Drivers and Kernel

Abstraction layers of Unix

- Hardware
- Kernel
- User-level programs

Kernel

- Hides system hardware to present an abstract programming interface in the form of system calls and library functions
- Provides abstraction and management for the following concepts
  - Processes
  - Resources
  - Virtual memory
  - Filesystem
  - Interprocess communications
- Provides devices drivers to manage interaction with specific devices
- Written mostly in C, with a little assembly language for low-level processing

Kernel types

- Solaris
  - Completely modular kernel
  - Can load device drivers as they are needed
  - Discovery of a new device automatically results in search and load of corresponding module
- BSD
  - Must be told at compilation time about devices on the system
  - You may have to take apart your system to discover some of the devices such as chipset on the ethernet card
- Linux
  - Cross between Solaris and BSD
  - Must know about all hardware beforehand resulting in one large kernel
  - Can be configured to load drivers only when needed
  - Limitations on modularity dictated by PC hardware

Configuring kernel

- Generic kernel contain too much extraneous drivers
- Kernel reconfigured to get rid of unused modules and options

  - System optimally tuned for the work
  - Unused modules may still use memory if not removed
- Add support for new types of devices
  - May have to modify configuration files, add new device, and rebuild the kernel from scratch
  - Another option may be to run a program to change configuration
- Loadable device driver
  - New code can be loaded into the kernel while it is running

Configuring Solaris kernel

- Solaris kernel probes the machine for devices at boot time and initializes a driver for each device found
- Extensive use of loadable modules
- Solaris kernel area
  - Rigid kernel directory structure
    - `/kernel` – modules common to machines that share an instruction set
    - `/platform/`*platform-name*`/kernel` – modules specific to one type of machine, such as `Ultra-5_10`
    - `/platform/`*hardware-class-name*`/kernel` – modules specific to one class of hardware, such as `sun4u`
    - `/usr/kernel` – similar to `/kernel`
  - Determine platform specific information using the `uname` command
  - `genunix`
    - Generic kernel for platform-independent portion of base kernel
    - Search path at boot time

      `/platform/`*platform-name*`/kernel:/kernel:/usr/kernel`
  - Each kernel directory contain several subdirectories
    - `cpu` – CPU-specific module for UltraSPARC
    - `drv` – Loadable object files for device drivers
    - `exec` – Modules to decode executable file formats
    - `fs` – Filesystem related kernel modules
    - `genunix` – Generic platform-independent kernel
    - `misc` – Loadable object files for miscellaneous kernel routines
    - `sched` – OS schedulers
    - `sparcv9` – 64-bit kernel
    - `strmod` – STREAMS modules
    - `sys` – Loadable system calls
    - `unix` – Base platform-specific kernel
  - You almost never have to change any file in the kernel directories
- Configuring the kernel with `/etc/system`
  - `/etc/system` is the master kernel configuration file
  - Used to customize the operation of OS kernel
    - Read only once, at boot time
  - You must preserve the original system file before modifying it

    - `boot -a` allows the specification of a backup copy to boot
  - Contains commands read by kernel during initialization and used to customize the operation of the system
  - Commands modify the treatment of loadable kernel modules
  - Syntax of file
    - List of keyword/value pairs
    - Comments can begin with an asterisk and end with newline character
    - Commands are case-insensitive, and limited to 80 characters
    - Commands that modify the system's operation with respect to loadable kernel modules require the specification of module type by listing the module's namespace as specified by kernel directory names above
    - Different `exec` modules provided include `aoutexec` (`coffexec` for x86 systems), `elfexec`, and `intpexec`
    - Commands are:
      - `exclude` – Modules that should not be loaded; every `exclude` entry in the file is added to the list to be excluded
      - `include` – Module to be loaded; default for the system;
      - `forceload` – Modules to be loaded during kernel initialization; default action is to load kernel module when its services are first accessed
      - `rootdev` – Location of root partition; to avoid using the default
      - `rootfs` – Root filesystem type
      - `moddir` – Search path for loadable kernel modules; multiple directories can be listed, delimited by blank spaces or colons
      - `set` – Sets kernel tuning variables
        `set [<module>:]<symbol> {=, !, &} [~][-]<value>`
        Set an integer or character pointer in kernel or kernel module to a new value
        Integer variables are modified by simple assignment, inclusive bitwise OR, bitwise AND, one's complement, and negation
        Values can be specified as hex, octal, or decimal
        Variables can be listed by the command `sysdef`

- Debugging a Solaris configuration
  - `prtconf`
    - Prints machine's general configuration
    - Output includes machine type, model number, amount of memory, and configuration of system peripherals formatted as a device tree
  - `sysdef`
    - Current system definition in tabular form
    - Lists all hardware devices, pseudo devices, system devices, loadable modules, and values of selected kernel tunable parameters
    - Generates the output by analyzing the named bootable OS file *namelist* (default: `/dev/kmem`) and extracting configuration information from it
  - `modinfo`
    - Displays information about loaded kernel modules
    - Format contains following entries (in order)
      `Id` – Module ID
      `Loadaddr` – Starting text address in hex
      `Size` – Size of text, data, and BSS in hex bytes
      `Info` – Module specific information
        - Block and character major numbers for drivers
        - System call number for system calls

