2. Operating System Case Study: Linux
Reference

Features of modern OS

- To facilitate easy, efficient, fair, orderly, and secure use of resources
  - Provide a user interface
  - Organize files on disk
  - Allocating resource to different users with security control
  - Co-ordinate programs to work with devices and other programs
Case study: Linux

A. Development of Linux
Before Linux

- In 80’s, Microsoft’s DOS was the dominated OS for PC
  - single-user, single-process system
- Apple MAC is better, but expensive
- UNIX is much better, but much much expensive. Only for minicomputer for commercial applications
- People was looking for a UNIX based system, which is cheaper and can run on PC
- Both DOS, MAC and UNIX are proprietary, i.e., the source code of their kernel is protected
  - No modification is possible without paying high license fees
2. Operating System Case Study: Linux

- GNU project
  - Established in 1984 by Richard Stallman, who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs
  - GNU is a recursive acronym for “GNU's Not Unix”
  - Aim at developing a complete Unix-like operating system which is free for copying and modification
  - Companies make their money by maintaining and distributing the software, e.g. optimally packaging the software with different tools (Redhat, Slackware, Mandrake, SuSE, etc)
  - Stallman built the first free GNU C Compiler in 1991. But still, an OS was yet to be developed
Beginning of Linux

- A famous professor Andrew Tanenbaum developed Minix, a simplified version of UNIX that runs on PC
- Minix is for class teaching only. No intention for commercial use
- In Sept 1991, Linus Torvalds, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1
- It was put to the Internet and received enormous response from worldwide software developers
- By December came version 0.10. Still Linux was little more than in skeletal form.
● Confrontation and Development

- Message from Professor Andrew Tanenbaum
  - "I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-)
    (Andrew Tanenbaum to Linus Torvalds)
  - "Linux is obsolete".
    (Remark made by Andrew Tanenbaum)
- But work went on. Soon more than a hundred people joined the Linux camp. Then thousands. Then hundreds of thousands
- It was licensed under GNU General Public License, thus ensuring that the source codes will be free for all to copy, study and to change.
2. Operating System Case Study: Linux

- Linux Today
  - Linux has been used for many computing platforms
    - PC, PDA, Supercomputer,…
  - Current kernel version 2.6.13
  - Not only character user interface but graphical user interface, thanks to the X-Window technology
  - Commercial vendors moved in Linux itself to provide freely distributed code. They make their money by compiling up various software and gathering them in a distributable format
    - Red Hat, Slackware, etc
  - Chinese distribution of Linux also appeared in Taiwan and China - CLE, Red Flag Linux
Linux Pros and Cons

**Advantages over Windows**
- It's almost free to relatively inexpensive
- Source code is included
- Bugs are fixed quickly and help is readily available through the vast support in Internet
- Linux is more stable than Windows
- Linux is truly multi-user and multi-tasking
  - **multiuser**: OS that can simultaneously serve a number of users
  - **multitasking**: OS that can simultaneously execute a number of programs
- Linux runs on equipment that other operating systems consider too underpowered, e.g. 386 systems, PDA, etc
Linux Pros and Cons (Cont)

- **Disadvantages compared with Windows**
  - Isn't as popular as Windows
  - No one commercial company is responsible for Linux
  - Linux is relatively hard to install, learn and use

- Hence currently, Linux is mainly used in commercial applications, server implementation

- More than 75% current network servers are developed based on Linux or Unix systems
  - Due to the relatively high reliability
Case study: Linux

B. Linux System Architecture
2. Operating System Case Study: Linux

- **Applications**: Compilers, word processors, X-based GUI
- **LINUX Shell**: Bourne Again (bash), TC, Z, etc.
- **Language libraries**
- **System call interface**
- **Kernel**
  - Memory management
  - File management
  - Process Management
- **Device Drives**
  - BIOS
  - Computer Hardware
- **Computer Hardware**
● **Kernel**
  - The part of an OS where the real work is done

● **System call interface**
  - Comprise a set of functions (often known as Application Programmer’s Interface API) that can be used by the applications and library routines to use the services provided by the kernel

● **Application User’s Interface**
  - Interface between the kernel and user
  - Allow user to make commands to the system
  - Divided into text based and graphical based
2. Operating System Case Study: Linux

- **File Management**
  - Control the creation, removal of files and provide directory maintenance
  - For a multiuser system, every user should have its own right to access files and directories

- **Process Management**
  - For a multitask system, multiple programs can be executed simultaneously in the system
  - When a program starts to execute, it becomes a process
  - The same program executing at two different times will become two different processes
  - Kernel manages processes in terms of creating, suspending, and terminating them
  - A process is protected from other processes and can communicate with the others
2. Operating System Case Study: Linux

