Process Management

Chapter 2

Objectives

- **After studying this unit, you should be able to**
  - Explain process state
  - Describe the process control block(PCB)
  - Discuss the process scheduling
  - Describe threads in Operating system

Process Concept

- An operating system executes a variety of programs
  - batch systems - jobs
  - time-shared systems - user programs or tasks
  - job and program used interchangeably
- Process - a program in execution
  - process execution proceeds in a sequential fashion
- A process is an activity of some kind, it has program, input, output and state.

Programs and Processes

- A process is different than a program.
- Consider the following analogy
- *Scenario-1: A computer scientist is baking a birthday cake for his son*
  - *Computer scientist* - *CPU*
  - *Birthday cake recipe* - *program*
  - *Ingredients* - *input data*
  - *Activities:* - *processes*
    - reading the recipe
    - fetching the ingredients
    - backing the cake

Programs and Processes

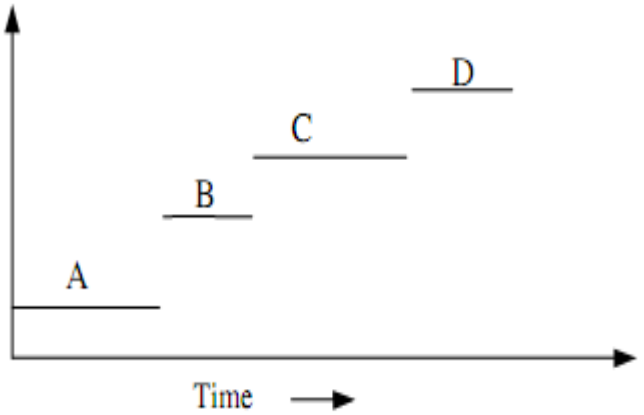
- *Scenario-2: Scientist's son comes running in crying, saying he has been stung by a bee.*
  - Scientist records where he was in the recipe
    - the state of running process saved
  - Reach first aid book and materials
    - Another process fetched
  - Follow the first aid action (high priority job)
    - Processor switched for new process
  - On completion of aid, cake baking starts again from where it was left
    - Completion of high priority job & return back to the last one

Process Model

- Uniprogramming
- Multiprogramming
- Multiprocessing

Uniprogramming Model

Only one process at a time

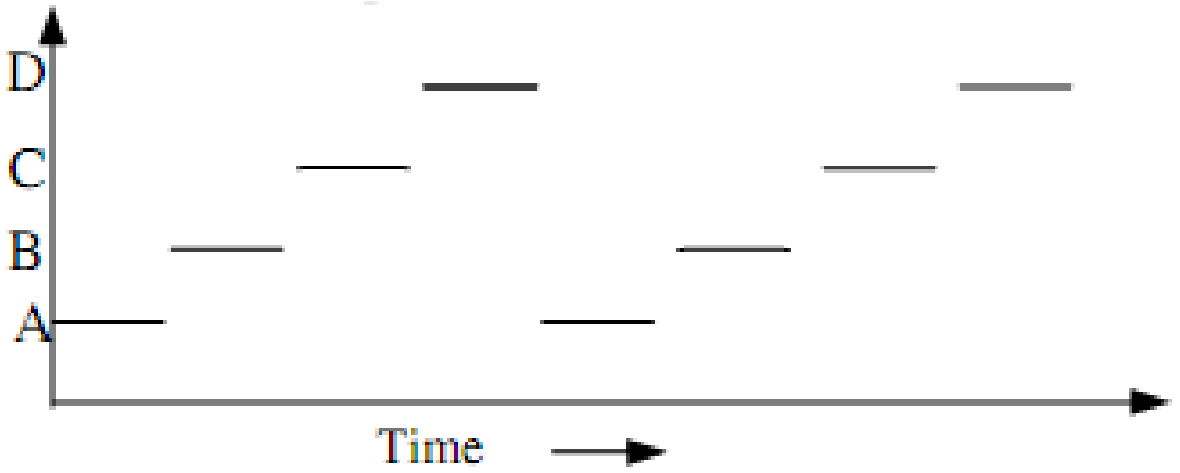


- Examples: Older systems
- Advantages: Easier for OS designer
- Disadvantages: Not convenient for user and poor performance

Multiprogramming Mode

multiple process at a time

- OS requirements for multiprogramming:
- Policy: to determine which process is to schedule.
- Mechanism: to switch between the processes.

