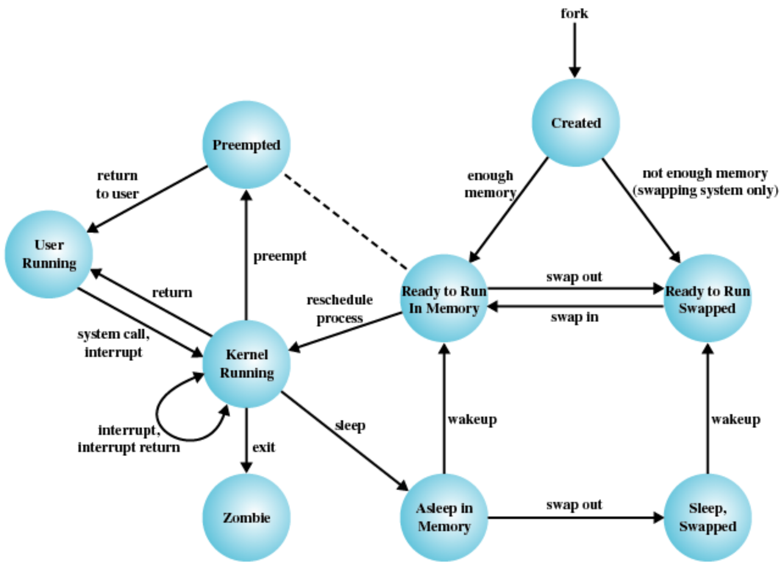
FREIE
UNIVERSITÄT
BERLIN

Operating Systems & Computer Networks
3. Processes

Dr. Larissa Groth
Computer Systems & Telematics (CST)



Roadmap

1. Introduction and Motivation
2. Interrupts and System Calls
3. **Processes**
4. Scheduling
5. Memory
6. I/O and File System
7. Booting, Services, and Security

Lernziele

- Sie nennen:
  - die wesentlichen Aufgaben eines general-purpose BS hinsichtlich Prozesse

- Sie beschreiben:
  - die Rolle des HW Timer Interrupt bei der interleaved execution von Prozessen
  - die Rolle des Process Control Blocks
  - warum vermieden werden sollte, Prozesse im Zustand „Ready/Suspend“ zu haben

- Sie stellen dar und beschreiben:
  - das Extended Process State Diagram

- Sie grenzen voneinander ab:
  - die Begriffe „Programm“, „Prozess“ und „Thread“

Definitions of a Process

Program in execution

Instance of a program running on a computer

- There may be multiple instances of the same program, each as a separate process

Unit characterized by

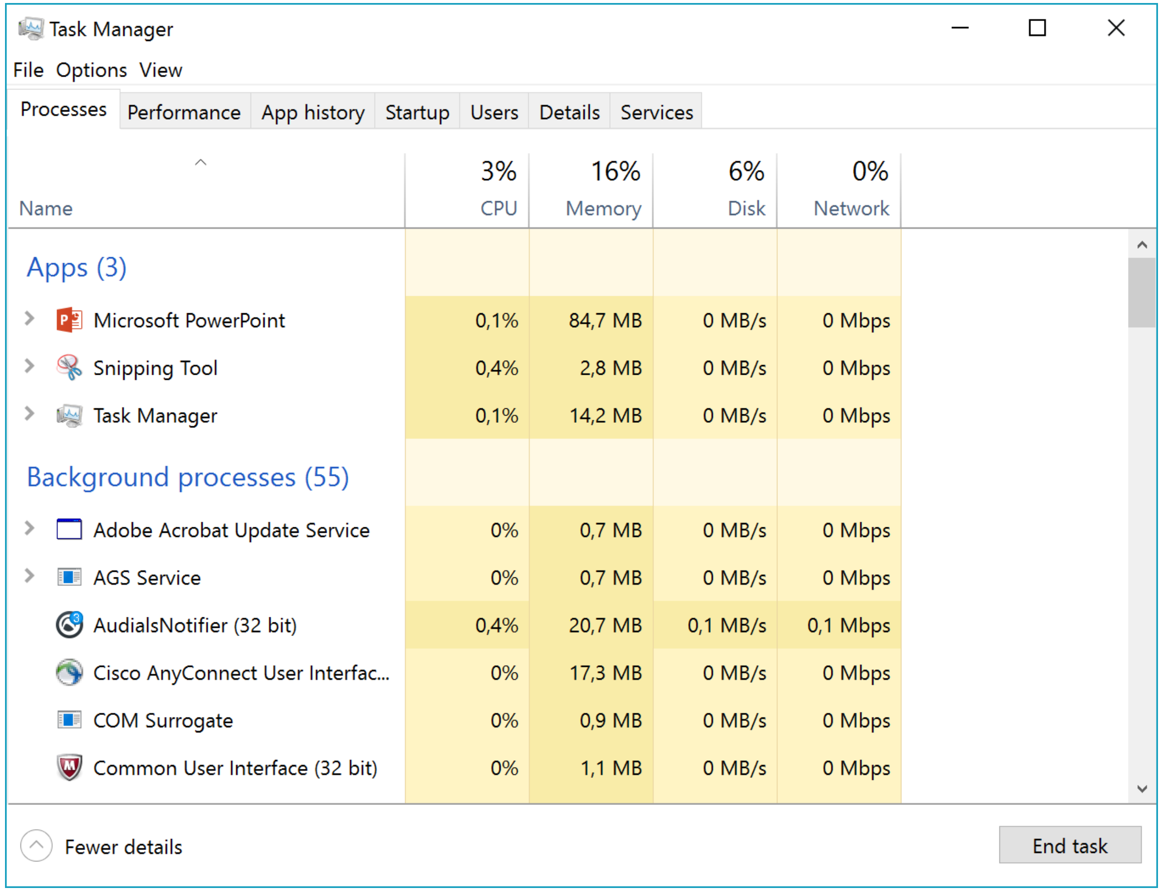
- Execution of a sequence of instructions
- Current state
- Associated block of memory

Related Concepts to “Process”

Thread: One (of several) runtime entities that **share the same address space**

- Easy cooperation, requires explicit synchronization
- A process may consist of several threads

Application: User-visible entity, one or more processes



Name	3% CPU	16% Memory	6% Disk	0% Network
Apps (3)				
Microsoft PowerPoint	0,1%	84,7 MB	0 MB/s	0 Mbps
Snipping Tool	0,4%	2,8 MB	0 MB/s	0 Mbps
Task Manager	0,1%	14,2 MB	0 MB/s	0 Mbps
Background processes (55)				
Adobe Acrobat Update Service	0%	0,7 MB	0 MB/s	0 Mbps
AGS Service	0%	0,7 MB	0 MB/s	0 Mbps
AudialsNotifier (32 bit)	0,4%	20,7 MB	0,1 MB/s	0,1 Mbps
Cisco AnyConnect User Interfac...	0%	17,3 MB	0 MB/s	0 Mbps
COM Surrogate	0%	0,9 MB	0 MB/s	0 Mbps
Common User Interface (32 bit)	0%	1,1 MB	0 MB/s	0 Mbps

