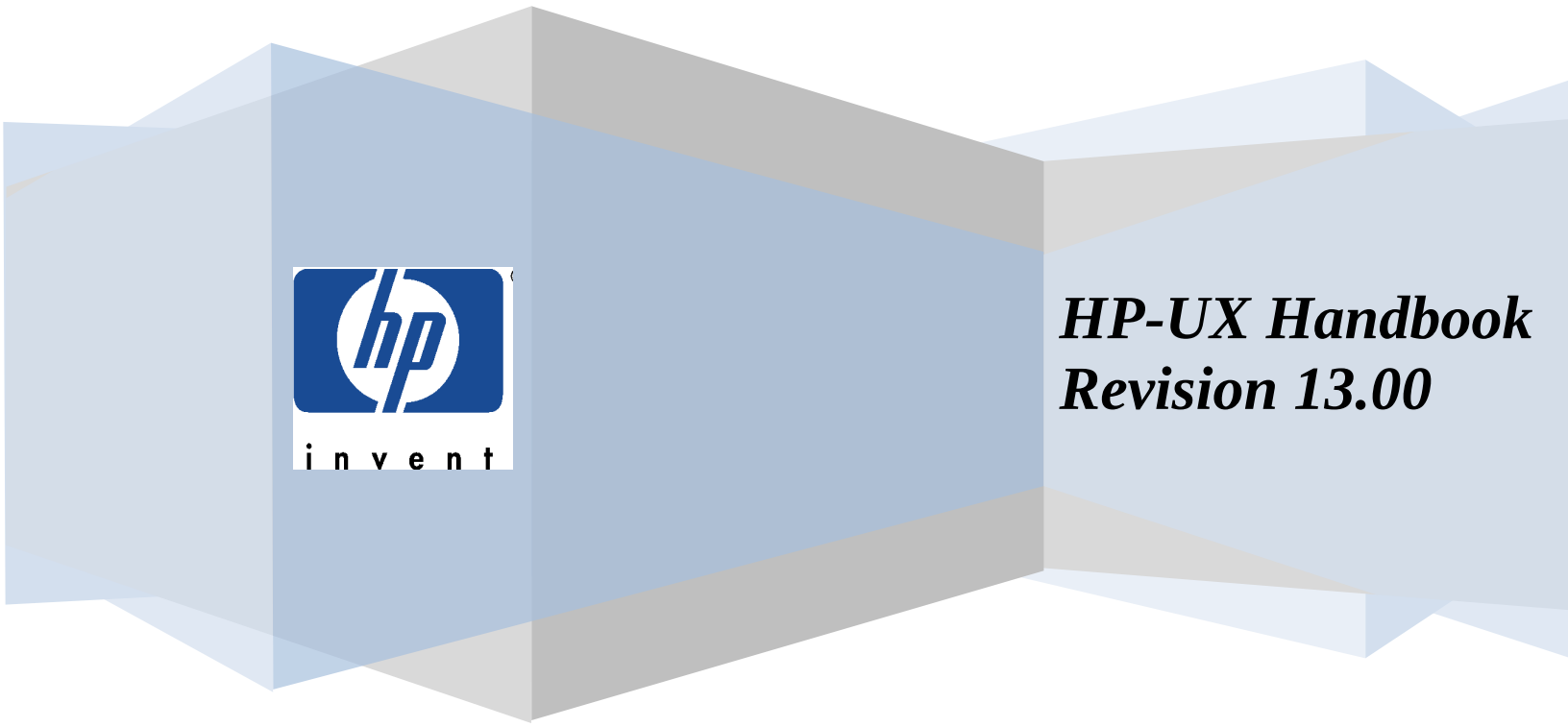
Chapter 07

Kernel Configuration



HP-UX Handbook
Revision 13.00

TERMS OF USE AND LEGAL RESTRICTIONS FOR THE HP-UX RECOVERY HANDBOOK

ATTENTION: PLEASE READ THESE TERMS CAREFULLY BEFORE USING THE HP-UX HANDBOOK. USING THESE MATERIALS INDICATES THAT YOU ACCEPT THESE TERMS. IF YOU DO NOT ACCEPT THESE TERMS, DO NOT USE THE HP-UX HANDBOOK.

THE HP-UX HANDBOOK HAS BEEN COMPILED FROM THE NOTES OF HP ENGINEERS AND CONTAINS HP CONFIDENTIAL INFORMATION.

THE HP-UX HANDBOOK IS NOT A PRODUCT QUALITY DOCUMENT AND IS NOT NECESSARILY MAINTAINED OR UP TO DATE. THE HP-UX HANDBOOK IS HERE MADE AVAILABLE TO HP CONTRACT CUSTOMERS FOR THEIR INTERNAL USE ONLY AND ON THE CONDITION THAT NEITHER THE HP-UX HANDBOOK NOR ANY OF THE MATERIALS IT CONTAINS IS PASSED ON TO ANY THIRD PARTY.

Use of the HP-UX Handbook: Hewlett-Packard Company ("HP") authorizes you to view and download the HP-UX Handbook only for internal use by you, a valued HP Contract Customer, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials. You may not modify the HP-UX Handbook in any way or publicly display, perform, or distribute or otherwise use them for any public or purpose outside your own business. The materials comprising the HP-UX Handbook are copyrighted and any unauthorized use of these materials may violate copyright, trademark, and other laws. If you breach any of these Terms, your authorization to use the HP-UX Handbook automatically terminates and you must immediately destroy any downloaded or printed materials.

Links To Other Web Sites: Links to third party Web sites provided by the HP-UX Handbook are provided solely as a convenience to you. If you use these links, you will leave this Site. HP has not reviewed all of these third party sites and does not control and is not responsible for any of these sites or their content. Thus, HP does not endorse or make any representations about them, or any information, software or other products or materials found there, or any results that may be obtained from using them. If you decide to access any of the third party sites linked to this Site, you do this entirely at your own risk.

Disclaimer: THE HP-UX HANDBOOK IS PROVIDED "AS IS" WITHOUT ANY WARRANTIES OF ANY KIND INCLUDING WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF INTELLECTUAL PROPERTY. HP further does not warrant the accuracy and completeness of the materials in the HP-UX Handbook. HP may make changes to the HP-UX Handbook at any time without notice. The HP-UX Handbook may be out of date, and HP makes no commitment to update the HP-UX Handbook. Information in the HP-UX Handbook may refer to products, programs or services that are not available in your country. Consult your local HP business contact for information regarding the products, programs and services that may be available to you.

Limitation of Liability: IN NO EVENT WILL HP, ITS SUPPLIERS, OR OTHER ANY THIRD PARTIES MENTIONED IN THE HP-UX HANDBOOK BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, THOSE RESULTING FROM LOST PROFITS, LOST DATA OR BUSINESS INTERRUPTION) ARISING OUT OF THE USE, INABILITY TO USE, OR THE RESULTS OF USE OF THE HP-UX HANDBOOK, WHETHER BASED ON WARRANTY, CONTRACT, TORT OR ANY OTHER LEGAL THEORY AND WHETHER OR NOT ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS DOES NOT APPLY IN CASE OF INTENT OR IF LIABILITY IS LEGALLY STIPULATED. IF YOUR USE OF THE HP-UX HANDBOOK RESULTS IN THE NEED FOR SERVICING, REPAIR OR CORRECTION OF EQUIPMENT OR DATA, YOU ASSUME ALL COSTS THEREOF.

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

Applicable Laws: These Terms will be governed by and construed in accordance with the laws of the State of California, without giving effect to any principles of conflicts of laws.

