澳 門 理 工 學 院
Instituto Politécnico de Macau
Macao Polytechnic Institute

School of Applied Sciences (B.Sc. in Computing)

Notes #3: Process Description and Control

COMP 213

(21121/21221)

Operating Systems

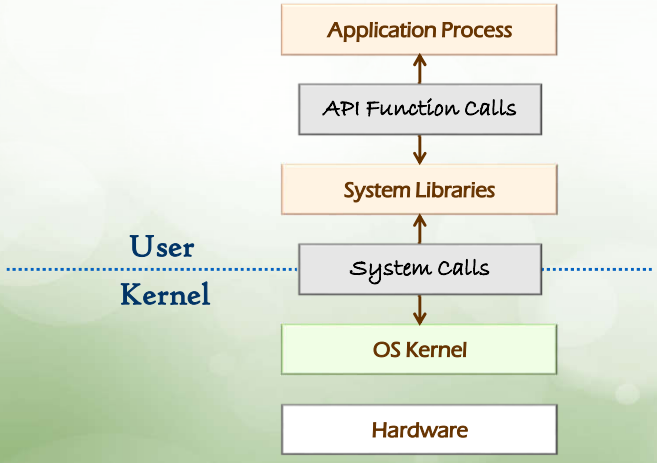
2019-2020 1st Semester

Review

- Computer: a collection of different hardware components
- Applications
  - Developed to perform tasks
  - Inflexible if written directly on hardware
- Operating system
  - Offers a set of features, secure, **consistent interface** for developing applications
  - **Managing hardware resources** with concept of virtual machines
- What is the concept of virtual machine?
  - The goal of OS is provide a uniform, **abstract representation** of resources to applications
  - More elaboration later

Don't mix up the virtual machine concept here with the "monitor" in early batch system design, and the VMware or the virtual box VM

System Calls: A Typical View

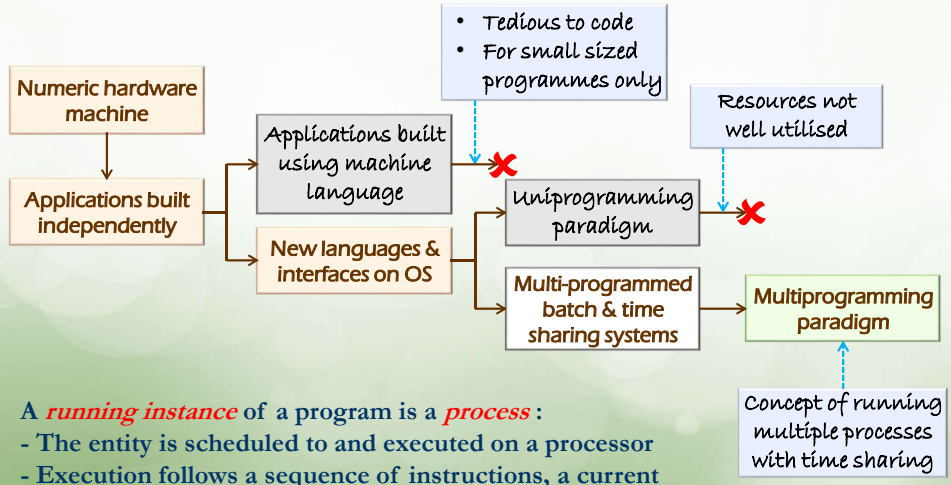


API: application programming interface

Fact Checks on Modes

- **User Mode**
  - Application processes run in user mode
  - Processes have no direct access to hardware or reference memory; must delegate to system calls to access hardware or memory
  - With protection and isolation, crashes in user mode are usually recoverable
- **Kernel Mode**
  - Has complete and unrestricted access to the underlying hardware; can execute any CPU instruction and reference any memory address
  - Generally reserved for the lowest-level, most trusted functions of the operating system
  - Crashes in kernel mode are catastrophic; will halt the computer

Review the Path to Running Multiple Processes



A *running instance* of a program is a *process*:
- The entity is scheduled to and executed on a processor
- Execution follows a sequence of instructions, a current state, and an associated set of system instructions

Eddie Law

Topics to Cover

- Uniprogramming wastes resources
- Should use processor and I/O devices efficiently
- An application program and running processes
- Process description and control
- Multiprogramming
  - Resources made available to multiple applications
  - (new idea) Processor runs multiple processes simultaneously – how?
- Chapters: 3.1 - 3.5

Eddie Law

Situation: Utilisation of the CPU

- Suppose the context switching time by CPU is negligible and ignored
- Suppose a process consumes CPU 5% of its execution time
  ⇒ 95% of the CPU time is idle ⇒ wasted resources
- A trivial design plan:
  - Running 20 similar processes (disregard the switching overhead [discussed later in this set of slides], and the utilization of other needed resources)
  ⇒ Close to 100% fully utilization of all the CPU time
  ⇒ Concept of ***multiprogramming***

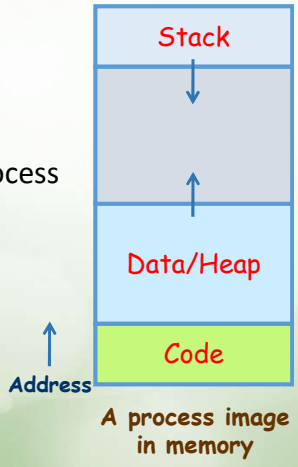
Process

- "A program in execution", also called task
- 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 the execution of a sequence of instructions, a current state, and an associated set of system instructions
- (sometimes, process, task, job are used interchangeably, should check carefully)

Note the difference between program and process.