- **Memory management**
  - Memory in a computer is divided into *main memory* (RAM) and *secondary storage* (usually refer to hard disk)
  - Memory is small in capacity but fast in speed, and hard disk is vice versa
  - Data that are not currently used should be saved to hard disk first, while data that are urgently needed should be retrieved and stored in RAM
  - The mechanism is referred as *memory management*

- **Device drivers**
  - Interfaces between the kernel and the BIOS
  - Different device has different driver
Case study: Linux

B.1 User interface
Linux User Login

- Linux is a multiuser OS
- Allow multiple users to use the resource of a computer at the same time
- Every user needs to login the system with the password provided to identify their right in using the resource
- Require for both client-server based system or desktop

![Client-server based system diagram]

Peter: admin
Paul: general
Mary: intruder
Linux User Interface

- Traditional Linux (Unix also) uses command-driven interface (or text-based interface)
  - User needs to type lines of command to instruct the computer to work, similar to DOS
  - **Advantage:** fast in speed. Very few resource is required for its implementation
  - **Disadvantages:** user needs to type, hence can easily make error. Besides, user needs to memorize all commands
  - Suitable for expert users and for the systems that interaction with user is not frequent, such as servers
By adopting the **X-Window technology**, graphical user interface (GUI) is available for Linux:

- Uses pointing devices (e.g. mouse) to control the system, similar to Microsoft’s Windows
- Provide menu-driven and/or icon-driven interfaces
  - **menu-driven**: user is provided with a menu of choices. Each choice refers to a particular task
  - **icon-driven**: tasks are represented by pictures (icon) and shown to user. Click on an icon invokes one task
- **Advantages**: No need to memorize commands. Always select task from menus or icons
- **Disadvantages**: Slow and require certain resource for its implementation
- Suitable for general users and systems, such as PC
2. Operating System Case Study: Linux

- A typical Linux GUI based on GNOME
- Similar to Microsoft’s Windows, however, different window systems can be chosen (e.g. GNOME, KDE, etc)
2. Operating System Case Study: Linux

The prompt $ shows that bash shell is using.

**Linux text-based interface**

command to show the content of current directory

command to show the content of current directory with option -al
Linux Shell

- **Shell** interprets the command and request service from kernel
- Similar to DOS but DOS has only one set of interface while Linux can select different shell
  - Bourne Again shell (Bash), TC shell (Tcsh), Z shell (Zsh)
- Different shell has similar but different functionality
- **Bash** is the default for Linux
- Graphical user interface of Linux is in fact an application program work on the shell
2. Operating System Case Study: Linux

- Frequently used commands available in most shells:
  - `ls`: to show (list) the names of the file in the current directory
  - `cd`: change directory,
    - e.g. `cd /` change to the root directory
    - `cd ..` change to the parent of that directory
  - `cp`: copy one file to another
    - e.g. `cp abc.txt xyz.txt` copy abc.txt to xyz.txt
  - `rm`: remove a file
  - `man`: ask for the manual (or help) of a command
    - e.g. `man cd` ask for the manual of the command cd
  - `pwd`: show the name of the present working directory
  - `cat`: to show the content of a text file
    - e.g. `cat abc.txt` show the content of abc.txt
  - `whoami`: to show the username of the current user
Case study: Linux

B.2 File management
Linux File Management

- In Linux, file is defined as simply the thing that deals with a sequence of bytes
- Hence **everything are files**
  - An ordinary file is a file; a directory is also file; a network card, a hard disk, any device are also files since they deal with a sequence of bytes
- Linux supports five types of files
  - simple/ordinary file (text file, c++ file, etc)
  - directory
  - symbolic (soft) link
  - special file (device)
  - named pipe (FIFO)
The concept of simple file and directory is similar to DOS

Names in blue are directories, indicated by a letter d at the beginning of the line
Symbolic (soft) link
- Not a real file, just a link to another file
- Allow giving another name to a file without actually duplicates it – hence save memory space

Special file (device)
- Each hardware device, e.g. keyboard, hard disk, CD-ROM, etc is associated with at least one file
- Usually store in /dev directory
- Applications can read and write any devices by reading and writing their associate file – hence the access method is known as device independent
- Divide into two types: character special files, e.g. keyboard, and block special files, e.g. disk
A symbolic link begins with a letter $l$.

Command that sets a symbolic link to a file called CUI to anotherCUI.

File size is only 6 bytes.
### 2. Operating System Case Study: Linux

Some of the special device files in `/dev`:

<table>
<thead>
<tr>
<th>Device</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fd0</td>
<td>block</td>
<td>floppy disk</td>
</tr>
<tr>
<td>md0</td>
<td>block</td>
<td>CD-Rom</td>
</tr>
</tbody>
</table>

- Some are character devices, hence start with a letter `c`.
- Both of them are block devices, hence start with a letter `b`.
Linux File System Structure

- According to the File System Standard (FSSTND) proposed in 1994, every LINUX system should contain a set of standard files and directories.
2. Operating System Case Study: Linux

- **Root Directory ( / )**
  - Top of the file system. Similar to \ in DOS

- **/bin**
  - Contain the binary (executable code) of most essential Linux commands, e.g. bash, cat, cp, ln, ls, etc.