· Index into appropriate kernel table for other module types

`Rev` – Revision of loadable modules system

`Module Name` – Filename and description of module

Adding device drivers

- Device driver
  - Program to manage system's interaction with a hardware device
  - Interface between hardware commands understood by device and programming style understood by kernel
    - Present a standard programming interface to the kernel
  - Part of the kernel but accessible from both kernel and user space
    - User level access provided through a special device file in `/dev` directory
    - Kernel translates operations on device files into calls to device driver code
  - New hardware devices contribute to chaos, particularly for Linux
    - Normally, device drivers for Linux lag behind those for Windows
- Device numbers
  - Files in `/dev` directory have major and minor device numbers associated with them
  - Kernel uses these numbers to map references to device file to corresponding driver
  - Major device number
    - Identifies the driver associated with the file (device type)
  - Minor device number
    - Also called *unit number*
    - Identifies the instance of the device
    - Used to select a partition on disk
    - May be used to select device characteristics such as tape density
  - Solaris uses a symbolic link in `/dev` to point to actual disk; you have to specify the absolute path to device file to get major and minor numbers for a device
    - Use `ls -l` to see the major and minor device numbers
  - Block-special device and character-special device
    - Some block-special devices can be addressed as character-special devices as well, such as disk and tape
    - Character-special devices like terminal and printer cannot be accessed as block-special devices
  - Pseudo devices
    - Phantom devices to provide abstraction as a device driver
    - Used to provide an interface like a terminal at pseudo TTY
  - Kernel catches the file operation meant for a device and transfers control to appropriate function in a table
    - Unusual operations (eject floppy) are performed by using the `ioctl` system call to send a message directly to the device
- Adding a Solaris device driver
  - Easy to add drivers as they are distributed as loadable kernel modules
  - Use `pkgadd` command to automatically add device driver to system
  - The object file (extension `.o`) and configuration file (extension `.conf`) are added to directory `/platform/sun4u/kernel/d`
    - Specific device parameters specified in the configuration file

  - Module loaded into running kernel with the command `add_drv`
- Adding a Linux device driver
  - Driver can be distributed as
    * A patch against a specific kernel version
    * A loadable module
    * Installation script to apply appropriate patches
  - Patching against specified kernel version
    * Most common way to add drivers
    * Apply as follows:

```
cd /usr/src/linux; patch -p1 < driver.diff
```

Device files

- Kept in the directory `/dev`
- Large systems like Solaris use separate directories for different types of devices
- Created by the `mknod(1M)` command or `mknod(2)` system call
- Check Section 7 of man pages for existing device drivers
  - See the section intro for all the device types available
  - To check for pseudo tty's, use the following command

```
man -s 7D pty
```

Naming conventions for devices

- Block devices may also have character identities
  - Convention to prefix the device name/directory by `r`
  - `/dev/dsk/c0t0d0s0` for block-special device and `/dev/rdsk/c0t0d0s0` for its character-special reincarnation
- Serial device files typically named `tty`, with the pseudo devices named `pty`

Loadable kernel modules

- Supported on all modern systems, including Solaris, Linux, and FreeBSD
- Allows a service to be added/removed in a running kernel
  - No need to modify the kernel
  - Smaller footprint for kernel because drivers are loaded as needed
- Implemented by providing a hook into the kernel
  - A user level command tells kernel to load new modules into memory
  - Untested module may cause kernel panic
- Loadable kernel modules in Solaris

  - Virtually, everything is loadable module
  - Use the `modinfo` command to see currently loaded modules
  - Easy for third parties to write drivers that seamlessly integrate into kernel
  - Kernel is informed about newly added Drivers by using the command `add_drv`
    * Loads drivers into kernel and makes appropriate device links
    * Drivers are removed with the command `rem_drv`
    * Run the command `drvconfig` to reconfigure the `/devices` directory and add appropriate files for newly loaded driver
  - Loadable modules not accessed through device files are added and removed through the commands `modload` and `modunload`

- Loadable kernel modules in Linux
  - Everything, except root filesystem type and device on which root filesystem resides, can be built as a loadable kernel module
  - Stored under `/lib/modules/`*version* where version is Linux kernel version (found by `uname -r`)
  - Currently loaded modules found by `lsmod` command
  - Manually added into kernel by the command `insmod` and removed using the command `rmmod`
    * `rmmod` works only if the number of current references to module, as returned by `lsmod`, is 0
  - Loaded semiautomatically by the command `modprobe`, using `/etc/modules.conf` to determine how to handle each module
  - Also loaded and unloaded dynamically by `kerneld`