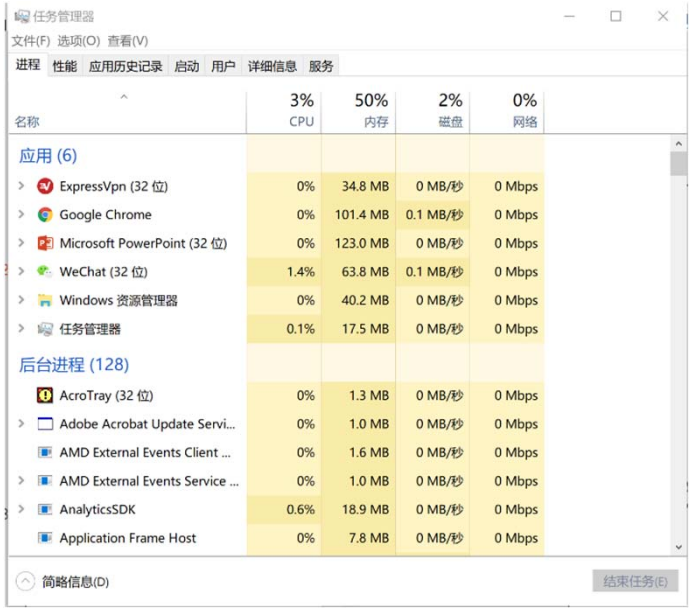
Operating Systems

Jinghui Zhong （钟竞辉）
Office： B3-515
Email： jinghuizhong@scut.edu.cn

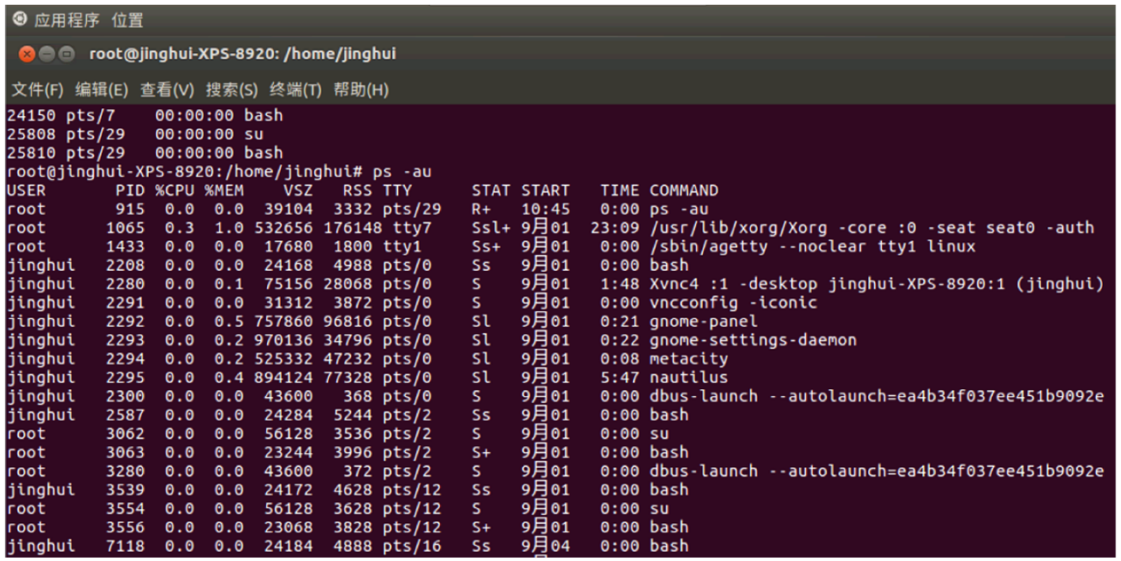
Processes

- **What is a process?**

There are a number of processes running in our computer.



Windows 10



Ubuntu 16.04

Questioning

- **Please use a concise sentence to define "Process"**

A key concept in all operating systems is the **process**. A process is basically a program in execution. Associated with each process is its **address space**, a list of memory locations from 0 to some maximum, which the process can read and write. The address space contains the executable program, the program's data, and its stack. Also associated with each process is a set of resources, commonly including registers (including the program counter and stack pointer), a list of open files, outstanding alarms, lists of related processes, and all the other information needed to run the program. A process is fundamentally a container that holds all the information needed to run a program.

Processes

- **What is a process?**

① **a program in execution**.

② **a container** that holds all the information needed to run a program.

Processes

- **Process Table**

-- Stores information about processes

-- Process control block，PCB

✓ **UID**.

✓ **PID**.

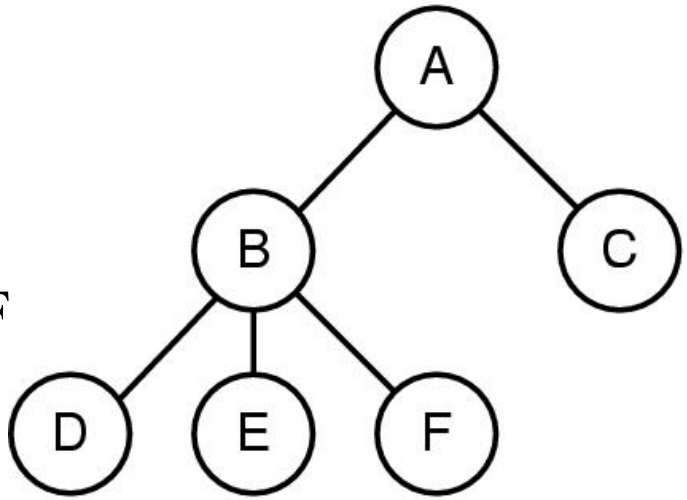
✓ **GID**.

