

- “Logical Volume Manager (LVM) Parameter Summary” on page 25

# 5 Interprocess Communication (IPC) Subsystem

# IPC Parameter Summary

See also “How to Specify Configurable Parameter Values” on page 13.

## Message Parameter Summary

The following parameters control allocation of space for messages.

See also “Overview of Message Queue Operations” on page 35.

| | | |
|---|---|---|
| IPC: Message | mesg | Enable or disable IPC messages at system boot time. Manpage: *mesg* (5). |
| IPC: Message | msgmap | Size of free-space resource map for allocating shared memory space for messages. Manpage: *msgmap* (5). |
| IPC: Message | msgmax | System-wide maximum size (in bytes) for individual messages. Manpage: *msgmax* (5). |
| IPC: Message | msgmnb | Maximum combined size (in bytes) of all messages that can be queued simultaneously in a message queue. Manpage: *msgmnb* (5). |
| IPC: Message | msgmni | Maximum number of message queues allowed on the system at any given time. Manpage: *msgmni* (5). |
| IPC: Message | msgseg | Maximum number of message segments that can exist on the system. Manpage: *msgseg* (5). |
| IPC: Message | msgssz | Message segment size in bytes. Manpage: *msgssz* (5). |
| IPC: Message | msgtql | Maximum number of messages that can exist on the system at any given time. Manpage: *msgtql* (5). |

## Semaphore Parameter Summary

The following kernel parameters control System V IPC semaphore operating limits.

See also “Overview of Semaphore Operations” on page 37.

| | | |
|---|---|---|
| IPC: Semaphore | sema | Enable or disable IPC semaphores at system boot time. Manpage: *sema* (5). |
| IPC: Semaphore | semaem | Maximum value by which a semaphore can be changed in a semaphore “undo” operation. Manpage: *semaem* (5). |
| IPC: Semaphore | semmni | Maximum number of sets of IPC semaphores allowed on the system at any one time. Manpage: *semmni* (5). |
| IPC: Semaphore | semmns | Maximum number of individual IPC semaphores available to system users, system-wide. Manpage: *semmns* (5). |
| IPC: Semaphore | semmnu | Maximum number of processes that can have undo operations pending on any given IPC semaphore on the system. Manpage: *semmnu* (5). |

| | | |
|---|---|---|
| IPC: Semaphore | semmsl | Maximum number of individual System V IPC semaphores per semaphore identifier. Manpage: *semmsl* (5). |
| IPC: Semaphore | semume | Maximum number of IPC semaphores that a given process can have undo operations pending on. Manpage: *semume* (5). |
| IPC: Semaphore | semvmx | Maximum value any given IPC semaphore is allowed to reach (prevents undetected overflow conditions). Manpage: *semvmx* (5). |

### Shared Memory Parameter Summary

The following parameters control allocation of space for shared memory.

See also “Overview of Shared Memory Operation” on page 38.

| | | |
|---|---|---|
| IPC: Share | core_addshmem_read | Flag to include readable shared memory in a process core dump. Manpage: *core_addshmem_read* (5). |
| IPC: Share | core_addshmem_write | Flag to include read/write shared memory in a process core dump. Manpage: *core_addshmem_write* (5). |
| IPC: Share | shmem | Enable or disable shared memory at system boot time. Manpage: *shmem* (5). |
| IPC: Share | shmmax | Maximum allowable shared memory segment size (in bytes). Manpage: *shmmax* (5). |
| IPC: Share | shmmni | Maximum number of shared memory segments allowed on the system at any given time. Manpage: *shmmni* (5). |
| IPC: Share | shmseg | Maximum number of shared memory segments that can be attached simultaneously to any given process. Manpage: *shmseg* (5). |

# System V Interprocess Communication Mechanisms

The HP-UX operating system uses shared memory to provide three mechanisms for communicating between cooperating programs and processes:

*Messages* Message data is stored in a given part of shared memory to be retrieved by receiving programs. For more information about how shared memory space for messages is allocated and managed, see “Overview of Message Queue Operations” on page 35.

*Semaphores* Shared storage locations that contain up/down counters for signaling the current status of cooperating processes. For more information about how shared memory space for semaphores is allocated and managed, see “Overview of Semaphore Operations” on page 37.

*Shared Memory* Reserved data storage area shared by two or more processes by means of identically-defined data structures in each cooperating process. For more information about how shared memory space for shared data structures is allocated and managed, see “Overview of Shared Memory Operation” on page 38.

# Overview of Message Queue Operations

See “Message Parameter Summary” on page 32 for the list of parameters.

## Messages

Messages are small collections of data (400 bytes, for example) that can be passed between cooperating programs through a message queue. Messages within a queue can be of different types, and any process with proper permissions can receive the messages. A receiving process can retrieve the first message, the first message of a given type, or the first message of a group of types. See *msgop* (2) for more information.

## Message Queues

Message queues are implemented as linked lists of data stored in shared memory. The message queue itself contains a series of data structures, one for each message, each of which identifies the address, type, and size of the message plus a pointer to the next message in the queue.

To allocate a queue, a program uses the `msgget()` system call (see *msgget* (2)). Messages are placed in the queue by `msgsnd()` system calls and retrieved by `msgrcv()` (see msgop(2)). Other operations related to managing a given message queue are performed by the `msgctl()` system call (see *msgctl* (2)).

For information about using messages in programs, refer to an advanced UNIX programming text such as *Advanced UNIX Programming* by Marc J. Rochkind, Prentice-Hall, Inc., ISBN 0-13-011800-1.

## Message and Message Queue Management

Message queues and the message header array are located in (swappable) shared memory space. Other data structure arrays necessary for managing them are located in the (nonswappable) kernel space.

Kernel configuration parameters for messages control:

- Maximum number of message queues on system,
- Maximum message queue size,
- Maximum message length,
- Maximum total combined length of messages in a queue,
- Maximum number of messages per queue,
- Maximum number of simultaneous messages system-wide,
- Maximum size of message header list,
- Maximum size of free-space map for locating new messages.

IPC messages require the following memory space allocations:

- Space in the kernel area for message identifiers.
- Shared memory space for message queues.
- Shared memory space for message headers.

### Message Queues

*Queue Size* Each message queue is created with enough space for `msgmnb` (message queue number of bytes) bytes of messages. Each message contains one or more message segments of `msgssz` (message segment size) bytes each.

*Message Size* To discourage malicious or poorly written programs from consuming excessive message space, individual messages cannot contain more than `msgmax` (message maximum) bytes per message. Each message is stored in the queue as a series of one or more segments containing `msgssz` bytes per segment. The number of segments used for a message is the smallest integer that, when multiplied by `msgssz`, is greater than the total number of bytes in the message.

*Queue Space* The total space consumed by all messages in any given queue cannot exceed `msgmnb` bytes (message-queue maximum number of bytes. This value must not be less than `msgmax` bytes.

*Total Messages* The maximum number of messages that can exist in a queue at any given time depends on the length of individual messages and the constraints of queue size and other factors listed above. However, a system-wide limit is imposed. The total number of messages that can reside on the system at any one time in all queues cannot exceed `msgtql` (see *Message Header Array* in "System Globals" below).

### System Globals

In addition to the constraints on individual message queues and messages listed in "Message Queues" above, the kernel imposes additional constraints on IPC message management

*Message Queue Identifiers*

Each message queue has an associated message queue identifier stored in the nonswappable kernel area. `msgmni` (message maximum number of identifiers) limits the total number of message queues allowed on the system at any given time.

