

# CHAPTER 8

---

## Configuring an HP-UX Kernel

### Introduction

Kernel management in HP-UX 11i is done with a set of kernel configuration commands or through the web-based **kcweb** tool. This chapter covers the kernel-related commands, gives some examples of using the commands to modify and build kernels, and gives an overview of **kcweb**.

Most applications require that you to modify your kernel in some way. You may need to create a new HP-UX kernel to add device drivers or subsystems, tune the kernel to get improved performance, alter configurable parameters, or to change the dump and swap devices. If you update or modify a dynamic element of your kernel, as shown in an example in this chapter, a reboot is not required. Updating or modifying a static element requires a reboot and may also require some additional steps, which a later example shows.

This chapter covers the following topics:

- Overview, examples, and running kernel-related commands: **kcmodule**, **kctune**, **kconfig**, **kcllog**, **kcalarm**, **kcusage**, and **kcmod**
- A flowchart showing a typical kernel rebuild
- Example of using the entries in the flowchart to rebuild a kernel
- Example of booting a saved kernel configuration

- The **system** file and making multiple kernel changes using it
- Overview of the Web-based kernel tool **kcweb**

## Kernel Commands

A new set of kernel-related commands has been developed that have a common behavior. This section covers all the commands and provides examples of using some of them.

### kcmodule

**kcmodule** queries and changes kernel modules in the currently running kernel configuration or a saved kernel configuration that you are staging for future use. Hundreds of modules are present in an HP-UX kernel that consists of device drivers, kernel subsystems, and other kernel code.

**kcmodule** with no options provides the modules in your system and both their current state and the state on next boot if any changes are pending, as shown in this abbreviated example:

```
# kcmodule
Module      State  Cause  Notes
DeviceDriver  unused
KeyboardMUX  unused
LCentIf     static best
MouseMUX    unused
UsbBootKeyboard  unused
UsbBootMouse  unused
UsbHub      unused
UsbMiniBus  unused
UsbOhci     unused
acpi_node   static best
arp         static depend
asio0       static best
audio       static best
autofsc     static best
azusa_psm   static best
beep        static depend
btlan       static best
c460gx_psm  static depend
c8xx        static best
cachefsc    static best
ccio        unused
cdfs        auto   best   auto-loadable, unloadable
cec_hp      static depend
cell        static best
cifs        static best
clone       static best
consp1      unused
```

```

diag2          static best
dlpi           static best
dm_sample_fsid unused
dmapi         unused
dmp           static depend
dmsample      unused
echo         static best
ehci         unused
fed          static best
fcms         static depend
fcp          static depend
fcp_cdio     static depend
fcparray     static depend
fcpdev       static depend
fcpmux       static depend
fddi4        static best
ffs          static best
framebuf     unused
gelan        static best
graph3       unused
gvid         unused
gvid_core    unused
hcd          unused
hid          unused
hpstreams    static best
hsx          static explicit loadable, unloadable
hub          unused
ia64_psm     static best
idds         unused
.
.
.

```

This abbreviated output shows some of the modules in the system without much detail. We modify the last module shown, `idds`, in a later example. To get detailed information on a specific module, use the `-v` option, as shown for `vxfs`:

```

# kcmodule -v vxfs
Module          vxfs [3F559170]
Description     Veritas Journal File System (VxFS)
State           static (best state)
State at Next Boot static (best state)
Capable         static unused
Depends On     module libvxfs:0.0.0
               interface HPVUX_11_23:1.0

#

```

This verbose output shows more information for the particular module that we specified. I could also have requested verbose output for every module.

For every module in the verbose output, there is a name and description such as a name of `vxfs`, the version number in square brackets after the name, and a short description of the module in the example. It is possible for multiple versions to be listed if, for instance, the currently running kernel uses a different version than will be used on the next boot.

The state of the module is relative to the currently running kernel (which is shown in the example), the next boot (both the currently running and next boot states are shown), or a saved configuration. The module in the example is for the currently running kernel, so *static* means that the module is statically bound into the kernel and changing this state would require relinking the kernel executable and rebooting. The module could also be in the *unused* state, which means it is installed but not used, the *loaded* state, which means it has been dynamically loaded into the kernel, or the *auto* state, which means it will be dynamically loaded when it is first needed but hasn't been loaded yet.

