AGH University of Science and Technology

Operating systems (3)

Process Description and Control

Department of Computer Science
Faculty of Computer Science, Electronics and Telecommunications
AGH University of Science and Technology, Krakow, POLAND

Roadmap

- **How are processes represented and controlled by the OS.**
- ***Process states*** which characterize the behaviour of processes.
- ***Data structures*** used to manage processes.
- Ways in which the OS uses these data structures to control process execution.
- Process management in UNIX SVR4.

Requirements of an Operating System

- *Fundamental Task: Process Management*
- The Operating System must
  - Interleave the execution of multiple processes
  - Allocate resources to processes, and protect the resources of each process from other processes,
  - Enable processes to share and exchange information,
  - Enable synchronization among processes.

The OS Manages Execution of Applications

- Resources are made available to multiple applications
- The processor is switched among multiple application
- The processor and I/O devices can be used efficiently

What is a "*process*"?

- *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 the execution of a sequence of instructions, a current state, and an associated set of system instructions

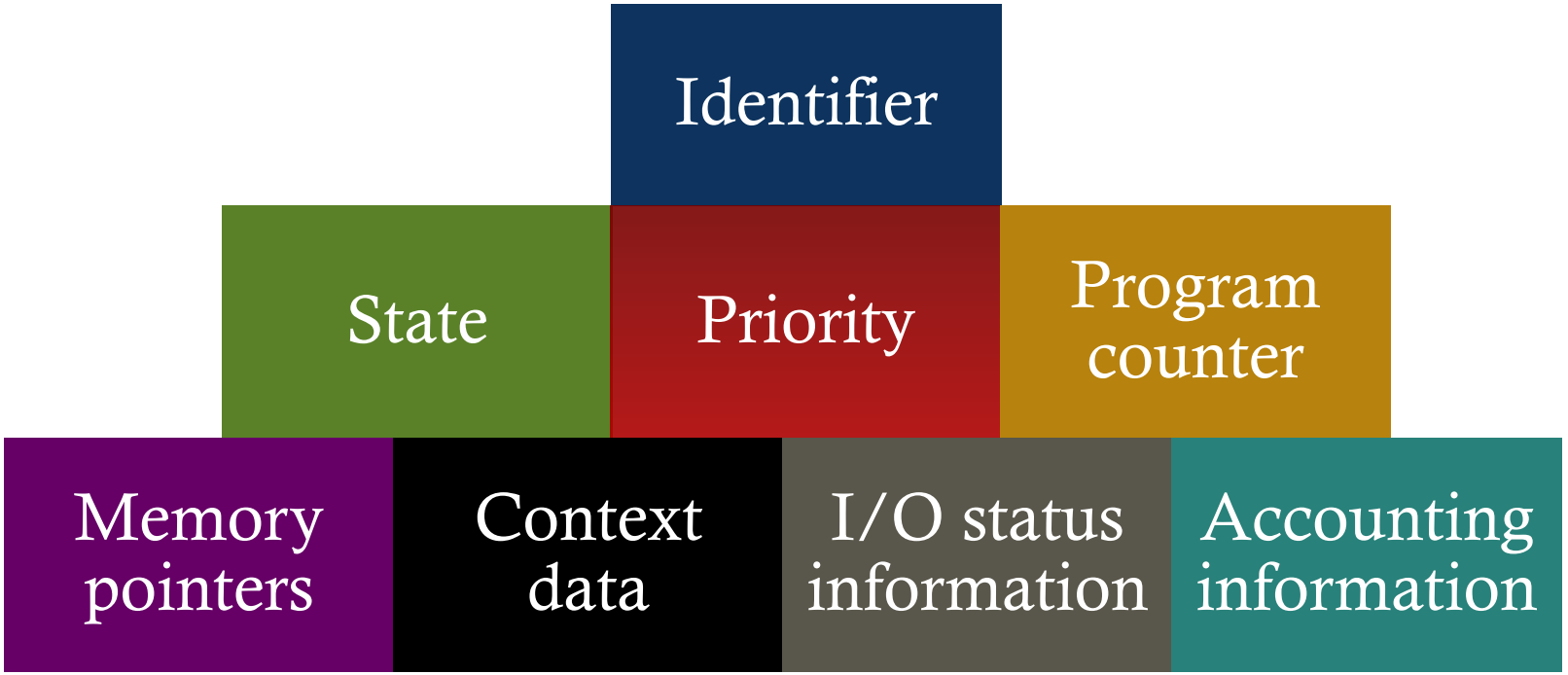
Process Elements

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

Process Elements

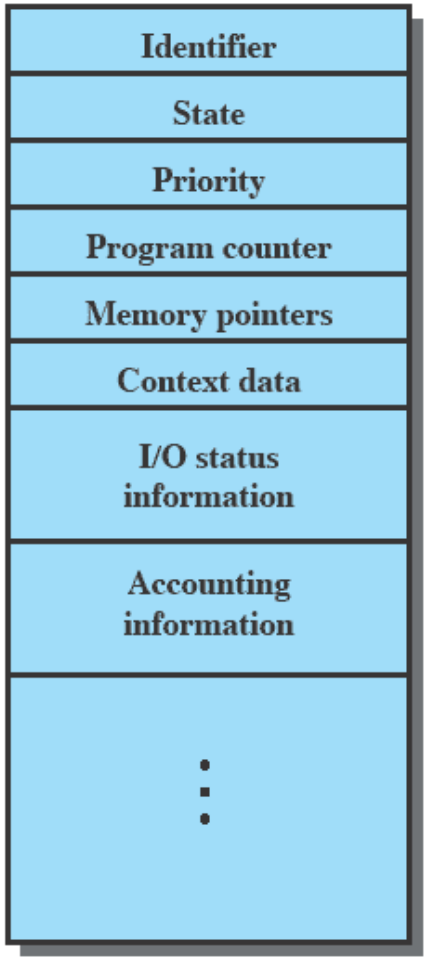
- While the process is running it has a number of elements including
  - Identifier
  - State
  - Priority
  - Program counter
  - Memory pointers
  - Context data
  - I/O status information
  - Accounting information

Process Elements (2)