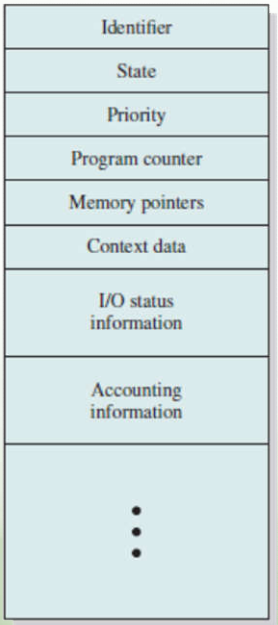
Process Elements

- A process is comprised of:
  - Program binary code (possibly shared)
  - A set of data
  - A number of attributes describing the state of the process



Process Elements

- While the process is running it has a number of elements including
  - Identifier
  - State
  - Priority
  - Program counter
  - Memory pointers
  - Context data
  - I/O status information
  - Accounting information



Requirements in Process Management

- Interleave the execution of several processes ...
  - To maximize processor utilization
  - While providing reasonable response time
- Allocate resources to processes
- Support inter-process communication, if needed

Examples on Creating Processes

- Submission of a batch job
- User logs on
- Create to provide a service such as printing
- Spawned by an existing process (parent-child processes)

On Terminating Processes (1)

- Batch job issues Halt instruction
- User logs off
- Process executes a service request to terminate
- Child processes terminate so parent may terminate

(Or vice-versa? Zombie processes?)

On Terminating Processes (2)

- Operating system intervention
  - E.g. when deadlock occurs (more elaboration on this later)
- Error and fault conditions
  - E.g. memory unavailable, protection error, arithmetic error, I/O failure, invalid instruction

Dispatcher

- (Recall, the old "monitor" in early batch system design)
- A small program that switches the processor from one process to another
- Prevents a single process from monopolizing processor time
  - Every process should be able to use the processor for a fair amount of time

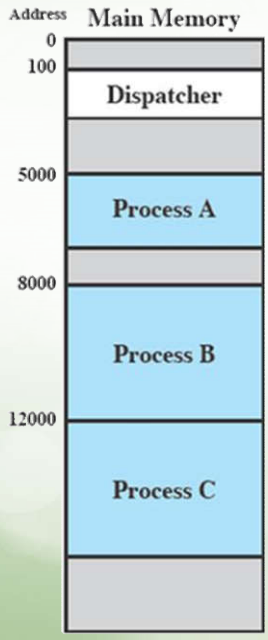
  $\Rightarrow$ Concept of time quantum (or time slots)

Trace of the Process

- The behavior of an individual process is shown by listing the sequence of instructions that are executed
- This list is called a ***Trace***

Process Execution

- Consider three processes being executed
- All are in memory (plus the dispatcher)



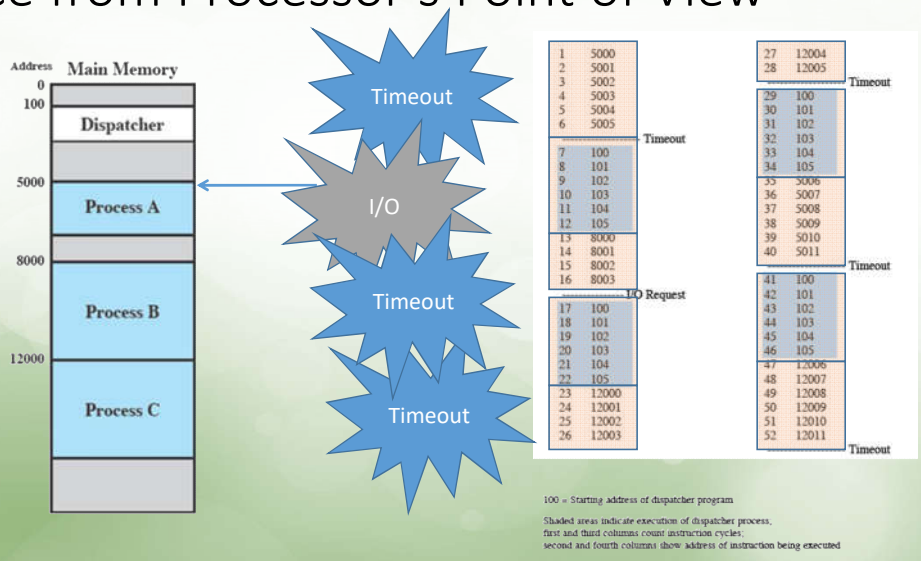
Trace from the *Processes'* Point of View:

- Virtual machine concept
- A user process may not know the existence of
  - Other processes, or
  - Even the OS
- Example: 3 processes run to completions as shown

(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

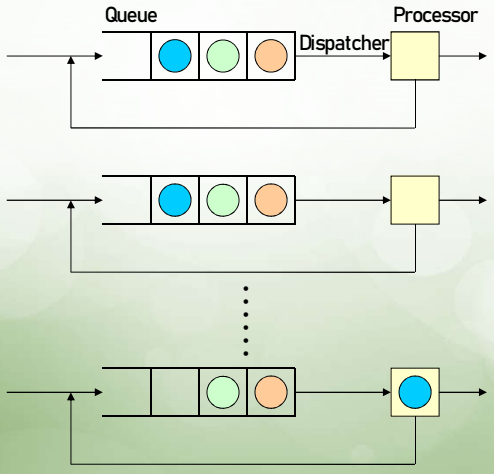
5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Trace from Processor's Point of View



Round-Robin Scheduling

Scheduling concepts from Chapter 9



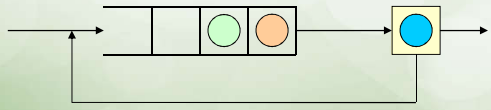
Time sharing system: each Running process is allotted to run at most one fixed time duration – "time quantum" or "time slice"

Each process is allowed to execute for at most a quantum/slice

At timeout, process switching occurs

Process is Preempted When…

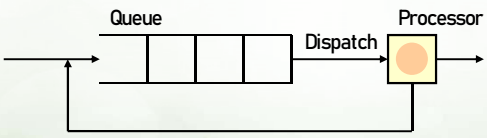
- In the following two cases, a process cannot continue running and must leave the processor, i.e., it is ***preempted***
  - Timeout
  - I/O, or wait for other events

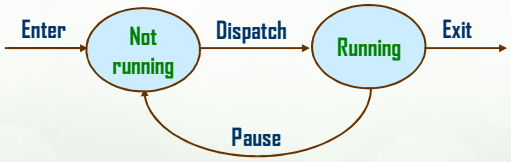


On process switching later in this chapter.

