Appendix A

IRIX Kernel Tunable Parameters

This appendix describes the tunable parameters that define kernel structures. These structures keep track of processes, files, and system activity. Many of the parameter values are specified in the files found in */var/sysgen/mtune* and */var/sysgen/master.d*.

If the system does not respond favorably to your tuning changes, you may want to return to your original configuration or continue making changes to related parameters. An unfavorable response to your tuning changes can be as minor as the system not gaining the hoped-for speed or capacity, or as major as the system becoming unable to boot or run. This generally occurs when parameters are changed to a great degree. Simply maximizing a particular parameter without regard for related parameters can upset the balance of the system to the point of inoperability. For complete information on the proper procedure for tuning your operating system, read Chapter 11, "System Performance Tuning."

Format of This Appendix

The rest of this appendix describes the more important tunable parameters according to function. Related parameters are grouped into sections. These sections include:

- General tunable parameters. See "General Parameters" on page 214.
- System limits tunable parameters. See "System Limits Parameters" on page 222.
- Resource limits tunable parameters. See "Resource Limits Parameters" on page 224.
- Paging tunable parameters. See "Paging Parameters" on page 232.
- IPC tunable parameters–including interprocess communication messages, semaphores, and shared memory. See "IPC Parameters" on page 238, "IPC Messages Parameters" on page 240, "IPC Semaphores Parameters" on page 243, and "IPC Shared Memory Parameters" on page 247.
- Streams tunable parameters. See "Streams Parameters" on page 249.
- Signal parameters. See "Signal Parameters" on page 251.

- Dispatch parameters. See "Dispatch Parameters" on page 252.
- Extent File System (EFS) parameters. See "EFS Parameters" on page 255.
- Loadable driver parameters. See "Loadable Drivers Parameters" on page 256.
- CPU actions parameters. See "CPU Actions Parameters" on page 258.
- Switch parameters. See "Switch Parameters" on page 259.
- Timer parameters. See "Timer parameters" on page 266.
- Network File System (NFS) parameters. See "NFS Parameters" on page 268.
- Socket parameters. See "Socket Parameters" on page 271.
- Indy Video parameters. See "VINO Parameters" on page 277.
- Extended Accounting parameters. See "Extended Accounting Parameters" on page 278.

Each section begins with a short description of the activities controlled by the parameters in that section, and each listed parameter has a description that may include:

- Name – the name of the parameter.
- Description – a description of the parameter, including the file in which the parameter is specified, and the formula, if applicable.
- Value – the default setting and, if applicable, a range. Note that the value given for each parameter is usually appropriate for a single-user graphics workstation. Values may be presented in either hex or decimal notation.
- When to Change – the conditions under which it is appropriate to change the parameter.
- Notes – other pertinent information, such as error messages.

Note that the tunable parameters are subject to change with each release of the system.

General Parameters

The following group of tunable parameters specifies the size of various system structures. These are the parameters you will most likely change when you tune a system.

- *cachefs_readahead* – specifies the number of readahead blocks for the filesystem.

- *cachefs_max_threads* – specifies the maximum number of ansynchronous I/O daemons per cachefs mount.
- *nbuf* – specifies the number of buffer headers in the file system buffer cache.
- *callout_himark* – specifies the high water mark for callouts
- *ncallout* – specifies the initial number of callouts
- *reserve_ncallout* – specifies the number of reserved callouts
- *ncsize* – specifies the name cache size.
- *ndquot* – used by the disk quota system.
- *nproc* – specifies the number of user processes allowed at any given time.
- *maxpmem* – specifies the maximum physical memory address.
- *syssegsz* – specifies the maximum number of pages of dynamic system memory.
- *maxdmasz* – specifies the maximum DMA transfer in pages.
- *nm_clusters* – specifies the maximum number of one-page clusters used in network buffers.
- *ecc_recover_enable* – specifies the system response to multibit errors.
- *vnode_free_ratio* – specifies the ratio of free vnodes to in-use vnodes.

cachefs_readahead

Description	This parameter specifies the number of blocks to read ahead of the current read block in the filesystem. The blocks are read asynchronously.
Value	Default: 1 (0x1)
	Range: 0 - 10

cachefs_max_threads

Description	This parameter specifies the maximum number of asynchronous I/O daemons that can be run per cachefs mount.
Value	Default: 5 (0x5)
	Range: 1 - 10

nbuf

Description The *nbuf* parameter specifies the number of buffer headers in the file system buffer cache. The actual memory associated with each buffer header is dynamically allocated as needed and can be of varying size, currently 1 to 128 blocks (512 to 64KB).

The system uses the file system buffer cache to optimize file system I/O requests. The buffer memory caches blocks from the disk, and the blocks that are used frequently stay in the cache. This helps avoid excess disk activity.

Buffers are used only as transaction headers. When the input or output operation has finished, the buffer is detached from the memory it mapped and the buffer header becomes available for other uses. Because of this, a small number of buffer headers is sufficient for most systems. If *nbuf* is set to 0, the system automatically configures *nbuf* for average systems. There is little overhead in making it larger for non-average systems.

The *nbuf* parameter is defined in */var/sysgen/mtune*.

Value Default: 0 (Automatically configured if set to 0)

Formula: 100 + (*total number of pages of memory*/40)

32-bit Range: up to 6000

64-bit Range: up to 250000

When To Change

The automatic configuration is adequate for average systems. If you see dropping ''cache hit'' rates in *sar* and *osview* output, increase this parameter. Also, if you have directories with a great number of files (over 1000), you may wish to raise this parameter.

callout_himark

Description The *callout_himark* parameter specifies the maximum number of callout table entries system-wide. The callout table is used by device drivers to provide a timeout to make sure that the system does not hang when a device does not respond to commands.

This parameter is defined in */var/sysgen/mtune* and has the following formula:

nproc + 32

where:

nproc is the maximum number of processes, system-wide.

Value
Default: 0 (Automatically configured if set to 0)

Formula: *nproc* + 32

Range: 42 - 1100

When to Change
Increase this parameter if you see console error messages indicating that no more callouts are available.

ncallout

Description
The *ncallout* parameter specifies the number of callout table entries at boot time. The system will automatically allocate one new callout entry if it runs out of entries in the callout table. However, the maximum number of entries in the callout table is defined by the *callout_himark* parameter.

Value
Default: 40

Range: 20–1000

When to Change
Change *ncallout* if you are running an unusually high number of device drivers on your system. Note that the system automatically allocates additional entries when needed, and releases them when they are no longer needed.

reserve_ncallout

Description
The *reserve_ncallout* parameter specifies the number of reserved callout table entries. These reserved table entries exist for kernel interrupt routines when the system has run out of the normal callout table entries and cannot allocate additional entries.

Value — Default: 5

Range: 0-30

ncsize

Description — This parameter controls the size of the name cache. The name cache is used to allow the system to bypass reading directory names out of the main buffer cache. A name cache entry contains a directory name, a pointer to the directory's in-core inode and version numbers, and a similar pointer to the directory's parent directory in-core inode and version number.

Value — Default: 0 (Automatically configured if set to 0)

Range: 268-6200

ndquot

Description — The *ndquot* parameter controls disk quotas.

Value — Default: 0 (Automatically configured if set to 0)

Range: 268-6200

nproc

Description — The *nproc* parameter specifies the number of entries in the system process (*proc*) table. Each running process requires an in-core *proc* structure. Thus *nproc* is the maximum number of processes that can exist in the system at any given time.

The default value of *nproc* is based on the amount of memory on your system. To find the currently auto-configured value of *nproc*, use the *systune*(1M) command.

The *nproc* parameter is defined in */var/sysgen/mtune*.

Value — Default: 0 (Automatically configured if set to 0)

Range: 30–10000

When to Change

Increase this parameter if you see an overflow in the *sar -v* output for the *proc -sz ov* column or you receive the operating system message:

```
no more processes
```

This means that the total number of processes in the system has reached the current setting. If processes are prevented from forking (being created), increase this parameter. A related parameter is *maxup*.

Notes

If a process can't fork, make sure that this is system-wide, and not just a user ID problem (see the *maxup* parameter).

If *nproc* is too small, processes that try to fork receive the operating system error:

```
EAGAIN: No more processes
```

The shell also returns a message:

```
fork failed: too many processes
```

If a system daemon such as *sched*, *vhand*, *init*, or *bdflush* can't allocate a process table entry, the system halts and displays:

```
No process slots
```

maxpmem

Description

The *maxpmem* parameter specifies the amount of physical memory (in pages) that the system can recognize. The page size is 4 Kilobytes for 32 bit IRIX version 5 or 6 kernels and 16 Kilobytes for 64 bit IRIX version 6 kernels. If set to zero (0), the system will use all available pages in memory. A value other than zero defines the physical memory size (in pages) that the system will recognize.

This parameter is defined in */var/sysgen/mtune*.

Value

Default: 0 (Automatically configured if set to 0)

Range: 1024 pages – total amount of memory

When to Change

You don't need to change this parameter, except when benchmarks require a specific system memory size less than the physical memory size of the system. This is primarily useful to kernel developers. You can also boot the system with the command:

```
maxpmem = memory_size
```

added on the boot command line to achieve the same effect.

syssegsz

Description

This is the maximum number of pages of dynamic system memory.

Value

Default: 0 (Autoconfigured if set to 0)

32-bit Range: 0x2000 - 0x20000

64-bit Range: 0x2000 - 0x10000000

When to Change

Increase this parameter correspondingly when *maxdmasz* is increased, or when you install a kernel driver that performs a lot of dynamic memory allocation.

maxdmasz

Description

The maximum DMA transfer expressed in pages of memory. This amount must be less than the value of *syssegsz* and *maxpmem*.

Value

Default: 1024 pages

32-bit Range: 1 - *syssegsz* (maximum 0x20000)

64-bit Range: 1 - *syssegsz* (maximum 0x10000000)

When to Change

Change this parameter when you need to be able to use very large read or write system calls, or other system calls that perform large scale DMA. This situation typically arises when using optical scanners, film recorders or some printers.

nm_clusters

Description — The maximum number of single page clusters that can be allocated to network buffers. This limits the total memory that network buffers (mbufs) can consume.