Process Control Block

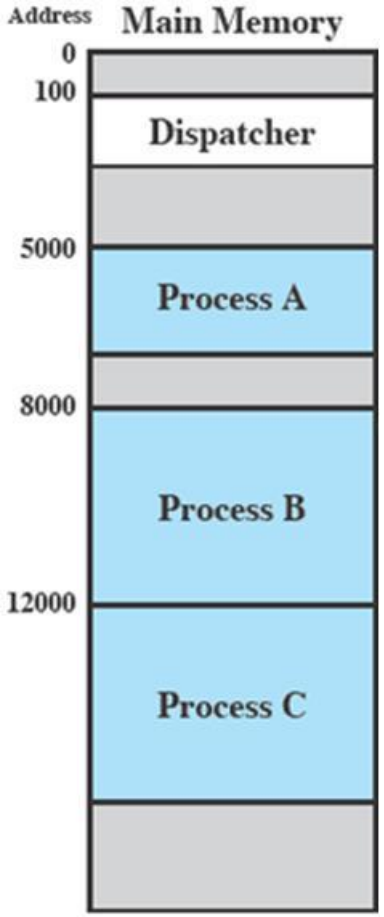
- Contains the process elements
- Created and manage by the operating system
- Allows support for multiple processes



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***
- ***Dispatcher*** is a small program which switches the processor from one process to another

Process Execution



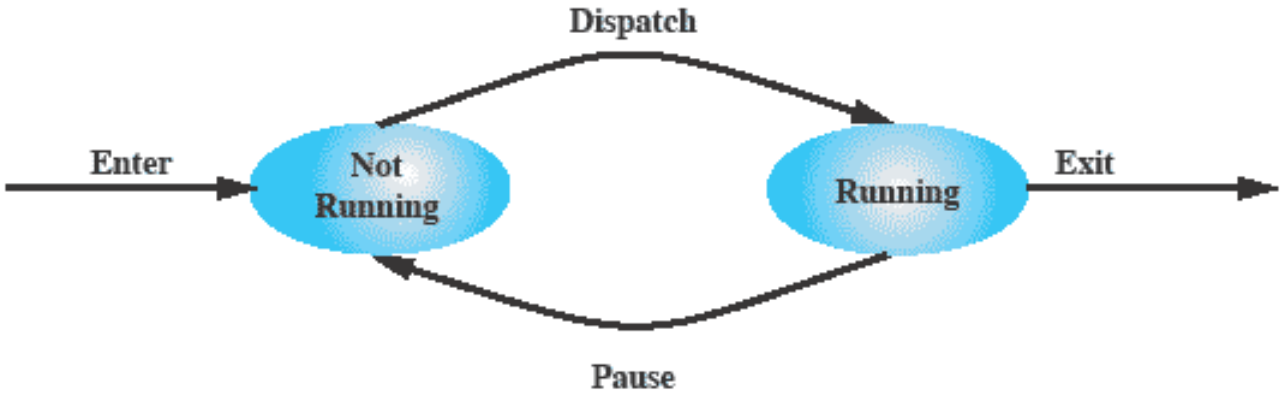
- Consider three processes being executed
- All are in memory (plus the dispatcher)
- Lets ignore virtual memory for this.

Roadmap

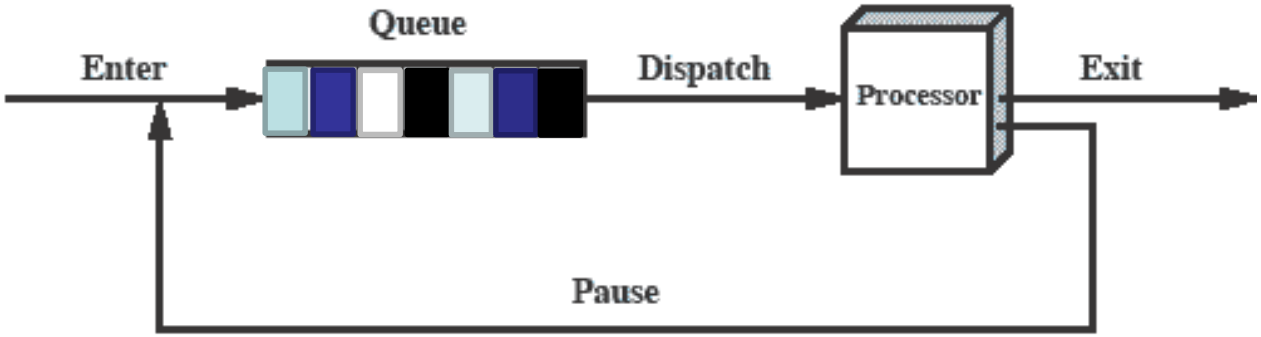
- How are processes represented and controlled by the OS.
- **Process states which characterize the behaviour of processes.**
- ***Data structures*** used to manage processes.
- Ways in which the OS uses these data structures to control process execution.
- Discuss process management in UNIX SVR4.

Two-State Process Model

- Process may be in one of two states
  - Running
  - Not-running



Queuing Diagram



Process Birth and Death

Creation	Termination
New batch job	Normal Completion
Interactive Login	Memory unavailable
Created by OS to provide a service	Protection error
Spawned by existing process	Operator or OS Intervention

