CSC-203 Operating Systems[3]

Lecture: 5

Instructor: Sanjog Sigdel

sigdelsanjog.com.np

Date: May 11, 2019

Lecture 5

Unit2: Process Management

Introduction:

Process Vs Program, Multiprogramming, Process Model, Process States, Process Control Block/Process Table

Lecture 5

Process Management

Process:

- A Process is an **instance of a program running in a computer**
- Like a task, a process is a running program with which a particular set of data is associated so that the process can be kept track of.
- It's a **program in execution**. So in computing, a process is the **instance of a computer program that is being executed**
- A Process has a **program, unput, output, and a state.**

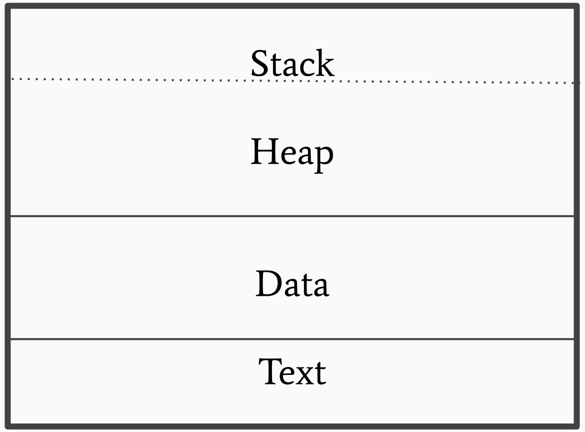
- A process can initiate a sub-process, which is called a child process

Lecture 5

Process Management

Process:

- We write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program
- When a program is loaded into the memory and it becomes a process, it can be divided into four sections
    - Stack, heap, text and data



Lecture 5

Process Management

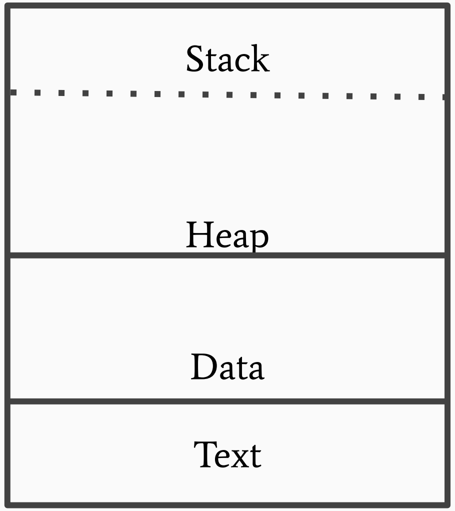
Process:

Stack: The Process stack contains the temporary data such as method/function, parameters return address

Heap: This is dynamically allocated memory to a process during its run time

Text: This includes the current activity represented by the value of Program Counter

Data: This section contains the global and static variables



Lecture 5

Process Management

Program:

- Program is an **executable file containing the set of instructions** written to perform a specific job on your computer.
- Programs are **not stored on the primary memory** in your computer but stored on a disk or a secondary memory on your computer
- Programs are **read into the primary memory and executed by the kernel**
- **Examples:** *notepad.exe, chrome.exe, explorer.exe*

Lecture 5

Process Management

Program vs Process

1. A Program is a **definite group of ordered operations that are to be performed**. On the other hand, **an instance of a program being executed is a process**
2. The nature of the program is **passive as it does nothing until it gets executed** whereas a process is **dynamic or active in nature as it is an instance of executing program** and performs the specific action
3. A program has a **longer lifespan** because it is stored in the memory **until it is not manually deleted** while a process has a **shorter lifespan** because it gets terminated to **the completion of the task**
4. The resource requirement is **much higher in case of a process**; it could need processing, memory. I/O resources for the successful execution. In contrast, a **program just requires memory for storage**

Lecture 5

Process Management

Multiprogramming

- Ability of an operating system to execute more than one program on a single processor machine
- More than one task/program/job/process can reside into the memory at the one point of time
- Example: Computer running Chrome and Media Player at the simultaneously

In a multiprogramming system there are **one or more programs loaded in main memory which are ready to execute**. Only **one program at a time is able to get the CPU for executing** its instructions (i.e. there is at most one process running on the system) **while others are waiting their turn**

Lecture 5

Process Management

Multiprogramming

- Main idea of multiprogramming is to **maximize the use of CPU time**, perform **process context switching** and provide resource.
- Suppose the currently running process is performing an I/o task(which, by definition, doesn't need the CPU to be accomplished). Then, the OS may interrupt that process and give the control to one of the other programs that are ready to execute

Issues:

1. Large programs may not fit at once in memory (can be solved by using virtual memory)
2. If N ready processes all highly CPU-bound, in worst case onr program might wait all the other N-1 ones to complete before executing

Process Management

Multiprogramming

- Main idea of multiprogramming is to **maximize the use of CPU time**, perform **process context switching** and provide resource.
- Suppose the currently running process is performing an I/o task(which, by definition, doesn't need the CPU to be accomplished). Then, the OS may interrupt that process and give the control to one of the other programs that are ready to execute

Issues:

1. Large programs may not fit at once in memory (can be solved by using virtual memory)
2. If N ready processes all highly CPU-bound, in worst case onr program might wait all the other N-1 ones to complete before executing

Process Management

Process Model

A process model **organizes all the runnable software on the computer into a number of sequential process**

Following are four principal events that cause the processes to be created

- System initialization
- Execution of a process creation system call by a running process
- A user request to create a new process
- Initiation of a batch work

Process Management

Process Model

- Some processes are created whenever an operating system is booted
- **Foreground** processes and **background** processes
- **Foreground:** process that interacts with the computer users of computer programmers
- **Background:** process that execute specific functions(*mostly resource handlers)*

In UNIX system **ps** program can be used to list all the running processes
In windows task manager.

