資訊系統原理

郭大維教授
臺灣大學資訊工程系

Contents

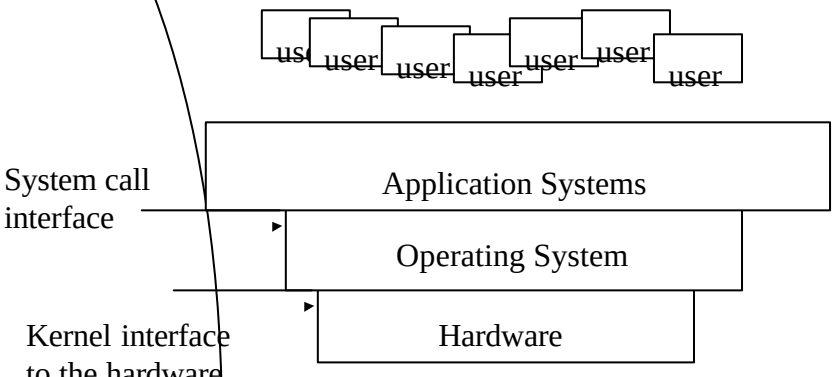
- Computer Systems Overview
  - Operating Systems Concept
  - UNIX
  - Other System Services
- Unified Modeling Language
  - UML Introduction
  - System Development Process
  - Use Cases, Class Diagrams, etc.

Operating Systems Concept

- What is an operating system?
- Operating system architecture
- Process concept
- CPU scheduling
- Memory management
- File and I/O systems
- Networking

What is an operating system?

- What is an operating system?
  - A package of software called OS!



* "Operating system concepts", Silberschatz and Galvin, Addison Wesley, pp. 3-5

What is an operating system?

- A control program
  - Control the execution of user programs
  - Prevent errors/misuse
- An environment for efficient/ convenient usage of a computer system.
- A resource allocator
  - CPU, memory space, file storage, I/O devices, shared code, data structures, etc.

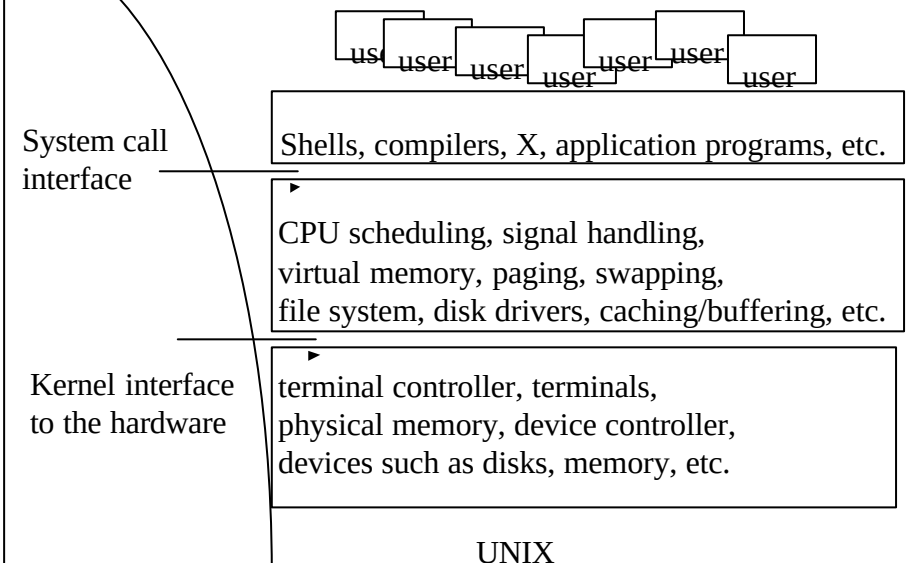
What is an operating system?

- Terminology
  - Multiprogramming
    - CPU/job scheduling – short/mid/long-term
  - Time-sharing
    - Multiprogramming + CPU switching ~ interactivities
  - Batch processing
    - Job pool – with/without multiprogramming
  - Spooling (Simultaneous Peripheral Operation On-Line)
    - Printf -> buffer (memory/disks) -> printout at a printer.
    - Card readers -> disks -> run process

Operating Systems Concept

- What is an operating system?
- Operating system architecture
- Process concept
- CPU scheduling
- Memory management
- File and I/O systems

OS Architecture



UNIX

OS Architecture

- What components/functionality it has?
  - Process Management
    - Creation/deletion/suspension/resumption of user/system processes
      - A process is a program in execution.
    - Process scheduling
    - Mechanisms for process synchronization
    - Interprocess communication mechanisms
  - Memory Management
    - Memory allocation/deallocation
    - Paging/segmentation memory management

* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 49-54

What is an operating system?

- File Management
  - Creation/deletion of files/directories
  - Mapping of files to secondary storage
- I/O Systems & Storage Management
  - Hide the peculiarity of specific H/W devices from users
  - Storage allocation and management
  - Disk scheduling
- Networking
  - Various networking service such as naming resolution
- Protection System
  - CPU, Memory, I/O devices

OS's on Parallel/Distributed Systems

- Parallel Systems
  - More than one processor in close communication, sharing of bus, clock, sometimes memory and peripheral devices – tightly coupled systems!
  - Symmetric/asymmetric operating systems.
- Distributed Systems
  - More than one processor without sharing of memory or any clock – loosely coupled systems!
  - Heterogeneous vs homogeneous systems!

* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 14-17

Real-Time OS

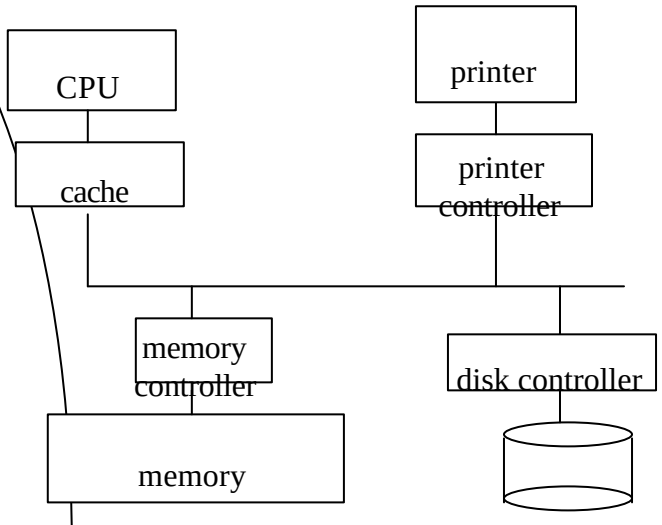
- Why RTOS
  - A convenient and reliable environment to develop time/safety-critical applications.
- Requirements – depending on applications!
  - Predictability – Verifiability & interrupt latency
  - Reliability - Strictness of Deadline Violations
  - Reconfigurability - System Size and Functionality
  - Efficiency of System Components - Time Granularity, Threads, and Resource Management
  - Variable Models of Task Communication - Characteristics of Applications

What is an operating system?

- Roadmap
  - Booting
  - I/O Structure
  - Storage Hierarchy
  - etc

* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 24-30, 35-37

Computer System Architecture

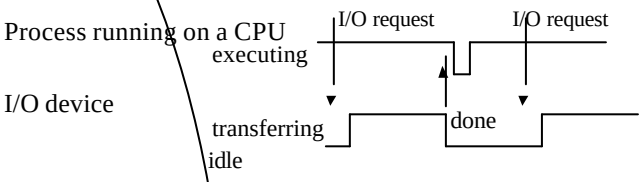


Booting

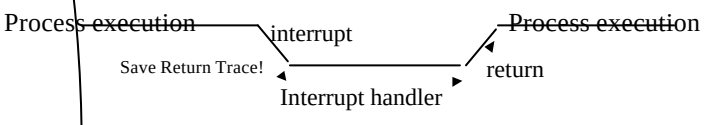
- Bootstrap program
  - Initialize all aspect of the systems
    - E.g., CPU registers, device controllers, memory, etc.
  - Load OS, and Run it!
    - Run init to initialize system services
      - Start virtual memory, various daemons, login processes, etc.