General: HP may revise these Terms at any time by updating this posting. ***Revised Oct 2013***

FEEDBACK or QUESTIONS: please email essam.ackleh@hp.com
(please use subject syntax: *HP-UX Handbook v13.00 Chapter <YY> - <Feedback Title>*

TABLE OF CONTENTS

Introduction

The architecture of the HP-UX kernel has been undergoing changes over several releases. The kernel is becoming more modular and more capable of being tuned while running.

- For HP-UX 11.11, the KM-commands should be used instead of editing the system file manually like the previous HPUX version (i.e. 10.X, 11.0). The dynamic tunables were introduced that could be modified without rebooting the system.
- For HP-UX 11.23 and 11.31, the concept of saved kernel configurations and new KC-commands were introduced. The kcweb interface was extended.

The next section describes the kernel configuration features for each HP-UX release and provides a procedure that describes how to tune the kernel parameters.

Kernel Configuration at HP-UX 11.11

Generating a new kernel is done via SAM or with a shell script called `mk_kernel(1M)`. Based on a user-provided **system file** (`/stand/system`) containing the names of the included device drivers and the settings of the kernel tunables, `mk_kernel` builds an executable file which can be used as a bootable kernel.

`mk_kernel(1M)` basically calls the `config(1M)` command that does the main work. `config(1M)` reads the system file and the master kernel configuration table information and generates the following output files:

- C program source files (`conf.c` and `space.h`) that define the configuration tables for various parts of the system.
- A makefile (`config.mk`) to compile the C program produced and re-link the newly configured system.

`config(1M)` executes the `make(1M)` command to compile `conf.c` and link the kernel with the appropriate kernel libraries. Many header files are needed to compile `conf.c`. Also, archive library files containing the kernel objects are needed to link the kernel. These files are supplied with the system and are contained in the directories found under `/usr/conf`. Libraries for the kernel are expected to be found in `/usr/conf/lib`. The master file used is the composite of files found under `/usr/conf/master.d/`.

If the build succeeds, the newly built binary will be a file named `vmunix_test` located in the build directory `/stand/build`. If another path is used to designate the system file, the build directory will be the present working directory.

NOTE: The bundled C compiler `/usr/ccs/bin/cc` is used for kernel compilation not the ANSI C compiler.

The System file /stand/system

The system file is the user-provided description of an HP-UX system. It comprises of three parts:

- Device driver
- Swap & dump devices
- Tunable system parameters

Details regarding the syntax can be found in the `config(1M)` manual page.

Device Driver, Pseudo Driver and Subsystems

Each line has the format

```
drivername
```

where the driver needs to exist in the kernel configuration table in the directory `/usr/conf/master.d`, i.e. a file called *drivername* must exist in that directory.

Example:

```
# grep -i fcms /stand/system
fcms

# ll /usr/conf/master.d/fcms
-r--r--r--   1 bin      bin      5828 Nov 20  1999 /usr/conf/master.d/fcms
```

Swap and Dump Devices

`/dev/vg00/lvol2` is the primary swap and dump device by default. If you like to configure a whole disk or a section of a disk as primary swap device, the following line needs to be added to the system file:

```
swap  [hw_path]  [offset]  [blocks]  [options]
```

This is not recommended because you won't get a warning if you mistakenly try to use this swap disk for LVM.

You can also configure dump devices on whole disks or sections. This is also not recommended for the same reasons. The system file should only contain the line by default:

```
dump   lvol
```

This entry is the default and it means that you can configure other dump devices in addition to the primary swap device, /dev/vg00/lvol2, (using `lvlnboot -d`).

I/O Statements for Devices

An I/O statement is used to link a device driver in the kernel with a dedicated HW address. This is necessary for devices that do not use a standard driver (like `spt`)

An I/O statement has the following syntax:

```
driver      hw_path      driver_name
```

Example:

```
driver      10/4/8.1.1      spt
```

Tunable Parameters

This part of the system file holds the settings of the kernel tunables. Tunables that do not appear in the system file keep their default value. Each line contains two fields. The first field represents the name of the tunable and the second represents the corresponding value. You can specify the value either in decimal or hexadecimal. Each line is independent, optional, and written in the following format:

```
parameter_name      number   or   formula
```

Examples:

```
nstrpty         60
msgmax          32768
maxdsiz         0x10000000
nproc           (100*maxusers)
```

Example of a System File

```
* Drivers and Subsystems

CentIf
CharDrv
c700
c720
ccio
cdfs
cio_ca0
 ...
 ...
tpiso
uipc
vxbase
wsio

* Kernel Device info

dump lvol
driver 10/4/8.1.1 spt
```

```
* Tunable parameters

nstrpty       60
msgmax        32768
msgmni        50
msgseg        7168
msgssz        8
msgtql        256
```

Kernel Configuration changes at HPUX 11.11

With the introduction of DLKM (Dynamically Loadable Kernel Modules) as of HP-UX 11.11, the kernel now is made of a static part and a number of dynamically loadable modules. The kernel does not only consist of a single file (`/stand/vmunix`), but of multiple files and directories (e.g. `/stand/dlkm/`) that are dependent from each other. The dynamically loadable modules can be activated *online* (i.e. without rebooting). They also can be activated by the kernel *on demand* (autoload).

The following files belong to a DLKM:

`/usr/conf/master.d/`*`module_name`*	Master configuration tables for kernel and kernel modules.
`/usr/conf/km.d/`*`module_name/mod.o`*	Source code file (object file).
`/stand/system.d/`*`module_name`*	Kernel module system file.
`/stand/dlkm/system.d/`*`module_name`*	Kernel module system file (copy).
`/stand/dlkm/mod.d/`*`module_name`*	Kernel module loadable image.

1. Procedure to generate a kernel at HPUX 11.11

Even though you wish to compile an old fashioned *static* kernel without using DLKMs, the procedure is different from the procedure for 10.X. The new command, `kmupdate(1M)`, is used to move the kernel file plus associated files to their appropriate places during system restart. This procedure is similar to HPUX 11.0.

In order to generate a new kernel follow the steps below:

1) Change to the kernel generation working directory and remove the files that have been left over from the last kernel generation:

```
# cd /stand/build
# rm -r *        (Be careful with this! You need to be in /stand/build/)
```

2) Backup the current system file:

```
# cp /stand/system /stand/system.prev
```

3) Is the system file up to date?

If you are not sure that the current system file (`/stand/system`) corresponds to the current kernel (`/stand/vmunix`), then use the `system_prep` script to extract a system file from the current kernel:

```
# /usr/lbin/sysadm/system_prep [-s /stand/system]
```

Compare the new system file to the original one. The diff command should not produce any results:

```
# diff /stand/system /stand/system.prev
```

4) Modify the system file

Use `kmtune(1M)` to modify kernel tunables. Use `kmsystem(1M)` to add/remove driver. These commands do nothing more than modifying `/stand/system`. Due to various dependencies, the system file should not be edited by hand, for example:

`# kmtune -s dbc_max_pct=35`	(changes tunable in system files)
`# kmsystem -c y diag2`	(adds driver `diag2` to system file)
`# kmsystem -c n diag2`	(removes driver `diag2` from system file)

The "km" commands are explained below in greater detail.

