Process Management

Process Concepts

- The notion of a process is fundamental to OS and it defines the fundamental unit of computation for the computer and used by OS for concurrent program execution.

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 a single sequential thread of execution, a current state, and an associated set of system resources

The components of process are:

- Object Program: Code to be executed
- Data: Data used for executing the program
- Resources: Resources needed for execution of the program
- Status of the process execution: Used for verifying the status of the process execution.
- A process is allocated with resources such as memory and is available for scheduling. It can run to completion only when all requested resources have been allocated to the process. Two or more processes could be executing the same program, each using their own data and resources.

Difference between a program and a process

A program is not the same as a process.

- A program is a static object which contains instructions that can exist in a file.
  - Program = static file (image)
- A process is dynamic object that is a program in execution.
  - Process = executing program = program + execution state.
- A program is a sequence of instructions whereas process is a sequence of instruction executions.
- A program exists at a single place in space and continues to exist as time goes forward whereas a process exists in a limited span of time.
- A program does not perform the action by itself whereas different processes may run different instances of the same program

Running a program

A program consists of code and data

- On running a program, the loader:
  - reads and interprets the executable file
  - sets up the process's memory to contain the code & data from executable file
  - pushes "argc", "argv" on the stack
  - sets the CPU registers properly & calls "_start()"

- Program starts running at _start()

```
_start(args) {
initialize_java();
ret = main(args);
exit(ret)
}
```

- Now “process” is running, and no longer it is “program”
- When main() returns, OS calls “exit()” which destroys the process and returns all resources

A process includes: program counter; stack; data section

Process image

– Collection of programs, data, stack, and attributes that form the process

– User data
  - Modifiable part of the user space
  - Program data, user stack area, and modifiable code

– User program
  - Executable code

– System stack
  - Used to store parameters and calling addresses for procedure and system calls

– Process control block
  - Data needed by the OS to control the process

– Location and attributes of the process
  - Memory management aspects: contiguous or fragmented allocation

Keeping track of a process

- A process has code.
  - OS must track program counter (code location).
- A process has a stack.
  - OS must track stack pointer.
- OS stores state of processes' computation in a Process Control Block (PCB).
  - E.g., each process has an identifier (process identifier, or PID)
- Data (program instructions, stack & heap) resides in memory, metadata is in PCB (which is a kernel data structure in memory)

Implementation of a Process

Process Control Block (PCB)

- Each process contains the process control block (PCB). PCB is the data structure used by the OS and all information about a particular process such as Process id, process state, priority, privileges, memory management information, accounting information etc. are grouped by OS.
- PCB also includes the information about CPU scheduling, I/O resource management, file management information, priority and so on.
- The PCB simply serves as the repository for any information that may vary from process to process.

Contents of PCB are

1.Pointer: Pointer points to another process control block. Pointer is used for maintaining the scheduling list.

2. Process id

3. Process State: Process state may be new, ready, running, waiting and so on.

4. Program Counter: It indicates the address of the next instruction to be executed for this process.

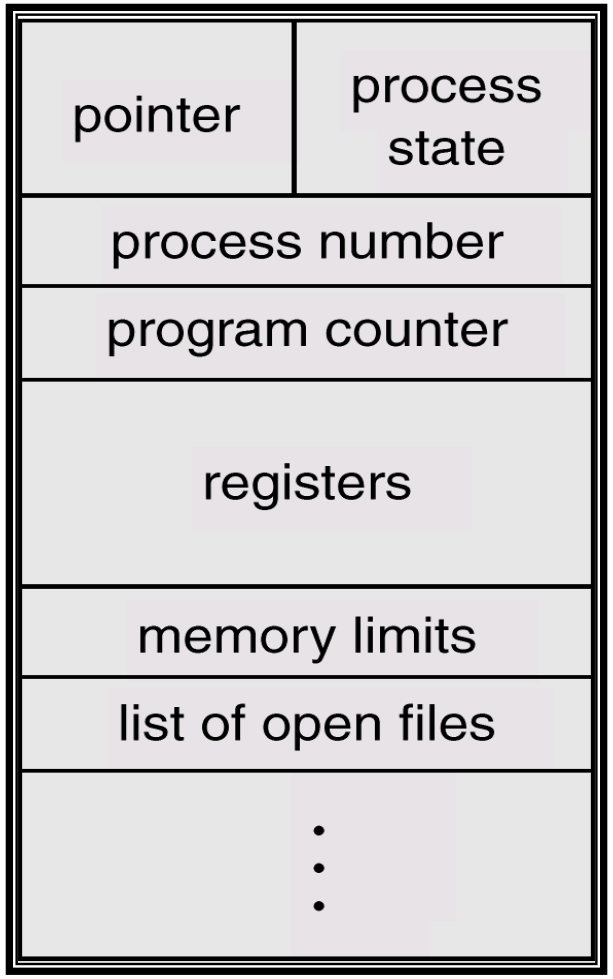
5. Event information: For a process in the blocked state this field contains information concerning the event for which the process is waiting.

