Linux System Administration-I
CSE-4043

Chapter 13 : Drivers and the Kernel

NIBEDITA JAGADEV

Department of CSE

Asst. Professor

SOA Deemed to be University, Bhubaneswar, Odisha , India

nibeditajagadev@soa.ac.in

Contents

- Drivers and the kernel
- Kernel Adaptation
- Kernel types
- Drivers and Device Files
- Device File Creation
- Device File and Device Numbers
- Naming Convention for Devices
- Custom Kernel VS Loadable Modules

Drivers and the Kernel

- ***UNIX system has three layers:***
  - The hardware
  - The operating system kernel
  - The user-level programs
- Kernel
  - Kernel hides the hardware, provides a abstract, high-level programming interface
    - Process
    - Signal and semaphores
    - Virtual memory
    - The filesystem
    - Interprocess communication
  - Kernel contains device drivers that manage its interaction with specific pieces of hardware.
  - Size of kernel

Kernel Adaptation

- Monolithic Kernel
- Micro Kernel

Kernel types

- Provide the kernel with explicit information about the hardware
- Depend on kernel prospects for devices on their own
  - Solaris: almost completely modular kernel
  - HP-UX: supports a relatively small and well-defined hardware base
  - FreeBSD: must be told explicitly at kernel compilation time about device
  - Linux: limited module support

Kernel types

- Kernel build directory and location by system

System	Build Directory	Kernel
Solaris		/kernel/unix
HP-UX	/stand	/stand/vmunix
Linux	/usr/src/linux	/vmlinuz or /boot/vmlinuz
FreeBSD	/usr/src/sys	/kernel

Drivers and Device Files

- A device driver is a program that manages the system's interaction with a particular type of hardware
- Device files and Device numbers
- -Major Devices
- -Minor devices

Device File Creation

- mknod filename type major minor
- File name- device file to be created
- Type-different types of device
- Major, Minor- device numbers

Device File and Device Numbers

- Two types of device files
- Character device file
- Block device file

Naming Convention for Devices

- Naming convention is random
- Character device name is prefaced with letter ‘r’
- Serial device files are usually named tty followed by sequence of letters.

Custom Kernel VS Loadable Modules

- System comes with a generic kernel
- Some drivers may also be dynamically inserted into running kernel
- Usually dynamic module approach is more preferable than the custom kernel

Linux System Administration-I
CSE-4043

Chapter 13 : Drivers and the Kernel

NIBEDITA JAGADEV

Department of CSE

Asst. Professor

SOA Deemed to be University, Bhubaneswar, Odisha , India

nibeditajagadev@soa.ac.in

Contents

- Why Configure the kernel
- Configuring a Solaris Kernel
- Accessing New Device Drivers
- Building an HP-UX kernel

Why Configure the kernel

- Generic kernel
  - Many device drivers
  - Optional packages
- Tailor the kernel for your need
  - Less memory
  - Well-tuned configuration
- Add support for new types of device
  - Some system is simple
  - Some may need to several steps.
- Read this:
  - Building the kernel is not difficult; its just difficult to fix when you break it.
  - Get a good reference book for your OS

Configuring a Solaris Kernel

- Structured around loadable modules
  - Modules are stored in subdirectories of /kernel
- Current system configuration
  - Display loaded modules:
    - modinfo [-c]
  - Show hardware configurations:
    - prtconf
  - Display loadable modules, hardware configuration and some tunable kernel:
    - Sysdef

Configuring a Solaris Kernel

- Automatic configuration
  - Probe device and initialize a driver for each device
- The Solaris kernel area
  - /kernel
    - Modules common to machines that share an instruction set
  - /platform/platform-name/kernel
    - Modules specific to one class of hardware
    - Platform example: ultra1, SUN-blade-100
    - Command: uname -i
  - /platform/hardware-class-name/kernel
    - Modules specific to one class of hardware
    - For example: sun3u, sun4u
    - Command: uname –m
  - /usr/kernel

Configuring a Solaris Kernel

– Subdirs under each kernel dir:

drv	loadable modules for device drivers Note: you can specify the device specific configuration parameters here.
misc	Loadable object files for misc kernel routines
cpu	CPU-specific module for the UltraSPARC
strmod	STREAMS modules
sched	Operating system schedulers
sys	loadable system calls
fs	Filesystem-related kernel modules

