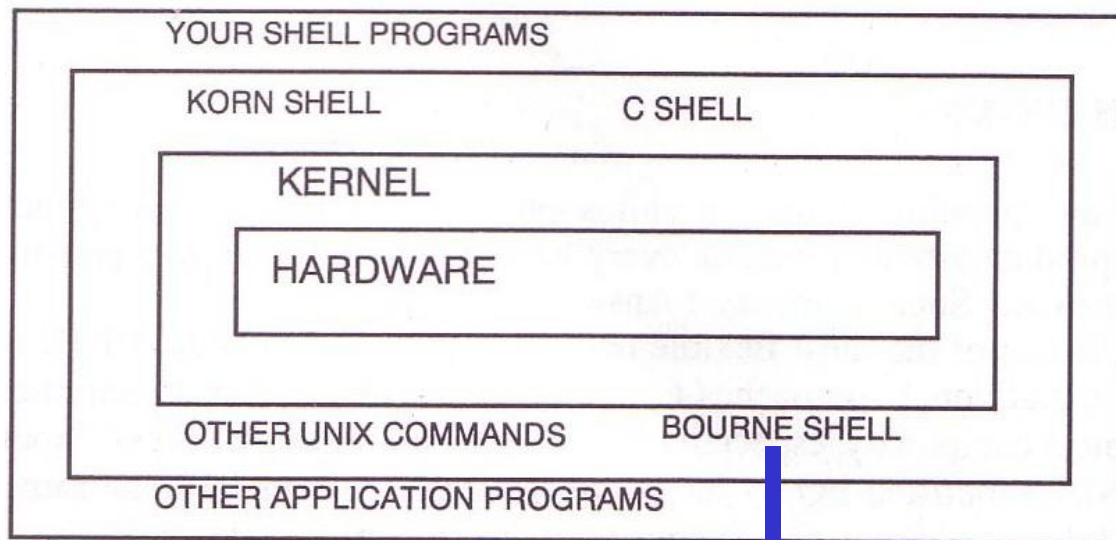


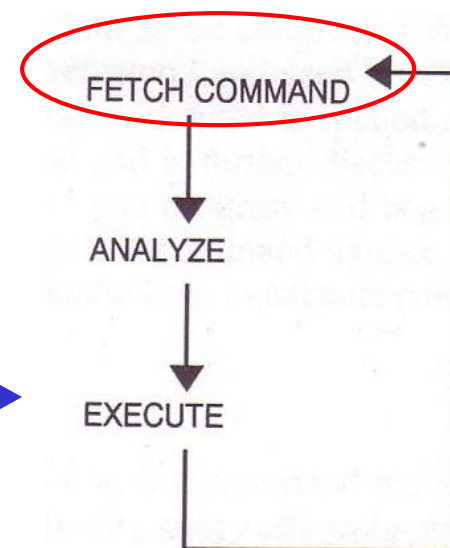


Drivers and the Kernel

Introduction – UNIX Kernel and Shell



interpret



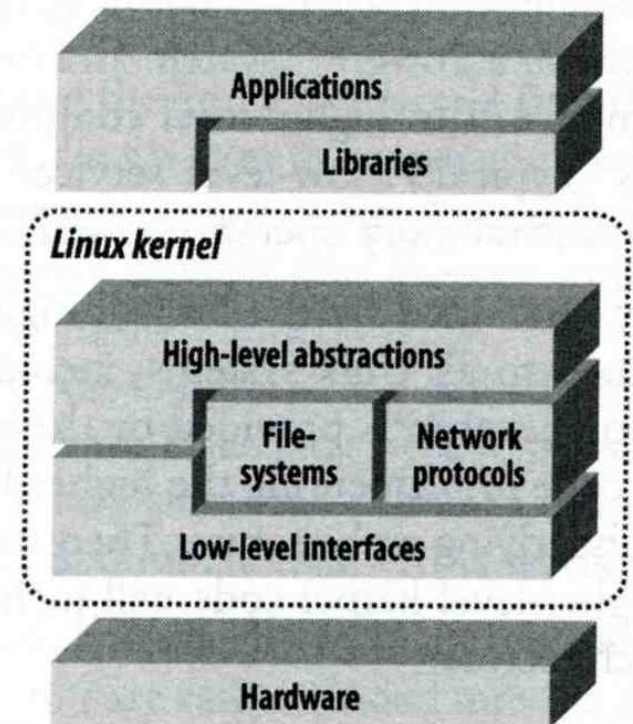
Roles of Kernel

❑ Components of a UNIX System

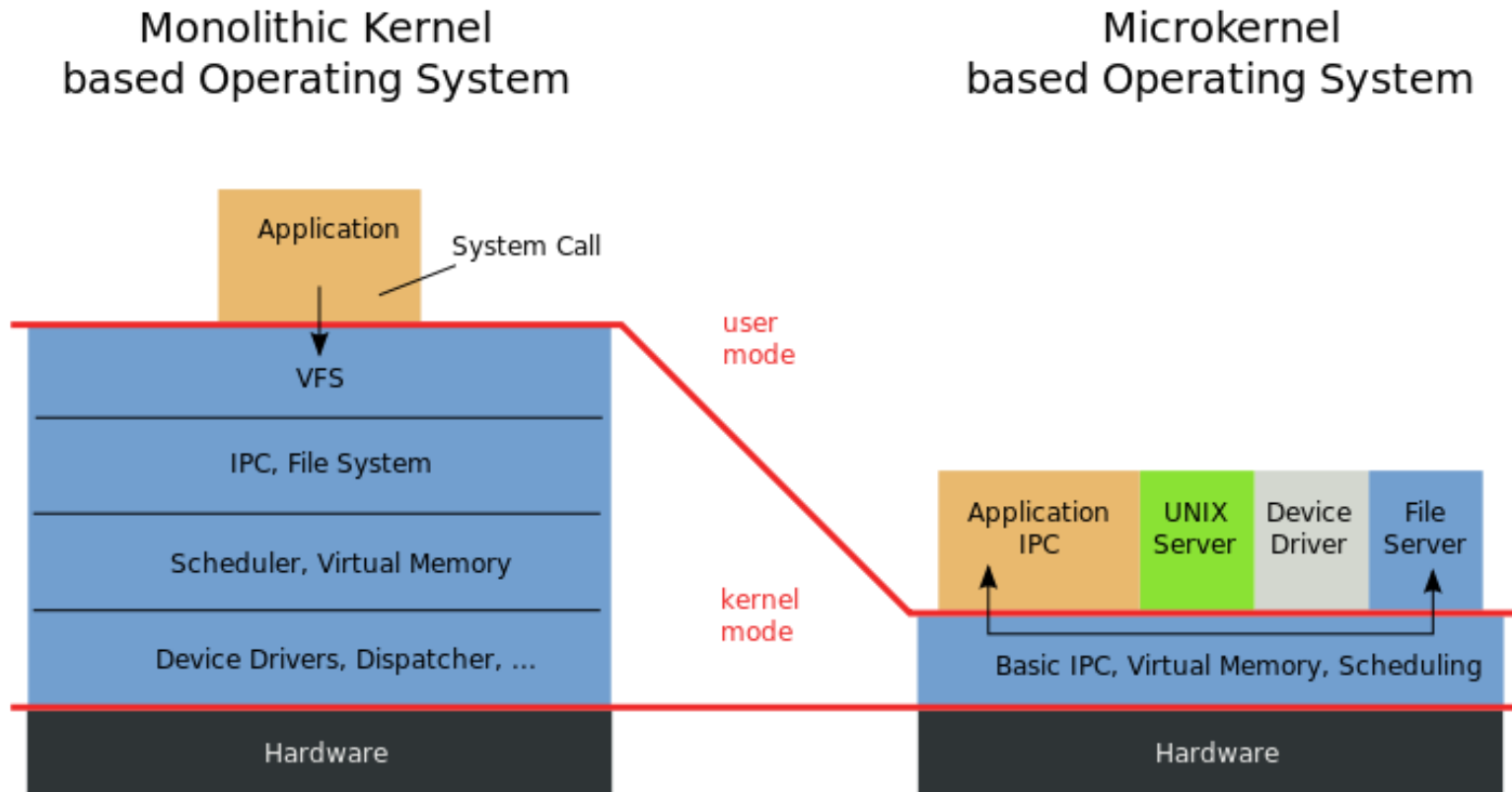
- User-level programs
- Kernel
- Hardware

❑ Two roles of kernel (OS)

- **High-level abstractions**
 - Process managements
 - Time sharing, memory protect
 - File system management
 - Memory management
 - I/O management
- Low-level interface
 - drivers



Kernel Types



Kernel Types

Concept of being modulized ...
only provides essential functionalities;
Put other sophisticated functions into user level
e.g., I/O management in the user level

- ❑ Two extreme types
 - **Microkernel**
 - Provide only necessarily, compact and small functionalities
 - Other functions is **added via well-defined interfaces**
 - **Monolithic kernel (龐大的kernel – e.g., UNIX)**
 - Whole functionalities in one kernel
- ❑ Modern OS
 - Solaris
 - Completely modular kernel
 - Load necessary module when it is needed
 - BSD/Linux-derived system
 - Much of the kernel's functionality is contained in modules

More integrated...