5) Generate the kernel

After you modified the system file, you can generate the new kernel using the mk_kernel script. The name of the new kernel will be `/stand/build/vmunix_test`, unless explicitly specified through the `-o` option. The `-s` option specifies the system file to use, `/stand/system` is the default:

```
# mk_kernel
```

6) Schedule kernel update

kmupdate is used to initiate the move of the new kernel (`/stand/build/vmunix_test`) and the corresponding files to their appropriate places at the next shutdown or startup. The current kernel will be copied to `/stand/vmunix.prev`, then.

```
# kmupdate
```

7) Restart the system

The new kernel will be activated by restarting the system:

```
# cd /
```

```
# shutdown -r 0
```

NOTE:
Do not use the `reboot` command. Otherwise the movement of the kernel files will be skipped and the system boots from the original kernel again. If you really need to reboot instead of shutdown, then execute the rc script manually before rebooting:

```
# /sbin/init.d/kmbuild stop
```

Overview about standard kernel files:

/stand/vmunix	kernel executable
/stand/vmunix.prev	kernel executable (backup)
/stand/system	system file
/stand/system.prev	system file (backup)
/stand/dlkm/	kernel function set directory
/stand/dlkm.vmunix.prev/	kernel function set directory (backup)
/stand/build/dlkm.vmunix_test/	kernel function set directory (built by mk_kernel)
/stand/build/vmunix_test	new kernel executable (built by mk_kernel)

2. Add a DLKM to the kernel

To install and activate a DLKM to the running kernel, you need the commands:

`kminstall(1M)` (installation of a module) and
`kmadmin(1M)` (activation of a module)

Generally a DLKM consists of three components:

code file	/usr/conf/km.d/*module_name*/mod.o
master file	/usr/conf/master.d/*module_name*
system file	/stand/system.d/*module_name*

Activating a DLKM

This example shows the installation and (de)activation of the EMS monitor "krm" (kernel resource monitor). The krmond daemon is a hardware monitor that is part of the EMS (Event Monitoring Services) subsystem. It simply monitors the usage of kernel tunables.

Check if krm is installed:

```
# swlist -l product EMS-KRMonitor
# Initializing...

  EMS-KRMonitor          A.11.11.04      EMS Kernel Resource Monitor
```

The product is not on the application CD (DART) but on the Support Plus CD together with the standard patch bundles. It is bundled with the Diagnostics bundle (OnlineDiag). See http://software.hp.com/. For HPUX 11.31, it's a MegaBundle which contains many sub-products such as OnlineDiag, SFM, WBEM, and so on.

It contains the following files:

`/dev/krm`	driver
`/etc/opt/resmon/lib/krm/mod.o`	code file
`/master`	master file
`/system`	system file
`/etc/opt/resmon/lbin/krmond`	daemon
`/opt/resmon/share/man/man1m/krmond.1m`	manual page
`/etc/opt/resmon/dictionary/krmond.dict`	dictionary file

In order to install the module, you need to copy the files (subsystem code, master and system file) to the correct places by using kminstall:

```
# cd /etc/opt/resmon/lib/krm
# kminstall -a krm
```

subsystem code to	`/usr/conf/km.d/`**`krm/mod.o`**
master file to	`/usr/conf/master.d/`**`krm`**
system file to	`/stand/system.d/`**`krm`**

Use `kmsystem` to check if the installation of the module was successful:

```
# kmsystem -q krm
Module                Configured          Loadable
================================================
krm                   Y                   Y
```

The command `mk_kernel` is used to generate a new kernel module, i.e. subsystem code, master and system file are combined into a single usable kernel module `/stand/dlkm/mod.d/`*`krm`*..
A copy of the system file will be placed in `/stand/dlkm/system.d/`*`krm`*.
Internally, the commands `config -M krm` and `kmupdate -M krm` are being executed.

```
# mk_kernel -M krm
Generating module: krm...

  Specified module(s) below is(are) activated successfully.
      krm

WARNING: Kernel update request is scheduled.
         Activated module(s) is(are) only available until system reboot.
```

`kmadmin` activates the module, i.e. it loads it into the running kernel.

```
# kmadmin -L krm
kmadmin: Module krm loaded, ID = 1
```

`kmadmin` is also useful to check whether the module has been loaded successfully:

```
# kmadmin -Q krm
Module Name             krm
Module ID               1
Module Path             /stand/dlkm/mod.d/krm
Status                  LOADED
Size                    8192
Base Address            0x429c000
BSS Size                0
BSS Base Address        0x0
Hold Count              1
Dependent Count         0
Unload Delay            0 seconds
Description             krm
Type                    WSIO
Block Major             -1
Character Major         240
Flags                   a5
```

The command

```
kminstall -d krm
```

removes the module again, i.e. the status is like before `kminstall -a` was executed.

Overview of the DLKM commands

The "km" commands can be found in the directory `/usr/sbin`

kminstall(1M)

Add/remove/update a kernel module. kminstall(1M) adds or removes the module's subsystem code, master and system file.

Removing the krm module:

```
# kminstall -d krm
# kmsystem -q krm
kmsystem: Invalid module, subsystem or driver name : krm
# cat /stand/system.d/krm
cat: Cannot open /stand/system.d/krm: No such file or directory
```

Adding the krm module again:

```
# kminstall -a krm
kminstall: Local directory must contain a mod.o file
# cd /etc/opt/resmon/lib/krm
# kminstall -a krm
```

```
# cat /stand/system.d/krm
* @(#)$Revision: 1.1 $ $Date: 2000-04-18 11:18:16-06 $
* Kernel Resource Monitor system file
* (C) Copyright 1999 Hewlett-Packard Company
$VERSION        1
$CONFIGURE      Y
```

The update option (-u) is neccessary whenever the subsystem code (mod.o) has changed.

Refer to kminstall(1M) man page for details

kmsystem(1M)

Displays/modifies the LOADABLE and CONFIGURATION flags for kernel modules. The LOADABLE flag determines whether the module gets loaded dynamically to the kernel or whether it gets statically linked to the kernel. The CONFIGURATION flag determines whether the module should be included during the next kernel compilation or not. Static drivers are added to/removed from the system file with this option.
kmsystem(1M) simply modifies the module's system file `/stand/system.d/<module>`

```
# kmsystem | more
Module               Configured          Loadable
================================================
CentIf               N                   -
CharDrv              N                   -
DlkmDrv              Y                   -
GSCtoPCI             Y                   -
PCItoPCI             Y                   -
SCentIf              N                   -
...
```

Query the current state of the krm module:

```
# kmsystem -q krm
Module               Configured          Loadable
================================================
krm                  Y                   Y
```

Set the LOADABLE flag to N:

```
# kmsystem -l N krm
# kmsystem -q krm
Module               Configured          Loadable
================================================
krm                  Y                   N

# cat /stand/system.d/krm
* @(#)$Revision: 1.1 $ $Date: 2000-04-18 11:18:16-06 $
* Kernel Resource Monitor system file
* (C) Copyright 1999 Hewlett-Packard Company
$VERSION        1
$CONFIGURE      Y
$LOADABLE       N
```

Refer to kmsystem(1M) man page for details.

kmtune(1M)

Displays/modifies/resets system tunables. kmtune simply modifies the system file.

```
# kmtune -q nproc
Parameter             Current Dyn Planned            Module     Version
================================================================================
nproc                    2068   -  2068

# kmtune -s nproc=3000
# kmtune -q nproc
Parameter             Current Dyn Planned            Module     Version
================================================================================
nproc                    2068   -  3000

# grep nproc /stand/system
nproc                 3000

# kmtune -s nproc+1000
# kmtune -q nproc
Parameter             Current Dyn Planned            Module     Version
================================================================================
nproc                    2068   -  4000

# kmtune -r nproc
# kmtune -q nproc
Parameter             Current Dyn Planned            Module     Version
================================================================================
nproc                    2068   -  (20+8*MAXUSERS)
# grep nproc /stand/system
```

Refer to kmtune(1M) man page for details.

kmadmin(1M)

Loads/unloads a DLKM to/from the running kernel. Displays status information of modules.

```
kmadmin -L module_name ...| path_name ...
kmadmin -U module_name ...
kmadmin -u module_id ...
kmadmin -Q module_name ...
kmadmin -q module_id ...
kmadmin -S | -s
kmadmin -k
kmadmin -d directory_name | -D
```

- `-L` (*load*) Loads the module(s) to the running kernel. *`path_name`* is the complete path to the source code file (`mod.o`) of the module.
- `-U` (*unload*) Unloads the module(s) from the running kernel.
- `-u` (*unload*) Same as `-U` (needs module-ID as input)
  Note: `kmadmin -u 0` unloads all modules.
- `-Q` (*query*) Displays status information of the module.
- `-q` (*query*) Same as `-Q` (needs module-ID as input).