Value — Default: 1/8 of physical memory

Range: Default - *Total System Memory (in pages)*

When to Change

For most workstations and small servers the default is sufficient. For very large systems larger values are appropriate. A 0 value tells the kernel to set the value (1/8 of physical memory) based on the amount of physical memory configured in the hardware.

ecc_recover_enable

Description — This parameter, when set to 0, directs the system not to attempt to recover from multibit errors. If set greater than 0, the parameter value is taken as a number of seconds. The system is directed to attempt recovery, but not in the event of more than 32 errors in each number of seconds specified by this parameter.

Value — Default: 60 seconds (0x3c)

vnode_free_ratio

Description — The ratio of free vnodes to vnodes in use that is to be maintained by the kernel.

Value — Default: 2:1

Range: 1:1 to 16:1, or 0 to auto-configure

When to Change

For most workstations and servers the default, auto-configured value is sufficient. Increase this parameter if you have an application that opens and closes many files frequently. When a program is opening and closing and reopening many files, the system may overflow the file buffer cache, even though the file contents are still in memory. Then the files will be read in again, wasting disk activity and time. Increasing this parameter increases the number of files that can be "remembered" by IRIX.

System Limits Parameters

IRIX has configurable parameters for certain system limits. For example, you can set maximum values for each process (its core or file size), the number of groups per user, the number of resident pages, and so forth. These parameters are listed below. All parameters are set and defined in */var/sysgen/mtune*.

- *maxup* – the number of processes per user
- *ngroups_max* – the number of groups to which a user may belong
- *maxwatchpoints* – the maximum number of "watchpoints" per process.
- *nprofile* – amount of disjoint text space to be profiled
- *maxsymlinks* – specifies the maximum number of symlinks expanded in a pathname.

maxup

Description
: The *maxup* parameter defines the number of processes allowed per user login. This number should always be at least 5 processes smaller than *nproc*.

Value
: Default: 150 processes

Range: 15–10000 (but never larger than *nproc* - 5)

When to Change
: Increase this parameter to allow more processes per user. In a heavily loaded time-sharing environment, you may want to decrease the value to reduce the number of processes per user.

ngroups_max

Description
: The *ngroups_max* parameter specifies the maximum number of multiple groups to which a user may simultaneously belong.

The constants NGROUPS_UMIN <= *ngroups_max* <= NGROUPS_UMAX are defined in <*/usr/include/sys/param.h*>. NGROUPS_UMIN is the minimum number of multiple groups that can be selected at *lboot* time. NGROUPS_UMAX is the maximum number of multiple groups that can be selected at *lboot* time and is the number of group-id slots for which space is set aside at compile time. NGROUPS, which is present for compatibility with networking code (defined in <*/usr/include/sys/param.h*>), must not be larger than *ngroups_max*.

Value Default: 16

Range: 0-32

When to Change

The default value is adequate for most systems. Increase this parameter if your system has users who need simultaneous access to more than 16 groups.

maxwatchpoints

Description *maxwatchpoints* sets the maximum number of watchpoints per process. Watchpoints are set and used via the *proc*(4) file system. This parameter specifies the maximum number of virtual address segments to be watched in the traced process. This is typically used by debuggers.

Value Default: 100

Range: 1-1000

When to Change

Raise *maxwatchpoints* if your debugger is running out of watchpoints

nprofile

Description *nprofile* is the maximum number of disjoint text spaces that can be profiled using the *sprofil*(2) system call. This is useful if you need to profile programs using shared libraries or profile an address space using different granularities for different sections of text.

Value Default: 100

Range: 100-200

When to Change

Change *nprofile* if you need to profile more text spaces than are currently configured.

maxsymlinks

Description
This parameter defines the maximum number of symbolic links that will be followed during filename lookups (for example, during the *open*(2) or *stat*(2) system calls) before ceasing the lookup. This limit is required to prevent loops where a chain of symbolic links points back to the original file name.

Value
Default: 30

Range: 0-50

When to Change
Change this parameter if you have pathnames with more than 30 symbolic links.

Resource Limits Parameters

You can set numerous limits on a per-process basis by using *getrlimit*(2), *setrlimit*(2), and *limit*, the shell built-in command. These limits are inherited, and the original values are set in */var/sysgen/mtune.* These limits are different from the system limits listed above in that they apply only to the current process and any child processes that may be spawned. To achieve similar effects, you can also use the *limit* command within the Bourne, C, and Korn shells (*/bin/sh*, */bin/csh*, and */bin/ksh*).

Each limit has a default and a maximum. Only the superuser can change the maximum. Each resource can have a value that turns off any checking RLIM_INFINITY. The default values are adequate for most systems.

The following parameters are associated with system resource limits:

- *ncargs* – the number of bytes of arguments that may be passed during an *exec*(2) call.
- *rlimit-core-cur* – the maximum size of a core file.
- *rlimit-core-max* – the maximum value *rlimit-core-cur* may hold.
- *rlimit-cpu-cur* – the limit for maximum cpu time available to a process.
- *rlimit-cpu-max* – the maximum value *rlimit-cpu-current* may hold.
- *rlimit-data-cur* – the maximum amount of data space available to a process.
- *rlimit-data-max* – the maximum value *rlimit-data-cur* may hold.

- *rlimit-fsize-cur* – the maximum file size available to a process.
- *rlimit-fsize-max* – the maximum value *rlimit-fsize-cur* may hold.
- *rlimit-nofile-cur* – the maximum number of file descriptors available to a process.
- *rlimit-nofile-max* – the maximum value *rlimit-nofile-cur* may hold.
- *rlimit-rss-cur* – the maximum resident set size available to a process.
- *rlimit-rss-max* – the maximum value *rlimit-rss-cur* may hold.
- *rlimit-stack-cur* – the maximum stack size for a process.
- *rlimit-stack-max* – the maximum value *rlimit-stack-cur* may hold.
- *rlimit-vmem-cur* – the maximum amount of virtual memory for a process.
- *rlimit-vmem-max* – the maximum value *rlimit-vmem-cur* may hold.
- *rsshogfrac* – the percentage of memory allotted for resident pages.
- *rsshogslop* – the number of pages above the resident set maximum that a process may use.
- *shlbmax* – the maximum number of shared libraries with which a process can link.

ncargs

Description
: The *ncargs* parameter specifies the maximum size of arguments in bytes that may be passed during an *exec*(2) system call.

  This parameter is specified in */var/sysgen/mtune.*

Value
: Default: 20480

  Range: 5120–262144

When to Change
: The default value is adequate for most systems. Increase this parameter if you get the following message from *exec*(2), *shell*(1), or *make*(1):

```
E2BIG arg list too long
```

Notes
: Setting this parameter too large wastes memory (although this memory is pageable) and may cause some programs to function incorrectly. Also note that some shells may have independent limits smaller than *ncargs*.

rlimit_core_cur

Description The current limit to the size of core image files for the given process.

Value Default: 0x7fffffffffffffff (9223372036854775807 minutes)

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a cap on core file size.

rlimit_core_max

Description The maximum limit to the size of core image files.

Value Default: 0x7fffffffffffffff (9223372036854775807 minutes)

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a maximum restriction on core file size. *rlimit_core_cur* cannot be larger than this value.

rlimit_cpu_cur

Description The current limit to the amount of cpu time in minutes that may be used in executing the process.

Value Default: 0x7fffffffffffffff (9223372036854775807 minutes)

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a cap on cpu usage.

rlimit_cpu_max

Description The maximum limit to the amount of cpu time that may be used in executing a process.

Value Default: 0x7fffffffffffffff (9223372036854775807 minutes)

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a maximum restriction on general cpu usage.

rlimit_data_cur

Description The current limit to the data size of the process.

Value Default: 0 (auto-configured to *rlimit_vmem_cur* ∗ NBPP (0x20000000))

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a cap on data segment size.

rlimit_data_max

Description The maximum limit to the size of data that may be used in executing a process.

Value Default: 0 (auto-configured to *rlimit_vmem_cur* ∗ NBPP (0x20000000))

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a maximum restriction on the size of the data segment of any process.

rlimit_fsize_cur

Description The current limit to file size on the system for the process.

Value Default: 0x7fffffffffffffff (9223372036854775807 bytes)

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a limit on file size.

rlimit_fsize_max

Description: The maximum limit to file size on the system.

Value: Default: 0x7fffffffffffffff (9223372036854775807 bytes)

Range: 0–0x7fffffffffffffff

When to change: Change this parameter when you want to place a maximum size on all files.

rlimit_nofile_cur

Description: The current limit to the number of open file descriptors that may be used in executing the process.

Value: Default: 200

Range: 40–0x7fffffffffffffff

When to change: Change this parameter when you want to place a cap on the number of open file descriptors.

rlimit_nofile_max

Description: The maximum limit to the number of open file descriptors that may be used in executing a process.

Value: Default: 2500

Range: 0–0x7fffffffffffffff

When to change: Change this parameter when you want to place a maximum restriction on the number of open file descriptors.

rlimit_rss_cur

Description
The current limit to the resident set size (the number of pages of memory in use at any given time) that may be used in executing the process. This limit is the larger of the results of the following two formulae:

physical_memory_size - 4 MB

or

physical_memory_size ∗ 9/10

Value
Default: 0 (Automatically configured if set to 0)

Range: 0–(*rlimit_vmem_cur* ∗ NBPP) (0x7fffffffffffffff)

When to change
Change this parameter when you want to place a cap on the resident set size of a process.

rlimit_rss_max

Description
The maximum limit to the resident set size that may be used in executing a process.

Value
Default: (*rlimit_vmem_cur* ∗ NBPP) (0x20000000)

Range: 0–(*rlimit_vmem_cur* ∗ NBPP) (0x7fffffffffffffff)

When to change
Change this parameter when you want to place a maximum restriction on resident set size.

rlimit_stack_cur

Description
The current limit to the amount of stack space that may be used in executing the process.

Value
Default: 64 MB (0x04000000)

Range: 0–0x7fffffffffffffff)

When to change
Change this parameter when you want to place a limit on stack space usage.

rlimit_stack_max

Description The maximum limit to the amount of stack space that may be used in executing a process.

Value Default: *rlimit_vmem_cur* ∗ NBPP (0x20000000)

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a maximum restriction on stack space usage.

rlimit_vmem_cur

Description The current limit to the amount of virtual memory that may be used in executing the process.