Processes

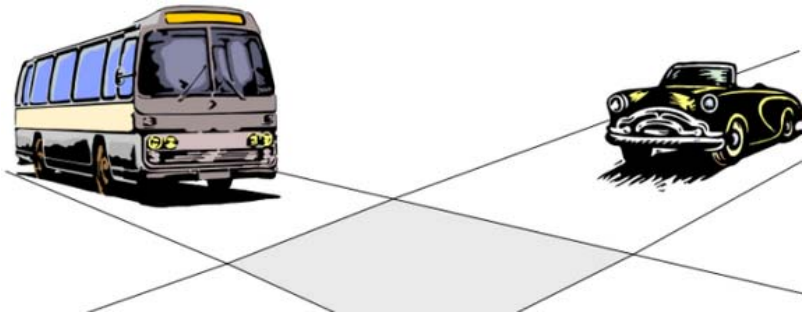
- **Process Tree**

  A created two child processes, B and C

  B created three child processes, D, E, and F

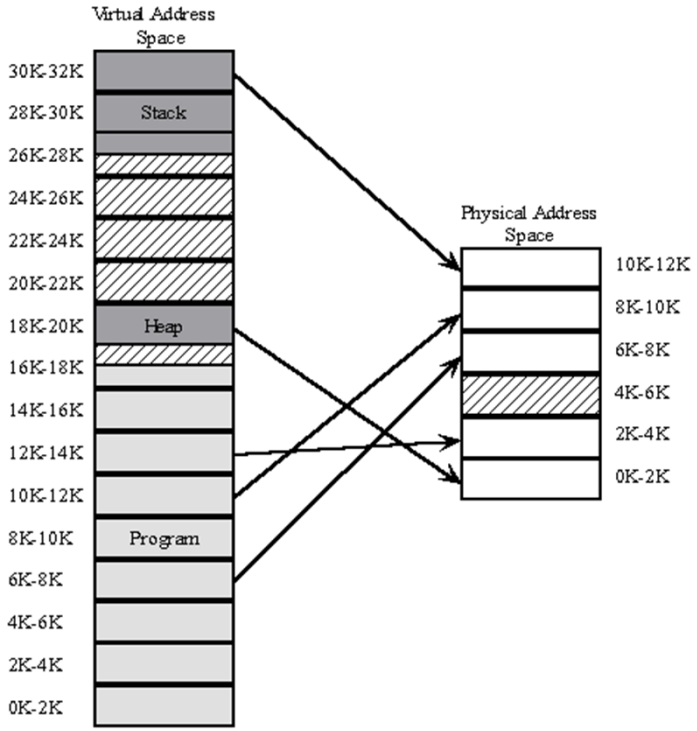


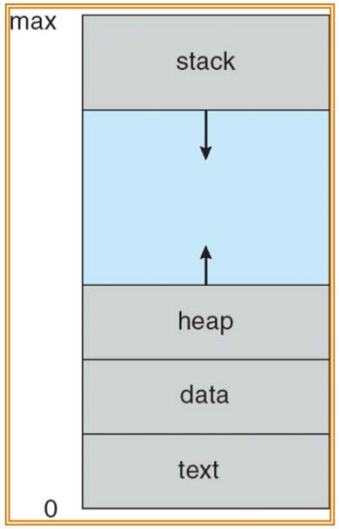
- The communication to cooperate and synchronize processes is called **inter-process communication (IPC)**.

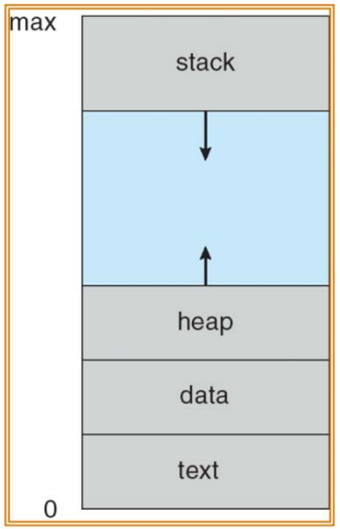


Processes

- **Address Space**
  - One process can't even see another's address space
  - Same pointer address in different processes point to different memory
  - Can change mapping dynamically