Lecture 5

Process Management

Process Model

A process normally exists due to following conditions:

Normal Exit: process terminates because they have done their work successfully

Error Exit: termination of a process is done because of an error caused by the process, sometimes due to the program bug

Fatal Exit: process terminates because it discovers a fatal error

Killed by another process: other system calls instruction might exit some other process

Lecture 5

Process Management

Process States

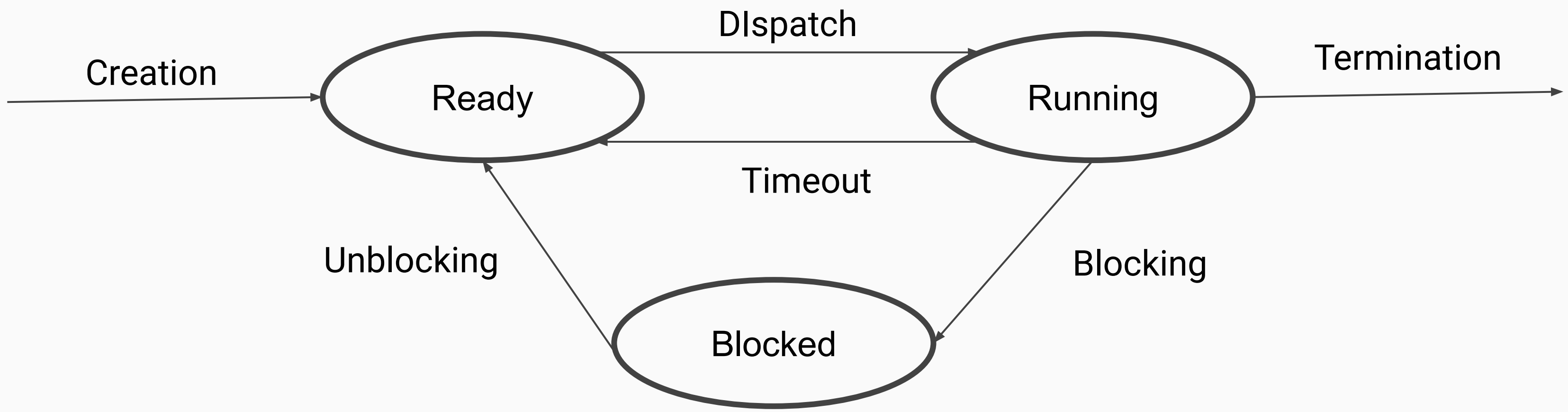


Figure: Process Life Cycle

Lecture 5

Process Management

Process States

Start: Initial state when a process is first started/created
Ready: process is waiting to be assigned to a processor
Running: Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions
Waiting: Process moves into the waiting state if it needs to **wait for a resource, such as waiting for user input, or waiting for a file to become available**
Terminated: process finishes its execution or terminated by the operating system

Process Management

Process Control Block(PCB)
To identify the processes in operating system each process is assigned with a process identification number(pid). As Operating System supports multiprogramming, it needs to keep track of all processes.

Process Control Block is used to track the process execution status.
A memory block contains information about the process state, program counter, stack pointer, status of opened files. All the information is required and must be saved when the process is switched from one state to another processes.

Lecture 5

Process Management

Role of Process Control Block(PCB)

- In process management PCB can accesss or can be modified by most OS utilities including those are involved with memory, scheduling and IO resource access
- Set of PCB gives the information of current state of operating system

Information contained by a process control block:

1. Naming the process
2. State of the process
3. Resources allocated to the process
4. Memory allocated to the process
5. Schedule information
6. Input/Output devices associated with process

Process Management

Components of PCB

1. **Process State:**
   new, running, waited, block, suspended, terminated
2. **Process Privileges:**
   Required to **allow/disallow** access to system resources
3. **Process ID:**
   Contains unique ID of running processes
4. **Pointer:**
   A pointer to parent process
5. **Program Counter:**
   Pointer to the address of the next instruction to be executed for this process

Process Management

Components of PCB

6. CPU registers

- Information comprising with the various registers, such as index and stack that are associated with the process

7. CPU Scheduling Information

- Scheduling information is used to set the priority of different processes

8. Memory Management Information

- Information of page table, memory limits, segment table depending on memory used by operating system

9. Accounting information

- Amount of CPU used for process execution time limits, execution ID etc.

10. *IO* status information

- List of I/O devices allocated to the process

Next Lecture

Threads: Definition, Thread vs Process, Thread Usage, User and Kernel Space Threads

Q&A

For Further Queries:
sigdelsanjog@gmail.com

Add [NISTBanepa] in Subject for any of your queries

THANK YOU