*Message Header Array*

Each message has an associated message header which is stored in an array in the swappable shared memory area. `msgtql` (message total quantity limit) defines the total number of messages that can be present system-wide at any given time.

*Free Space Management*

As messages are sent and received, space to contain those messages is allocated then released, making the space available for other messages. The kernel maintains a resource map which is used for identifying free space to allocate for new messages. The size of this map is controlled by the `msgmap` parameter.

# Overview of Semaphore Operations

System V IPC semaphores are used mainly to keep processes properly synchronized to prevent collisions when accessing shared data structures. Their use in software tends to be complex, so many programmers use alternate means of control where practical.

See “Semaphore Parameter Summary” on page 32 for the list of parameters.

## Semaphore Operations

The `semget()` system call allocates an array containing a specified number of semaphores (see *semget* (2)). Assigning an array with only one semaphore in the set is usually the most sensible for keeping programs reasonably simple. Subsequent semaphore operations are performed atomically on the entire array; individual semaphores in the array are manipulated by an array of semaphore operations (see *semop* (2)). `semctl()` is used to ascertain or change a semaphore's permissions, change time, or owner (see *semctl* (2)).

Semaphores are typically incremented by a process to block other processes while it is performing a critical operation or using a shared resource. When finished, it decrements the value, allowing blocked processes to then access the resource. Semaphores can be configured as binary semaphores which have only two values: 0 and 1, or they can serve as general semaphores (or counters) where one process increments the semaphore and one or more cooperating processes decrement it. To prevent undetectable overflow conditions, the kernel imposes a maximum value limit beyond which semaphores cannot be incremented. This limit, defined by the `semvmx` kernel parameter, must not exceed the maximum value of 65535. Semaphores are not allowed to have negative (less than zero) values.

## Semaphore Undo Operations

It may occasionally be necessary — when errors occur, a process must abort, a process dies, etc. — to change one or more semaphores to a new or previous value. This is called undoing a semaphore. Since the value of any semaphore when such conditions occur is unpredictable, the system enforces a limit on how much the value of a semaphore can change any undo operation. This limit is defined by the `semaem` kernel parameter.

For more information about System V IPC semaphore operation, consult an advanced UNIX programming text such as *Advanced UNIX Programming* by Marc J. Rochkind, Prentice-Hall, Inc., ISBN 0-13-011800-1.

## Semaphore Limits

Configurable kernel parameters are available to limit the number of sets of semaphores that can exist simultaneously on the system and the total number of individual semaphores available to users that can exist simultaneously on the system. The number of semaphores that can exist in a set is not configurable. It is fixed at the POSIX standard value, 500.

# Overview of Shared Memory Operation

See “Shared Memory Parameter Summary” on page 33 for the list of parameters.

## Shared Memory

Shared memory is reserved memory space for storing data structures and data being shared between or among cooperating processes. Sharing a common memory space eliminates the need for copying or moving data to a separate location before it can be used by other processes, reducing processor time and overhead as well as memory consumption.

## Shared Memory Access and Use

Shared memory management is similar in many respects to messages and semaphores. A process requests a shared memory segment allocation by means of the `shmget()` system call, specifying the segment size in one of the function parameters (see *shmget* (2)). One or more processes can then attach to the allocated segment by using the `shmat()` system call and detach when finished with the `shmdt()` system call (see *shmop* (2)). `shmctl()` is used to obtain information about the segment, and to remove the segment when it is no longer needed (see *shmctl* (2)).

Semaphores can be used to prevent shared memory read/write access collisions, but the more common method is for one process to populate (write to) a given shared memory segment and other processes read from the segment (write once, read many). In such arrangements, when a subsequent write operation is to be performed, the writing process allocates a new segment, and cooperating processes attach to that new segment when ready to use it.

Shared memory is allocated in swappable shared memory space. Data structures for managing shared memory are located in the kernel.

# 6 Kernel Crash Dump Subsystem

## Kernel Crash Dump Parameter Summary

The following parameters affect dump operations when a kernel panic occurs and parts of system memory are dumped to disk.

See also “How to Specify Configurable Parameter Values” on page 13.

| | | |
|---|---|---|
| Kernel Crash Dump | `alwaysdump` | Select which classes of system memory pages are to be dumped if a kernel panic occurs. Manpage: *alwaysdump* (5). |
| Kernel Crash Dump | `dontdump` | Select which classes of system memory pages are not to be dumped if a kernel panic occurs. Manpage: *dontdump* (5). |
| Kernel Crash Dump | `initmodmax` | Maximum size of the dump table of dynamically loaded kernel modules. Manpage: *initmodmax* (5). |
| Kernel Crash Dump | `modstrmax` | Maximum size, in bytes, of the `savecrash` kernel module table that contains module names and their locations in the file system. Manpage: *modstrmax* (5). |

# 7 Memory Paging Subsystem

## Memory Paging Parameter Summary

| | | |
|---|---|---|
| Memory Paging | `allocate_fs_swapmap` | Enable or disable preallocation of file system swap space when `swapon()` is called as opposed to allocating swap space when `malloc()` is called. Enabling allocation reduces risk of insufficient swap space and is used primarily where high availability is important. Manpage: *allocate_fs_swapmap* (5). |
| Memory Paging | `max_mem_window` | Maximum number of group-private 32-bit shared memory windows. Manpage: *max_mem_window* (5). |
| Memory Paging | `nswapdev` | Maximum number of devices, system-wide, that can be used for device swap. Set to match actual system configuration. Manpage: *nswapdev* (5). |
| Memory Paging | `nswapfs` | Maximum number of mounted file systems, system-wide, that can be used for file system swap. Set to match actual system configuration. Manpage: *nswapfs* (5). |
| Memory Paging | `remote_nfs_swap` | Enable or disable swap to mounted remote NFS file system. Used on cluster clients for swapping to NFS-mounted server file systems. Manpage: *remote_nfs_swap* (5). |
| Memory Paging | `swapmem_on` | Enable or disable pseudo-swap allocation. This allows systems with large installed memory to allocate memory space as well as disk swap space for virtual memory use instead of restricting availability to defined disk swap area. Manpage: *swapmem_on* (5). |
| Memory Paging | `swchunk` | Amount of space allocated for each chunk of swap area. Chunks are allocated from device to device by the kernel. Changing this parameter requires extensive knowledge of system internals. *Without such knowledge, do not change this parameter from the normal default value.* Manpage: *swchunk* (5). |
| Memory Paging | `allocate_fs_swapmap` | Enable or disable preallocation of file system swap space when `swapon()` is called as opposed to allocating swap space when `malloc()` is called. Enabling allocation reduces risk of insufficient swap space and is used primarily where high availability is important. See *allocate_fs_swapmap* (5). |

### Variable Page Size Parameter Summary

| | | |
|---|---|---|
| Memory Paging: Size | `vps_ceiling` | Maximum system-selected page size (in KB) if the user does not specify a page size. Manpage: *vps_ceiling* (5). |
| Memory Paging: Size | `vps_chatr_ceiling` | Maximum page size a user can specify with the `chatr` command in a program. Manpage: *vps_chatr_ceiling* (5). |

| | | |
|---|---|---|
| Memory Paging: Size | `vps_pagesize` | Minimum user page size (in KB) if no page size is specified using `chatr`. Manpage: *vps_pagesize* (5). |

### Additional Information


- "Overview of Memory Paging Parameters" on page 44
- "How to Specify Configurable Parameter Values" on page 13

# Overview of Memory Paging Parameters

Configurable kernel parameters for memory paging enforce operating rules and limits related to virtual memory (swap space). They fall into the following categories:

*Total System Swap*

Maximum swap space that can be allocated, system-wide. Parameters include: swchunk.