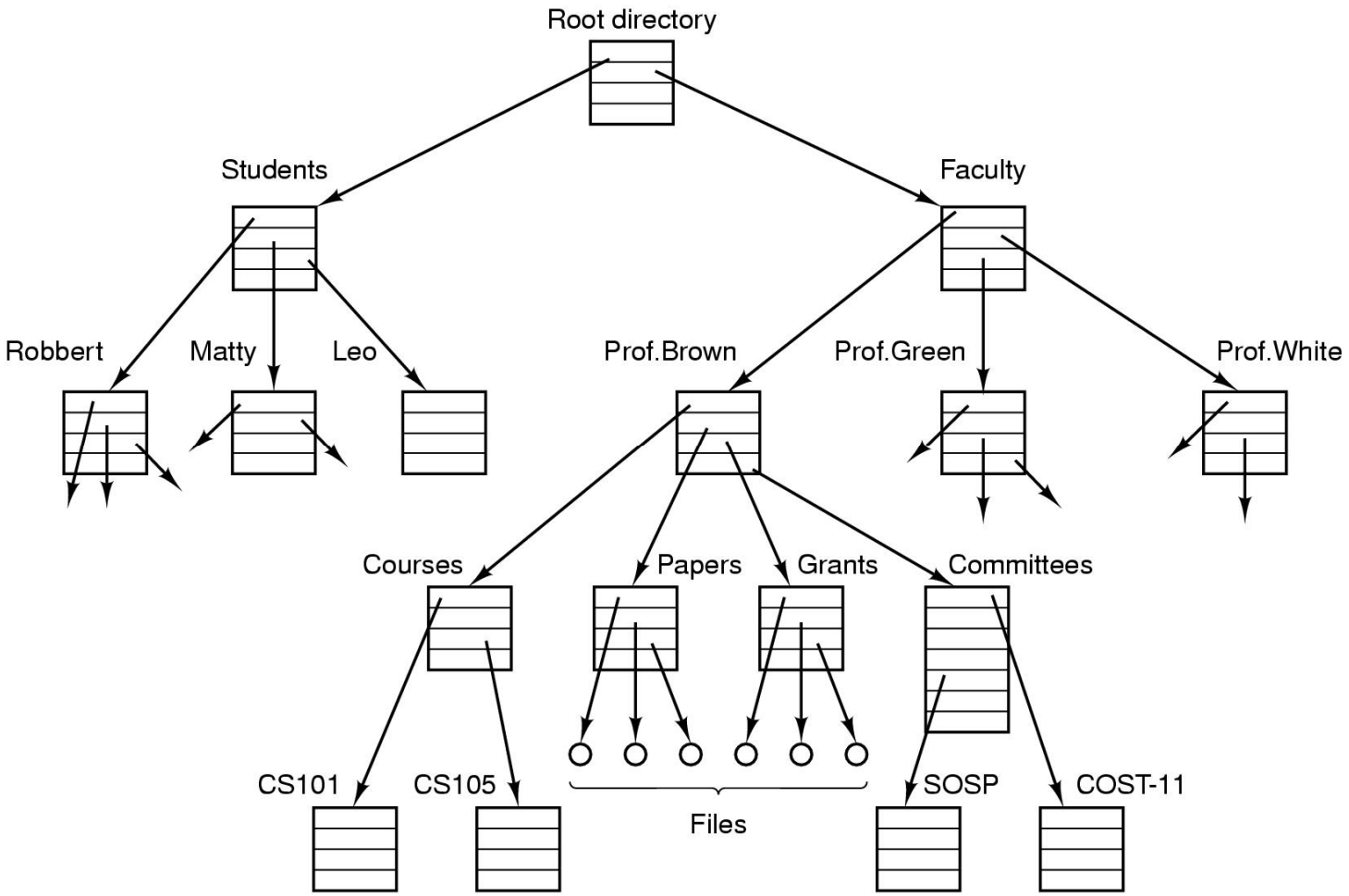


"32-bit processors have 2^32 possible addresses, while 64-bit processors have a 48-bit address space", why?

Files

- A **file system** provides users with a nice, clean abstract model of device-independent files.
- A **directory** is used to group files together.
- A **path name** can specify the location of a file.
- A **root directory** is the top of the directory hierarchy. In Unix/Linux, / is the root directory.
- Each process has a current **working directory** (PWD).

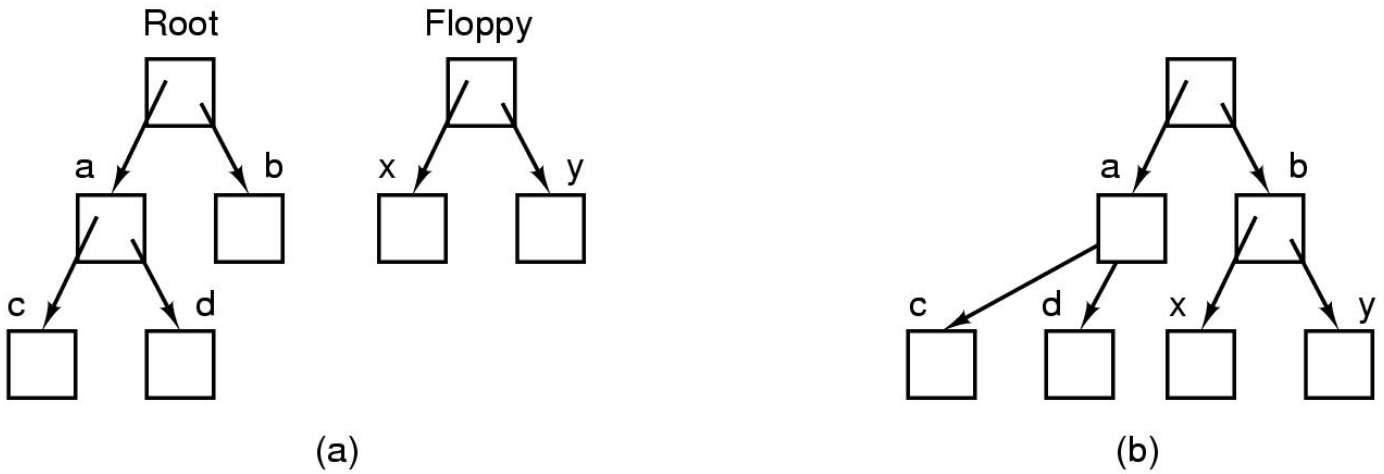
File Hierarchies



An Example File system for a university department

Files

- **File descriptor:** a small integer.
- Mount a file system in removable media.