I/O Structure

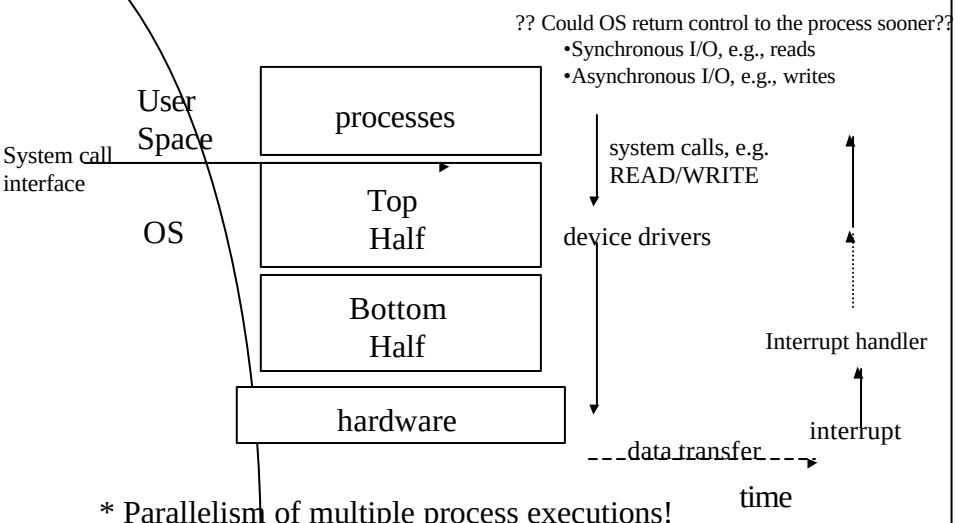
- Parallelism of CPU and I/O activities



- Interrupts

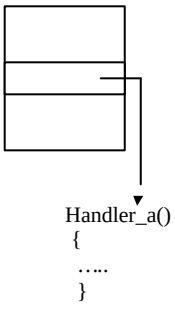


I/O Structure



* Parallelism of multiple process executions!

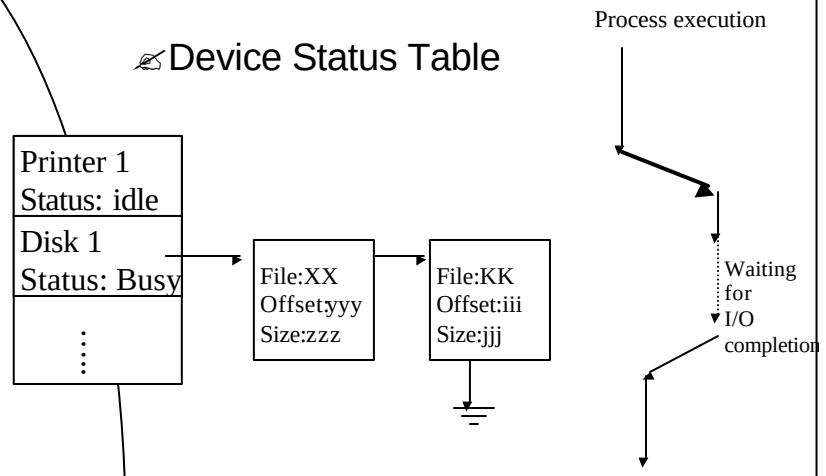
I/O Structure



- Interrupt types:
  - Software interrupts (traps)
    - Signals, division-by-zero, etc.
  - Hardware interrupts
    - Service requests of I/O devices, etc.
- Servicing of interrupts
  - Generic handler
  - Interrupt vector (UNIX)
- Masks, Disabling, Enabling
  - bits to mask interrupts

I/O Structure

- Device Status Table

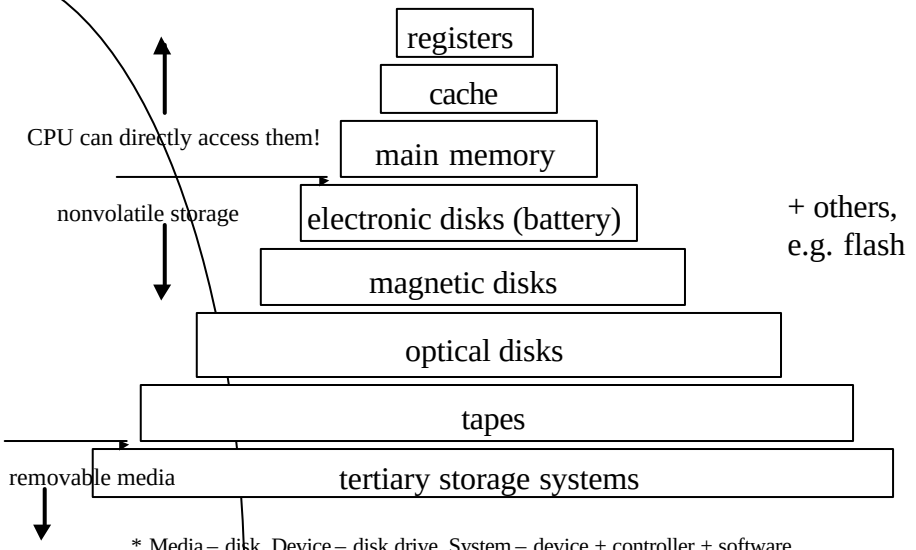


I/O Structure

- DMA (Direct Memory Access)
  - Release CPU from handling excessive interrupts
    - e.g., a high-speed device:
      - 2us service / 4us
  - Procedure, e.g., WRITE
    1. Use device driver to set up the registers of the DMA controller.
    2. DMA moves blocks of data between memory and its own buffers.
    3. Transfer from its buffers to its devices
    4. Interrupt the CPU

Until done

Storage Hierarchy



* Media – disk, Device – disk drive, System – device + controller + software
* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 24-30, 35-37

I/O and Storage Systems

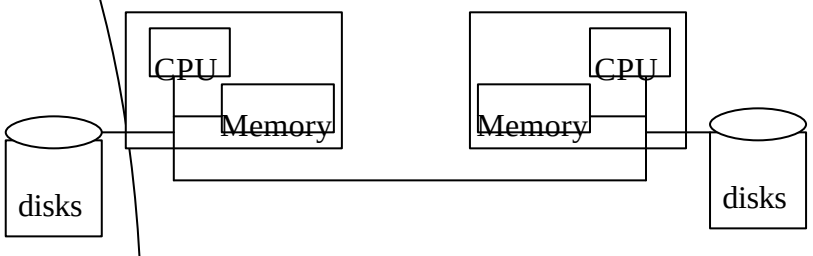
- Cost/bit vs Capacity/dollar vs volatility
- Device I/O
  - Memory-Mapped I/O
    - For devices with fast response time
  - Program I/O (PIO)
    - polling
  - Interrupt-Driven I/O

Storage Hierarchy

- Caching
  - When an information is used and might be used again, it is cached at a faster storage system.
  - Strategies at different levels?
- Buffering
  - When an information is “pushed” out to a slower storage system, it is buffered at a faster system for later actions.
  - Relationship with caching?