*Device Swap*

Swap space allocated on hard disk devices. Parameters include: nswapdev.

*File System Swap*

Swap space allocated on mounted file systems. Parameters include: `allocate_fs_swapmap` and nswapfs.

*Pseudo-Swap*

Use of installed RAM as pseudo-swap, allowing virtual memory space allocation beyond the limit of swap space on disk devices. Parameters include: swapmem_on.

*Variable Page Sizes*

The size of virtual memory pages can be altered to make swap operations more efficient in particular applications. Parameters include: vps_ceiling, vps_chatr_ceiling, and vps_pagesize.

## Total System Swap

The swchunk parameter determines the size of each chunk of swap area that is created on successive devices or file systems. Selecting an appropriate value for this parameter requires extensive knowledge of kernel operation and system internals. *Without such knowledge, do not change swchunk to a nondefault value.*

## Device Swap

When devices are connected to the system and configured, part of the disk can be reserved for device swap. The only configurable kernel parameter related to device swap is nswapdev, which specifies how many devices have been configured with space allocated for device swap. Data structure storage space is reserved in the kernel for nswapdev devices.

- If the value of nswapdev is greater than the actual number of swap devices, the small amount of memory space (<50 bytes per device) allocated for data structures to support nonexistent devices is wasted and cannot be used for other purposes.
- If nswapdev is less than the number of available swap devices, only nswapdev devices can be accessed because no data structure storage space is available for supporting the remaining devices.

## File System Swap

In addition to swap space on individual devices, swap space can also be created in existing mounted file systems. This file system swap is somewhat slower than device swap because it must be handled by reading from and writing to open files rather than transferring data directly between the operating system and the disk device. nswapfs is the file system swap counterpart to nswapdev and defines how many locally mounted file systems can be configured system-wide to support file system swap. Like nswapdev, nswapfs should be set to match the actual number of file systems that are normally mounted and intended to be used for file system swap (about nswapfs times 300 bytes of kernel space is required for data structure storage). Only hard disk read/write file systems can be used for file system swap.

A second configurable parameter, allocate_fs_swapmap can be set to enable or disable the allocation of swap space at the time swapon() is called rather than waiting to allocate space using malloc(). Preallocating space ensures the unconditional availability of file system swap space when malloc() is called (otherwise a file-system-full error could occur in some circumstances). Preallocation of file system swap space when swapon() is called is commonly used when high availability is important, but it does prevent other processes from using any resources that are not being use by the process that reserved them.

## Pseudo-Swap

Memory allocation is normally based on the availability of virtual memory (swap space available on disks and file systems). Total available (swappable) memory is the same as total swap space. However, on large systems with massive amounts of installed RAM this can lead to inefficiencies.

Consider a system, for example, that contains 200 MB of RAM, and has 1 GB of swap on the root disk. It is inappropriate to limit such a system running in single-user mode with only the root disk mounted to only 1 GB of memory space when the kernel occupies only 10% or less of the available system RAM. By allowing the use of pseudo-swap, any unused RAM can also be allocated for use by the swap system, allowing larger, more demanding processes to run. On a workstation with 100 or 200 MB of swap on the root device and 16 MB of RAM, the advantage of this capability is much less significant.

swapmem_on is used to enable or disable the allocation of pseudo-swap space in system RAM.

## Additional Information

- “Memory Paging Parameter Summary” on page 42

# 8 Process Management Subsystem

# Process Management Parameter Summary

See also “How to Specify Configurable Parameter Values” on page 13.

## Timesharing Parameter Summary

See also “CPU Timesharing Management” on page 50.

| | | |
|---|---|---|
| ProcessMgmt: CPU | `timeslice` | Maximum time a process can use the CPU until it is made available to the next process having the same process execution priority. This feature also prevents runaway processes from causing system lock-up. Manpage: *timeslice* (5). |

## Memory Space Parameter Summary

See also “Memory Allocation Management” on page 51.

| | | |
|---|---|---|
| ProcessMgmt: Memory | `maxdsiz` | Maximum process data storage segment space that can be used for statics and strings, as well as dynamic data space allocated by `sbrk()` and `malloc()` (32-bit processes). Manpage: *maxdsiz* (5). |
| ProcessMgmt: Memory | `maxdsiz_64bit` | Maximum process data storage segment space that can be used for statics and strings, as well as dynamic data space allocated by `sbrk()` and `malloc()` (64-bit processes). Manpage: *maxdsiz* (5). |
| ProcessMgmt: Memory | `maxrsessiz` | Maximum size (in bytes) of the RSE stack for any user process on the IPF platform. Manpage: *maxrsessiz* (5). |
| ProcessMgmt: Memory | `maxrsessiz_64bit` | Maximum size (in bytes) of the RSE stack for any user process on the IPF platform. Manpage: *maxrsessiz* (5). |
| ProcessMgmt: Memory | `maxssiz` | Maximum dynamic storage segment (DSS) space used for stack space (32-bit processes). Manpage: *maxssiz* (5). |
| ProcessMgmt: Memory | `maxssiz_64bit` | Maximum dynamic storage segment (DSS) space used for stack space (64-bit processes). Manpage: *maxssiz* (5). |
| ProcessMgmt: Memory | `maxtsiz` | Maximum allowable process text segment size, used by unchanging executable-code (32-bit processes). Manpage: *maxtsiz* (5). |
| ProcessMgmt: Memory | `maxtsiz_64bit` | Maximum allowable process text segment size, used by unchanging executable-code (64-bit processes). Manpage: *maxtsiz* (5). |
| ProcessMgmt: Memory | `pa_maxssiz_32bit` | Maximum size (in bytes) of the stack for a user process running under the PA-RISC emulator on IPF. Manpage: *pa_maxssiz* (5). |

| | | |
|---|---|---|
| ProcessMgmt: Memory | pa_maxssiz_64bit | Maximum size (in bytes) of the stack for a user process running under the PA-RISC emulator on IPF. Manpage: *pa_maxssiz* (5). |

### Process Parameter Summary

See also “Process Management” on page 50.

| | | |
|---|---|---|
| ProcessMgmt: Process | executable_stack | Allows or denies program execution on the stack. Manpage: *executable_stack* (5). |
| ProcessMgmt: Process | maxuprc | Maximum number of processes that any single user can have running at the same time, including login shells, user interface processes, running programs and child processes, I/O processes, etc. If a user is using multiple, simultaneous logins under the same login name (user ID) as is common in X Window, CDE, or Motif environments, all processes are combined, even though they may belong to separate process groups. Processes that detach from their parent process group, where that is possible, are not counted after they detach (line printer spooler jobs, certain specialized applications, etc.). Manpage: *maxuprc* (5). |
| ProcessMgmt: Process | nproc | Defines the maximum number of processes that can be running simultaneously on the entire system, including remote execution processes initiated by other systems via remsh or other networking commands. Manpage: *nproc* (5). |
| ProcessMgmt: Process | secure_sid_scripts | Controls whether setuid and setgid bits on scripts are honored. Manpage: *secure_sid_scripts* (5). |

### Kernel Threads Parameter Summary

See also “Kernel Threads” on page 50.

| | | |
|---|---|---|
| ProcessMgmt: Threads | max_thread_proc | Maximum number of threads that any single process can create and have running at the same time. Manpage: *max_thread_proc* (5). |
| ProcessMgmt: Threads | nkthread | Maximum number of kernel threads allowed on the system at the same time. Manpage: *nkthread* (5). |

# Overview of Process Management Parameters

Process management includes:

- Managing the number of processes on the system and processes per user to keep system resources effectively distributed among users for optimal overall system operation.
- Managing allocation of CPU time to competing processes at equal and different priority levels.
- Allocation of virtual memory between processes, protecting the system and competing users against unreasonable demands of abusive or run-away processes.

