Department of Physics,
Computer Science & Engineering

CPSC 410 – Operating Systems I

Chapter 3: Process Description & Control

Keith Perkins

Adapted from original slides by Dr. Roberto A. Flores

Chapter 3 Topics

- Everything about Processes
  - Elements
  - Control blocks
  - States
  - Description
  - Control
- OS Execution
- Security Issues

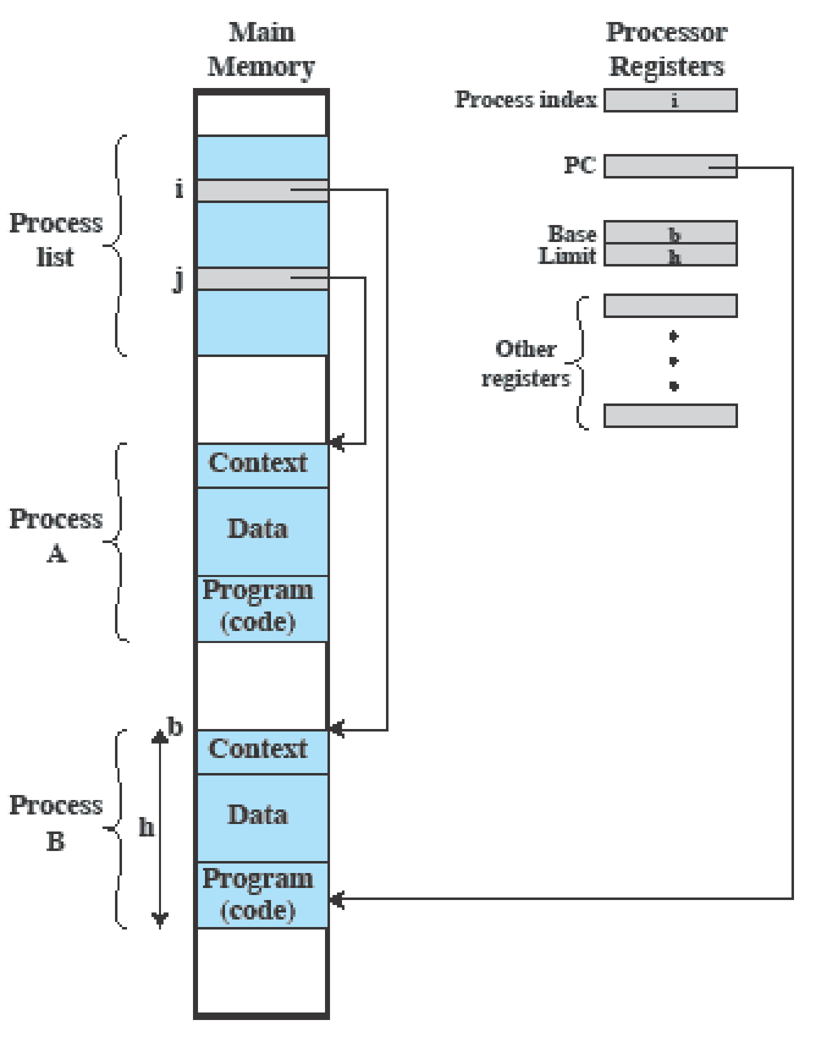
Processes

Elements

- Code
  - accessed by 1+ threads
- Data
  - consumed/produced by code (program state)
  - used by OS to control processes (process control block)

Revisit - Process Management

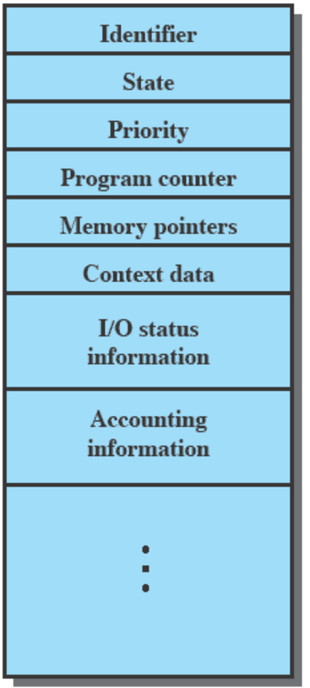
- Scheduler chooses a process to run (more later)
- Dispatcher runs it
- How? What's in the Process List?
- BTW this list is a simplification



Processes

Control Blocks

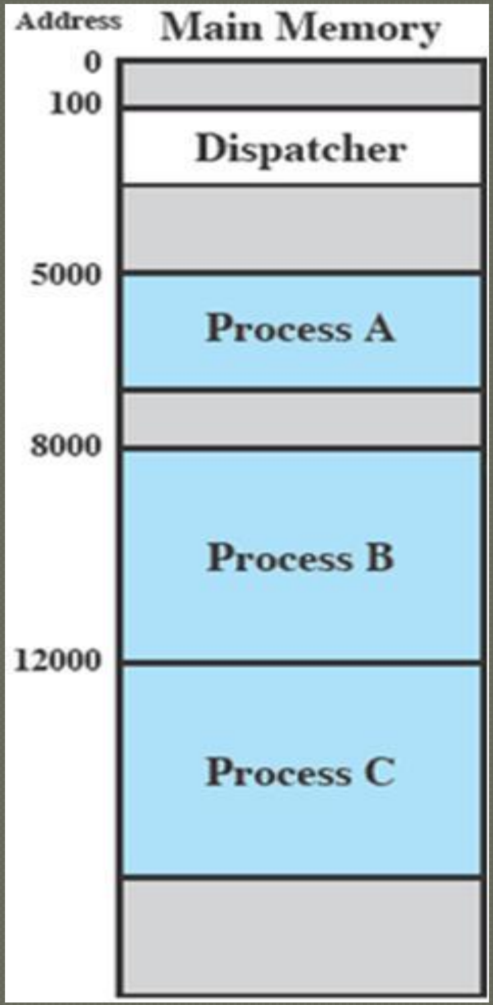
- data structure created & managed by OS
  - Identifier: unique ID
  - State: (e.g., running, blocked)
  - Priority: relative to other processes
  - Program counter: address of next instruction
  - Memory pointers: to code & data
  - I/O status: I/O in use/pending
  - Accounting: CPU time used, IDs, …
- data to hold/restore process state on interrupt/resume
  - key to support multiprocessing


