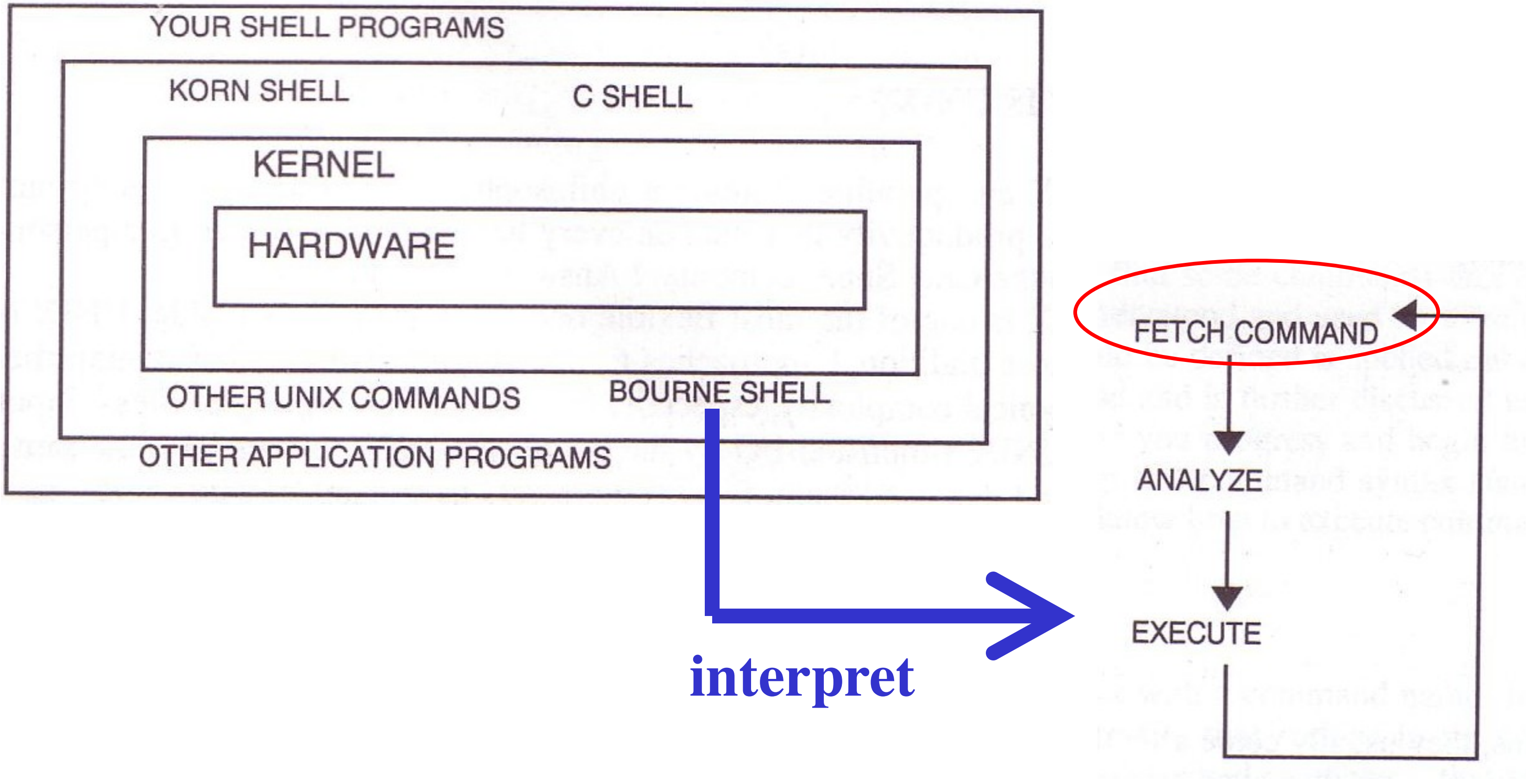
Drivers and the Kernel

lctseng

arr. by pschiu

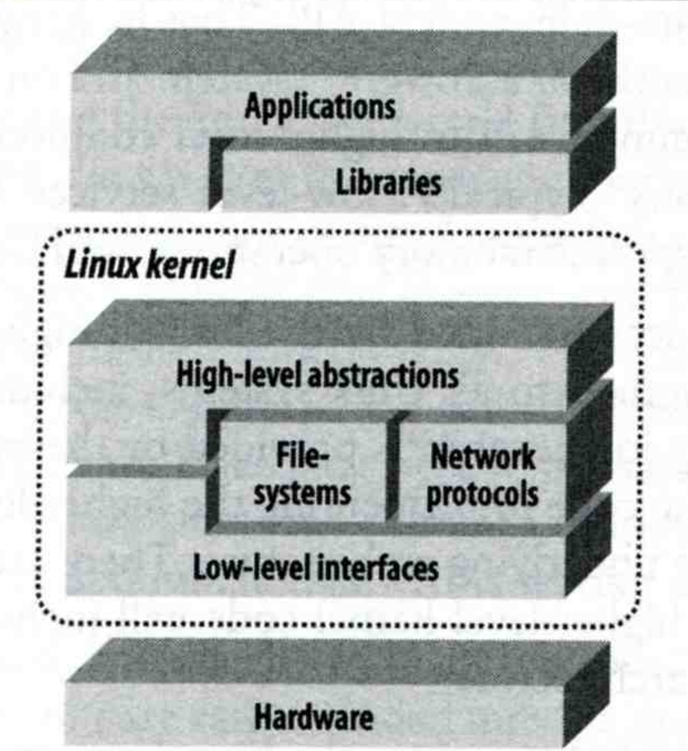
Introduction – UNIX Kernel and Shell



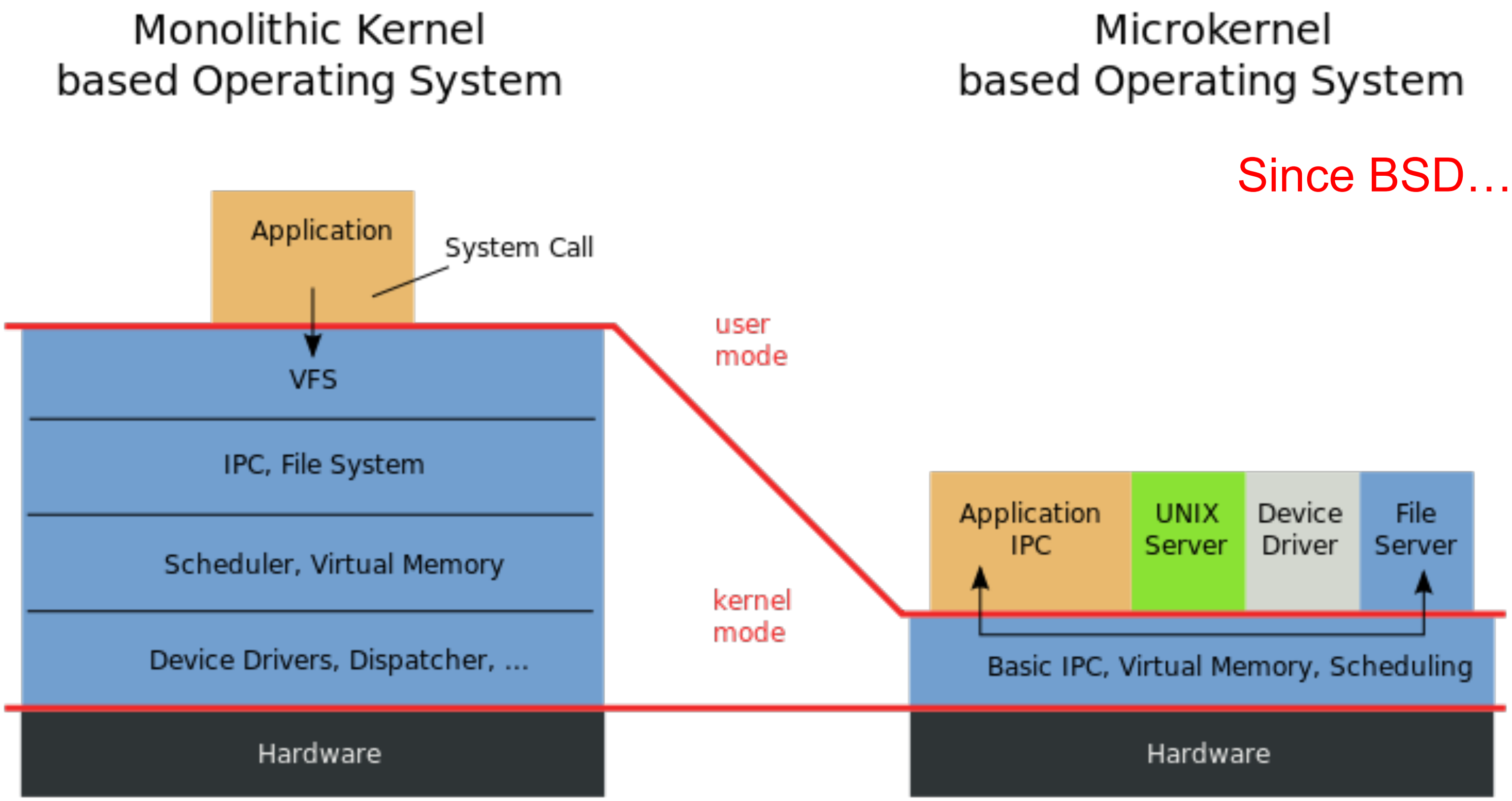
interpret

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



Since BSD…

單體式核心(單核心)

微核心

<The picture is cited from wiki>

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

- **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**

More integrated…

❑ 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**

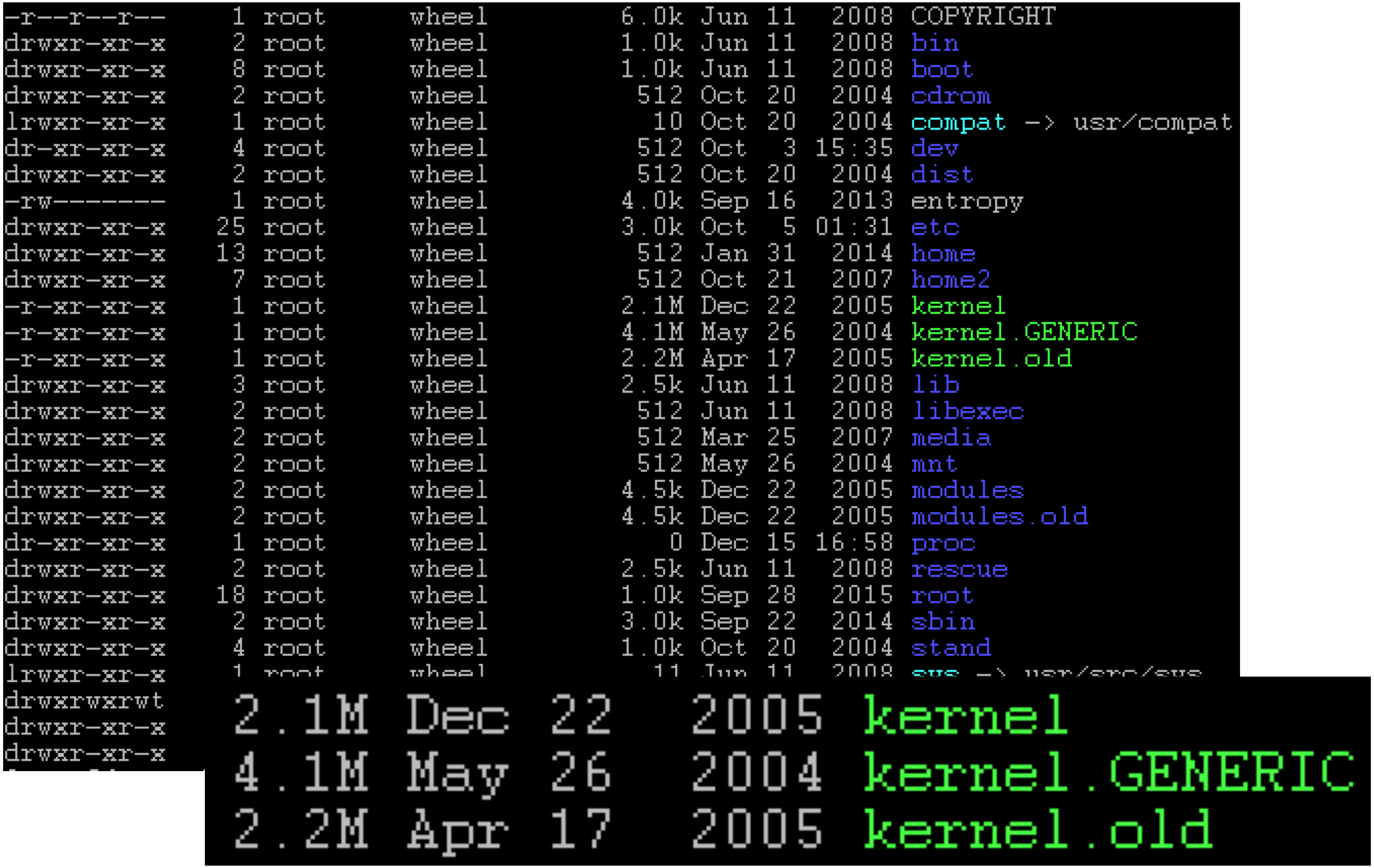
Monolithic kernel developing towards micro kernel (being more modulized), but without IPC (message passing) problem