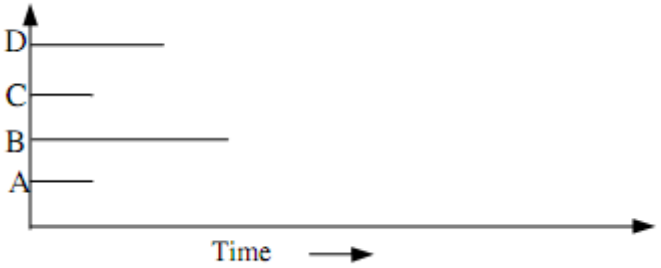


- Examples: Unix, WindowsNT
- Advantages: Better system performance and user convenience.
- Disadvantages: Complexity in OS

Multiprocessing Model

System with multiple processor

- Distributed OS and Network OS



Process States diagram

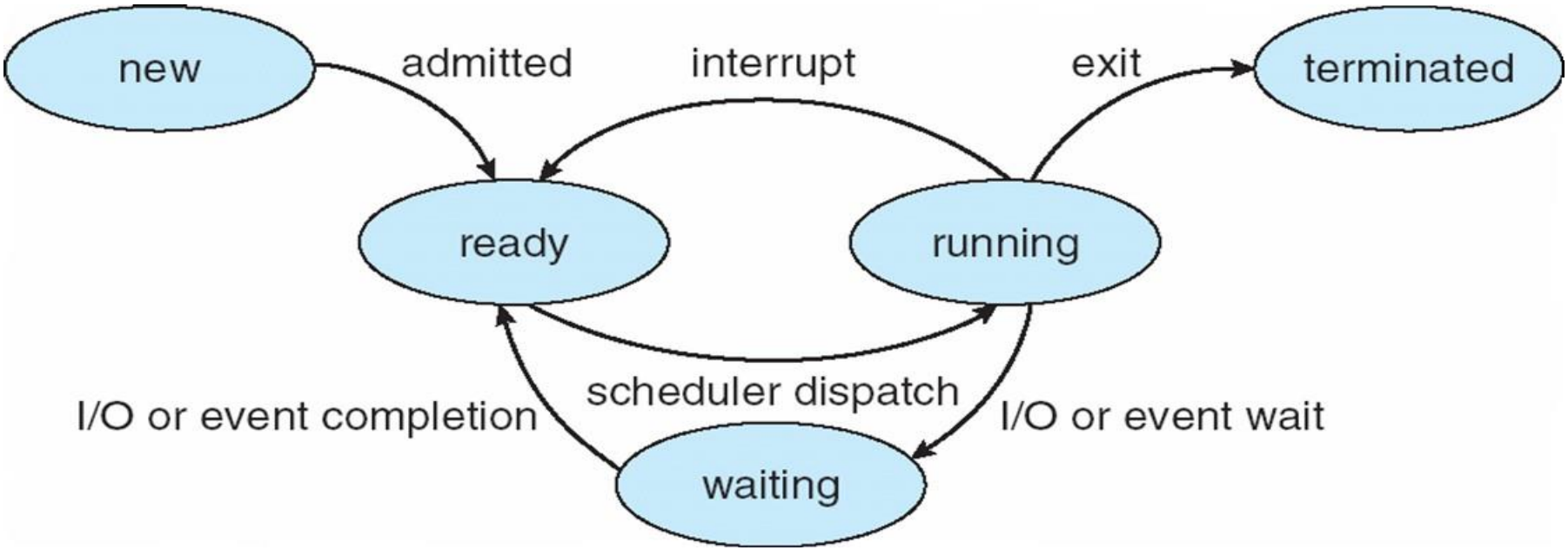


Figure: Process State Transition Diagram

Process States

- **New -** The process is being created.
- **Running -** Instructions are being executed.
- **Waiting -** Waiting for some event to occur.
- **Ready -** Waiting to be assigned to a processor.
- **Terminated -** Process has finished execution.

