3.OS Structures

* (A)System Components(依內部來看)

(1) Process management

– Process: an <u>active</u> entity,

a program in execution,

a running program.

需要"社會"資源,含

(a) resources(memory , I/o buffers)

(b) data structure(來描述其動態行為)
to represent current activity , 例

program counter ,
stack ,
data section ,
CPU register.

– (print a1) ≠ (print a2)

– 甚至(print <u>a1</u>) ≠ (print a1)(program≠ process)

– 為了 support process management , OS 必須提供之 services:

～Process creation and deletion

～Process suspension and resumption

～Process communication

～Process synchronization（互送訊息、互等）

～Deadlock handling

(2) Main Memory Management

– Memory: a large array of words or bytes , each with its own address.(CPU 可直接 access)

– Must keep several programs in memory to improve CPU utilization and user response time.

– 為了 support memory management , OS 必須提供：

～ memory usage (記住哪些 memory 被誰使用到) and availability.(可否再 load 一個 job , to where?
free space management —同停車場管理一樣 ；
garbage collection.)

～ decision of memory assignment
(當有空位時，load 誰進來？)

～ memory allocation and deallocation
(無需時，則還回).

(a) keep track of which parts of memory are currently being used and by whom.

(b) decide which process are to be loaded into memory when memory space becomes available.

(c) allocate and deallocate memory as needed.

(3) second-storage management（因為 memory space is too small)

- on-line storage medium for programs and data.
- OS 必須提供之 services—>
  disk management (類似 memory management).
  (a)Free –space management.
  (b)Storage allocation (連續/不連續).
  (c)Disk scheduling.

(4) I/O System Management

- Drivers

(5) File Management

- OS 必須提供之 Services ,含
  (a) File creation and deletion.
  (b) Directory creation and deletion.
  (c) Primitives for file and directory manipulation.
  (d)Mapping of files onto second storage.
  (e)File backup.

(6) Protection System

- (因為有多個 users + concurrent processes)
- Protected resources.含 files , CPU , memory space.
- OS 必須提供 Specification/controlling mechanisms.(for controlling the access of programs/data).
  Ex. Unix ～rwx rwx rwx (d/(u+g+a)+(r/w/x))

(7) Networking (distributed systems).

- For resources sharing.

(8) Command-Interpreter System

- UNIX Shell (between users and OS).

作業系統的內部觀點

以內部觀點而言，作業系統的功用有：

- 資源配置與管理
- 記帳統計
- 資源保護

作業系統是一個 event-driven 的程式（或稱 interrupt-driven）。

Interrupt 的來源有：

- system call }trap
- program error
- I/O device interrupt
- Timer interrupt

當產生 event (interrupt/trap)後作業系統會從 user processes 中的一個（正執行的 process）取回 CPU 的控制權，並執行必要的處理後，經由 process scheduler 選擇其中一個 process，並將 CPU 控制權交出，以執行所選擇的 process。

(B) Operating System Servicers (依外部來看,就是對 programmer/user 而言,OS 必須提供 an environment for execution of programs.)

(1) program execution
loader , linker , compiler , debugger.

(2) I/O operations

– goal: efficiency and protection.

(3) file system manipulation
read , write , create , delete--

(4) communications

- intraprocessor (in the same computer) on interprocessor communication.

(5) error detection

- ensure correct and consistent computing.

(6) OS functions for ensuring the efficient usage of the system itself.

- Resource allocation
  (utilization and efficiency)

- Accounting

- Protection (ensuring that all access to system resources is controlled).
  and security(authenticate himself,證實身份).

作業系統的外部觀點

就外部觀點而言，OS 需提供下列功能：(Services)

1. 程式的執行 (execution environment)
2. 輸出入操作 (I/O)
3. 檔案系統管理 (file management)
4. 錯誤偵測 (debug)

使用方式：

1. System call
2. 系統程式 (或 commands)
   - File manipulation：create, delete, copy, rename, print, list, directory, etc.
   - Status Information：data, time, number of user etc.
   - File modification：text editor etc.
   - Programming language support：Fortran, Cobol, Pascal, Basic.
   - Program loading and execution：link-editor, loader, run etc.
   - Application programs：compiler-compilers, text formatter, database system, statistical analysis packages, etc.

∗ 3.3 System Calls

- Interface between a process and OS
- Parameter passing
  - (a) registers
  - (b) registers pointing to a block
  - (c) stack

(1) --Process and Job Control

- end, abort
- load, execute, e.g., shell load & execute command
- create and / or terminate processes
- get or set process attributes
- wait for time or event, signal event
- profile, tracing, memory allocation & deallocation

(2) --File Manipulation

- Create, delete
- Open, close
- Read, write, response (ex., rewinding)
- Get or set file attributes

(3) -- Device Management

- Request, release
- Read, write, reposition (ex., rewinding)
- Get or set file attributes
- Logically attach or detach devices

(4) --Information Maintenance

- Get or set date or time
- Get or set system data (such as the amount of free memory)

(5) --Communication

- Open, close, accept connections
- Send and receive messages
- Transfer status information
- Map memory

＊3.4 Command Interpreter (A System Program)

(1) Contain codes to execute commands

- fast but tends to be big！

(2) Otherwise

➔ search exec files corresponding to commands (UNIX)