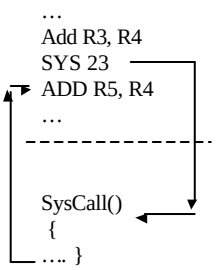
Storage Hierarchy

- Coherency and Consistency
  - Vertical information flow
  - “Horizontal” information flow



Dual Mode Protection

- What is “dual mode”?
  - User mode
  - Kernel mode
    - Privileged instructions, such as I/O, memory-setting related instructions
- Rationale
  - Only execute privileged instructions at the kernel mode to protect errors and misuse!
- Requirement
  - Hardware support

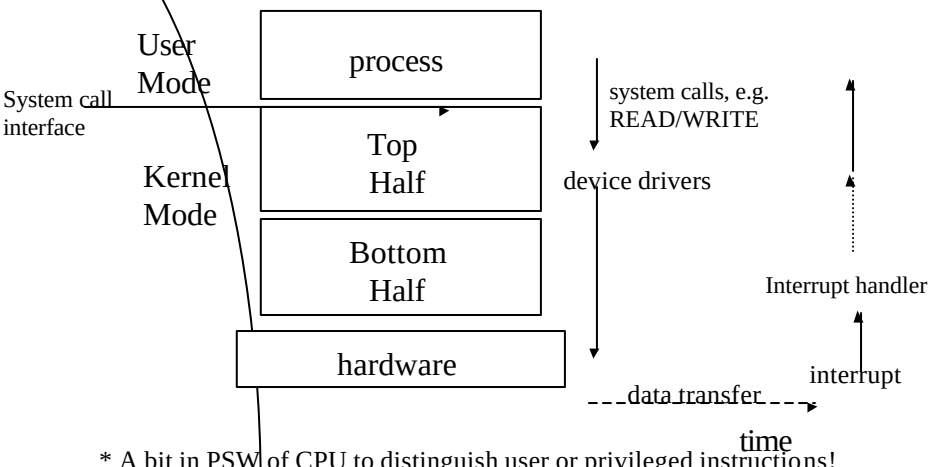


* “Operating system concept”, Silberschatz and Galvin, Addison Wesley, pp. 39

Booting - Revisiting

- Bootstrap program – Kernel and Single User Mode
  - Initialize all aspect of the systems
    - E.g., CPU registers, device controllers, memory, etc.
  - Load OS, and Run it! – Kernel and Multi-User Mode
    - Run init to initialize system services
      - Start virtual memory, various daemons, login processes, etc.
  - Shell Processes – User and Multi-User Mode

I/O Structure Revisiting



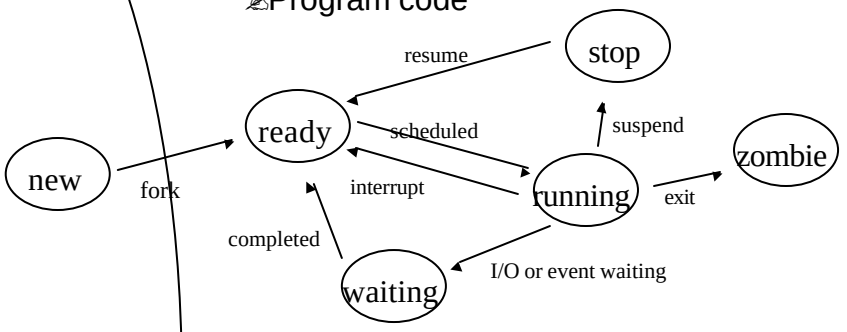
* A bit in PSW of CPU to distinguish user or privileged instructions!

Operating Systems Concept

- What is an operating system?
- Operating system architecture
- Process concept
- CPU scheduling
- Memory management
- File and I/O systems

Process Concept

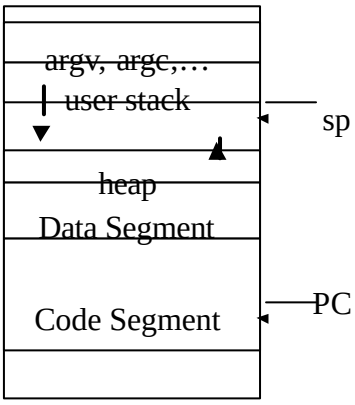
- Process
  - An active entity
  - The corresponding passive entity
    - Program code



* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 89-97, p 126.

Process Concept

- Process Image in Memory



- System Resources
  - Program Counter (PC)
  - Process Status Register
  - Stack Pointers (sp)
  - Memory
  - etc.

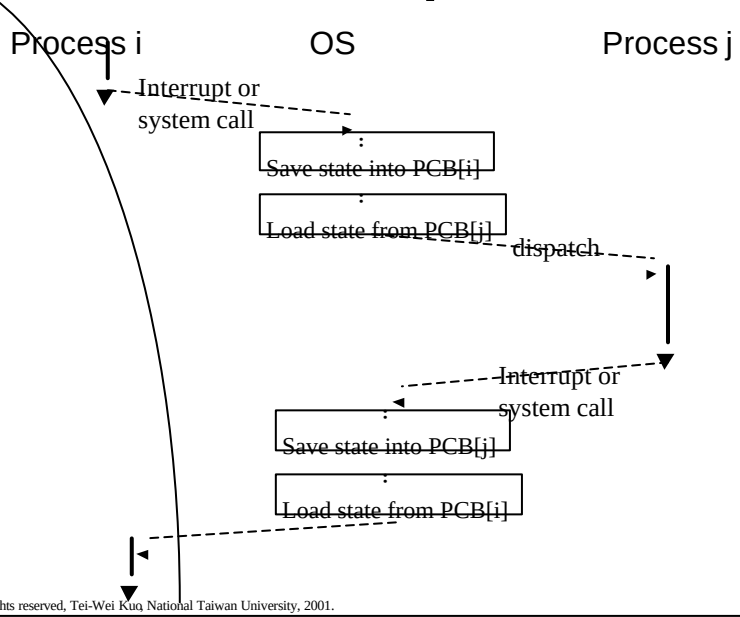
Process Concept

- Process Control Block

```
Struct PCB {
   char p_pid;
   char p_pri;
   char p_ppid;
   int pc; /* program counter */
  …
   int rq_former, rq_next; /* for ready queue*/
   int files[NFILE];
} PCB[NPROC];
```

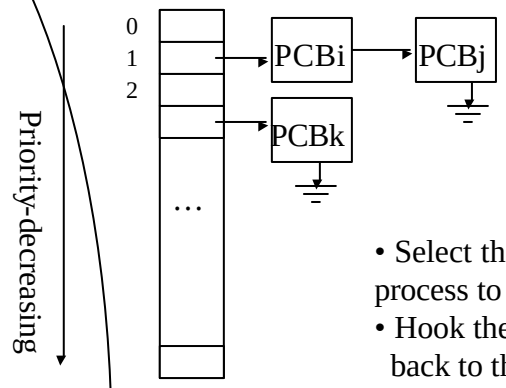
- major queues:
  - I/O queues
  - ready queue

Process Concept



Process Scheduling

- A Ready Queue



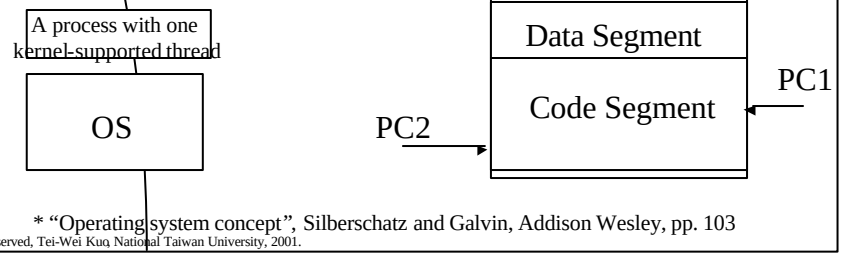
• Select the highest-priority process to run and de-queue it!
• Hook the running process back to the ready queue!

Context Switching – Necessary Overheads for Multiprogramming

- Def.
  - Switch the CPU from one process to another process
    - Save the state (or called context) of the running process
    - Reload the state of the ready process
- Context Switching Time
  - Hardware-dependent!
    - Multiple register sets!
    - Special hardware instructions!