Configuring a Solaris Kernel

- /etc/system
  - Master configuration file
    - Take a look at your /etc/system in the lab
      - Long dev name comes from the link
      - Force to load certain module
      - Set the search path
      - Non default values for variables can be added with set command
        - » Example: set maxusers=40
  - Consulted at boot time
    - Specify another /etc/system with boot –a
    - Or boot with /dev/null if no good copy
      - You will be prompted to answer questions

Accessing New Device Drivers

- Solaris drivers are usually distributed as a package
  - Run pkgadd
- Should you add new device drivers they should be installed in **/kernel**. You can
  - add drivers with the **add_drv** command, then run **drvconfig**
  - remove them with the **rm_drv** command.
  - Once the driver is installed and the new device connected reboot the system with:

    Ok boot -r

Accessing New Device Drivers

- Alternatively, you can create the file /reconfigure before rebooting. The kernel will then be reconfigured during the boot process.

```
# touch /reconfigure
# reboot
```

- One of these procedures is required for all drivers not installed initially. It causes the kernel to properly recognize the new drivers during the boot process.

Building an HP-UX kernel

- Use SAM for the first few time
- Kernel configuration file /stand/system
- Run mk_kernel to rebuild the kernel
- Copy the new kernel in place to /stand/vmunix

Linux System Administration-I
CSE-4043

Chapter 13 : Drivers and the Kernel

NIBEDITA JAGADEV

Department of CSE

Asst. Professor

SOA Deemed to be University, Bhubaneswar, Odisha , India

nibeditajagadev@soa.ac.in

Contents

- Building an Tru64Unix kernel
- Configuring a Linux kernel
- Loadable kernel modules in Linux
- Summary

Building an Tru64Unix kernel

- Digital UNIX recommends that you be in single user mode when building the kernel.
  - cp /vmunix /vmunix.save
    - save the old kernel
  - cp /genunix /vmunix
    - install the generic kernel to be the running kernel
  - /usr/sbin/shutdown -r +5
    - reboot the system
  - /usr/sbin/shutdown +1
    - Log on as root and take the system down to single user mode
  - mount /usr
    - remount the /usr file system

Building an Tru64Unix kernel

- /usr/sbin/doconfig –
  - you will be prompted for system configuration information. If you need to edit the resulting configuration file answer "yes" at the prompt. The new kernel will then be built and the path to it will be displayed.
- mv /sys/{hostname}/vmunix /vmunix
  - move the kernel from the path displayed in the step above to the root directory
- /usr/sbin/shutdown -r now
  - reboot the system
- If the system fails to boot you can reboot to single user mode using the generic kernel (/genunix) and try again.

- The master configure file is /etc/sysconfigtab

Configuring a Linux kernel

- Save the current kernel
  - #cp /vmlinuz /safe
- Save the current kernel source if applys
  - #cd /usr/src; tar zcvf old-tree.tar.gz
- Unpack the source code and apply the patches
  - #cd /usr/src
  - # tar xvfz XXX.gz
  - # zcat YYY??.gz | patch –p0

Configuring a Linux kernel

- Use tool
  - make menuconfig
  - make xconfig
  - make config
- Building the linux kernel binary
  - Run make xconfig/menuconfig/config
  - Run make dep
  - Run make clean
  - Run make bzImage
  - Run make modules
  - Run make modules_install
  - Copy image file to /boot/vmlinuz
  - Config lilo/grub about the new kernel

Configuring a Linux kernel

- /proc
  - Allow to view and set kernel options at run time
  - Are not remembered across reboots

Loadable kernel modules in Linux

- Check the currently loaded module
  - Run lsmod
- Insert module
  - Run insmod/modprode
  - Can only be removed explicitly
- Remove a module
  - Run rmmod

Summary

- Again, what we covered here does NOT intend to give you exact step by step instructions for a particular OS on a particular platform
  - The basic flow/idea remains same for a relatively long period of time for particular vendor.
  - Things change from one release to another and vendor to vendor
  - Check the vendor's latest documents for:
    - Release notes
    - New feature updates
    - Installation and upgrade documents
    - System administration documents
    - … …

Thank You