## Process Management

See also “Process Parameter Summary” on page 49.

Process requirements on individual systems can vary widely. For example, it is not uncommon for a modest workstation running a CDE or Motif environment to have over 100 simultaneous processes supporting a single system user. On the other hand, a large, multiuser system could have 1000 simple ASCII user terminals connected to it, each running only two or three specialized processes. Obviously the process limits imposed on the system by the kernel must be quite different for these two common examples.

Two configurable parameters apply specifically to managing system processes, `nproc` and `maxuprc`.

Select a value for `nproc` that is sufficient to provide enough processes for every user at any given time when the maximum normal number of users are logged in.

Select a value for `maxuprc` that is adequate to meet the normal needs of all system users, but low enough to prevent a run-away program from spawning too many processes (thus preventing or restricting new process availability to other users), and to protect normal users from malicious system abuse by any other user.

## Kernel Threads

See also “Kernel Threads Parameter Summary” on page 49.

On large systems with multiple processors, parts of processes can sometimes be split into threads and run simultaneously on separate processors. Two kernel parameters help manage the consumption of system resources by threaded processes, `max_thread_proc` and `nkthread`.

## CPU Timesharing Management

See also “Timesharing Parameter Summary” on page 48.

The kernel checks frequently for other processes requesting CPU time. Checks for CPU requests from higher-priority processes are made every 10 milliseconds. When two or more processes at the same priority level (such as from two different system users) are competing for CPU time, CPU time is “sliced” into segments, defined by `timeslice`, and passed from process to process in a round-robin fashion, preventing a single process from monopolizing the CPU until it blocks or terminates.

## Memory Allocation Management

See also “Memory Space Parameter Summary” on page 48.

PA-RISC hardware supports the allocation of up to 2 GB of virtual memory to a single process. This memory is divided between the text space where programs are stored, data space (globals, statics, locals to `main()`, strings, etc.), dynamic storage (space allocated by `malloc()`, stack, registers, etc.), and approximately 200 MB reserved for other purposes. Most processes never need that much room, and furthermore, the 2 GB limit exceeds the installed disk space on many systems.

These configurable kernel parameters place protective limits on allocation of process space: `maxtsiz`, `maxtsiz_64bit`, `maxdsiz`, `maxdsiz_64bit`, `maxssiz`, and `maxssiz_64bit`.

Different users on different systems have widely divergent process space needs, making it impractical or frustrating to enforce a fixed limit on text or data storage, for example, for every user on every system. For example, some users on a research or development system may have very modest text segment needs yet require very large data segment space (a small program operating on a very large array, for example), while other users on the same system may need little data space yet have rather large text (program storage) requirements.

On the other hand, a production system supporting hundreds of users may require moderate data storage space per user, yet the aggregate total users on the system may require massive swap space, making it appropriate to use relatively small space limits to protect other users from malfunctioning or run-away iterative processes started by a single user.

### Selecting Limits

Selecting limits that meet the needs of all users on the system while also protecting all users from run-away programs or abuses by others requires a thorough understanding of individual users' program needs. When selecting values, be aware that the limits chosen for `maxtsiz`, `maxdsiz`, and `maxssiz` are safety nets to prevent monopolization of virtual memory resources to the detriment of other users. They are not intended to restrict individual users' access to needed space.

`maxtsiz`, `maxdsiz`, and `maxssiz` are usually set to values whose combined totals can exceed available swap space. This is appropriate because most user program requirements are significantly less than the limit values imposed by these parameters, and selecting values that protect users from occasional malfunctions rather than using smaller values that tend to ration space among users improves overall system performance for everyone on the system.

# 9 Spinlock Pool

## Spinlock Pool Parameter Summary

See also “Spinlock Pool Parameters” on page 55 and “Tunables by Subsystem” on page 10.

| | | |
|---|---|---|
| Spinlock Pool | bufcache_hash_locks | Buffer-cache spinlock pool. NO MANPAGE. |
| Spinlock Pool | chanq_hash_locks | Channel queue spinlock pool. Manpage: *chanq_hash_locks* (5). |
| Spinlock Pool | dnlc_hash_locks | Number of locks for directory cache synchronization. NO MANPAGE. |
| Spinlock Pool | ftable_hash_locks | File table spinlock pool. NO MANPAGE. |
| Spinlock Pool | hdlpreg_hash_locks | Set the size of the pregion spinlock pool. Manpage: *hdlpreg_hash_locks* (5). |
| Spinlock Pool | io_ports_hash_locks | I/O port spinlock pool. NO MANPAGE. |
| Spinlock Pool | pfdat_hash_locks | Pfdat spinlock pool. Manpage: *pfdat_hash_locks* (5). |
| Spinlock Pool | region_hash_locks | Process-region spinlock pool. Manpage: *region_hash_locks* (5). |
| Spinlock Pool | sysv_hash_locks | System V interprocess communication spinlock pool. Manpage: *sysv_hash_locks* (5). |
| Spinlock Pool | vnode_cd_hash_locks | Vnode clean/dirty spinlock pool. NO MANPAGE. |
| Spinlock Pool | vnode_hash_locks | Vnode spinlock pool. NO MANPAGE. |

# Spinlock Pool Parameters

The parameters related to spinlock pools for multiprocessor computers are used similarly and are documented together here. Each parameter allocates the specified number of spinlocks for the corresponding system resource:

These parameters are for use by advanced users only who have a thorough understanding of how spinlocks are used by multiple processors and how the number of spinlocks needed are related to system size and complexity. Do not change these from their default value unless you understand the consequences of any changes. In general, these values should not be altered without the advice of HP support engineers who are thoroughly familiar with their use.

Setting these parameters to inappropriate values can result in severe performance problems in multiprocessor systems.

## Acceptable Values

All of these parameters have the same minimum and maximum values. Only the defaults are different as indicated:

| | |
|---|---|
| Minimum | 64 |
| Maximum | 4096 |
| Default | 64 (`ftable_hash_locks`, `io_ports_hash_locks`) |
| Default | 128 (`bufcache_hash_locks`, `pfdat_hash_locks`, `region_hash_locks`, `sysv_hash_locks`, `vnode_hash_locks`, `vnode_cd_hash_locks`) |
| Default | 256 (`chanq_hash_locks`) |

Specify a value that is an integer exponent of 2. If you specify any other value, SAM or the kernel itself will change the parameter value to the next larger integer exponent of two (for example, specifying 100 results in the value of 128. For more information, see “How to Specify Configurable Parameter Values” on page 13.

## Description

In simple terms, spinlocks are a mechanism used in multiple-processor systems to control the interaction of processors that must be held off while waiting for another processor to finish a task so the results can be passed to the waiting processor. Spinlocks control access to file system vnodes, I/O ports, buffer cache, and various other resources.

Earlier HP-UX versions allocated a fixed number of spinlocks for all resources, but beginning with HP-UX 11.0, spinlocks can be allocated for each resource type to accommodate very large and complex systems.

In general, if the system is encountering lock contention problems that are associated with one of these hashed pools, first identify the resource spinlock pool that is associated with the contention, then increase the spinlock pool parameter for that resource.

As stated above, these parameters are for use by experienced, knowledgeable system administrators only. They should not be altered unless you are quite certain that what you are doing is the correct thing to do.

# 10 Streams Subsystem

## Streams Parameter Summary

The following configurable kernel parameters are used for managing resources used by character-mode I/O streams.

