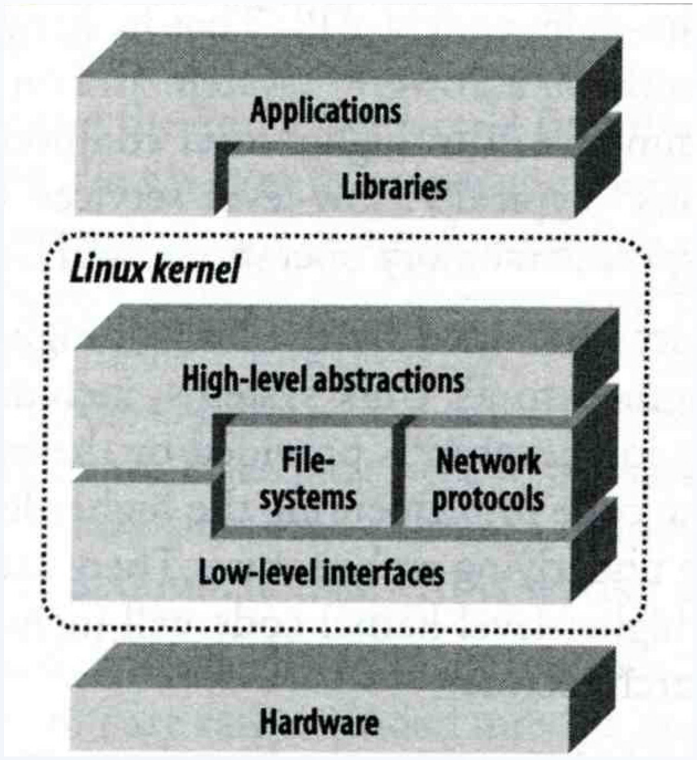
Chapter 12
Drivers and the Kernel

Roles of Kernel

- Components of a UNIX System
  - User-level programs
  - Kernel
  - Hardware
- Two roles of kernel
  - High-level abstractions
    - **Process managements**
    - **File system management**
    - **Memory management**
    - **I/O management**
  - Low-level interface
    - **drivers**



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)**
    - Whole functionalities in one kernel
- Modern OS
  - Solaris
    - Completely modular kernel
    - Load necessarily module when it is needed
  - BSD-derived system
    - Explicitly specify the devices on kernel compile process
  - Linux
    - Between BSD and Solaris System

Kernel related directory

> Build directory and location

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

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

Building a FreeBSD Kernel

> Kernel source
  - /usr/src/sys

> Kernel configuration file
  - /usr/src/sys/i386/conf
    - **GENERIC, LINT (4.X)**
    - **GENERIC, "make LINT" under this dir (5.x)**

> Steps to build a new kernel
  - Edit /usr/src/sys/i386/conf/TYBSD
  - % cd /usr/src ;
  - % make KERNCONF=TYBSD buildkernel
  - % make KERNCONF=TYBSD installkernel

Building a FreeBSD Kernel – Configuration file (1)

- 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	TYBSD
maxusers	Sets the kernel's table sizes	0
options	Sets various comiple-time options	INET or INET6
device	Declares devices	fxp
Pseudo-device	Declares pseudo-devices	loop

Building a FreeBSD Kernel – Configuration file (2)

> maxusers keyword

- The maximum number of simultaneous users
- Control the static sizing of a number of internal system tables by formula in subr_param.c
  - **# of processes**
  - **# of file table entries**
  - **# of buffers for terminal I/O**
  - **…**
- 0 will cause the system to auto-size

```
#define NPROC (20 + 16 * maxusers)
#ifndef NBUF
#define NBUF 0
#endif
#ifndef NSFBUFS
#define NSFBUFS (512 + maxusers * 16)
```

```
if (maxusers == 0) {
        maxusers = physpages / (2 * 1024 * 1024 / PAGE_SIZE);
        if (maxusers < 32)
            maxusers = 32;
        if (maxusers > 384)
            maxusers = 384;
}
```

Building a FreeBSD Kernel – Configuration file (3)

> options keywords

- Define preprocessor symbol
  - **Ex: options INET**

```
#ifdef INET
static int       do_setopt_accept_filter(struct socket *so, struct sockopt *sopt);
#endif /* INET */
```

- Preprocessor symbol with specific value
  - **Ex: MAXDSIZ="256*1024*1024"**

```
maxtsiz = MAXTSIZ;
TUNABLE_QUAD_FETCH("kern.maxtsiz", &maxtsiz);
dfldsiz = DFLDSIZ;
TUNABLE_QUAD_FETCH("kern.dfldsiz", &dfldsiz);
maxdsiz = MAXDSIZ;
TUNABLE_QUAD_FETCH("kern.maxdsiz", &maxdsiz);
dflssiz = DFLSSIZ;
TUNABLE_QUAD_FETCH("kern.dflssiz", &dflssiz);
maxssiz = MAXSSIZ;
TUNABLE_QUAD_FETCH("kern.maxssiz", &maxssiz);
```

From /usr/src/sys/kern/subr_param.c

Building a FreeBSD Kernel – Configuration file (4)

> device keyword

- Format:

  device *device-name* at *connection-info* port *address* irq *interrupt*
- *Ex:*

```
device fxp
device sio1 at isa? port IO_COM2  irq 3
device apm0             at nexus? flag 0x20
```

- *connection-info*
  - Tell the kernel where to find the device and what kind of device it is
- *address*
  - the location of the device's command and status registers in the address space of the bus
- *interrupt*
  - the IRQ the device has been configured to use
- **PCI drivers will determine the address, interrupt of device dynamically**

Building a FreeBSD Kernel – Configuration file (5)

> Pseudo-device keyword

- Programs that act as device drivers but don’t have any real hardware
- Format:

  pseudo-device *device-name number-of-instances*
- *Ex*
  - pseudo-device loop
  - pseudo-device either
  - pseudo-device pty

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

- 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
```