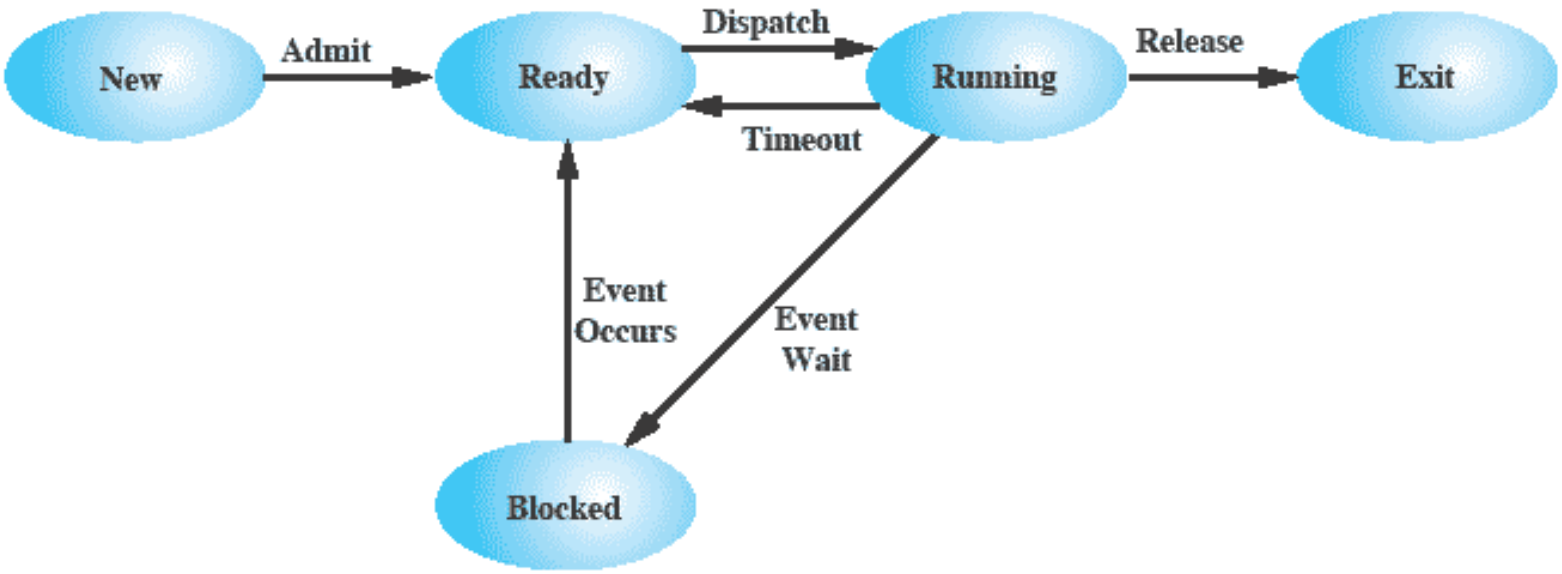
Process Creation

- The OS builds a data structure to manage the process
- Traditionally, the OS created all processes
  - But it can be useful to let a running process create another
- This action is called ***process spawning***
  - ***Parent Process*** is the original, creating, process
  - ***Child Process*** is the new process

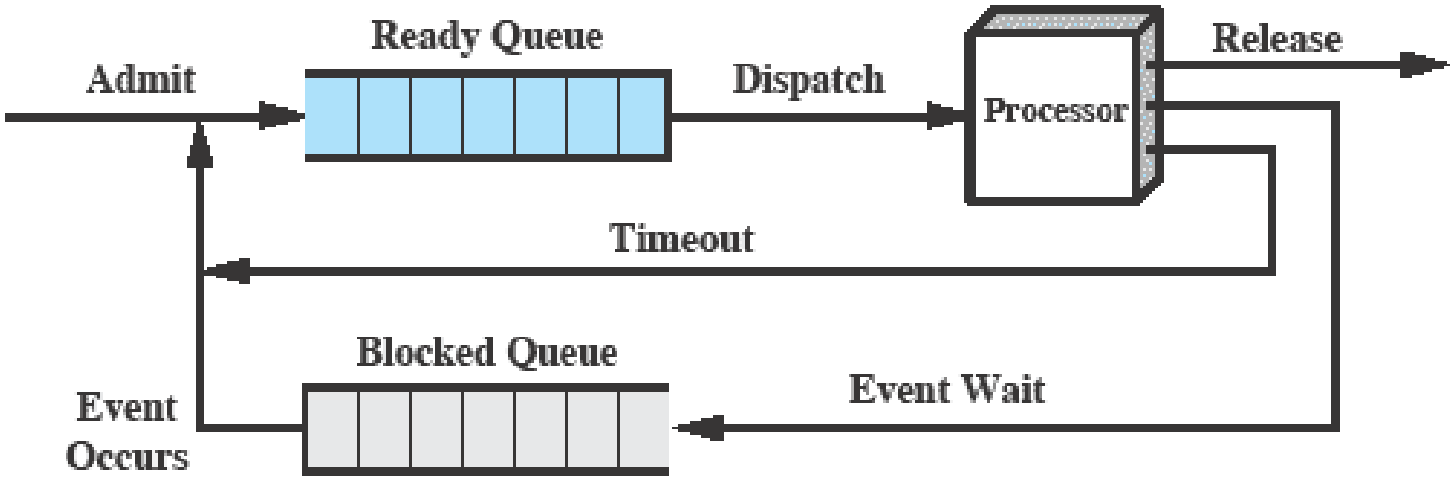
Process Termination

- There must be some way that a process can indicate completion.
- This indication may be:
  - A HALT instruction generating an interrupt alert to the OS.
  - A user action (e.g. log off, quitting an application)
  - A fault or error
  - Parent process terminating