- **/boot**
  - Contain all the files needed to boot the Linux system, including the binary of the Linux kernel. E.g., on Red Hat Linux 6.1, the kernel is in /boot/vmlinux-2.2.5-15 file

- **/dev**
  - Contain the special files for devices, e.g. fd0, hd0, etc.
• /etc
  – Contain host-specific files and directories, e.g. information about system configuration
  – /etc/passwd
    – This file contains login information of users in the system
    – For every user, one line of record is stored in the following format:

login_name : dummy_or_encrypted_password : user_ID : group_ID : user_info : home_directory : login_shell
2. Operating System Case Study: Linux

- E.g. davis:x:134:105:James A Davis:/home/davis:/bin/bash
  - davis: login name
  - x: means that it is a dummy password. The encrypted password is stored in /etc/shadow. This field can also be used to store the actual encrypted password. In any case, the original (unencrypted) password cannot be seen by anyone, including the administrator.
  - 134: a user id given to that user. Range from 0 to 65535. 0 is assigned to super-user. 1 to 99 are reserved.
  - 105: a group id given to that user to indicate which group he belongs to. Range from 0 to 65535. 0 to 99 reserved.
  - James A Davis: user info, usually user's full name.
  - /home/davis: home directory of the user.
  - /bin/bash: the location of the shell the user is using.
2. Operating System Case Study: Linux

- **/home**  
  - Contain the **home directories of every user in the system**, e.g. dlun, guest, etc

- **/lib**  
  - Store all **essential libraries** for different language compilers

- **/lost+found**  
  - Contain all the files on the system **not connected to any directory**.
  - System administrator should determine the fate of the files in this directory
2. Operating System Case Study: Linux

- **/mnt**
  - Use by system administrator to mount file systems temporarily by using the mount command
  - Before using any devices, they have to be mounted to the system for registration
  - For example, after mounting a CD-ROM, the file system in it will be mapped to /mnt/cdrom directory
  - User can then read and write files in the CD-ROM by accessing this directory
  - Similar to mapping a drive letter to a CD-ROM in Windows
  - Different from the special file in /dev. Special file is only a place where data of the CD-ROM is transferred or stored. No file system concept
2. Operating System Case Study: Linux

- **/opt**
  - Use to install add-on software packages, e.g. star office, etc.

- **/proc**
  - Contain process and system information

- **/root**
  - Home directory of the user root, usually the administrator

- **/sbin**
  - The directories /sbin, /usr/sbin, and /usr/local/sbin contain system administration tools, utilities and general root only commands, such as halt, reboot and shutdown
2. Operating System Case Study: Linux

- **/tmp**
  - Contain temporary files. Usually files in this directory will be deleted from time to time to avoid the system fills with temp files

- **/usr**
  - One of the largest sections of the Linux file system
  - Contain read-only data that are shared between various users, e.g. the manual pages needed for the command man. Stored in /usr/man directory

- **/var**
  - Contain data that keeps on changing as the system is running. E.g. /var/spool/mail directory keeps the mail of user
Linux File Access Privilege

- Linux is a multiuser system, the files of all users are stored in a single file structure.
- Mechanism is required to restrict one user to access the files of another user, if he is not supposed to.
- User can impose access permission to each file to restrict its access.
- The term “access permission” refers to:
  - read permission
  - write permission
  - execute permission
The file access permission can be seen by using the command `ls -l` or `ls -al`.
It is a directory.

The directory can be read, written and executed by the user dlun.

The directory can be read and executed but not written by other users in the same group of dlun.

The directory can be read and executed but not written by other users in different group of dlun.

The group of a user is assigned by the administrator when a user is added to the system.
Access permission can also be assigned to a directory.

Directory is also a file that contains the attributes of the files inside it.