Process Concept

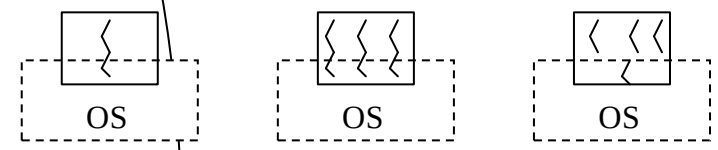
- Kernel-Supported Thread
  - A program counter
  - A register set
  - A stack space
- User-Level Thread



* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 103

Process Concept

- Thread – Lightweight process
  - An ordinary process contains one thread!
  - Types
    - Kernel-Supported Threads
    - User-Level Threads



Ordinary process

3-kernel-supported threads in a Process

3-user-level threads in a Process

Operating Systems Concept

- What is an operating system?
- Operating system architecture
- Process concept
- CPU scheduling
- Memory management
- File and I/O systems

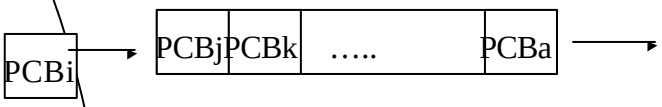
CPU Scheduling

- Scheduling Criteria
  - CPU Utilization
  - Throughput = Number-of-Completed-Processes / Sec
  - Turnaround Time
    - Completion time – Submission time
  - Waiting Time
    - Time spent in the ready queue
- Average vs Worst-Case vs Combination
  - variance

* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 127-130..

CPU Scheduling

- First-In-First-Out (FIFO) Scheduling

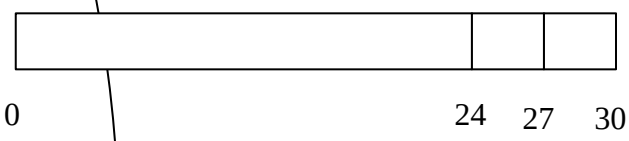


Ready Queue

- Select the first process in the queue for execution!
- "Fair" but lengthy waiting for urgent processes! -> Shortest Job First??

CPU Scheduling – FIFO

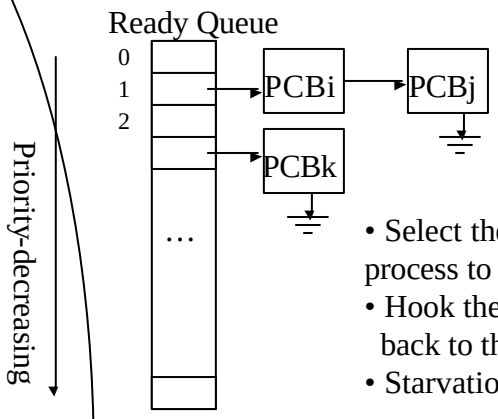
Processes	Burst CPU Time	Arrival Time
P1	24	0
P2	3	1
P3	3	1



Average Waiting Time = (0 + 23 +26) / 3

CPU Scheduling

- Priority-Driven Scheduling



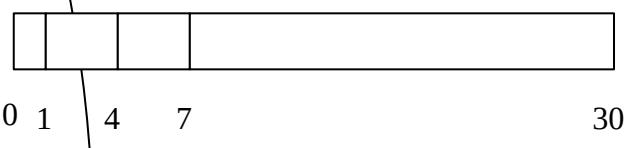
- Select the highest-priority process to run and de-queue it!
- Hook the running process back to the ready queue!
- Starvation problem?!!

* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 133-137.

CPU Scheduling – Priority

- Processes | Burst CPU Time | Arrival Time | Priority

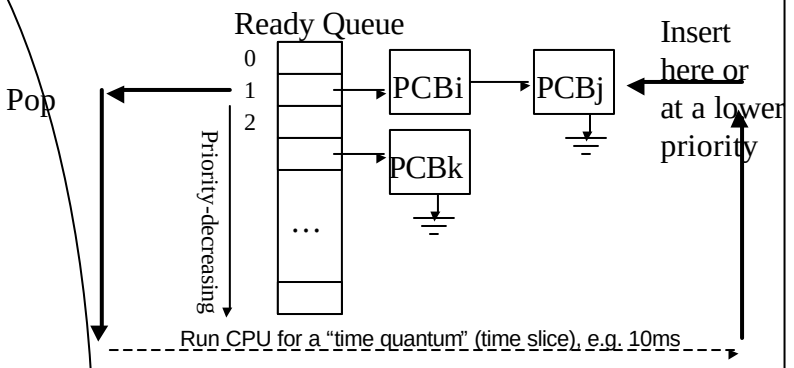
P1	24	0	3
P2	3	1	2
P3	3	1	1



Average Waiting Time = (6 + 0 +3) / 3

CPU Scheduling

- Round-Robin Priority-Driven Scheduling



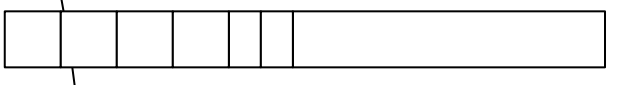
- N processes, quantum = ?, (N * ?) cycle time!
- How small ? should be? Any limitation?

CPU Scheduling – RR

- Processes Burst CPU Time Arrival Time Priority

P1	24	0	1
P2	3	1	1
P3	3	1	1

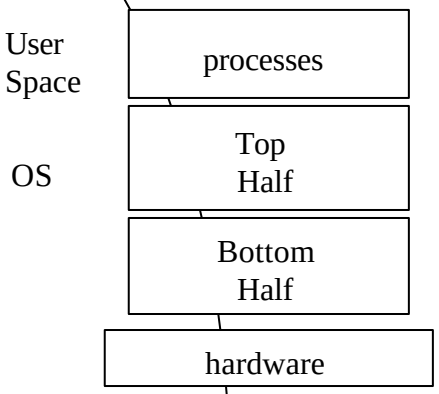
Quantum = 2



0 2 4 6 8 9 10 30

Average Waiting Time = (6 + 5 +6) / 3

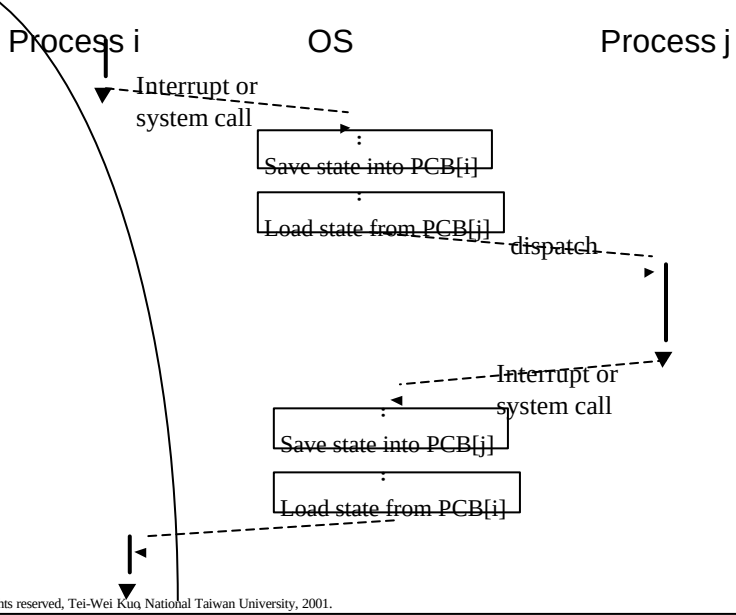
CPU Scheduling



System calls such as I/O req which may cause the releasing CPU of another process!

