CSE 306 Operating Systems
Operating System Overview 2

YoungMin Kwon

Major Achievements

- Four major theoretical advances in the development of operating systems
  - Processes
  - Memory management
  - Information protection and security
  - Scheduling and resource management

Process

- Many definitions
  - A program in execution
  - An instance of a program running on a computer
  - The entity that can be assigned to and executed on a processor
  - A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources.

Process: Multiprogramming

- Three major developments related to the concept of process

- 1. Multiprogramming
  - Keep processors and I/O devices simultaneously busy to improve resource utilization
  - In response to signals for I/O completion, the processor is switched to a program in main memory

Process: Time Sharing

- 2. Time sharing
  - Be responsive to the needs of individual user
  - Able to support many users simultaneously (for cost reasons)
  - Example
    - In general, users' reactions are relatively slow
    - If each user needs 2 sec of processing time per minute, about 30 users can share the system without noticing the interference
    - OS overhead must be factored

Process: Real-time Transaction Processing

- 3. Real-time transaction processing system
  - Exclusive access to resources with commit or abort operations
  - Many users are entering queries or updates against a database

Process: Interrupt

- Interrupt:
  - A key tool for multiprogramming and timesharing systems
  - On an interrupt (periodic timer, I/O completion) the activity of a process can be suspended
    - The processor save a context (PC, registers, …) and branch to the interrupt handler (in the kernel mode)
    - After processing the interrupt, resume the interrupted process or other processes

Process: Program Coordination

- Errors in program coordination
  - Context switch can occur at any time → difficult to analyze concurrent execution of multiple processes
  - Coordination errors are subtle
    - May occur relatively rarely (can be once in million executions)
    - Hard to reproduce → difficult to determine the cause

Process: Program Coordination

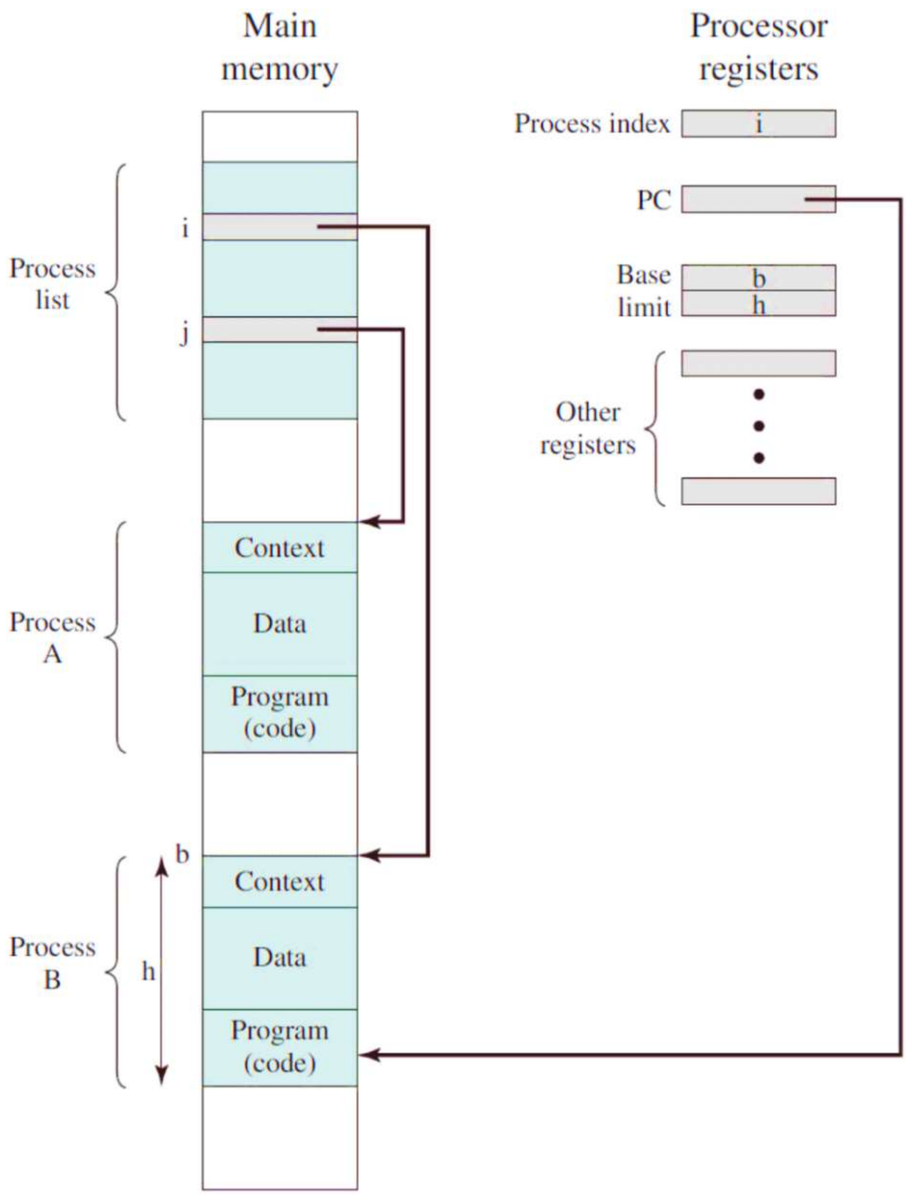
- Four main causes of errors
  - Improper synchronization
    - A process waits for an I/O to complete
    - Signals from an I/O completion can be lost or duplicated
  - Failed mutual exclusion
    - Shared resources are accessed by more than one processes
  - Nondeterministic execution
    - Depending on the memory footprint from other programs, the execution of a program differs
  - Deadlocks
    - Two or more programs are waiting for the resources held by others indefinitely