`-S` (*status*) `kmadmin -Q` for all modules.

`-s` (*status*) Short form of `kmadmin -S`.

`-k` Prints a list of all statically configured modules.

`-d` (*directory*) Specifies the search path to the modules. Default is `/stand/dlkm/mod.d/`

`-D` (*directory*) Sets the search patch to the modules back to default: `/stand/dlkm/mod.d/`.

kmupdate(1M)

Update default kernel file and files associated with the kernel, or update specified kernel module(s).

(i) `kmupdate [kernel_file]`

(ii) `kmupdate -M module_name [[-M module_name]...] [-i|-a]`

form (i)

The specified kernel file will be moved to `/stand/vmunix` during next system **shutdown**. If no kernel file is specified, `/stand/build/vmunix_test` will be used. The associated kernel function set directory will be moved to `/stand/dlkm/`.

form (ii)

The files accociated with the specified module will be copied to their correct locations.

`-i` (*immediately*) When specified, kmupdate will only attempt an immediate update.

`-a` (*asynchronously*) When specified, kmupdate will update asynchronously without attempting an immediate update.

NOTE: `kmupdate` will first try `-i` option and (if this does not work) `-a` option.

kcweb

kcweb is a new web based interface, that offers the following features:

Kernel Configuration changes
You can modify tunables, add or remove static drivers, modify the state of kernel modules, compile a new kernel and reboot if needed.

Monitoring
Certain tunables can be monitored. A daily percentage usage graph helps you to determine the appropriate adjustments to tunables.

Alarms

With kcweb alarms can be set for certain parameters. It is possible to be alarmed if e.g. nproc usage is greater than 80%. kcweb alarms use EMS notifications. All alarms are managed by kcalarm(1M).

Range checking
Another change compared to UX 11.00/11.11 is that the range checking of tunables is now performed directly in the kernel. kctune reports out of range errors, kcweb shows stderr output of kctune.

Help
You can access the man page of each tunable and get a detailed explanation.

Command preview
A great advantage compared to SAM is that for each task (e.g. modifying a tunable, module, or alarm) you can use the command preview feature by choosing the ? button; this will show the kernel configuration command invocation that will perform the requested task.

kcweb can be launched from SAM GUI (as of UX 11.23 additionally from SAM TUI) or manually. To access kcweb interface start the kcweb server on the system and access it from any browser.

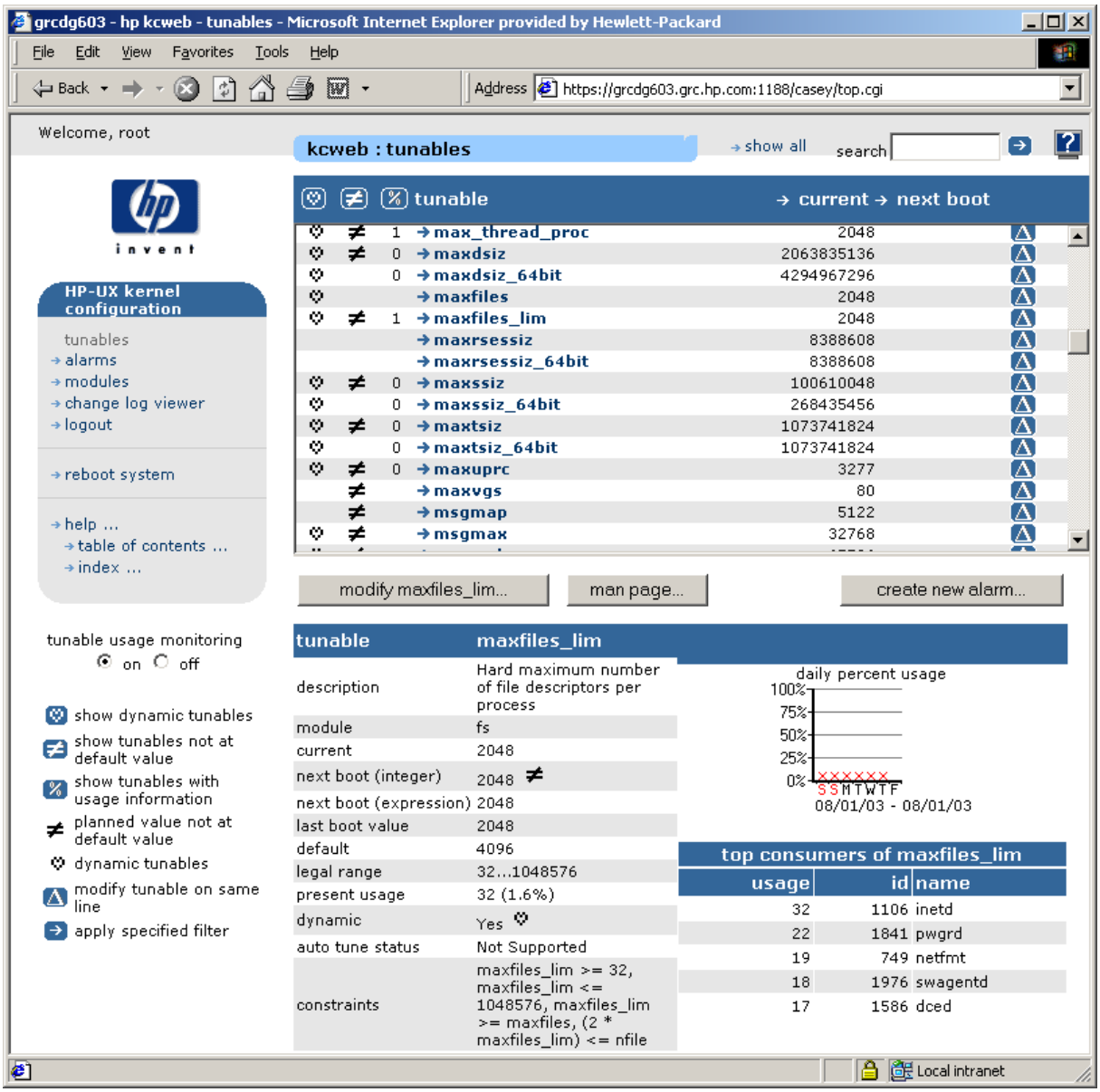
```
# kcweb -s startssl
Attempting to start server...
Server successfully started.
```

This creates server certificates (if needed), starts the kcweb administration server, connects to the server and presents a login screen.