Identifier
State
Priority
Program counter
Memory pointers
Context data
I/O status information
Accounting information
⋮

Elements
Control blocks
States
Description
Control

Processes

- Dispatcher
  - Program that switches processes in/out of the CPU

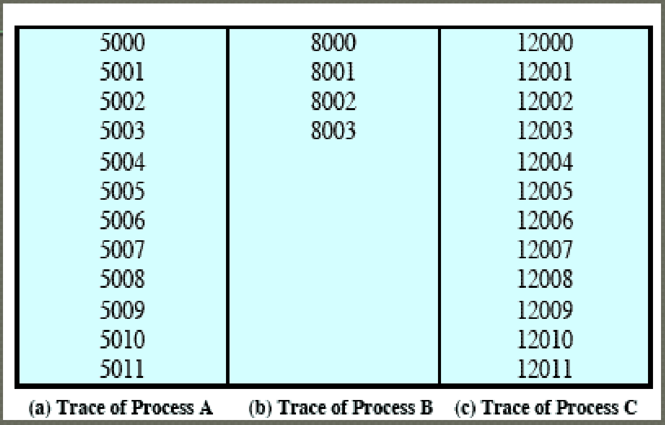


Elements
Control blocks
States
Description
Control

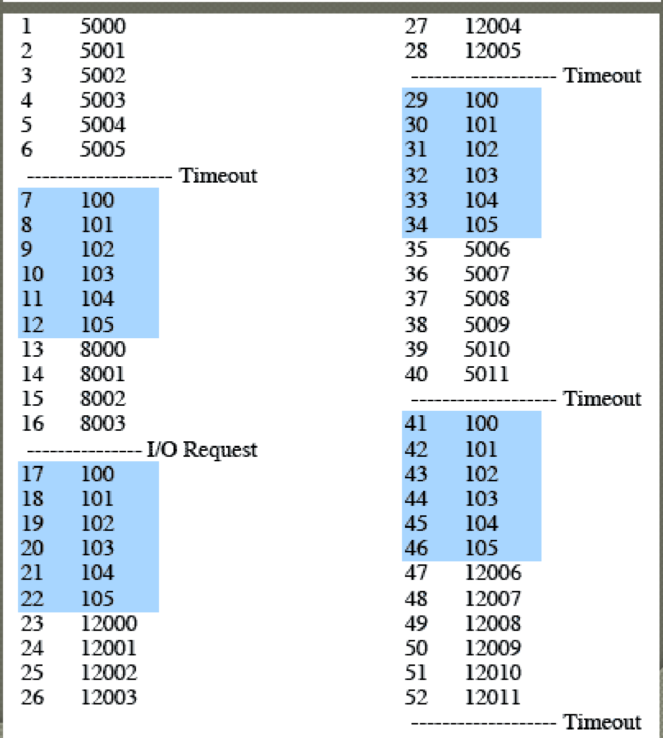
Processes

States

- Trace
  - Instructions executed by a process
  - In multiprogramming:
    - interleaving of instructions as processes alternate using the CPU
- The pale blue lower right
- is dispatcher code



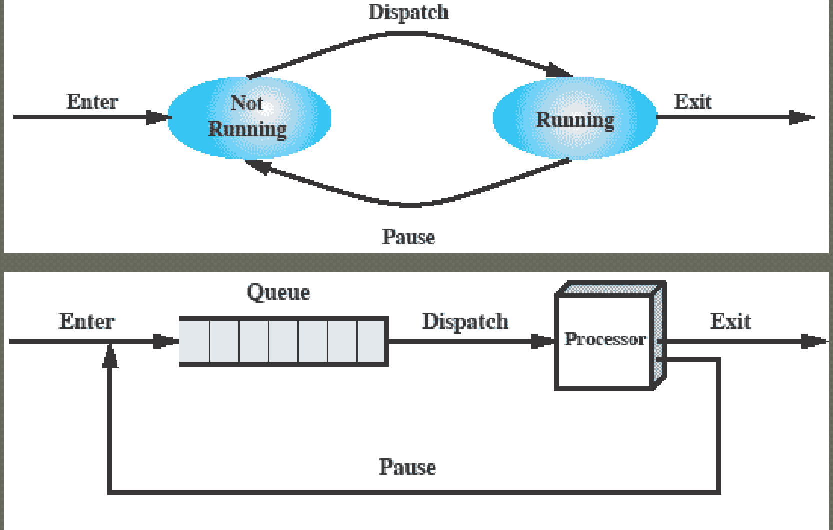
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011
(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C



1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
-------------------	Timeout
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
---------------	I/O Request
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003
27	12004
28	12005
-------------------	Timeout
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
-------------------	Timeout
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
-------------------	Timeout

Processes

- States (2 states)
  - One CPU
  - Round-robin (timeout)
- Running: CPU time!
- Not running: or not



(b) Queuing diagram

- Where do processes come from?
- When do they stop?

