Unit-II
Lecture: 5
(Process Management)
(Part-II)

Process Creation:

When an operating system is booted, typically numerous processes are created. Some of these are foreground processes i.e. processes that interact with users and work for them. Other runs in the background and are not associated with particular users, but instead have some specific function.

Four principal events cause processes to be created:

1. System initialization
2. Execution of a process-creation system call by a running process.
3. A user request to create a new process
4. Initiation of a batch job

Daemon Process:

A daemon process is a background process that is not under the direct control of the user.

This process is usually started when the system is bootstrapped/started and is terminated with the system shut down.

The daemon process names normally end with a d.

Some of the examples of daemon processes in Unix are:

- crond: This is a job scheduler that runs jobs in the background.
- syslogd: this is a system logger that implements the system logging facility and collects system messages.
- httpd: this is the web server daemon process that handles the Hypertext Transfer Protocol.
- dhcpd: This daemon configures the TCP/IP information for users dynamically.

In UNIX, the ps program can be used to list the running processes while in Windows task manager can be used.

In addition to the processes created at boot time, new processes can be created afterwards as well.

Often a running process will issue system calls to create one or more new processes to help it do its job.

In interactive systems, users can start a program by typing a command or double clicking an icon. Taking either of these actions starts a new process and runs the selected program in it.

Technically, a new process is created by having an existing process execute a process creation system call. That process may be a running user process, a system process invoked from the keyboard or mouse. What that process does is execute a system call to create a new process.

In UNIX, there is only one system call to create a new process: fork.
This call creates an exact clone of the calling process.

\In windows, a single Win32 function call, CreateProcess, handles both process creation and loading the correct program into the new process

What does a process look like in memory?

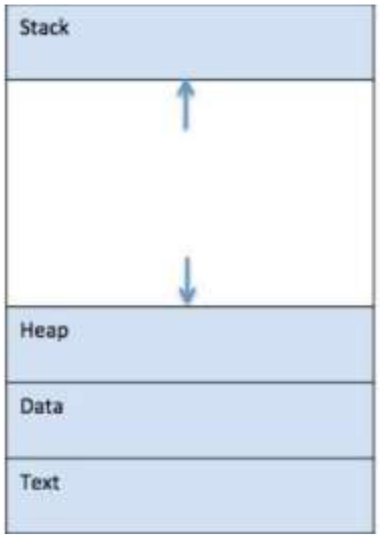
A process has its state of execution which is described with the program counter(PC), the stack pointer(SP) and all this information is used by the operating system to decide how to schedule the process, how to swap between multiple process and for other management of tasks.

- A stack pointer is a small register that stores the address of the last program request in a stack.

A process encapsulated all of the data for running applications and this includes the code, the data, the variables etc.
Every single element of the process state has to be uniquely identified by its address. So an OS abstraction used to encapsulate all of the process states is address space.

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.**

The following image shows a simplified layout of a process inside main memory:



- **Stack:**
  - The process Stack contains the temporary data such as method/ function parameters, return address and local variables.
  - It is a dynamic part of the address space and it grows and shrinks during the execution with a last-in,first-out (LIFO) order.
  - In the diagram, arrows show that stack and heap grow in opposite direction.
- **Heap:**
  - This is dynamically allocated memory to a process during its run time.
  - During the execution, the process dynamically
    - creates some state
    - allocated memory
    - store the temporary results etc.
- **Data:**
  - This section contains the global and static variables.
- **Text:**
  - This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
  - This segment is a read-only space.