The following list shows commonly used options to **kcmodule**:

**kcmodule** command-line flags:

<i>-a</i>	Includes all modules in the output.
<i>-B</i>	Backs up the currently running configuration prior to changing it.
<i>-c config</i>	Specifies the saved configuration to manage. If none is specified, manage the currently running configuration.
<i>-C comment</i>	Includes a comment pertaining to this invocation of the command in the kernel configuration log file.
<i>-d</i>	Adds the description for each item.
<i>-D</i>	Displays elements for which there is a pending change at the next boot.
<i>-h</i>	Holds the specified change(s) for the next boot.
<i>-K</i>	Keeps the currently running configuration, but does not back it up. Keep the existing backup unmodified.
<i>-P</i>	Parses using the specified output format.
<i>-S</i>	Displays the elements that have been set to something other than the default.
<i>-v</i>	Displays items using verbose output.

Some of these options will be used in the upcoming example of updating the kernel.

## kctune

**kctune** queries and changes the value of kernel tunable parameters in the currently running kernel configuration or a saved kernel configuration that you are staging for future use.

**kctune** with no options provides the parameters in your system, as shown in the following abbreviated example:

```
# kctune
Tunable          Value Expression Changes
NSTREVENT        50 Default
NSTRPUSH         16 Default
NSTRSCHEDED      0 Default
STRCTLSZ        1024 Default
STRMSGSZ         0 Default
acctresume       4 Default
acctsuspend      2 Default
aio_listio_max   256 Default Immed
aio_max_ops      2048 Default Immed
aio_monitor_run_sec 30 Default Immed
aio_phymem_pct   10 Default
aio_prio_delta_max 20 Default Immed
aio_proc_thread_pct 70 Default Immed
aio_proc_threads 1024 Default Immed
aio_req_per_thread 1 Default Immed
allocate_fs_swapmap 0 Default
alwaysdump       0 Default Immed
bufcache_hash_locks 128 Default
chang_hash_locks 256 Default
core_addshmem_read 1 1 Immed
core_addshmem_write 1 1 Immed
create_fastlinks 0 Default
dbc_max_pct      50 Default Immed
dbc_min_pct      5 Default Immed
default_disk_ir  0 Default
disksort_seconds 0 Default
dma32_pool_size  268435456 Default
dmp_rootdev_is_vol 0 Default
dmp_swapdev_is_vol 0 Default
dnlc_hash_locks  512 Default
dontdump         0 Default Immed
dst              1 Default
dump_compress_on 1 Default Immed
enable_idds      0 Default Immed
eqmемsize       15 Default
executable_stack 0 Default Immed
fs_async         0 Default
fs_symlinks     20 Default Immed
ftable_hash_locks 64 Default
hp_hfs_mtra_enabled 1 Default
io_ports_hash_locks 64 Default
ksi_alloc_max    33600 Default Immed
ksi_send_max     32 Default
max_acct_file_size 2560000 Default Immed
max_async_ports  50 Default
max_mem_window   0 Default
max_thread_proc  256 Default Immed
```

```

maxdsiz          1073741824 Default Immed
maxdsiz_64bit    4294967296 Default Immed
maxfiles         8192      8192
maxfiles_lim     8192      8192 Immed
maxrsessiz       8388608   Default
maxrsessiz_64bit 8388608   Default
maxssiz          8388608   Default Immed
maxssiz_64bit    268435456   Default Immed
maxtsiz          100663296   Default Immed
maxtsiz_64bit    1073741824   Default Immed
maxuprc          256      Default Immed
maxvgs           10      Default
msgmap           1026   Default
msgmax           8192   Default Immed
msgmnb           16384  Default Immed
msgmni           512   Default
msgseg           8192   Default
msgssz           96    Default
msgttl           1024  Default
ncdnode          150   Default Immed
nclist           8292  Default
ncsize           8976  Default
nfile            65536  Default Auto
nflocks          4096  Default Auto
ninode           4880  Default

```

```

.
.
.

```

This output shows the tunable parameter, its current value, the expressions used to compute the value (which is the default in all cases in the example except for maxfiles), and changes to the value if any are pending. In an upcoming example, I modify the nproc tunable.

Using the `-d` option, which also works with `kcmodule`, adds a description for each parameter, as shown in the following truncated example:

```

# kctune -d
Tunable          Value Expression  Changes
Description
NSTREVENT        50 Default
Maximum number of concurrent Streams bufcalls
NSTRPUSH         16 Default
Maximum number of Streams modules in a stream
NSTRSCHED        0 Default
Number of Streams scheduler daemons to run (0 = automatic)
STRCTLSZ         1024 Default
Maximum size of the control portion of a Streams message (bytes)
STRMSGSZ         0 Default
Maximum size of the data portion of a Streams message (bytes; 0 = unlimited)
acctresume       4 Default
Relative percentage of free disk space required to resume accounting
acctsuspend      2 Default
Relative percentage of free disk space below which accounting is suspended
aio_listio_max   256 Default Immed
Maximum number of async IO operations that can be specified in lio_list call
aio_max_ops      2048 Default Immed
Maximum number of async IO operations that can be queued at any time
aio_monitor_run_sec 30 Default Immed
Frequency of AIO Thread Pool Monitor Execution (in seconds)
aio_physmem_pct  10 Default

```

```

.
.
.