Process State Transitions

- *When a job is admitted to the system, a corresponding process is created and normally inserted at the back of the ready list.*
- When the CPU becomes available, the process is said to make a state transition from ready to running.
  - *ready -> running.*
- To prevent any one process from monopolizing the CPU, OS specify a time period (quantum) for the process. When the quantum expire or interrupt to CPU makes state transition running to ready.
  - *running -> ready.*
- When the process requires an I/O operation before quantum expire or interrupt, the process voluntarily relinquishes the CPU and changed to the wait state.
  - *running -> wait.*
- When an I/O operation completes. The process make the transition from wait state to ready state.
  - wait -> *ready.*
- When process finished execution it will terminated

Process Control Block

- Process must be saved when the process is switched from one state to another so that it can be restarted later as it had never been stopped.
- The PCB is the data structure containing certain important information about the process -also called process table or processor descriptor.
  - **Process state:** running, ready, blocked.
  - **Program counter:** Address of next instruction for the process.
  - **Registers:** Stack pointer, accumulator, PSW etc.
  - **Scheduling information:** Process priority, pointer to scheduling queue etc.
  - **Memory-allocation:** value of base and limit register, page table, segment table etc.
  - **Accounting information:** time limit, process numbers etc.
  - **Status information:** list of I/O devices, list of open files etc.

PCB

- The PCB simply serves as repository of any information that may vary from process to process.

Pointer	Process state
Process number	
Program counter	
Registers	
Memory Limits	
List of open files	
…….	

Operation on process

- The processes in the system can execute concurrently, and they must be created and deleted dynamically. OS provide the mechanism for process creation and termination.
  - Process Creation.
  - Process Termination

Process Creation

- There are four principal events that cause the process to be created:
  - System initialization.
  - Execution of a process creation system call.
  - User request to create a new process.
  - Initiation of a batch job.
- A process may create several new processes during the course of execution. The creating process is called a parent process, where as the new processes are called children of that process.

Process Termination

- Process are terminated on the following conditions
  - Normal exit.
  - Error exit.
  - Fatal error.
  - Killed by another process.
- Example:
  - In Unix the normal exit is done by calling a exit system call. The process return data (output) to its parent process via the wait system call.
  - kill system call is used to kill other process.

Process Scheduling

- Main objective of multiprogramming is to maximize the CPU utilization
- At any time, a process is in any one of the new, ready, running or waiting state.
- The OS module known as scheduler schedules processes from ready queue for execution by CPU
- Scheduler select one of the many process from ready queue on certain criteria.
- (Will discuss it in detail in next chapter)

Context Switch

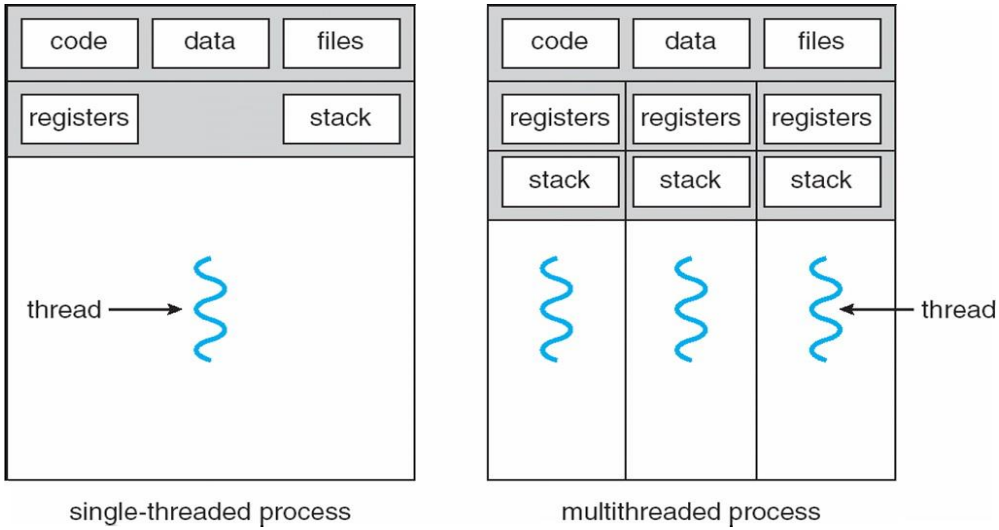
- CPU switching form one process to another requires saving the state of the current process and loading the latest state of next process.
- This is known as context switch.
- The time require to context switch is a overhead since it does not produce any useful work in aspect of process execution time.

Thread

- It is a single sequence stream which allows a program to split itself into two or more simultaneously running task.
- It is the basic unit of CPU utilization and consists of program counter, a register set and a stack space.
- Sometime called light weight process.