Value Default: 0 (auto-configured to *rlimit_vmem_cur* ∗ NBPP (0x20000000))

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a cap on virtual memory usage.

rlimit_vmem_max

Description The maximum limit to the amount of virtual memory that may be used in executing a process.

Value Default: 0 (aut-configured to *rlimit_vmem_cur* ∗ NBPP (0x20000000))

Range: 0–0x7fffffffffffffff

When to change
Change this parameter when you want to place a maximum restriction on virtual memory usage.

rsshogfrac

Description The number of physical memory pages occupied by a process at any given time is called its resident set size (RSS). The limit to the RSS of a process is determined by its allowable memory-use resource limit.

rsshogfrac is designed to guarantee that even if one or more processes are exceeding their RSS limit, some percentage of memory is always kept free so that good interactive response is maintained.

Processes are permitted to exceed their RSS limit until either:

- one or more processes exceed their default RSS limit (thereby becoming an ''RSS hog'') and the amount of free memory drops below *rsshogfrac* of the total amount of physical memory; or
- the amount of free memory drops below GPGSHI

In either of these cases, the paging daemon runs and trims pages from all RSS processes exceeding the RSS limit.

The parameter RSSHOGFRAC is expressed as a fraction of the total physical memory of the system. The default value is 75 percent.

This parameter is specified in */var/sysgen/mtune*. For more information, see the *gpgshi*, *gpgslo*, and *rsshogslop* resource limits.

Value — Default: 75% of total memory

Range: 0–100% of total memory

When to Change

The default value is adequate for most systems.

rsshogslop

Description — To avoid thrashing (A condition where the computer devotes 100% of its CPU cycles to swapping and paging), a process can use up to *rsshogslop* more pages than its resident set maximum (see "Resource Limits Parameters" on page 224).

This parameter is specified in */var/sysgen/mtune.* For more information, see the *rsshogfrac* resource limit.

Value — Default: 20

When to Change

The default value is adequate for most systems.

shlbmax

Description The *shlbmax* parameter specifies the maximum number of shared libraries with which a process can link.

This parameter is specified in */var/sysgen/mtune.*

Value Default: 8

Range: 3–32

When to Change
The default value is adequate for most systems. Increase this parameter if you see the following message from *exec*(2):

```
ELIBMAX cannot link
```

Paging Parameters

The paging daemon, *vhand*, frees up memory as the need arises. This daemon uses a "least recently used" algorithm to approximate process working sets and writes those pages out to disks that have not been touched during a specified period of time. The page size is 4K for 32 bit kernels and 16K for 64 bit kernels. When memory gets exceptionally tight, *vhand* may swap out entire processes.

The *vhand* program reclaims memory by:

- stealing memory from processes that have exceeded their permissible resident set size maximum, forcing delayed write data buffers out to disk (with *bdflush*) so that the underlying pages can be reused
- calling down to system resource allocators to trim back dynamically sized data structures
- stealing pages from processes in order of lowest-priority process first, and the least-recently-used page first within that process
- swapping out the entire process

The following tunable parameters determine how often *vhand* runs and under what conditions. Note that the default values should be adequate for most applications.

The following parameters are included:

- *bdflushr* – specifies how often, in seconds, the *bdflush* daemon is executed; *bdflush* performs periodic flushes of dirty file system buffers.
- *gpgsmsk* – specifies the mask used to determine if a given page may be swapped.
- *gpgshi* – the number of free pages above which *vhand* stops stealing pages.
- *gpgslo* – the number of free pages below which *vhand* starts stealing pages
- *maxlkmem* – The *maxlkmem* parameter specifies the maximum number of physical pages that can be locked in memory (by *mpin*(2) or *plock*(2)) by a non-superuser process.
- *maxsc* – the maximum number of pages that may be swapped by the *vhand* daemon in a single operation.
- *maxfc* – the maximum number of pages that will be freed at once.
- *maxdc* – the maximum number of pages that will be written to disk at once.
- *minarmem* – the minimum available resident pages of memory.
- *minasmem*– the minimum available swappable pages of memory.
- *tlbdrop* – number of clock ticks before a process' wired entries are flushed.
- *vfs_syncr* – The rate at which *vfs_syncr* is run, in seconds.
- *maxpglst* – The maximum number of pages that can be held in each pageout queue.

bdflushr

Description

The *bdflushr* parameter specifies how often, in seconds, the *bdflush* daemon is executed; *bdflush* performs periodic flushes of dirty file system buffers. It is parallel and similar to *vfs_syncr*.

This parameter is specified in */var/sysgen/mtune.* For more information, see the *autoup* kernel parameter.

Value

Default: 5

Range: 1–31536000 (1 Year)

When to Change

This value is adequate for most systems. Increasing this parameter increases the chance that more data could be lost if the system crashes. Decreasing this parameter increases system overhead.

gpgsmsk

Description

The *gpgsmsk* parameter specifies the mask used to determine if a given page may be swapped. Whenever the pager (*vhand*) is run, it decrements software reference bits for every active page. When a process subsequently references a page, the counter is reset to the limit (NDREF, as defined in */usr/include/sys/immu.h*). When the pager is looking for pages to steal back (if memory is in short supply), it takes only pages whose reference counter has fallen to *gpgsmsk* or below.

This parameter is specified in */var/sysgen/mtune.*

Also see */usr/include/sys/immu.h* and */usr/include/sys/tuneable.h* and the *gpgshi* and *gpgslo* kernel parameters for more information.

Value

Default: 2

Range: 0–7

When to Change

This value is adequate for most systems.

Notes

If the value is greater than 4, pages are written to the swap area earlier than they would be with the default value of *gpgsmsk*. Thus swapping/paging may occur before it should, unnecessarily using system resources.

gpgshi

Description

When the *vhand* daemon (page handler) is stealing pages, it stops stealing when the amount of free pages is greater than *gpgshi*.

In other words, *vhand* starts stealing pages when there are fewer than *gpgslo* free pages in the system. Once *vhand* starts stealing pages, it continues until there are *gpgshi* pages.

If, at boot time, *gpgslo* and *gpgshi* are 0, the system sets *gpgshi* to 8% of the number of pages of memory in the system, and sets *gpgslo* to one half of *gpgshi*.

This parameter is specified in */var/sysgen/mtune.* For more information, see the kernel parameters *gpgsmsk* and *gpgslo*.

Value

Default: 0 (automatically configured to 8% of memory if set to 0)

Range: 30 pages – 1/2 of memory

When to Change
This value is adequate for most systems.

Notes
If this parameter is too small, *vhand* cannot supply enough free memory for system-wide demand.

gpgslo

Description
When the *vhand* daemon (page handler) executes, it won't start stealing back pages unless there are fewer than *gpgslo* free pages in the system. Once *vhand* starts stealing pages, it continues until there are *gpgshi* pages.

This parameter is specified in */var/sysgen/mtune.* For more information, see the *gpgshi* and *gpgsmsk* kernel parameters.

Value
Default: 0 (automatically configured to half of *gpgshi* if set to 0)

Range: 10 pages – 1/2 of memory

When to Change
This value is adequate for most systems.

Notes
If this parameter is too small, *vhand* does not start swapping pages; thus entire processes must be swapped. If this parameter is too large, *vhand* swaps pages unnecessarily.

maxlkmem

Description
The *maxlkmem* parameter specifies the maximum number of physical pages that can be locked in memory (by *mpin*(2) or *plock*(2)) per non-superuser process.

This parameter is specified in */var/sysgen/mtune.*

Value
Default: 2000

Range: 0 pages – 3/4 of physical memory

When to Change
Increase this parameter only if a particular application has a real need to lock more pages in memory.

On multi-user servers, you may want to decrease this parameter and also decrease *rlimit_vmem_cur*.

Notes — When pages are locked in memory, the system can't reclaim those pages, and therefore can't maintain the most efficient paging.

maxfc

Description — The *maxfc* parameter specifies the maximum number of pages that may be freed by the *vhand* daemon in a single operation. When the paging daemon (*vhand*) starts stealing pages, it collects pages that can be freed to the general page pool. It collects, at most, *maxfc* pages at a time before freeing them. Do not confuse this parameter with *gpgshi*, which sets the total number of pages that must be free before *vhand* stops stealing pages.

This parameter is specified in */var/sysgen/mtune.*

Value — Default: 100

Range: 50–100

When to Change — This value is adequate for most systems.

maxsc

Description — The *maxsc* parameter specifies the maximum number of pages that may be swapped by the *vhand* daemon in a single operation. When the paging daemon starts tossing pages, it collects pages that must be written out to swap space before they are actually swapped and then freed into the general page pool. It collects at most *maxsc* pages at a time before swapping them out.

This parameter is specified in */var/sysgen/mtune.*

Value — Default: 100

Range: 8–100

When to Change — You may want to decrease this parameter on systems that are swapping over NFS (Network File System). This is always the case for diskless systems to increase performance.

maxdc

Description *maxdc* is the maximum number of pages which can be saved up and written to the disk at one time.

Value Default: 100 pages

Range: 1-100

When to Change

If the system is low on memory and consistently paging out user memory to remote swap space (for example, mounted via NFS), decrease this parameter by not more than 10 pages at a time. However, this parameter's setting does not usually make any measurable difference in system performance.

minarmem

Description This parameter represents the minimum available resident memory that must be maintained in order to avoid deadlock.

Value Default: 0 (Autoconfigured if set to 0)

When to Change

The automatically configured value of this parameter should always be correct for each system. You should not have to change this parameter.

minasmem

Description This parameter represents the minimum available swappable memory that must be maintained in order to avoid deadlock.

Value Default: 0 (Autoconfigured if set to 0)

When to Change

The automatically configured value of this parameter should always be correct for each system. You should not have to change this parameter.

tlbdrop

Description This parameter specifies the number of clock ticks before a process' wired entries are flushed.

Value Default: 100

When to Change

If *sar* indicates a great deal of transaction lookaside buffer (*utlbmiss*) overhead in a very large application, you may need to increase this parameter. In general, the more the application changes the memory frame of reference in which it is executing, the more likely increasing *tlbdrop* will help performance. You may have to experiment somewhat to find the optimum value for your specific application.

vfs_syncr

Description This parameter specifies the number of seconds between runs of the *vfs_syncr* process. It is similar and parallel to *bdflushr*.