If read permission is not given to a directory:
- cannot show the structure of this directory
- e.g. cannot use ls

If write permission is not given to a directory:
- cannot modify anything of the directory structure
- e.g. cannot copy a file into this directory since it will modify the directory structure by adding one more file

If execute permission is not given to a directory:
- nearly nothing can be done with this directory, even cd
The access permission of a file or directory can be changed by using the command

```
chmod xyz filename/directory name
```

- `xyz` refers to 3 digit in octal form
- E.g.

```
660 : 110 110 000
  ⇒ rw- rw- ---
545 : 101 100 101
  ⇒ r-x r-- r-x
```
temp does not have execution right

even cd is not workable

execution right is added

now we can change the directory to temp
2. Operating System Case Study: Linux

File Storage in Linux

- Data storage on hard disk
  - Data in a hard disk are stored on a magnetic flat plate
  - Disk’s surface needs to be partitioned and labeled so that computer can go directly to a specific point on it
  - Achieve by low level formatting the disk
    - Create magnetic concentric circles called tracks
    - Each track is split into smaller parts called sectors and numbered

- Each sector: hold 512 bytes data
  - E.g. 80 tracks (from outer to inner 0 .. 79), 18 sectors disk can store 80x18x512 bytes data.
2. Operating System Case Study: Linux

Density of data is higher for inner tracks than outer tracks.
2. Operating System Case Study: Linux

- Must read or write whole sector at a time
- OS allocates groups of sectors called cluster to files
- Files smaller than the cluster will still be allocated the whole cluster, but the rest left unused
- In Linux, every file is associated with an **inode** that records its location in the disk
- The inode of all files are put together in a data structure called **inode table**
- In the **directory**, every file is associated with a **inode number** that points to an entry of the inode table
2. Operating System Case Study: Linux

Contents of the directory /home/dlun

<table>
<thead>
<tr>
<th>Access</th>
<th>Link</th>
<th>User ID</th>
<th>Time created</th>
<th>Time last updated</th>
<th>Lab type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1076</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2083</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13059</td>
<td>lab1.c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17488</td>
<td>lab2.c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18995</td>
<td>lab3.c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Case study: Linux

B.3 Process management
Linux Process Management

- Linux is a **multitasking** system
- Multiple programs can be executed at the same time
- Ultimately, a program needs to be executed by a CPU
- If there is only one CPU, how multiple programs can be executed at the same time?
  ⇒ By **time sharing**
- That is, all programs are claimed to be executing. In fact, most of them are **waiting** for the CPU
A program that is claimed to be executing is called a **process**.

For a multitasking system, a process has at least the following three **states**:

- **Ready**
- **Running**
- **Sleeping**

Start execution → Ready → Running → Finish execution
2. Operating System Case Study: Linux

- **Ready state**
  - All processes that are ready to execute but **without the CPU** are at the ready state
  - If there is only 1 CPU in the system, all processes except one are at the ready state

- **Running state**
  - The process that **actually possesses the CPU** is at the running state
  - If there is only 1 CPU in the system, at most there is only one process is at the running state

- **Sleeping state**
  - The process that is **waiting for other resources**, e.g. I/O, is at the sleeping state
Processes will alternatively get into the CPU one after the other (called the **round robin scheme**).

A process will be “in” a CPU for a very short time (**quantum**).
- For Linux, each quantum is about 100msec.

At the time that a process is selected to be “in” the CPU:
- It goes from **ready state to running state**.

After that, it will be swapped out:
- It goes from **running state back to ready state**.

Or it may due to the waiting of an I/O device, e.g. mouse:
- It goes from **running state to sleeping state**.

When obtaining the required resource:
- It goes from **sleeping state to ready state**.
2. Operating System Case Study: Linux
The mechanism to determine which process should “get into” the CPU is called **Process scheduling**

For example,

Program A

Actual sequence of operations
Program B

Actual sequence of operations
2. Operating System Case Study: Linux

- Program A and B will be at the running state alternatively, depends on the quantum size and the availability of the required resource.
The processes of a system can be seen by using the command `ps`.

Terminal pts/0 has the editor `vi` running.

Terminal pts/1 is executing `ps` to see the processes of both terminals.
### 2. Operating System Case Study: Linux

<table>
<thead>
<tr>
<th>PID</th>
<th>TTY</th>
<th>STAT</th>
<th>TIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>14748</td>
<td>pts/1</td>
<td>S</td>
<td>0:00</td>
<td>–bash</td>
</tr>
<tr>
<td>14795</td>
<td>pts/0</td>
<td>S</td>
<td>0:00</td>
<td>–bash</td>
</tr>
<tr>
<td>14974</td>
<td>pts/0</td>
<td>S</td>
<td>0:00</td>
<td>vi test1.txt</td>
</tr>
<tr>
<td>14876</td>
<td>pts/1</td>
<td>R</td>
<td>0:00</td>
<td>ps ...</td>
</tr>
</tbody>
</table>

**Process ID**

**Terminal name**

**State:**
- S – Sleeping
  - (waiting for input)
- R – Running

How much time the process is continuously executing
For the example above, both bash processes, which are the shell of both terminals, are waiting for the input of user. They must be in the **sleeping state**.

The vi process, which is an editor, is also waiting for the input of user. Hence it is also in **sleeping state**.

When `ps` reporting the processes in the system, it is the only process that is running. Hence it is in **running state**.
A process can be forced to terminate by using the command `kill -9 PID`

The vi process is terminated by using the command `kill -9 14874`