Program vs. Process

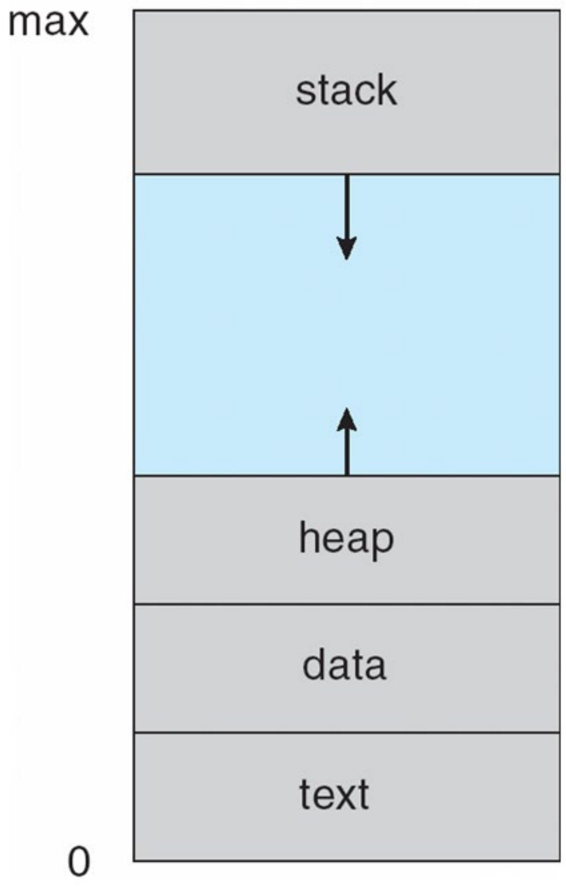
Multiple parts

- Program code → text section
- Current activity → program counter, processor registers
- Stack → temporary data
- Data section → global variables
- Heap → dynamic memory

Program is passive entity, process is active

- Program becomes process when executable file loaded into memory

One program can be several processes



Tasks of an OS concerning processes

On-demand user-level process creation

- Structuring of applications

Interleaved execution (by scheduling) of multiple processes

- Maximization of processor utilization
- Minimization of response time

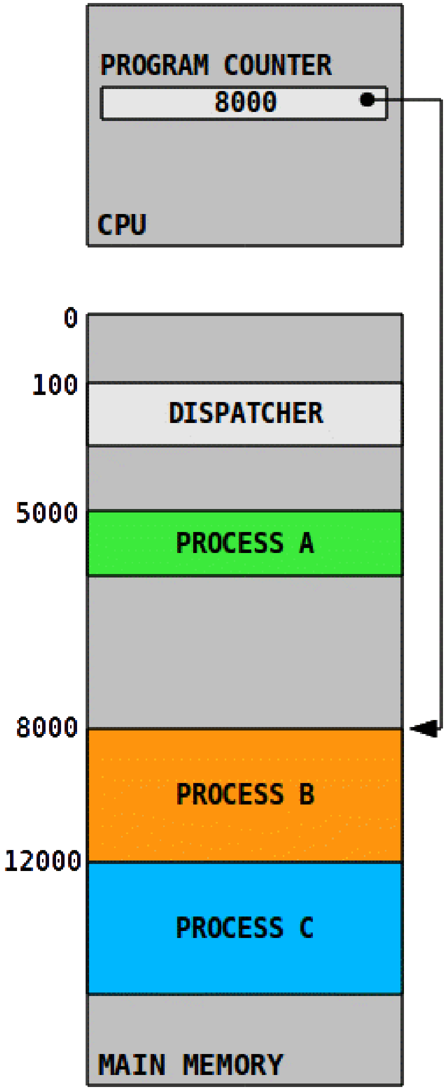
➢ conflicting objectives!

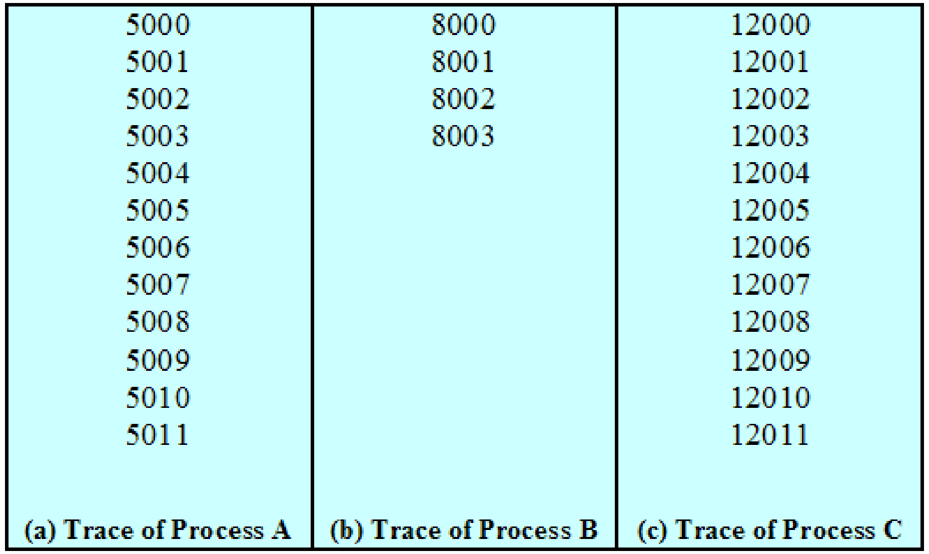
Allocation of resources for processes

- Consideration of priorities
- Avoidance of deadlocks

Support for Inter-Process Communication (IPC)

Process execution (Trace)