Before mounting, files on floppy are inaccessible； After mounting floppy on b, files on floppy are part of file hierarchy。

Protection

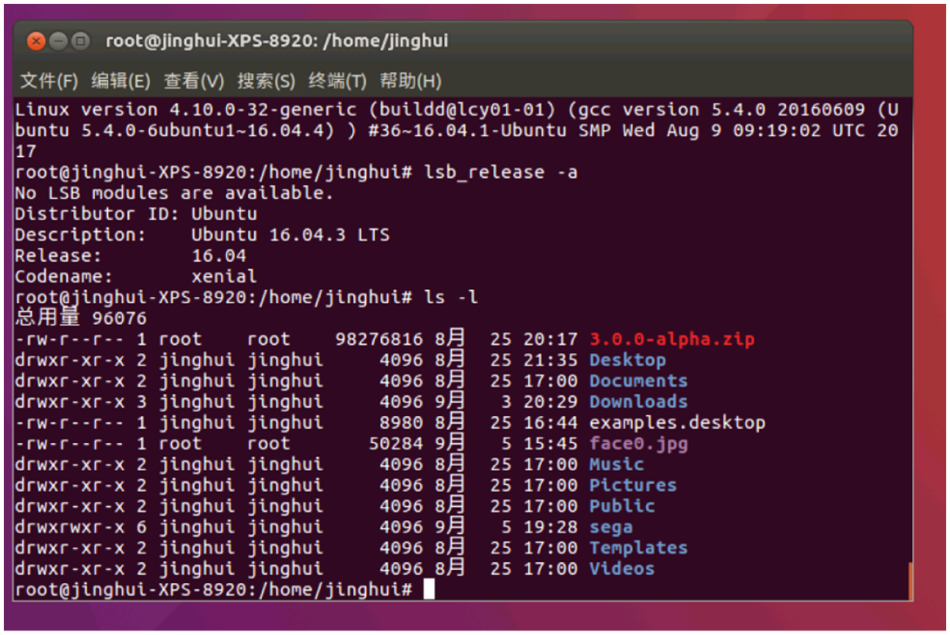
- **Important information: emails, documents, etc.**
- **Task of OS:**

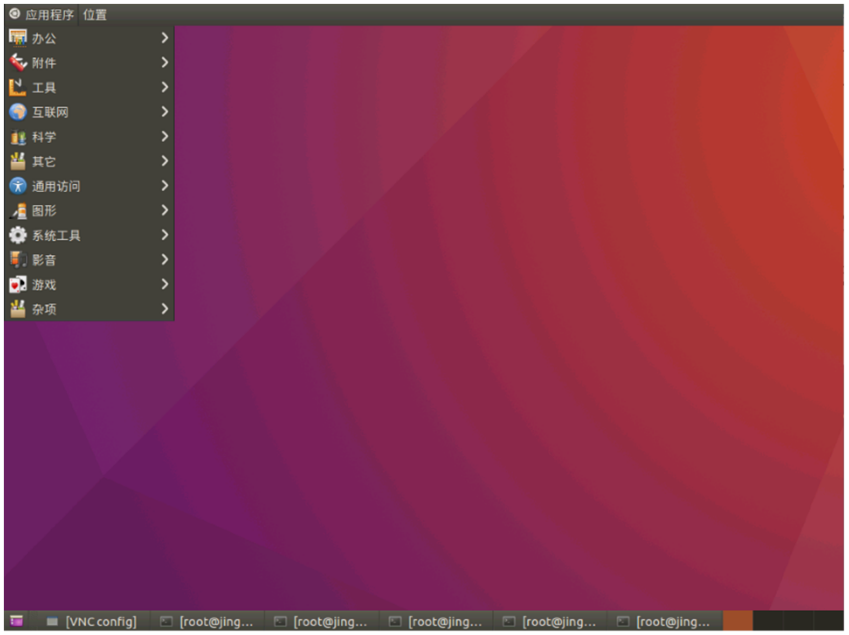
  Ensure files are accessible to authorized users.
- **Files in UNIX are protected by a 9-bit binary protection code.**
  - The protection code consists of three 3-bit fields, one for owner, one for group, and one for others.
  - Example: rwx r-x --x

```
root@jinghui-XPS-8920:/home/jinghui# ls -l
总用量 96076
-rw-r--r-- 1 root    root    98276816 8月  25 20:17 3.0.0-alpha.zip
drwxr-xr-x 2 jinghui jinghui     4096 8月  25 21:35 Desktop
```