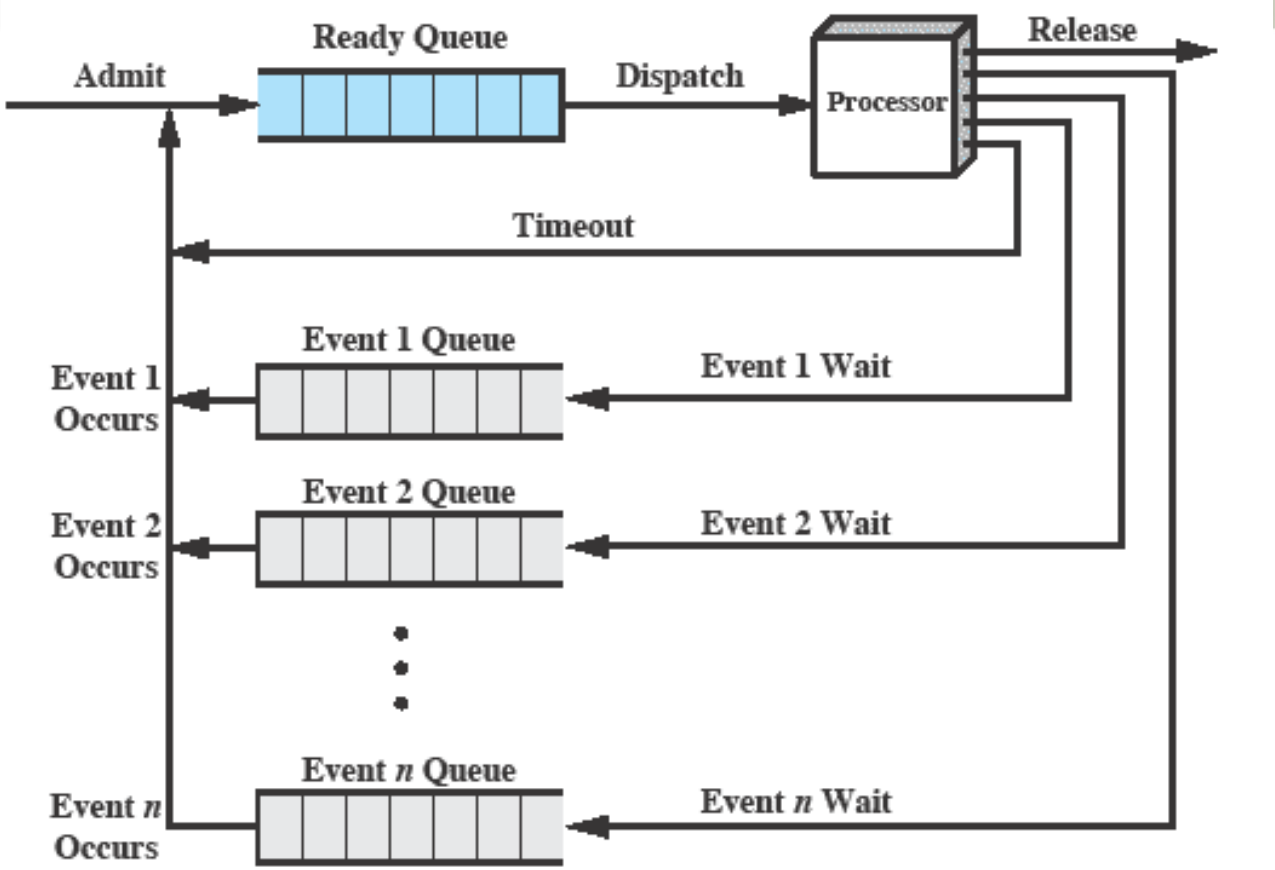
Five-State Process Model



Using Two Queues



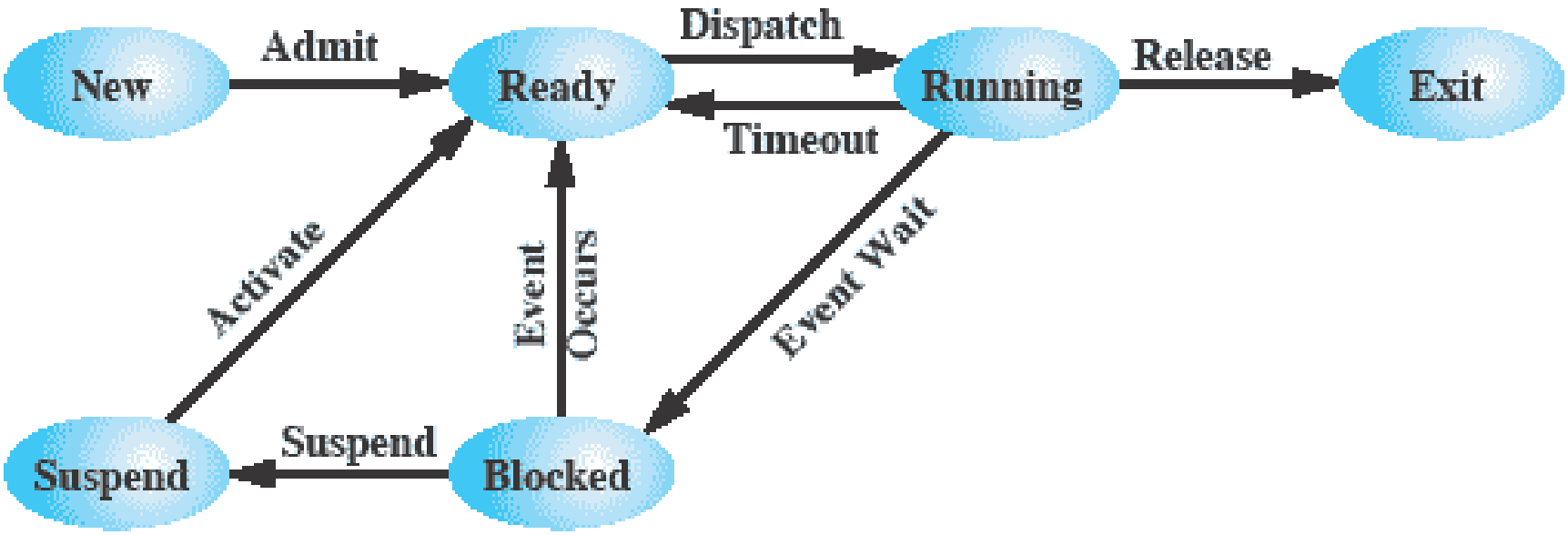
Multiple Blocked Queues



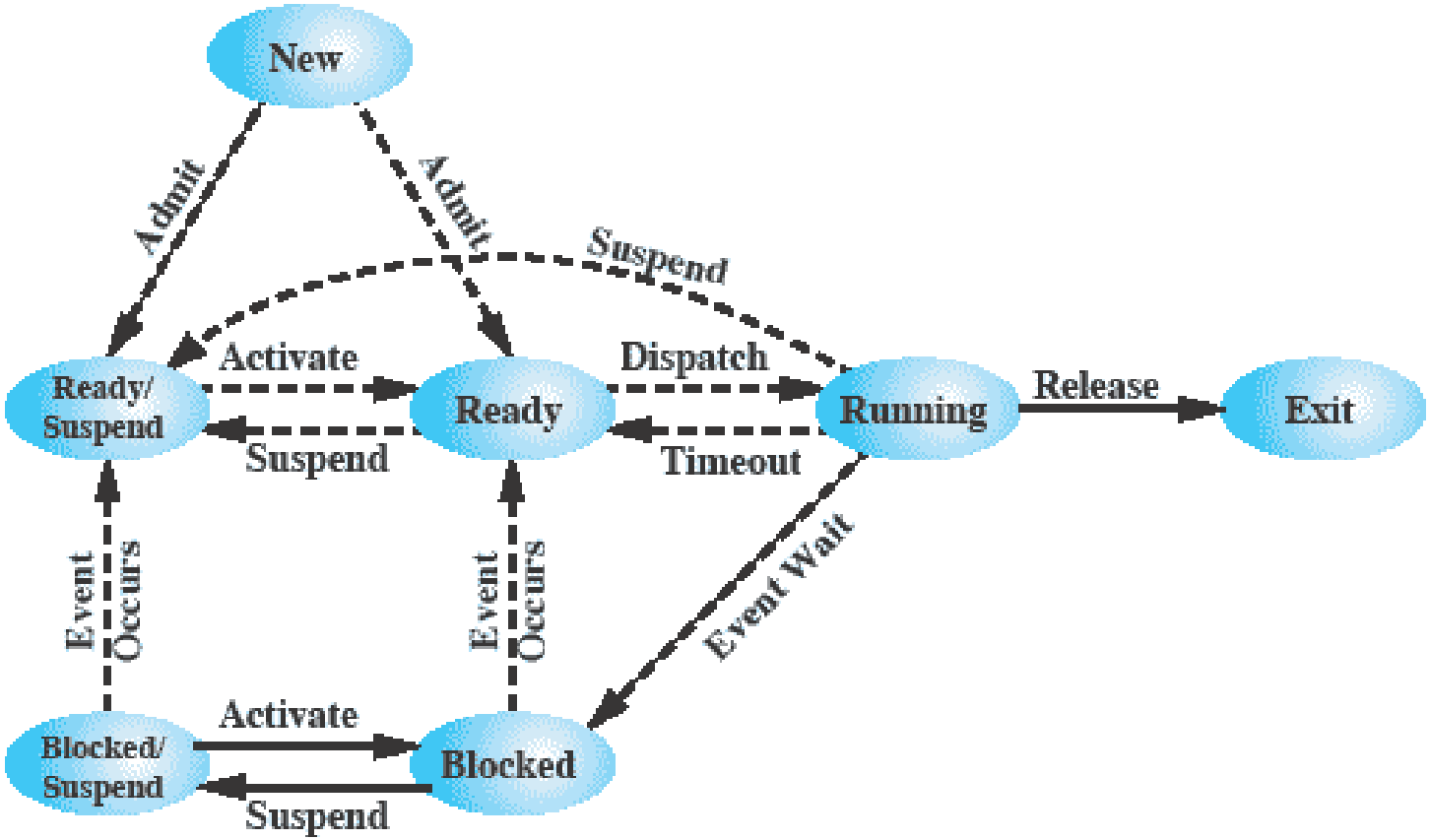
Suspended Processes

- Processor is faster than I/O so all processes could be waiting for I/O
  - Swap these processes to disk to free up more memory and use processor on more processes
- Blocked state becomes ***suspend*** state when swapped to disk
- Two new states
  - Blocked/Suspend
  - Ready/Suspend

One Suspend State



Two Suspend States