Value Default: 30

Range: 1 - 31536000 (1 year)

When to Change

Under normal circumstances it is not necessary to change this parameter.

maxpglst

Description This parameter specifies that can be held in each of the pager's pageout queues.

Value Default: 0 (Autoconfigured)

Range: 50 - 1000

IPC Parameters

The IPC tunable parameters set interprocess communication (IPC) structures. These structures include IPC messages, specified in */var/sysgen/mtune/msg*; IPC semaphores, specified in */var/sysgen/mtune/sem*; and IPC shared memory, specified in */var/sysgen/mtune/shm*.

If IPC (interprocess communication) structures are incorrectly set, certain system calls will fail and return errors.

Before increasing the size of an IPC parameter, investigate the problem by using *ipcs*(1) to see if the IPC resources are being removed when no longer needed. For example, *shmget* returns the error ENOSPC if applications do not remove semaphores, memory segments, or message queues.

Note that IPC objects differ from most IRIX objects in that they are not automatically freed when all active references to them are gone. In particular, they are not deallocated when the program that created them exits.

Table A-1 lists error messages, system calls that cause the error, and parameters to adjust. Subsequent paragraphs explain the details you need to know before you increase the parameters listed in this table.

Table A-1 System Call Errors and IPC Parameters to Adjust

Message	System Call	Parameter
EAGAIN	*msgsnd*()	see below
EINVAL	*msgsnd*() *shmget*()	msgmax shmmax
EMFILE	*shmat*()	sshmseg
ENOSPC	*semget*() *shmget*()	msgmni semmni, semmns, shmmni

EAGAIN — If IPC_NOWAIT is set, *msgsnd* can return EAGAIN for a number of reasons:

- The total number of bytes on the message queue exceeds *msgmnb*.
- The total number of bytes used by messages in the system exceeds *msgseg* * *msgsz*.
- The total number of system-wide message headers exceeds *msgmax*.

EINVAL — *shmget* (which gets a new shared memory segment identifier) will fail with EINVAL if the given size is not within *shmmin* and *shmmax*. Since *shmmin* is set to the lowest possible value (1), and *shmmax* is very large, it should not be necessary to change these values.

EMFILE *shmat* will return EMFILE if it attaches more than *sshmseg* shared memory segments. *sshmseg* is the total number of system-shared memory segments per process.

ENOSPC *shmget* will return ENOSPC if:

- *shmmni* (the system-wide number of shared memory segments) is too small. However, applications may be creating shared memory segments and forgetting to remove them. So, before making a parameter change, use *ipcs*(1) to get a listing of currently active shared memory segments.

semget will return ENOSPC if:

- *semmni* is too small, indicating that the total number of semaphore identifiers is exceeded.
- *semmns* (the system-wide number of semaphores) is exceeded. Use *ipcs* to see if semaphores are being removed as they should be.

msgget will return ENOSPC if:

- *msgmni* is too small. Use *ipcs* to see if message queues are being removed as they should be.

IPC Messages Parameters

If no one on the system uses or plans to use IPC messages, you may want to consider excluding this module. The following tunable parameters are associated with interprocess communication messages (see the *msgctl*(2) reference page):

- *msgmax* – specifies the maximum size of a message.
- *msgmnb* – specifies the maximum length of a message queue.
- *msgmni* – specifies the maximum number of message queues system-wide.
- *msgseg* – specifies the maximum number of message segments system-wide.
- *msgssz* – specifies the size, in bytes, of a message segment.
- *msgtql* – specifies the maximum number of message headers system-wide.

msgmax

Description — The *msgmax* parameter specifies the maximum size of a message.

This parameter is specified in */var/sysgen/mtune/msg*.

Value — Default: 16 ∗ 1024 (0x4000)

Range: 512-0x8000

When to Change

Increase this parameter if the maximum size of a message needs to be larger. Decrease the value to limit the size of messages.

msgmnb

Description — The *msgmnb* parameter specifies the maximum length of a message queue.

This parameter is specified in */var/sysgen/mtune/msg*.

Value — Default: 32 * 1024 (0x8000)

Range: *msgmax*–1/2 of physical memory

When to Change

Increase this parameter if the maximum number of bytes on a message queue needs to be longer. Decrease the value to limit the number of bytes per message queue.

msgmni

Description — The *msgmni* parameter specifies the maximum number of message queues system-wide.

This parameter is specified in */var/sysgen/mtune/msg*.

Value — Default: 50

Range: 10–1000

When to Change

Increase this parameter if you want more message queues on the system. Decrease the value to limit the message queues.

Notes — If there are not enough message queues, a *msgget*(2) system call that attempts to create a new message queue returns the error:

```
ENOSPC: No space left on device
```

msgseg

Description — The *msgseg* parameter specifies the maximum number of message segments system-wide. A message on a message queue consists of one or more of these segments. The size of each segment is set by the *msgssz* parameter.

This parameter is specified in */var/sysgen/mtune/msg*.

Value — Default: 1536

When to Change — Modify this parameter to reserve the appropriate amount of memory for messages. Increase this parameter if you need more memory for message segments on the system. Decrease the value to limit the amount of memory used for message segments.

Notes — If this parameter is too large, memory may be wasted (saved for messages but never used). If this parameter is too small, some messages that are sent will not fit into the reserved message buffer space. In this case, a *msgsnd*(2) system call waits until space becomes available.

msgssz

Description — The *msgssz* parameter specifies the size, in bytes, of a message segment. Messages consist of a contiguous set of message segments large enough to accommodate the text of the message. Using segments helps to eliminate fragmentation and speed the allocation of the message buffers.

This parameter is specified in */var/sysgen/mtune/msg*.

Value — Default: 8

When to Change — This parameter is set to minimize wasted message buffer space. Change this parameter only if most messages do not fit into one segment with a minimum of wasted space.

If you modify this parameter, you may also need to change the *msgseg* parameter.

Notes — If this parameter is too large, message buffer space may be wasted by fragmentation, which in turn may cause processes that are sending messages to sleep while waiting for message buffer space.

msgtql

Description — The *msgtql* parameter specifies the maximum number of message headers system-wide, and thus the number of outstanding (unread) messages. One header is required for each outstanding message.

This parameter is specified in */var/sysgen/mtune/msg*.

Value — Default: 40

Range: 10–1000

When to Change — Increase this parameter if you require more outstanding messages. Decrease the value to limit the number of outstanding messages.

Notes — If this parameter is too small, a *msgsnd*(2) system call attempting to send a message that would put *msgtql* over the limit waits until messages are received (read) from the queues.

IPC Semaphores Parameters

If no one on the system uses or plans to use IPC semaphores, you may want to consider excluding this module.

The following tunable parameters are associated with interprocess communication semaphores (see the *semctl*(2) reference page):

- *semmni* – specifies the maximum number of semaphore identifiers in the kernel.
- *semmns* – specifies the number of ipc semaphores system-wide.
- *semmnu* – specifies the number of ”undo” structures system-wide.
- *semmsl* – specifies the maximum number of semaphores per semaphore identifier.

- *semopm* – specifies the maximum number of semaphore operations that can be executed per *semop*(2) system call.
- *semume* – specifies the maximum number of ''undo'' entries per undo structure.
- *semvmx* – specifies the maximum value that a semaphore can have.
- *semaem* – specifies the adjustment on exit for maximum value.

semmni

Description The *semmni* parameter specifies the maximum number of semaphore identifiers in the kernel. This is the number of unique semaphore sets that can be active at any given time. Semaphores are created in sets; there may be more than one semaphore per set.

This parameter is specified in */var/sysgen/mtune/sem*.

Value Default: 10

When to Change

Increase this parameter if processes require more semaphore sets. Increasing this parameter to a large value requires more memory to keep track of semaphore sets. If you modify this parameter, you may need to modify other related parameters.

semmns

Description The *semmns* parameter specifies the number of ipc semaphores system-wide. This parameter is specified in */var/sysgen/mtune/sem*.

Value Default: 60

When to Change

Increase this parameter if processes require more than the default number of semaphores.

Notes If you set this parameter to a large value, more memory is required to keep track of semaphores.

semmnu

Description — The *semmnu* parameter specifies the number of "undo" structures system-wide. An undo structure, which is set up on a per-process basis, keeps track of process operations on semaphores so that the operations may be "undone" if the structure terminates abnormally. This helps to ensure that an abnormally terminated process does not cause other processes to wait indefinitely for a change to a semaphore.

This parameter is specified in */var/sysgen/mtune/sem*.

Value — Default: 30

When to Change

Change this parameter when you want to increase/decrease the number of undo structures permitted on the system. *semmnu* limits the number of processes that can specify the UNDO flag in the *semop*(2) system call to undo their operations on termination.

semmsl

Description — The *semmsl* parameter specifies the maximum number of semaphores per semaphore identifier.

This parameter is specified in */var/sysgen/mtune/sem*.

Value — Default: 25

When to Change

Increase this parameter if processes require more semaphores per semaphore identifier.

semopm

Description — The *semopm* parameter specifies the maximum number of semaphore operations that can be executed per *semop*() system call. This parameter permits the system to check or modify the value of more than one semaphore in a set with each *semop*() system call.

This parameter is specified in */var/sysgen/mtune/sem*.

Value — Default: 10

When to Change
Change this parameter to increase/decrease the number of operations permitted per *semop*() system call. You may need to increase this parameter if you increase *semmsl* (the number of semaphore sets), so that a process can check/modify all the semaphores in a set with one system call.

semume

Description The *semume* parameter specifies the maximum number of "undo" entries per undo structure. An undo structure, which is set up on a per-process basis, keeps track of process operations on semaphores so that the operations may be "undone" if it terminates abnormally. Each undo entry represents a semaphore that has been modified with the UNDO flag specified in the *semop*(2) system call.

This parameter is specified in */var/sysgen/mtune/sem*.

Value Default: 10

When to Change
Change this parameter to increase/decrease the number of undo entries per structure. This parameter is related to the *semopm* parameter (the number of operations per *semop*(2) system call).

semvmx

Description The *semvmx* parameter specifies the maximum value that a semaphore can have.