Process and State Models

- Even if a time sharing system, consider one process running



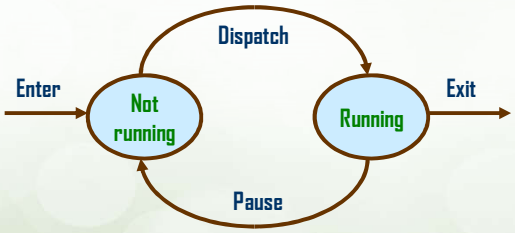


Two-state process model

- Similar to traditional uniprogramming
  - Run a program from start to finish
- Can it be more accurate?
  - It may not run, if waiting for I/O
  - Two-state process model

Simplest Two-State Process Model

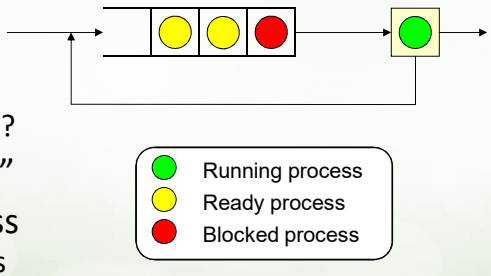
- A process is either
  - Running, or
  - Not-running
- Plus ***process creation*** and ***termination***
- How to fit this model in a multiprogramming environment?
  - Put not-running process in queue?



Eddie Law

Does "Not-Running" Mean "Ready"?

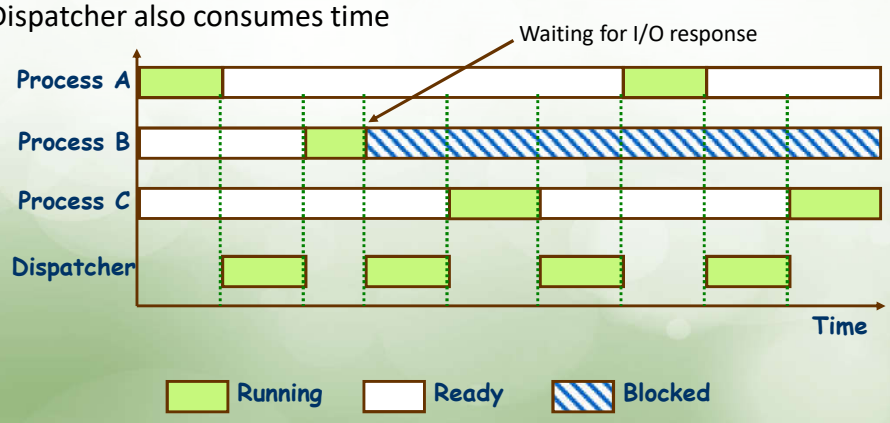
- When a running process terminates
  - CPU is idle
  - Pick an arbitary "not-running" process to run?
- In fact, "not-running" ≠ "ready to execute"
- Consider a process is waiting for I/O access
  - Which may be being used by another process
  - The process is actually "**blocked**"
- A dispatcher just cannot select the process that has been waiting in queue the longest because
  - It could possibly be "blocked"



Eddie Law

Process States: A Scenario

- More accurate picture
  - Dispatcher also consumes time

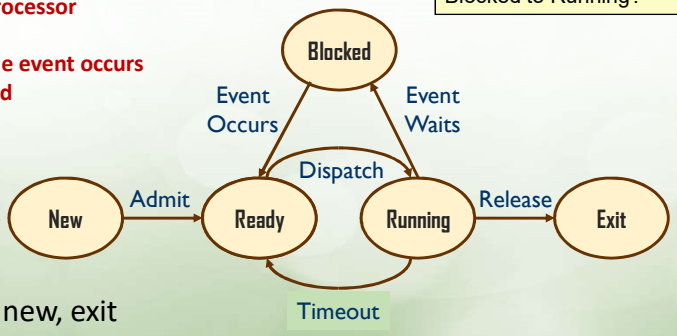


Five-State Model: An Evolvement

- Should subdivide the "Not-Running" state

Why no transitions from Ready to Blocked, and Blocked to Running?

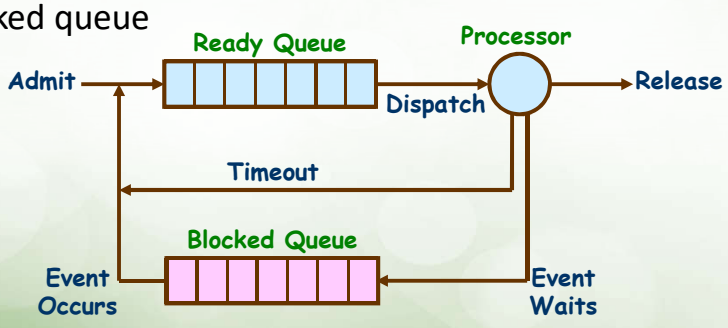
Running – being executed by the processor
Ready – ready to execute
Blocked – cannot execute until some event occurs
New – created, but not yet admitted
Exit – released