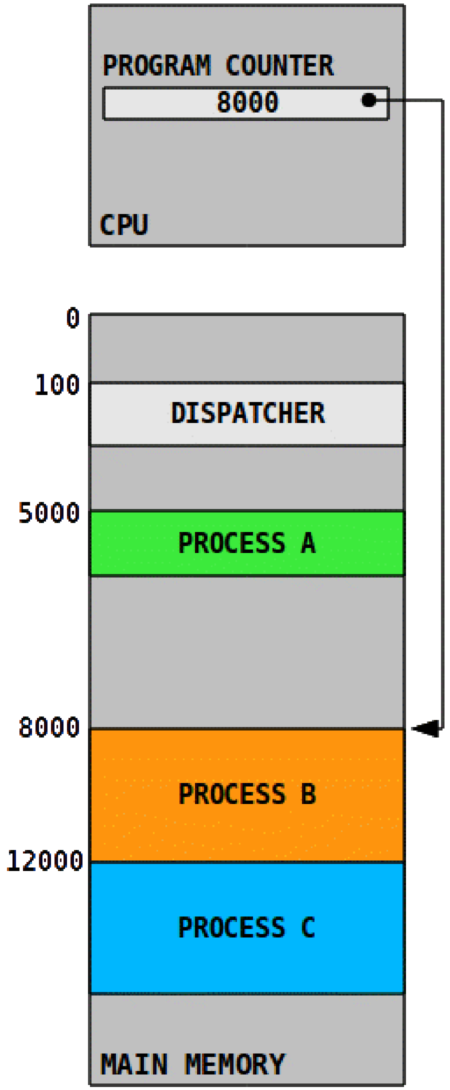


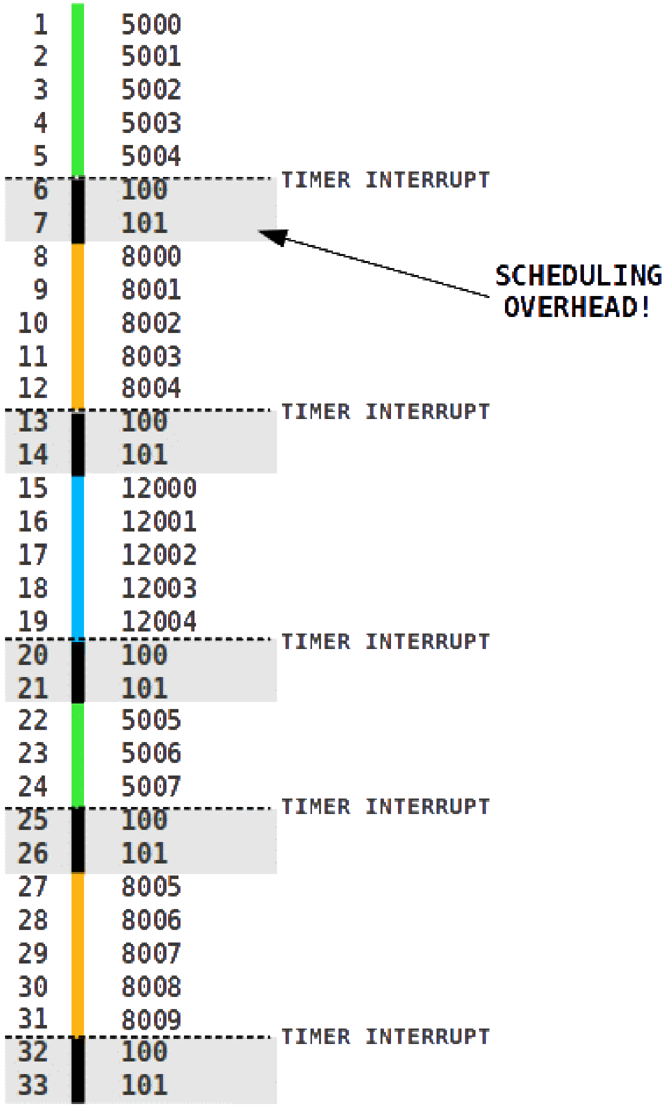
(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

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

Figure 3.3 Traces of Processes of Figure 3.2

Process execution (Trace)



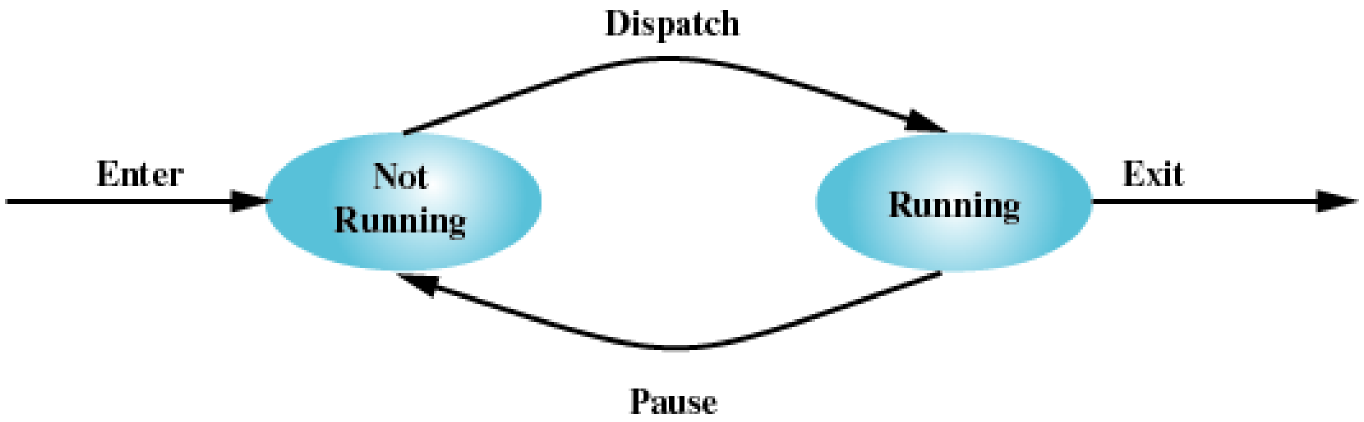


1	5000	
2	5001	
3	5002	
4	5003	
5	5004	
6	100	TIMER INTERRUPT
7	101	
8	8000	
9	8001	
10	8002	
11	8003	
12	8004	
13	100	TIMER INTERRUPT
14	101	
15	12000	
16	12001	
17	12002	
18	12003	
19	12004	
20	100	TIMER INTERRUPT
21	101	
22	5005	
23	5006	
24	5007	
25	100	TIMER INTERRUPT
26	101	
27	8005	
28	8006	
29	8007	
30	8008	
31	8009	
32	100	TIMER INTERRUPT
33	101	

Simple Process Model

Process is in one of two states:

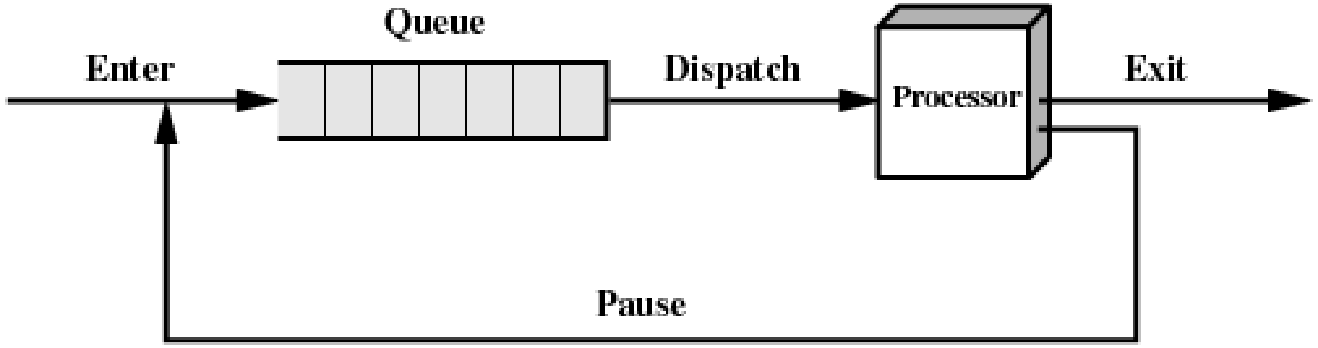
- running
- not running



How to implement?