See also:

- “Overview of Streams Kernel Parameters” on page 59
- “How to Specify Configurable Parameter Values” on page 13
- *STREAMS/UX for the HP 9000 Reference Manual*

| | | |
|---|---|---|
| Streams | NSTREVENT | Maximum number of outstanding streams bufcalls that are allowed to exist at any given time on the system. This number should be equal to or greater than the maximum bufcalls that can be generated by the combined total modules pushed onto any given stream, and serves to limit run-away bufcalls. Manpage: *nstrevent* (5). |
| Streams | nstrpty | System-wide maximum number of streams-based pseudo-ttys that are allowed on the system. Manpage: *nstrpty* (5). |
| Streams | NSTRPUSH | Maximum number of streams modules that are allowed to exist in any single stream at any one time on the system. This provides a mechanism for preventing a software defect from attempting to push too many modules onto a stream, but it is not intended as adequate protection against malicious use of streams. Manpage: *nstrpush* (5). |
| Streams | NSTRSCHED | Maximum number of streams scheduler daemons that are allowed to run at any given time on the system. This value is related to the number of processors installed in the system. Manpage: *nstrsched* (5). |
| Streams | STRCTLSZ | Maximum number of control bytes allowed in the control portion of any streams message on the system. Manpage: *strctlsz* (5). |
| Streams | streampipes | Force all pipes to be streams-based. Manpage: *streampipes* (5). |
| Streams | STRMSGSZ | Maximum number of bytes that can be placed in the data portion of any streams message on the system. Manpage: *strmsgsz* (5). |

## Overview of Streams Kernel Parameters

A stream is an I/O pipeline through which serial data passes between the HP-UX operating system and a kernel driver associated with a raw (character-mode) device file and corresponding device such as a terminal or pty. One or more streams modules can be “plugged” or inserted into the stream to perform various functions such as data encryption or compression, character or message translation, attaching information to data packets, adding protocol data to the message, etc.

Each module in the stream performs a specific task on or associated with the data being transmitted through the pipe, and multiple modules can be pushed onto a given stream. The configurable kernel parameters for streams are used to manage the resources allocated for or consumed by character-mode I/O streams and to help prevent inappropriate pushing or other streams behaviors.

For more information about streams and streams programming, refer to the *STREAMS/UX for the HP 9000 Reference Manual*.

Note that streams I/O pipes are not the same as conventional HP-UX and UNIX file system pipelines.

# 11 Miscellaneous Parameters

# Miscellaneous Parameter Summary

The following miscellaneous configurable kernel parameters affect various, usually unrelated, HP-UX subsystems.

See also “How to Specify Configurable Parameter Values” on page 13.

## CD-ROM Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: CD | ncdnode | Maximum number of entries in the vnode table and therefore the maximum number of open CD-ROM file system nodes that can be in memory. Manpage: *ncdnode* (5). |

## System Clock Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Clock | dst | Enable/disable daylight savings time. Manpage: *timezone* (5). |
| Miscellaneous: Clock | timezone | The offset between the local time zone and Coordinated Universal Time (UTC), often called Greenwich Mean Time or GMT. Manpage: *timezone* (5). |

## Disk I/O Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Disk I/O | default_disk_ir | Immediate reporting for disk writes; whether a write() returns immediately after the data is placed in the disk's write buffer or waits until the data is physically stored on the disk media. Manpage: *default_disk_ir* (5). |
| Miscellaneous: Disk I/O | dma32_pool_size | Amount of memory to set aside for 32-bit DMA (bytes). Manpage: *dma32_pool_size* (5). |
| Miscellaneous: Disk I/O | o_sync_is_o_dsync | Specifies whether an open() or fcntl() with the O_SYNC flag set can be converted to the same call with the O_DSYNC flag instead. This controls whether the function can return before updating the file access. NO MANPAGE. |
| Miscellaneous: Disk I/O | physical_io_buffers | Total buffers for physical I/O operations. Manpage: *physical_io_buffers* (5). |

## Intrusion Detection System/9000

| | | |
|---|---|---|
| Miscellaneous: IDS | enable_idds | Flag to enable the IDDS daemon, which gathers data for IDS/9000. Manpage: *enable_idds* (5). |

### Fast Symbolic Link Traversal Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Links | create_fastlinks | Create fast symbolic links using a newer, more efficient format to improve access speed by reducing disk block accesses during path name look-up sequences. Manpage: *create_fastlinks* (5). |

### Reserved System Memory Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Memory | eqmemsize | Number of pages of memory to be reserved for equivalently mapped memory, used mostly for DMA transfers. Manpage: *eqmemsize* (5). |
| Miscellaneous: Memory | nsysmap | Number of entries in the kernel dynamic memory virtual address space resource map (32-bit processes). Manpage: *nsysmap* (5). |
| Miscellaneous: Memory | nsysmap64 | Number of entries in the kernel dynamic memory virtual address space resource map (64-bit processes). Manpage: *nsysmap* (5). |
| Miscellaneous: Memory | unlockable_mem | Amount of system memory to be reserved for system overhead and virtual memory management, that cannot be locked by user processes. Manpage: *unlockable_mem* (5). |

### Network Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Network | tcphashsz | TCP hash table size, in bytes. Manpage: *tcphashsz* (5). |

### Queued Signals Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Queue | ksi_alloc_max | Maximum number of system-wide queued signals that can be allocated. Manpage: *ksi_alloc_max* (5). |
| Miscellaneous: Queue | ksi_send_max | Maximum number of queued signals that a process can send and have pending at one or more receivers. Manpage: *ksi_send_max* (5). |

### Real-Time Priority Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Schedule | rtsched_numpri | Number of distinct real-time interrupt scheduling priority levels are available on the system. Manpage: *rtsched_numpri* (5). |

### Terminal Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Terminal | `nclist` | Maximum number of cblocks available for data transfers through tty and pty devices. Manpage: *nclist* (5). |
| Miscellaneous: Terminal | `npty` | Maximum number of pseudo-tty entries allowed on the system at any one time. Manpage: *npty* (5). |
| Miscellaneous: Terminal | `nstrpty` | System-wide maximum number of streams-based pseudo-ttys that are allowed on the system. Manpage: *nstrpty* (5). |
| Miscellaneous: Terminal | `nstrtel` | Number of telnet session device files that are available on the system. Manpage: *nstrtel* (5). |
| Miscellaneous: Terminal | `scroll_lines` | Defines the number of lines that can be scrolled on the internal terminal emulator (ITE) system console. Manpage: *scroll_lines* (5). |

### Maximum Users Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Users | `maxusers` | Maximum number of users expected to be logged in on the system at one time; used by other system parameters to allocate system resources. Manpage: *maxusers* (5). |

### Web Server Parameter Summary

| | | |
|---|---|---|
| Miscellaneous: Web | `sendfile_max` | The amount of buffer cache that can be used by the `sendfile()` system call on HP-UX web servers. Manpage: *sendfile_max* (5). |

