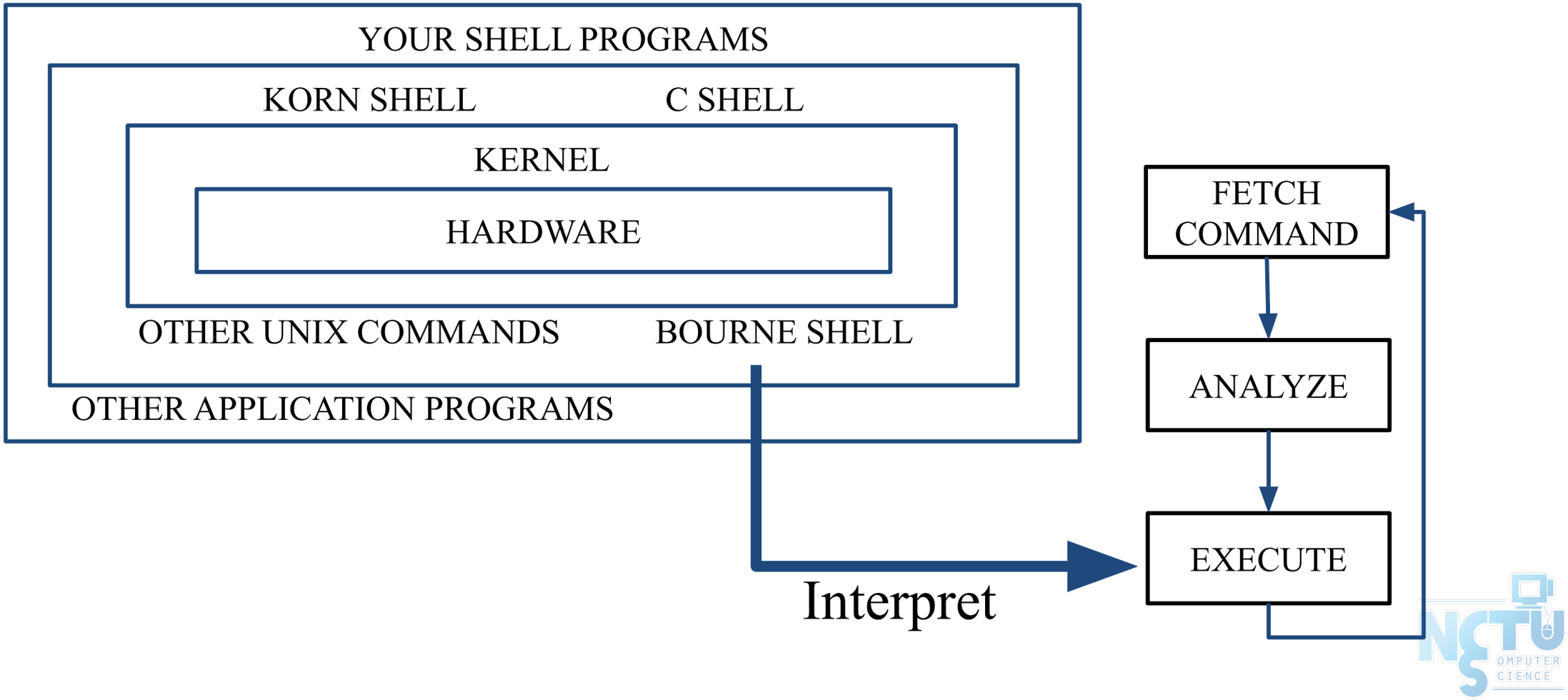
Drivers and the Kernel

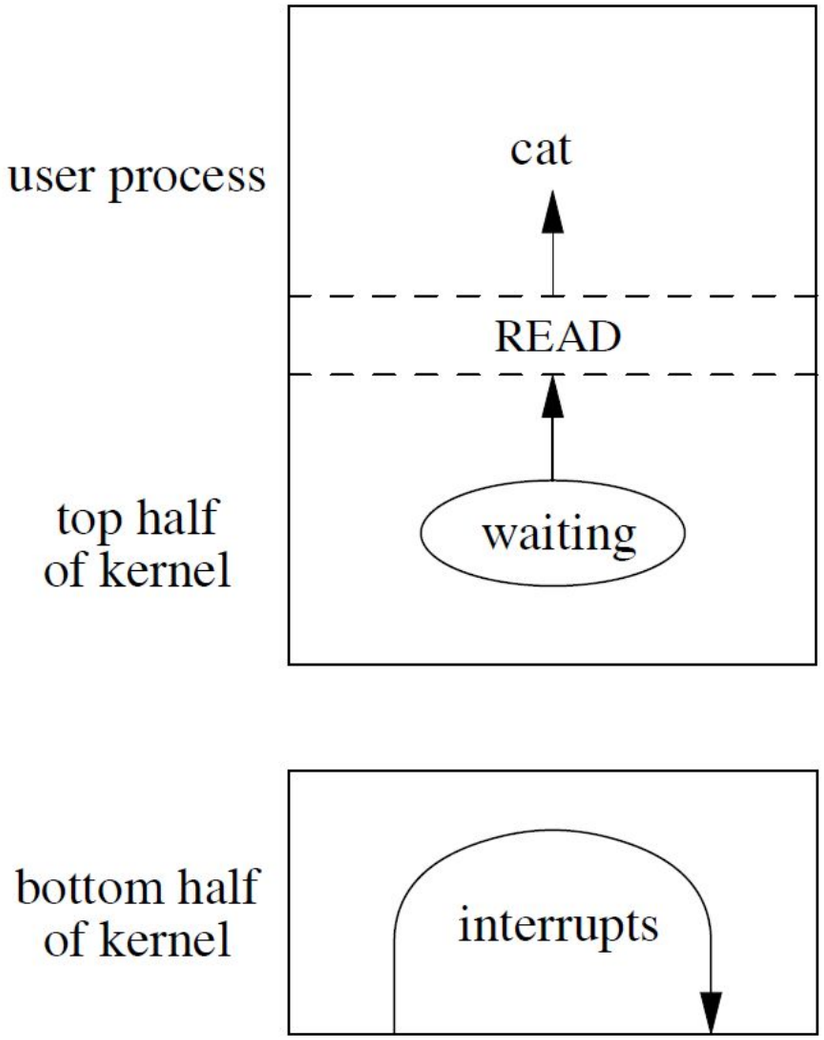
lwhsu (2019-2020, CC BY)
? (?-2018)

交大資工系資訊中心
Computer Center of Department of Computer Science, NCTU

Introduction – UNIX Kernel and Shell



Run-time structure of the kernel



Preemptive scheduling;
cannot block;
runs on user stack in user address space.