This parameter is specified in */var/sysgen/mtune/sem*.

Value Default: 32767 (maximum value)

When to Change
Decrease this parameter if you want to limit the maximum value for a semaphore.

semaem

Description — The *semaem* parameter specifies the adjustment on exit for maximum value, alias *semadj*. This value is used when a semaphore value becomes greater than or equal to the absolute value of *semop*(2), unless the program has set its own value.

This parameter is specified in */var/sysgen/mtune/sem*.

Value — Default: 16384 (maximum value)

When to Change — Change this parameter to decrease the maximum value for the adjustment on exit value.

IPC Shared Memory Parameters

The following tunable parameters are associated with interprocess communication shared memory:

- *shmmax* – specifies the maximum size of an individual shared memory segment.
- *shmmin* – specifies the minimum shared memory segment size.
- *shmmni* – specifies the maximum number of shared memory identifiers system-wide.
- *sshmseg* – specifies the maximum number of attached shared memory segments per process.

shmmax

Description — The *shmmax* parameter specifies the maximum size of an individual shared memory segment.

This parameter is specified in */var/sysgen/mtune/shm*.

Value — Default: (*rlimit_vmem_cur* ∗ NBPP) (0x20000000)

32-bit Range: 0x1000 - 0x7fffffff

64-bit Range: 0x1000 - 0x7fffffffffffffff

When to Change
Keep this parameter small if it is necessary that a single shared memory segment not use too much memory.

shmmin

Description The *shmmin* parameter specifies the minimum shared memory segment size.

This parameter is specified in */var/sysgen/mtune/shm*.

Value Default: 1 byte

When to Change
Increase this parameter if you want an error message to be generated when a process requests a shared memory segment that is too small.

shmmni

Description The *shmmni* parameter specifies the maximum number of shared memory identifiers system-wide.

This parameter is specified in */var/sysgen/mtune/shm*.

Value 32-bit Default: 100

64-bit Default: 400

Range: 10–1000

When to Change
Increase this parameter by one (1) for each additional shared memory segment that is required, and also if processes that use many shared memory segments reach the *shmmni* limit.

Decrease this parameter if you need to reduce the maximum number of shared memory segments of the system at any one time. Also decrease it to reduce the amount of kernel space taken for shared memory segments.

sshmseg

Description — The *sshmseg* parameter specifies the maximum number of attached shared memory segments per process. A process must attach a shared memory segment before the data can be accessed.

This parameter is specified in */var/sysgen/mtune/shm*.

Value — Default: 100

Range: 1–SHMMNI

When to Change

Keep this parameter as small as possible to limit the amount of memory required to track the attached segments.

Increase this parameter if processes need to attach more than the default number of shared memory segments at one time.

Streams Parameters

The following parameters are associated with STREAMS processing.

- *nstrpush* – maximum number of modules that can be pushed on a stream.
- *nstrintr* – maximum number.
- *strctlsz* – maximum size of the *ctl* part of message.
- *strmsgsz* – maximum stream message size.
- *strholdtime* – maximum stream hold time.
- *strpmonmax* –

nstrpush

Description — *nstrpush* defines the maximum number of STREAMS modules that can be pushed on a single stream.

Value — Default: 9 (0x9)

Range: 9-10

When to Change

Change *nstrpush* from 9 to 10 modules when you need an extra module.

nstrintr

Description This parameter defines the streams buffers used at interrupt time.

Value Default: 1024 (0x400)

Range: 32-4096

strctlsz

Description *strctlsz* is the maximum size of the *ctl* buffer of a STREAMS message. See the *getmsg*(2) or *putmsg*(2) reference pages for a discussion of the *ctl* and *data* parts of a STREAMS message.

Value Default: 1024 (0x400)

When to Change

Change *strctlsz* when you need a larger buffer for the *ctl* portion of a STREAMS message.

strmsgsz

Description *strmsgsz* defines the maximum STREAMS message size. This is the maximum allowable size of the *ctl* part plus the *data* part of a message. Use this parameter in conjunction with the *strctlsz* parameter described above to set the size of the *data* buffer of a STREAMS message. See the *getmsg*(2) or *putmsg*(2) reference pages for a discussion of the *ctl* and *data* parts of a STREAMS message.

Value Default: 0x8000

When to Change

Change this parameter in conjunction with the *strctlsz* parameter to adjust the sizes of the STREAMS message as a whole and the *data* portion of the message.

strholdtime

Description *strholdtime* defines the STRHOLDTIME macro in <*strsubr.h*>. The purpose of this macro is to cut down on overhead on streams drivers. This happens at a cost to system latency.

Value Default: 50 (0x32)

Range: 0-1000

strpmonmax

Description *strpmonmax* defines the maximum number of private streams monitors.

Value Default: 4 (0x4)

Range: 0-1024

Signal Parameters

The following signal parameters control the operation of interprocess signals within the kernel:

- *maxsigq* – specifies the maximum number of signals that can be queued.

maxsigq

Description The maximum number of signals that can be queued. Normally, multiple instances of the same signal result in only one signal being delivered. With the use of the SA_SIGINFO flag, outstanding signals of the same type are queued instead of being dropped.

Value Default 64

Range 32-32767

When to Change

Raise *maxsigq* when a process expects to receive a great number of signals and 64 queue places may be insufficient to avoid losing signals before they can be processed. Change *maxsigq* to a value appropriate to your needs.

Dispatch Parameters

One of the most important functions of the kernel is "dispatching" processes. When a user issues a command and a process is created, the kernel endows the process with certain characteristics. For example, the kernel gives the process a priority for receiving CPU time. This priority can be changed by the user who requested the process or by the Superuser. Also, the length of time (*slice-size*) that a process receives in the CPU is adjustable by a dispatch parameter. The Periodic Deadline Scheduler (PDS) is also part of the dispatch group. The deadline scheduler is invoked via the *schedctl*(2) system call in a user program and requires the inclusion of <*sys/schedctl.h*>. The following parameters are included in the dispatch group:

- *ndpri_hilim* – sets the highest non-degrading priority a user process may have.
- *ndpri_lolim* – sets the lowest non-degrading priority a user process may have.
- *runq_dl_refframe* – sets a limit on the amount of the reference frame that can be allocated.
- *runq_dl_nonpriv* – controls the amount of the reference frame that can be allocated by non-privileged user processes.
- *runq_dl_use* – specifies the longest interval that a deadline process may request.
- *slice_size* – specifies the amount of time a process receives at the CPU.

ndpri_hilim

Description
: The *ndpri_hilim* parameter sets the highest non-degrading priority a user process may have.

  Note that the higher the numeric value of *ndpri_hilim*, the lower the priority of the process.

Value
: Default 128

  Range 30-255

When to Change
: Change this parameter when you want to limit user process priority in a busy system with many users or when you want to enable a high priority user process, for example, a real-time graphics application.

ndpri_lolim

Description *ndpri-lolim* sets the lowest possible non-degrading priority for a user process. Note that lower priority values give a process higher scheduling priority.

Value Default: 39

Range: 30-255

When to Change
The default value is adequate for most systems.

runq_dl_maxuse

Description This parameter sets an absolute limit on the amount of the reference frame (set by the *runq_dl_refframe* parameter) that can be allocated under any circumstances.

Value Default: 700

Range: 0-100000

When to Change
If your deadline-scheduled processes require more scheduled CPU time, increase the value of *runq_dl_maxuse* and *runq_dl_nonpriv.*

runq_dl_nonpriv

Description This parameter controls the amount of the reference frame (set by the *runq_dl_refframe* parameter) that can be allocated by non-privileged user processes.

Value Default: 200

Range: 0-100000

When to change
If your non-privileged deadline processes require more CPU time, increase this value.

runq_dl_refframe

Description
: This parameter specifies the longest interval that a deadline process may request.

Value
: Default: 1000

  Range: 0-100000

When to Change
: If you change the values of *runq_dl_nonpriv* and *runq_dl_use*, you may need to change this parameter as well, to expand the reference frame in which *runq_dl_nonpriv* and *runq_dl_use* act.

slice_size

Description
: *slice_size* is the default process time slice, expressed as a number of ticks of the system clock. The frequency of the system clock is expressed by the constant Hz, which has a value of 100. Thus each unit of *slice_size* corresponds to 10 milliseconds. When a process is given control of the CPU, the kernel lets it run for *slice_size* ticks. When the time slice expires or when the process voluntarily gives up the CPU (for example, by calling *pause*(2) or by doing some other system call that causes the process to wait), the kernel examines the run queue and selects the process with the highest priority that is eligible to run on that CPU.

  The *slice_size* parameter is defined in */var/sysgen/mtune/disp* and has the following formula:

  ```
  #define slice_size Hz / 30 int slice_size = slice_size
  ```

  Since *slice_size* is an integer, the resulting value is 3. This means that the default process time slice is 30 milliseconds.

Value
: Default: 3

  Range: 1–100

When to Change
: If you use the system primarily for compute-intensive jobs and interactive response is not an important consideration, you can increase *slice_size*. For example, setting *slice_size* to 10 gives greater efficiency to the compute jobs, since each time they get control of the CPU, they are able to run for 100 milliseconds before getting switched out.

In situations where the system runs both compute jobs and interactive jobs, interactive response time will suffer as you increase the value of *slice_size*.

EFS Parameters

The IRIX Extent File System works closely with the operating system kernel, and the following parameters adjust the kernel's interface with the file system.

The following parameters are defined in the *efs* group:

- *efs_inline* – Whether to store symbolic link information in the inode or not.

The following parameters are set in the *kernel* parameter group, and are used for file system tuning. They determine how many pages of memory are clustered together into a single disk write operation. Adjusting these parameters can greatly increase file system performance. Available parameters include:

- *dwcluster* – number of delayed-write pages to cluster in each push.
- *autoup* – specifies the age, in seconds, that a buffer marked for delayed write must be before the *bdflush* daemon writes it to disk.

efs_inline

Description — If this parameter is set to any value other than 0, any symbolic llink data is stored within a file's inode, rather than out-of-line.