Timer expires to
- Expire the running process's time quota
- Keep the accounting info for each process

Process Concept - Revisiting

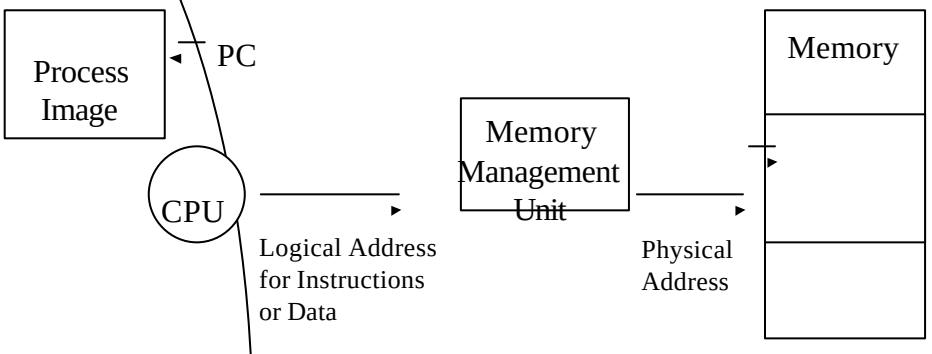


Operating Systems Concept

- What is an operating system?
- Operating system architecture
- Process concept
- CPU scheduling
- Memory management
- File and I/O systems

Memory Management

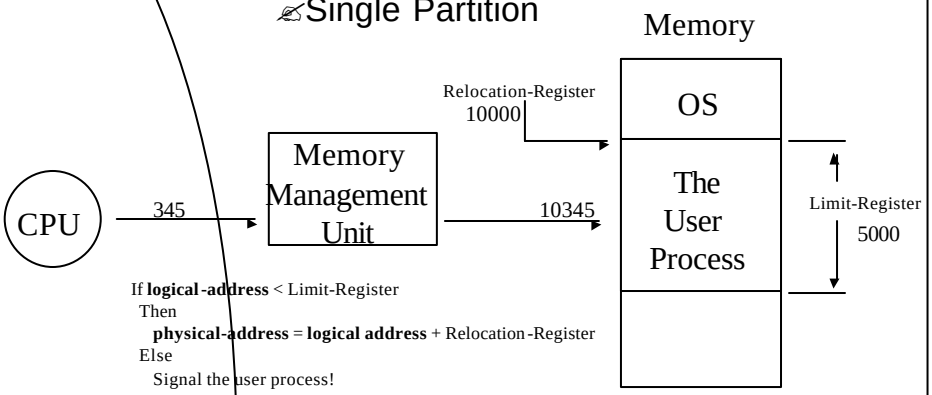
- Logical address vs Physical Address



* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 245-246.

Memory Management

- Contiguous Allocation
  - Single Partition

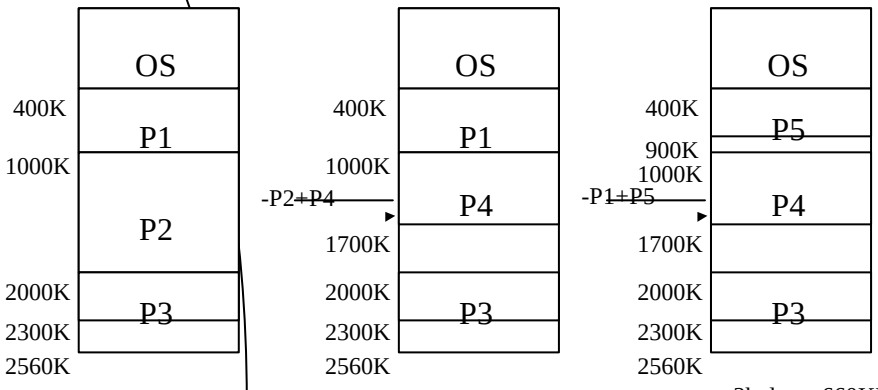


If **logical-address** < Limit-Register
Then
physical-address = **logical address** + Relocation-Register
Else
Signal the user process!

* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 249-264.

Memory Management

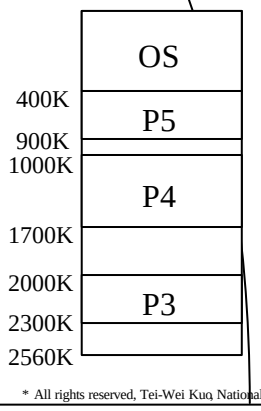
- Contiguous Allocation
  - Multiple Partitions

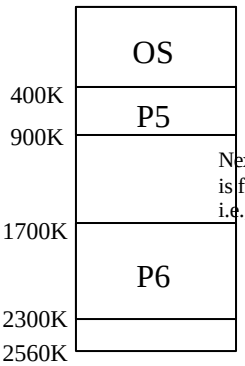


3holes = 660KB!!!

Memory Management

- Contiguous Allocation
  - Multiple Partitions
    - First-Fit
    - Best-Fit
    - Worst-Fit
  - Fragmentation
    - External
    - Internal