# A Table of Tunable Kernel Parameters

**Table A-1 Tunable Kernel Parameters in Alphabetic Order**

| Category | Tunable | Description |
|---|---|---|
| Accounting | acctresume | Resume accounting when free space on the file system where accounting log files reside rises above acctresume plus minfree percent of total usable file system size. Manpage: *acctsuspend* (5). |
| Accounting | acctsuspend | Suspend accounting when free space on the file system where accounting log files reside drops below acctsuspend plus minfree percent of total usable file system size. Manpage: *acctsuspend* (5). |
| Asynchronous I/O | aio_listio_max | Maximum number of POSIX asynchronous I/O operations allowed in a single lio_listio( ) call. Manpage: *aio_listio_max* (5). |
| Asynchronous I/O | aio_max_ops | System-wide maximum number of POSIX asynchronous I/O operations allowed at one time. Manpage: *aio_max_ops* (5). |
| Asynchronous I/O | aio_physmem_pct | Maximum percentage of total system memory that can be locked for use in POSIX asynchronous I/O operations. Manpage: *aio_physmem_pct* (5). |
| Asynchronous I/O | aio_prio_delta_max | Maximum priority offset (slowdown factor) allowed in a POSIX asynchronous I/O control block (aiocb). Manpage: *aio_prio_delta_max* (5). |
| Memory Paging | allocate_fs_swapmap | Enable or disable preallocation of file system swap space when swapon( ) is called as opposed to allocating swap space when malloc( ) is called. Enabling allocation reduces risk of insufficient swap space and is used primarily where high availability is important. Manpage: *allocate_fs_swapmap* (5). |
| Kernel Crash Dump | alwaysdump | Select which classes of system memory pages are to be dumped if a kernel panic occurs. Manpage: *alwaysdump* (5). |
| Spinlock Pool | bufcache_hash_locks | Buffer-cache spinlock pool. NO MANPAGE. |
| File System: Buffer | bufpages | Number of 4 KB pages in file system static buffer cache. Manpage: *bufpages* (5). |
| Spinlock Pool | chanq_hash_locks | Channel queue spinlock pool. Manpage: *chanq_hash_locks* (5). |
| IPC: Share | core_addshmem_read | Flag to include readable shared memory in a process core dump. Manpage: *core_addshmem_read* (5). |
| IPC: Share | core_addshmem_write | Flag to include read/write shared memory in a process core dump. Manpage: *core_addshmem_write* (5). |

**Table A-1 Tunable Kernel Parameters in Alphabetic Order (Continued)**

| Category | Tunable | Description |
|---|---|---|
| Miscellaneous: Links | `create_fastlinks` | Create fast symbolic links using a newer, more efficient format to improve access speed by reducing disk block accesses during path name look-up sequences. Manpage: *create_fastlinks* (5). |
| File System: Buffer | `dbc_max_pct` | Maximum percentage of memory for dynamic buffer cache. Manpage: *dbc_max_pct* (5). |
| File System: Buffer | `dbc_min_pct` | Minimum percentage of memory for dynamic buffer cache. Manpage: *dbc_min_pct* (5). |
| Miscellaneous: Disk I/O | `default_disk_ir` | Immediate reporting for disk writes; whether a `write()` returns immediately after the data is placed in the disk's write buffer or waits until the data is physically stored on the disk media. Manpage: *default_disk_ir* (5). |
| File System: Buffer | `disksort_seconds` | Maximum wait time for disk requests. NO MANPAGE. |
| Miscellaneous: Disk I/O | `dma32_pool_size` | Amount of memory to set aside for 32-bit DMA (bytes). Manpage: *dma32_pool_size* (5). |
| Spinlock Pool | `dnlc_hash_locks` | Number of locks for directory cache synchronization. NO MANPAGE. |
| Kernel Crash Dump | `dontdump` | Select which classes of system memory pages are not to be dumped if a kernel panic occurs. Manpage: *dontdump* (5). |
| Miscellaneous: Clock | `dst` | Enable/disable daylight savings time. Manpage: *timezone* (5). |
| Miscellaneous: IDS | `enable_idds` | Flag to enable the IDDS daemon, which gathers data for IDS/9000. Manpage: *enable_idds* (5). |
| Miscellaneous: Memory | `eqmemsize` | Number of pages of memory to be reserved for equivalently mapped memory, used mostly for DMA transfers. Manpage: *eqmemsize* (5). |
| ProcessMgmt: Process | `executable_stack` | Allows or denies program execution on the stack. Manpage: *executable_stack* (5). |
| File System: Write | `fs_async` | Enable/disable asynchronous writes of file system data structures to disk. Manpage: *fs_async* (5). |
| Spinlock Pool | `ftable_hash_locks` | File table spinlock pool. NO MANPAGE. |
| Spinlock Pool | `hdlpreg_hash_locks` | Set the size of the pregion spinlock pool. Manpage: *hdlpreg_hash_locks* (5). |
| File System: Read | `hfs_max_ra_blocks` | The maximum number of read-ahead blocks that the kernel may have outstanding for a single HFS file system. Manpage: *hfs_max_ra_blocks* (5). |

**Table A-1 Tunable Kernel Parameters in Alphabetic Order (Continued)**

| Category | Tunable | Description |
|---|---|---|
| File System: Read | hfs_max_revra_blocks | The maximum number of reverse read-ahead blocks that the kernel may have outstanding for a single HFS file system. Manpage: *hfs_max_revra_blocks* (5). |
| File System: Read | hfs_ra_per_disk | The amount of HFS file system read-ahead per disk drive, in KB. Manpage: *hfs_ra_per_disk* (5). |
| File System: Read | hfs_revra_per_disk | The amount of memory (in KB) for HFS reverse read-ahead operations, per disk drive. Manpage: *hfs_revra_per_disk* (5). |
| File System: Read | hp_hfs_mtra_enabled | Enable or disable HFS multithreaded read-ahead. NO MANPAGE. |
| Kernel Crash Dump | initmodmax | Maximum size of the dump table of dynamically loaded kernel modules. Manpage: *initmodmax* (5). |
| Spinlock Pool | io_ports_hash_locks | I/O port spinlock pool. NO MANPAGE. |
| Miscellaneous: Queue | ksi_alloc_max | Maximum number of system-wide queued signals that can be allocated. Manpage: *ksi_alloc_max* (5). |
| Miscellaneous: Queue | ksi_send_max | Maximum number of queued signals that a process can send and have pending at one or more receivers. Manpage: *ksi_send_max* (5). |
| ProcessMgmt: Memory | maxdsiz | Maximum process data storage segment space that can be used for statics and strings, as well as dynamic data space allocated by `sbrk()` and `malloc()` (32-bit processes). Manpage: *maxdsiz* (5). |
| ProcessMgmt: Memory | maxdsiz_64bit | Maximum process data storage segment space that can be used for statics and strings, as well as dynamic data space allocated by `sbrk()` and `malloc()` (64-bit processes). Manpage: *maxdsiz* (5). |
| File System: Open/Lock | maxfiles | Soft limit on how many files a single process can have opened or locked at any given time. Manpage: *maxfiles* (5). |
| File System: Open/Lock | maxfiles_lim | Hard limit on how many files a single process can have opened or locked at any given time. Manpage: *maxfiles_lim* (5). |
| ProcessMgmt: Memory | maxrsessiz | Maximum size (in bytes) of the RSE stack for any user process on the IPF platform. Manpage: *maxrsessiz* (5). |
| ProcessMgmt: Memory | maxrsessiz_64bit | Maximum size (in bytes) of the RSE stack for any user process on the IPF platform. Manpage: *maxrsessiz* (5). |
| ProcessMgmt: Memory | maxssiz | Maximum dynamic storage segment (DSS) space used for stack space (32-bit processes). Manpage: *maxssiz* (5). |
| ProcessMgmt: Memory | maxssiz_64bit | Maximum dynamic storage segment (DSS) space used for stack space (64-bit processes). Manpage: *maxssiz* (5). |