- A **five-state** model
  - Running, ready, blocked, new, exit

Using Two Queues

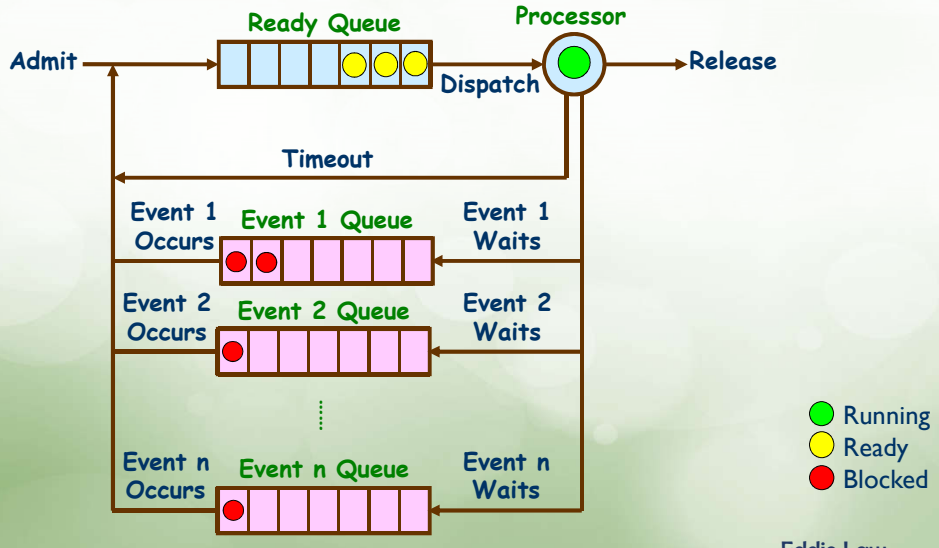
- Single blocked queue



- Different I/O devices may have different waiting times and triggering events, that is, we'll have

Eddie Law

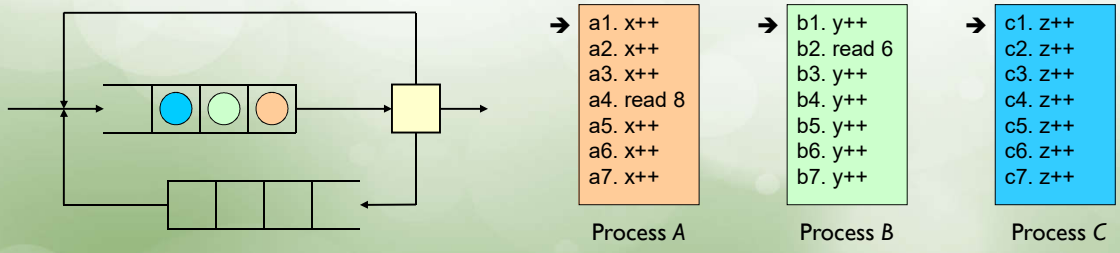
Multiple Blocked Queues



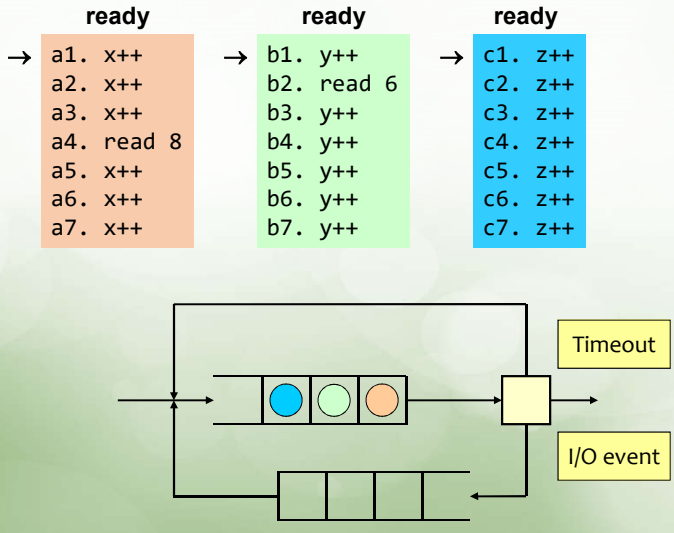
Eddie Law

How Process State Changes

- Suppose 3 processes in a 5-state model
  - Suppose a time slice of 3 instruction cycles
  - Suppose process B takes 5 cycles to read data



How Process State Changes



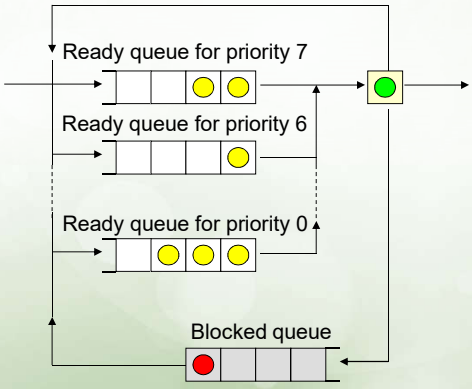
Priority Scheduling (1)

- Each process is assigned a priority
- One ready queue for the ready processes of a certain priority
- When the OS is going to dispatch a process, it chooses a ready process of the highest priority
- When a high priority process becomes ready, it may preempt a running process of lower priority

Note: more scheduling policies in ch. 9

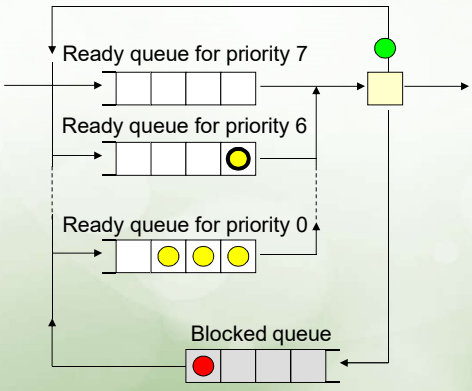
Priority Scheduling (2)

Each process is assigned a priority. There is one ready queue for the ready processes of a certain priority.



Priority Scheduling (3)

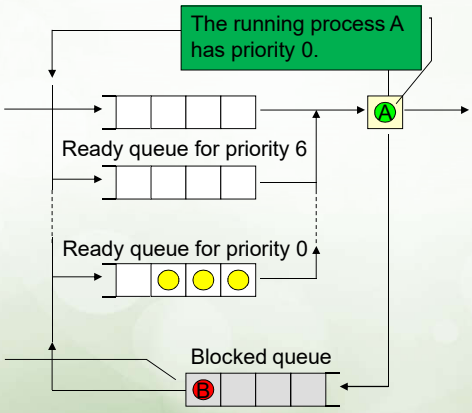
When the OS is going to dispatch a process, it chooses a ready process of the highest priority.



Priority Scheduling (4)

When a high priority process becomes ready, it may preempt a running process of lower priority.