Simple Process Model - Implementation

Running processes managed in queue:



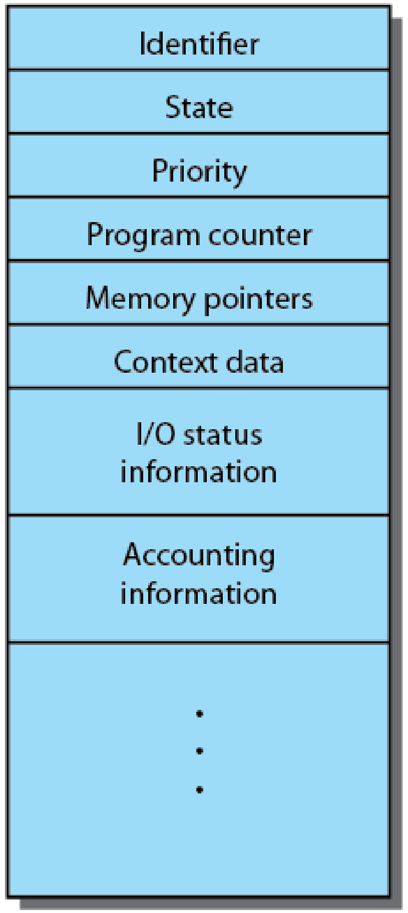
What information is required?

Process Control Block (PCB)

Definition: OS data structure which contains the information needed to manage a process (one PCB per process)

Process identifiers	• IDs of process, parent process, and user
CPU state	• User-visible registers • Control and status registers: • Stack pointer (SP) • Program counter (PC) • Processor status word (PSW)
Control information	• Scheduling information: • Process state, priority, awaited event • Accounting information: • Amount of memory used, CPU time elapsed • Memory management: • Location and access state of all user data • I/O management: • Devices currently opened (files, sockets)

Process Control Block (PCB)



Identifier

State

Priority

Program counter

Memory pointers

Context data

I/O status information

Accounting information

•
•
•

Figure 3.1 Simplified Process Control Block

Events that cause Process Creation

1) System initialization
- Operating system creates first process at boot time
- All other processes are always spawned by existing process
  → Processes are organized in a tree-like structure (`pstree`)

2) Execution of process-creation system call
- Separation of a program into separate processes for algorithmic purposes

3) User requests to start new process

4) Initiation of a batch job
- Applies only to batch systems found on large mainframes
- OS decides if it has the resources for another job
  → Takes next batch job and creates process from it

Process Termination

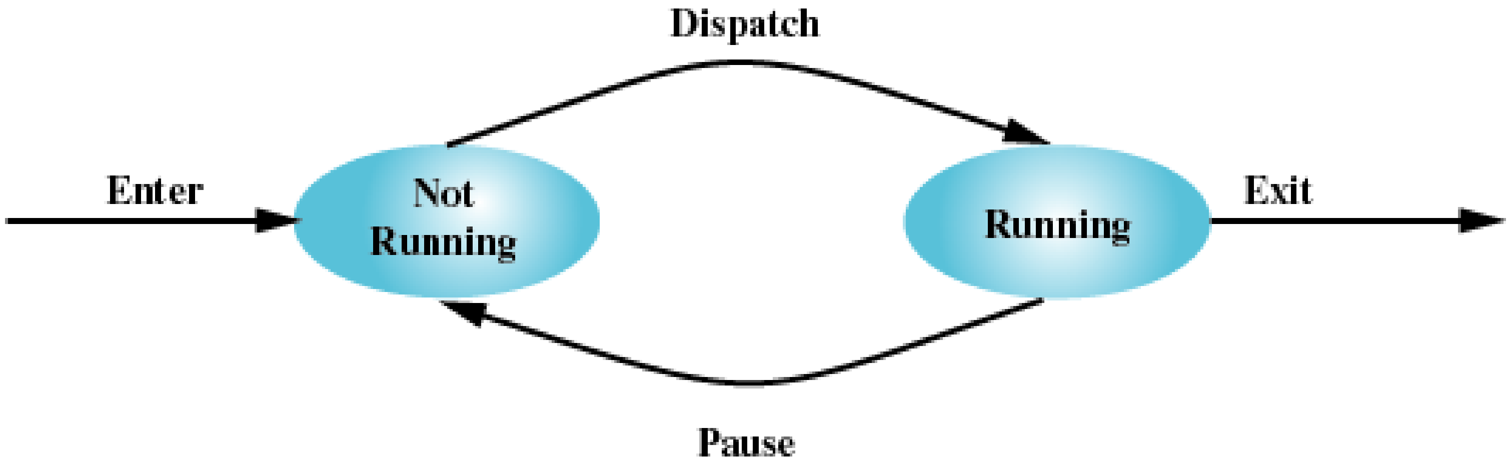
1) Normal exit
- voluntary
- execution of process is completed
- process terminates itself by system call

2) OS terminates process for protection reasons
- Invalid instruction (process tries to execute data)
- Privileged instruction in user mode
- Process tries to access memory without permission
- I/O-Error
- Arithmetic error

3) Killed by another user process
- involuntary
- Parent process or other authorized processes

Process Model

Simple model with two states



Problems

- Most of the processes will be waiting for IO
- Different IO devices
- Different priorities

➔ Extend the model

Extended Process Model

Five states including creation, termination, and resource handling:

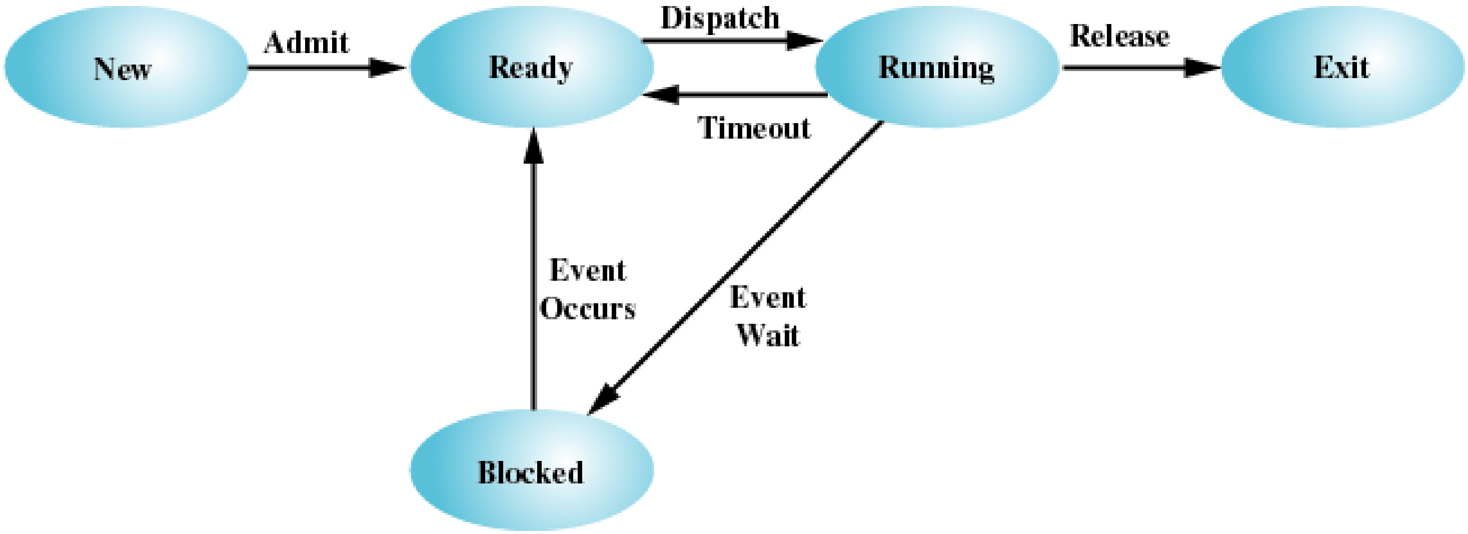
Running: currently being executed

Ready: ready to run, waiting for execution

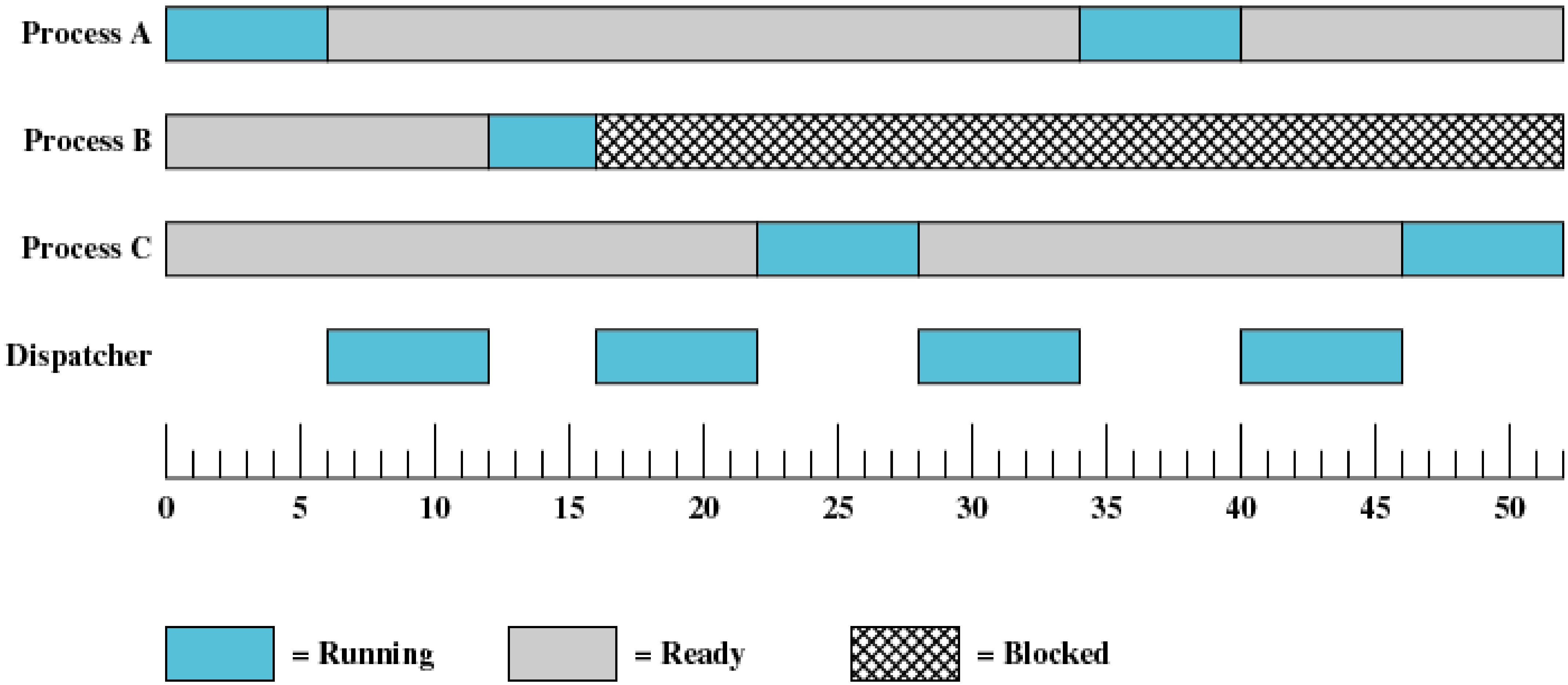
Blocked: not ready to run, waiting for external event, e.g., completion of I/O operation

New: newly created process, not yet in running set

Exit: completed/terminated process, removed from running set



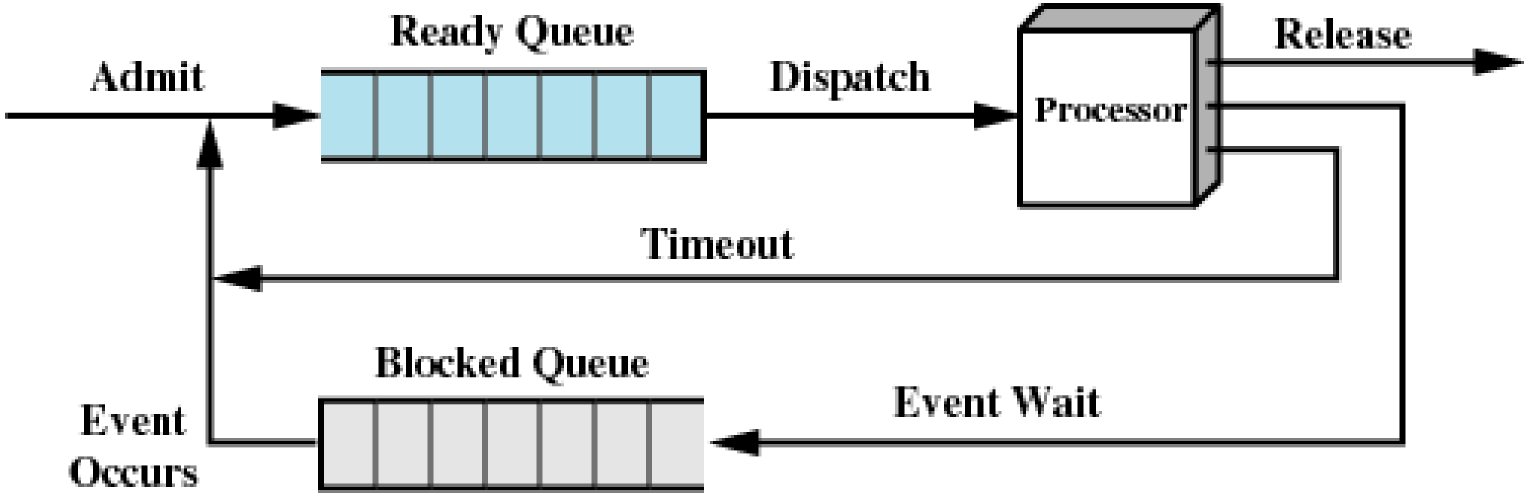
Process States over Time



Implementation of Process States

Assign process to different queues based on state of required resources
Two queues:

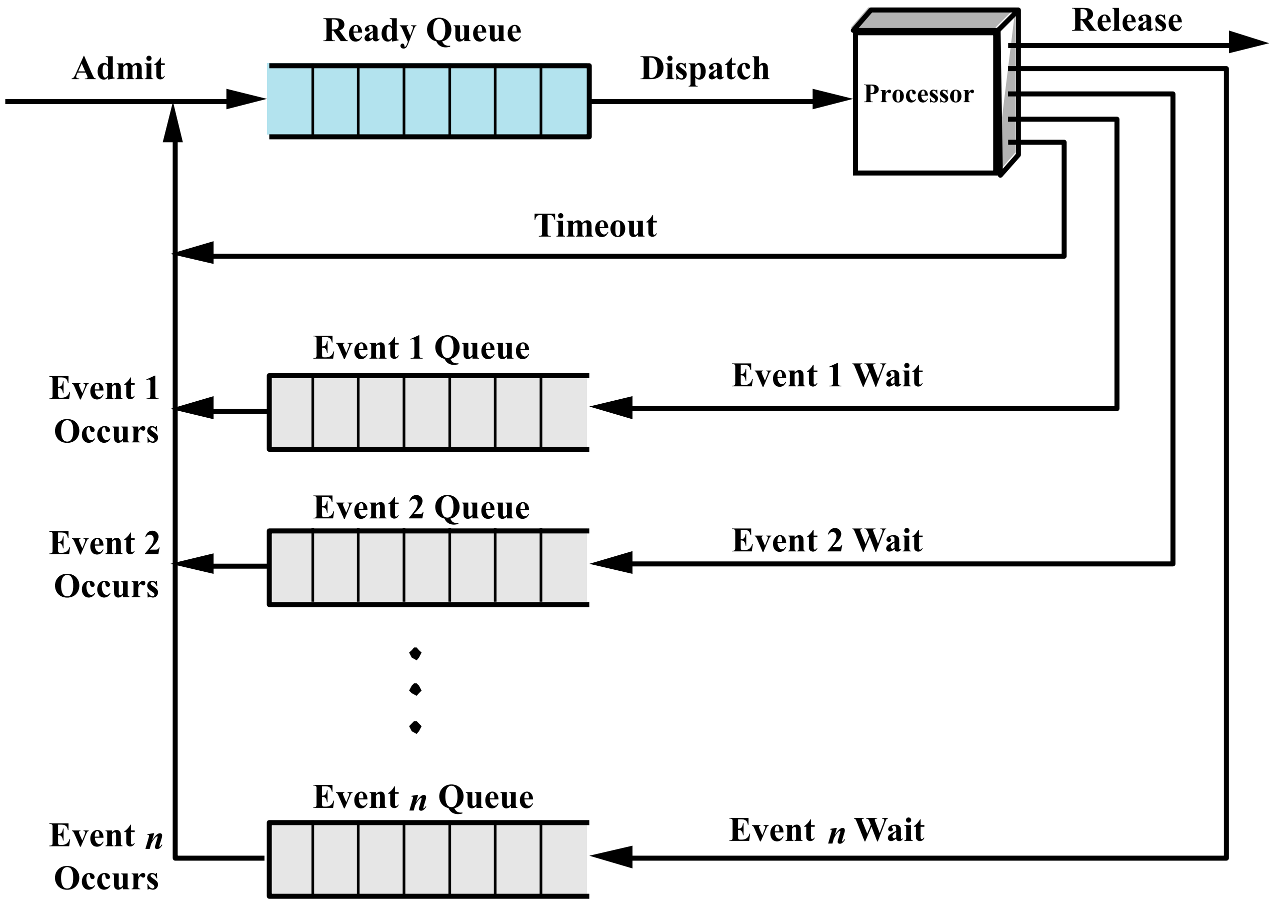
- Ready processes (all resources available)
- Blocked processes (at least one resource busy)



But what happens if processes need different resources?

Improved Implementation

Several queues one for each resource / type of resource



(b) Multiple blocked queues

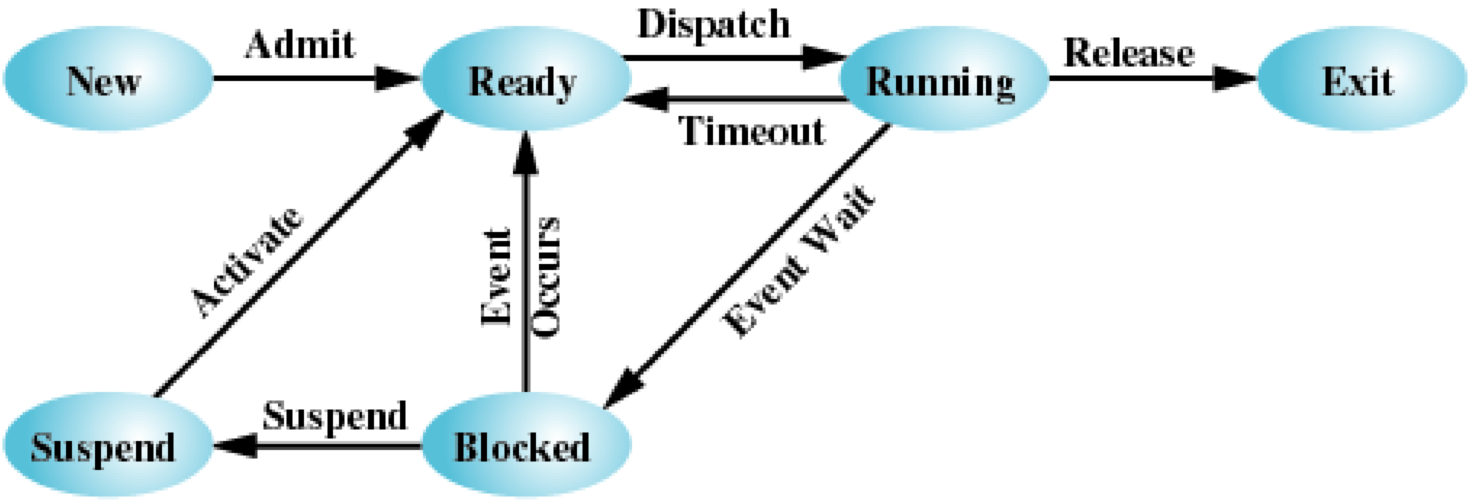
More efficient, but fairness issues must be considered

Suspension / Swapping of Processes

Swapping motivated by two observations:

- Physical main memory is (was) a scarce resource
- Blocked processes may wait for longer periods of time (e.g. during I/O, while waiting for requests, ...)