At the following address you can access kcweb:

https://<hostname>:1188/casey/top.cgi

After logging in as root you will see the following window:



To stop the web server do:

```
# kcweb -s stop
```

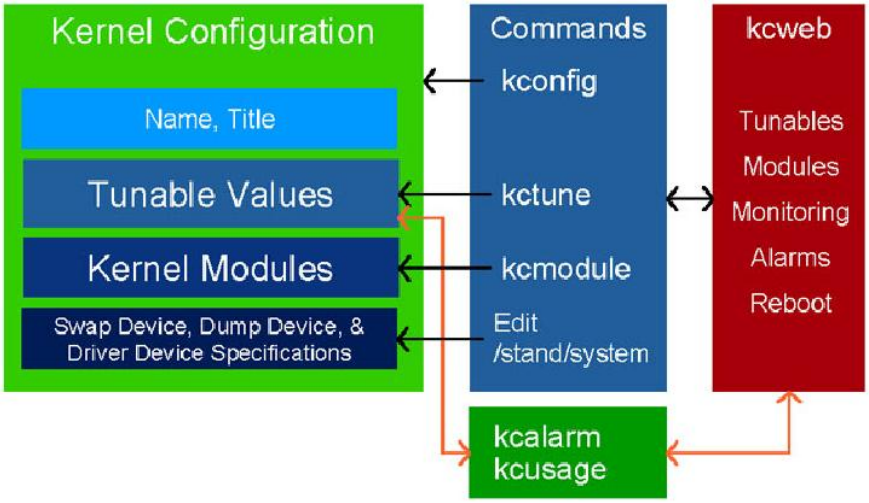
Kernel Configuration changes at UX 11.23 (11iv2) and 11.31 (11iv3)

General changes

HPUX 11.23 and 11.31 introduce some new features. The primary design goals were:

- No longer compile the kernel and reboot
- No longer copy kernel executables
- Instead you manage *Kernel Configurations*

The main feature is the kernel configuration (KC) itself. Logically, a KC is a collection of all the administrators' choices and settings needed to determine the behavior and capabilities of the HP-UX kernel. A kernel configuration includes:



Physically, a KC is a directory under `/stand` that contains the files needed to realize the specified behavior. The directory includes:

- An HP-UX kernel executable
- A set of HP-UX kernel module files
- A kernel registry database, containing all of the above settings
- A system file, describing the above settings in a human-readable form
- Various other implementation-specific files

The entire kernel configuration treated as a single unit that can be copied, backed up, selected by

name, etc. You do not recompile the kernel anymore but modify a KC.

New commands

There are three new primary commands to manage kernel configurations, `kconfig(1M)`, `kctune(1M)` and `kcmodule(1M)`.

kconfig(1M)

is used to manage whole kernel configurations. It allows configurations to be saved, loaded, copied, renamed, deleted, exported, imported, etc. It can also list existing saved configurations and give details about them.

```
# kconfig -v
Configuration backup
Title          Automatic Backup
Save Time      Tue Jul 22 13:09:53 2003
Modify Time    Tue Jul 22 13:09:53 2003
Kernel Path    /stand/backup/vmunix

Configuration last_install
Title          Created by last OS install
Save Time      Thu Jul  3 09:23:03 2003
Modify Time    Thu Jul  3 09:23:04 2003
Kernel Path    /stand/last_install/vmunix
```

kctune(1M)

is used to manage kernel tunable parameters. kctune will display or change the value of any tunable parameter in the currently running configuration or any saved configuration.

```
# kctune nproc=2000
       * The automatic 'backup' configuration has been updated.
       * The requested changes have been applied to the currently
         running system.
Tunable            Value  Expression  Changes
nproc    (before)   4200  Default     Immed
         (now)      2000  2000


# kctune nproc=default
       * The automatic 'backup' configuration has been updated.
       * The requested changes have been applied to the currently
         running system.
Tunable            Value  Expression  Changes
nproc    (before)   2000  2000        Immed
         (now)      4200  Default
```

kcmodule(1M)

is used to manage kernel modules. Kernel modules can be device drivers, kernel subsystems,

or other bodies of kernel code. Each module can be unused, statically bound into the main kernel executable, or dynamically loaded. kcmodule will display or change the state of any module in the currently running configuration or any saved configuration.

```
# kcmodule -v cdfs
Module                  cdfs  (0.1.0)
Description             CD File System
State                   auto (best state)
State at Next Boot      auto (best state)
Capable                 auto static loaded unused
Depends On              interface HPUX_11_23:1.0

# kcmodule -v c8xx
Module                  c8xx  [3EFFDFF0]
Description             SCSI C8xx Driver
State                   static (best state)
State at Next Boot      static (best state)
Capable                 static unused
Depends On              module wsio:0.0.0
                        interface HPUX_11_23:1.0

# kcmodule -v audio
Module                  audio  [3EFFDFF0]
Description             Audio Driver
State                   unused
State at Next Boot      unused
Capable                 static unused
Depends On              module wsio:0.0.0
                        module beep:0.0.0
                        interface HPUX_11_23:1.0
```

kclog(1M)

is used to view and search the contents of the kernel configuration log file.

kcpath(1M)

is used by scripts to determine the path of the running kernel.

kcusage(1M)

is used to query the usage of kernel resources controlled by various kernel tunables.

Deprecated/obsoleted commands

The commands `mk_kernel(1M)`, `kmtune(1M)` and `kmpath(1M)` present in previous HP-UX releases can still be used. They have been re-implemented as small shell scripts that invoke the new commands listed above. These commands will be removed in a future release.

`mk_kernel`	invokes `kconfig` (`-i`)
`system_prep`	invokes `kconfig` (`-e`)
`kmtune`	invokes `kctune`

`kmpath`	invokes `kcpath`
`config`	obsolete
`kmadmin`	obsolete
`kminstall`	obsolete
`kmmodreg`	obsolete
`kmsystem`	obsolete
`kmupdate`	obsolete

Overview of changes

HPUX 11.11 and before	**HPUX 11.23 & 11.31**
Use SAM to configure kernel	Use `kcweb` to configure the kernel
System files	Kernel configurations
Recompile and link the kernel	Reconfigure the kernel
Configure the kernel by editing `/stand/system` and run `mk kernel`	KC commands (`kconfig`) automatically update the running kernel or relink it, if necessary
Always reboot after kernel changes	Use KC commands or kcweb to tune the kernel; changes will be dynamic, if possible.
Use `cp` to make a backup kernel executable	Use `kconfig -c` to copy a kernel configuration
Copy vmunix to another system	Use `kconfig` to import a kernel configuration
Use `system_prep` to create a current system file	System file is kept up to date automatically. Use `kconfig –e` to create a system file from a saved configuration
Manage DLKMs with `kminstall`, `kmsystem`, `kmmodreg`, `kmupdate` and `config`.	Manage DLKMs using `kcmodule` or kcweb.
Header, master and space files in `/usr/conf`	Metadata built into modules

Examples of changes

	Task	HPU UX 11.11	HPUX 11.23 & 11.31
Tunables	List all tunables	`kmtune` (no options)	`kctune`
	List all tunables detailed	`kmtune -l`	`kctune -v`
	Query a tunable	`kmtune -q maxuprc`	`kctune maxuprc`
	Set a tunable to value	`kmtune -u -s maxuprc=512`	`kctune maxuprc=512`
	Increment tunable by value	`kmtune -u -s maxuprc+128`	`kctune maxuprc+=128`
	Set a value using a formula	`kmtune -u -s maxuprc=nproc/30`	`kctune maxuprc=nproc/30`
	Set tunable to default value	`kmtune -r maxuprc`	`kctune maxuprc=default`
	Set tunable to at least *n*	not possible	`kctune maxuprc>=512`
	Hold for next boot	`kmtune -s maxuprc=512`	`kctune -h maxuprc=512`
	List values held for next boot	`kmtune -d`	`kctune -D`
	Set a tunable in a saved configuration	not possible	`kctune -c WeekendConfig maxuprx=512`