Process B of priority 6 becomes ready. The OS may preempt the process A and dispatch process B.



Processes with Limited Resources

- Multiprogramming with multiple blocked queues
- Computing resources are limited
  - For CPU and I/O devices ← different state models
  - For memory??
- Not enough memory for all processes, what can we do?
  - To "suspend" some idle processes
  - What does "suspend" mean?

Eddie Law

What Processes are to Suspend?

- CPU is faster than I/O ⇒ many processes could be waiting for I/O
- Swap these processes to disk to free up more memory
- Blocked state upon swapping to disk is called "suspend" state
- Any implication??
  - Nature of processes: I/O-bound and CPU-bound processes
  - E.g., different priority settings

Eddie Law

Process Suspension: Examples

- Swapping
  - OS releases sufficient main memory to bring in a "ready" process to execute
- Other OS reasons
  - OS suspends 1) a background, or 2) utility process, or 3) a process that is suspected of causing a problem
- Interactive user request
  - User suspends execution of a program 1) for purposes of debugging, or 2) in connection with the use of a resource
- Timing
  - Process is executed periodically (e.g., an accounting or system monitoring process), and is suspended while waiting for the next time interval
- Parent process request
  - Parent process suspends execution of a descendent to examine or modify the suspended process, or coordinates the activity of various descendants

Eddie Law

Six-State Model: In-Transition

- Blocked processes could be suspended!

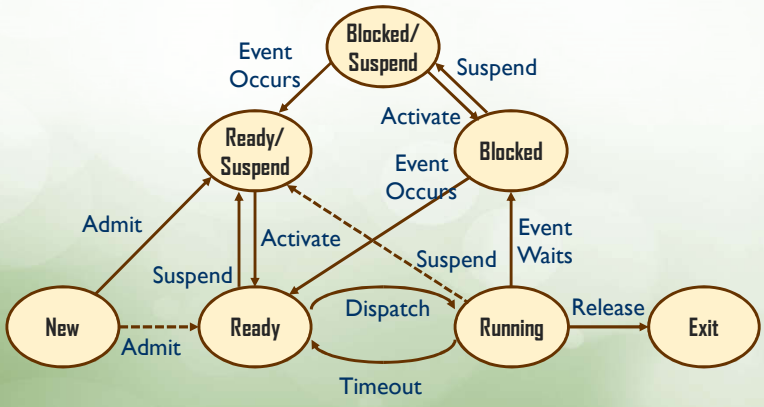
- In fact, ready processes may also be suspended if not enough resources!!

⇒ Two new possible states: Blocked/Suspend, Ready/Suspend

Seven-State Model

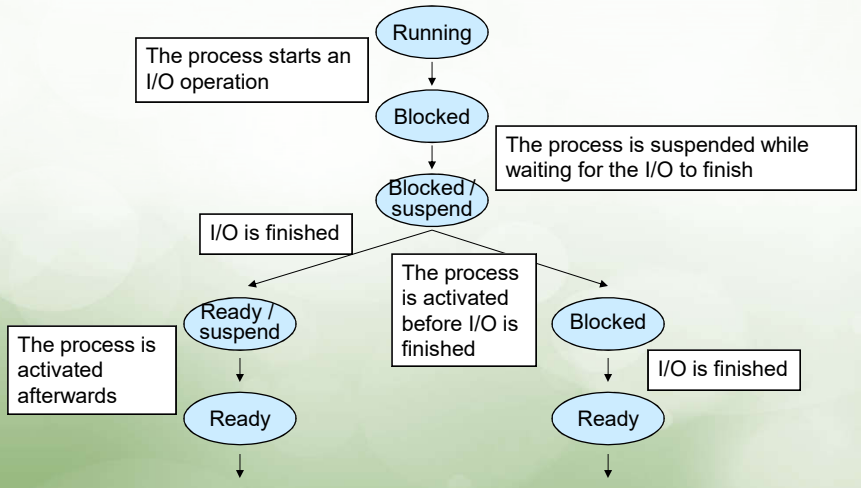
- Two suspended states

What's the meaning of the two new states?
Why not a single 'suspend' state?
Why no 'running, suspend' state?



Eddie Law 39

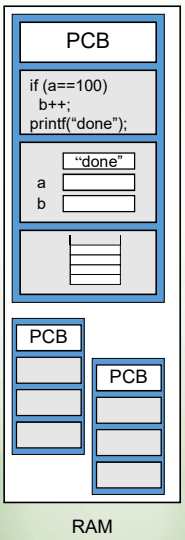
An Example of State Transition



Eddie Law 40

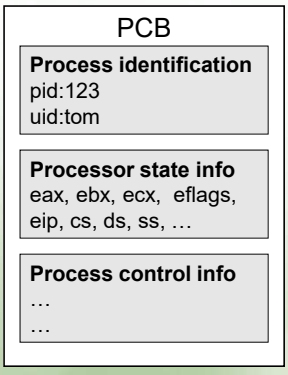
Process Image

- OS maintains the following for each process
  - Process control block (PCB)
  - User program (.EXE, .DLL, etc)
  - User data (global and dynamic variables, constant, buffer, …)
  - User stack (local variables, calling address of procedure)



Process Control Block (PCB)

- PCB contains data that the OS needs to control the process
- PCB consists of
  - Process identification
  - Processor state information
  - Process control information



Process Identification

- Process ID, a unique numeric identifier
- User identifier
  - Who is responsible for the job, or who runs the process
  - Used to determine what access rights the process has

Processor State Information

- Processor state: contents of processor registers, incl.
  - User-visible registers
  - Control and status registers, incl. Program Counter, condition codes and status in PSW
  - Stack pointers (which point to the top of the stack)
- Used to save and restore the processor state in process switching

Process Control Information

- Additional information needed by the operating system to control and coordinate the various active processes
  - scheduling and state information,
    e.g. process state, priority, waiting event, etc.
  - data structuring,
    e.g. for keeping the ready queue as linked list, parent-child relationship

Process Control Information (cont'd)