- Issues
  a. parameter passing
  b. slow because of program loading

Command interpreter 的製作方法

製作方法有二：

1. command interpreter 本身就包含各種 codes 以執行有關的 command。
2. 各 command 就相當於一個可執行檔，因此 command interpreter 的工作僅是將 command 所對應執行檔尋找出來，並令其執行。

＊3.5 System Structure

--Layered approach –

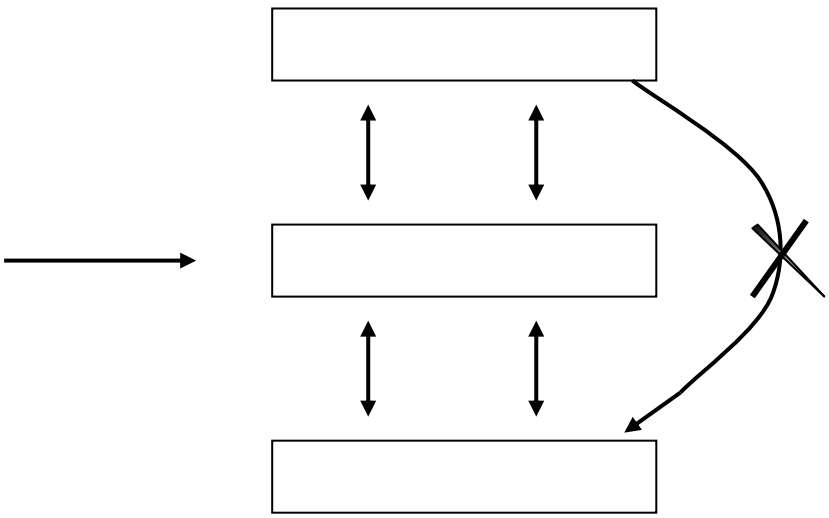
Each layer is implemented using only these operations provided in the lower-layer.

Adv.：modularity ～ debugging & verification

Difficulty：appropriate def. of layers less efficient due to overhead！

L5	User programs
L4	I/O buffering
L3	Operator-Console device driver
L2	Memory Management
L1	CPU scheduling
L0	Hardware

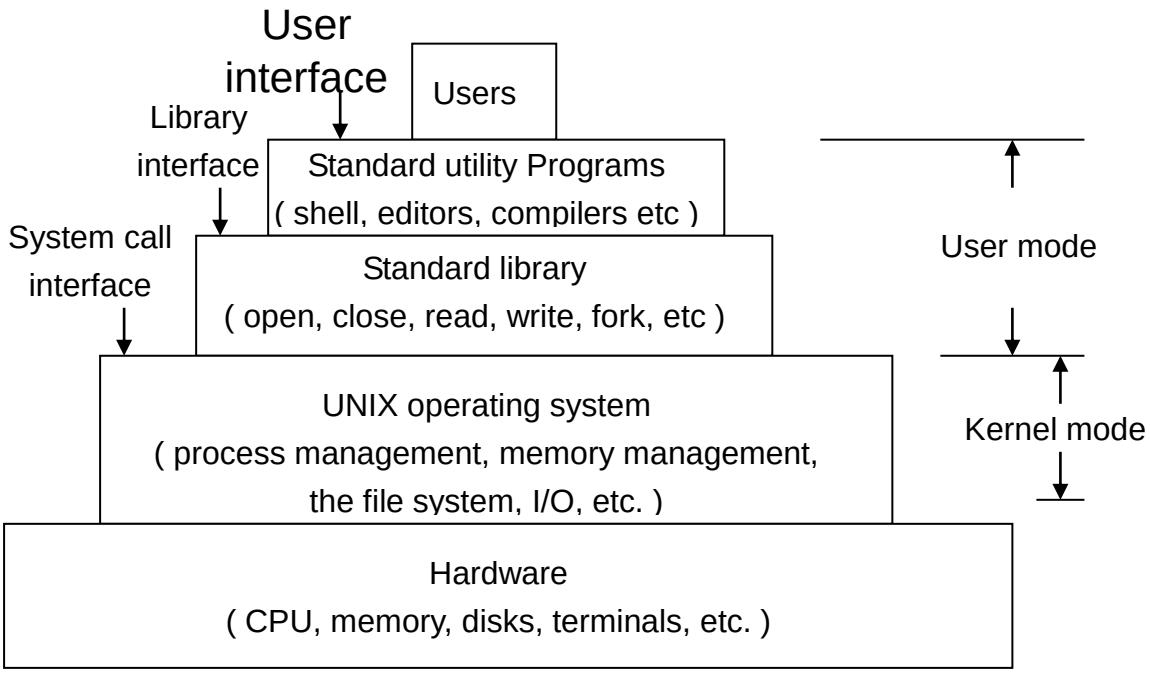
The Layer Structure



(the users)		
Shells and commands Compilers and interpreter System libraries		
System-call interface to the kernel		
Signals Terminal handling Character I/O system Terminal drivers	File system Swapping Block I/O system Disk and tape drivers	CPU scheduling Page replacement Demand paging Virtual memory
Kernel interface to the hardware		
Terminal controllers terminals	Device controllers disks and tapes	Memory controllers physical memory

UNIX system structure

Fig.3.7 (vs. Fig.3.6 MS/DOS, not layered approach)



The layers in a UNIX system

∗3.6 Virtual Machine

∗3.7 system Design & Implementation

-- policy：what will be done

-- mechanism：how to do things