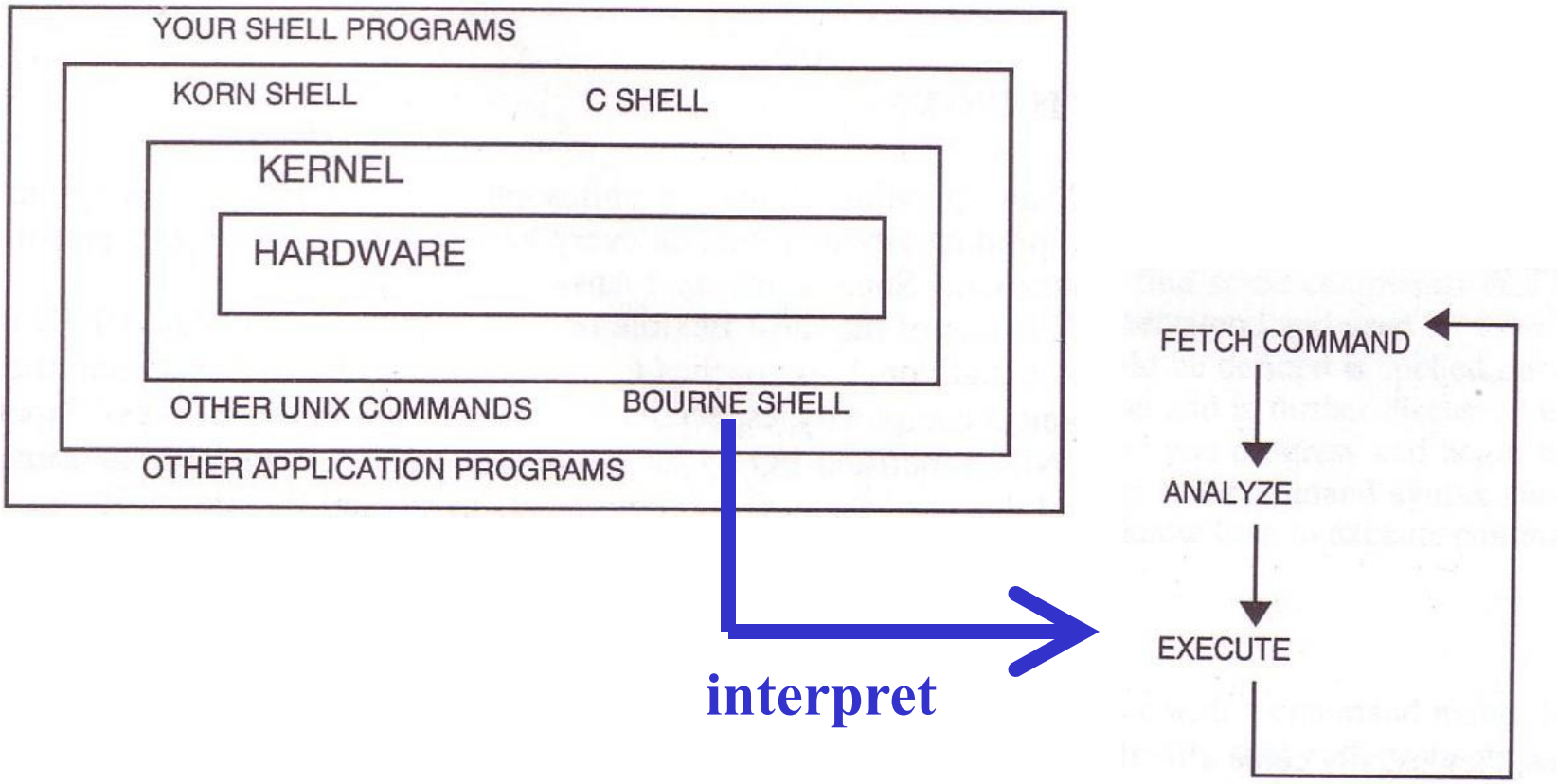
Drivers and the Kernel

MarsW

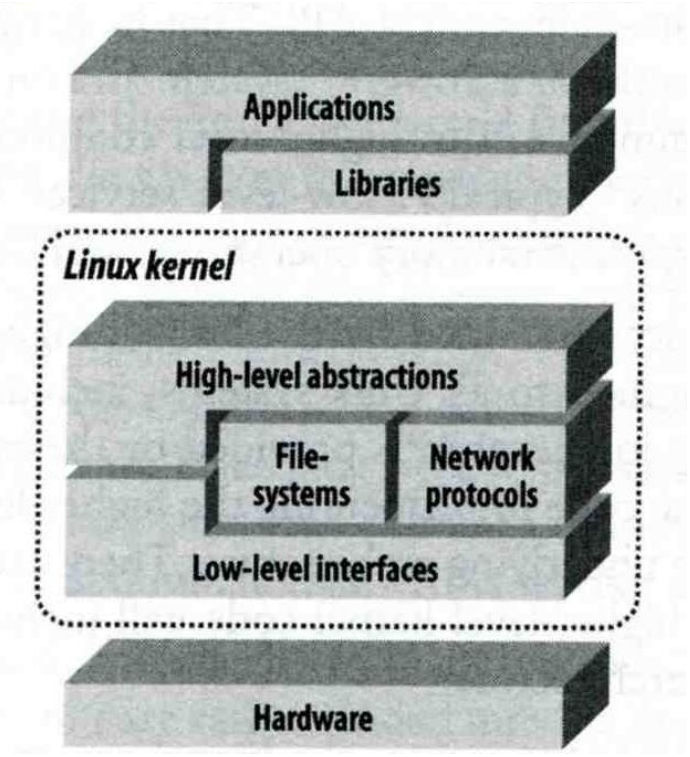
Introduction – UNIX Kernel and Shell

The User

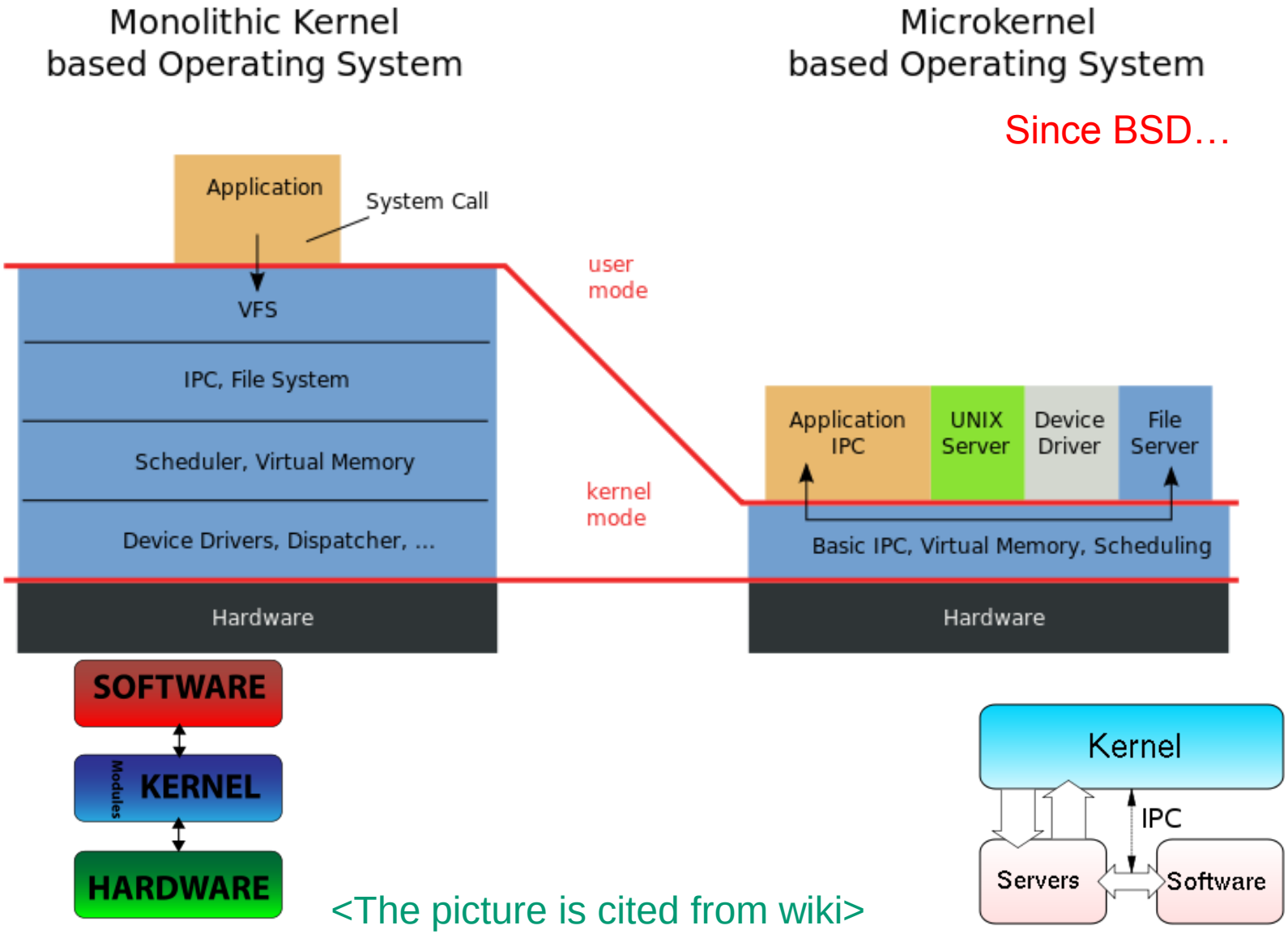


Roles of Kernel

- 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



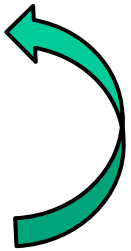
Kernel Types



<The picture is cited from wiki>

Kernel Types

- Two extreme types
  - **Micro kernel**
    - **Provide only necessarily, compact and small functionalities**
    - **Other functions is added via well-defined interface**
  - **Monolithic kernel (龐大的kernel – e.g. unix)**
    - **Whole functionalities in one kernel**



- Modern OS
  - Solaris
    - **Completely modular kernel**
    - **Load necessarily module when it is needed**
  - BSD/Linux-derived system
    - **Much of the kernel's functionality is contained in modules**

Why configure the kernel?

- 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

To Build a FreeBSD Kernel…

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

Finding the system hardware

- Before venturing into kernel configuration
  - Get an inventory of the machine's hardware
  - dmesg
    - 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
```

  - 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
```

Building a FreeBSD Kernel

- Kernel source
  - /usr/src/sys
- Kernel configuration file
  - /usr/src/sys/<ARCH>/conf
    - GENERIC, "make LINT" under this dir
    - LINT*
- Steps to build a new kernel
  - Edit /usr/src/sys/<ARCH>/conf/<KERNCONF>
  - % cd /usr/src ;
  - % make buildkernel KERNCONF=SABSD
  - % make installkernel KERNCONF=SABSD

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/kernelconfig-building.html

Building a FreeBSD Kernel – Configuration file

- Each line is a control phrase

  [Ref] http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/kernelconfig-config.html