Processes

- Where do processes come from? (start)
  - New batch job: Next job in the incoming batch stream
  - Interactive logon: User in terminal logs in
  - OS service: OS-provided service (e.g., print spooler)
  - Spawned by process: uses parallelism (parent spawns child)

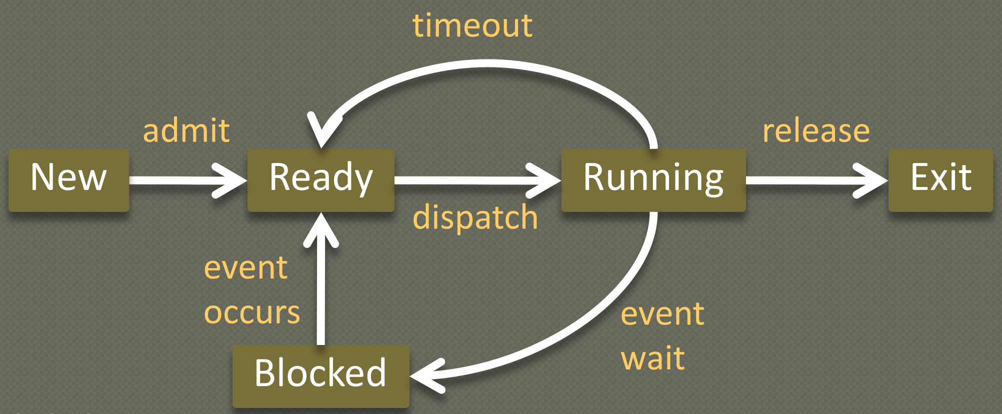
- When do they end? (termination)
  - Normal
    - Job finishes, user logs off, OS shutting down, etc.
  - Abnormal
    - Timeout: running too long
    - Resource error: out of memory, I/O device unresponsive, deadlock
    - Runtime error: arithmetic operation, uninitialized variable
    - Authorization error: memory out of bounds, resource/instruction privilege

Processes

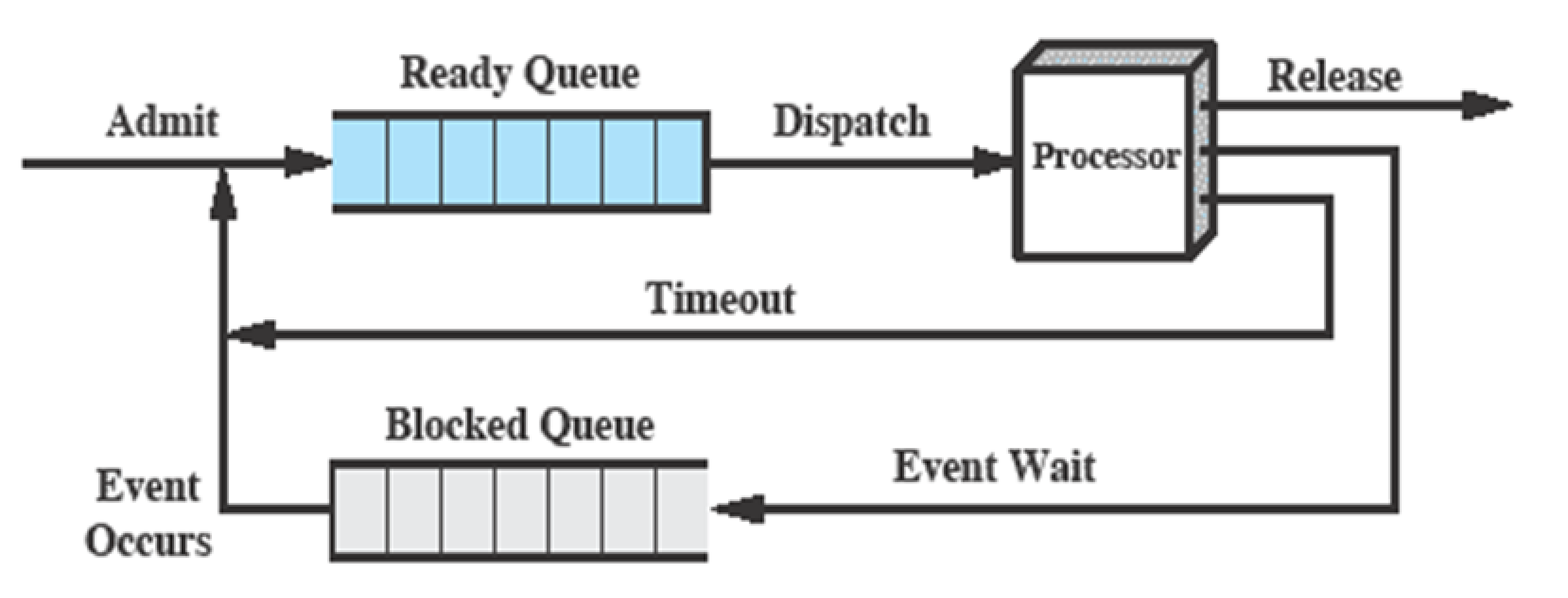
States (5 states)

- non-ready processes may be waiting I/O

- New: not yet in memory
- Ready: awaiting its turn
- Running: CPU time!
- Blocked: waiting for I/O
- Exit: done & gone