Value — Default: 0

Range: 0-1

When to Change

It should not be necessary to change this variable. Increasing the value improves the lookup speed of symbolc links to some degree, however filesystems thus treated cannot be transferred to systems running releases of IRIX prior to version 5.3.

dwcluster

Description
: This parameter sets the maximum number of delayed-write pages to cluster in each push.

Value
: Default: 64

When to Change
: It should not be necessary to change this parameter. The automatically configured value is sufficient.

autoup

Description
: The *autoup* parameter specifies the age, in seconds, that a buffer marked for delayed write must be before the *bdflush* daemon writes it to disk. This parameter is specified in */var/sysgen/mtune.* For more information, see the entry for the *bdflushr* kernel parameter.

Value
: Default: 10

  Range: 1–30

When to Change
: This value is adequate for most systems.

Loadable Drivers Parameters

IRIX allows you to load and run device drivers while the system remains up and running. Occasionally, you may have to make adjustments to the running kernel to allow for the extra resources these loadable drivers require. The following parameters allow you to make the necessary adjustments:

- *bdevsw_extra* – specifies an extra number of entries in the block device switch.
- *cdevsw_extra* – specifies an extra number of entries in the character device switch.
- *fmodsw_extra* – specifies an extra number of entries in the streams module switch.
- *vfssw_extra* – specifies an extra number of entries in the virtual file system module switch.
- *munlddelay* – specifies the timeout for auto-unloading loadable modules.

bdevsw_extra

Description
This parameter specifies an extra number of entries in the block device switch. This parameter is for use by loadable drivers only. If you configured a block device into the system at *lboot*(1M) time, you will not need to add extra entries to *bdevsw*.

Value
Default: 21

Range: 1-254

When to Change
Change this parameter when you have more than 21 block devices to load into dynamically the system. IRIX provides 21 spaces in the *bdevsw* by default.

cdevsw_extra

Description
This parameter specifies an extra number of entries in the character device switch. This parameter is for use by loadable drivers only. If you configured a character device into the system at *lboot*(1M) time, you will not need to add extra entries to *cdevsw*.

Value
Default: 23

Range: 3-254

When to Change
Change this parameter when you have more than 23 character devices to load dynamically into the system. IRIX provides 23 spaces in the *cdevsw* by default.

fmodsw_extra

Description
This parameter specifies an extra number of entries in the streams module switch. This parameter is for use by loadable drivers only. If you configured a streams module into the system at *lboot*(1M) time, you will not need to add extra entries to *fmodsw*.

Value
Default: 20

Range: 0-254

When to Change
Change this parameter when you have more than 20 streams modules to load dynamically into the system. IRIX provides 20 spaces in the *fmodsw* by default.

vfssw_extra

Description
This parameter specifies an extra number of entries in the *vnode* file system module switch. This parameter is for use by loadable drivers only. If you configured a *vfs* module into the system at *lboot* time, you will not need to add extra entries to *vfssw*.

Value
Default: 5

Range: 0-254

When to Change
Change this parameter when you have more than 5 virtual file system modules to load dynamically into the system. IRIX provides 5 spaces in the *vfssw* by default.

munlddelay

Description
This parameter specifies the timeout in minutes after which to auto-unload loadable modules.

Value
Default: 5 (0x5) minutes

CPU Actions Parameters

CPU actions parameters are used in multi-processor systems to allow the user to select the processor or processors that will be used to perform a given task.

The following parameters are defined:

- *nactions* – specifies the number of action blocks.

nactions

Description — The *nactions* parameter controls the number of action blocks. An action block lets you queue a process to be run on a specific CPU. The value of the *nactions* parameter is found by the formula:

maxcpu + 60

Value — Default: 0 (Auto-configured if set to 0)

Range: 60-200

When to Change — Increase the value of *nactions* if you see the kernel error message:

```
PANIC: Ran out of action blocks
```

Switch Parameters

The following parameters are simple on/off switches within the kernel that allow or disallow certain features, such as whether shells that set the user ID to the superuser are allowed:

- *dump_all_pages* – controls page dumping during kernel panics.
- *panic_on_sbe* – controls special factory debugging mode.
- *sbe_log_errors* – controls system logging of single bit errors.
- *sbe_mfr_override* – overrides default action of disabling single bit errors.
- *sbe_report_cons* – controls single bit error console reporting.
- *corepluspid* – controls core filenames.
- *r4k_div_patch* – controls patch code for *r4kpp* binaries.
- *mload_auto_rtsyms* – controls loading of the kernel symbol table.
- *xpg4_sticky_dir* – controls removal of files in "sticky" directories.
- *tty_auto_strhold* – controls the setting of STRHOLD on tty/pty lines.
- *reset_limits_on_exec* – controls resetting of rlimit values on new setuid processes.
- *ip26_allow_ucmem* – controls whether to allow access to uncached system memory on Power Indigo2 systems.
- *restrict_fastprof* – controls whether users can do fast (1ms) user level profiling.

- *reboot_on_panic* – specifies that the system should automatically reboot after a kernel panic.
- *svr3pipe* – controls whether SVR3.2 or SVR4 pipes are used.
- *nosuidshells* – when set to 0, allows applications to create superuser-privileged shells. When set to any value other than 0, such shells are not permitted.
- *posix_tty_default* – if the value of this switch is 0, the default Silicon Graphics line disciplines are used. If the value is set to 1, POSIX line disciplines and settings are used.
- *restricted_chown* – allows you to decide whether you want to use BSD UNIX style *chown*(2) system call or the System V style.
- *use_old_serialnum* – When set to 1, forces the kernel to use the old method of calculating a 32-bit serial number for *sysinfo -s*. This variable affects only Onyx and Challenge L or XL systems.
- *subnetsarelocal* – When set to 1, other subnets of a directly-connected subnetted network are considered to be local.

Note that all the above listed parameters are enforced system-wide. It is not possible to select different values on a per-process basis.

dump_all_pages

Description	This parameter, when set to 1, directs the system to dump all pages, kernel, user, and free, during a system panic. When set to 0, this feature is disabled and kernel pages only are dumped during a panic.
Value	Default: 1 (0x1)
	Range: 0 or 1

panic_on_sbe

Description	This parameter, when set to 1, turns on a special factory debugging mode called Single Bit Errors. When set to 0, this feature is disabled.
Value	Default: 0 (0x0)
	Range: 0 or 1

sbe_log_errors

Description	This parameter, when set to 1, directs the system to log single bit errors to the SYSLOG. When set to 0, this feature is disabled.
Value	Default: 0 (0x0)
	Range: 0 or 1
Notes	This parameter applies only to systems containing ECC memory (R8000-based systems such as CHALLENGE/Onyx)

sbe_mfr_override

Description	This parameter, when set to 1, overrides the default action of disabling single bit errors if the rate of single bit errors exceeds a predetermined limit. When set to 0, this feature is disabled.
Value	Default: 0 (0x0)
	Range: 0 or 1
Notes	This parameter applies only to systems containing ECC memory (R8000-based systems such as CHALLENGE/Onyx)

sbe_report_cons

Description	This parameter, when set to 1, directs the system to display single bit errors on the system console. When set to 0, this feature is disabled.
Value	Default: 0 (0x0)
	Range: 0 or 1
Notes	This parameter applies only to systems containing ECC memory (R8000-based systems such as CHALLENGE/Onyx)

corepluspid

Description	This parameter, when set to 1, directs the system to name core files with the process ID number appended. When set to 0, this feature is disabled.
Value	Default: 0 (0x0)
	Range: 0 or 1

r4k_div_patch

Description	This parameter, when set to 1, enables the exec patch code for binaries that have been processed with r4kpp for the divide in branch delay slot problem that occurs on R4000 SC rev 2.2 and 3.0 parts. When set to 0, this feature is disabled.
Value	Default: 0 (0x0) Range: 0 or 1

mload_auto_rtsyms

Description	This parameter, when set to 1, enables the standard automatic loading of the kernel’s run-time symbol table. When set to 0, this feature is disabled.
Value	Default: 1 (0x1) Range: 0 or 1

xpg4_sticky_dir

Description	This parameter, when set to 1, specifies that write access to a file does not allow that file to be removed if the "sticky bit" is set in the directory in which it resides. When set to 0, such files can be removed.
Value	Default: 1 (0x1) Range: 0 or 1

tty_auto_strhold

Description	This parameter, when set to 1, automatically sets STRHOLD on ttys/ptys whenever the line discipline is in canonical & echo mode and automatically clears STRHOLD otherwise.. When set to 0, this feature is under user control.
Value	Default: 0 (0x0) Range: 0 or 1

reset_limits_on_exec

Description: This parameter, when set to 1, directs the kernel to reset rlimit values on processes that run as **root**, in order to prevent non-**root** processes from enforcing resource limits. When set to 0, this feature is disabled and resource limits are not reset.

Value: Default: 1 (0x1)

Range: 0 or 1

ip26_allow_ucmem

Description: This parameter, when set to 0, prevents users from accessing uncached system memory on IP26 (Power Indigo2) systems. When set to 1, this feature is allowed

Value: Default: 0 (0x0)

Range: 0 or 1

When to Change

If this parameter is set to 0, attempts to access uncached memory will cause a system panic. If the feature is allowed, there is a substantial memory performance degradation.

restrict_fastprof

Description: This parameter, when set to 0, allows users to do fast (1 ms) profiling of programs with *prof*. When set to 1, this feature is disallowed

Value: Default: 0 (0x0)

Range: 0 or 1

reboot_on_panic

Description: This parameter, when set to 1, specifies that the system should automatically reboot after a kernel panic. This is especially useful for servers or other systems that frequently go unattended or are used remotely, where the user may not conveniently be able to physically reset and reboot the system.

When set to 0, the system must be rebooted from the console. Some systems (those with IP19, IP21, or IP22 processors, check your *hinv* listing for your processor type) store an environment variable in the PROM monitor called "*rebound*." If you have this variable and set the *reboot_on_panic* parameter to -1, your system will check the PROM environment for instructions. If the *rebound* variable is set to "*y*" then the system will reboot automatically. If the *rebound* variable is set to "*n*" the system will require manual reset. If you set a system without this environment variable to -1, it behaves as though the setting is 0 and does not automatically reboot.

Value
: Default: -1 (automatically reboot according to hardware platform implementation) or 0, depending on processor type.