Shell

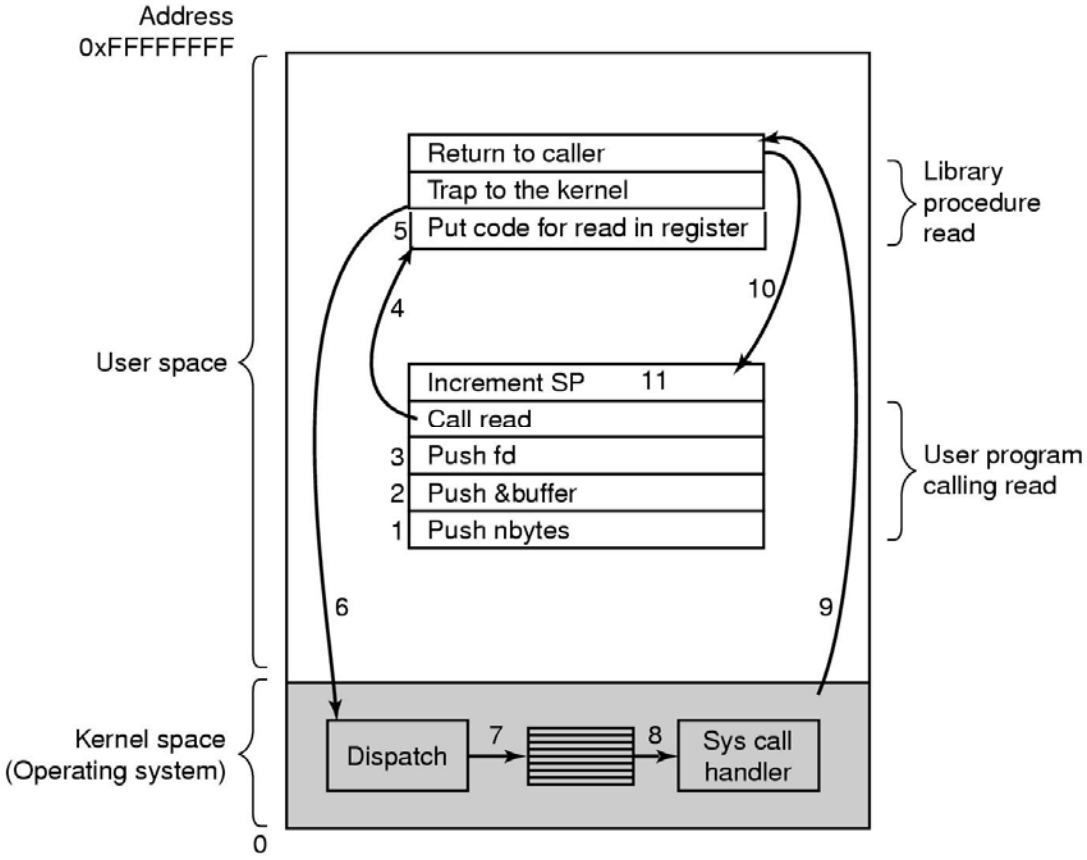
- A shell is a user interface for access to an operating system's services, e.g., *sh*, *bash*, etc.
- Many personal computers use a GUI,
  e.g., KDE, Gnome





System Calls

- The interface between a running program and the OS.
  - In assembly-language instructions.



There are 11 steps in making the system call read (fd, buffer, nbytes)

System Calls For Process Management

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

- ✓ **fork:** to creates a new process.
- ✓ **execve:** to replace the process' memory space with a new program (used after **fork)**.
- ✓ **waitpid:** to move the parent process off the ready queue until the termination of the child.
- ✓ **exit:** be used when a process is finished.
- ✓ #include<unistd.h>

Fork

An Example

```
while (TRUE) {                                  /* repeat forever */
    type_prompt( );                             /* display prompt */
    read_command (command, parameters)                  /* input from terminal */
    if (fork() != 0) {                          /* fork off child process */
          /* Parent code */
          waitpid( -1, &status, 0);                     /* wait for child to exit */
   } else {
          /* Child code */
           execve (command, parameters, 0);             /* execute command */
    }
}
```

Forking Processes

```
main() {
  int i, pid;
  for (i=1; i<=3; i++) {
    if ((pid = fork()) == 0) {
      printf("In child %d. \n", i);
    }else{
      wait(&status);
    }
  }
}
```