	Create a user defined tunable	not possible	`kctune -u _mytunable=256`
Modules	Load a module	`kmadmin -L module`	`kcmodule module=loaded`
	Unload a module	`kmadmin -U module`	`kcmodule module=unused`
	Query a module	`kmadmin -Q module`	`kcmodule -v module`
	Print status of all modules	`kmadmin -s`	`kcmodule`
	Print detailed status of all modules	`kmadmin -S`	`kcmodule -v`
	Print all static modules	`kmadmin -k`	`kcmodule`
	Add, delete or update module	`kminstall`	no longer needed
	(Un-)register module with the running kernel	`kmmodreg`	no longer needed
	List all modules	`kmsystem` (no options)	`kmcmodule`
	Set module to configured and loadable.	`kmsystem -c y -l y module`	`kcmodule module=loaded`
	Set module to configured and not loadable.	`kmsystem -c y -l n module`	`kcmodule module=static`
	Set module to unconfigured.	`kmsystem -c n module`	`kcmodule module=unused`
	Query module state	`kmsystem -q module`	`kcmodule -v module`
	Configure a loadable module	`mk_kernel -M`	no longer needed
	Path to running kernel (`/stand/vmunix`)	`kmpath` (no options)	`kcpath -x`
	Name of running kernel (`vmunix`)	`kmpath -k`	`kcpath -b`
	Path to current kernel function set directory (`/stand/dlkm`)	`kmpath -c`	`kcpath -d`
	Mark kernel/configuration for next boot	`kmupdate kernel`	`kconfig -n configuration`

Procedure to modify the kernel configuration

If you like to modify only a few tunables or modules use `kctune` or `kcmodule` respectively:

```
# kctune nproc=1000
# kcmodule krm=loaded
```

If tunables/modules are dynamic you're done, else you need to reboot to take the new effect.

In order to perform many changes at once, e.g. apply tunable settings from another system do the following:

1) Export the current kernel configuration to a file:

```
# kconfig -e system_config
       * The current configuration (including any changes being held for
         next boot) has been exported to /root/system_config.
```

NOTE: The obsolete `system_prep` script has been rewritten to incoke `kconfig -e`

2) Now change the tunable values by editing this file:

```
# vi system_config
*
* Created on Thu Jul 31 11:08:37 2003
*
version 1
configuration nextboot "" [3f27e819]
*
* Module entries
*
module drmfglrx auto 0.1.0
module drmfgl auto 0.1.0
module gvid_him_rad auto 0.1.0
...
...
module lba best [3F0E6070]
module sba best [3F0E6070]
module root best [3F0E6070]
*
* Swap entries
*
*
* Dump entries
*
*
* Driver binding entries
*
*
* Tunables entries
*
tunable secure_sid_scripts 0x0
tunable nstrpty 60
tunable dbc_max_pct 20
tunable vx_ninode 30000
```

NOTE: Do not try to change modules. Tunables only.

3) Now import the modified configuration:

If you modified only dynamic tunables, then changes will be applied immediately:

```
# kconfig -i system_config
WARNING: The automatic 'backup' configuration currently contains the
         configuration that was in use before the last reboot of this
         system.
     ==> Do you wish to update it to contain the current configuration
         before making the requested change? y
       * The automatic 'backup' configuration has been updated.
       * system has been imported.  The changes have been applied to the
         currently running system.
```

If you modified at least one static tunable it takes a reboot to apply the changes. The changes

of dynamic tunables will also be held at next boot:

```
# kconfig -i system_config
       * The automatic 'backup' configuration has been updated.
NOTE:    The configuration being loaded contains changes that cannot be
         applied immediately:
      -- The tunable vx_ninode cannot be changed in a dynamic fashion.
NOTE:    The changes will be held for next boot.
       * system_config has been imported.  The changes will take effect
         at next boot.
```

Booting a kernel configuration

At the Boot Loader prompt enter:

```
HPUX> boot configname
```

Boot to failsafe mode (i.e. single user mode, hard coded tunables, no DLKMs):

```
HPUX> boot -tm [configname]
```

kcweb changes

HPUX 11.23 introduces the `waconf(1M)` command which allows to configures automatic startup of the web administration server used by kcweb (and pdweb) at boot time. `waconf` configures autostart by editing `/etc/inetd.conf` and `/etc/services`.

You can turn autostart on, off or query the current state using:

```
# waconf [-a on|off]
```

Example:

```
# waconf -a on
/etc/inetd.conf and /etc/services have been edited to allow
web administration tools to be autostarted using port 1110.

If NIS is in use you may need to edit /etc/nsswitch.conf or
change the configuration of the NIS server for the new webadmstart
service to work.

Network firewalls can also keep autostarting from working.

# waconf
Autostart is enabled

# grep web /etc/services
webadmstart        1110/tcp                # start web admin server
```

Kernel Tunables

Tunable Types

There are a few different **types** of kernel tunables:

Type	Description	Introduced
Static	Traditional tunable parameters	prior to HPUX 10.20
Dynamic	Can be changed without rebooting the system	in HPUX 11.11
Automatic	If an automatic tunable is set to default then the system determines an appropriate value at boot time. Automatic tunables are not changed automatically while the system is running!	in HPUX 11.23 and 11.31
User-Defined	Used like variables in other tunable formulas	in HPUX 11.23 and 11.31

For HPUX 11.31, a lot of new parameters are introduced and some parameters are obsolete. Here is the full list:

Tunable	Value	Expression	Changes
NSTREVENT	50	Default	
NSTRPUSH	16	Default	
NSTRSCHED	0	Default	
STRCTLSZ	1024	Default	
STRMSGSZ	0	Default	
acctresume	4	Default	
acctsuspend	2	Default	
aio_iosize_max	0	Default	Immed
aio_listio_max	256	Default	Immed
aio_max_ops	2048	Default	Immed
aio_monitor_run_sec	30	Default	Immed
aio_physmem_pct	10	Default	Immed
aio_prio_delta_max	20	Default	Immed
aio_proc_max	0	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
audit_memory_usage	5	Default	Immed
audit_track_paths	0	Default	Auto
base_pagesize	4	Default	
copy_on_write	1	Default	Immed
core_addshmem_read	0	Default	Immed
core_addshmem_write	0	Default	Immed
create_fastlinks	0	Default	
default_disk_ir	0	Default	