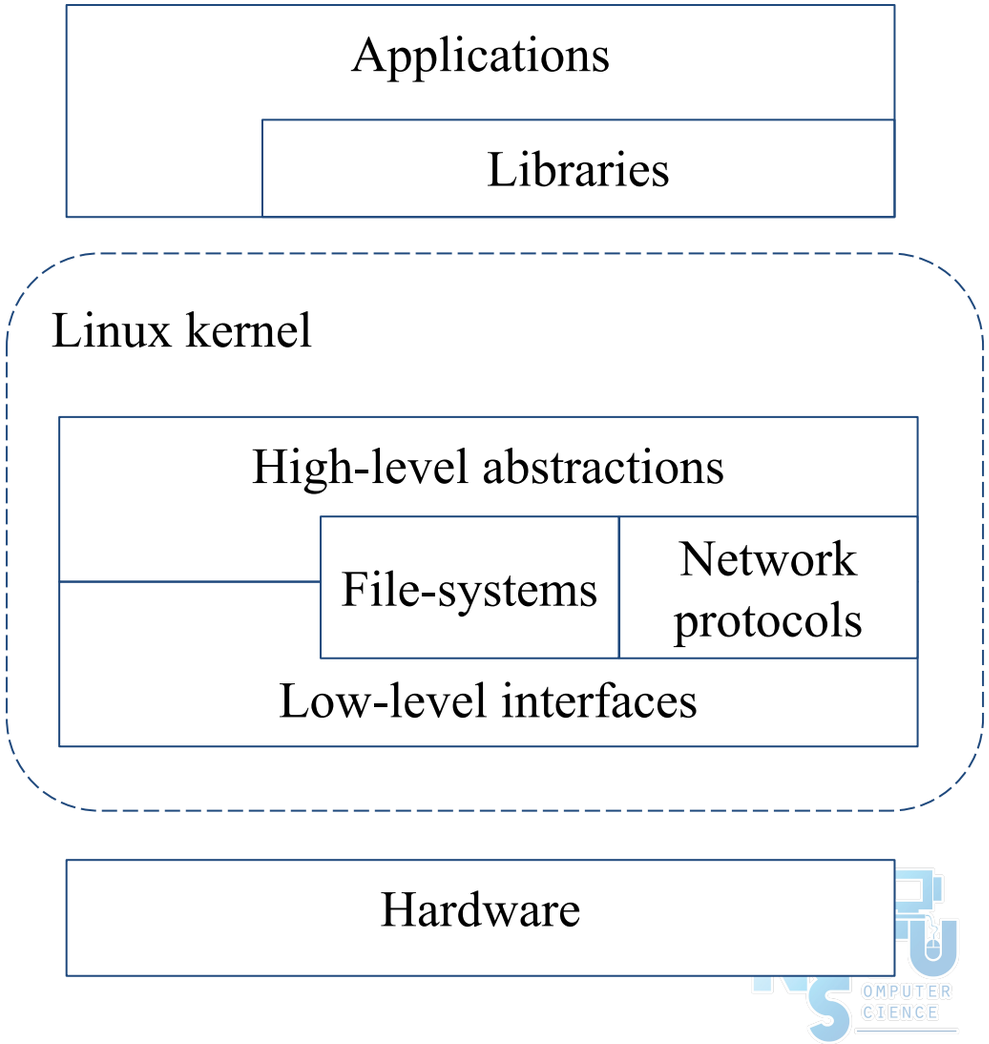
Runs until blocked or done;
can block to await a resource;
runs on per-process kernel stack.

Scheduled by interrupts;
can block to await a resource;
runs on per-interrupt kernel stack.

Figure 3.1 - Design and Implementation of the FreeBSD Operating System, The, 2nd Edition