Process

- To tackle such errors $\rightarrow$ Process
  - Systematic way to monitor and control various programs running on a processor

- A process is composed of
  - An executable program
  - Associated data needed by the program
    - Variables, workspace, buffers, …
  - The execution context (process state) of the program
    - Registers, process priority, I/O waiting state, …

Process



- OS maintains a list of processes
  - Locations of the blocks of memory
  - Locations of the contexts
- A process has
  - Program, Data, Context
- A processor has
  - Base and limit registers: where the data/code begins and their size

Memory Management

- OS's main storage management responsibilities
  - Process isolation:
    - Prevent processes from interfering with other's memory
  - Automatic allocation and management:
    - Memory hierarchy should be used dynamically
    - Hide the allocation details from the programmer
  - Support of modular programming:
    - Programmers can define modules that can be dynamically created, destroyed, …

Memory Management

- OS's main storage management responsibilities (cont'd)
  - Protection and access control:
    - Sharing of memory enables one process to access the address space of another
    - OS must allow portions of memory to be accessible in various ways by various users
  - Long-term storage:
    - Storing information for extended period of time after the computer has been powered down

Memory Management

- File system
  - A long term store
  - Information is stored in named objects called files
  - Convenient concept for programmers
  - Useful unit of access control and protection for the OS

Memory Management

- Virtual memory
  - Allows programs to address memory from a logical point of view without considering the physical amount of main memory
  - Allows multiple user-processes concurrently reside in main memory

Memory Management

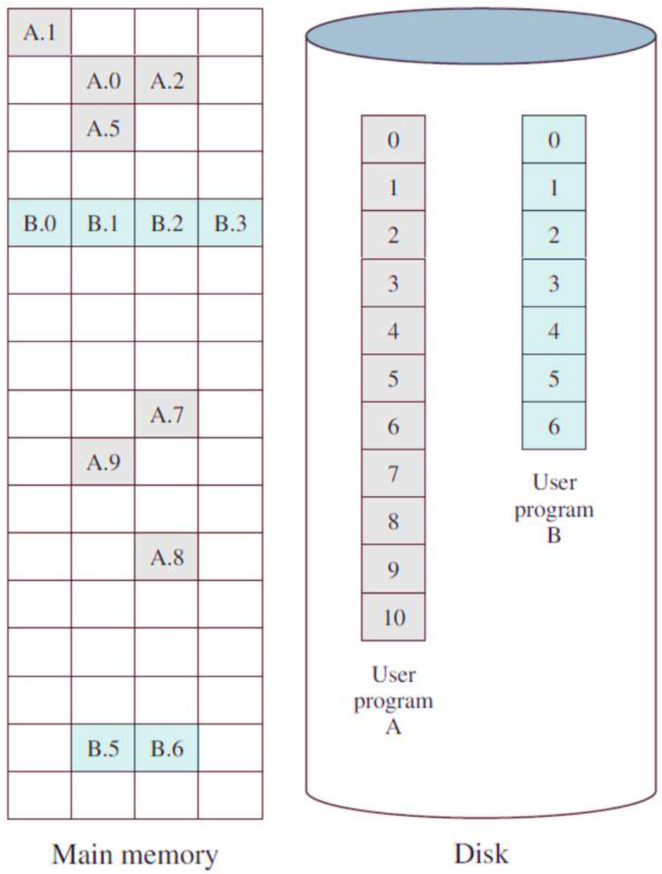
- Virtual memory
  - Paging
    - Allows processes to be comprised of a number of fixed-size blocks (called page)
    - Reduces fragmentation in virtual memory
    - Virtual address (page number & page offset)
  - Process isolation
    - Give each process a unique, non-overlapping memory
  - Memory sharing
    - Overlap portions of two virtual memory space