System Calls For File Management

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

System Calls For Directory Management

Directory and file system management

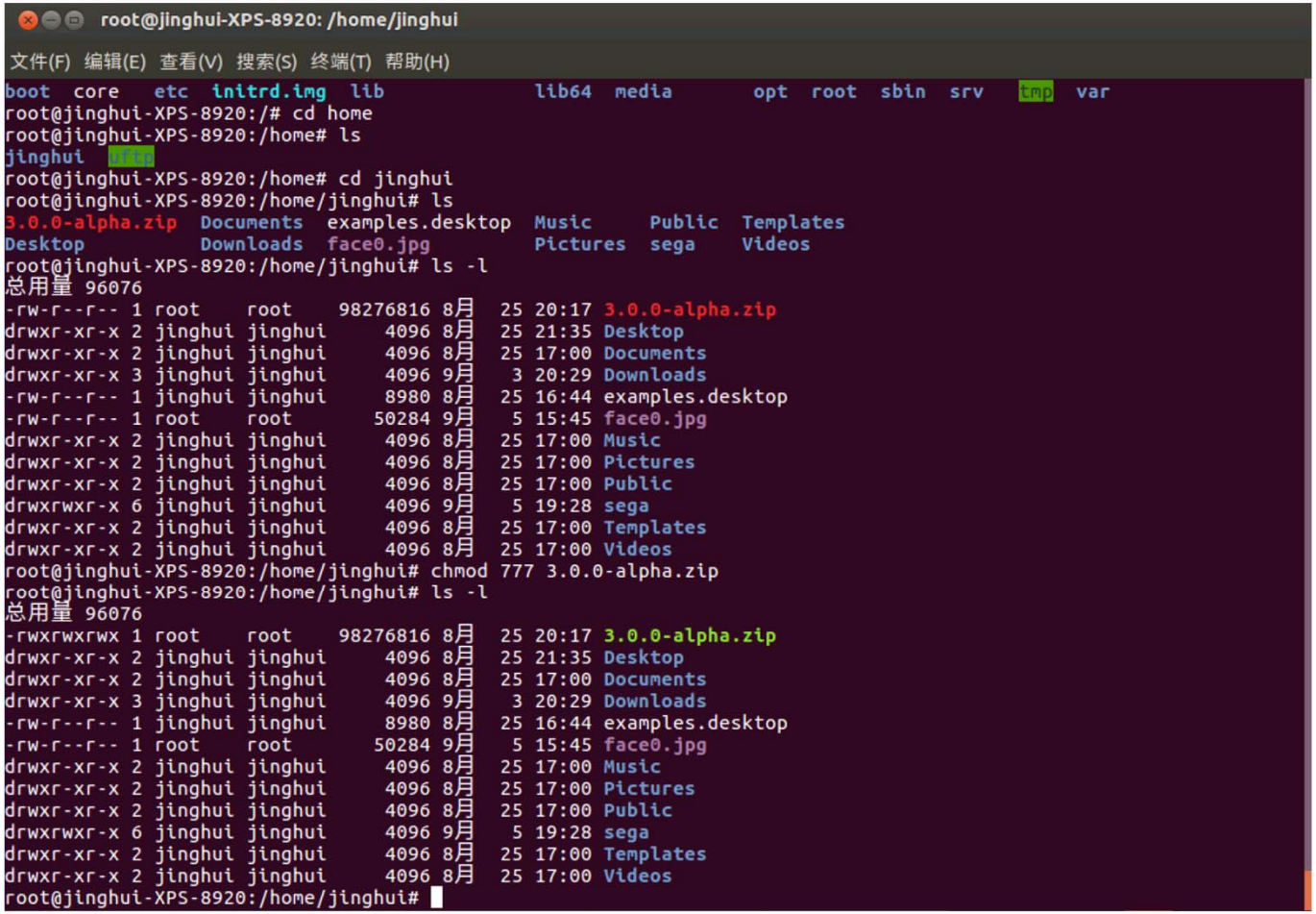
Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

mount(“/dev/fd0”, “/mnt”, 0)

System Calls For Miscellaneous Tasks

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970



System Calls (Win32)

- A window program is usually **even-driven**.
- There is almost a **one-to-one relationship** between the system calls of Unix and Windows.
- Microsoft has defined **Win32 API** （application program interface）to get OS services.
- The **number** of Win32 API calls **is** extremely **large**, not all of them run in kernel mode.