Using Two Queues

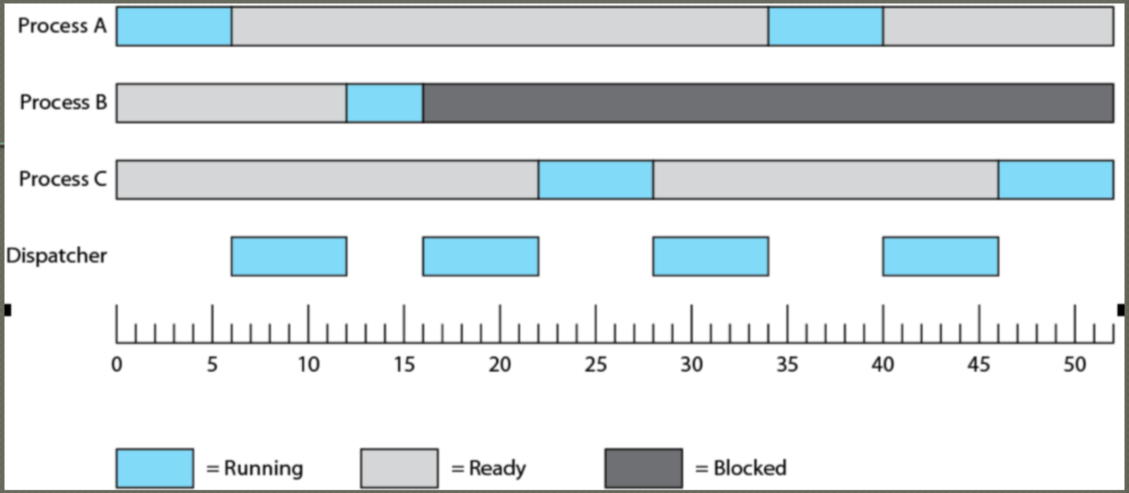


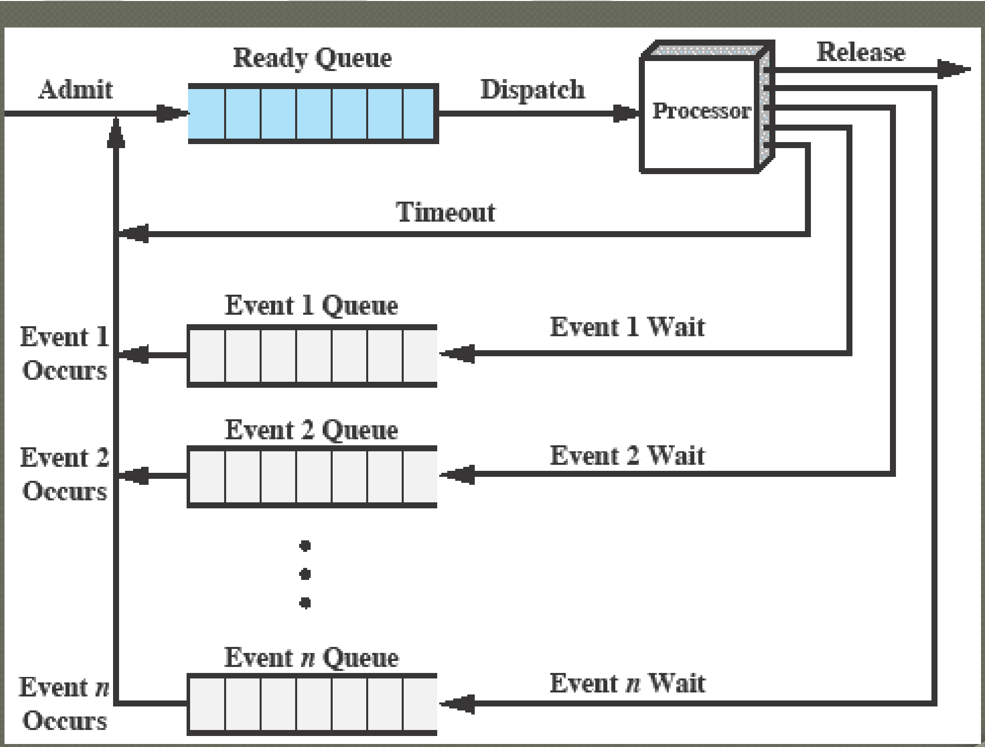
(a) Single blocked queue

Elements
Control blocks
States
Description
Control

States (5 states)

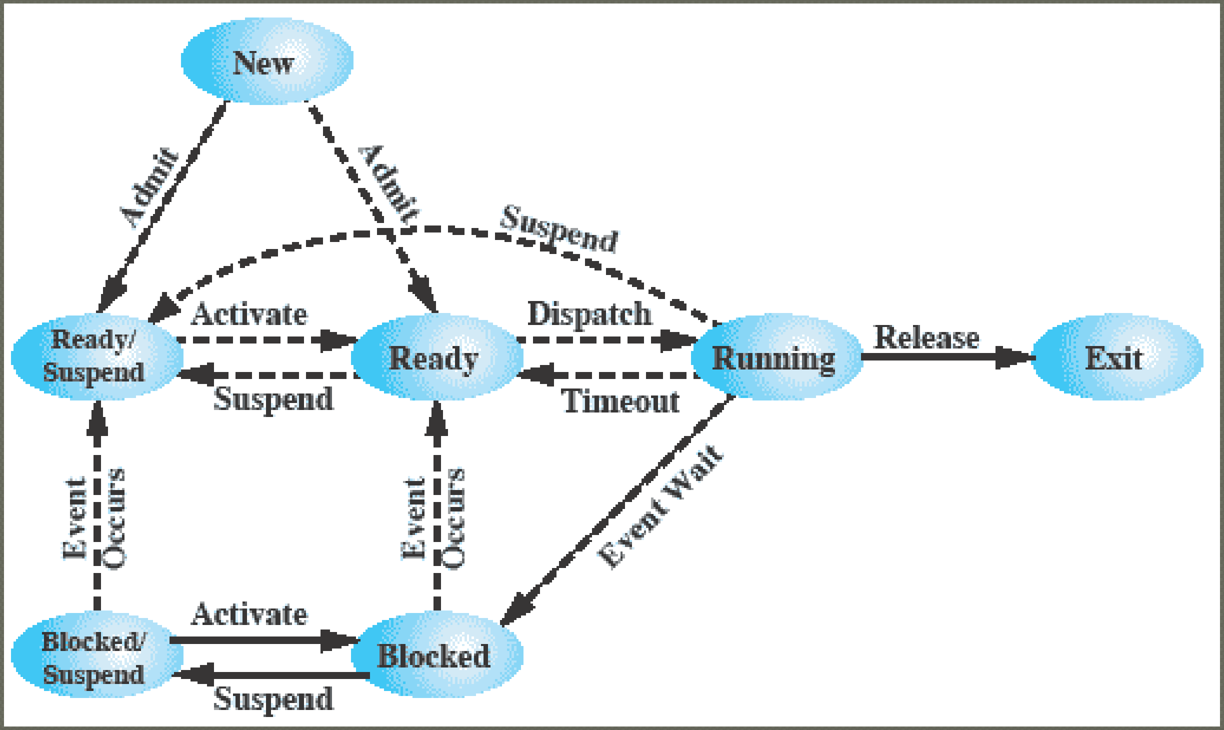
- e.g., Processes A, B & C
- Multiple block queues (1 per I/O device)