- …
- interprocess communication
- process privileges
- memory management
- resource ownership and utilization,
  e.g. open files, and a history of utilization of the processor or other resources

Mechanisms for Interrupting Process Execution

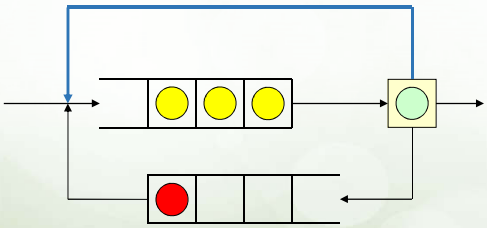
Mechanism	Cause	Use
Interrupt	External to the execution of the current instruction	Reaction to an asynchronous external event
Trap	Associated with the execution of the current instruction	Handling of an error or an exception condition
Supervisor call	Explicit request	Call to an operating system function

When to Switch Processes

- Interrupt - interruption request from hardware external to the CPU
  - Clock
  - I/O: I/O completion, I/O error
- Traps - an error condition or exceptional condition associated with the current instruction
  - Memory fault (details in ch. 7, 8)
- Supervisor call - call of some special functions of the OS
  - File open
  - Synchronization primitive (details in ch. 5)

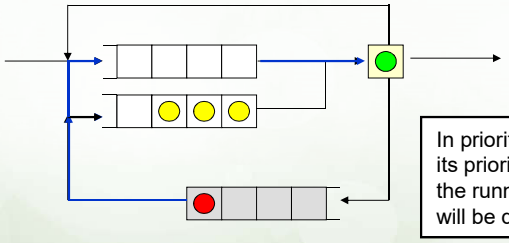
Process Switching at Clock Interrupt

Timeout checked in clock interrupt



The OS determines whether the currently running process has been executing for the maximum allowable unit of time, called quantum. If so, this process must be switched to a Ready state and another process dispatched.

Process Switching at I/O Interrupt

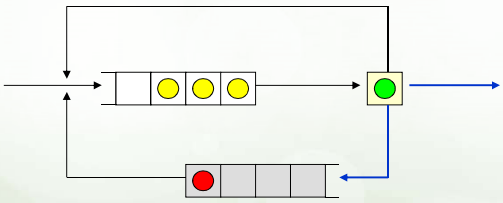


In priority scheduling, if its priority is higher than the running process, it will be dispatched.

When the I/O operation is finished for a blocked process, the process is waken up (moved to Ready state or Ready/Suspend state).

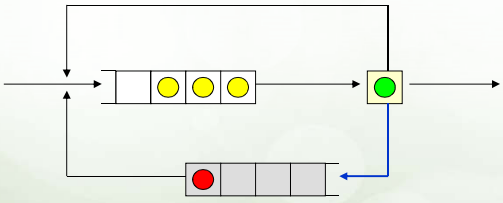
In case of I/O error (interrupt), the blocked process that is waiting for the I/O may be terminated.

Process Switching at Trap



If the error condition caught is fatal (e.g. illegal instruction), the OS moves the process to Exit state. For other traps (e.g. memory fault), the OS may suspend the process and perform some recovery procedure before resuming the process.

Process Switching at Supervisor Call



When a process executes a supervisor call (e.g. for I/O, synchronization), the OS regains control and may move the process to a Blocked state.

Steps of Process Switching

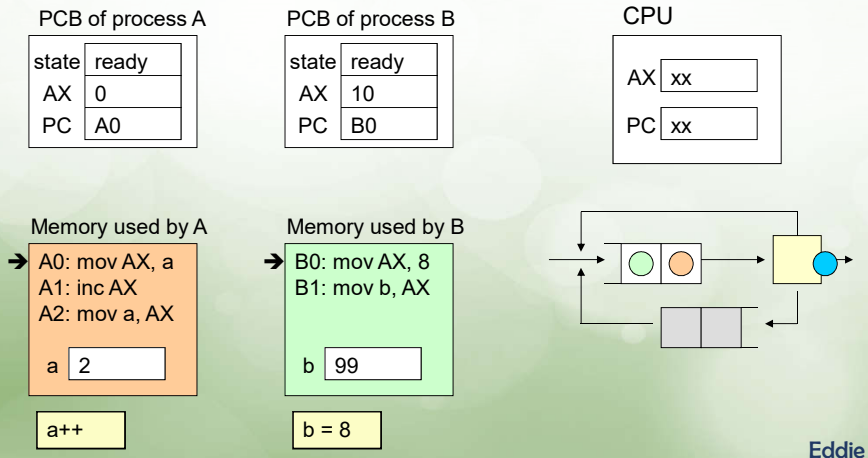
- Consists of the following steps
  1. Save processor state (incl. program counter and other registers) in the PCB: processor state information
  2. Update the PCB with the new state and any accounting information
  3. Move the PCB to appropriate queue – ready, blocked

Steps in Process Switching (cont'd)

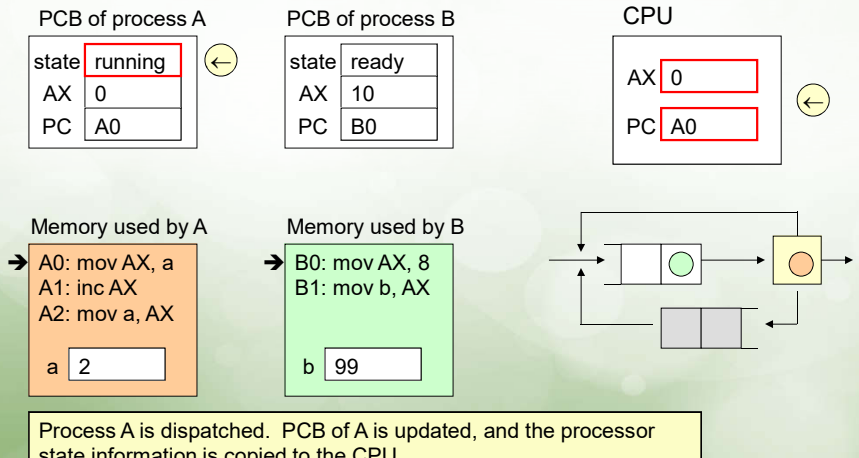
4. Select another process for execution
5. Update the PCB of the process selected (new state and accounting information)
6. Update memory-management data structures - depends on how address translation is managed
7. Restore context of the selected process
   - restore the previous value of the program counter and other registers from the PCB