Memory Management

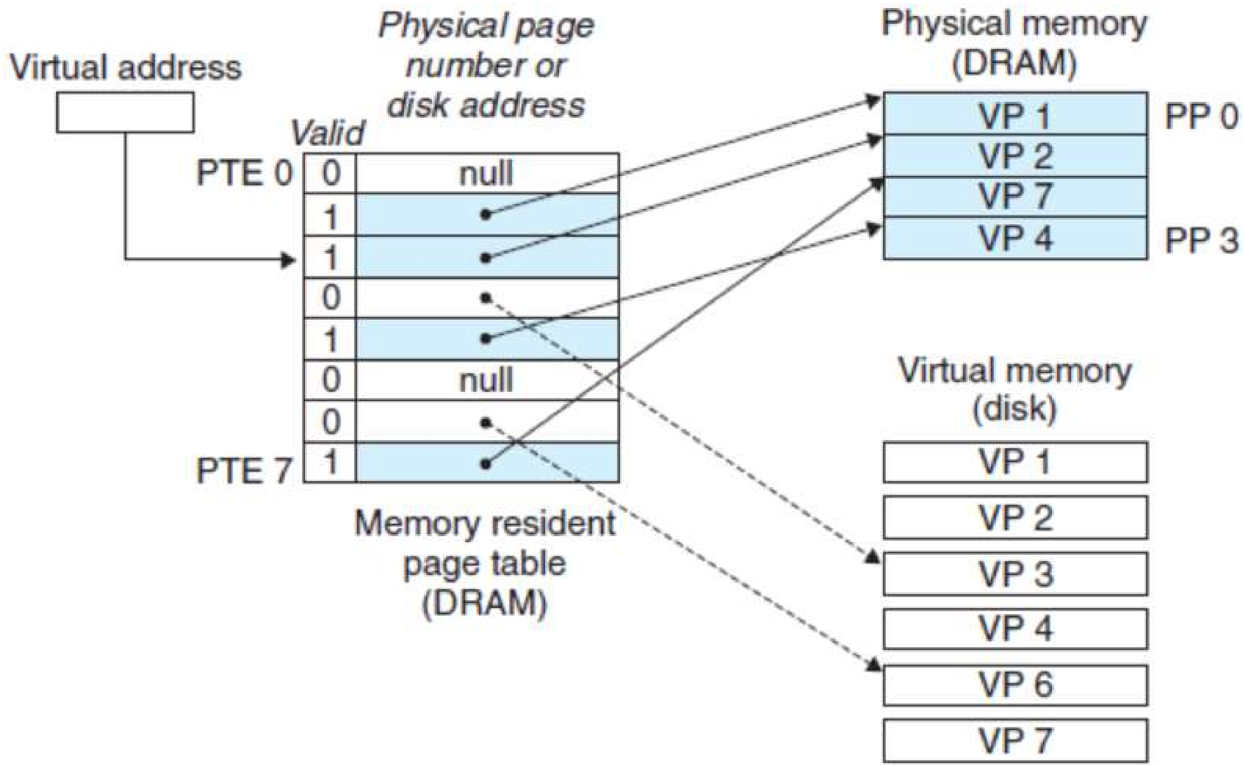
- Virtual memory
  - Page
    - Processes are divided into a number of fixed-size blocks (called page)
    - Reduces fragmentation in virtual memory
  - Page Table
    - Dynamic mapping from virtual address to physical address
    - Virtual address (page number & page offset) → Physical address (frame number & offset)

- Paging

  - All pages of a process are maintained on a disk
  - When a process is accessing memory, it's containing page might be in main memory
  - If not, MMU signals OS so that the page is loaded from a disk to main memory