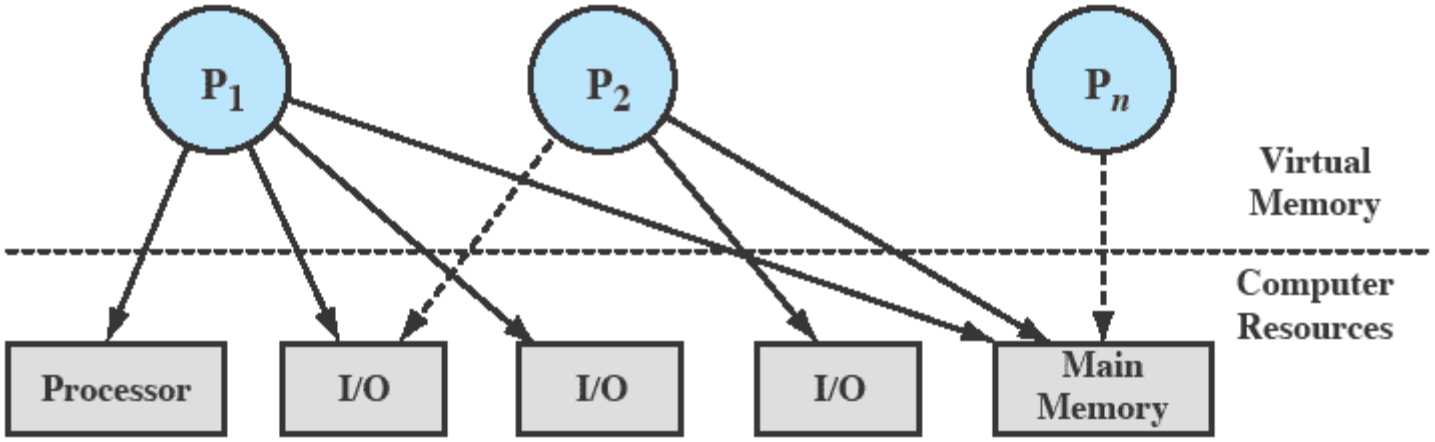
Reason for Process Suspension

Reason	Comment
Swapping	The OS needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS Reason	OS suspects process of causing a problem.
Interactive User Request	e.g. debugging or in connection with the use of a resource.
Timing	A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time.
Parent Process Request	A parent process may wish to suspend execution of a descendent to examine or modify the suspended process, or to coordinate the activity of various descendants.