Kernel related directory

- ❑Build directory and location

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

FreeBSD 4.x and < 4.0 kernel files



```
-r--r--r--    1 root      wheel        6.0k Jun 11   2008 COPYRIGHT
drwxr-xr-x    2 root      wheel        1.0k Jun 11   2008 bin
drwxr-xr-x    8 root      wheel        1.0k Jun 11   2008 boot
drwxr-xr-x    2 root      wheel         512 Oct 20   2004 cdrom
lrwxr-xr-x    1 root      wheel          10 Oct 20   2004 compat -> usr/compat
dr-xr-xr-x    4 root      wheel         512 Oct  3  15:35 dev
drwxr-xr-x    2 root      wheel         512 Oct 20   2004 dist
-rw-------    1 root      wheel        4.0k Sep 16   2013 entropy
drwxr-xr-x   25 root      wheel        3.0k Oct  5  01:31 etc
drwxr-xr-x   13 root      wheel         512 Jan 31   2014 home
drwxr-xr-x    7 root      wheel         512 Oct 21   2007 home2
-r-xr-xr-x    1 root      wheel        2.1M Dec 22   2005 kernel
-r-xr-xr-x    1 root      wheel        4.1M May 26   2004 kernel.GENERIC
-r-xr-xr-x    1 root      wheel        2.2M Apr 17   2005 kernel.old
drwxr-xr-x    3 root      wheel        2.5k Jun 11   2008 lib
drwxr-xr-x    2 root      wheel         512 Jun 11   2008 libexec
drwxr-xr-x    2 root      wheel         512 Mar 25   2007 media
drwxr-xr-x    2 root      wheel         512 May 26   2004 mnt
drwxr-xr-x    2 root      wheel        4.5k Dec 22   2005 modules
drwxr-xr-x    2 root      wheel        4.5k Dec 22   2005 modules.old
dr-xr-xr-x    1 root      wheel           0 Dec 15  16:58 proc
drwxr-xr-x    2 root      wheel        2.5k Jun 11   2008 rescue
drwxr-xr-x   18 root      wheel        1.0k Sep 28   2015 root
drwxr-xr-x    2 root      wheel        3.0k Sep 22   2014 sbin
drwxr-xr-x    4 root      wheel        1.0k Oct 20   2004 stand
lrwxr-xr-x    1 root      wheel          11 Jun 11   2008 sys -> usr/src/sys
drwxrwxrwt
drwxr-xr-x
drwxr-xr-x
```

```
2.1M Dec 22   2005 kernel
4.1M May 26   2004 kernel.GENERIC
2.2M Apr 17   2005 kernel.old
```

Why configure the kernel?