  Range: -1, 0, or 1

When to Change
: Change this parameter if you wish to automatically reboot after a system panic.

svr3pipe

Description
: This parameter, when set to 1, specifies SVR3.2 style pipes, which are unidirectional. When set to 0, SVR4 style pipes are specified, which are bidirectional.

Value
: Default: 1 (SVR3.2 style pipes)

  Range: 0 or 1

When to Change
: Change this parameter if you wish to take advantage of SVR4 style pipes. SVR3 pipes are the default because they provide faster performance.

nosuidshells

Description
: Some programs are written so that they perform actions that require superuser privilege. In order to perform these actions, they create a shell in which the user has superuser privilege. Such shells pose a certain manageable risk to system security, but application developers are generally careful to limit the actions taken by the application in these

shells. The *nosuidshells* switch, when set to 0, allows these applications to create superuser-privileged shells. When set to any value other than 0, such shells are not permitted.

Value Default: 1 (setuid shells not permitted)

When to Change
Change this switch to allow setuid shells.

posix_tty_default

Description IRIX uses a default system of line disciplines and settings for serial lines. These default settings are different from those specified by POSIX. If the value of this switch is 0, the default Silicon Graphics line disciplines are used. If the value is set to 1, POSIX line disciplines and settings are used.

Value Default: 0

Range: 0 or 1

When to Change
Change this switch if you need to use POSIX line disciplines.

restricted_chown

Description This switch allows you to decide whether you want to use a BSD UNIX style *chown*(2) system call or the System V style. Under the BSD version, only the Superuser can use the *chown* system call to ″give away″ a file – to change the ownership to another user. Under the System V version, any user can give away a file or directory. If the value of the switch is 0, System V *chown* is enabled. If the value is not zero, BSD *chown* is enabled.

Value Default: 0

Range: 0 or 1

When to Change
Change this switch to choose which behavior you prefer for the *chown*(2) system call.

use_old_serialnum

Description — When set to 1, this parameter forces the kernel to use the old method (before IRIX Version 5) of calculating a 32-bit serial number for *sysinfo -s*. This variable affects only Onyx and Challenge L or XL systems.

Value — Default: 0

Range: 0 or 1

When to Change — Change this parameter on your Challenge or Onyx system if you need to use some older software that requires a 32-bit serial number.

subnetsarelocal

Description — When set to 1, all other subnets of a directly-connected subnetted network are considered to be local.

Value — Default: 0

Range: 0 or 1

When to Change — Change this parameter if no subnetted systems are directly connected to any external network, such as the internet.

Timer parameters

Timer parameters control the functioning of system clocks and timing facilities. The following parameters are defined:

- *fasthz* – sets the profiling/fast itimer clock speed.
- *itimer_on_clkcpu* – determines whether *itimer* requests are queued on the clock processor or on the running processor, respectively.
- *timetrim* – the system clock is adjusted every second by the signed number of nanoseconds specified by this parameter.

fasthz

Description This parameter is used to set the profiling/fast *itimer* clock speed.

Value Default: 1000

Range: 500-2500

When to Change
Change this parameter to give a finer or coarser grain for such system calls as *gettimeofday*(3B), *getitimer*(2) and *settimer*(2).

itimer_on_clkcpu

Description This parameter is set to either 0 or 1, to determine whether *itimer* requests are queued on the clock processor or on the running processor, respectively.

Value Default: 0

Range: 0 or 1

When to Change
If a process uses the *gettimeofday*(2) call to compare the accuracy of the *itimer* delivery, then you should set this parameter to 1, to take advantage of the clock processor. If the *itimer* request is for the purpose of implementing a user frequency-based scheduler, then set this parameter to 0 to queue the requests on the current running processor.

timetrim

Description The system clock is adjusted every second by the signed number of nanoseconds specified by this parameter. This adjustment is limited to 3 milliseconds or 0.3%. The *timed*(1M) and *timeslave*(1M) utilities periodically place suggested values in */var/adm/SYSLOG*.

Value Default: 0

Range: 0-0.3% of a second (3 milliseconds)

When to Change
Change this parameter as suggested by *timed* and *timeslave*. Read the relevant reference pages before taking any action.

NFS Parameters

The following parameters control the kernel-level functions of the Network File System (NFS). Reducing these values is likely to cause significant performance decreases in your system:

- *portmap_timeout* – sets the portmapper query timeout.
- *sm_timeout* – sets the status monitor timeout.
- *GraceWaitTime* – sets the NLM grace period wait time.
- *first_retry* – sets the number of retries on the first contact with the portmapper.
- *normal_retry* – sets the number of retries on later attempts to contact the portmapper.
- *lockd_grace_period* – sets the grace period for NMI timeouts.
- *lock_share_requests* – applies the corresponding IRIX file locks to share requests.
- *lockd_blocking_thresh* – sets the number of daemons allowed to block before a new daemon is created.
- *nfs_portmon* – set to 0, clients may use any available port. If it is set to 1, clients must use only privileged ports.
- *svc_maxdupreqs* – sets the number of cached NFS requests.

portmap_timeout

Description	This parameter specifies the portmapper query timeout, in 1/10 second increments.
Value	Default: 5 (0x5)
	Range: 1 to 200 (20 seconds)
Notes	Note that decreasing timeouts can severely degrade system performance.

sm_timeout

Description	This parameter specifies the status monitor communication timeout in 1/10 second increments.
Value	Default: 5 (0x5)

	Range: 1 to 150 (15 seconds)
Notes	Note that decreasing normal and working timeouts can severely degrade system performance.

GraceWaitTime

Description	This parameter specifies the NLM grace period, in seconds.
Value	Default: 5 (0x5)
	Range: 1 to 60 (1 minute)
Notes	Note that decreasing normal and working timeouts can severely degrade system performance.

first_retry

Description	This parameter specifies the number of retries to be made for the first contact with the portmapper.
Value	Default: 1 (0x1)
	Range: 0 to 10000

normal_retry

Description	This parameter specifies the number of subsequent retries after communication has once been established with the portmapper.
Value	Default: 1 (0x1)
	Range: 0 to 10000

lockd_grace_period

Description	This parameter specifies the NLM grace period, in seconds.
Value	Default: 45 (0x2d)
	Range: 1 to 3600 (1 hour)

lock_share_requests

Description This parameter determines whether or not the system will apply IRIX file locks to share requests.

Value Default: 0 (0x0)

Range: 0 or 1

lockd_blocking_thresh

Description This parameter specifies the number of daemons allowed to block before a new lock daemon will be spawned.

Value Default: 0 (0x0)

Range: 0 to 1000

nfs_portmon

Description This parameter determines whether or not a client must use a "privileged" port for NFS requests. Only processes with superuser privilege may bind to a privileged port. The *nfs_portmon* parameter is binary. If it is set to 0, clients may use any available port. If it is set to 1, clients must use only privileged ports.

Value Default: 0

Range: 0 or 1

When to Change

You should change this parameter only if it is absolutely necessary to maintain root privilege on your NFS mounted file systems and you have checked each NFS client to be sure that it requests a privileged port. If there are any clients requesting non-privileged ports, they will be unable to mount the file systems.

Additionally, changing the value of *nfs_portmon* to 1 can give a false sense of security. A process must have *root* privilege in order to bind to a privileged port, but a single "insecure" machine compromises the security of this privilege check.

svc_maxdupreqs

Description This parameter sets the number of cached NFS requests.

Value Default: 409

Range: 400-4096

When to Change
This parameter should be adjusted to the service load so that there is likely to be a response entry when the first retransmission comes in.

Socket Parameters

This section describes socket parameters. Both UDS sockets and TCP/IP sockets are covered.

Under UNIX domain sockets, there is a pair of buffers associated with each socket in use. There is a buffer on the receiving side of the socket, and on the sending side. The size of these buffers represent the maximum amount of data that can be queued. The behavior of these buffers differs depending on whether the socket is a streams socket or a data-gram socket.

On a streams socket, when the sender sends data, the data is queued in the receive buffer of the receiving process. If the receive buffer fills up, the data begins queueing in the sendspace buffer. If both buffers fill up, the socket blocks any further data from being sent.

Under data-gram sockets, when the receive buffer fills up, all further data-grams sent are discarded and the error EWOULDBLOCK is generated. Because of this behavior, the default receive buffer size for data-gram sockets is twice that of the send buffer.

In order to improve networking performance with a large number of connections, TCP and UDP packet lookup is performed using a hashing scheme. The number of hash buckets used by the lookup code is normally computed at system boot time, and is based on the number of megabytes of available system RAM. UDP uses four buckets per megabyte of system RAM, plus one additional bucket. TCP uses eight buckets per megabyte of system RAM, plus one additional bucket.

Each hash bucket requires 24 bytes of RAM on a 32-bit system, and 48 bytes of RAM on a 64-bit system. So, for example, on a 128MB Challenge XL system, UDP would use 513 hash buckets and TCP would use 1025 hash buckets. At 48 bytes per bucket, a total of 73824 bytes of system RAM are used hold hash table information when the default auto-configuration is used. On a 64MB Indy, which has less memory and a smaller word size, the total memory consumption using the default values is 18456 bytes of system RAM.

If the default size is not optimal for your system, the table sizes can be changed using the *udp_hashtablesz* and *tcp_hashtablesz* parameters listed below. Note that the kernel enforces some restrictions on the values that may be specified. The sizes do not include the dynamic space allocation for each TCP or UDP socket during its life time. These parameters are part of the "inpcb" parameter group, and are modified using *systune*(1M).

The following parameters control sockets:

- *unpst_sendspace* – UNIX domain socket stream send buffer size.
- *unpst_recvspace* – UNIX domain socket stream receive buffer size.
- *unpdg_sendspace* – UNIX domain socket data-gram send buffer size.
- *unpdg_recvspace* – UNIX domain socket data-gram receive buffer size.
- *udp_hashtablesz* – UNIX domain socket hash table size.
- *tcp_sendspace* – TCP/IP socket send buffer size.
- *tcp_recvspace* – TCP/IP socket receive buffer size.
- *tcp_hashtablesz* – TCP/IP hash table size.

unpst_sendspace

Description — This parameter controls the default size of the *send* buffer on streams sockets.