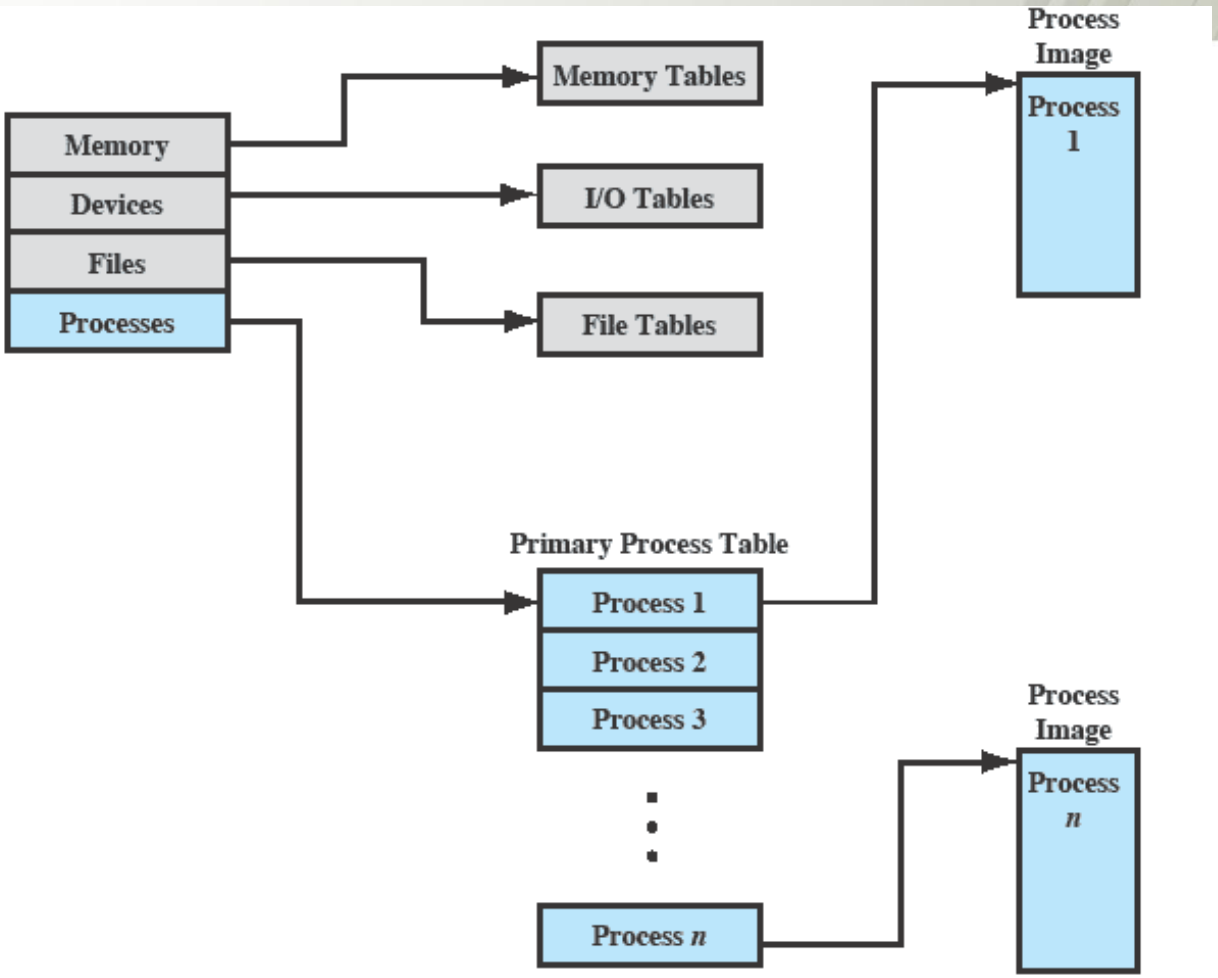
Roadmap

- How are processes represented and controlled by the OS.
- ***Process states*** which characterize the behaviour of processes.
- **Data structures used to manage processes.**
- Ways in which the OS uses these data structures to control process execution.
- Discuss process management in UNIX SVR4.

Processes and Resources



OS Control Tables



Memory Tables

- Memory tables are used to keep track of both main and secondary memory.
- Must include this information:
  - Allocation of main memory to processes
  - Allocation of secondary memory to processes
  - Protection attributes for access to shared memory regions
  - Information needed to manage virtual memory

I/O Tables

- Used by the OS to manage the I/O devices and channels of the computer.
- The OS needs to know
  - Whether the I/O device is available or assigned
  - The status of I/O operation
  - The location in main memory being used as the source or destination of the I/O transfer

File Tables

- These tables provide information about:
  - Existence of files
  - Location on secondary memory
  - Current Status
  - other attributes.
- Sometimes this information is maintained by a file management system

Process Tables

- To manage processes the OS needs to know details of the processes
  - Current state
  - Process ID
  - Location in memory
  - etc
- Process control block
  - ***Process image*** is the collection of program. Data, stack, and attributes

Process Attributes

- We can group the process control block information into three general categories:
  - Process identification
  - Processor state information
  - Process control information

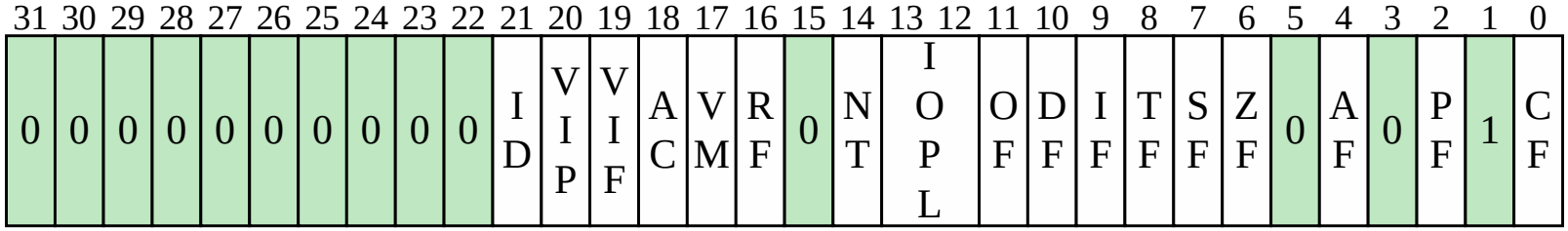
Process Identification

- Each process is assigned a unique numeric identifier.
- Many of the other tables controlled by the OS may use process identifiers to cross-reference process tables

Processor State Information

- This consists of the contents of processor registers.
  - User-visible registers
  - Control and status registers
  - Stack pointers
- Program status word (PSW)
  - contains status information
  - Example: the EFLAGS register on Pentium processors