Processes

- States (7 states)
  - What if not all processes fit in memory at once?
    - Suspended: when a process has been swapped to disk



Structure of Process

Images in Virtual Memory

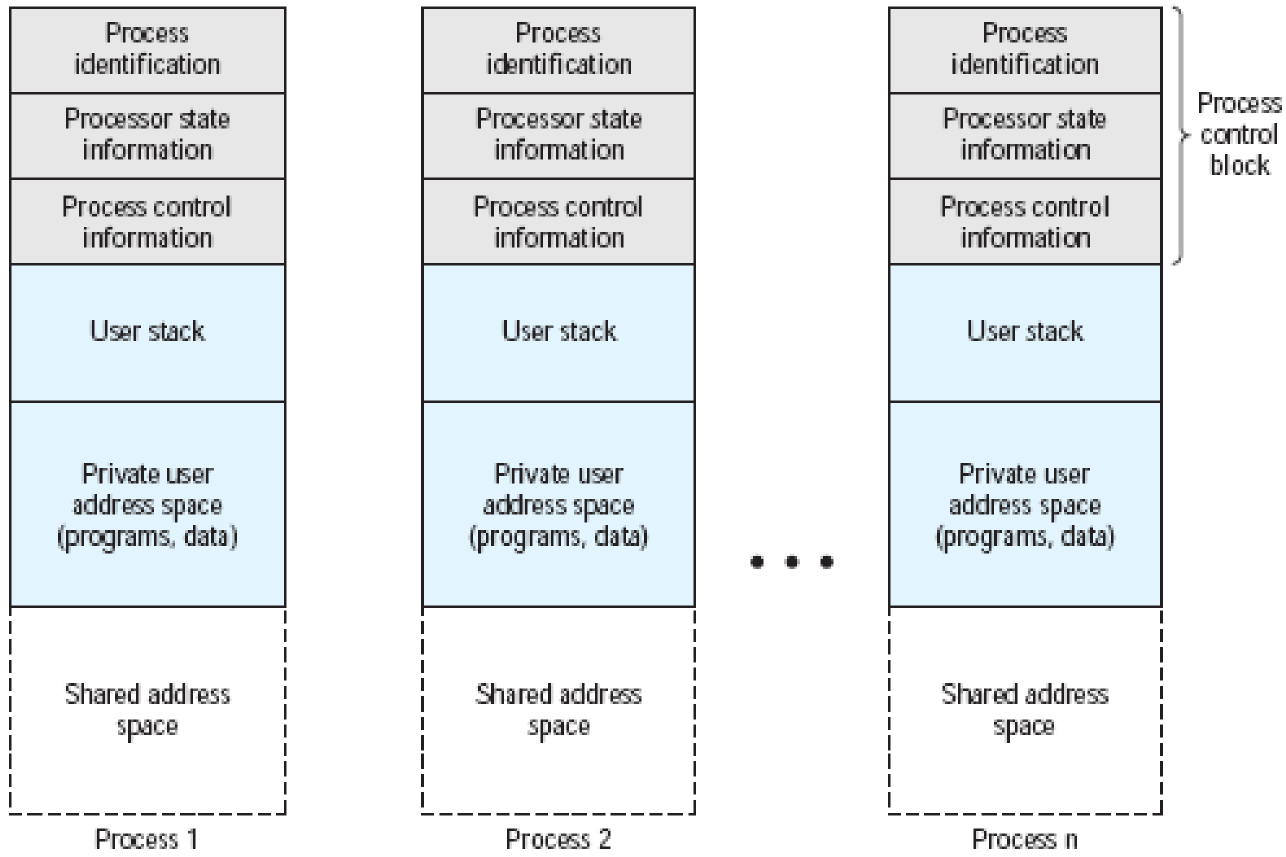

Process identification
Processor state information
Process control information
User stack
Private user address space (programs, data)
Shared address space
Process 1
Process identification
Processor state information
Process control information
User stack
Private user address space (programs, data)
Shared address space
Process 2
• • •
Process identification
Processor state information
Process control information
User stack
Private user address space (programs, data)
Shared address space
Process n
Process control block