Monolithic kernel developing towards micro kernel (being more modulized),
but without IPC (message passing) problem

Kernel related directory

❑ Build directory and location

System	Build Directory	Kernel file
FreeBSD	/usr/src/sys	/kernel (< 4.x) /boot/kernel/kernel (> 5.x)
Red Hat	/usr/src/linux	/vmlinuz or /boot/vmlinuz
Solaris	-	/kernel/unix
SunOS	/usr/kvm/sys	/vmunix

Why configure the kernel?

Generic: with various devices...,
functions supported

- ☐ The native kernel is often big and common
- ☐ Tailoring kernel to match site situation
 - Purge unnecessary kernel devices and options
 - Add functionalities that you want
- ☐ OS patch
 - Remedy security hole of kernel implementation
- ☐ Fine-tune system performance
 - Such as adjusting important system parameters
- ☐ Adding device drivers
- ☐ Fast boot time
- ☐ Lower memory usage

Building a FreeBSD Kernel

- ❑ Kernel source
 - /usr/src/sys
- ❑ Kernel configuration file
 - /usr/src/sys/<ARCH>/conf
 - GENERIC, LINT (< 4.X) <ARCH> represents one of i386, amd64, ia64, powerpc, sparc64
 - GENERIC, "make LINT" under this dir (> 5.x) LINT file: lists all options
- ❑ Steps to build a new kernel
 - Edit /usr/src/sys/<ARCH>/conf/<KERNCONF>
 - For example, save a configuration file named as SABSD
 - % cd /usr/src ;
 - % make buildkernel KERNCONF=SABSD
 - % make installkernel KERNCONF=SABSD

<https://www.freebsd.org/doc/en/books/handbook/kernelconfig-building.html>

To Build a FreeBSD Kernel...

- ☐ What to Choose?
- ☐ What to Load?
- ☐ Option Settings?
- ☐ Device Drivers?

Finding the system hardware (1)

Listing devices from M\$ windows

❑ Before venturing into kernel configuration

- Get an inventory of the machine's hardware
- Microsoft's **Device Manager**

❑ dmesg

Listing devices from dmesg

- `cat /var/run/dmesg.boot`

```
psm0: <PS/2 Mouse> irq 12 on atkbd0  
psm0: [GIANT-LOCKED]  
psm0: [ITHREAD] psm0: model Generic PS/2 mouse, device ID 0
```

Finding the system hardware (2)

❑ pciconf

- pciconf -l

```
ath0@pci0:3:0:0: class=0x020000 card=0x058a1014 chip=0x1014168c  
vendor = 'Atheros Communications Inc.'  
device = 'AR5212 Atheros AR5212 802.11abg wireless'  
class = network subclass = ethernet
```

May not support by GENERIC...

Finding the system hardware (3)

❑ pciconf & man page

- `man -k Atheros`
 - Find drivers from company name
- `pciconf -l & man`
 - List all attached devices

```
ehci1@pci0:0:29:7:   class=0x0c0320 card=0x3a3a8086 chip=0x3a3a8086 rev=0x00 hdr=0x00
pcib10@pci0:0:30:0:  class=0x060401 card=0x244e8086 chip=0x244e8086 rev=0x90 hdr=0x01
isab0@pci0:0:31:0:   class=0x060100 card=0x3a168086 chip=0x3a168086 rev=0x00 hdr=0x00
ahci0@pci0:0:31:2:   class=0x010601 card=0x3a228086 chip=0x3a228086 rev=0x00 hdr=0x00
none8@pci0:0:31:3:   class=0x0c0500 card=0x3a308086 chip=0x3a308086 rev=0x00 hdr=0x00
em0@pci0:3:0:0:      class=0x020000 card=0x00008086 chip=0x10d38086 rev=0x00 hdr=0x00
em1@pci0:2:0:0:      class=0x020000 card=0x00008086 chip=0x10d38086 rev=0x00 hdr=0x00
```

➤ `man [device]`

– `man em`

```
EM(4)                                FreeBSD Kernel Interfaces Manual                                EM(4)

NAME
  em - Intel(R) PRO/1000 Gigabit Ethernet adapter driver
```

Finding the system hardware (4)

❑ Man page for devices

- `man [device]`

NAME

`em` – Intel(R) PRO/1000 Gigabit Ethernet adapter driver

SYNOPSIS

To compile this driver into the kernel, place the following line in your kernel configuration file:

```
device em
```

Alternatively, to load the driver as a module at boot time, place the following line in `loader.conf(5)`:

```
if_em_load="YES"
```

Building a FreeBSD Kernel – Configuration file

The explanations on options and devices...

❑ Each line is a control phrase

- Keyword + arguments

Keyword	Function	Example
machine	Sets the machine type	i386 or amd64
cpu	Sets the CPU type	I586_CPU or HAMMER
ident	Sets the name of the kernel	SABSD
maxusers	Sets the kernel's table sizes	0
options	Sets various compile-time options	INET or INET6
device	Declares devices	fxp or em

```

cpu      I486_CPU
cpu      I586_CPU
cpu      I686_CPU
ident    GENERIC
options  SCHED_ULE      # ULE scheduler
options  PREEMPTION     # Enable kernel thread preemption
options  INET            # InterNETworking
device   em
```

i386/conf/GENERIC

<https://www.freebsd.org/doc/en/books/handbook/kernelconfig-config.html>

Kernel backup

Your last chance to prevent module missing...to survive!!

❑ Kernel file locations

Old kernel is automatically moved to kernel.old when you're making the new kernel

- Put in the /boot directory
- /boot/GENERIC/kernel, /boot/kernel.old/kernel
- /kernel.GENERIC, /kernel.old (Freebsd 4.x)

Or just simply cp your GENERIC /boot/kernel first!

❑ If something goes wrong

- **ok mode !**
 - unload kernel; load kernel.old/kernel
 - load kernel modules
- mv /boot/kernel /boot/kernel.bad

Ok mode

```

Welcome to FreeBSD

1. Boot Multi User [Enter]
2. Boot Single User
3. Escape to loader prompt
4. Reboot

Options:
5. Kernel: default/kernel (1 of 2)
6. Configure Boot Options...

```

```

Type '?' for a list of commands, 'help' for more detailed help.
OK unload kernel
OK load /boot/kernel.old/kernel
/boot/kernel.old/kernel text=0x34a274 data=0x40df4+0x72d84 syms=[0x4+0x483e0+0x4
+0x64b7e]
OK _

```

Or “enable modules” in the ok mode..

Tuning the FreeBSD Kernel

❑ sysctl command

- Dynamically set or get kernel parameters
- All changes made by sysctl will be lost across reboot
- Use sysctl to tune the kernel and test it, then recompile the kernel

The other way is to write your settings into `/etc/sysctl.conf...`

- Format:

```
% sysctl [options] name[=value] ...
```

Ex:

```
% sysctl -a          list all kernel variables
```

```
% sysctl -d kern.maxfiles    print the description of the variable
```

```
% sysctl kern.maxfiles      print the value of the variable
```

```
% sudo sysctl kern.maxfiles=2048
```

Kernel modules

❑ Kernel module location

- /boot/kernel/*.ko
- /modules (FreeBSD 4.x)

❑ zfs[/boot/kernel] -chiahung- kldstat

Id	Refs	Address	Size	Name
1	15	0xc0400000	4abd60	kernel
2	1	0xc08ac000	13b0fc	zfs.ko
3	2	0xc09e8000	3d5c	opensolaris.ko
4	2	0xc09ec000	16b84	krpc.ko
5	1	0xc0a03000	8c48	if_le.ko

❑ Load/unload kernel modules

- kldload(8), kldunload(8)
 - E.g., kldload if_fxp

Procedure of Loading a Device Module

❑ Loading a device module

1. pciconf -l for a device
2. man vendor name for module name in BSD
3. grep the name in /boot/kernel/*.ko
4. kldload [module name]
5. Setup permanently by
 - a) **Recompile the kernel** or
 - b) **Add [module name]_enable="YES" in /boot/loader.conf**

Reference

- ❑ <http://www.freebsd.org/doc/en/books/handbook/kernelconfig-config.html>
- ❑ /usr/src/sys/<ARCH>/conf
 - NOTES → machine dependent kernel configuration notes.
 - LINT
 - GENERIC