Generic: with various devices…, functions supported

- The native kernel is often big and common

kernel image → memory usage

- 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
- A Custom kernel benefits
  - Fast boot time
  - Lower memory usage
  - Additional hardware support

https://www.freebsd.org/doc/handbook/kernelconfig-custom-kernel.html

Building a FreeBSD Kernel - 1

- □ Kernel Source
  - /usr/src/sys
- □ Kernel Configuration File
  - /usr/src/sys/<ARCH>/conf
    - ➢ GENERIC, LINT (< 4.X)
    - ➢ GENERIC, "make LINT" under this dir (> 5.x)
- □ All Support Options
  - /usr/src/sys/<ARCH>/conf/NOTES

<ARCH> represents one of i386, amd64, ia64, powerpc, sparc64

LINT file: lists all options

→ To generate LINT file

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

Building a FreeBSD Kernel - 2

- ❑ All Supported Architectures

amd64	- 64bit Intel or AMD CPU
i386	- 32bit Intel or AMD CPU, Microsoft XBOX
ia64	- Intel's IA-64, Intel Itanium® Processor Family
pc98	- NEC PC-98x1
powerpc	- Sony Playstation 3, IBM pSeries, Apple PowerMac G3 G5
sparc64	- Sun Fire Server 15K, 10K, Ultra10
arm	- ex: Raspberry Pi

- ❑ Steps to Build a New Kernel
  - Edit /usr/src/sys/<ARCH>/conf/<KERNCONF>
    - ➢ For example, save a conf file named as SABSD
  - % cd /usr/src ;
  - % make buildkernel KERNCONF=SABSD
  - % make installkernel KERNCONF=SABSD

SABSD: configuration file

http://www.freebsd.org/doc/en_US.ISO8859-1/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
  - cat /var/run/dmesg.boot