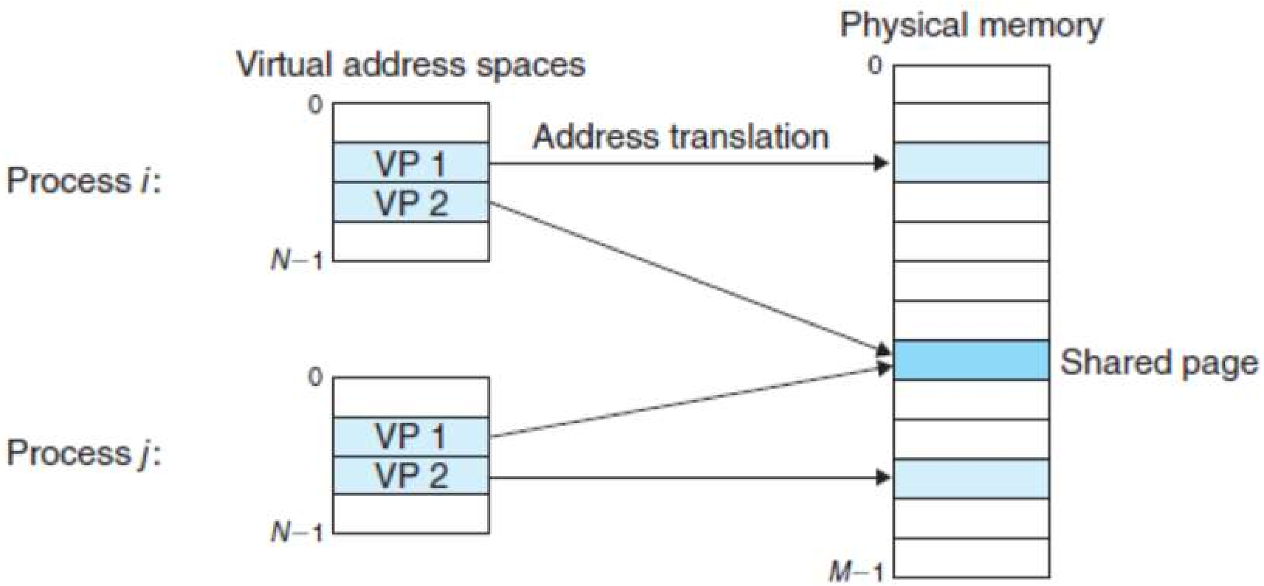


Virtual Memory: Paging



- If valid bit is set, MMU uses the physical address in PTE (page table element) to construct the physical address of the word
- If not, OS loads the page from disk