Example Win32 System Calls

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

Operating System Structure

- **Jigsaw reading**

Use one or two sentences to define the follow systems

① **Monolithic Systems**

② **Layered System**

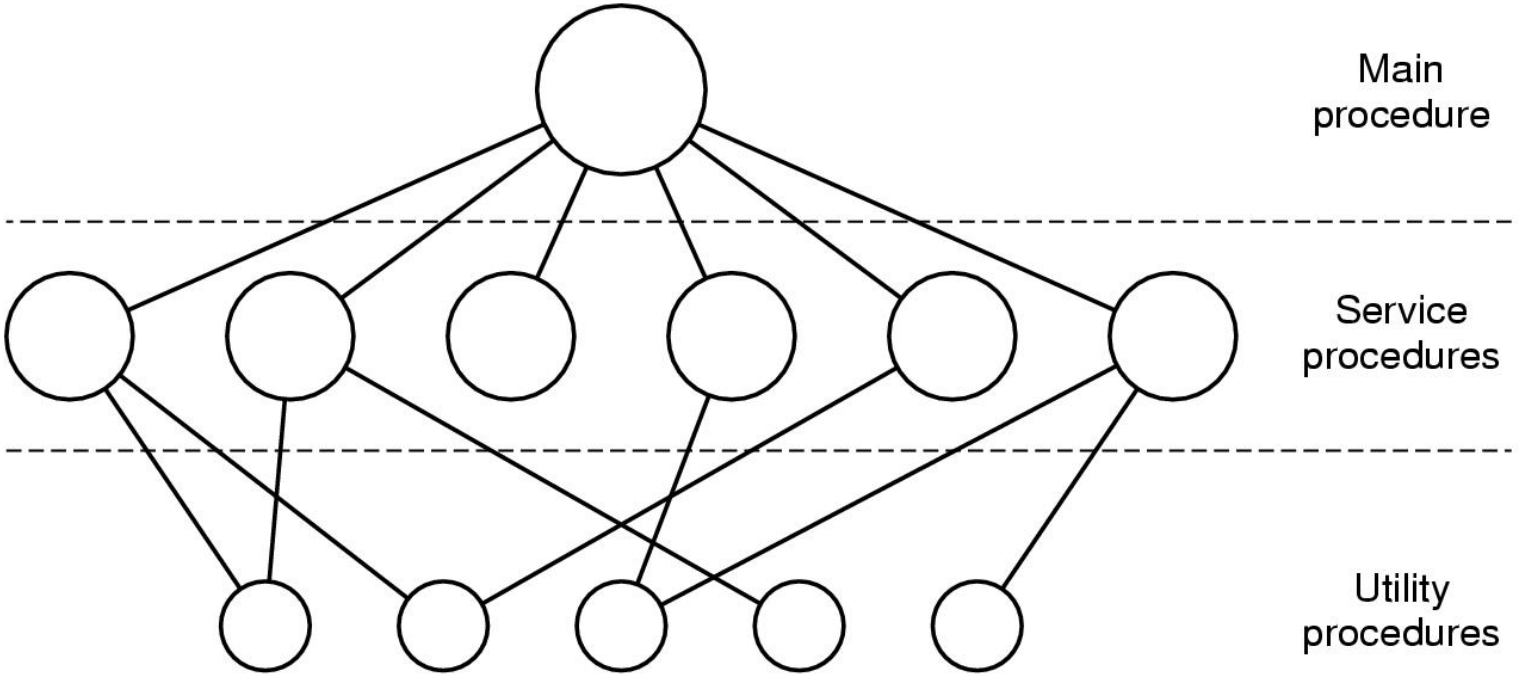
③ **Microkernels**

④ **Client-Server Model**

⑤ **Virtual Machine**

Monolithic System

- All operating system operations are put into a single file. The operating system is a collection of procedures, each of which can call any of the others. (e.g., Linux, windows)

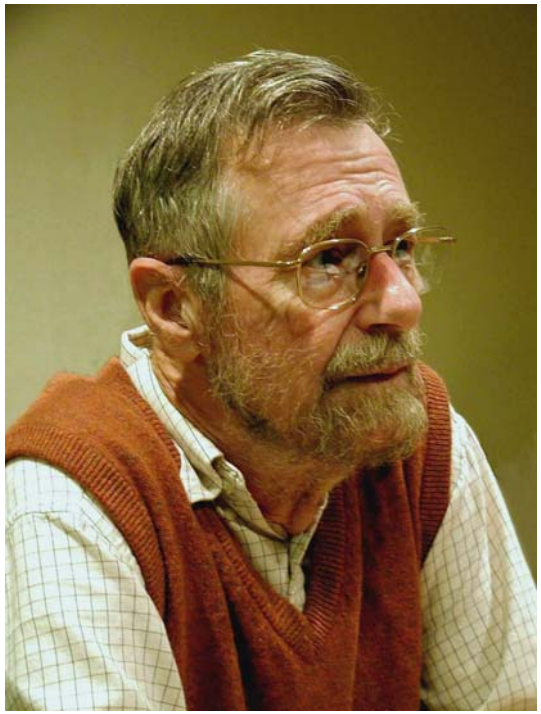


Layered System

- The operating system is organized as a hierarchy of layers of processes.

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

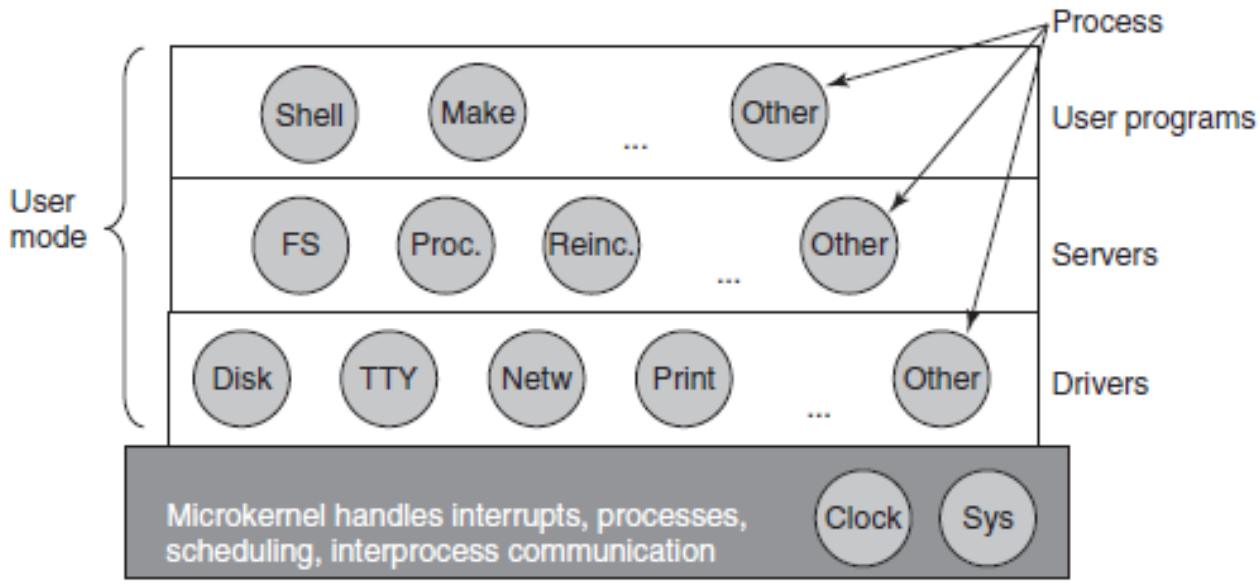
Structure of the “THE” operating system, which was built by E. W. Dijkstra and his students.