Listing devices from dmesg

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

- ❑ Personal Habit
  - When installed fresh FreeBSD
  - # dmesg > /root/hardware

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
```

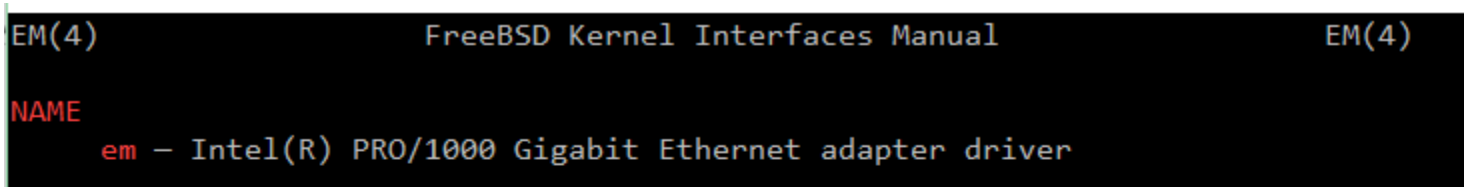
May not support by GENERIC…

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



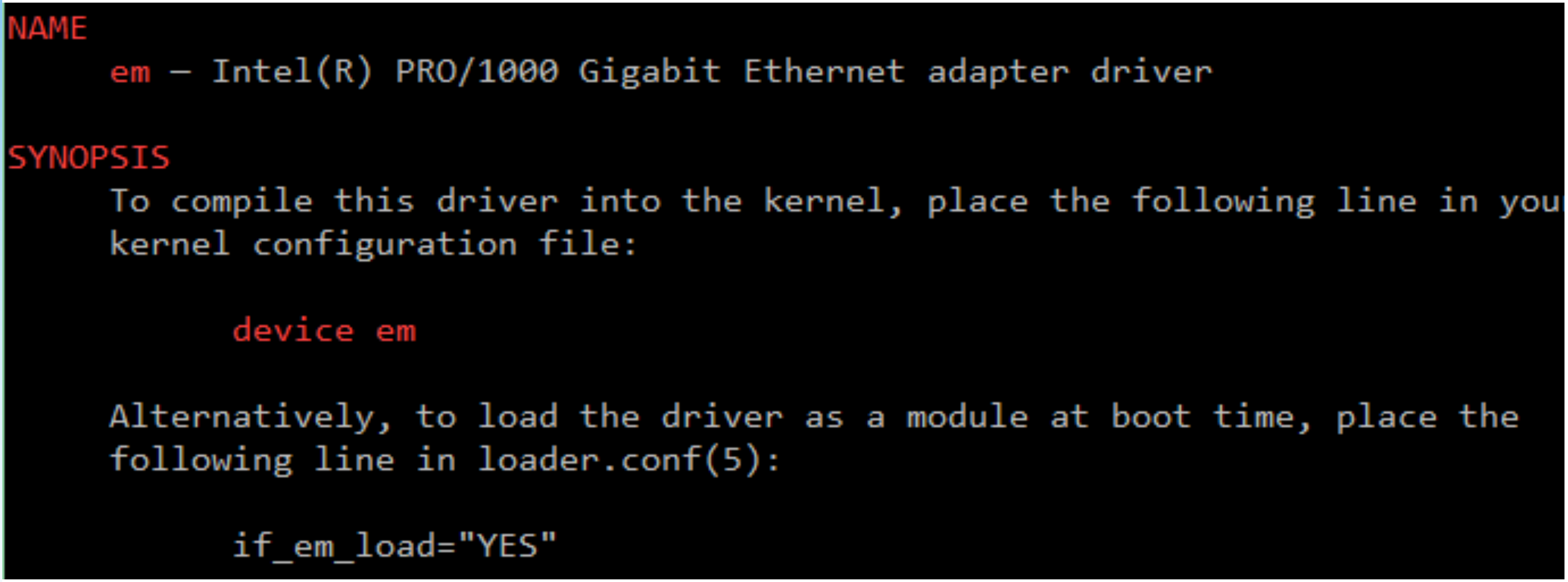
```
EM(4)                  FreeBSD Kernel Interfaces Manual                  EM(4)