Example (1)

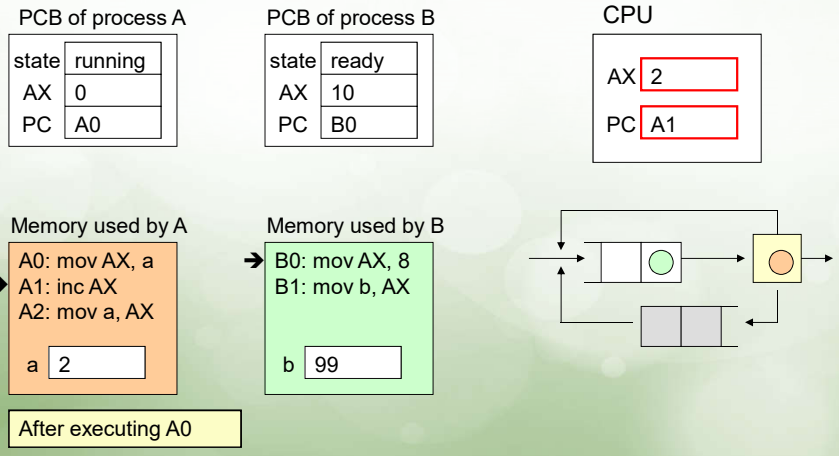
- Two active processes, a time slice of 2 instruction cycles



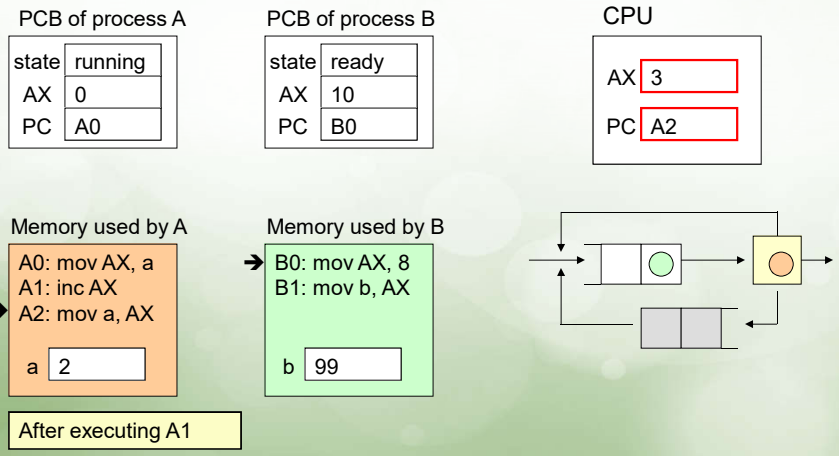
Example (2)



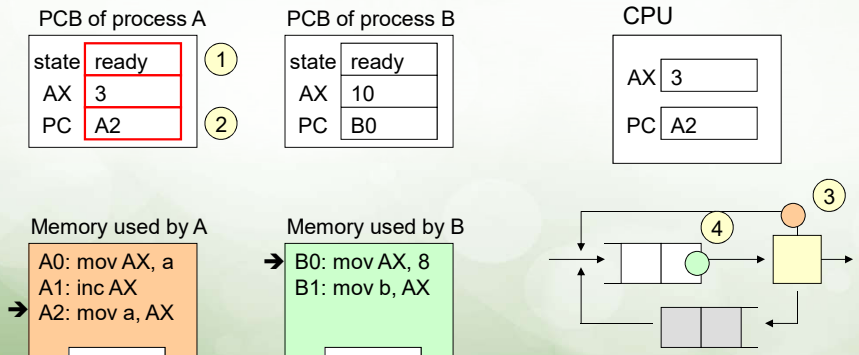
Example (3)



Example (4)

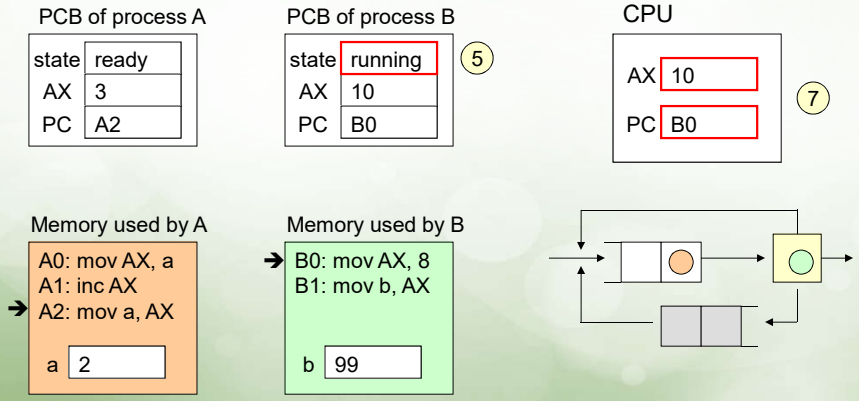


Example (5)



Timeout for Process A, which state changes from running to ready. PCB of A is updated, and the processor state information is copied from the CPU.

Example (6)



Process B is dispatched. PCB of B is updated, and the processor state information is copied to the CPU.