desfree_pct	0	Default	Immed
diskaudit_flush_interval	5	Default	Immed
dlpi_max_clones	3992	Default	Immed
dlpi_max_ub_promisc	1	Default	Immed
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
dump_concurrent_on	1	Default	Immed
executable_stack	0	Default	Immed
expanded_node_host_names	0	Default	Immed
fcache_fb_policy	0	Default	Immed
fcache_seqlimit_file	100	Default	Immed
fcache_seqlimit_scope	0	Default	Immed
fcache_seqlimit_system	100	Default	Immed
fcache_vhand_ctl	0	Default	Immed
fcd_disable_mgmt_lun	0	Default	Immed
fclp_ifc_disable_mgmt_lun	0	Default	Immed
fcoc_ifc_disable_mgmt_lun	0	Default	Immed
filecache_max	16287690752	Default	Auto
filecache_min	1628766208	Default	Auto
fr_rulecache	0	Default	Immed
fr_statemax	800000	Default	Immed
fr_tcpidletimeout	86400	Default	Immed
fr_tcptimewait	120	Default	Immed
frnat_tcptimewait	120	Default	Immed
fs_async	0	Default	
fs_symlinks	20	Default	Immed
ftable_hash_locks	64	Default	
gvid_no_claim_dev	0	Default	
hires_timeout_enable	0	Default	Immed
hp_hfs_mtra_enabled	1	Default	
intr_strobe_ics_pct	80	Default	Immed
io_ports_hash_locks	64	Default	
ipf_icmp6_passthru	0	Default	Immed
ipl_buffer_sz	8192	Default	Immed
ipl_logall	0	Default	Immed
ipl_suppress	1	Default	Immed
ipmi_watchdog_action	0	Default	Immed
ipnat_enable	1	Default	Immed
kmem_aggressive_caching	0	Default	Immed
ksi_alloc_max	33600	Default	Immed
ksi_send_max	32	Default	
lcpu_attr	0	0	Imm (auto disabled)
lotsfree_pct	0	Default	Immed
max_acct_file_size	2560000	Default	Immed
max_async_ports	4096	Default	Immed
max_mem_window	0	Default	Immed
max_thread_proc	3000	3000	Immed
maxdsiz	2147483648	2147483648	Immed
maxdsiz_64bit	2147483648	2147483648	Immed
maxfiles	2048	Default	
maxfiles_lim	4096	Default	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
mca_recovery_on	1	Default	Auto
mpas_readonly_text	0	Default	Immed
mprotect_reduce_protid_on	0	Default	Immed
msgmbs	8	Default	Immed
msgmnb	16384	Default	Immed
msgmni	512	Default	Immed
msgtql	1024	Default	Immed
ncdnode	150	Default	
nclist	8292	Default	
ncsize	8976	Default	
nflocks	4096	Default	Auto
nfs2_max_threads	8	Default	Immed
nfs2_nra	4	Default	Immed
nfs3_bsize	32768	Default	Immed
nfs3_do_readdirplus	1	Default	Immed
nfs3_jukebox_delay	1000	Default	Immed
nfs3_max_threads	8	Default	Immed
nfs3_max_transfer_size	1048576	Default	Immed
nfs3_max_transfer_size_cots	1048576	Default	Immed
nfs3_nra	4	Default	Immed
nfs4_bsize	32768	Default	Immed
nfs4_max_threads	8	Default	Immed
nfs4_max_transfer_size	1048576	Default	Immed
nfs4_max_transfer_size_cots	1048576	Default	Immed
nfs4_nra	4	Default	Immed
nfs_portmon	0	Default	Immed
ngroups_max	20	Default	Immed
ninode	8192	Default	
nkthread	8416	Default	Immed
nproc	4200	Default	Immed
npty	60	Default	
nstrpty	60	Default	
nstrtel	60	Default	
nswapdev	32	Default	
nswapfs	32	Default	
numa_mode	0	Default	
numa_policy	0	Default	Immed
numa_sched_launch	1	Default	Immed
override_umask	0	Default	Immed
pa_maxssiz_32bit	83648512	Default	Immed
pa_maxssiz_64bit	536870912	Default	Immed
pagezero_daemon_enabled	1	Default	Immed
patch_active_text	1	Default	Immed
pci_eh_enable	1	Default	
pci_error_tolerance_time	1440	Default	Immed
process_id_max	30000	Default	Auto
process_id_min	0	Default	Auto
pwr_idle_ctl	0	Default	Immed
remote_nfs_swap	0	Default	
rng_bitvals	9876543210	Default	
rng_sleeptime	2	Default	
rtsched_numpri	32	Default	
rusage_hires_enable	0	Default	Immed
sched_thread_affinity	6	Default	Immed
scroll_lines	100	Default	
secure_sid_scripts	0	0	Immed
semaem	16384	Default	

semmni	2048	Default	
semmns	10000	10000	
semmnu	256	Default	
semmsl	2048	Default	Immed
semume	100	Default	
semvmx	32767	Default	
shlib_debug_enable	0	Default	Immed
shmmax	1073741824	Default	Immed
shmmni	400	Default	Immed
shmseg	300	Default	Immed
streampipes	0	Default	
swchunk	2048	Default	
sysv_hash_locks	128	Default	
tcphashsz	0	Default	
timeslice	10	Default	
timezone	420	Default	
uname_eoverflow	1	Default	Immed
vnode_cd_hash_locks	128	Default	
vnode_hash_locks	128	Default	
vol_checkpt_default	10240	Default	
vol_dcm_replay_size	262144	Default	
vol_default_iodelay	50	Default	
vol_failfast_on_write	0	Default	
vol_fmr_logsz	4	Default	
vol_kmsg_gab_max_send	512	Default	Immed
vol_kmsg_max_receive_q_cnt	6144	Default	Immed
vol_kmsg_receive_q_high	8192	Default	Immed
vol_kmsg_receive_q_low	6144	Default	Immed
vol_max_bchain	32	Default	
vol_max_nconfigs	20	Default	
vol_max_nlogs	20	Default	
vol_max_nmpool_sz	16777216	Default	Immed
vol_max_prm_dgs	1024	Default	
vol_max_rdback_sz	67108864	Default	Immed
vol_max_vol	8388608	Default	
vol_max_wrspool_sz	16777216	Default	Immed
vol_maxio	2048	Default	
vol_maxioctl	32768	Default	
vol_maxkiocount	2048	Default	
vol_maxparallelio	256	Default	
vol_maxspecialio	256	Default	
vol_maxstablebufsize	256	Default	
vol_min_lowmem_sz	532480	Default	Immed
vol_mvr_maxround	256	Default	
vol_nm_hb_timeout	10	Default	
vol_rootdev_is_vol	0	Default	
vol_rvio_maxpool_sz	134217728	Default	Immed
vol_stats_enable	1	Default	Immed
vol_subdisk_num	4096	Default	
vol_swapdev_is_vol	0	Default	
vol_vvr_transport	1	Default	
vol_vvr_use_nat	0	Default	
vol_vxtrace	1	Default	
volcvm_cluster_size	16	Default	
volcvm_smartsync	1	Default	
voldco_max_updates	256	Default	
voldrl_max_drtregs	2048	Default	
voldrl_min_regionsz	512	Default	
voldrl_volumemax_drtregs	256	Default	Immed
voldrl_volumemax_drtregs_20	1024	Default	Immed

voliomem_chunk_size	65536	Default	
voliomem_maxpool_sz	134217728	Default	
voliot_errbuf_dflt	16384	Default	
voliot_iobuf_default	8192	Default	
voliot_iobuf_limit	131072	Default	
voliot_iobuf_max	65536	Default	
voliot_max_open	32	Default	
volpagemod_max_memsz	65536	Default	Immed
volraid_rsrtransmax	1	Default	
vps_ceiling	16	Default	Immed
vps_chatr_ceiling	1048576	Default	Immed
vps_pagesize	16	Default	Immed
vx_dexh_sz	524287	Default	Immed
vx_era_nthreads	5	Default	
vx_maxlink	32767	Default	
vx_ninode	0	Default	Immed
vxfs_bc_bufhwm	0	Default	Immed
vxfs_ifree_timelag	0	Default	Immed
vxtask_max_monitors	32	Default	

And the constraints of each parameter can be obtained by kctune –v. For example:

```
# kctune -v nproc
Tunable          nproc
Description      Maximum number of processes on the system
Module           pm_proc
Current Value              4200 [Default]
Value at Next Boot         4200 [Default]
Value at Last Boot         4200
Default Value              4200
Constraints      nproc >= 100
                 nproc <= 262144
                 nproc >= semmnu + 4
                 nproc >= maxuprc + 5
                 nproc <= nkthread - 100
Can Change       Immediately or at Next Boot
```

Refer to the man page of each parameter for further explanation. In addition, if a parameter is defined below the Failsafe value, then at the boot time, that parameter will be set automatically to the Failsafe value. For example:

```
NAME
     nproc - limits the number of processes allowed to exist simultaneously

VALUES
   Failsafe
     4200

   Default
     4200
```

Allowed values
100 - 60000

Overview of HP SMH for Kernel Configuration

You can configure and manage the kernel without remembering the syntax of the kernel configuration commands or the exact names of modules and tunables by using HP SMH, the web- and text-based HP-UX kernel configuration tool to configure and manage the kernel of your system. HP SMH has the following features:

- Web-based and text-based interfaces.
- Kernel tunable management: monitor and modify.
- Alarm management: add, modify and remove.
- Kernel module state management: modify.
- Access to manpages for tunables.
- Command preview – When a tunable, module or alarm is modified, you can use the command preview feature by choosing the **Preview** button. This will show the kernel configuration command invocation that will perform the requested task.