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 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"
```

bge	# Broadcom BCM570xx Gigabit Ethernet
fxp	# Intel EtherExpress PRO/100B (82557, 82558)
em	# Intel PRO/1000 Gigabit Ethernet Family
igb	# Intel PRO/1000 PCIE Server Gigabit Family
rl	# RealTek 8129/8139
vmx	# VMware VMXNET3 Ethernet

Building a FreeBSD Kernel – Configuration file

The explanations on options and devices…

- Each line is a control phrase

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

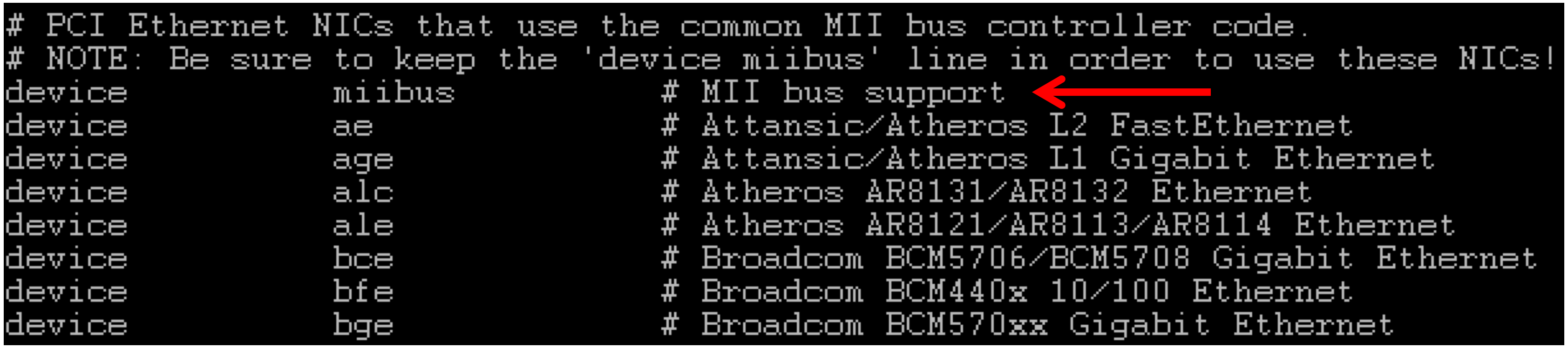
  - Keyword + arguments e.g. device fxp

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 or em

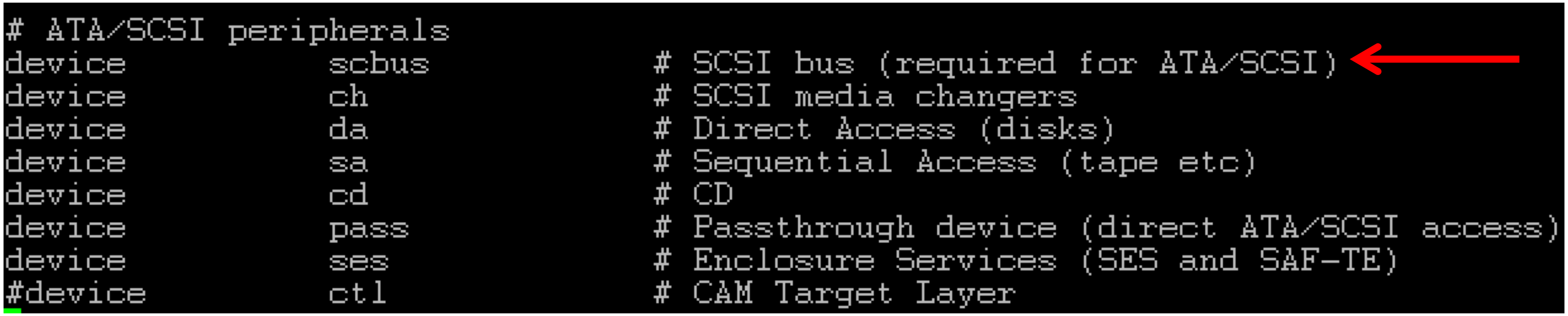
i386/conf/GENERIC

```
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
```

Dependency



```
# PCI Ethernet NICs that use the common MII bus controller code.
# NOTE: Be sure to keep the 'device miibus' line in order to use these NICs!
device          miibus          # MII bus support
device          ae              # Attansic/Atheros L2 FastEthernet
device          age             # Attansic/Atheros L1 Gigabit Ethernet
device          alc             # Atheros AR8131/AR8132 Ethernet
device          ale             # Atheros AR8121/AR8113/AR8114 Ethernet
device          bce             # Broadcom BCM5706/BCM5708 Gigabit Ethernet
device          bfe             # Broadcom BCM440x 10/100 Ethernet
device          bge             # Broadcom BCM570xx Gigabit Ethernet
```