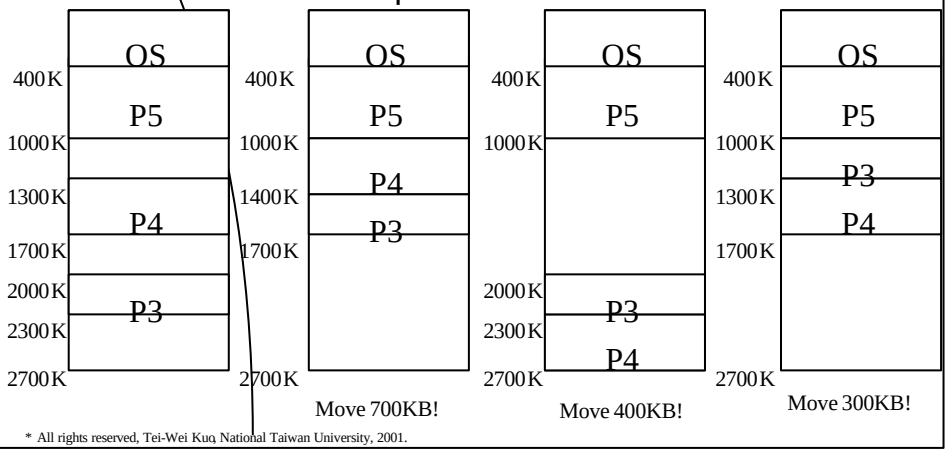




Next request is for 819000B, i.e., 800KB-200B

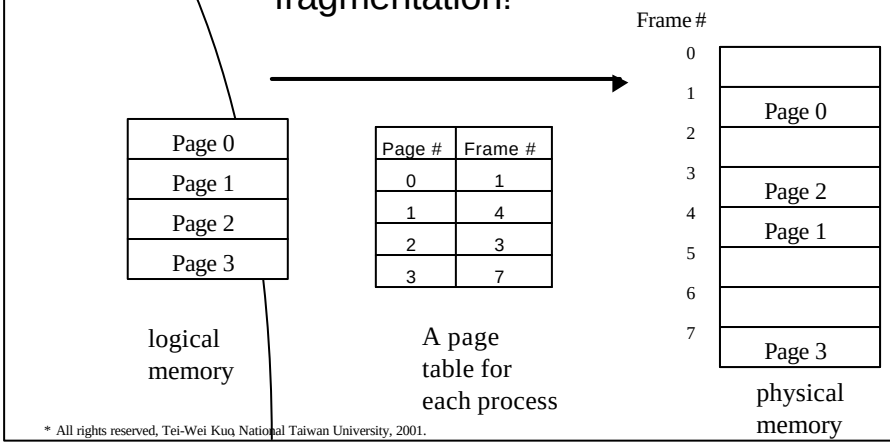
Memory Management

- Solutions to Fragmentation
  - Compaction



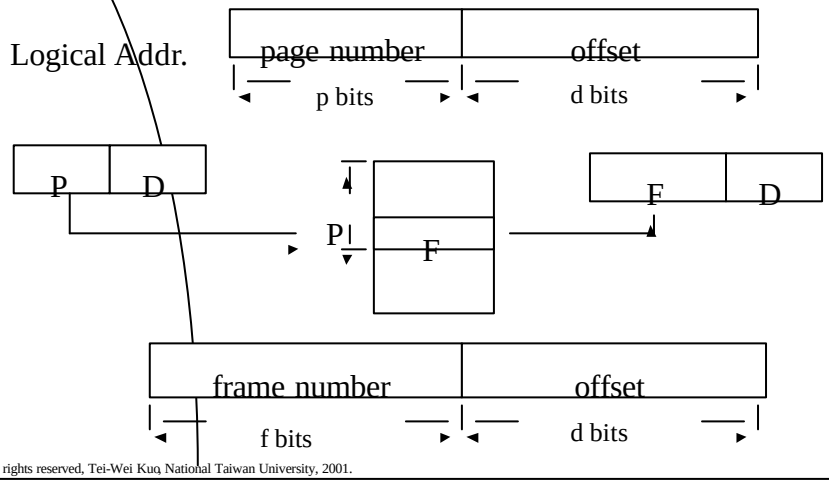
Memory Management

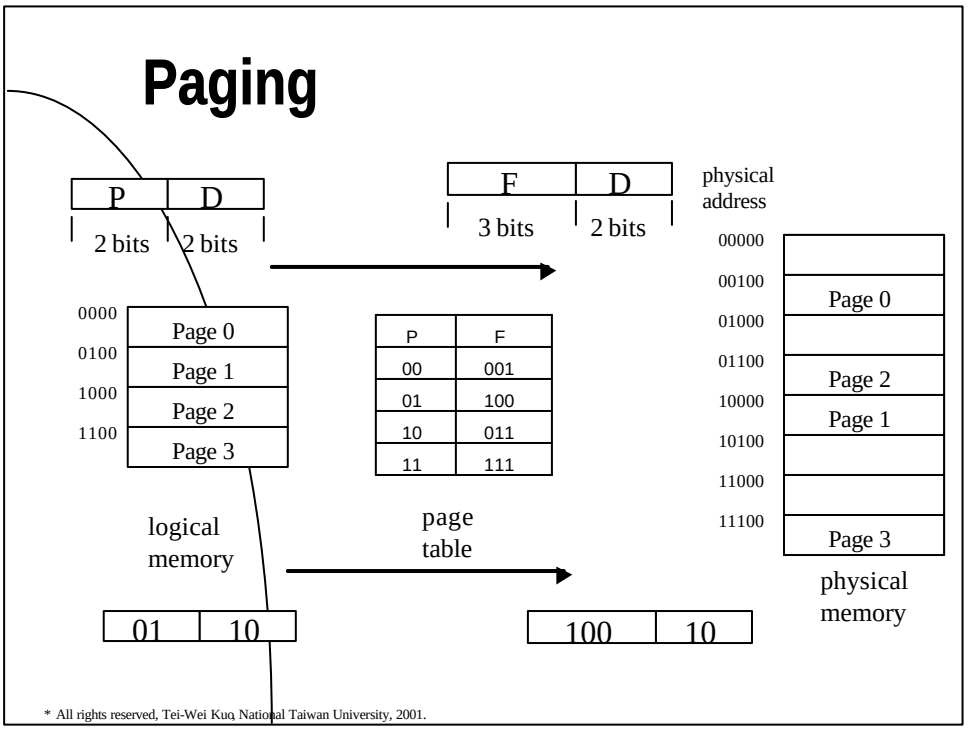
- Paging – Another solution to external fragmentation!

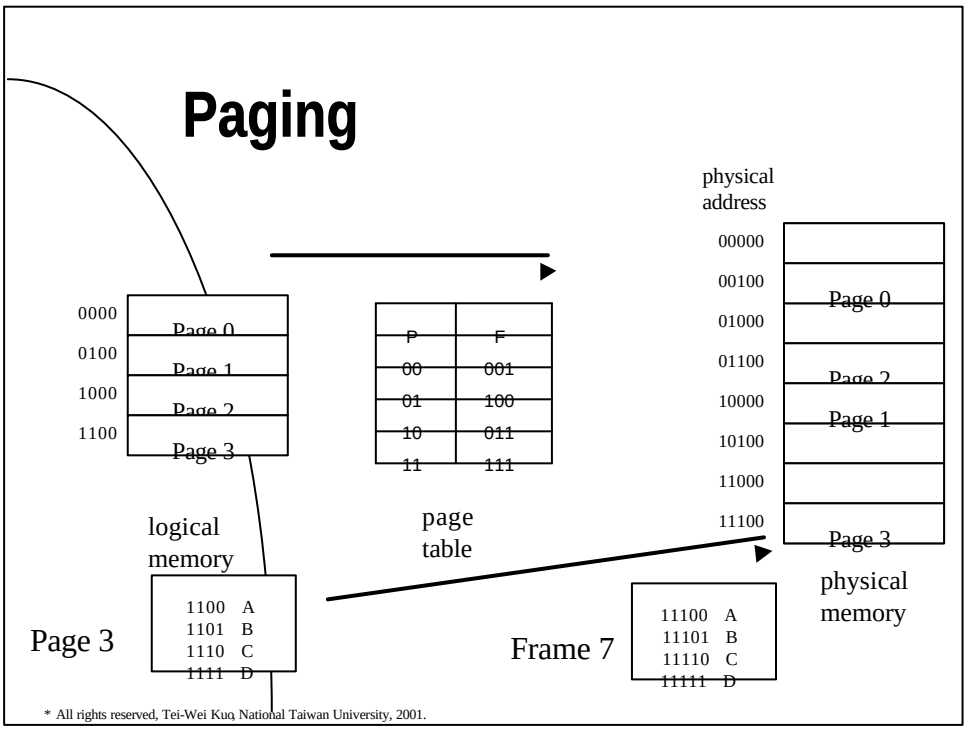


Paging

- Address Translation

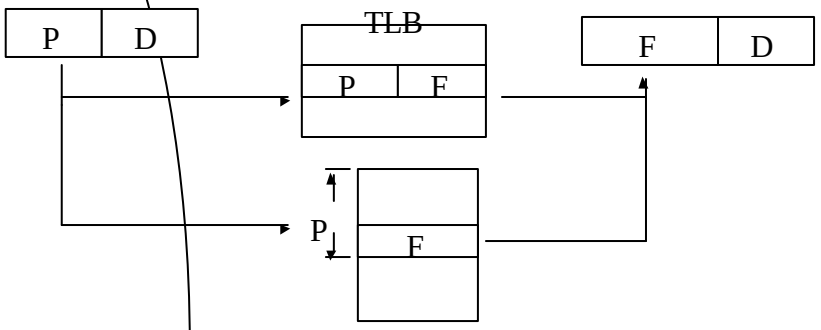



Paging
P
D
2 bits
2 bits
F
D
3 bits
2 bits
physical address
0000
0100
1000
1100
Page 0
Page 1
Page 2
Page 3
P
F
00
001
01
100
10
011
11
111
00000
00100
01000
01100
10000
10100
11000
11100
Page 0
Page 2
Page 1
Page 3
logical memory
page table
physical memory
01
10
100
10
* All rights reserved, Tei-Wei Kuo, National Taiwan University, 2001.


Paging
physical address
0000
0100
1000
1100
Page 0
Page 1
Page 2
Page 3
P
F
00
001
01
100
10
011
11
111
00000
00100
01000
01100
10000
10100
11000
11100
Page 0
Page 2
Page 1
Page 3
logical memory
page table
physical memory
Page 3
1100 A
1101 B
1110 C
1111 D
Frame 7
11100 A
11101 B
11110 C
11111 D
* All rights reserved, Tei-Wei Kuo, National Taiwan University, 2001.

Paging

- Hardware Support for Paging
  - Registers as Page Tables
  - Memory-Resident Page Tables
    - Translation Look-aside Buffer (TLB)



Paging

- Paging-TLB
  - TLB Hit
    - Access time = TLB-access-time + Inst/Data-Memory-Access
      - E.g., 20ns + 100ns
  - TLB Miss
    - Access Time = TLB-access-time + Page-Table-Memory-Access + Inst/Data-Memory-Access
      - E.g., 20ns + 100ns + 100ns
- Hit Ratio 80%
  - Effective access time = 20ns + 100ns + 0.2 *100ns = 140ns

Virtual Memory

- Definition
  - A technique that allows the execution of processes that may not be completed in memory.
- Swapping
  - Process image may reside in the backing store rather than swap the entire image in.
  - Page fault: occurs when program references a non-memory-resident page.
- Thrashing
  - A process is spending more time in page faults than executing.

* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 289,291-293,317.

Context Switching-Revisiting

- Def.
  - Switch the CPU from one process to another process
    - Save the state (or called context) of the running process
    - Reload the state of the ready process
- Context Switching Time
  - Hardware-dependent!
    - Multiple register sets!
    - Special hardware instructions!

Process Concept-Revisiting

- Process Control Block

```
Struct PCB {
    char p_pid;
    char p_pri;
    char p_ppid;
    int pc; /* program counter */
  …
    int files[NFILE];
} PCB[NPROC];
```


Operating Systems Concept

- What is an operating system?
- Operating system architecture
- Process concept
- CPU scheduling
- Memory management
- File and I/O systems

File System

- Provides a mechanism for on-line storage of and access to both data and programs
- Components
  - Files: logical unit abstracted by the OS.
  - A directory structure: special files to organize and provide information to files.
- Implementation Issues
  - File storage
    - Disk space allocation, free-space recovery, etc.
  - File access
    - Direct/sequential access, protection, etc.

* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 337-342,346-348.

File System

- What should be considered?
  - File attributes such as name, file operations such as seek, file types such as executable, file structures such as those for Word.
- File System Structure
  - File systems usually reside on block-oriented devices – block transfer
  - Characteristics
    - In-place update
    - Any given block can be accessed directly.

File System

- File Attributes
  - Name, type, (disk) location, size, protection, modified/created time/date, user/group ID, etc.
- File Operations
  - Create, write, read, reposition, delete, truncate.
- File Structure
  - File: free form – a sequence of bytes in UNIX
  - Directory: a structured file in UNIX

File System – File Type

- Why file type?
  - OS can recognize a file and operate on it in a proper way.
- Common Tecniques
  - Include the type as a part of the name:
    - Executable . Exe, .com, .bin
    - Object .obj, .o
  - Read information such as creators in file attributes
    - UNIX – store a crude magic number at the file beginning

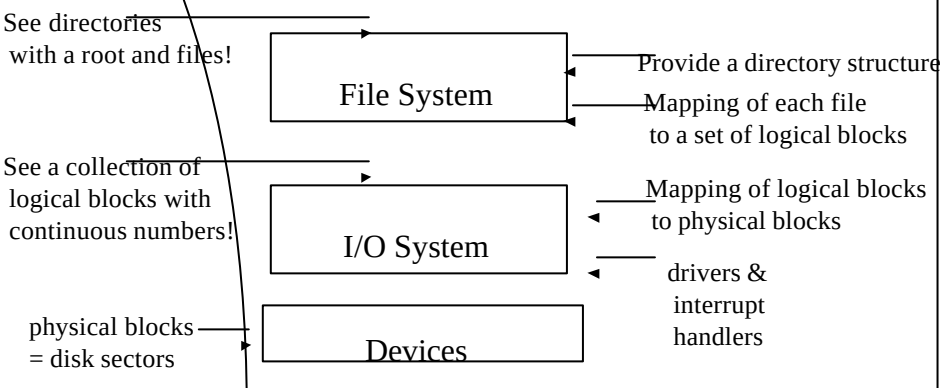
* In many cases, extensions are only considered as hints.

File System – File Structure

- Approaches
  - A minimum number of file structures
    - UNIX – flexible but with limited support
    - The file structure for executables must be supported
  - A number of file structures and special operations
    - No support of file structures requireed by new applications
    - Large OS size!

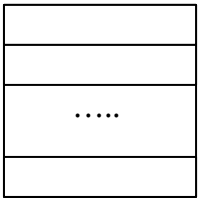
File System

- File System vs I/O System vs Device



File Systems

File: a sequence of bytes



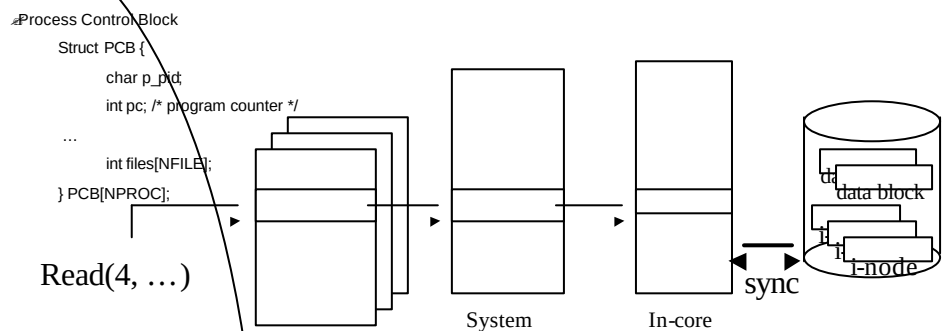
a sequence of (logical) blocks

- File Methods
  - Sequential Access
    - Basic Operations
      - READ, WRITE + file pointers
  - Direct Access
    - Basic Operations
      - READ N or Write N, where N is the relative block number.

File System – A UNIX Approach

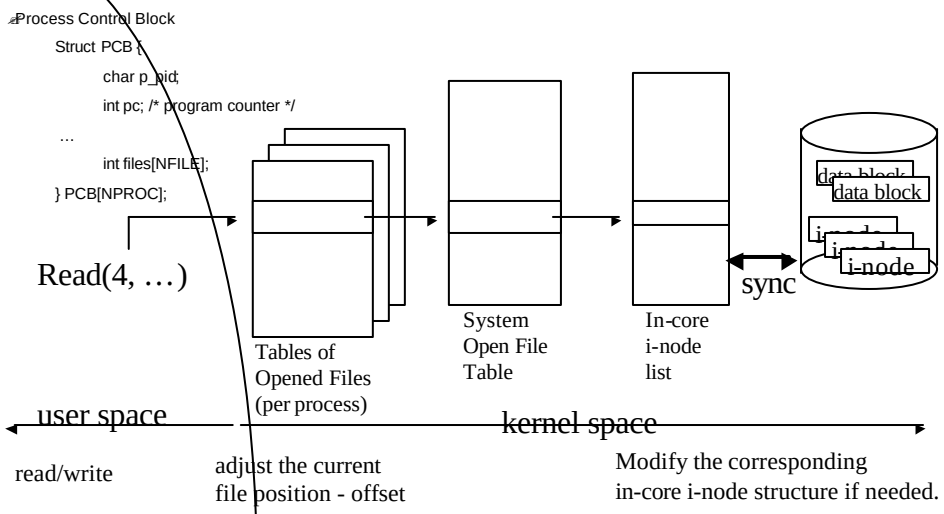
- Process Control Block

```
Struct PCB {
        char p_pid;
        int pc; /* program counter */
        …
        int files[NFILE];
} PCB[NPROC];
```



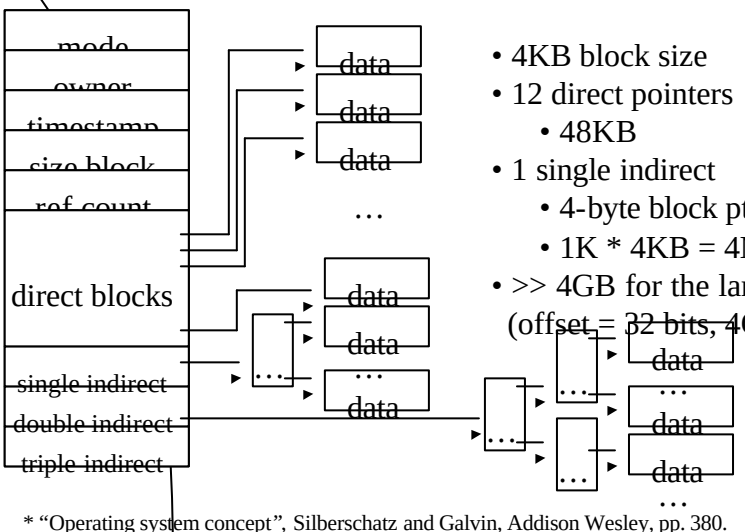
* The i-node structure of a file includes info regarding the disk location of the file.

File System – A UNIX Approach



* When the file is closed, and file-open-count = 0, then the disk-resident i-node should be modified!