6. CPU register: It indicates general purpose register, stack pointers, index registers and accumulators etc. number of register and type of register totally depends upon the computer architecture.

7. Memory Management Information: This information may include the value of base and limit register. This information is useful for deallocating the memory when the process terminates.

8. Accounting Information: This information includes the amount of CPU and real time used, time limits, job or process numbers, account numbers etc.

Process Control Block (PCB)



- When a process is created, hardware registers and flags are set to the values provided by the loader or linker.
- Whenever that process is suspended, the contents of the processor register are usually saved on the stack and the pointer to the related stack frame is stored in the PCB.
- In this way, the hardware state can be restored when the process is scheduled to run again.

Operations on Processes

- The OS as well as other processes can perform operations on a process. Several operations are possible on the process such as create, kill, signal, suspend, schedule, change priority, resume etc.
- OS must provide the environment for the process operations.
- Two main operations that are to be performed are :
  - process creation
  - process deletion

Process Creation

OS creates the very first process when it initializes. That process than starts all other processes in the system using the create system calls.

OS creates a new process with the specified or default attributes and identifier.

A process may create several new sub-processes.

Syntax for creating new process is :

- CREATE (processid, attributes)

- When OS issues a CREATE system call, it obtains a new process control block from the pool of free memory, fills the fields with provided and default parameters, and insert the PCB into the ready list.
- Thus it makes the specified process eligible for running.
- When a process is created, it requires some parameters. These are priority, level of privilege, requirement of memory, access right, memory protection information etc.
- Process will need certain resources, such as CPU time, memory, files and I/O devices to complete the operation.

Process Hierarchy

- When one process creates another process, the creator is referred as parent and the created process is the child process. The Child process may create another process. So it forms a tree of processes which is referred as process hierarchy.
- When process creates a child process, that child process may obtain its resources directly from the OS, otherwise it uses the resources of parent process. Generally a parent is in control of a child process.

- In Operating System, the fork() system call is used by a process to create another process. The process that used the fork() system call is the parent process and process consequently created is known as the child process.

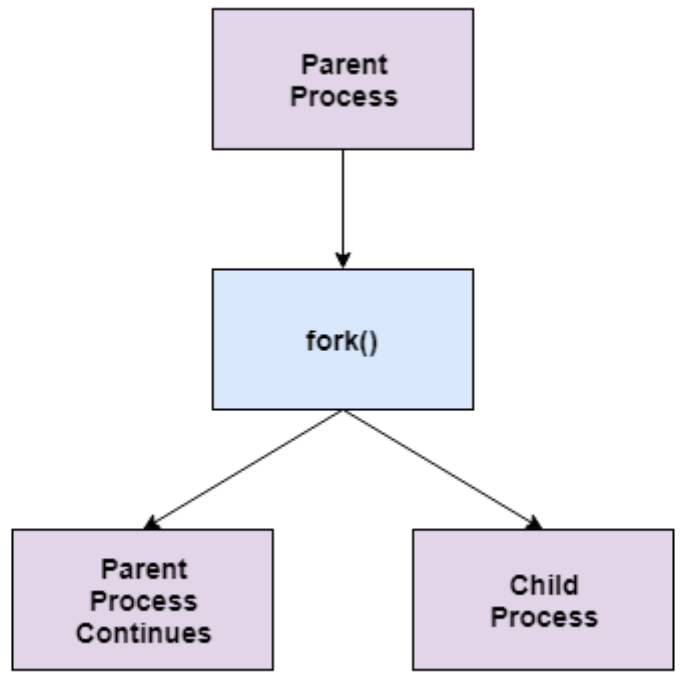
Parent Process

- All the processes in operating system are created when a process executes the fork() system call except the startup process. The process that used the fork() system call is the parent process. In other words, a parent process is one that creates a child process. A parent process may have multiple child processes but a child process only one parent process.

- On the success of a fork() system call, the PID of the child process is returned to the parent process and 0 is returned to the child process. On the failure of a fork() system call, -1 is returned to the parent process and a child process is not created.

Child Process

- A child process is a process created by a parent process in operating system using a fork() system call. A child process may also be called a subprocess or a subtask.
- A child process is created as its parent process's copy and inherits most of its attributes. If a child process has no parent process, it was created directly by the kernel.
- If a child process exits or is interrupted, then a SIGCHLD signal is send to the parent process.


Parent
Process
fork()
Parent
Process
Continues
Child
Process

When a process creates a new process, two possibilities exist in terms of execution.

1. Concurrent : The parent continues to execute concurrently with its children.

2. Sequential :The parent waits until some or all of its children have terminated.

- For address space, two possibilities:
    1. The child process is a duplicate of the parent process.
    2. The child process has a program loaded into it.
- Resource sharing possibilities
    1. Parent and child share all resources
    2. Children share subset of parent’s resources
    3. Parent and child share no resources

Process Termination

- DELETE system call is used for terminating a process.
- A process may delete itself or by another process. A process can cause the termination of another process via an appropriate system call.
- The OS reacts by reclaiming all resources allocated to the specified process, closing files opened by or for the process. PCB is also removed from its place of residence in the list and is returned to the free pool.
- The DELETE service is normally invoked as a part of orderly program termination. Parent may terminate execution of children processes (abort) if Child has exceeded allocated resources or Task assigned to the child is no longer required or parent is exiting.
- OS may not allow child to continue if its parent terminates. This may result in cascading termination.

Zombie Process:

A child process which has finished the execution but still has entry in the process table goes to the zombie state.

A child process always first becomes a zombie before being removed from the process table.

The parent process reads the exit status of the child process which reaps off the child process entry from the process table.

Orphan Process:

A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an orphan process.

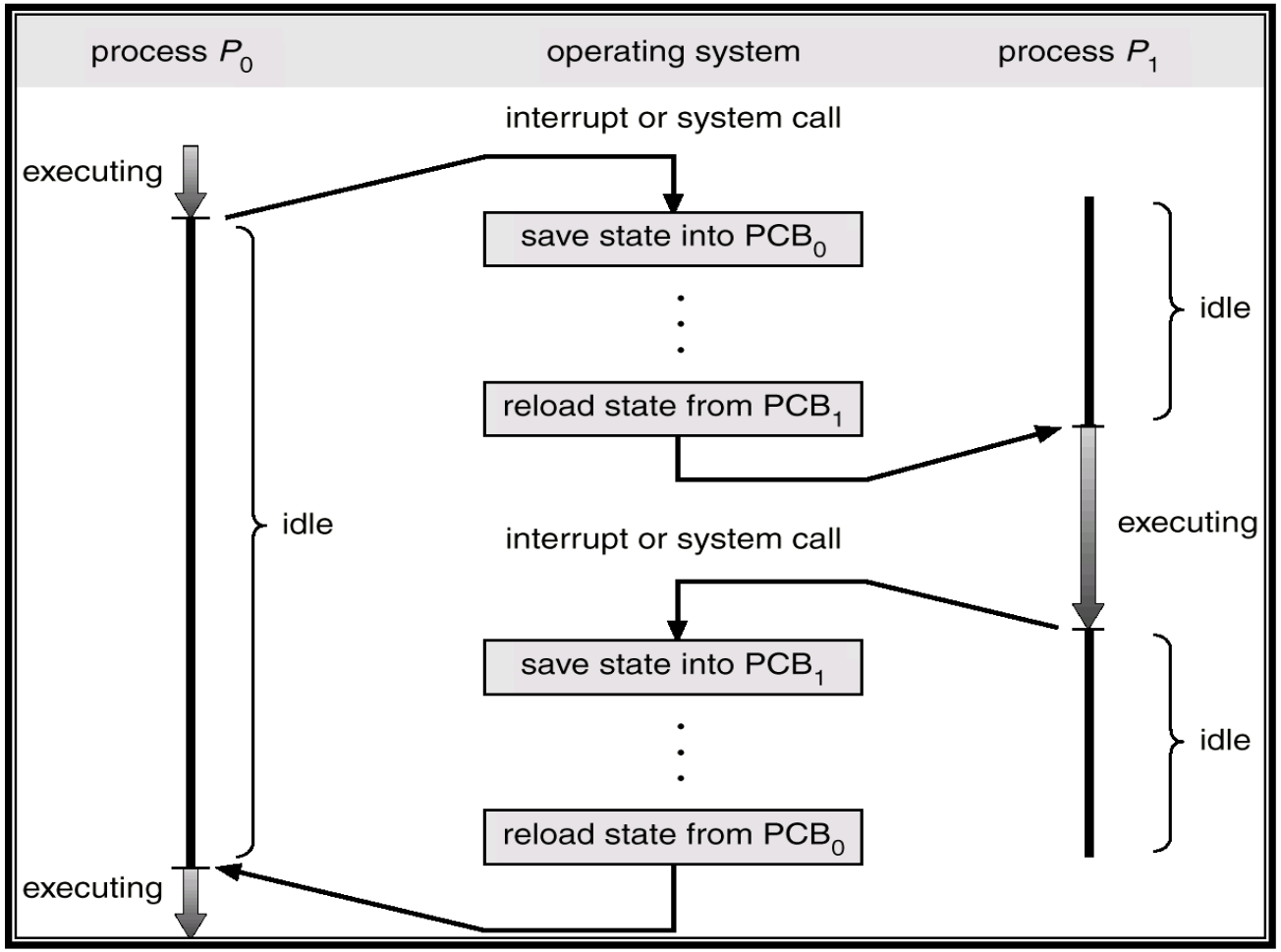
Static and Dynamic Process:

- A process that does not terminate while the OS is functioning is called static.
- A process that may terminate is called dynamic.

Context Switch

- When the scheduler switches the CPU from executing one process to executing another, the context switcher saves the content of all processor registers for the process being removed from the CPU in its process descriptor.
- The context of a process is represented in the PCB of a process. Context switch time is purely an overhead.
- Context switching can significantly affect performance, since modern computers have a lot of general and status registers to be saved.
- Context switch times are highly dependent on hardware support. Some hardware systems employ two or more sets of processor registers to reduce the amount of context switching time.
- When the process is switched, the information stored are Program Counter value, Scheduling Information, Base and limit register value, Currently used register, Changed State, I/O State and accounting.

CPU Switch from Process to Process

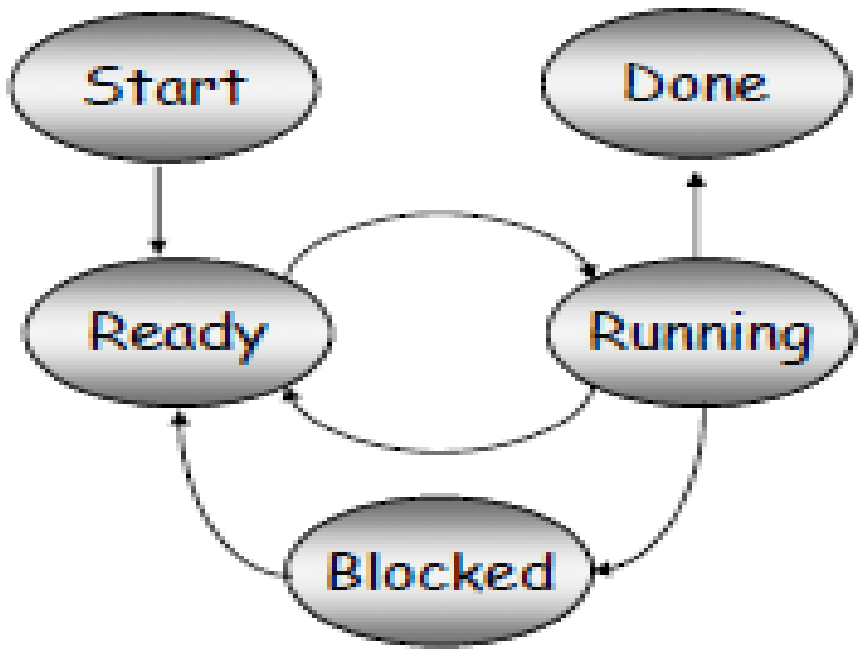


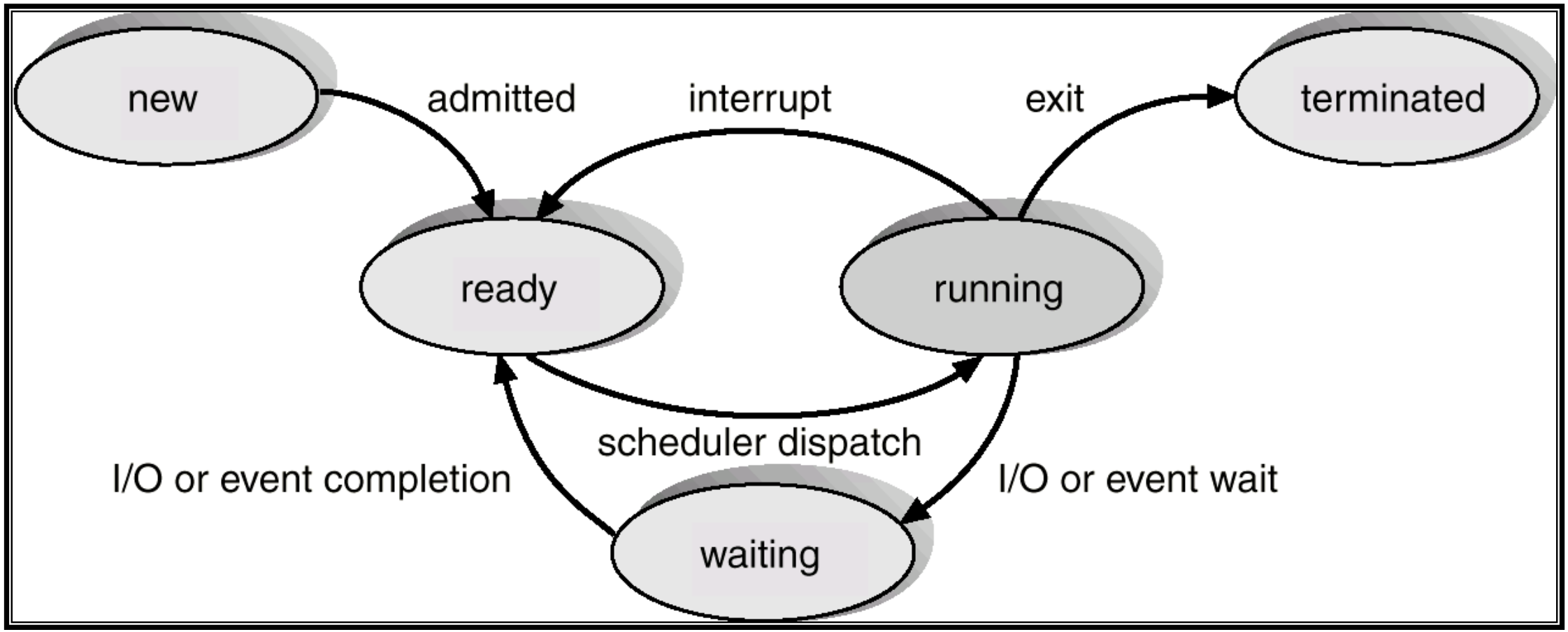
Life Cycle of a Process

When process executes, it changes state. Process state is defined as the current activity of the process.

Once created a process can be in any of the following three basic states:

- Ready: The process is ready to be executed, but CPU is not available for the execution of this process.
- Running: The CPU is executing the instructions of the corresponding process. A running process possesses all the resources needed for its execution, including the processor.
- Blocked/ Waiting: The process is waiting for an event to occur.
  - The event can be I/O operation to be completed
  - Memory to be made available
  - A message to be received etc.


Start
Done
Ready
Running
Blocked


new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting

- A running process gets blocked because of a requested resource is not available or can become ready because of the CPU decides to execute another process.
- A blocked process becomes ready when the needed resource becomes available to it. A ready process starts running when the CPU becomes available to it.
- Whenever processes changes state, the OS reacts by placing the process PCB in the list that corresponds to its new state. Only one process can be running on any processor at any instant and many processes may be in ready and waiting state.