```
# ATA/SCSI peripherals
device          scbus           # SCSI bus (required for ATA/SCSI)
device          ch              # SCSI media changers
device          da              # Direct Access (disks)
device          sa              # Sequential Access (tape etc)
device          cd              # CD
device          pass            # Passthrough device (direct ATA/SCSI access)
device          ses             # Enclosure Services (SES and SAF-TE)
#device         ctl             # CAM Target Layer
```

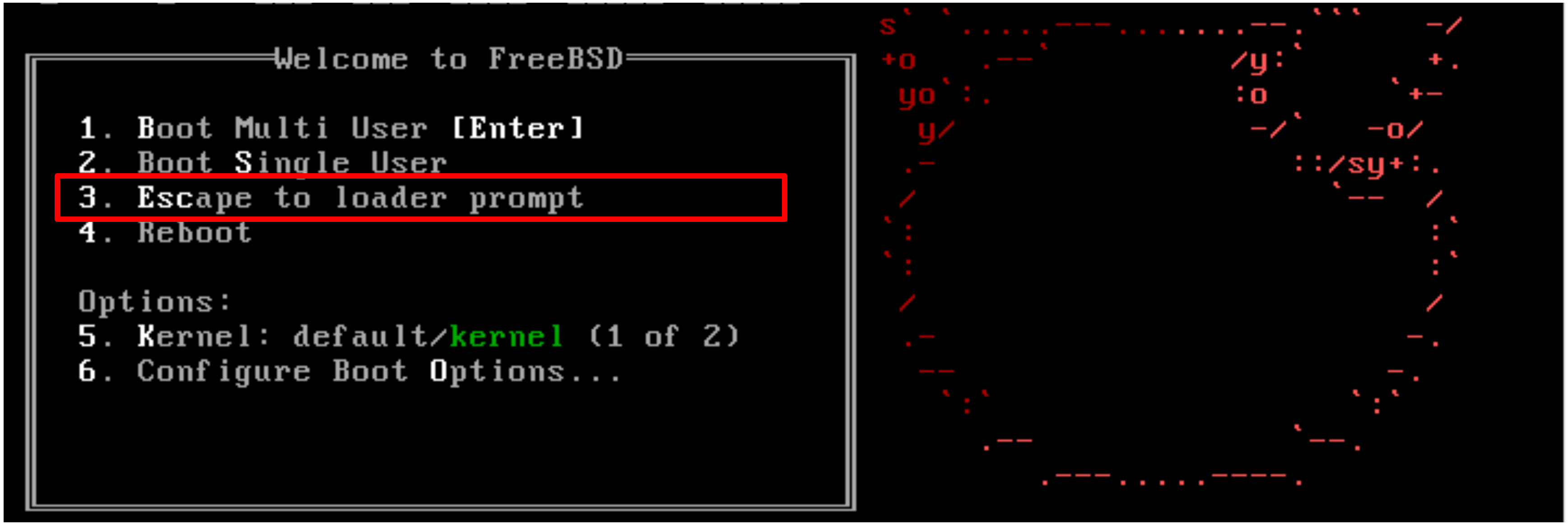
Kernel backup

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

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

- ❑ Kernel file locations
  - Put in the /boot directory
  - /boot/GENERIC/kernel, /boot/kernel.old/kernel
  - /kernel.GENERIC, /kernel.old (Freebsd 4.x)
- ❑ Backup bootable kernel
  - #cp -Ra /boot/kernel /boot/kernel.GENERIC
- ❑ If something goes wrong
  - ok mode !
    - ➢ unload kernel; load kernel.old/kernel or kernel.GENERIC
    - ➢ load kernel modules
  - mv /boot/kernel */boot/kernel.bad*

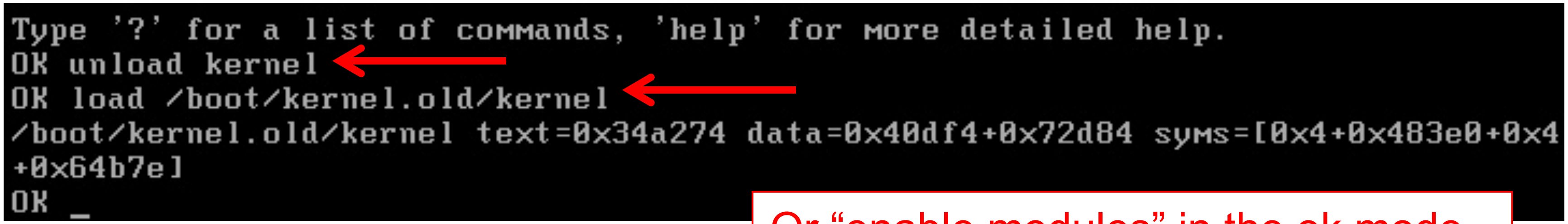
Ok mode



```
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

e.g. maxusers/maxfiles and providing www service…

❑ 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

Module loading…
e.g. kldload if_fxp

- ❑ Kernel module location
  - /boot/kernel/*.ko → Where details can be viewed
  - /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)
- ❑ Note
  - Some security environments prevent the loading and unloading of kernel modules

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

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