Example (7)

PCB of process A

state	ready
AX	3
PC	A2

PCB of process B

state	running
AX	10
PC	B0

CPU

AX	8
PC	B2

Memory used by A

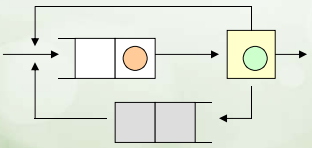
A0: mov AX, a
A1: inc AX
A2: mov a, AX

a 2

Memory used by B

B0: mov AX, 8
B1: mov b, AX

b 8



After executing B0 and B1

Example (8)

PCB of process A

state	ready
AX	3
PC	A2

PCB of process B

state	ready	1
AX	8	
PC	B2	2

CPU

AX	8
PC	B2

Memory used by A

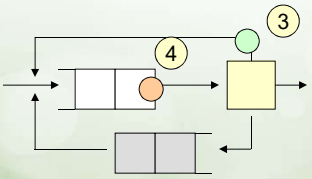
A0: mov AX, a
A1: inc AX
A2: mov a, AX

a 2

Memory used by B

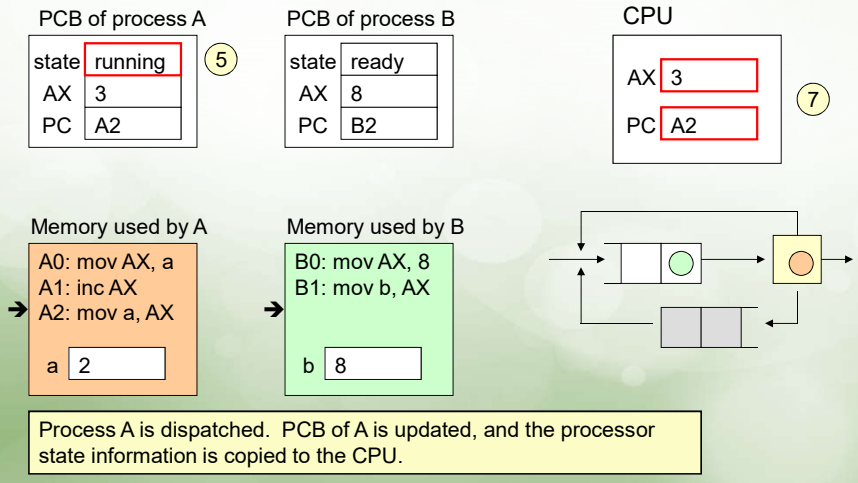
B0: mov AX, 8
B1: mov b, AX

b 8



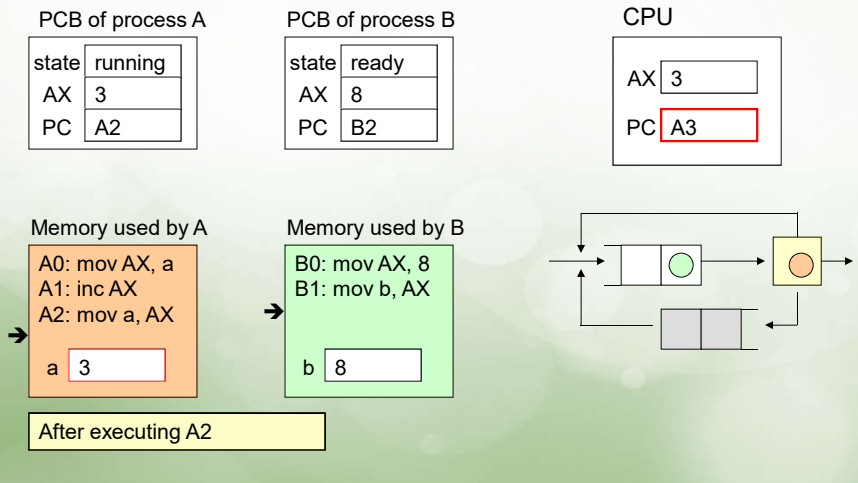
Timeout for Process B, which state changes from running to ready. PCB of B is updated, and the processor state information is copied from the CPU.

Example (9)



Process A is dispatched. PCB of A is updated, and the processor state information is copied to the CPU.

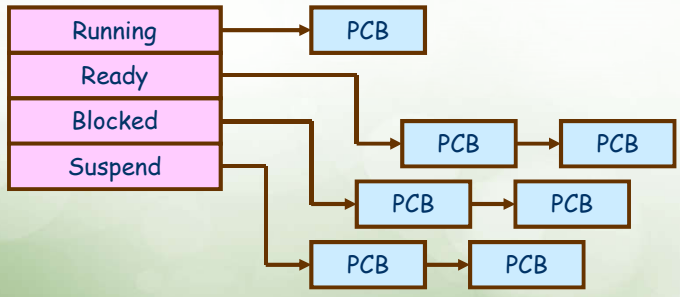
Example (10)



After executing A2

Process List Structures

- States, queues and lists



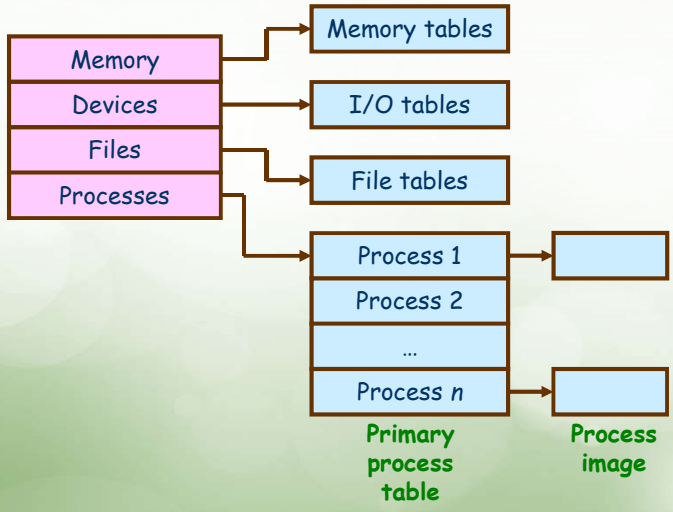
Eddie Law

Operating System Control Structures

- Associate available resources to all processes
- Tables are constructed for each entity
  - Memory tables
  - I/O tables
  - File tables
  - Process table

Eddie Law

Control Tables



Eddie Law

Conclusion

- What is a process?
- Process state (5 state model, 7 state model)
- Process scheduling (round-robin, priority)
- Process Image and PCB
- Process switching (when, how)

Eddie Law 68

Next Topic

- Threads
- Read Ch. 4