Value — Default: 0x4000 (16KB)

When to Change — It is generally recommended that you change the size of socket buffers individually, since changing this parameter changes the send buffer size on all streams sockets, using a tremendous amount of kernel memory. Also, increasing this parameter increases the amount of time necessary to wait for socket response, since all sockets will have more buffer space to read.

See the *setsockopt*(2) reference page for more information on setting specific socket options.

unpst_recvspace

Description This parameter controls the default size of the receive buffer of streams sockets.

Value Default: 0x4000 (16 Kbytes)

When to Change

It is generally recommended that you change the size of socket buffers on an individual basis, since changing this parameter changes the receive buffer size on all streams sockets, using a tremendous amount of kernel memory. Also, increasing this parameter will increase the amount of time necessary to wait for socket response, since all sockets will have more buffer space to read.

See the *setsockopt*(2) reference page for more information on setting specific individual socket options.

unpdg_sendspace

Description This parameter controls the size of a data-gram that can be sent over a socket.

Value Default: 0x2000 (8 KB)

When to Change

Data-gram sockets operate slightly differently from streams sockets. When a streams socket fills the receive buffer, all data is then sent to the send buffer, and when the send buffer fills up, an error message is issued. Data-gram sockets allow data-grams to fill the receive buffer, and when the receive buffer is full, all future data-grams are discarded, and the error EWOULDBLOCK is generated. Therefore, the *unpdg_sendspace* parameter serves only to limit the size of a data-gram to not more than can be received by the receive buffer.

Note that the default data-gram socket receive buffers are twice the size of the default data-gram send buffers, thus allowing the process to appear the same as the streams sockets.

It is generally recommended that you not change the size of this parameter without also changing the default receive buffer size for data-gram sockets. If you raise the value of this parameter (*unpdg_sendspace*) without raising the receive buffer size (*unpdg_recvspace*), you will allow the sending half of the socket to send more data than can be received by the receiving half. Also it is generally recommended that socket buffer sizes be set individually via the *setsockopt*(2) system call. See the *setsockopt*(2) reference page for more information on setting specific individual socket options.

unpdg_recvspace

Description
: This parameter controls the default size of data-gram socket receive buffers.

Value
: Default: 0x4000 (16 Kbytes)

When to Change
: It is generally recommended that you change the size of socket buffers individually, since changing this parameter changes the receive buffer size on all data-gram sockets, using a tremendous amount of kernel memory. Also, increasing this parameter increases the amount of time necessary to wait for socket response, since all sockets will have more buffer space to read.

See the *setsockopt*(2) reference page for more information on setting specific individual socket options.

udp_hashtablesz

Description
: This parameter controls the size of the UDP hash table.

Value
: Default: 0 (Auto-configure)

Range: 64-2048

When to Change
: This should not normally need to be changed, but might be changed in order to reduce the consumption of system RAM on systems that are not used to handle large numbers of networking applications, or to increase the number of hash buckets on systems that primarily provide networking services.

Using a zero value specifies that the table is to be sized based on system RAM. A non-zero value specifies the number of hash buckets to use for UDP. The system automatically adjusts this number to support the requirements of the hashing algorithm. For example, entering a value of 512 will create 513 buckets.

Restrictions — The system silently enforces a minimum value of 64 hash buckets and a maximum value of 2048 hash buckets.

tcp_sendspace

Description — This parameter controls the default size of the send buffer on TCP/IP sockets. There is a pair of buffers associated with each socket in use. There is a buffer on the receiving side of the socket, and on the sending side. The size of these buffers represent the maximum amount of data that can be queued.

Value — Default: 60Kbytes

Range: 0-512Kbytes

When to Change

The *tcp_sendspace* parameter may require tuning for slow throughput links, and for compatibility with TCP implementations based on 4.2BSD. For slow throughput links (such as lower speed PPP or SLIP connections), lower the value of this and the *tcp_recvspace* parameters until the best performance level is found. Very slow links such as PPP or SLIP over modems usually find a 3 Kbyte or 4 Kbyte buffer size to be the most efficient.

The *tcp_sendspace* parameter may require tuning for slow throughput links, and for compatibility with TCP implementations based on 4.2BSD. For slow throughput links (such as a Wide Area Net with a lower speed PPP or SLIP component), lower the value of this and the *tcp_recvspace* parameters on systems where the connection goes from fast (ethernet, or FDDI, etc) to slow (PPP or SLIP) until the best performance level is found. Very slow links such as PPP or SLIP over modems usually find a 3 Kbyte or 4 Kbyte buffer size to be the most efficient.

Conversely, high speed connections may profit from increasing the buffer size incrementally from 60 KBytes until maximum performance is achieved.

See the *setsockopt*(2) reference page for more information on setting specific socket options.

tcp_recvspace

Description — This parameter controls the default size of the TCP/IP receive buffer. There is a pair of buffers associated with each socket in use. There is a buffer on the receiving side of the socket, and on the sending side. The size of these buffers represent the maximum amount of data that can be queued.

Value — Default: 60 Kbytes

Range: 0-512Kbytes

When to Change

The *tcp_recvspace* parameter may require tuning for slow throughput links, and for compatibility with TCP implementations based on 4.2BSD. For slow throughput links (such as a Wide Area Net with a lower speed PPP or SLIP component), lower the value of this and the *tcp_sendspace* parameters on systems where the connection goes from fast (ethernet, or FDDI, etc) to slow (PPP or SLIP) until the best performance level is found. Very slow links such as PPP or SLIP over modems usually find a 3 Kbyte or 4 Kbyte buffer size to be the most efficient.

Conversely, high speed connections may profit from increasing the buffer size incrementally from 60 KBytes until maximum performance is achieved.

See the *setsockopt*(2) reference page for more information on setting specific individual socket options.

tcp_hashtablesz

Description — This parameter controls the size of the TCP hash table.

Value — Default: 0 (Auto-configure)

Range: 64-8192

When to Change

This parameter should not normally need to be changed, but might be changed in order to reduce the consumption of system RAM on systems that are not used to handle large numbers of networking applications, or to increase the number of hash buckets on systems that primarily provide networking services.

Using a zero value specifies that the table is to be sized based on system RAM. A non-zero value specifies the number of hash buckets to use for TCP. The system automatically adjusts this number to support the requirements of the hashing algorithm. For example, entering a value of 512 will create 1025 buckets, since TCP uses the extra space for processing connections in the TIME-WAIT state.

Restrictions
The system silently enforces a minimum value of 64 hash buckets and a maximum value of 8192 hash buckets.

VINO Parameters

This section describes Indy Video In No Out tunable parameters. The following parameter pertains to VINO:

- *vino_mtune_dmrbpages* – Limits the size of the DMA descriptor table used by the VINO video capture.

vino_mtune_dmrbpages

Description
The *vino_mtune_dmrbpages* parameter limits the size of the DMA descriptor table used by the VINO video capture. This in turn limits how many pages allocated to the Digital Media Ring Buffers are usable when capturing video. The number of pages in *vino_mtune_dmrbpages* is what the DMRB size specifically for VINO is limited to when allocated.

Value
Default: 0 (auto-configure)

If this parameter is set to zero, then a default based on the amount of memory available at run time is used. If a limit of less than that is desired, set this variable to the maximum number of pages that the ring buffers can use. A value of -1 indicates do not preallocate any space for VINO DMA descriptors.

Notes	Note that if a DMA descriptor table size is needed that is greater than the size specified here and cannot be allocated contiguously at run time (possibly due to memory fragmentation), then the video transfer will not be started, and will return an ENOMEM error. The error recovery is to attempt the capture with a smaller DMRB.

Extended Accounting Parameters

The following parameters control system accounting features. The parameters are:

- *do_procacct* - controls BSD process accounting.
- *do_extpacct* - controls extended process accounting.
- *do_sessacct* - controls array session accounting.
- *use_astbl* - controls array session table.
- *narsess* - controls the number of entries in an array session table.
- *dfltash* - controls the default array session handle for special sessions.
- *minash* - controls the first array session handle that can be assigned by the kernel.
- *maxash* - controls the largest array session handle that can be assigned by the kernel before wrapping back to *minash*.
- *asmachid* - controls the machine ID for generating global ASHs.
- *dfltprid* - controls the default project ID for special sessions.

do_procacct

Description	The *do_procacct* parameter, when set to 1, directs the system to perform BSD system accounting. When set to 0, the system does not perform the accounting, overriding the *acct*(2) system call.
Value	Default: 1 Range: 0-1

do_extpacct

Description	The *do_extpacct* parameter, when set to 1, directs the system to perform extended process accounting. When set to 0, the system does not perform the accounting.
Value	Default: 0 Range: 0-1

do_sessacct

Description	The *do_sessacct* parameter, when set to 1, directs the system to perform array session accounting when a process exits. When set to 0, the system does not perform the accounting.
Value	Default: 0 Range: 0-1

use_astbl

Description	The *use_astbl* parameter, when set to 1, directs the system to allocate array sessions in a single preallocated table. When set to 0, the system array sessions are allocated dynamically.
Value	Default: 0 for 32-bit kernels, 1 for 64-bit kernels Range: 0-1

narsess

Description	The *narsess* parameter specifies the number of entries in an array session table. When set to 0, the system auto-configures the value at 1/10th of the value of *nproc*.
Value	Default: 0 (autoconfigured at 1/10th *nproc*) Range: 10-10000

dfltash

Description The *dfltash* parameter specifies the default array session handle for special system sessions and other sessions that bypass ordinary session handle assignment.

Value Default: 0

Range: 0-0x7fffffffffffffff

minash

Description The *minash* parameter specifies the first array session handle that can be assigned by the kernel.

Value Default: 1

Range: 1-0x7ffffff00

maxash

Description The *maxash* parameter specifies the largest array session handle that can be assigned by the kernel before wrapping back to minash.

Value Default: 65535

Range: 255 -0x7fffffff

asmachid

Description The *asmachid* parameter specifies the system ID for generating global ASHs. No other system in an array should have the same system ID. If 0 is specified, the kernel will only generate local ASHs.

Value Default: 0

Range: 0 -0x7fff

dfltprid

Description	The *dfltprid* parameter specifies the default project ID for special system sessions and other sessions that bypass ordinary project ID assignment.
Value	Default: 0
	Range: 0 -0x7fffffffffffffff