Chapter 21: The Linux System

- Linux History
- Design Principles
- Kernel Modules
- Process Management
- Scheduling
- Memory Management
- File Systems
- Input and Output
- Interprocess Communication
- Network Structure
- Security

實務補充資料： 鳥哥的 Linux 私房菜 (http://linux.vbird.org/)

Objectives

- To explore the history of the UNIX operating system from which Linux is derived and the principles which Linux is designed upon
- To examine the Linux process model and illustrate how Linux schedules processes and provides interprocess communication
- To look at memory management in Linux
- To explore how Linux implements file systems and manages I/O devices

21.1 History (不考)

- Linux is a modern, free operating system based on UNIX standards
  - First developed as a small but self-contained kernel in 1991 by Linus Torvalds, with the major design goal of UNIX compatibility
  - Its history has been one of collaboration by many users from all around the world, corresponding almost exclusively over the Internet
- It has been designed to run efficiently and reliably on common PC hardware, but also runs on a variety of other platforms
  - The core Linux operating system kernel is entirely original, but it can run much existing free UNIX software, resulting in an entire UNIX-compatible operating system free from proprietary code
  - Many, varying Linux Distributions including the kernel, applications, and management tools

Linux 2.0

- Released in June 1996, 2.0 added two major new capabilities:
  - Support for multiple architectures, including a fully 64-bit native Alpha port
    - Available for Motorola 68000-series processors, Sun Sparc systems, and for PC and PowerMac systems
  - Support for multiprocessor architectures
  - Other new features included:
    - Improved memory-management code, with a unified cache for file-system data
    - Improved TCP/IP performance
    - Support for internal kernel threads, for handling dependencies between loadable modules, and for automatic loading of modules on demand
    - Standardized configuration interface
- 2.4 and 2.6 increased SMP support, added journaling file system, preemptive kernel, 64-bit memory support

The Linux System

- Linux uses many tools developed as part of Berkeley's BSD operating system, MIT's X Window System, and the Free Software Foundation's GNU project
- The main system libraries were started by the GNU project, with improvements provided by the Linux community
- Linux networking-administration tools were derived from 4.3BSD code; recent BSD derivatives such as Free BSD have borrowed code from Linux in return
- The Linux system is maintained by a loose network of developers collaborating over the Internet, with a small number of public ftp sites acting as de facto standard repositories. The *File System Hierarchy Standard* specifies the overall layout of a standard Linux file system

Linux Distributions

- Standard, precompiled sets of packages, or distributions, include the basic Linux system, system installation and management utilities, and ready-to-install packages of common UNIX tools
- The first distributions managed these packages by simply providing a means of unpacking all the files into the appropriate places; modern distributions include advanced package management
- Early distributions included SLS and Slackware
  - Red Hat and Debian are popular distributions from commercial and noncommercial sources, respectively
- The RPM Package file format permits compatibility among the various Linux distributions

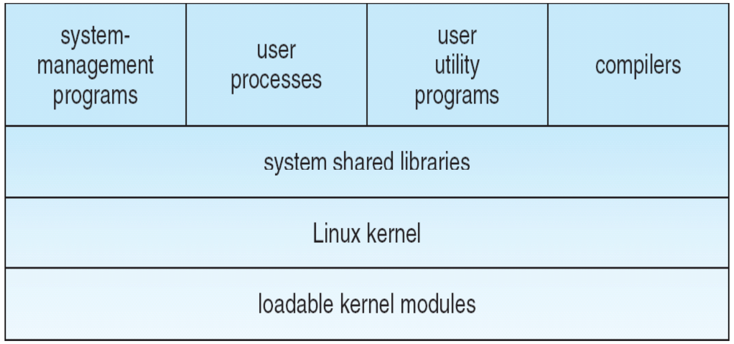
Linux Licensing

- The Linux kernel is distributed under the GNU General Public License (GPL), the terms of which are set out by the Free Software Foundation
  - If you release software that includes any component covered by the GPL, then you must make source code available alongside any binary distributions
- Linux is *free*, but *not public-domain* software
  - Copyrights are still held by various authors
- Anyone using Linux, or creating their own derivative of Linux, may not make the derived product proprietary; software released under the GPL may not be redistributed as a binary-only product

21.2 Design Principles

- Linux is a multiuser, multitasking system with a full set of UNIX-compatible tools
- Its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model
- Main design goals are speed, efficiency, and standardization
  - To avoid the lessons learned in UNIX
- Linux is designed to be compliant with the relevant POSIX (Portable Operating System Interface for Unix) standards and threading extensions; at least two Linux distributions have achieved official POSIX certification
- The Linux programming interface adheres to the SVR4 UNIX semantics, rather than to BSD behavior

Components of a Linux System



Components of a Linux System (Cont)

- Like most UNIX implementations, Linux is composed of three main bodies of code: kernel, system libraries, and system utilities; the most important distinction between the kernel and all other components
- The kernel is responsible for maintaining the important abstractions of the operating system
  - Kernel code executes in *kernel mode* with full access to all the physical resources of the computer
  - Implemented as a single, monolithic binary. All kernel code and data structures are kept in the same single address space, so that no context switches are necessary for system calls or hardware interrupt.
  - The single address space contains all kernel codes, including all device drivers, file systems, and networking code

Components of a Linux System (Cont)

- The **system libraries** define a standard set of functions through which applications interact with the kernel, and which implement much of the operating-system functionality that does not need the full privileges of kernel code
- The **system utilities** perform individual specialized management tasks
  - Some to invoke initialize and configure some aspects of the system
  - Some may run permanently, known as *daemons*

21.3 Kernel Modules

- Sections of kernel code that can be compiled, loaded, and unloaded independent of the rest of the kernel
- A kernel module may typically implement a device driver, a file system, or a networking protocol
- The module interface allows third parties to write and distribute, on their own terms, device drivers or file systems that could not be distributed under the GPL
- Kernel modules allow a Linux system to be set up with a standard, minimal kernel, without any extra device drivers built in
- Three components to Linux module support:
  - The module management
  - The driver registration
  - A conflict resolution mechanism

Module Management

- Supports loading modules into memory and letting them talk to the rest of the kernel
  - Make sure that any reference to the kernel symbols or entry points are updated to point to the correct locations in the kernel's address space
- Module loading is split into two separate sections:
  - Managing sections of module code in kernel memory
  - Handling symbols that modules are allowed to reference
- The module requestor manages loading requested, but currently unloaded, modules; it also regularly queries the kernel to see whether a dynamically loaded module is still in use, and will unload it when it is no longer actively needed

Driver Registration

- Allows modules to tell the rest of the kernel that a new driver has become available
- The kernel maintains dynamic tables of all known drivers, and provides a set of routines to allow drivers to be added to or removed from these tables at any time
- Registration tables include the following items:
  - Device drivers
  - File systems
  - Network protocols
  - Binary format

Conflict Resolution

- A mechanism that allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver
- The conflict resolution module aims to:
  - Prevent modules from clashing over access to hardware resources
  - Prevent *autoprobes* from interfering with existing device drivers
  - Resolve conflicts with multiple drivers trying to access the same hardware