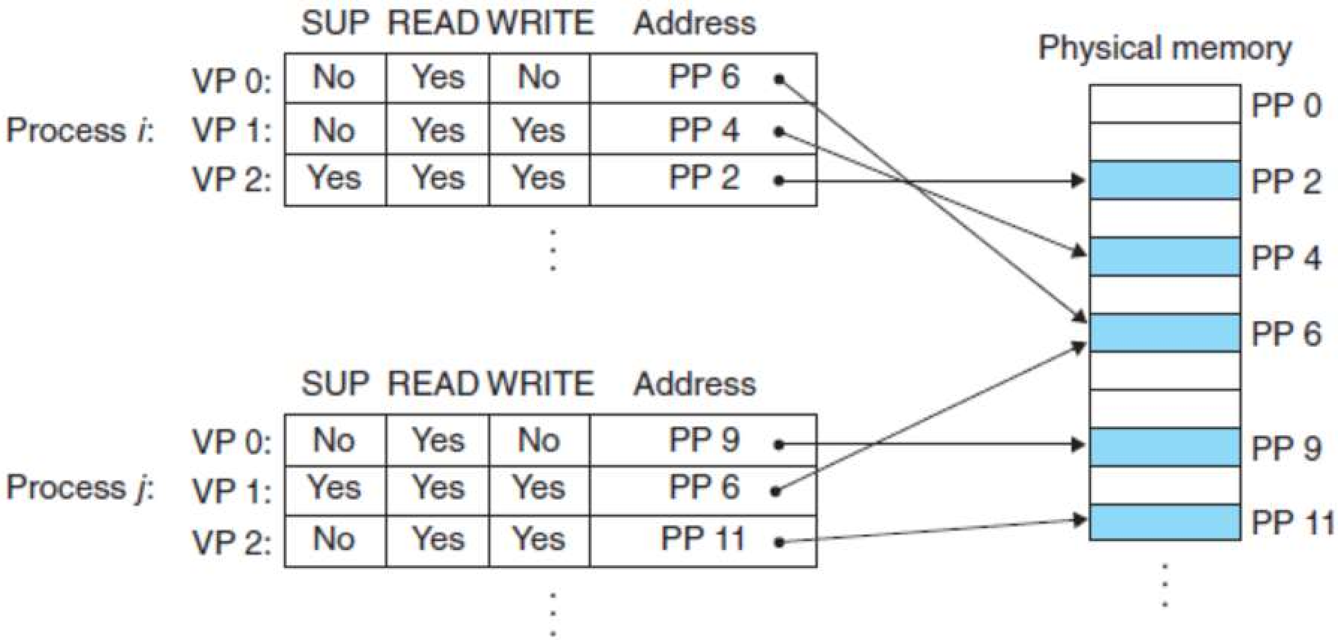
Virtual Memory: Paging



- Process isolation: giving each process a unique non-overlapping virtual memory

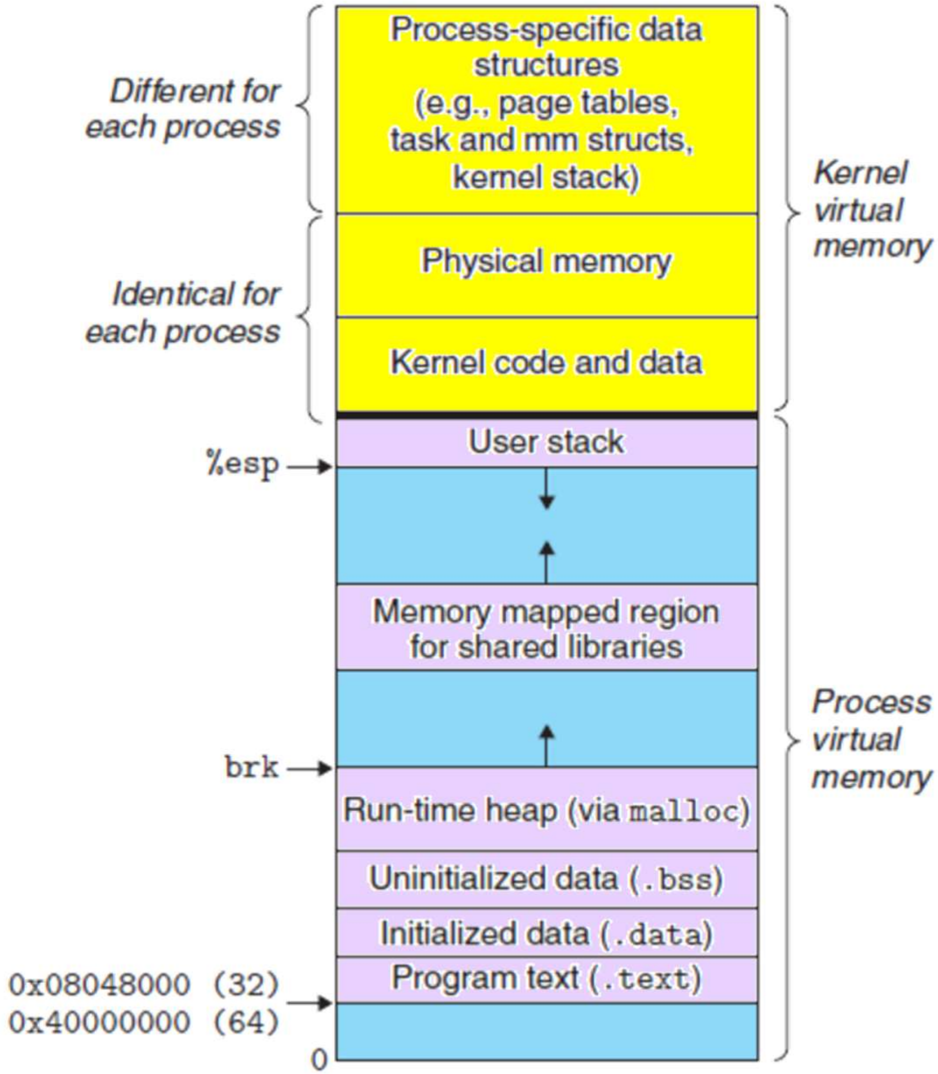
- Memory sharing: overlapping portions of two virtual memory spaces

Virtual Memory: Paging

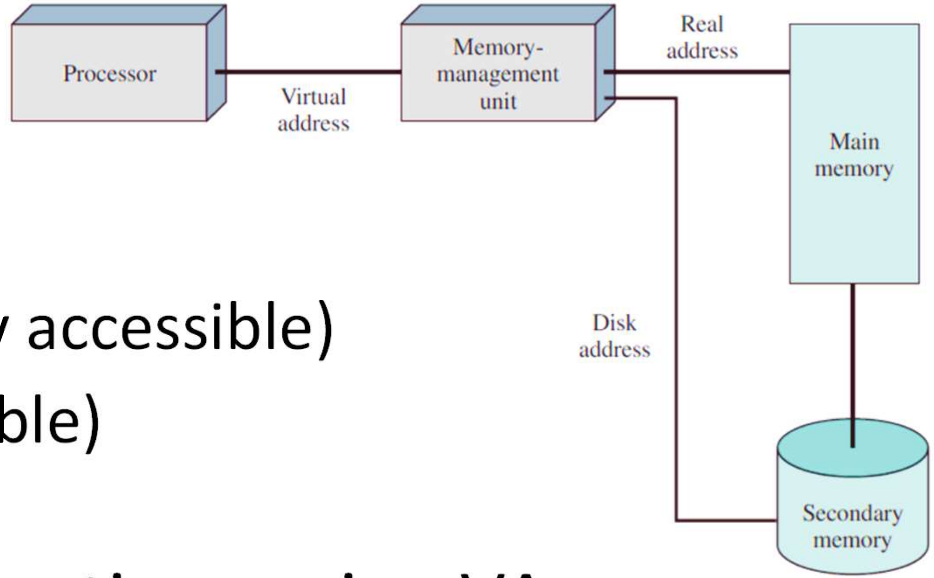


- Memory protection
  - Control the access to the contents of a virtual page by additional permission bits
  - SUP: can be accessed in kernel mode
  - READ, WRITE: read/write control

