Chapter 3 How to develop Operating Systems

- OS 之 services (課本 p.3-5)
    - 程式的執行
    - I/O 運作
    - 檔案系統的處理
    - Process Communication
    - Error Detection
    - 資源分配(Resource Allocation)
    - Accounting
    - Protection

- OS 之 user interface

	Command Interpreter	System Calls
Def.	作為 users 與 OS 之溝通介面	作為執行中的 Process 與 OS 之溝通介面
目的	1. 接收 user input commands 2. 判斷命令名稱，格式正確與否 3. 如果正確，驅動 command routines 執行，最後結果回傳給 users	執行中的 process，若需要 OS 提供服務(ex, I/O request)，則發出此類中斷，通知 OS，由 OS 執行對應的服務請求(ex, system call)，再將服務結果回傳給 process
種類	Question: Is possible for user to develop a new command interpreter using the system call interface provided by the operating system ? ● Yes, 1. The command interpreter allows an user to create and manage process and also determine ways by which they communicate 2. All of this functionally could be accessed by user-program	1. Process control 2. File operation 3. Device manipulation 4. Information maintain 5. Process communication

	using the system call	6. Protection (8th)
設計 issues		
法 1	所有 command routines 包含在 command interpreter 模組中 優點: 因為 command routines 皆在 Memory 中 所以命令啟動執行較快 缺點: 命令刪減不異 不常用的命令檔，浪費了 memory space	使用 register 保存 優點: 存取速度快(without memory access) 缺點: 若 system calls 之參數數量眾多，則不適用
法 2	所有 command routines 不包含在 command interpreter 模組中 優缺點與法 1 相反	使用 Memory，以 table 的方式儲存參數，並用一個 register 紀錄該 table 的起始，來 place 此 register 給 OS 即可 優點: 適用於參數數目眾多的狀況 缺點: 速度慢(need memory access)
法 3	Command interpreter 模組與 kernel 之關係 • Tightly-coupled 優點: 每個 user 所看到之使用的操作環境/命令意 義是一樣的 缺點: 1. command Interpreter 之改變，連帶影響 kernel 亦須調整	使用 Stack 保存，參數會被 push 到特定的 stack，之後再 pop 取得參數 優點: 可支援 system call 之 recursive 運作及彼此間的呼叫 缺點: 可能要考慮 stack overflow 的情況

	2. 各 user 較無法建立自己專屬的介面	
法 4	Command interpreter 模組與 kernel 之關係 • Loosely-coupled 優缺點與法 3 相反	

- System call, Interrupt and Subroutine call

System Call	Interrupt	Subroutine Call
	大多由 I/O device, CPU, system call 所引起	由 user process 所引起
kernel mode	kernel mode	user mode
會引起 interrupt 及 context switching	可能會伴隨 context switching 由 user process 交給 ISR	process 內部之控制權轉移(不會有 interrupt)
參數轉移較麻煩		參數轉移較 easy

- Compiler v.s. Interpreter
  - Layered Approched
    - 採"Top-Down" Decomposition(分解)的方式
    - 各模組間之呈現以階層方式呈現且"上層可使用下層的功能予以製作，但下層不可使用上層"
    - 採"Bottom-up" testing 方式
      - 優點
        - 易於模組化
        - 可以分工
        - 降低設計複雜度
        - 有助於 Testing Debug
      - 缺點: 精確的層次劃分，極為困難

- Policy && Mechanism

	Policy	Mechanism
Def.	決定 "What to be done" 可能隨時改變	決定 "How to do what" 通常 underlying mechanism 較少改變或不變
Example	Priority scheduling 方法 RR scheduling 函式(Function)	優先權高低的定義 CPU time Quantum 大小之定義 參數值(Value)
原則	Policy 與 mechanism 應該分開(Separate) or 獨立(Independent)	
目的	提升系統修改/調整的 flexibility	

- Virtual Machine

Def.	OS 透過 software simulation 的技術，提供/創造出一份與底層 HW 元件一模一樣的功能介面的虛擬機器 (Abstract machine)
技術	1. 使用 CPU scheduling 技術，創造出其他顆 CPU 的效果 2. 使用 Virtual Memory 擴大 Physical Memory 之假象 3. 利用 Spooling 之技術創造出多套 Virtual I/O Devices
目的	(for programmer) 提供一個良好的測試平台環境對於測試/開發中的 OS
優點	1. OS 測試是在 Virtual machine 上面，所以其他 user process 的工作並不會受到影響 2. 萬一測試中的 OS 毀損，對系統而言也只是毀了一個 Process，並不會對 real machine, real OS 及其他 user process 造成影響 3. 同一部 machine 上面可以執行多套 OS

缺點	1. Virtual machine 之製作極為困難 2. Virtual machine 之 performance 比 Real machine 差
Note	Virtual Machine -> user mode of real machine Virtual Machine software -> kernel mode of real machine

- Java Virtual Machine(JVM)
  - (筆)
  - Byte code 之特性
    i. 是 JVM 的 object code
    ii. 與任何機器無關(Independent)
    iii. 可跨不同平台，可攜性極高
    iv. Byte code 須經由"Java interpreter" 之解譯在 target CPU 執行，然而 Java interpreter 之 performance 並不好，所以 Sun Microsystem 又推出"Just-in-time compiler"(JITC)，可即時將 Byte code 轉譯成特定 cpu 之 object code，以提升效能
  - JVM 包含三部份
    i. Class loader
    ii. Class verifier
    iii. Java interpreter

- Microkernel

Def.	將 kernel 中，較不基本/重要的 services 自 kernel 中移除，改成以在 user mode 執行的 system SW or Library 方式來提供服務，如此可以得到較小的 kernel，稱之 一般而言，Microkernel 所保留下的 essential service 有三: 1. Process Management 2. Min Memory Management

	3. Process Communication(Message passing)
優點	1. 安全性高(因為大部分 service 皆在 user 端執行，所以萬一 service fails，也不會對系統造成重大危害) 2. Service 之措施刪除方便、容易(因為加入的 service 是在 user 端，所以 kernel 不需改變) 3. 移植性高(從一平台移至另一平台，microkernel 的修改較方便，因為元件不多)
缺點	與 monolithic kernel 相比(所有 device 皆在 kernel 執行)，效能較差
Note	➢ 不在 microkernel 提供的服務 Virtual memory, Disk Management, I/O subsystem Management, File Management, Security Network, etc. ➢ the overheads associated with inter-process communication and frequent use of operating systems' messaging functions in order to enable the user process and the system services to interact with each other ➢ How do user program and system device interact in a microkernel system ? ➔ usually communicate by message passing

- Compiler v.s. Interpreter

Compiler	Interpreter
Source code 整批翻成 object code 之後即離開 memory, 不佔用 memory space	翻譯一行執行一行，程式執行時 Interpreter 必須駐留在 memory 中
Object code 執行較有效率，因為有作 code optimization	反之
會產生 Object code	只會產生中間碼
翻譯速度慢	快
Debug 不易	易
程式重新執行不需 re-compiling	需要再作 Interpreting

※List five major activity of an operating system in regard to file management ?

1. the creation and deletion of files
2. the creation and deletion of directories
3. the support of primitives of manipulating files and directories
4. the mapping of files onto secondary storage
5. the backup of files on stable storage media

※what are advantages and disadvantages of using the same system-call interface for manipulating both files and directories ?

優點：the development of both user program, which can be written to access devices and files in the same manner, and device driver code, which can written to support a well-defined API.

缺點：it might be difficult to capture the functionality of certain devices within the context of file access API. thus either resulting in a loss of functionality and performance.