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 I/O structure

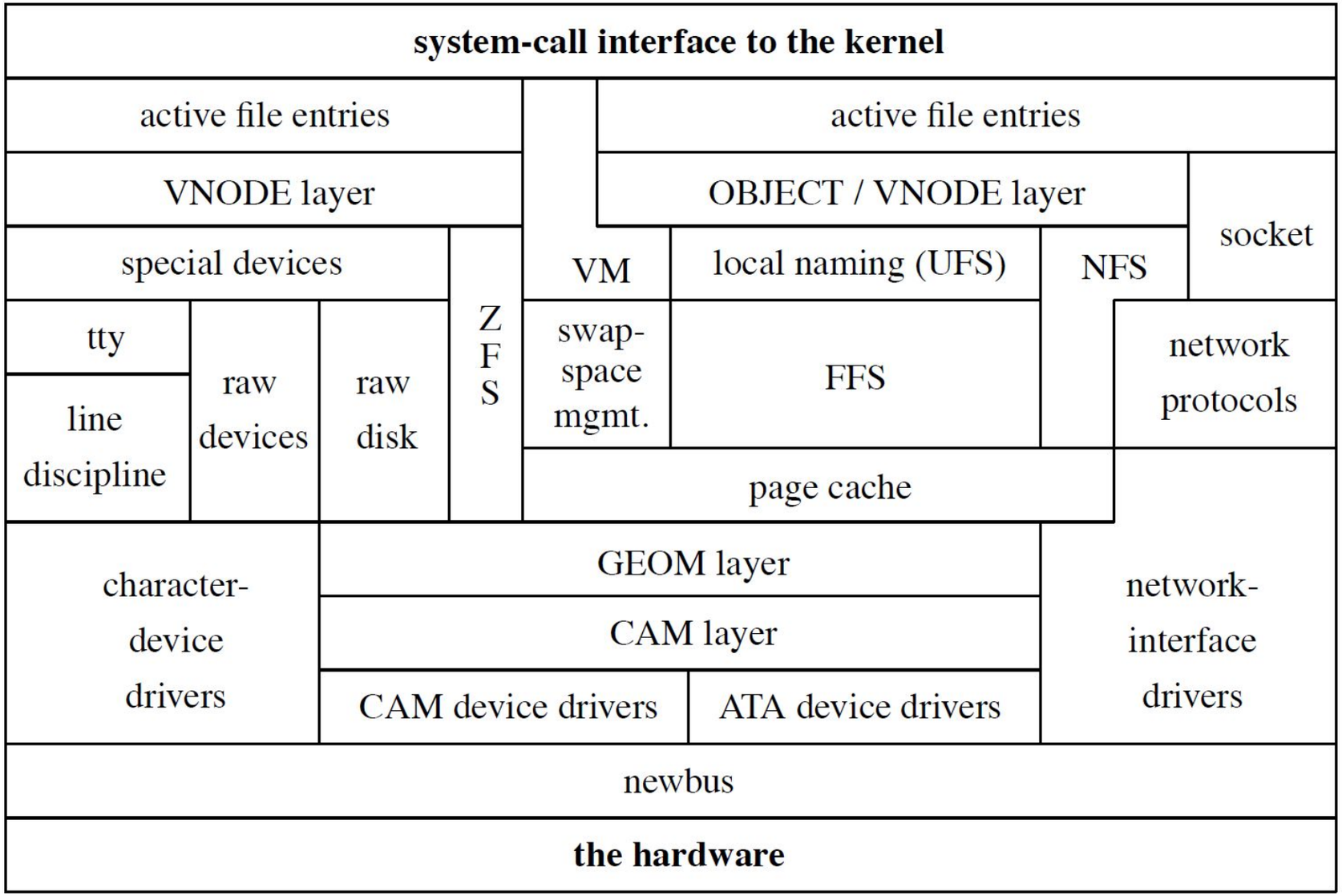
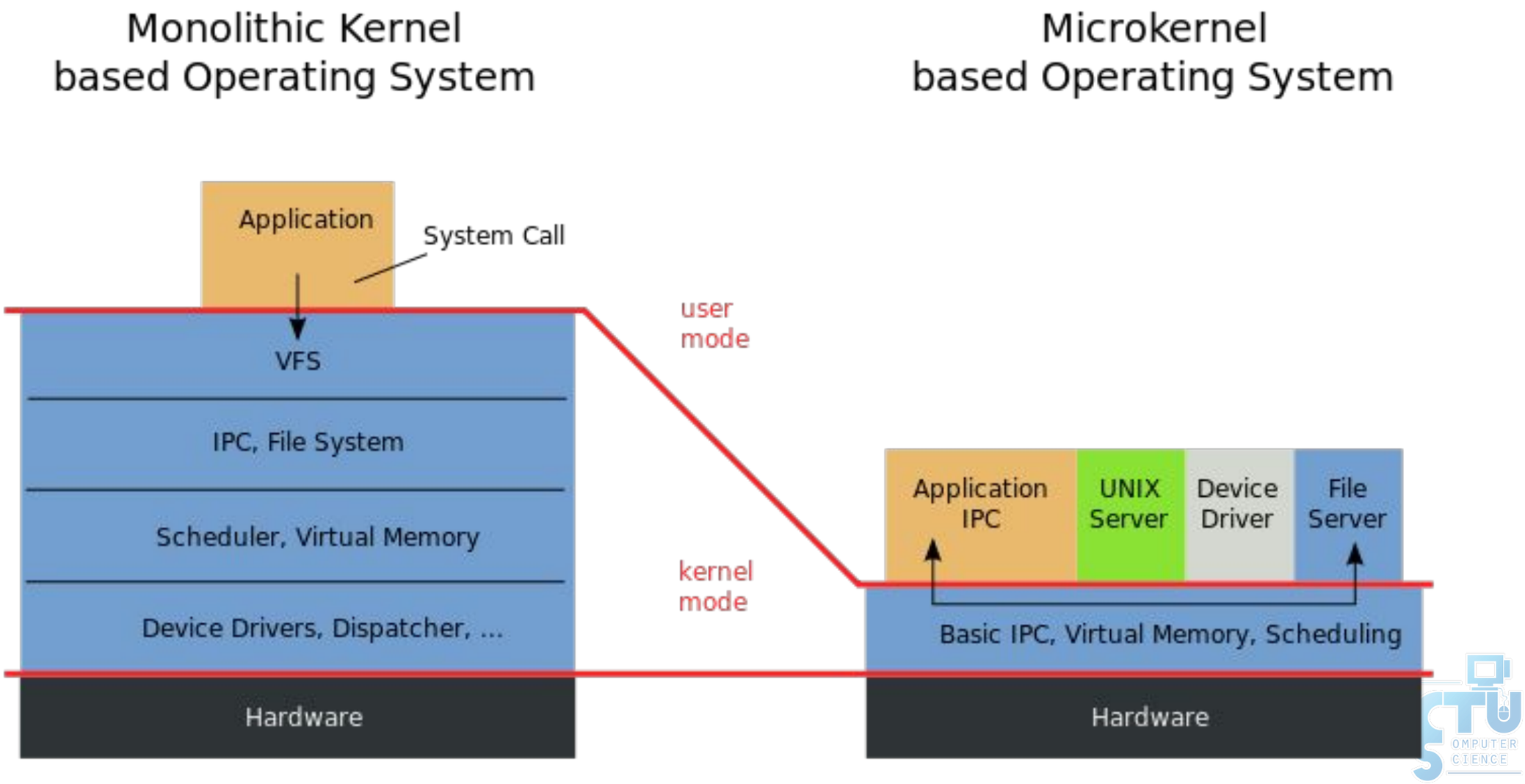