You can access Kernel Configuration in any of the following ways:

- From the command line with the kcweb -t command.
- With a web browser through the Kernel Configuration area of HP-UX System Management Homepage.

By default, the kcweb command invokes the Mozilla web browser. If you want to invoke kcweb with any other browser, set the BROWSER environment variable to the path name of the browser you wish to use. For more details, see the *kcweb*(1M) manpage.

Interpreting Module Information

Looking at the sample output in "Getting Information About Modules" below, you can see that each module has a name and a textual description. Each module also has a version, which typically looks like 1.0.

A kernel configuration can only use one version of any given module. However, multiple versions may be listed if, for example, your currently running system is using a different version of a module from the one that will be used at next boot. Version numbers are normally omitted from the short listing, but will be included if there's more than one version of a module.

For example:

```
# kcmodule -P ALL autofs cachefs
name autofs
desc Automounter File System
version 1.0
timestamp Tue Sep 12 21:53:28 2006 [45078EC8]
state static
cause best
next_state static
next_cause best
capable static unused
depend module nfswrp:0.0.0
depend interface HPUX_11_31_PERF:1.0
```

Each kernel module in the currently running configuration has a state, which describes how the module is being used. The possible states are:

unused	The module is installed on the system but not in use.
static	The module is statically bound into the kernel executable. This is the most common state. Moving a module into or out of this state requires relinking the kernel executable and rebooting.
loaded	The module is dynamically loaded into the kernel. Newer modules support this state. Such modules may be added to the kernel configuration or removed from it without rebooting.
auto	The module will be dynamically loaded into the kernel when it is first needed, but it hasn't been needed yet.

When kcmodule is giving information about the currently running system, and there are configuration changes being held for next boot, kcmodule will list both the current state and the state at next boot. For next boot, the same states are used, with complementary meanings:

unused	The module will not be used.
static	The module will be statically bound into the kernel executable.
loaded	The module will be dynamically loaded into the kernel during the boot process..
auto	The module will be dynamically loaded into the kernel when it is first needed after each boot

When kcmodule is giving information about a saved configuration, the same states are used. Next to each module state is a Cause that tells why the module is (or will be) in that state. The causes are:

explicit	The system administrator explicitly chose the state.
best	The system administrator chose to use the module, but didn't choose a specific state, so the module is in its best state as determined by the module developer.
auto	The module was in auto state, and was automatically loaded when something tried to use it.
required	The module was marked required by its developer.
depend	The module is in use because some other module in the configuration depends on it.

Different modules can support different states. Nearly all modules can be in static state, but only a few support loaded or auto states. Many modules can be in unused state, but required modules cannot. The Capable line in the output shows which states a module supports.

Modules often have dependencies between them. For example, device drivers typically cannot be configured into the kernel unless the driver support modules are also configured. Dependencies like this are shown on the Depends On lines in the output. A module can be dependent on a particular other module, specified by name and version. A module can also be dependent on an interface that must be supplied by some other module, without saying specifically which modules supply that interface. Modules that supply such interfaces have an Exports line in the output, listing the interfaces they export.

Interpreting Tunable Information

Looking at the sample output below, for example:

```
# kctune -P ALL nproc
name nproc
module pm_proc
desc Maximum number of processes on the system
defvalue 4200
bootvalue 4200
current 4200
next_boot 4200
expr Default
next_expr Default
min 100
max 131072
dynamic y
```

```
canauto n
default y
auto_default n
next_default y
signed n
flags 0x6c3
constraint nproc >= 100
constraint nproc <= 131072
174 Configuring the Kernel
constraint nproc >= semmnu + 4
constraint nproc >= maxuprc + 5
constraint nproc <= nkthread - 100
```

Use a comma-separated list with the -P option to display the categories you want.

```
# kctune -P name,current maxuprc nproc
name maxuprc
current 256
name nproc
current 4200
```

You can see that each tunable has a name and a textual description. Each tunable is associated with a kernel module whose name is listed in the verbose output (or in the table output if -g is specified). Tunables can be seen and changed only if they are associated with a module that is installed on the system (or are user-defined). The module does not have to be in use.

When displaying tunable information for the currently running system, kctune includes the current tunable value and the expression used to compute it. If changes to the tunable's value are being held for next boot, the next boot value and expression are also shown. Verbose listings also show the value the tunable had when the system was last booted. When displaying tunable information for a saved configuration, kctune displays only a current value.

Tunable values are computed integer expressions, which can refer to other tunable values. (Circular references are not permitted.) The value of a tunable could be 4200, or 0x400, or 12*1024, or 4*nproc+20. Values and expressions use the syntax of the C programming language. Therefore, numbers can be written in decimal (256), octal (01000), or hexadecimal (0x100). Expressions can use the following operators and symbols:

( ) ~ ! - + * / % << >> < <= > >= & ^ | == != && || ?:

A few tunables also support values specified as percentages, for example, 10%. White space is not permitted in any tunable expression. For backward compatibility, tunable names used in expressions can appear in all capitals, but this usage is discouraged and support for it will be removed in a future release.

All kernel tunables have a default value, which is chosen by the developer, and is shown in the verbose output. For some tunables, the default value is fixed and never changes. For other tunables, a new default value is chosen by the system at boot time. Still others can be automatically tuned, which means that the default value can change periodically while the

system is running, in response to changing system resources and needs. When a tunable is set to default, its expression is reported as Default, as seen in the examples above. In these cases, the system is free to choose the value it thinks optimal, and to change it as needed. HP recommends that tunables be left set to default unless the default is known to be unsatisfactory.

NOTE: Setting a tunable to Default is not the same thing as setting it explicitly to the default value reported by kctune. Using the example above, if you set nproc to 4200, its value will remain 4200 until you change it. However, if you set nproc to Default, its value will be kept up to date with any changes HP makes to the default value for nproc.

Some tunables have constraints on their values, which are shown in the verbose output. Sometimes these are minimum and/or maximum values, as shown for nproc above. Other times these are fixed relationships between tunables (for example, acctresume must be greater than acctsuspend) or restrictions on the allowed values (for example, dnlc_hash_locks must be a power of two). These constraints are enforced whenever tunable values are changed. There are other constraints, not shown by kctune, that are based on the current state of the system and can change over time (for example, nproc cannot be set to less than the number of processes currently running). These constraints are enforced only when changing the currently running system, and not when making changes held for use at next boot or changes to a saved configuration.

Some tunables have restrictions on when their values can be changed. These restrictions are noted in the kctune output. Tunables whose values can be changed immediately are marked Immed. Tunables whose values can be automatically tuned by the system are marked Auto. Tunables without either marking can only be changed with a reboot.

All HP-UX tunables have manpages. To obtain information about the behavior, allowed values, and side effects of a tunable, consult the manpage for that tunable, which can be found in Section 5 of the online document.

Additional information

Reconfiguring the Kernel in HP-UX 11iv1
http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01918628/c01918628.pdf

HP-UX System Administrator's Guide: Configuration Management
HP-UX 11i Version 3
http://bizsupport1.austin.hp.com/bc/docs/support/SupportManual/c02281490/c02281490.pdf

Kernel parameters for HP-UX systems running Oracle E-Business Suite software applications

http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA1-6355ENW.pdf