Edsger Wybe Dijkstra

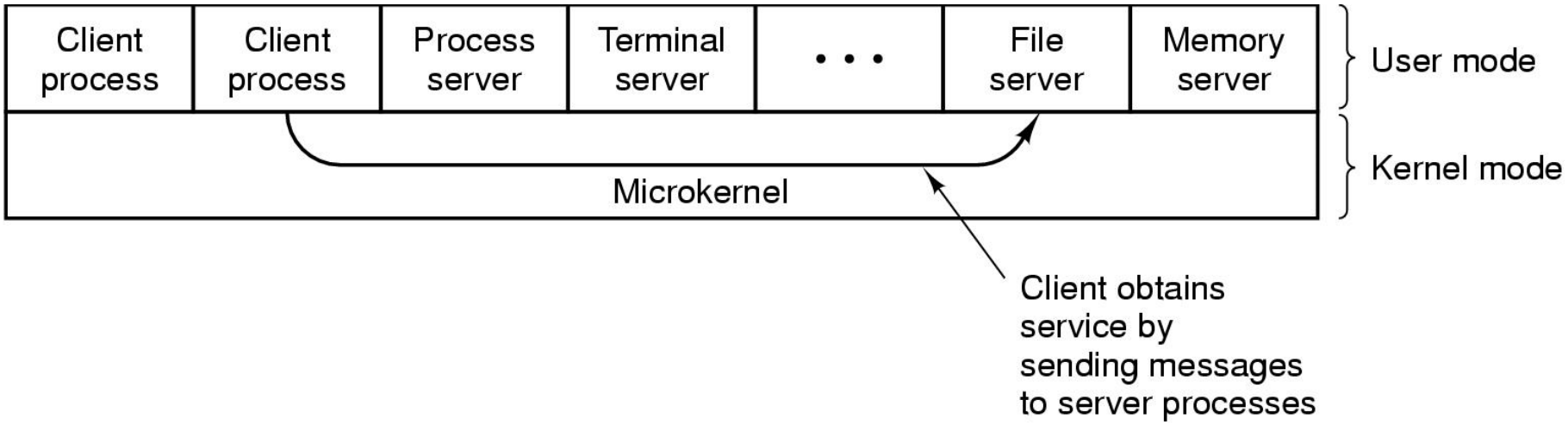
Microkernels

- Split the OS into modules, only one runs in kernel mode and the rest run in user mode; (a lot communication)
- Put as little as possible into kernel model to improve the reliability of the system.
- Examples: MINIX 3



Client-Server Model

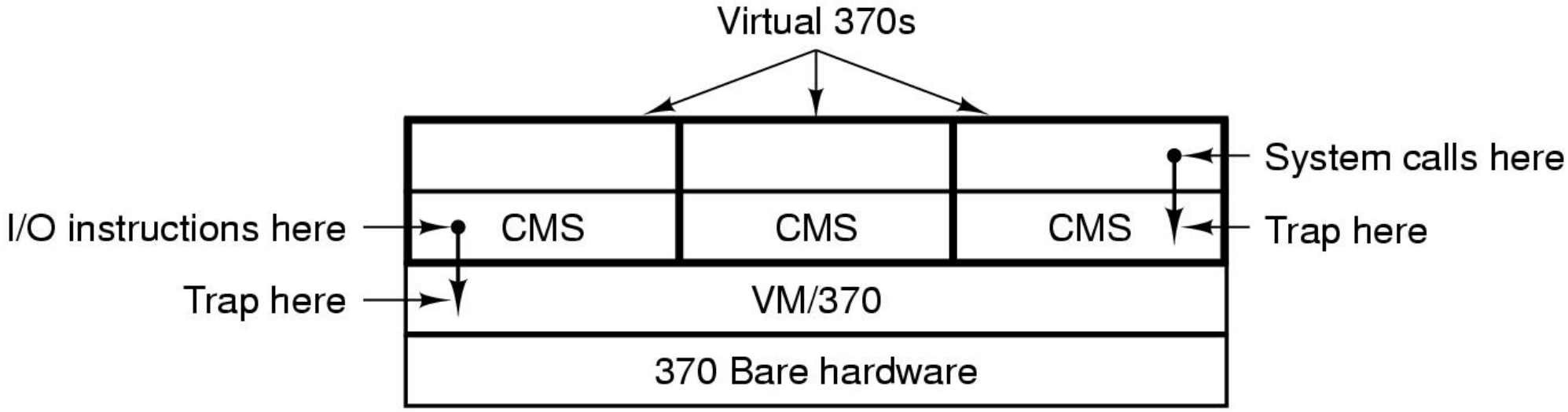
- Contains two classes of processes, the **servers** and the **clients**. Communication between servers and clients is done by message passing. It is an abstraction that can be used for a single machine or a network of machines



The client-server model over a network.

Virtual Machine

- The operating system is a timesharing system that provides multiprogramming； VM monitors are exact copies of the bare hardware, e.g., VM/370, JVM (Java Virtual Machine)



Structure of VM/370 with CMS

Problems

1. What are the major tasks of an OS?
2. What is multiprogramming?
3. A computer has a pipeline with 4 stages. Each stage takes the same time to do its work, namely 1 nsec. How many instructions per second can this machine execute?
4. What is the purpose of a “system call” in an OS?
5. What are monolithic system and microkernel?
6. What is a process?