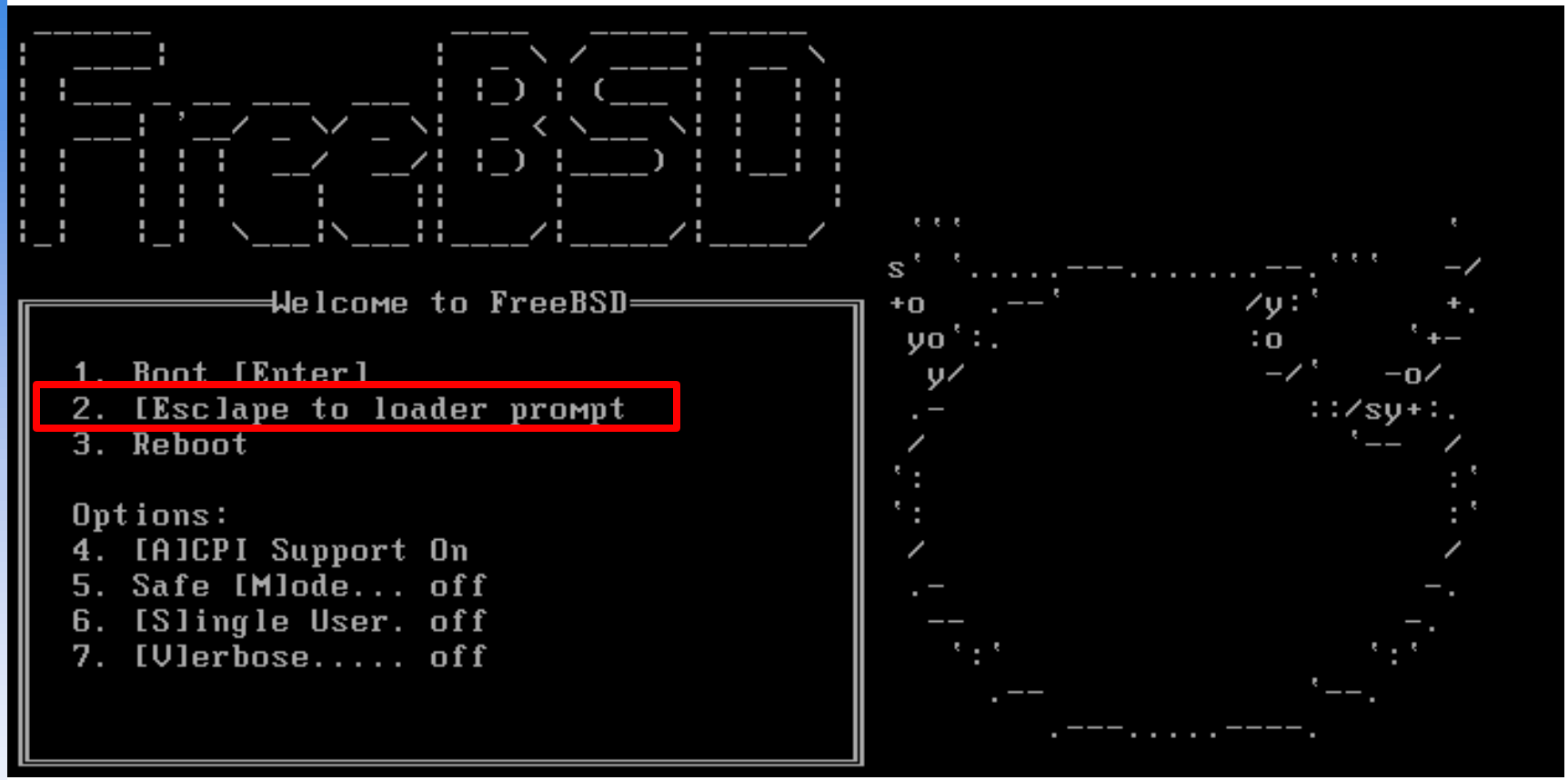
  - Keyword + arguments e.g. device = fxp

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 comiple-time options	INET or INET6
device	Declares devices	fxp

Kernel backup

- Kernel file locations
  - Put in the /boot directory
  - /boot/kernel/kernel
  - /boot/kernel.old/kernel
- If something goes wrong
  - ok mode !
    - unload kernel
    - load kernel.old/kernel
  - mv /boot/kernel */boot/kernel.bad*

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 _
```

reboot

Tuning the FreeBSD Kernel

- sysctl command
  - Dynamically set or get kernel parameters
  - All changes made by sysctl will be lost across reboot
  - Use `sysctl -w` to tune the kernel and test it, then recompile the kernel
  - /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
- 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. 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
   - recompile the kernel, or
   - add [module name]_enable="YES" in /boot/loader.conf