Virtual Memory System: Linux



Virtual Memory: Addressing



- VM scheme
  - Storage consists of
    - Main memory (directly accessible)
    - Disk (indirectly accessible)
  - Processes reference locations using VA
    - MMU translates VA to PA using a page table
    - If its corresponding PA is in main memory, it is read
    - If not, a trap event is generated and the page is loaded from disk, possibly after swapping out a page
    - The process that generated the address is suspended

Information Protection and Security

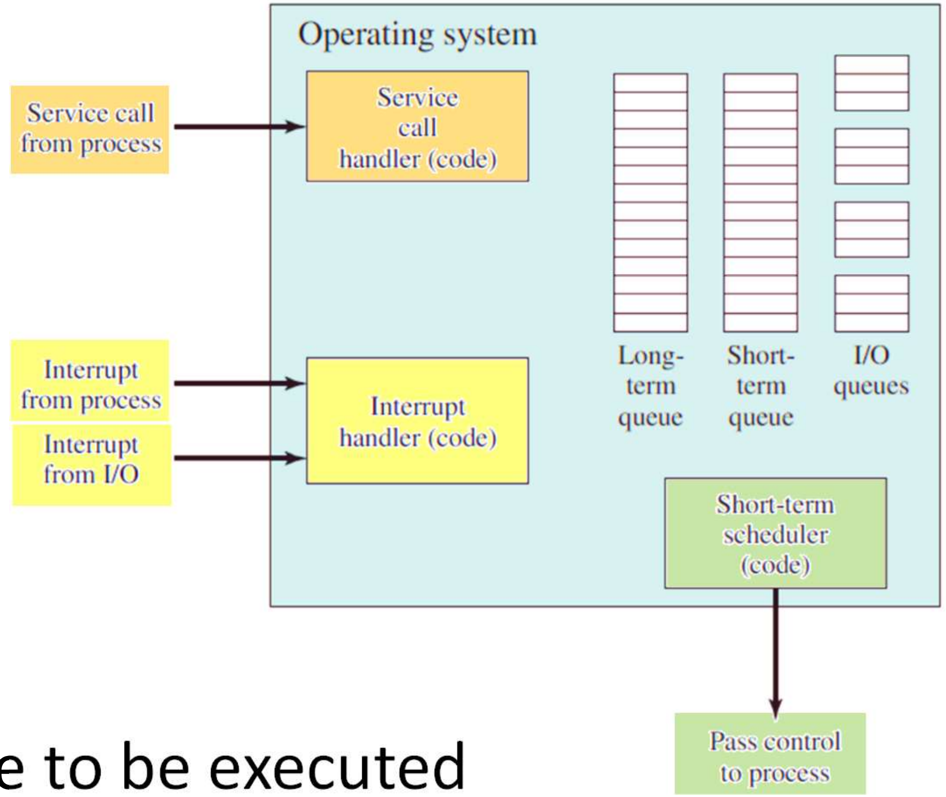
- Controlling the access to computer systems and the information stored in them

- Four categories:
  - Availability: protecting the system against interruption
  - Confidentiality: prevent unauthorized reading
  - Data integrity: prevent unauthorized writing
  - Authenticity: verification of the identity of the user and the validity of message or data