Multithreading

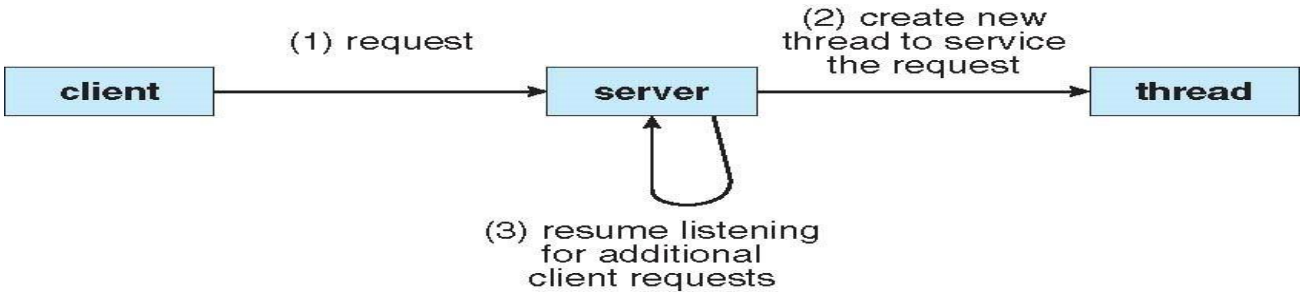
- All threads share the same address space, global variables, set of open file, child processes, alarms and signals etc.



- *Figure (b):* three threads are actually part of the same job and are actively and closely cooperating with each other.
- Each thread maintain its own stack.

Multithreading Benefits

- In modern operating system multithreading has several benefits as :
  - Responsiveness – program can run continuously even if the part of it is blocked
  - Resource Sharing – Share the code and data
  - Economy – economic in resources and context switching
  - Scalability – supports multiprocessor architecture that increase concurrency
- Threads also play a vital role in remote procedure call (RPC) systems.
- When a server receives a message, it services the message using a separate thread. This allows the server to service several concurrent requests



- Most operating system kernels are now multithreaded; several threads operate in the kernel, and each thread performs a specific task, such as managing devices or interrupt handling.

Thread and Process

- Threads are not independent as process.
- Thread shares with other thread: code section, data section, Other resource such as opens files.
- Similarities:
  - Like process, thread share CPU and only one thread is running at a time.
  - Like process, thread within processes execute sequentially
  - Like process if one thread is blocked, another can run.

Thread and Process

- Dissimilarities
  - Unlike process, threads are not independent of each other.
  - Unlike process, all threads can access every address space.
  - Unlike process, threads are designed to assist one other.

Benefits of thread

- A process with multiple thread makes great server. Example print server
- Because thread can share common data, they don’t need to use interprocess communication
- Threads are chips in the sense that
  - They only need a stack and registers, they are cheap to create.
  - Thread use very little resources of an OS i.e threads don't need new address space, global data new OS resources etc.

Benefits of threads

- Context switching
  - Threads are very inexpensive to create and destroy and they are inexpensive to represent.
  - For example thread require space to store the PC, SP(stack pointer), general propose register but do not require space to share memory information , information about opens files etc. With this little context, it is much easier to switch between thread.
- Sharing
  - Threads allow the sharing of a lot resources that cannot be shared in process. For example, sharing of code section, data section and other OS resources.

Users and Kernel Threads

User Threads:

- Thread management done by user-level threads library.
  - Implemented as a library
  - Library provides support for thread creation, scheduling and management with no support from the kernel.
  - Fast to create
  - If kernel is single threaded, blocking system calls will cause the entire process to block.

    Examples: POSIX threads, Win32 API threads, Java threads

Kernel Threads:

- Supported by the Kernel
  - Kernel performs thread creation, scheduling and management in kernel space.
  - Slower to create and manage
  - Blocking system calls are no problem
  - Most OS's support these threads

    Examples: Windows XP/2000, Solaris, Linux, Tru64 UNIX, Mac OS X

Home Works

1. What are disadvantages of too much multiprogramming?
2. List the definitions of process.
3. What are the two differences between the kernel level threads and user level threads? Which one has a better performance?
4. List the differences between processes and threads.
5. What resources are used when a thread is created? How do they differ from those used when a process is created?