**Table A-1 Tunable Kernel Parameters in Alphabetic Order (Continued)**

| Category | Tunable | Description |
|---|---|---|
| ProcessMgmt: Memory | maxtsiz | Maximum allowable process text segment size, used by unchanging executable-code (32-bit processes). Manpage: *maxtsiz* (5). |
| ProcessMgmt: Memory | maxtsiz_64bit | Maximum allowable process text segment size, used by unchanging executable-code (64-bit processes). Manpage: *maxtsiz* (5). |
| ProcessMgmt: Process | maxuprc | Maximum number of processes that any single user can have running at the same time, including login shells, user interface processes, running programs and child processes, I/O processes, etc. If a user is using multiple, simultaneous logins under the same login name (user ID) as is common in X Window, CDE, or Motif environments, all processes are combined, even though they may belong to separate process groups. Processes that detach from their parent process group, where that is possible, are not counted after they detach (line printer spooler jobs, certain specialized applications, etc.). Manpage: *maxuprc* (5). |
| Miscellaneous: Users | maxusers | Maximum number of users expected to be logged in on the system at one time; used by other system parameters to allocate system resources. Manpage: *maxusers* (5). |
| File System: LVM | maxvgs | Maximum number of volume groups configured by the Logical Volume Manager on the system. Manpage: *maxvgs* (5). |
| Accounting | max_acct_file_size | Maximum size of the accounting file. Manpage: *max_acct_file_size* (5). |
| Asynchronous I/O | max_async_ports | System-wide maximum number of ports to the asynchronous disk I/O driver that processes can have open at any given time. Manpage: *max_async_ports* (5). |
| Memory Paging | max_mem_window | Maximum number of group-private 32-bit shared memory windows. Manpage: *max_mem_window* (5). |
| ProcessMgmt: Threads | max_thread_proc | Maximum number of threads that any single process can create and have running at the same time. Manpage: *max_thread_proc* (5). |
| IPC: Message | mesg | Enable or disable IPC messages at system boot time. Manpage: *mesg* (5). |
| Kernel Crash Dump | modstrmax | Maximum size, in bytes, of the savecrash kernel module table that contains module names and their locations in the file system. Manpage: *modstrmax* (5). |
| IPC: Message | msgmap | Size of free-space resource map for allocating shared memory space for messages. Manpage: *msgmap* (5). |

**Table A-1 Tunable Kernel Parameters in Alphabetic Order (Continued)**

| Category | Tunable | Description |
|---|---|---|
| IPC: Message | msgmax | System-wide maximum size (in bytes) for individual messages. Manpage: *msgmax* (5). |
| IPC: Message | msgmnb | Maximum combined size (in bytes) of all messages that can be queued simultaneously in a message queue. Manpage: *msgmnb* (5). |
| IPC: Message | msgmni | Maximum number of message queues allowed on the system at any given time. Manpage: *msgmni* (5). |
| IPC: Message | msgseg | Maximum number of message segments that can exist on the system. Manpage: *msgseg* (5). |
| IPC: Message | msgssz | Message segment size in bytes. Manpage: *msgssz* (5). |
| IPC: Message | msgtql | Maximum number of messages that can exist on the system at any given time. Manpage: *msgtql* (5). |
| File System: Buffer | nbuf | System-wide number of static file system buffer and cache buffer headers. Manpage: *nbuf* (5). |
| Miscellaneous: CD | ncdnode | Maximum number of entries in the vnode table and therefore the maximum number of open CD-ROM file system nodes that can be in memory. Manpage: *ncdnode* (5). |
| Miscellaneous: Terminal | nclist | Maximum number of cblocks available for data transfers through tty and pty devices. Manpage: *nclist* (5). |
| File System: Open/Lock | ncsize | Inode space needed for directory name lookup cache (DNLC). NO MANPAGE. |
| File System: Open/Lock | nfile | Maximum number of files that can be open simultaneously on the system at any given time. Manpage: *nfile* (5). |
| File System: Open/Lock | nflocks | Maximum combined number of file locks that are available system-wide to all processes at one time. Manpage: *nflocks* (5). |
| File System: Open/Lock | ninode | Maximum number of open inodes that can be in memory. Manpage: *ninode* (5). |
| ProcessMgmt: Threads | nkthread | Maximum number of kernel threads allowed on the system at the same time. Manpage: *nkthread* (5). |
| ProcessMgmt: Process | nproc | Defines the maximum number of processes that can be running simultaneously on the entire system, including remote execution processes initiated by other systems via remsh or other networking commands. Manpage: *nproc* (5). |
| Miscellaneous: Terminal | npty | Maximum number of pseudo-tty entries allowed on the system at any one time. Manpage: *npty* (5). |

**Table A-1 Tunable Kernel Parameters in Alphabetic Order (Continued)**

| Category | Tunable | Description |
|---|---|---|
| Streams | NSTREVENT | Maximum number of outstanding streams bufcalls that are allowed to exist at any given time on the system. This number should be equal to or greater than the maximum bufcalls that can be generated by the combined total modules pushed onto any given stream, and serves to limit run-away bufcalls. Manpage: *nstrevent* (5). |
| Miscellaneous: Terminal | nstrpty | System-wide maximum number of streams-based pseudo-ttys that are allowed on the system. Manpage: *nstrpty* (5). |
| Streams | nstrpty | System-wide maximum number of streams-based pseudo-ttys that are allowed on the system. Manpage: *nstrpty* (5). |
| Streams | NSTRPUSH | Maximum number of streams modules that are allowed to exist in any single stream at any one time on the system. This provides a mechanism for preventing a software defect from attempting to push too many modules onto a stream, but it is not intended as adequate protection against malicious use of streams. Manpage: *nstrpush* (5). |
| Streams | NSTRSCHED | Maximum number of streams scheduler daemons that are allowed to run at any given time on the system. This value is related to the number of processors installed in the system. Manpage: *nstrsched* (5). |
| Miscellaneous: Terminal | nstrtel | Number of telnet session device files that are available on the system. Manpage: *nstrtel* (5). |
| Memory Paging | nswapdev | Maximum number of devices, system-wide, that can be used for device swap. Set to match actual system configuration. Manpage: *nswapdev* (5). |
| Memory Paging | nswapfs | Maximum number of mounted file systems, system-wide, that can be used for file system swap. Set to match actual system configuration. Manpage: *nswapfs* (5). |
| Miscellaneous: Memory | nsysmap | Number of entries in the kernel dynamic memory virtual address space resource map (32-bit processes). Manpage: *nsysmap* (5). |
| Miscellaneous: Memory | nsysmap64 | Number of entries in the kernel dynamic memory virtual address space resource map (64-bit processes). Manpage: *nsysmap* (5). |
| Miscellaneous: Disk I/O | o_sync_is_o_dsync | Specifies whether an open( ) or fcntl( ) with the O_SYNC flag set can be converted to the same call with the O_DSYNC flag instead. This controls whether the function can return before updating the file access. NO MANPAGE. |