Scheduling and Resource Management

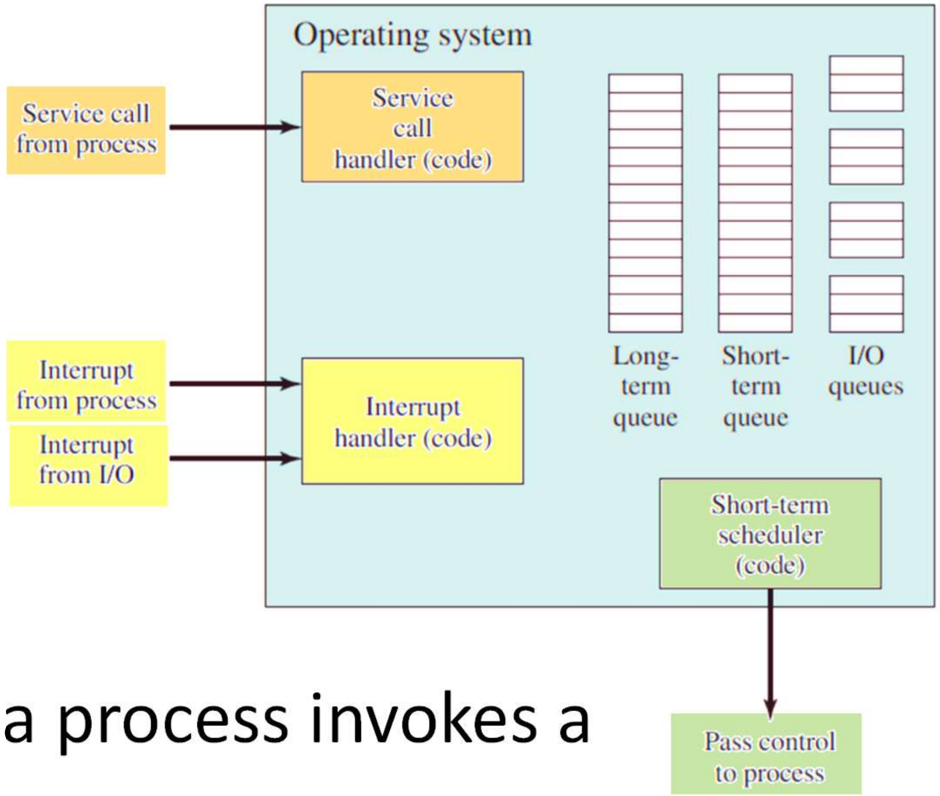
- Three factors to consider in resource allocation
  - Fairness
    - Give equal and fair access to resources to all processes in the same class
  - Differential responsiveness
    - Discriminate among different classes of jobs
  - Efficiency
    - Maximize throughput
    - Minimize response time
    - Accommodate as many users as possible

Process Scheduling



- Short-term queue
  - Processes in main memory and ready to run
  - Round-robin or priority order
- Long-term queue
  - New processes waiting
  - Moved to the short-term queue to be executed
- I/O queue
  - All processes waiting for use each device is lined up in the device's queue

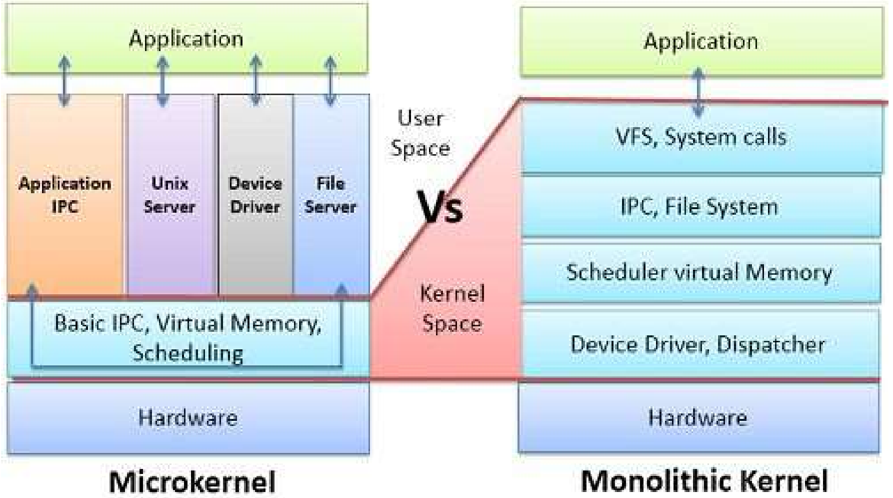
Process Scheduling



- OS can receive control of the processor
  - At the interrupt handler if an interrupt occurs
  - At the service call handler if a process invokes a service call (system call)
  - After handling the interrupt or the service call, a short-term scheduler is invoked to pick the next process to run

Developments Leading to Modern Operating Systems

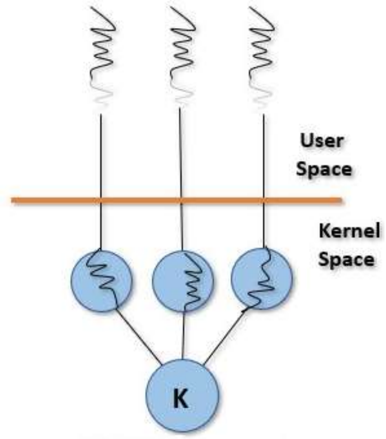
- Microkernel architecture
  - Monolithic kernel:
    - A large kernel provides most of the OS functionality
    - A single process with all elements sharing the same address space



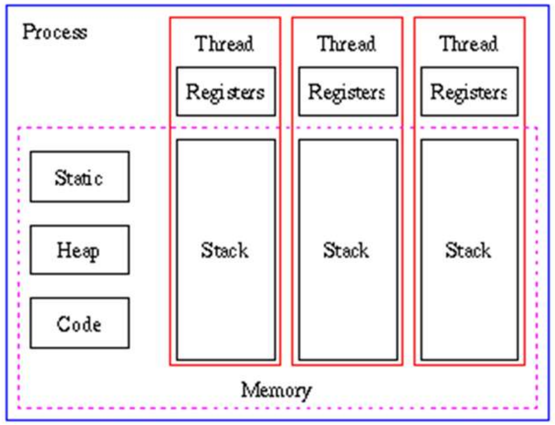
  - Microkernel architecture: kernel has only few essential functionalities
    - Address space management, inter-process communication, basic scheduling
    - Other OS services are provided by user mode processes.