```

Each module now has a more detailed description associated with it as a result of using the `-d` option.

To group parameters based on the kernel module that defines the tunable, use the `-g` option, as shown in the following abbreviated example:

```
# kctune -g
Module      Tunable          Value Expression Changes
cdfs        ncdnode          150 Default  Immed
dump        alwaysdump      0 Default  Immed
dump        dontdump         0 Default  Immed
dump        dump_compress_on 1 Default  Immed
fs          bufcache_hash_locks 128 Default
fs          dbc_max_pct     50 Default  Immed
fs          dbc_min_pct     5 Default  Immed
fs          disksort_seconds 0 Default
fs          dnlc_hash_locks 512 Default
fs          fs_async        0 Default
fs          fs_symlinks     20 Default  Immed
fs          ftable_hash_locks 64 Default
fs          maxfiles        8192 8192
fs          maxfiles_lim   8192 8192  Immed
fs          ncsize         8976 Default
fs          nfile          65536 Default  Auto
fs          nflocks        4096 Default  Auto
fs          o_sync_is_o_dsync 0 Default
fs          sendfile_max   0 Default
fs          vnode_cd_hash_locks 128 Default
fs          vnode_hash_locks 128 Default
hpstreams  NSTREVENT       50 Default
hpstreams  NSTRPUSH        16 Default
hpstreams  NSTRSCHED       0 Default
hpstreams  STRCTL SZ      1024 Default
hpstreams  STRMSG SZ       0 Default
hpstreams  streampipes     0 Default
iddd       enable_iddd     0 Default  Immed
inet       tcpshashsz     2048 Default
io         aio_listio_max  256 Default  Immed
io         aio_max_ops     2048 Default  Immed
io         aio_monitor_run_sec 30 Default  Immed
io         aio_physmem_pct 10 Default
io         aio_prio_delta_max 20 Default  Immed
io         aio_proc_thread_pct 70 Default  Immed
io         aio_proc_threads 1024 Default  Immed
io         aio_req_per_thread 1 Default  Immed
io         io_ports_hash_locks 64 Default
io         max_async_ports 50 Default
ite        scroll_lines    100 Default  Immed
lvm        maxvgs         10 Default
pm         acctresume     4 Default
pm         acctsuspend    2 Default
pm         chang_hash_locks 256 Default
pm         dst            1 Default
          .
          .
          .
```

This output shows all the tunables grouped with their kernel modules. You can see that, in the case of the `fs` module, for example, many tunables are associated with some modules.

The `-v` output, as shown in the following abbreviated example, provides a lot of tunable-related information:

```

# kctune -v
Tunable          NSTREVENT
Description      Maximum number of concurrent Streams bufcalls
Module           hpstreams
Current Value    50 [Default]
Value at Next Boot 50 [Default]
Value at Last Boot 50
Default Value    50
Can Change       At Next Boot Only

Tunable          NSTRPUSH
Description      Maximum number of Streams modules in a stream
Module           hpstreams
Current Value    16 [Default]
Value at Next Boot 16 [Default]
Value at Last Boot 16
Default Value    16
Can Change       At Next Boot Only

Tunable          NSTRSCHED
Description      Number of Streams scheduler daemons to run (0 = automatic)
Module           hpstreams
Current Value    0 [Default]
Value at Next Boot 0 [Default]
Value at Last Boot 0
Default Value    0
Can Change       At Next Boot Only

Tunable          STRCTLSZ
Description      Maximum size of the control portion of a Streams message (bytes)
Module           hpstreams
Current Value    1024 [Default]
Value at Next Boot 1024 [Default]
Value at Last Boot 1024
Default Value    1024
Can Change       At Next Boot Only

Tunable          STRMSGSZ
Description      Maximum size of the data portion of a Streams message (bytes
; 0 = unlimited)
Module           hpstreams
Current Value    0 [Default]
Value at Next Boot 0 [Default]
Value at Last Boot 0
Default Value    0
Can Change       At Next Boot Only
Standard input

.
.
.

```

This output shows additional information, including the value the parameter will have upon the next boot.

All tunables have manual pages. If, for example, you want to know more about a tunable, just issue the **man** for the tunable, as shown in the following example for *nproc*: