Process Description and Control

Summary of Earlier Concepts

- A computer platform consists of a collection of hardware resources
- Computer applications are developed to perform some task
- It is inefficient for applications to be written directly for a given hardware platform
- The OS was developed to provide a convenient, feature-rich, secure, and consistent interface for applications to use
- We can think of the OS as providing a uniform, abstract representation of resources that can be requested and accessed by applications

Management of Application Execution

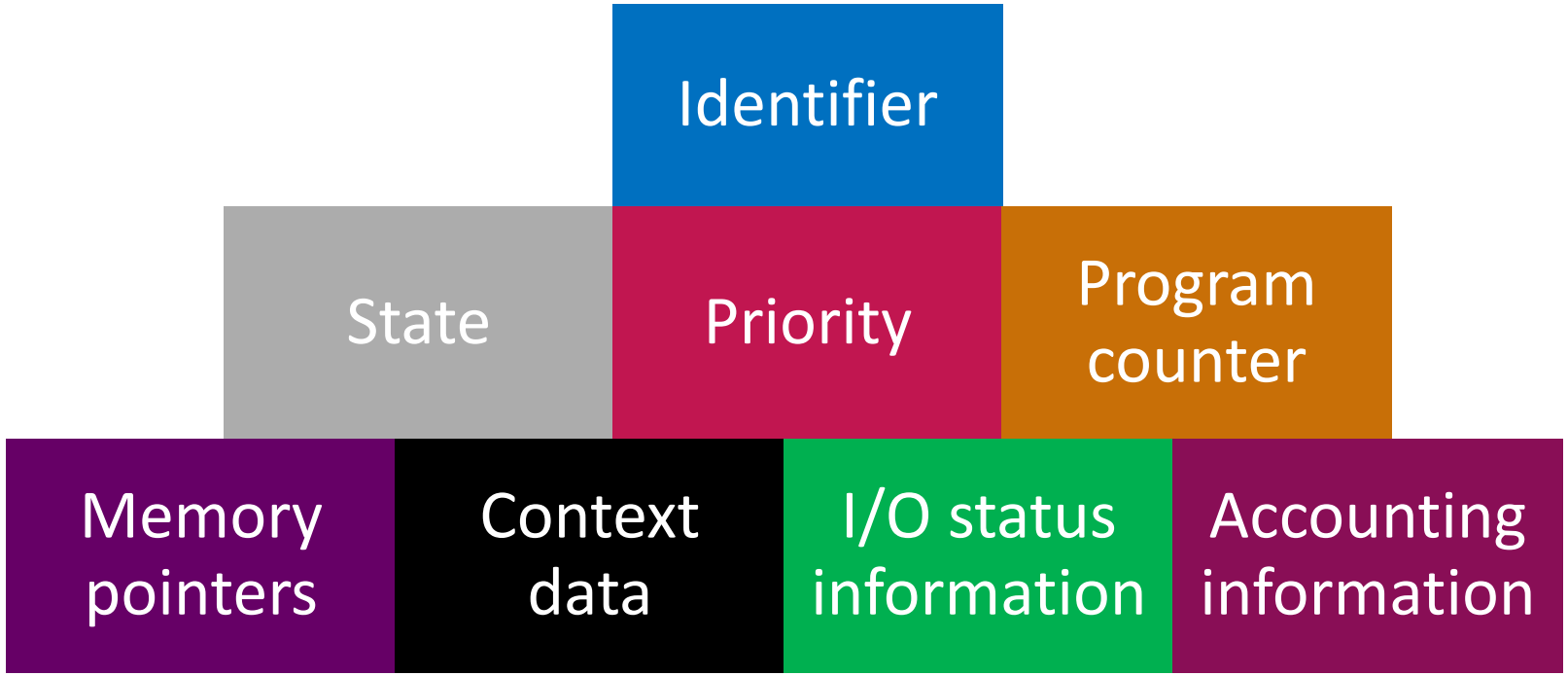
- Resources are made available to multiple applications
- The processor is switched among multiple applications so all will appear to be progressing
- The processor and I/O devices can be used efficiently

Process Elements

- Two essential elements of a process are:
- Program code
  - which may be shared with other processes that are executing the same program
- A set of data associated with that code
  - when the processor begins to execute the program code, we refer to this executing entity as a process

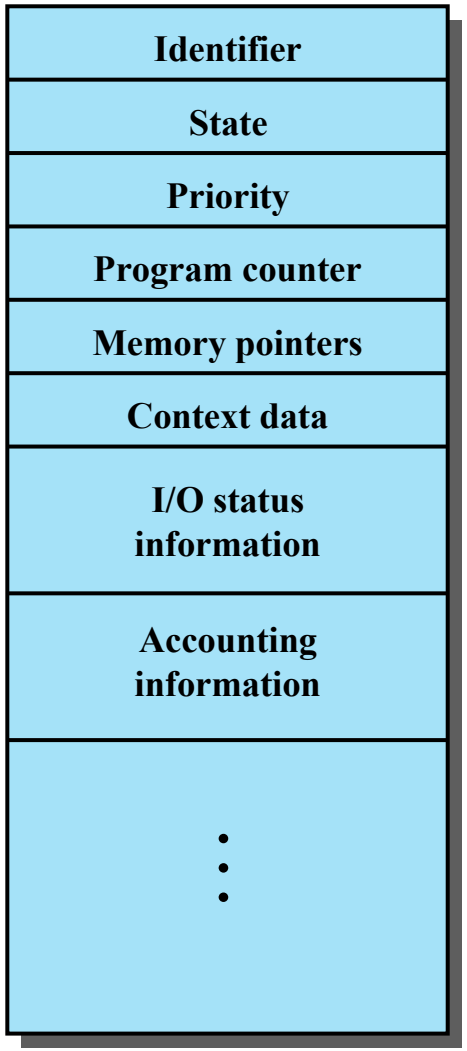
Process Elements

- While the program is executing, this process can be uniquely characterized by a number of elements, including:

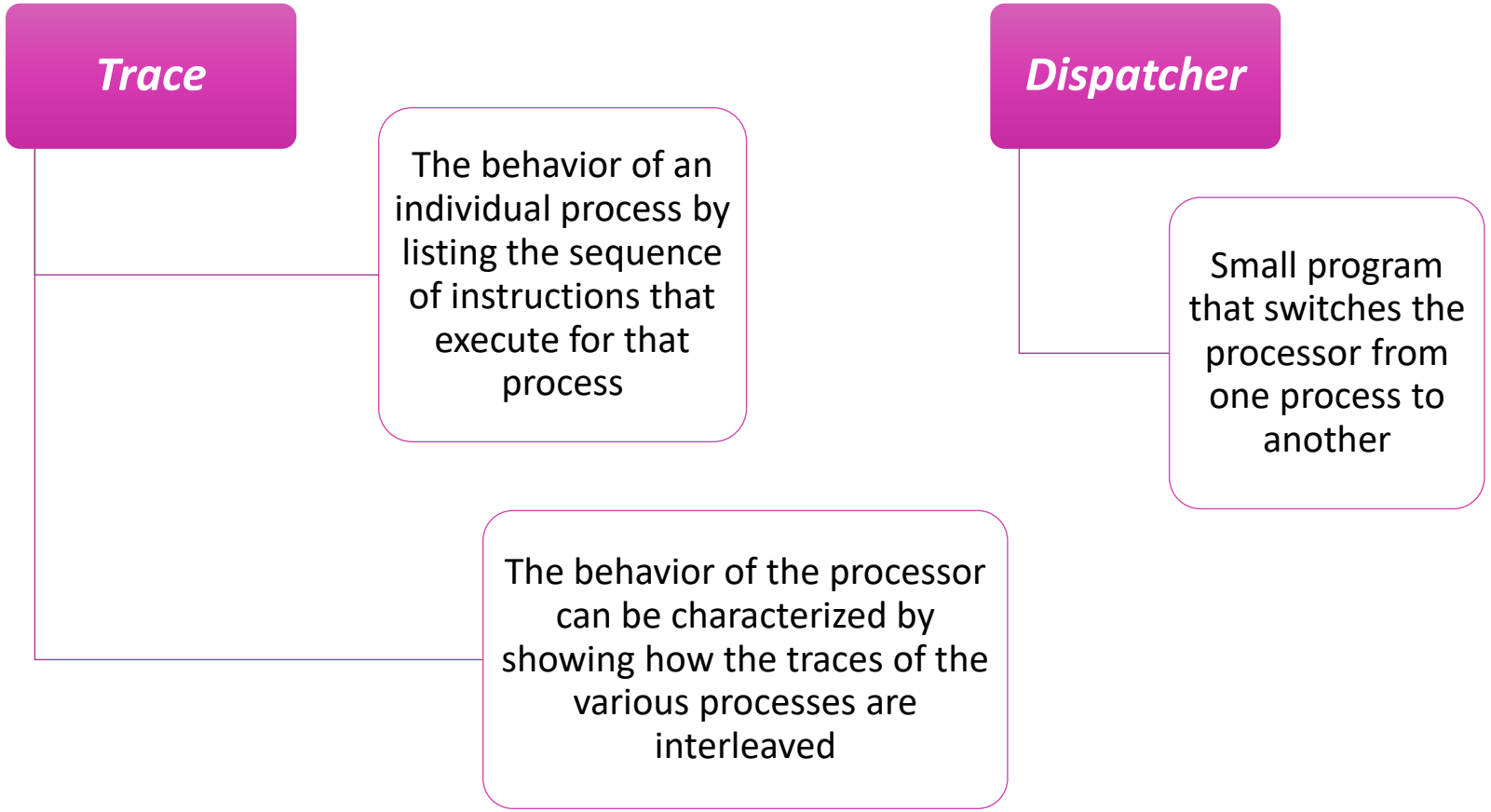


Process Control Block

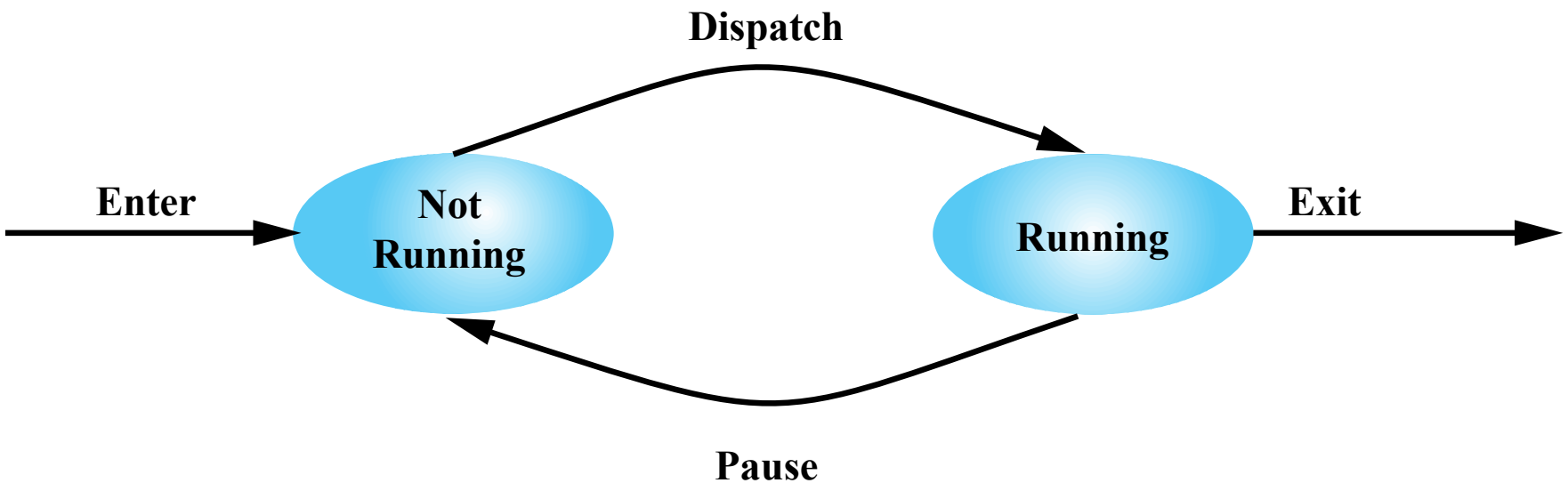
- Contains the process elements
- It is possible to interrupt a running process and later resume execution as if the interruption had not occurred
- Created and managed by the operating system
- Key tool that allows support for multiple processes



Process States



Two-State Process Model



Reasons for Process Creation

New batch job	The OS is provided with a batch job control stream, usually on tape or disk. When the OS is prepared to take on new work, it will read the next sequence of job control commands.
Interactive log-on	A user at a terminal logs on to the system.
Created by OS to provide a service	The OS can create a process to perform a function on behalf of a user program, without the user having to wait (e.g., a process to control printing).
Spawned by existing process	For purposes of modularity or to exploit parallelism, a user program can dictate the creation of a number of processes.

Process Creation

Process spawning	*Parent process*	*Child process*
• When the OS creates a process at the explicit request of another process	• Is the original, creating, process	• Is the new process

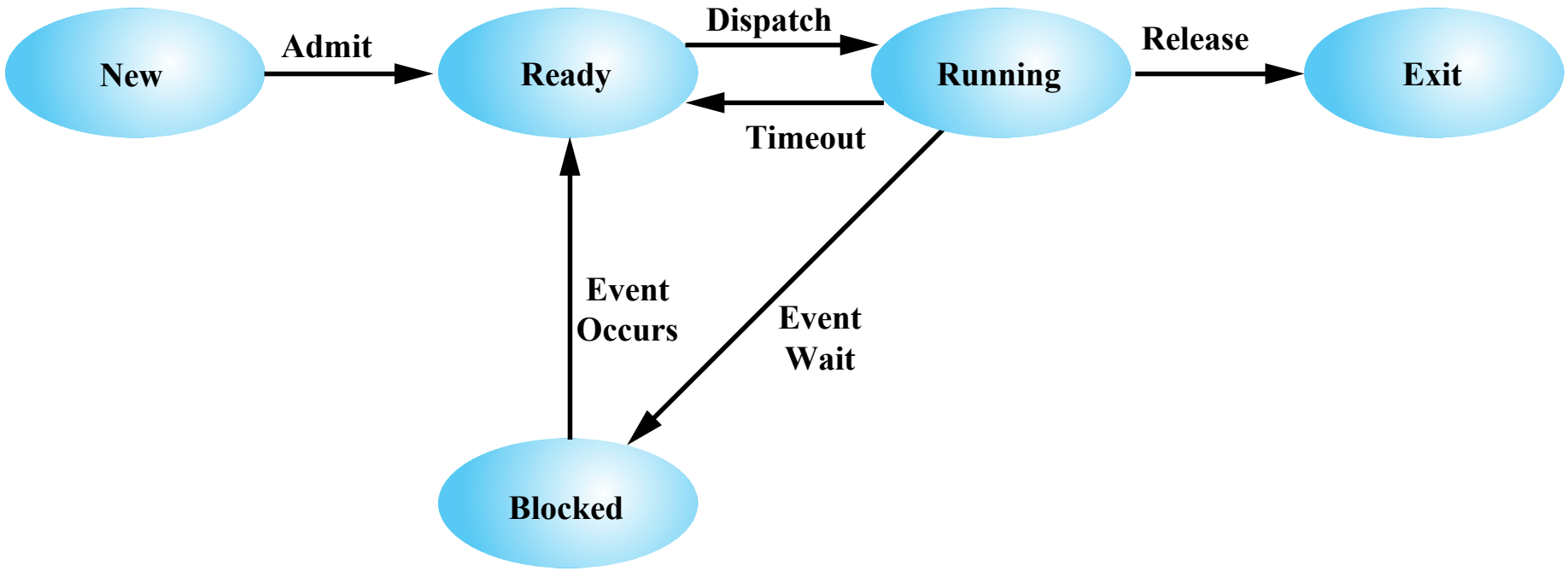
Process Termination

- There must be a means for a process to indicate its completion
- A batch job should include a HALT instruction or an explicit OS service call for termination
- For an interactive application, the action of the user will indicate when the process is completed (e.g. log off, quitting an application)

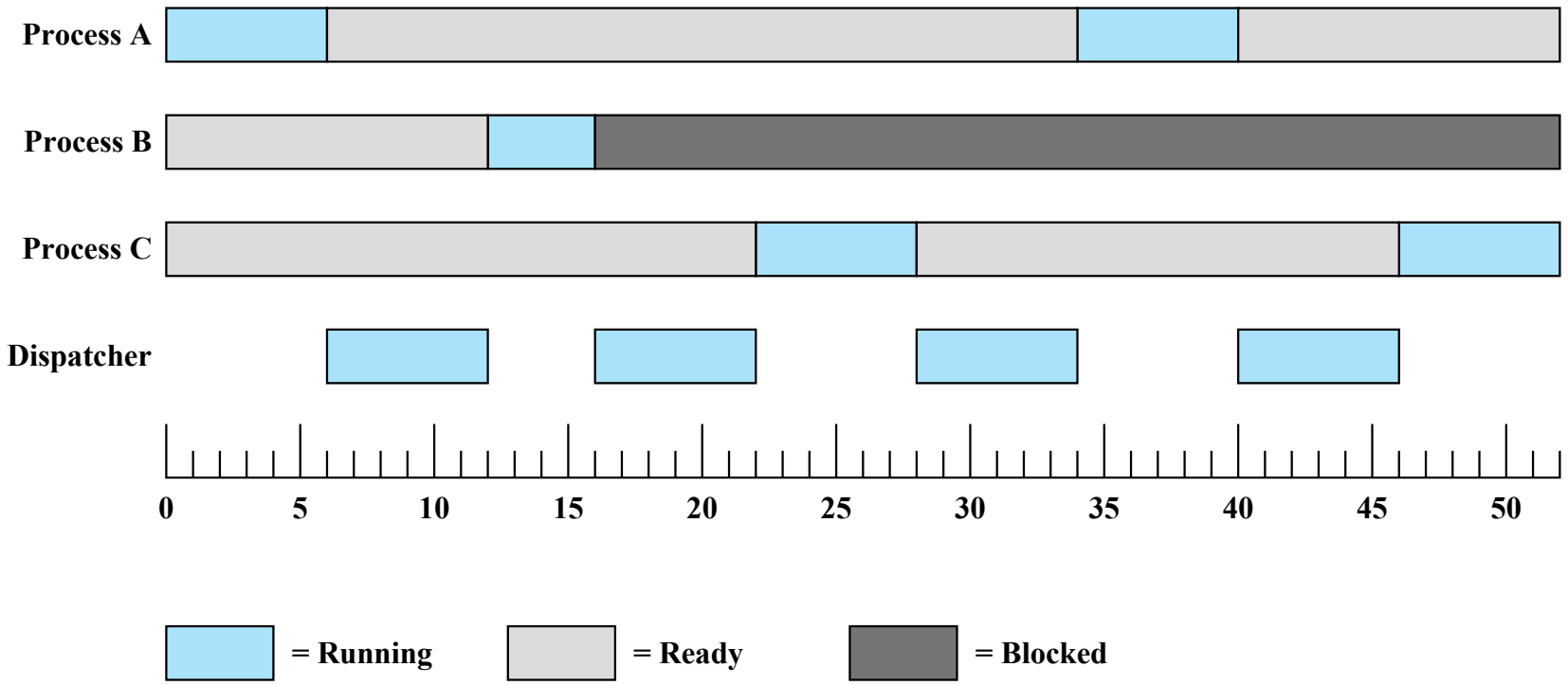
Reasons for Process Termination

Normal completion	The process executes an OS service call to indicate that it has completed running.
Time limit exceeded	The process has run longer than the specified total time limit. There are a number of possibilities for the type of time that is measured.
Memory unavailable	The process requires more memory than the system can provide.
Bounds violation	The process tries to access a memory location that it is not allowed to access.
Protection error	The process attempts to use a resource such as a file that it is not allowed to use, or it tries to use it in an improper fashion, such as writing to a read-only file.
Arithmetic error	The process tries a prohibited computation (such as division by zero) or tries to store numbers larger than the hardware can accommodate.
Time overrun	The process has waited longer than a specified maximum for a certain event to occur.
I/O failure	An error occurs during input or output, such as inability to find a file, failure to read or write after a specified maximum number of tries (when, for exam- ple, a defective area is encountered on a tape), or invalid operation (such as reading from the line printer).
Invalid instruction	The process attempts to execute a nonexistent instruction (often a result of branching into a data area and attempting to execute the data).
Privileged instruction	The process attempts to use an instruction reserved for the operating system.
Data misuse	A piece of data is of the wrong type or is not initialized.
Operator or OS intervention	For some reason, the operator or the operating system has terminated the pro- cess (e.g., if a deadlock exists).
Parent termination	When a parent terminates, the operating system may automatically terminate all of the offspring of that parent.
Parent request	A parent process typically has the authority to terminate any of its offspring.

Five-State Process Model



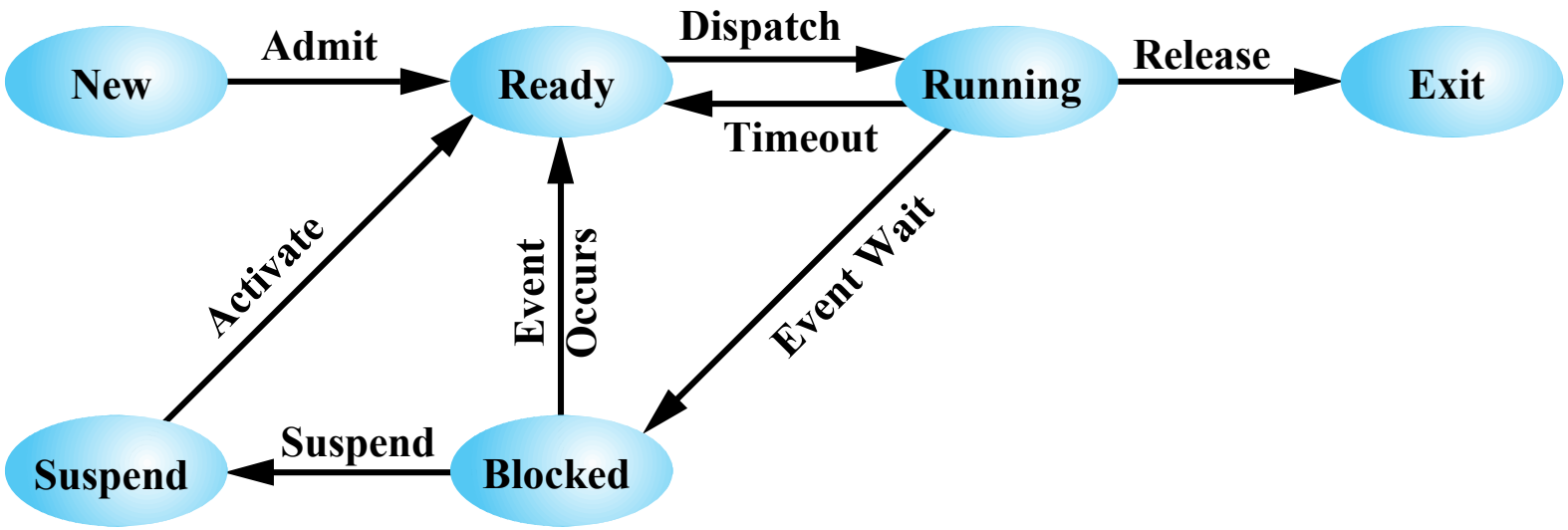
Process States for Trace



Suspended Processes

- Swapping
  - Involves moving part of all of a process from main memory to disk
  - When none of the processes in main memory is in the Ready state, the OS swaps one of the blocked processes out on to disk into a suspend queue
    - This is a queue of existing processes that have been temporarily kicked out of main memory, or suspended
    - The OS then brings in another process from the suspend queue or it honors a new-process request
    - Execution then continues with the newly arrived process
  - Swapping, however, is an I/O operation and therefore there is the potential for making the problem worse, not better. Because disk I/O is generally the fastest I/O on a system, swapping will usually enhance performance

Process State with Suspend State



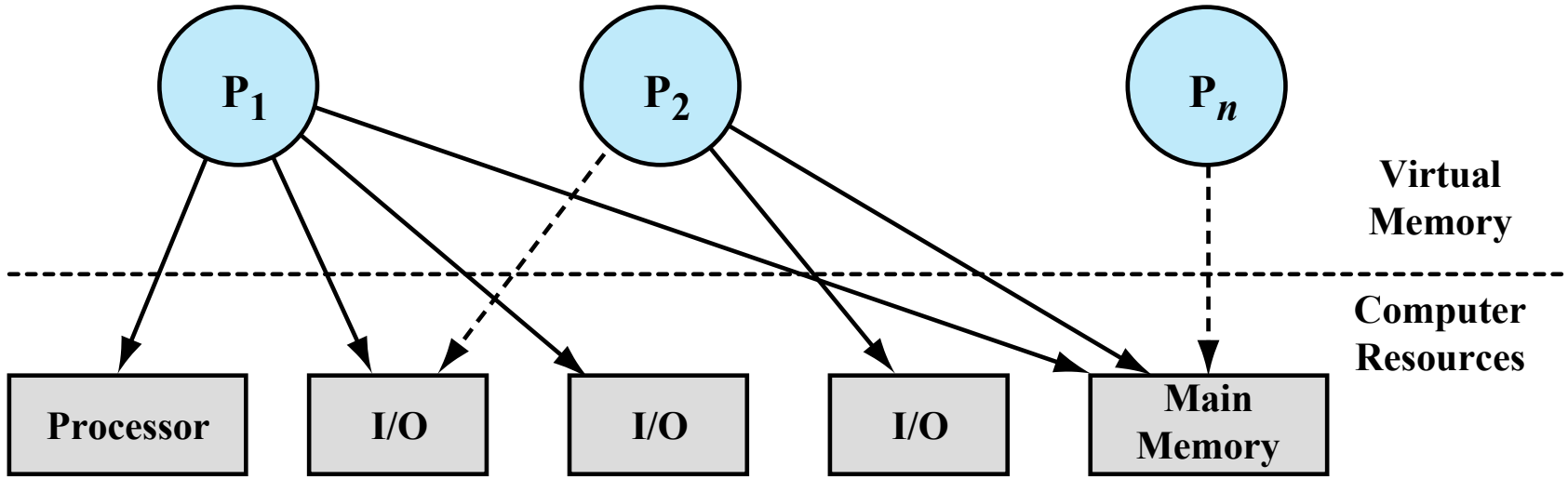
Characteristics of a Suspended Process

- The process is not immediately available for execution
- The process was placed in a suspended state by an agent: either itself, a parent process, or the OS, for the purpose of preventing its execution
- The process may or may not be waiting on an event
- The process may not be removed from this state until the agent explicitly orders the removal

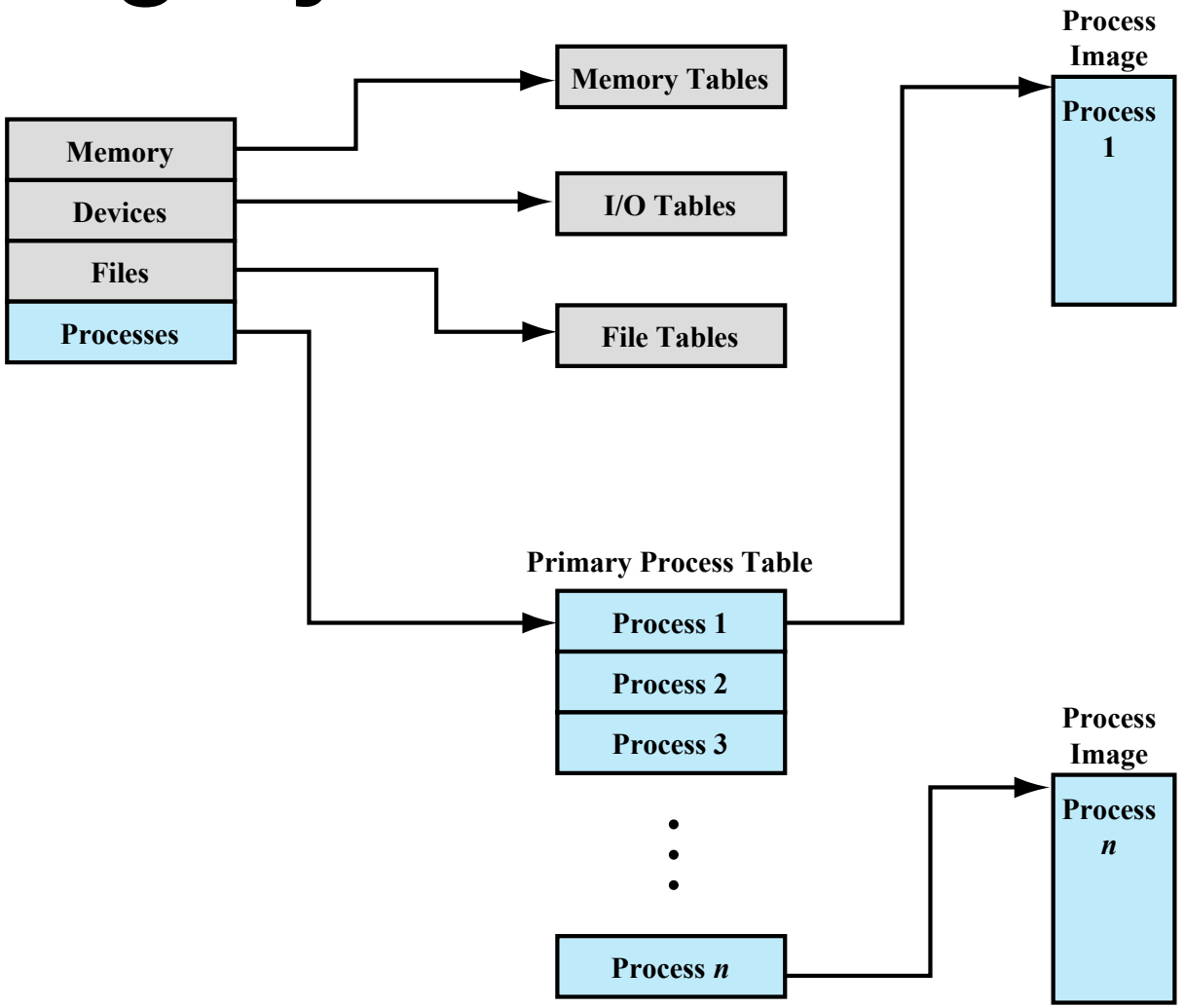
Reasons for Process Suspension

Swapping	The OS needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS reason	The OS may suspend a background or utility process or a process that is suspected of causing a problem.
Interactive user request	A user may wish to suspend execution of a program for purposes of 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 interval.
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.

Processes and Resources

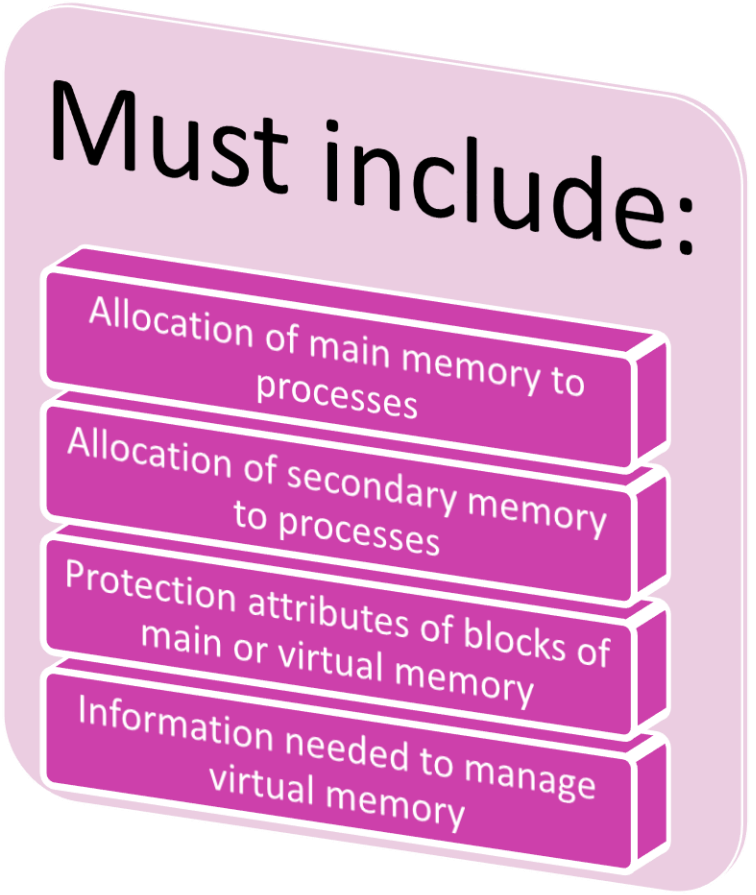


Operating System Control Tables



Memory Tables

- Used to keep track of both main (real) and secondary (virtual) memory
- Processes are maintained on secondary memory using some sort of virtual memory or simple swapping mechanism



I/O Tables

- Used by the OS to manage the I/O devices and channels of the computer system
- At any given time, an I/O device may be available or assigned to a particular process

If an I/O operation is in progress, the OS needs to know:

- The status of the I/O operation
- The location in main memory being used as the source or destination of the I/O transfer

File Tables

- Information may be maintained and used by a file management system
  - In which case the OS has little or no knowledge of files
  - In other operating systems, much of the detail of file management is managed by the OS itself

These tables provide information about:

- Existence of files
- Location on secondary memory
- Current status
- Other attributes

Process Tables

- Must be maintained to manage processes
- There must be some reference to memory, I/O, and files, directly or indirectly
- The tables themselves must be accessible by the OS and therefore are subject to memory management

Process Control Structures

To manage and control a process the OS must know:

- Where the process is located
- The attributes of the process that are necessary for its management

Process Control Structures

Process Location

- A process must include a program or set of programs to be executed
- A process will consist of at least sufficient memory to hold the programs and data of that process
- The execution of a program typically involves a stack that is used to keep track of procedure calls and parameter passing between procedures

Process Attributes

- Each process has associated with it a number of attributes that are used by the OS for process control
- The collection of program, data, stack, and attributes is referred to as the process image
- Process image location will depend on the memory management scheme being used

Typical Elements of a Process Image

- User Data
  - The modifiable part of the user space. May include program data, a user stack area, and programs that may be modified.
- User Program
  - The program to be executed.
- Stack
  - Each process has one or more last-in-first-out (LIFO) stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls.
- Process Control Block
  - Data needed by the OS to control the process.

Process Control Block

- Process identification
- Processor state information
- Process control information

Process Identification

- Numeric identifiers that may be stored with the process control block include
  - Identifier of this process.
  - Identifier of the process that created this process (parent process).
  - User identifier.

Processor State Information

- A **user-visible register** is one that may be referenced by means of the machine language that the processor executes while in user mode.
- **Control and Status Registers** are a variety of processor registers that are employed to control the operation of the processor.
  - **Program counter**
  - **Condition codes**
  - **Status information**
- **Stack Pointers** - Each process has one or more last-in-first-out (LIFO) system stacks associated with it.

Process Control Information

- **Scheduling and State Information** is information that is needed by the operating system to perform its scheduling function.
  - Process state
  - Priority
  - Scheduling-related information
  - Event
- **Data Structuring** - A process may be linked to other process in a queue, ring, or some other structure.
- **Interprocess Communication** - Various flags, signals, and messages may be associated with communication between two independent processes.
- **Process Privileges** - Processes are granted privileges in terms of the memory that may be accessed and the types of instructions that may be executed.
- **Memory Management** section may include pointers to segment and/or page tables that describe the virtual memory assigned to this process.
- **Resource Ownership and Utilization** - Resources controlled by the process may be indicated, such as opened files.

Process Identification

- Each process is assigned a unique numeric identifier
  - Otherwise there must be a mapping that allows the OS to locate the appropriate tables based on the process identifier
- Many of the tables controlled by the OS may use process identifiers to cross-reference process tables
- Memory tables may be organized to provide a map of main memory with an indication of which process is assigned to each region
  - Similar references will appear in I/O and file tables
- When processes communicate with one another, the process identifier informs the OS of the destination of a particular communication
- When processes are allowed to create other processes, identifiers indicate the parent and descendents of each process

Processor State Information

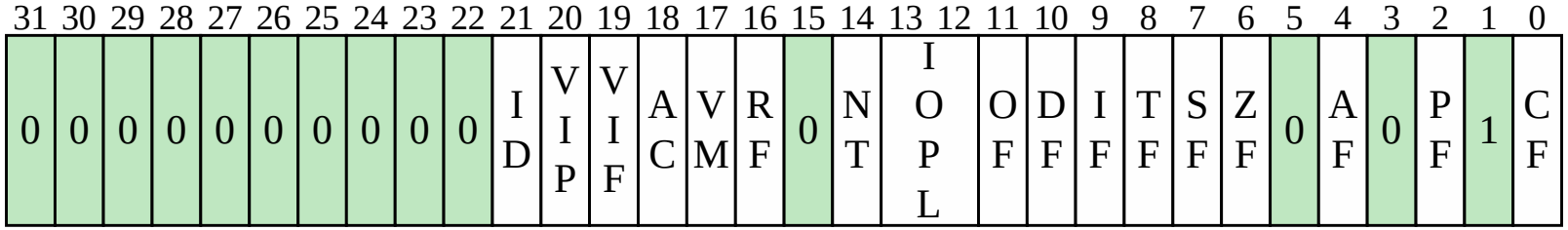
Consists of the contents of processor registers

- User-visible registers
- Control and status registers
- Stack pointers

Program status word (PSW)

- Contains condition codes plus other status information
- EFLAGS register is an example of a PSW used by any OS running on an x86 processor

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

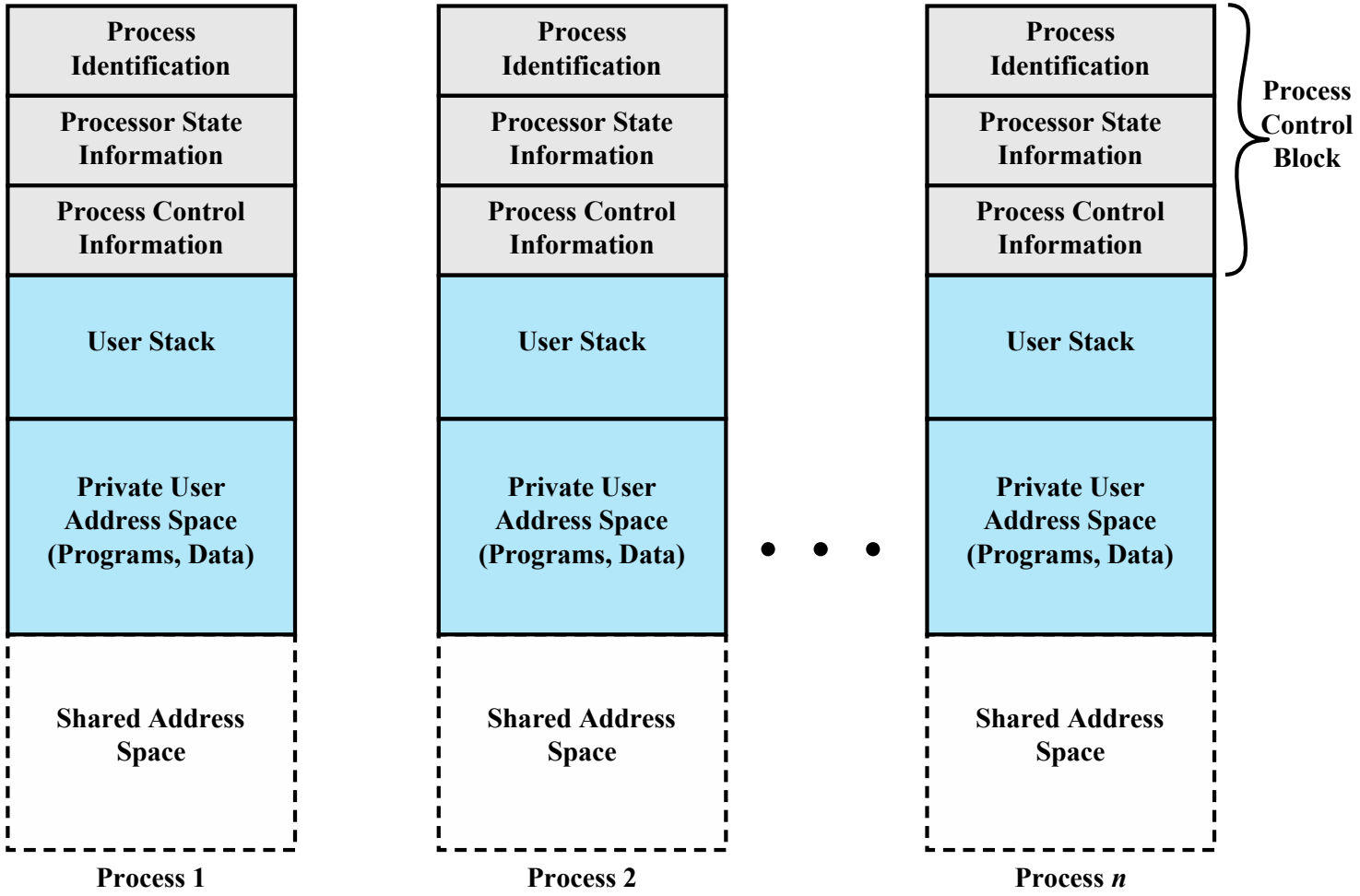
C DF = Direction flag
X IF = Interrupt enable flag
X TF = Trap flag
S SF = Sign flag
S ZF = Zero flag
S AF = Auxiliary carry flag
S PF = Parity flag
S CF = Carry flag

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

Process Control Information

- The additional information needed by the OS to control and coordinate the various active processes

User Processes in Virual Memory



Role of the Process Control Block

- The most important data structure in an OS
  - Contains all of the information about a process that is needed by the OS
  - Blocks are read and/or modified by virtually every module in the OS
  - Defines the state of the OS
- Difficulty is not access, but protection
  - A bug in a single routine could damage process control blocks, which could destroy the system's ability to manage the affected processes
  - A design change in the structure or semantics of the process control block could affect a number of modules in the OS

Modes of Execution

User Mode

- Less-privileged mode
- User programs typically execute in this mode

System Mode

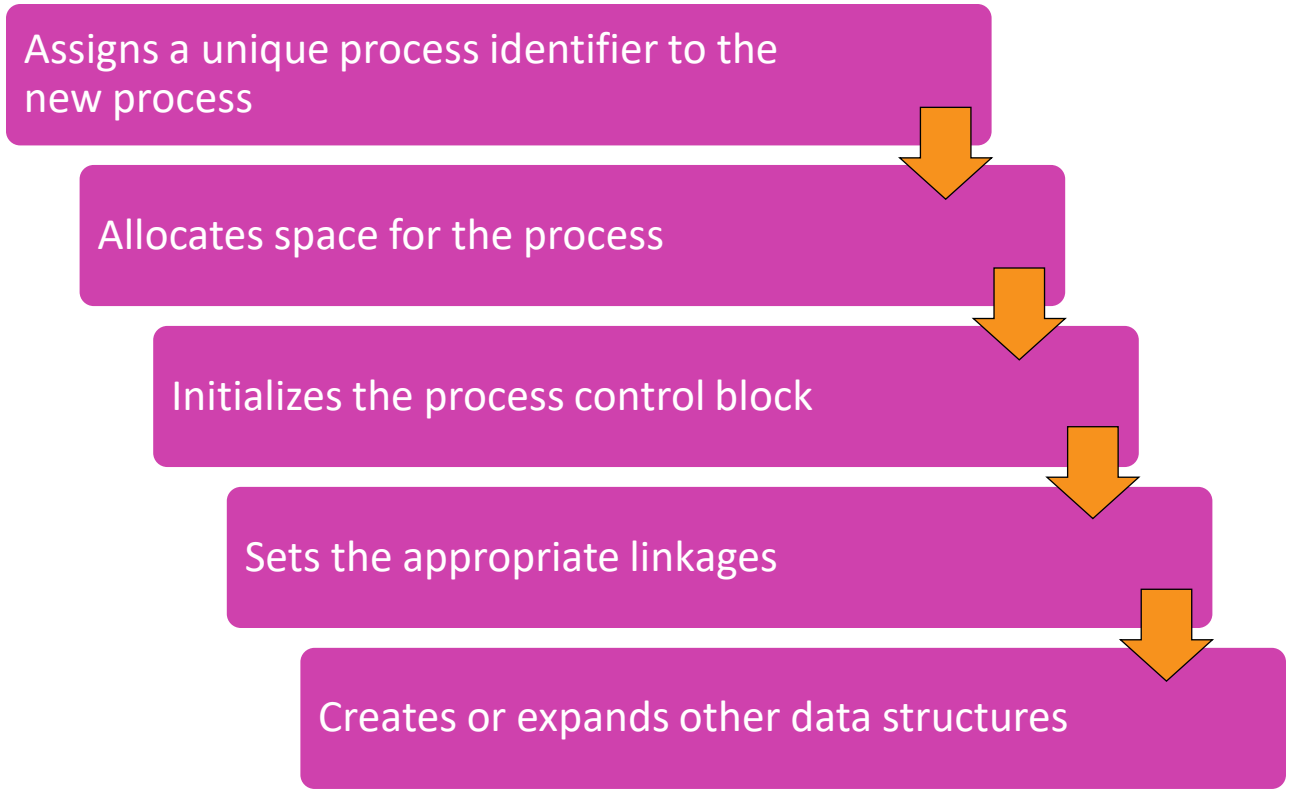
- More-privileged mode
- Also referred to as control mode or kernel mode
- Kernel of the operating system

Functions of an Kernel

- Process Management
  - Process creation and termination
  - Process scheduling and dispatching
  - Process switching
  - Process synchronization and support for interprocess communication
  - Management of process control blocks
- Memory Management
  - Allocation of address space to processes
  - Swapping
  - Page and segment management
- I/O Management
  - Buffer management
  - Allocation of I/O channels and devices to processes
- Support Functions
  - Interrupt handling
  - Accounting
  - Monitoring

Process Creation

- Once the OS decides to create a new process it:



Mechanisms for Interrupting the Execution of a Process

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

System Interrupts

Interrupt

- Due to some sort of event that is external to and independent of the currently running process
  - Clock interrupt
  - I/O interrupt
  - Memory fault
- Time slice
  - The maximum amount of time that a process can execute before being interrupted

Trap

- An error or exception condition generated within the currently running process
- OS determines if the condition is fatal
  - Moved to the Exit state and a process switch occurs
  - Action will depend on the nature of the error the design of the OS

Mode Switching

If no interrupts are pending the processor:

Proceeds to the fetch stage and fetches the next instruction of the current program in the current process

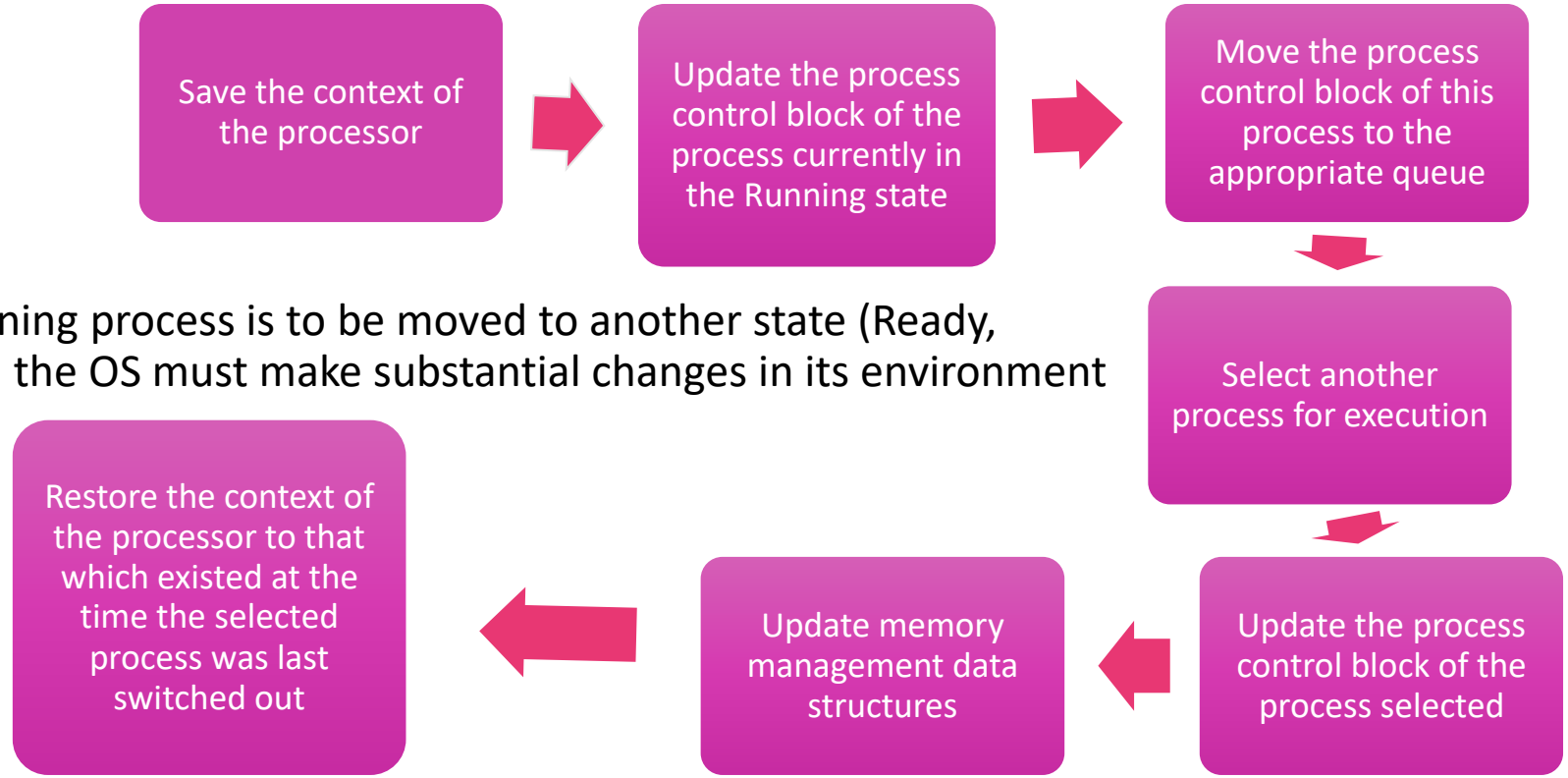
If an interrupt is pending the processor:

Sets the program counter to the starting address of an interrupt handler program

Switches from user mode to kernel mode so that the interrupt processing code may include privileged instructions

Change of Process State

- The steps in a full process switch are:



If the currently running process is to be moved to another state (Ready, Blocked, etc.), then the OS must make substantial changes in its environment

Process Control

- Process creation is by means of the kernel system call, *fork()*
- When a process issues a fork request, the OS performs the following functions:

1. Allocates a slot in the process table for the new process
2. Assigns a unique process ID to the child process
3. Makes a copy of the process image of the parent, with the exception of any shared memory
4. Increments counters for any files owned by the parent, to reflect that an additional process now also owns those files
5. Assigns 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. Control returns to user mode at the point of the fork call of the parent.
  - Transfer control to the child process. The child process begins executing at the same point in the code as the parent, namely at the return from the fork call.
  - Transfer control to another process. Both parent and child are left in the Ready to Run state.

Summary

- What is a process?
  - Background
  - Processes and process control blocks
- Process states
  - Two-state process model
  - Creation and termination
  - Five-state model
  - Suspended processes
- Process description
  - Operating system control structures
  - Process control structures
- Process control
  - Modes of execution
  - Process creation
  - Process switching
- Execution of the operating system
  - Nonprocess kernel
  - Execution within user processes
  - Process-based operating system
- UNIX proces creation

Thank you for your attention!