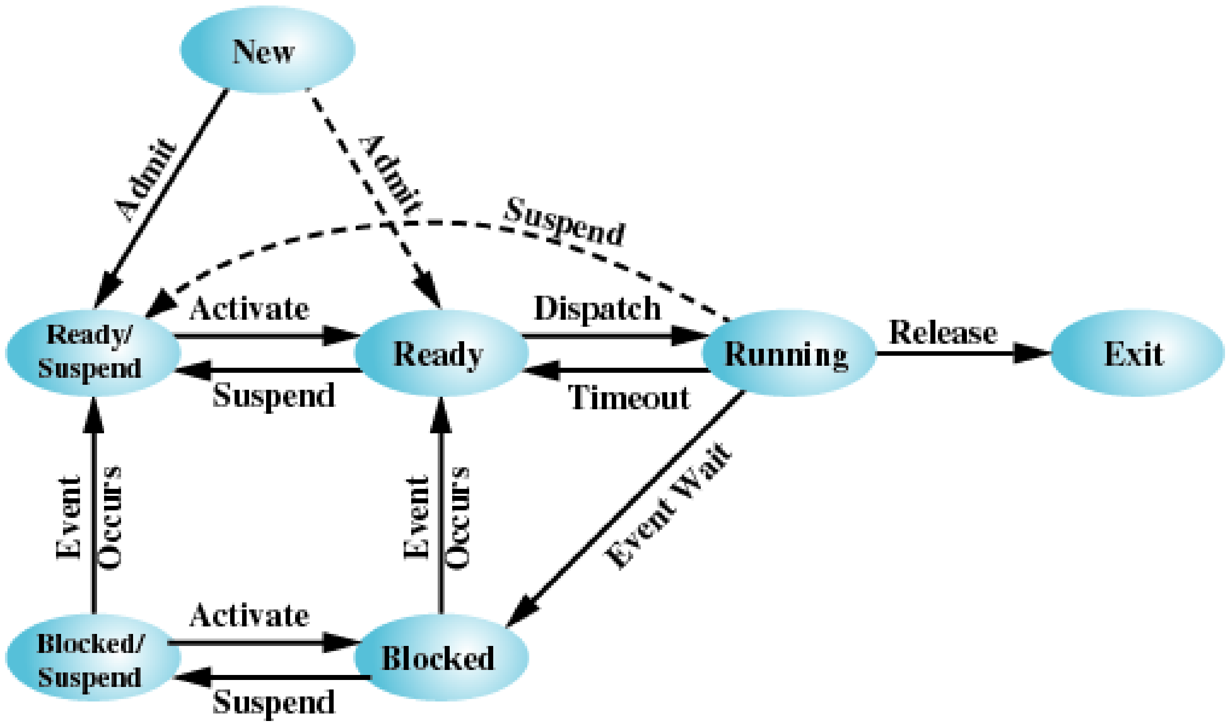
➔ Swap blocked processes to secondary storage thereby reducing memory usage



Extended Process State Diagram

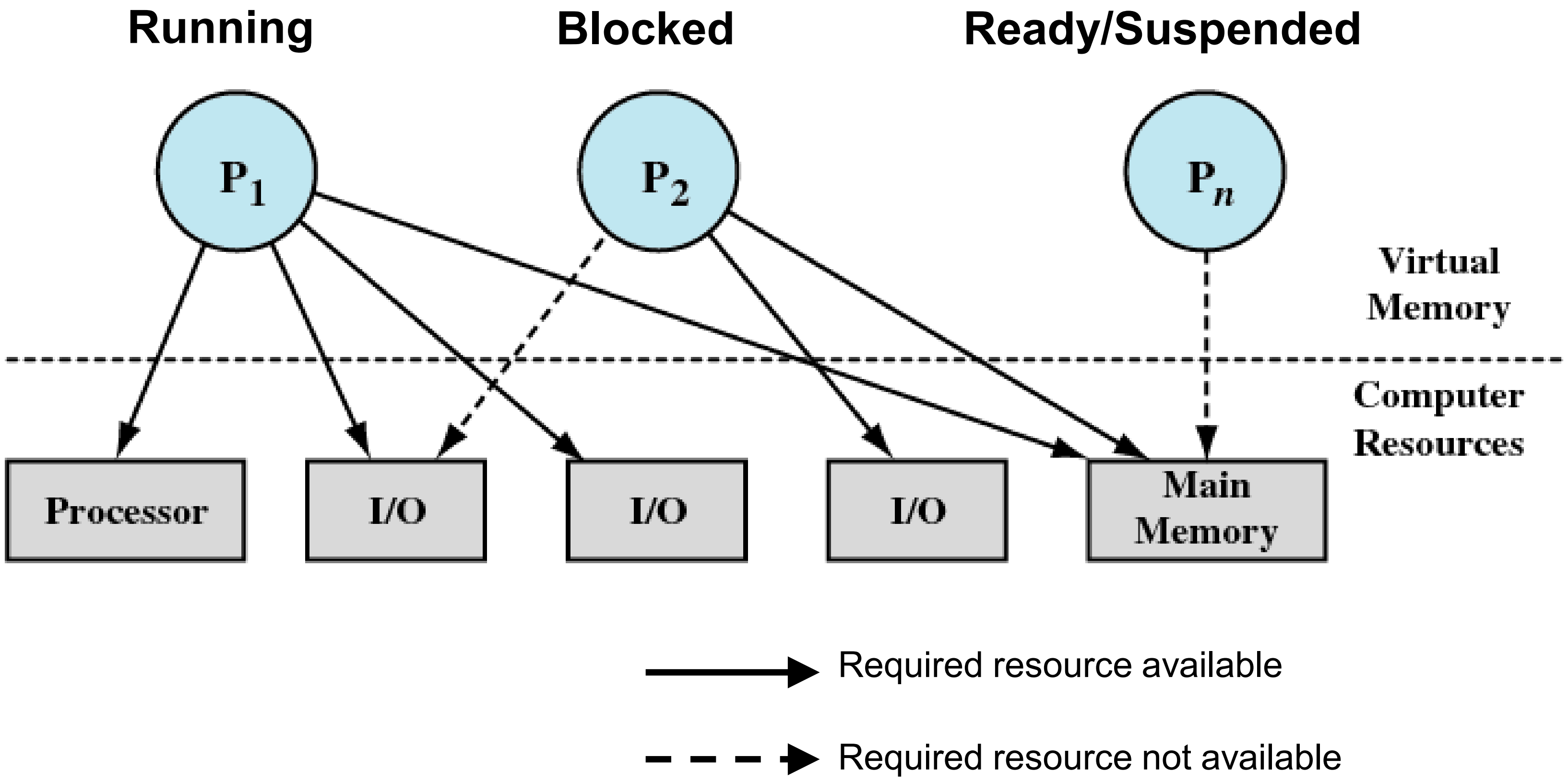
Two additional considerations

- Blocked/swapped processes may become ready to run when event occurs
- Ready and/or running processes may be swapped even without waiting for event