Figure 7.1 - Design and Implementation of the FreeBSD Operating System, The, 2nd Edition

Kernel Types



https://en.wikipedia.org/wiki/Microkernel

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

- increase scalability and less difficult in maintenance
- How to communicate?
  - ⇒ Message passing – less efficient

- Two extreme types
  - **Microkernel**
    - **Provide only necessarily, compact and small functionalities**
    - **Other functions is added via well-defined interfaces**
  - **Monolithic kernel (a huge kernel - e.g., UNIX)**
    - **Whole functionalities in one kernel, tightly integrated**
- 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**

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)
Linux	/usr/src/linux	/vmlinuz or /boot/vmlinuz
Solaris	-	/kernel/unix
SunOS	/usr/kvm/sys	/vmunix

Why customize kernel?

Generic: with most common device and feature supported

- The GENERIC kernel is often big and general purposed
- Tailoring kernel to match site situation
  - Purge unnecessary kernel devices and options
  - Add functionalities that you want

kernel image □ memory usage

- Patching
  - Remedy security hole of kernel implementation
- Fine-tune system performance
  - Such as adjusting important system parameters
- Add device drivers or features
- Decrease boot time
- Lower memory usage

Build and install FreeBSD Kernel

- Kernel source
  - /usr/src/sys
- Kernel configuration file
  - /usr/src/sys/<ARCH>/conf
    - GENERIC
    - LINT (generated by `make LINT` under this directory)
    - NOTES (all options with comments)
- 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

  - dmesg(8) - display the system message buffer
  - cat /var/run/dmesg.boot

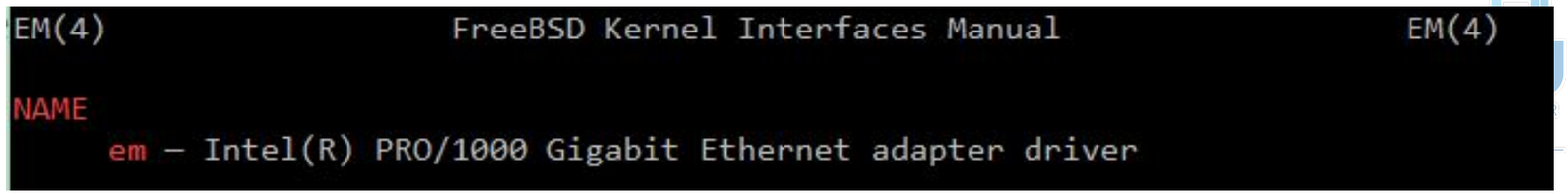
```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD] psm0: model Generic PS/2 mouse, device ID 0
```