**Table A-1 Tunable Kernel Parameters in Alphabetic Order (Continued)**

| Category | Tunable | Description |
|---|---|---|
| ProcessMgmt: Memory | pa_maxssiz_32bit | Maximum size (in bytes) of the stack for a user process running under the PA-RISC emulator on IPF. Manpage: *pa_maxssiz* (5). |
| ProcessMgmt: Memory | pa_maxssiz_64bit | Maximum size (in bytes) of the stack for a user process running under the PA-RISC emulator on IPF. Manpage: *pa_maxssiz* (5). |
| Spinlock Pool | pfdat_hash_locks | Pfdat spinlock pool. Manpage: *pfdat_hash_locks* (5). |
| Miscellaneous: Disk I/O | physical_io_buffers | Total buffers for physical I/O operations. Manpage: *physical_io_buffers* (5). |
| Spinlock Pool | region_hash_locks | Process-region spinlock pool. Manpage: *region_hash_locks* (5). |
| Memory Paging | remote_nfs_swap | Enable or disable swap to mounted remote NFS file system. Used on cluster clients for swapping to NFS-mounted server file systems. Manpage: *remote_nfs_swap* (5). |
| Miscellaneous: Schedule | rtsched_numpri | Number of distinct real-time interrupt scheduling priority levels are available on the system. Manpage: *rtsched_numpri* (5). |
| Miscellaneous: Terminal | scroll_lines | Defines the number of lines that can be scrolled on the internal terminal emulator (ITE) system console. Manpage: *scroll_lines* (5). |
| File System: SCSI | scsi_maxphys | Maximum record size for the SCSI I/O subsystem, in bytes. Manpage: *scsi_maxphys* (5). |
| File System: SCSI | scsi_max_qdepth | Maximum number of SCSI commands queued up for SCSI devices. Manpage: *scsi_max_qdepth* (5). |
| ProcessMgmt: Process | secure_sid_scripts | Controls whether setuid and setgid bits on scripts are honored. Manpage: *secure_sid_scripts* (5). |
| IPC: Semaphore | sema | Enable or disable IPC semaphores at system boot time. Manpage: *sema* (5). |
| IPC: Semaphore | semaem | Maximum value by which a semaphore can be changed in a semaphore “undo” operation. Manpage: *semaem* (5). |
| IPC: Semaphore | semmni | Maximum number of sets of IPC semaphores allowed on the system at any one time. Manpage: *semmni* (5). |
| IPC: Semaphore | semmns | Maximum number of individual IPC semaphores available to system users, system-wide. Manpage: *semmns* (5). |
| IPC: Semaphore | semmnu | Maximum number of processes that can have undo operations pending on any given IPC semaphore on the system. Manpage: *semmnu* (5). |

**Table A-1 Tunable Kernel Parameters in Alphabetic Order (Continued)**

| Category | Tunable | Description |
|---|---|---|
| IPC: Semaphore | semmsl | Maximum number of individual System V IPC semaphores per semaphore identifier. Manpage: *semmsl* (5). |
| IPC: Semaphore | semume | Maximum number of IPC semaphores that a given process can have undo operations pending on. Manpage: *semume* (5). |
| IPC: Semaphore | semvmx | Maximum value any given IPC semaphore is allowed to reach (prevents undetected overflow conditions). Manpage: *semvmx* (5). |
| Miscellaneous: Web | sendfile_max | The amount of buffer cache that can be used by the sendfile() system call on HP-UX web servers. Manpage: *sendfile_max* (5). |
| IPC: Share | shmem | Enable or disable shared memory at system boot time. Manpage: *shmem* (5). |
| IPC: Share | shmmax | Maximum allowable shared memory segment size (in bytes). Manpage: *shmmax* (5). |
| IPC: Share | shmmni | Maximum number of shared memory segments allowed on the system at any given time. Manpage: *shmmni* (5). |
| IPC: Share | shmseg | Maximum number of shared memory segments that can be attached simultaneously to any given process. Manpage: *shmseg* (5). |
| Streams | STRCTLSZ | Maximum number of control bytes allowed in the control portion of any streams message on the system. Manpage: *strctlsz* (5). |
| Streams | streampipes | Force all pipes to be streams-based. Manpage: *streampipes* (5). |
| Streams | STRMSGSZ | Maximum number of bytes that can be placed in the data portion of any streams message on the system. Manpage: *strmsgsz* (5). |
| File System: SCSI | st_ats_enabled | Flag whether to reserve a tape device on open. Manpage: *st_ats_enabled* (5). |
| File System: SCSI | st_fail_overruns | SCSI tape read resulting in data overrun causes failure. Manpage: *st_fail_overruns* (5). |
| File System: SCSI | st_large_recs | Enable large record support for SCSI tape. Manpage: *st_large_recs* (5). |

**Table A-1 Tunable Kernel Parameters in Alphabetic Order (Continued)**

| Category | Tunable | Description |
|---|---|---|
| Memory Paging | swapmem_on | Enable or disable pseudo-swap allocation. This allows systems with large installed memory to allocate memory space as well as disk swap space for virtual memory use instead of restricting availability to defined disk swap area. Manpage: *swapmem_on* (5). |
| Memory Paging | swchunk | Amount of space allocated for each chunk of swap area. Chunks are allocated from device to device by the kernel. Changing this parameter requires extensive knowledge of system internals. *Without such knowledge, do not change this parameter from the normal default value.* Manpage: *swchunk* (5). |
| Spinlock Pool | sysv_hash_locks | System V interprocess communication spinlock pool. Manpage: *sysv_hash_locks* (5). |
| Miscellaneous: Network | tcphashsz | TCP hash table size, in bytes. Manpage: *tcphashsz* (5). |
| ProcessMgmt: CPU | timeslice | Maximum time a process can use the CPU until it is made available to the next process having the same process execution priority. This feature also prevents runaway processes from causing system lock-up. Manpage: *timeslice* (5). |
| Miscellaneous: Clock | timezone | The offset between the local time zone and Coordinated Universal Time (UTC), often called Greenwich Mean Time or GMT. Manpage: *timezone* (5). |
| Miscellaneous: Memory | unlockable_mem | Amount of system memory to be reserved for system overhead and virtual memory management, that cannot be locked by user processes. Manpage: *unlockable_mem* (5). |
| Spinlock Pool | vnode_cd_hash_locks | Vnode clean/dirty spinlock pool. NO MANPAGE. |
| Spinlock Pool | vnode_hash_locks | Vnode spinlock pool. NO MANPAGE. |
| Memory Paging: Size | vps_ceiling | Maximum system-selected page size (in KB) if the user does not specify a page size. Manpage: *vps_ceiling* (5). |
| Memory Paging: Size | vps_chatr_ceiling | Maximum page size a user can specify with the chatr command in a program. Manpage: *vps_chatr_ceiling* (5). |
| Memory Paging: Size | vps_pagesize | Minimum user page size (in KB) if no page size is specified using chatr. Manpage: *vps_pagesize* (5). |
| File System: Journaled | vxfs_max_ra_kbytes | Maximum amount of read-ahead data, in KB, that the kernel may have outstanding for a single VxFS file system. Manpage: *vxfs_max_ra_kbytes* (5). |

**Table A-1 Tunable Kernel Parameters in Alphabetic Order (Continued)**

| Category | Tunable | Description |
|---|---|---|
| File System: Read | vxfs_max_ra_kbytes | Maximum amount of read-ahead data, in KB, that the kernel may have outstanding for a single VxFS file system. Manpage: *vxfs_max_ra_kbytes* (5). |
| File System: Journaled | vxfs_ra_per_disk | Maximum amount of VxFS file system read-ahead per disk, in KB. Manpage: *vxfs_ra_per_disk* (5). |
| File System: Read | vxfs_ra_per_disk | Maximum amount of VxFS file system read-ahead per disk, in KB. Manpage: *vxfs_ra_per_disk* (5). |
| File System: Journaled | vx_fancyra_enable | Enable or disable VxFS file system read-ahead. NO MANPAGE. |
| File System: Journaled | vx_maxlink | Number of subdirectories created within a directory. NO MANPAGE. |
| File System: Journaled | vx_ncsize | Memory space reserved for VxFS directory path name cache. Manpage: *vx_ncsize* (5). |
| File System: Journaled | vx_ninode | Number of entries in the VxFS inode table. NO MANPAGE. |