Figure 3.13 User Processes in Virtual Memory

Process List Structures

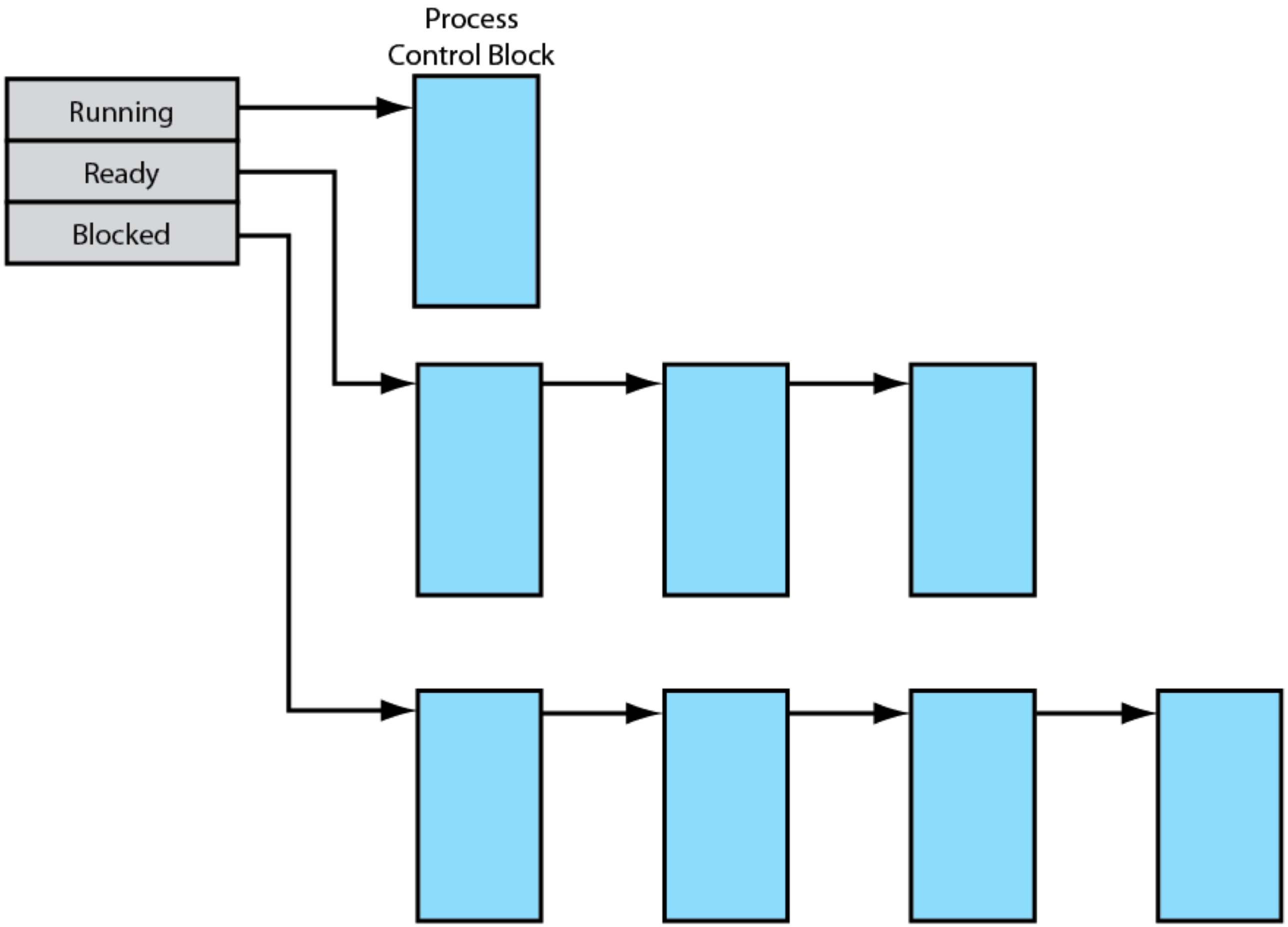
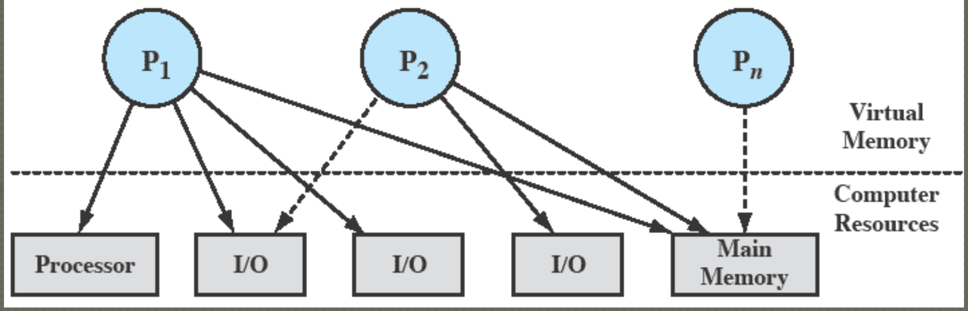


Figure 3.14 Process List Structures

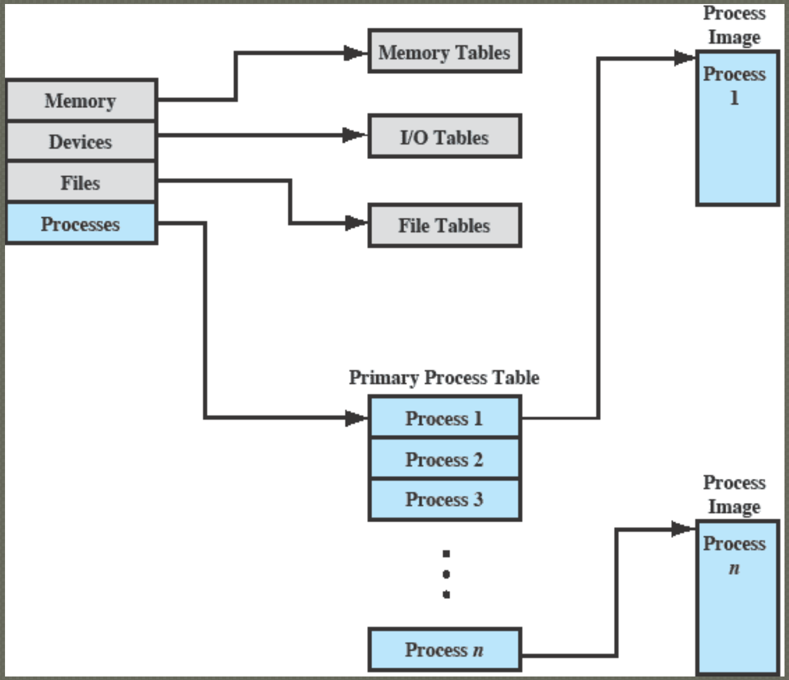
Processes

Description

- Runtime snapshot



- What data structures are implemented in the OS to support them?
  - memory tables
  - I/O table
  - file tables
  - process tables