Developments Leading to Modern Operating Systems

- Multithreading
  - Thread: a logical flow that runs in the context of a process



  - Process: a collection of one or more threads and associated system resources



Developments Leading to Modern Operating Systems

- Symmetric Multiprocessing (SMP)
  - OS schedules processes and threads across all processors
  - Benefits
    - Performance: more than one processes can run simultaneously
    - Availability: a failure of a single processor does not halt the system
    - Incremental growth: a user can enhance the performance of a system by adding additional processors
    - Scaling: vendors can offer a range of products with difference prices and performances

Developments Leading to Modern Operating Systems

- Distributed operating systems
  - Provides an illusion that a cluster of machines is running as a single computer
- Object-oriented design
  - Adding modular extension to a small kernel
  - At the OS level, an object-based structure enables programmers to customize OS without disrupting system integrity

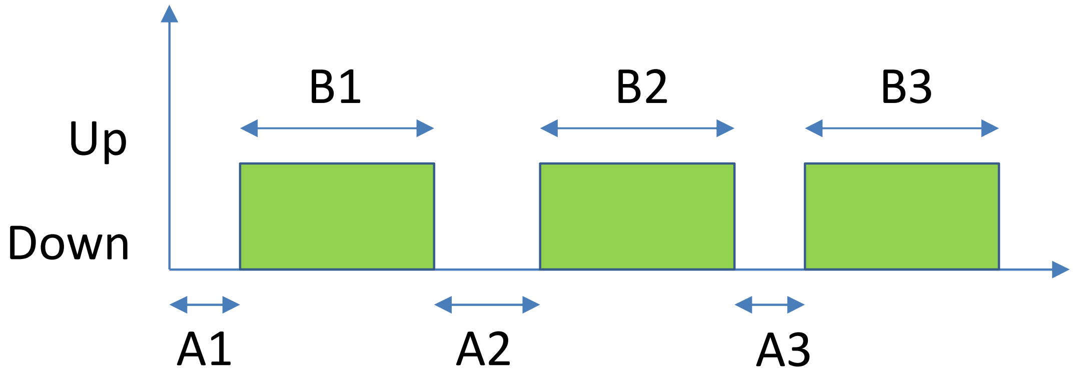
Fault Tolerance

- Fault tolerance
  - The ability of a system to continue normal operation despite the presence of hardware or software error
- Reliability
  - $R(t)$: the probability that a system operates correctly up to time t
- Mean time to failure (MTTF)
  - $$\mathrm{MTTF} = \int_0^\infty R(t)dt$$
- Mean time to repair (MTTR)
  - The average time it takes to repair a faulty element

Fault Tolerance

- Availability
  - Fraction of time the system is available to serve users' requests

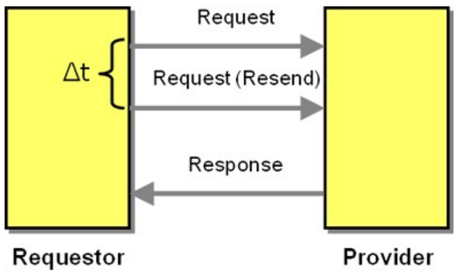
$$\mathrm{A} = \frac{\mathrm{MTTF}}{\mathrm{MTTF} + \mathrm{MTTR}}$$



$$\mathrm{MTTF} = \frac{B1 + B2 + B3}{3} \qquad \mathrm{MTTR} = \frac{A1 + A2 + A3}{3}$$

Fault Tolerance

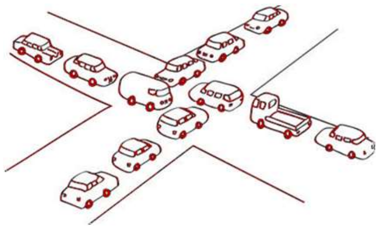
- Solutions: adding redundancy
  - Spatial (physical) redundancy: use multiple components performing the same function or backup
    - Backup name server on the Internet
  - Temporal redundancy: repeating a function or operation when an error is detected
    - Data retransmission
  - Information redundancy: replicating or coding data such that an error can be detected and corrected
    - RAID disks



Fault Tolerance

- Operating System Mechanisms
  - Process isolation: main memory, file access, flow of execution

  - Virtual machines: application isolation and redundancy
  - Concurrency controls: recover from fault conditions like deadlock



  - Checkpoints and rollback:
    - Checkpoint: a copy of application's state
    - Rollback: restart the execution from a checkpoint