x86 EFLAGS register

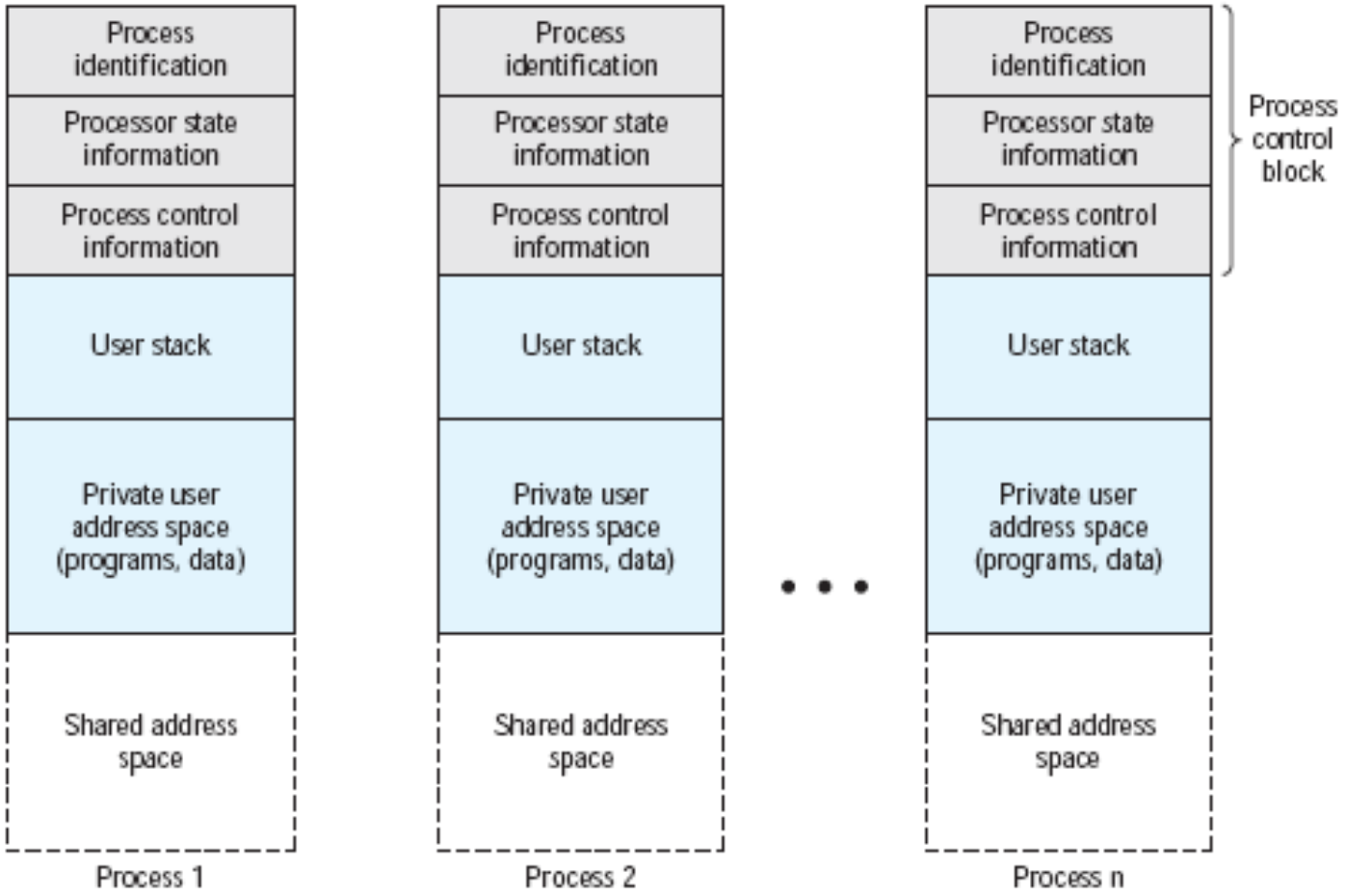


X ID = Identification flag
X VIP = Virtual interrupt pending
X VIF = Virtual interrupt flag
X AC = Alignment check
X VM = Virtual 8086 mode
X RF = Resume flag
X NT = Nested task flag
X IOPL = I/O privilege level
S OF = Overflow flag

C DF = Direction flag
X IF = Interrupt enable flag
X TF = Trap flag
S SF = Sign flag
S ZF = Zero flag
S AF = Auxiliary carry flag
S PF = Parity flag
S CF = Carry flag

S Indicates a Status Flag
C Indicates a Control Flag
X Indicates a System Flag
Shaded bits are reserved

Structure of Process Images in Virtual Memory



AGH

Role of the Process Control Block

- The most important data structure in an OS
  - It defines the state of the OS
- Process Control Block requires protection
  - A faulty routine could cause damage to the block destroying the OS's ability to manage the process
  - Any design change to the block could affect many modules of the OS

Roadmap

- How are processes represented and controlled by the OS.
- ***Process states*** which characterize the behaviour of processes.
- ***Data structures*** used to manage processes.
- **Ways in which the OS uses these data structures to control process execution.**
- Discuss process management in UNIX SVR4.

Modes of Execution

- Most processors support at least two modes of execution
- User mode
  - Less-privileged mode
  - User programs typically execute in this mode
- System mode
  - More-privileged mode
  - Kernel of the operating system

Process Creation

- OS decides to create a new process:
  - Assigns a unique process identifier
  - Allocates space for the process
  - Initializes process control block
  - Sets up appropriate linkages
  - Creates or expand other data structures

Switching Processes

- Several design issues are raised regarding process switching
  - What events trigger a process switch?
  - We must distinguish between mode switching and process switching.
  - What must the OS do to the various data structures under its control to achieve a process switch?

When to switch processes

A process switch may occur any time that the OS has gained control from the currently running process. Possible events giving OS control are:

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

Change of Process State …

- The steps in a process switch are:
  1. Save context of processor including program counter and other registers
  2. Update the process control block of the process that is currently in the Running state
  3. Move process control block to appropriate queue – ready; blocked; ready/suspend

Change of Process State cont…

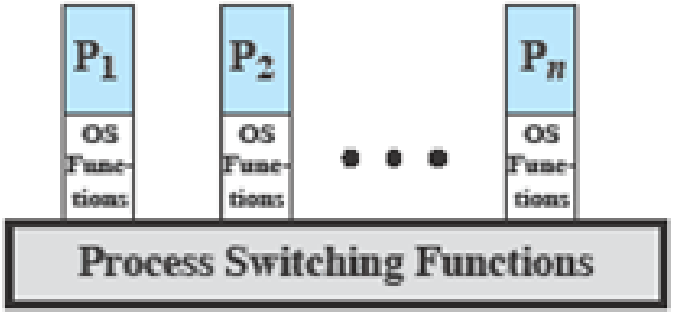
4. Select another process for execution
5. Update the process control block of the process selected
6. Update memory-management data structures
7. Restore context of the selected process