Processes

- Memory tables
  - keep track of main (real) & secondary (virtual) memory
    - knows about allocation & protection in both memories
    - knows data for managing virtual memory
- I/O tables
  - keep track of I/O device data
  - if I/O operation in progress, keep track of
    - status of I/O operation
    - memory location where data is been transferred
- File tables
  - existing files: location & attributes

Elements
Control blocks
States
Description
Control

Processes

Process tables

- keep data about each process (process image)
  - user data: modifiable part of program, e.g., variables
  - user program: program to execute
  - stack: stores method calls & parameters
  - process control block (PCB): data OS uses to control process
    - process identification: process/parent/user ID
    - processor state information: user/control registers, stack pointers
    - process control information: scheduling, inter-process comms, …
- reference (directly/indirectly) memory, I/O & file tables

Processes

Process tables

Process identification

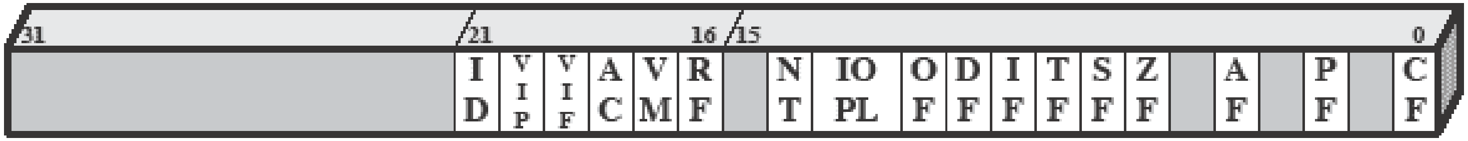
- Each process has a unique ID
- IDs are used for reference:
  - in other tables
  - in inter-process communication
  - when a parent spawns a child process

- process identification: process/parent/user ID
- processor state information: user/control registers, stack pointers
- process control information: scheduling, inter-process comms, …

- reference (directly/indirectly) memory, I/O & file tables

Process state information

- stack pointers
- user-visible registers
- control & status registers
  - program status word (PSW), e.g., EFLAGS in x86 processors



ID	=	Identification flag	DF	=	Direction flag
VIP	=	Virtual interrupt pending	IF	=	Interrupt enable flag
VIF	=	Virtual interrupt flag	TF	=	Trap flag
AC	=	Alignment check	SF	=	Sign flag
VM	=	Virtual 8086 mode	ZF	=	Zero flag
RF	=	Resume flag	AF	=	Auxiliary carry flag
NT	=	Nested task flag	PF	=	Parity flag
IOPL	=	I/O privilege level	CF	=	Carry flag
OF	=	Overflow flag			

- processor state information: user/control registers, stack pointers
- process control information: scheduling, inter-process comms, …

- reference (directly/indirectly) memory, I/O & file tables

Elements
Control blocks
States
➡ Description
Control

Processes

Process tables

- keep data about each process (process image)

Process control information

Scheduling: process state, priority, events awaiting (if any)

- Data structures: relationship with other processes, e.g., blocked, child
- Inter-process communication: state of current communications (if any)
- Privileges: to access instructions, resources/services
- Memory: references to process pages
- Resources: resources used (if any)

➡ process control information: scheduling, inter-process comms, …

- reference (directly/indirectly) memory, I/O & file tables

Processes

Control

- Modes of execution
  - User mode (-privileged) … Kernel mode (+privileged)
- Process creation
  - What does OS do when a process is created?
    - assigns a new unique ID
    - allocates space for the process
    - initializes its process control block & sets it in place (e.g., ready list)
- Process switching
  - Process is running…what events can give control back to OS?
    - interrupt: reaction to asynchronous external event (clock, I/O, …)
    - trap: reaction to an error or exception (recovery…?)
    - supervisor call: call to an OS instruction

Processes

Control

- Process is running…is an interrupt pending?
  - If not, fetch next instruction
  - If yes, point PC to interrupt handler, switch to kernel mode
- Process is running…but it's changing state
  - (e.g., running->blocked) what does OS do?
    - save context of the processor
    - update process control block (PCB)
    - move PCB to appropriate queue (e.g., to blocked list)
    - select another process for execution (e.g., from ready list)
    - update PCB of process selected
    - update memory management data structures
    - restore context of the processor

Chapter 3 Topics

- Everything about Processes
    - Elements
    - Control blocks
    - States
    - Description
    - Control
- OS Execution
- Security Issues

OS Execution

- OS is software, right?
  - How is it different from just another process?
  - How is it controlled?