BSD UNIX i-node

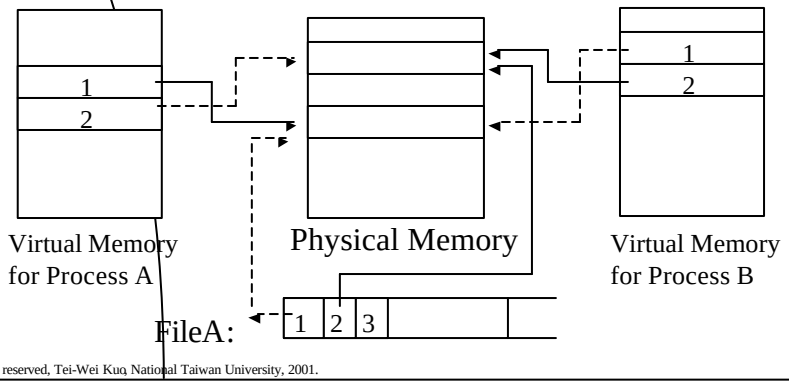


- 4KB block size
- 12 direct pointers
  - 48KB
- 1 single indirect
  - 4-byte block ptr
  - 1K * 4KB = 4MB
- >> 4GB for the largest file! (offset = 32 bits, $4G=2^{32}$)

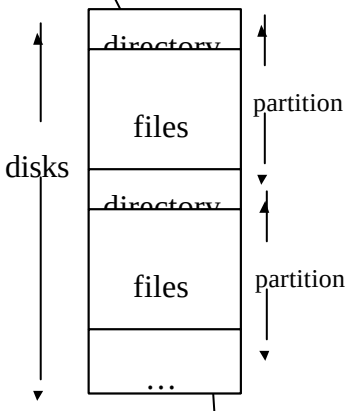
* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 380.

Memory-Mapped File

- Allow a part of the virtual address space to be logically associated with a section of a file.



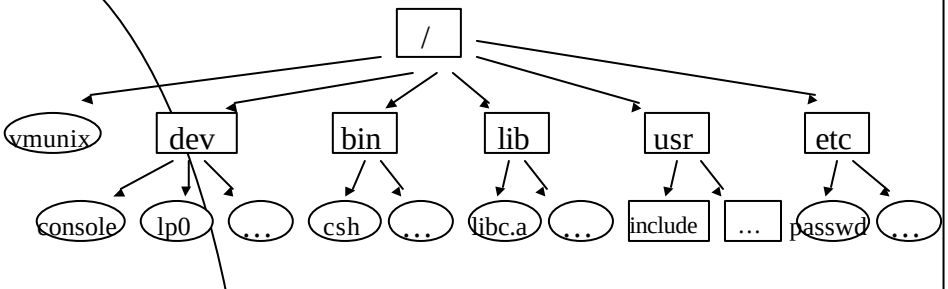
File System – Directory Structure



- Partition (/Volume):
  - a low level structure in which files and directories reside.
- Directory:
  - Records info for “all” files on a partition.

* “Operating system concept”, Silberschatz and Galvin, Addison Wesley, pp. 349,354-358.

File System – Tree-Based Directory

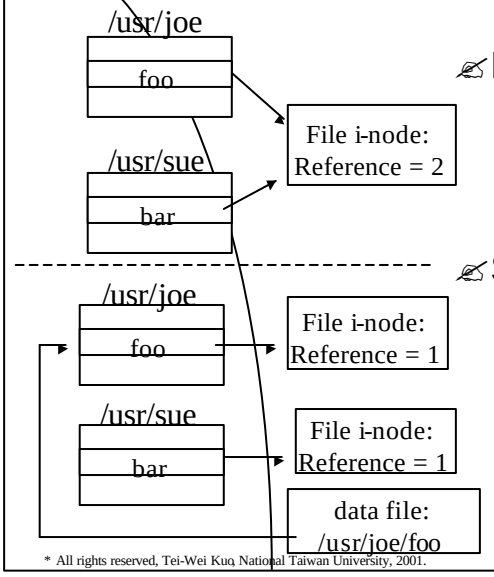


- Path name
  - Specify a file by listing "node" names from the root to the corresponding node in the structure
    - /users/userA/info-sys/test

File System – Tree-Based Directory

- Path name
  - Absolute path name
    - Begin at the root
      - /usrs/usrA/fileA
  - Relative path name
    - Begin at the current directory
      - usrA/fileA
- Path Names vs Devices
  - Mixing of device names and directories – MS DOS/Windows
    - C:/mine/ntu/mail
  - No distinguished features – A UNIX approach

Sharing of Files

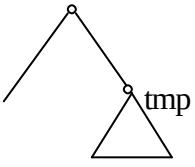


- Hard Link
  - Each directory entry creates a hard link of a filename to the i-node that describes the file's contents.
- Symbolic Link (Soft Link)
  - It is implemented as a file that contains a pathname.
  - Example: Shortcut on Windows

* Problem – infinite loop in tracing a path name with symbolic links – 4.3BSD, no 8 passings of soft links
* Dangling pointers

File Systems - Mounting

- (name of the device, mount point)



Use an appropriate device driver to read the device directory and verify the format => mount!

- Mount point: the location within the file structure at which to attach the file system.
- A bit in the i-node indicates that whether a file system is mounted on it! -> find the i-node of the root of the mounted file system.

UNIX: manual mounting

Protection

- Major concern for information storage
  - Reliability – prevent physical damage
    - Information Duplication!
  - Protection – prevent improper access
    - Access Control!

Protection

- How to prevent improper access?
  - Access control!
    - Read, Write, Execute, Append, Delete, List, etc
- Approaches:
  - Complete isolation
  - No protection at all
  - Controlled access by limiting the “types” of file access based on some factors:

* “Operating system concept”, Silberschatz and Galvin, Addison Wesley, pp. 360-362.

Protection

- Access user list for each file/directory
  - { < user-ID, allowed access types> … }
  - Tedious in maintenance and variable directory sizes -> condense such info (UNIX)
    - Read/write/execute over owner/grp/others
    - Issues
      - Control in group memberships
      - Membership per user?
- A password for each file/directory
  - A large number of passwords
  - Different passwords for different levels of protection?

Operating Systems Concept

- What is an operating system?
- Operating system architecture
- Process concept
- CPU scheduling
- Memory management
- File and I/O systems

I/O Subsystems

- Why?
  - Provide the simplest interface to the rest of the system
  - Optimize I/O for the maximum concurrency
- Variations:
  - Block vs Character I/O
  - Sequential/Random Access
  - Synchronous/Asynchronous Transfer
  - Dedicated/Share
  - Read-Only/Read-Write

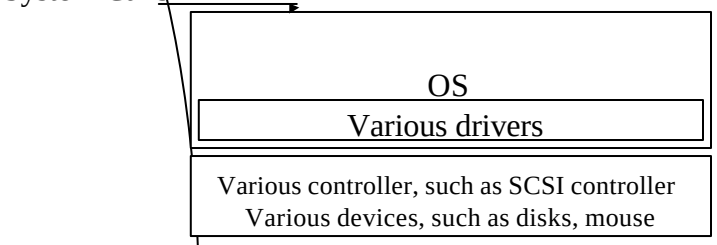
* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 398, 408-410, 414-415.

I/O Subsystems

- Conflicting trends
  - Increasing standardization of software/ hardware interfaces
  - Increasing broad variety of I/O devices
    - Device drivers which provide a uniform device-access interface to the I/O subsystem.
- How the I/O system improves the efficiency of the computer?
  - Schedule I/O operations, e.g., those on disks.
  - Use techniques such as buffering, caching, or spooling.

I/O Subsystems

- Approaches
  - Abstraction – interface
  - Encapsulation – device drivers



Principles in Doing I/O

- Reduce the number of context switches.
- Reduce the number of data copyings.
- Reduce the frequency of interrupts
  - Large transfer, smart controller, etc.
- Increase concurrency
  - DMA controller
- Move processing primitives into hardware.
- Balance CPU, memory subsystems, bus, and I/O memory.

* "Operating system concept", Silberschatz and Galvin, Addison Wesley, pp. 424-425.