Finding the system hardware (2)

- 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



Finding the system hardware (3)

- pciconf
  - pciconf -lv

```
none3@pci0:0:20:3:      class=0x028000 card=0x00348086 chip=0x9df08086 rev=0x30 hdr=0x00
    vendor     = 'Intel Corporation'
    device     = 'Cannon Point-LP CNVi [Wireless-AC]'
    class      = network
```

```
em0@pci0:0:31:6:        class=0x020000 card=0x20748086 chip=0x15be8086 rev=0x30 hdr=0x00
    vendor     = 'Intel Corporation'
    device     = 'Ethernet Connection (6) I219-V'
    class      = network
    subclass   = ethernet
nvme0@pci0:109:0:0:     class=0x010802 card=0x2263c0a9 chip=0x2263c0a9 rev=0x03 hdr=0x00
    vendor     = 'Micron/Crucial Technology'
    device     = 'P1 NVMe PCIe SSD'
    class      = mass storage
    subclass   = NVM
```

May not support by GENERIC because of size, license, or…

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 you
     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

- Each line is a control phrase
  - Keyword + arguments

The explanations on options and devices…

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
(no)options	Sets various compile-time options	INET, INET6
device	Declares devices	fxp, em

amd64/conf/GENERIC

```
cpu             HAMMER
ident           GENERIC
makeoptions     DEBUG=-g            # Build kernel with gdb(1) debug symbols
options         SCHED_ULE           # ULE scheduler
options         NUMA                # Non-Uniform Memory Architecture support
options         PREEMPTION          # Enable kernel thread preemption
options         INET                # InterNETworking
device          em
```

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

Kernel backup

- Kernel file locations
  - Put in the /boot directory
  - /boot/GENERIC/kernel, /boot/kernel.old/kernel

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

- If something goes wrong
  - ok mode !
    - unload kernel; load kernel.old/kernel
    - load kernel modules
  - mv /boot/kernel /boot/kernel.bad

https://www.freebsd.org/doc/en/books/handbook/kernelconfig-trouble.html

Ok mode


```
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] …
  - E.g.:
    - % sysctl -a # list all kernel variables
    - % sysctl -d vfs.zfs.arc_max # print the description of the variable
    - % sysctl vfs.zfs.arc_max # print the value of the variable
    - % sudo sysctl vfs.zfs.arc_max=4294967296 # set (only root writable) value
- tuning(7)

Kernel modules

- Kernel module location
  - /boot/kernel/*.ko
  - /boot/modules
- kldstat

```
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
- Examples in share/examples/kld

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. find 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 or
     C. Put to “kld_list” in /etc/rc.conf
- **devmatch(8)**

Building Linux Kernel

- General procedure
  - Install kernel toolchain
  - Get source code from https://kernel.org
  - Extract to /usr/src/linux
  - make menuconfig
  - make –jN
  - make modules
  - make modules_install
  - make install
  - Check /boot/{initramfs.img,System.map,vmlinuz}
- Check the distribution specified method
  - Kernel package

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
- "building kernel" of Linux distributions documents
  - https://kernel-team.pages.debian.net/kernel-handbook/ch-common-tasks.html#s-common-official
  - https://wiki.ubuntu.com/Kernel/BuildYourOwnKernel
  - https://wiki.archlinux.org/index.php/Kernel/Arch_Build_System
  - https://wiki.centos.org/HowTos/Custom_Kernel

Backup Slides

Kernel Module (c.)

- Build & install kernel module from 3rd party
- DKMS (Dynamic Kernel Module Support)