a) Non-process Kernel

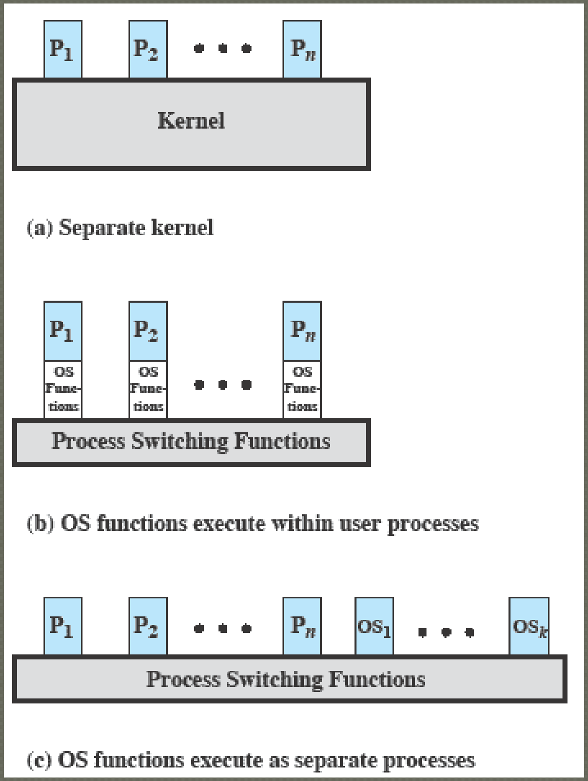
- Processes are processes.
  The kernel is the kernel.

b) Execution within user processes

- OS is a bare process switching mechanism
- OS routines are linked to user programs (OS data is shared)

c) Process-based OS

- OS routines run as independent processes
- Modular approach for parallelism (e.g., OS in one CPU, user processes in another)



(a) Separate kernel

(b) OS functions execute within user processes

(c) OS functions execute as separate processes

Execution *Within* User Processes

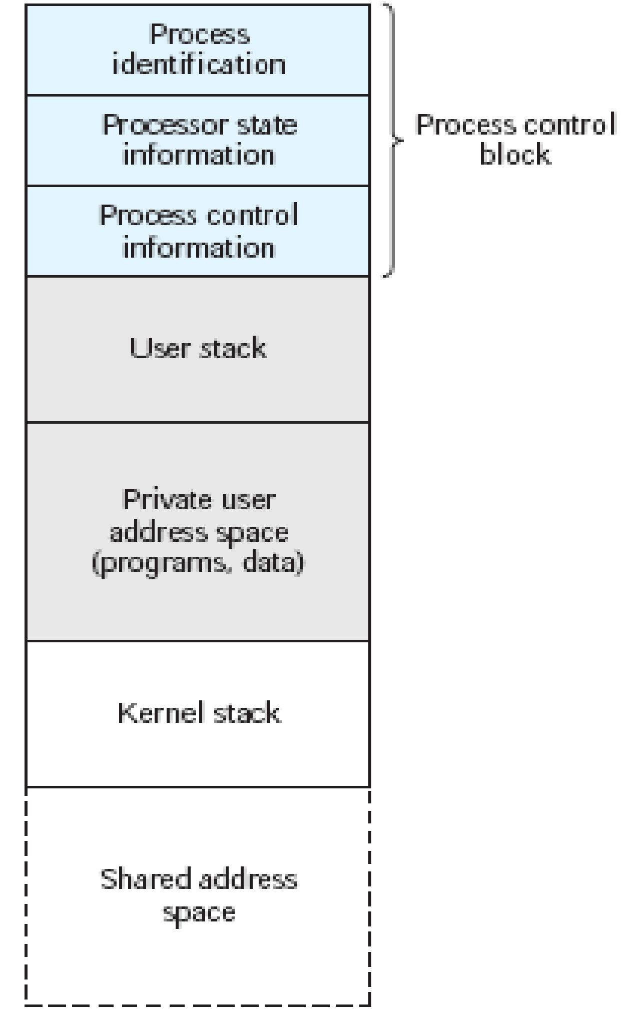

Process identification
Processor state information
Process control information
Process control block
User stack
Private user address space (programs, data)
Kernel stack
Shared address space

Figure 3.16 **Process Image: Operating System Executes within User Space**

Chapter 3 Topics

- Everything about Processes
  - Elements
  - Control blocks
  - States
  - Description
  - Control
- OS Execution
- Security Issues

Security

- Protecting computer resources
  - OS should prevent (or at least detect) users/malware attempts to gain unauthorized access
  - Privileges
    - Users have privilege levels (highest: administrator/root)
    - Processes have (at most) the same privilege as their user
- Threats
  - A potential violation of security, given a circumstance/capability/action/event breaching security and causing harm.
- Countermeasures
  - An action/technique that eliminates/prevents/minimizes/reports a threat.

[RFC 4949, 2007, http://tools.ietf.org/html/rfc4949]

Security

- Threats
  - Goal: gain access to / increase privileges in system
  - Intruders (hacker | cracker)
    - Misfeasor: user seeking more than allowed | misusing resources
    - Masquerader: non-user posing as legitimate user
    - Clandestine user: (non-) user seeking root privilege
  - Malicious software (malware)
    - Sophisticated (harmless -> crippling)
    - Parasitic (needs host program)
      - virus: self-replicating code embedded into another program
      - logic bomb: routine activated under certain conditions
      - backdoor: non-regular access to system (left by designers)
    - Independent: worm (virus-minus-host)

Security

- Countermeasures
  - Intrusion detection
    - Service monitoring system events, warning about attempts to access resources in an unauthorized manner.
    - 3 logical components
      - sensing >> analyzing >> reporting (UI)
  - Authentication
    - Process of verifying an identity claimed by a system entity.
      - Identification: representative token
      - Verification: examining token
  - Firewalls
    - Computer controlling network traffic (based on policies)

Chapter 3 Topics

- Everything about Processes
  - Elements
  - Control blocks
  - States
  - Description
  - Control
- OS Execution
- Security Issues

Done!