Roadmap

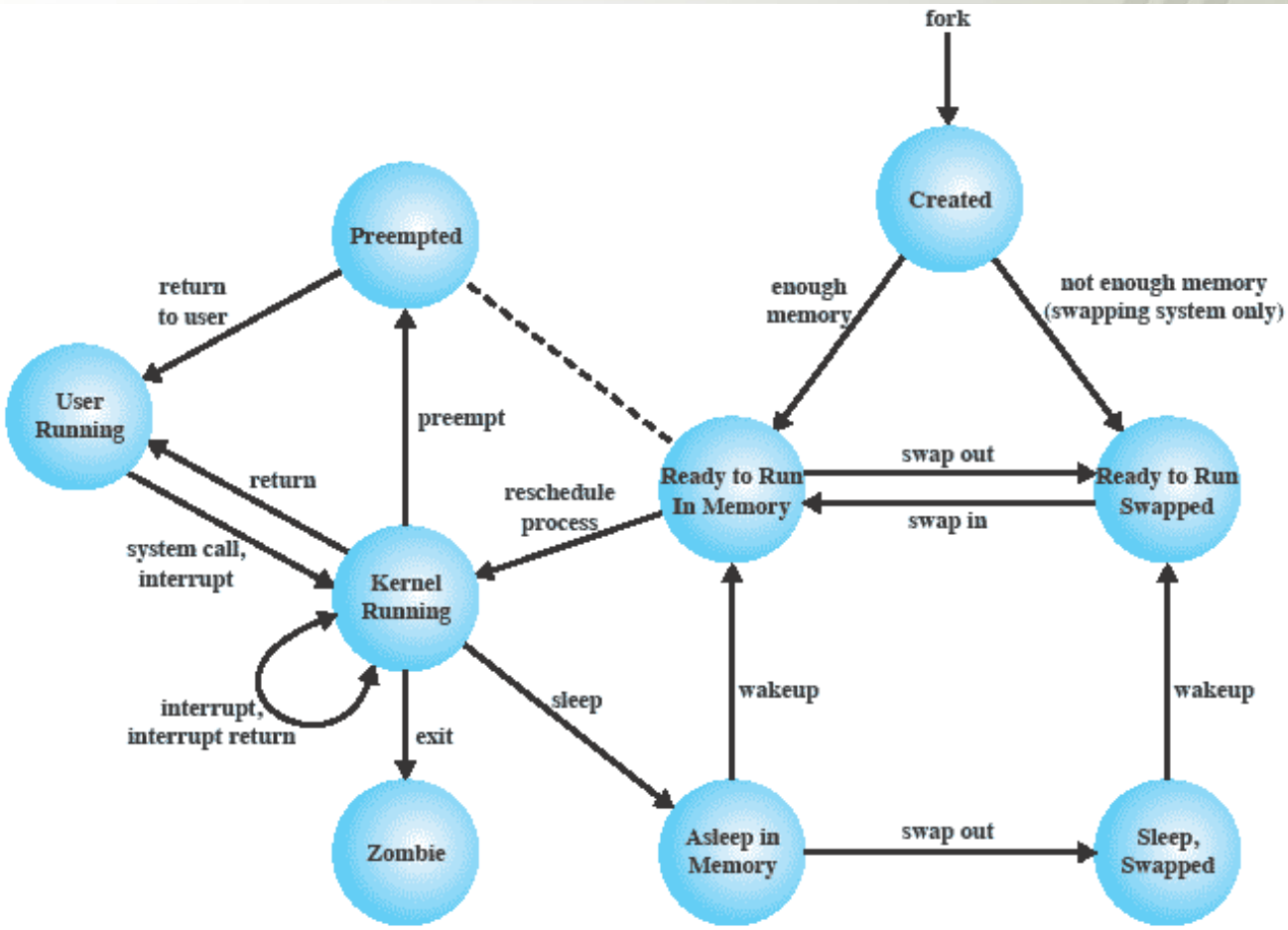
- How are processes represented and controlled by the OS.
- ***Process states*** which characterize the behaviour of processes.
- ***Data structures*** used to manage processes.
- Ways in which the OS uses these data structures to control process execution.
- Discuss process management in UNIX SVR4.

Unix SVR4
System V Release 4

- System Processes - Kernel mode only
- User Processes
  - User mode to execute user programs and utilities
  - Kernel mode to execute instructions that belong to the kernel.



UNIX Process State Transition Diagram



UNIX Process States

User Running	Executing in user mode.
Kernel Running	Executing in kernel mode.
Ready to Run, in Memory	Ready to run as soon as the kernel schedules it.
Asleep in Memory	Unable to execute until an event occurs; process is in main memory (a blocked state).
Ready to Run, Swapped	Process is ready to run, but the swapper must swap the process into main memory before the kernel can schedule it to execute.
Sleeping, Swapped	The process is awaiting an event and has been swapped to secondary storage (a blocked state).
Preempted	Process is returning from kernel to user mode, but the kernel preempts it and does a process switch to schedule another process.
Created	Process is newly created and not yet ready to run.
Zombie	Process no longer exists, but it leaves a record for its parent process to collect.

A Unix Process

- A process in UNIX is a set of data structures that provide the OS with all of the information necessary to manage and dispatch processes.
- Elements of the proces are organized into three parts:
  - user-level context,
  - register context,
  - system-level context.

Unix Process Image

User-Level Context	
Process text	Executable machine instructions of the program
Process data	Data accessible by the program of this process
User stack	Contains the arguments, local variables, and pointers for functions executing in user mode
Shared memory	Memory shared with other processes, used for interprocess communication
Register Context	
Program counter	Address of next instruction to be executed; may be in kernel or user memory space of this process
Processor status register	Contains the hardware status at the time of preemption; contents and format are hardware dependent
Stack pointer	Points to the top of the kernel or user stack, depending on the mode of operation at the time or preemption
General-purpose registers	Hardware dependent
System-Level Context	
Process table entry	Defines state of a process; this information is always accessible to the operating system
U (user) area	Process control information that needs to be accessed only in the context of the process
Per process region table	Defines the mapping from virtual to physical addresses; also contains a permission field that indicates the type of access allowed the process: read-only, read-write, or read-execute
Kernel stack	Contains the stack frame of kernel procedures as the process executes in kernel mode

Process Creation

- Process creation is by means of the kernel system call,fork( ).
- This causes the OS, in Kernel Mode, to:
  1. Allocate a slot in the process table for the new process.
  2. Assign a unique process ID to the child process.
  3. Copy of process image of the parent, with the exception of any shared memory.

Process Creation cont…

4. Increment the counters for any files owned by the parent, to reflect that an additional process now also owns those files.
5. Assign the child process to the Ready to Run state.
6. Returns the ID number of the child to the parent process, and a 0 value to the child process.

After Creation

- After creating the process the Kernel can do one of the following, as part of the dispatcher routine:
  - Stay in the parent process.
  - Transfer control to the child process
  - Transfer control to another process.

Bibliography

- William Stallings, „Operating Systems. Internals and Design Principles”. Ninth Edition, Pearson Prentice Hall, 2017
- Dave Bremer, Otago Polytechnic, N.Z., Prentice Hall