Processes and Resource Allocation

Process state reflects allocated resources:



Global data structures for processes and resources usage

Process tables:

- Process Control Block (PCB)
- Location of process image in memory
- Resources (process-specific view)

Memory tables:

- Allocation of primary and secondary memory
- Protection attributes of blocks of (shared) memory
- Virtual memory management

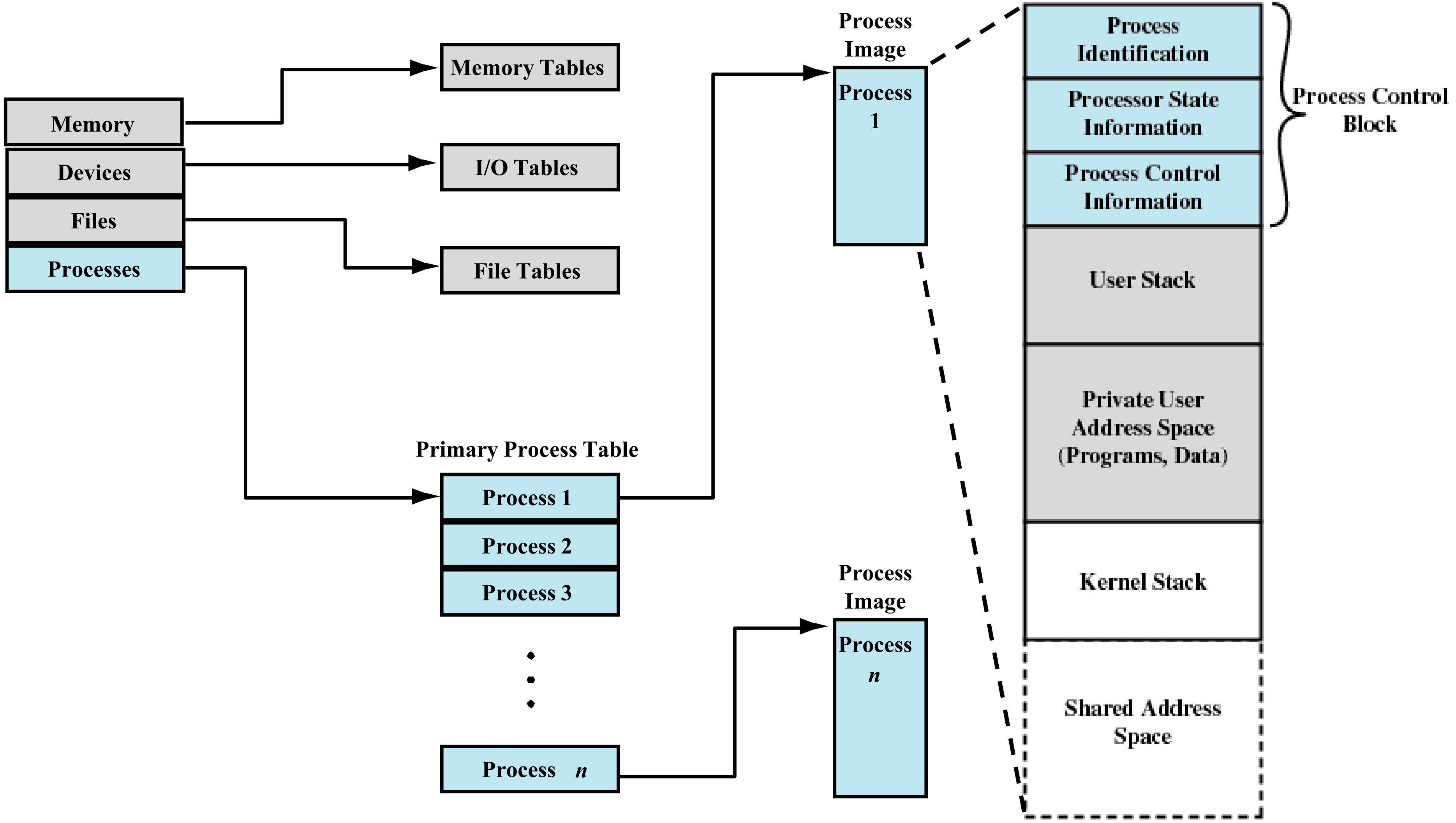
I/O tables:

- Allocation of I/O devices, assignment to processes
- State of current operation and corresponding memory region

File tables:

- Currently open files
- Location on storage media / secondary memory
- State and attributes

Process Control Table and Image

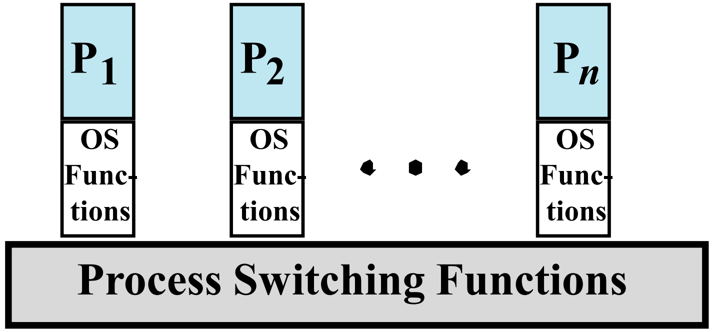


Kernel / Process Implementations

Execution of system calls as part of user process, but in kernel mode:

- Kernel functions use same address space
- Same process switches into privileged mode (Ring 0)

➔ Less expensive and quite safe



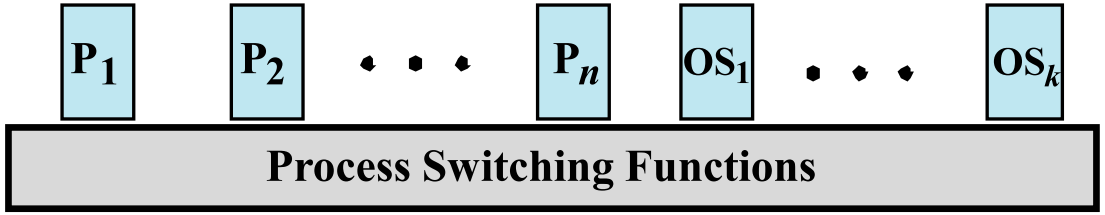
(a) OS functions execute within user processes

Kernel / Process Implementations

Microkernel:

- Collection of system processes that provide OS services

➔ Quite expensive but very safe



(b) OS functions execute as separate processes

Roadmap

1. Introduction and Motivation
2. Interrupts and System Calls
3. **Processes**
4. Scheduling
5. Memory
6. I/O and File System
7. Booting, Services, and Security