**What is process?**

Process is a program under execution. Remember, a program is not a process by itself. A program is a *passive* entity, such as a file containing a list of instructions stored on disk (often called an executable file). However, a process is an *active* entity. Process reside in the main memory and it should occupy the CPU. There are some cases when a process reside in the secondary memory, I will discuss that in later slides. Two commons ways to load an executable file into the memory are double clicking on the executable file or by typing the name of the executable file in the command line.

The structure of the process in the memory looks like this. Text section contains the program code. Data section contains global variable, Heap contains a memory that is dynamically allocated during the run time whereas stack contains temporary data such as function parameters, return addresses, and local variables. Those two arrows represents the current activity based on the value of program counter.

Lets take an example, if a user is running many copies of same program such as web browser, than each of these copies is a separate process. In that case text section will be same but the data, heap and stack sections vary from process to process.

One other important points is that a process itself can be an execution environment for others. For example, an executable java program is executed in JVM. In this case, JVM executes a process that interprets the loaded java code and takes action according to that code. If we are compiling the java program in command line we would write java classname.class, so that java command run the JVM as an ordinary process and then executed that program in JVM.

The process is also know as job.

There are two types of processes with respect to the execution time

CPU BOUND PROCESS: The process which requires more CPU time is called a CPU bound process. This type of processes spend more time in running state. I will shortly explain you what the process state is?

I/O BOUND PROCESS: The process which requires more I/O time is called I/O bound process. These processes spend more time in waiting state.

**Attributes of a process:**

1. Process state: contains the current state information of the process where it is residing

2. Program counter: contains the address of the next instruction to be executed

3. CPU registers: contains registers information used by the process in order to execute the instructions. So this state information saved if interrupt occurs and help the process to continue from where it was interrupted.

4. CPU scheduling information: contains scheduling parameters such as process priority

5. Memory management information: contains information related to memory system used by the os

6. Accounting information: contains amount of CPU and real time used, time limits, process id and so on. Process id is a unique identification number which is assigned by the os at the time of process creation

7. I/O status information: includes list of file which are currently opened by the process and similarly list of devices used by the process

So all the attributes of the process is called as context of the process. The context of the process will be stored in PCB i.e Process control Block. Here you can see the representation of PCB. Every process will have its own PCB. PCB of the process will be stored in the main memory.

PROCESS STATES

First of all process is in the new state which means it is under creation. Once the process is created then it is moved to ready state. Ready state can have more than one process at a time. Furthermore one of the process from ready state is dispatched to CPU and change its state to running. Remember that there should be only one process in the running state because a CPU can only run 1 process at a time. So let assume that the process need I/O or some other event then it will perform the I/O request and moves to waiting state. Waiting state can also have more than 1 process. Now CPU is vacant which mean some other process from ready state can be dispatched to CPU. Lets say our process has finished its I/O request and need the CPU again. So for getting the CPU it again goes to ready state and wait for its turn. When this process again went through the running state and finish all of its instructions then it goes to terminated state. Let me explain one more point here, assume we have 3 processes in the ready sate and process 3 is dispatched to CPU. So now we cannot remove this process from the CPU. It will be in the CPU until it has finished all of the instructions. Of course it will leave CPU automatically if it need I/O. What I mean is we cannot call it back. This type of scheduling algorithm is called non-pre-emptive like FIFO. Lets take one more example, assume one process called p4 comes to ready state and it has very priority and we have to run it immediately. Then we will perform an interrupt which brings back the process 3 and dispatch the process 4. This type of scheduling is called pre -emptive. Other scenario can be a time slice. Eg: we gave 10 seconds to p4 but it cannot finished all of its instructions in that time then we will call it back and run some there process in CPU.

What is process schedular?

It selects a process from a set of available processes and send to it CPU for execution. In other words, process schedular is responsible for the transition of process which is currently residing in the ready state to running state. Process scheduling is a part of multiprogramming system. For example the batch system does not support multiprogramming which means if a process is running and suddenly leave the CPU because it needs an I/0 so in this case processor will not run any other program. CPU have to wait for the process to come back from I/O request to continue the execution. However, in the modern system things does not work like this. As I showed you in previous slide that when a process leave the CPU for any reason another process is dispatched to CPU because the objective of multiprogramming is to run some process all the times to get maximum CPU untilization. Process scheduling helps the modern systems to achieve this goal.  Other important point to know is in single processor system like your laptops no matter how many programs you are running on it, in reality only one process is running at a time which means all the other process have to wait until CPU is free and rescheduled again. However, this rescheduling happens so fast that it looks like all the processes are running at same time. In other case where you are not executing any program in your computer, the CPU is still executing the internal activities of OS.

what is scheduling queue?

What is schedulers?

A process goes through various queues throughout its lifetime. The operating system must select a process from these queues for scheduling purposes in some appropriate way or algorithm. This selection process is done by the scheduler.

LTS: LTS or job scheduler is responsible for creating new process and bringing them into the system. which means it create the process and brings it into ready state. As I mentioned earlier that process is also know as job that is why LTS is also know as job scheduler .

STS: STS or CPU scheduler is responsible for scheduling one of the process from ready state to running state. It allocate CPU to the process that is why it is called CPU scheduler.

MTS: is responsible for suspending and resuming the process. Basically if there are too many processes in the queue then it just swap the process from main memory to secondary memory. Once the queue is empty it brings back the process to the main memory. It is also known as swapper.

Lets fit these scheduler in our process state diagram.Bringing processes from new to ready state is the responsibility of LTS. Now we are in ready state which means there are lot of processes present in this queue. Here we have to select a process and dispatch to running state. This procedure is done by STS. ready, waiting and running state reside in the main memory. Here I will show you that how process can also reside in the secondary memory. Assume there are too many processes in the waiting queue and OS cannot hold it then it will suspend or swap the processes to suspended waiting state. This state reside in secondary memory. Once space is available in the queue, process swap back to main memory and waiting state. This procedure is done by MTS. Similarly, ready queue can have more process than it can handle, it also suspend to suspended ready queue and calls back when space is available. MTS is also responsible for this procedure.

Degree of multiprogramming:

The number of process presented in the main memory at any point of the time is called degree of programming. Lets see in the picture we have 4 processes in the memory so in this case degree of programming is 4 because of 4 processes in the memory. LTS controls the degree of multiprogramming because it bring processes from new to ready state. The LTS should select good combination of CPU bound and I/O bound process in order to get good throughput. For example, if LTS only brings CPU bound processes then no dough the CPU utilisation will be more but it will decrease the system throughput because no. of dispatched processes will decrease as other processes have to wait so long for their turn. On the other hand, if LTS will bring more I/O bound processes then CPU utilisation will decrease because most of the time processes will be in I/O wait state. So LTS must have to bring a mixture of both processes for better CPU performance.

Context Switching

Saving the context of 1 process and loading the context of other process is called context switching. In other words, We want to run process 1 but CPU is already running process 2, so we have to remove it for running process 1. Like I showed in process state diagram where we perform interrupt to run other process. Let me show you in diagram. Lets say p1 is running and we have to remove it for running other process. First we will save its context, remember context i told you in the first slide they are in PCB. Then we will load the p2. So this in between time is called context switching time. here you can see that p1 is in running state and it will go to ready state whereas p2 comes from ready to running state. Remember each time when process is moving from 1 state to other, the context of the process will change. We need minimum two process in order to perform context switching, one to remove another to load instead of first one. However it is possible to perform context switching with 1 process in Round Robin scheduling. If the context of the process is big then context switching will take more time which is undesirable that is why context switching is considered as overhead for the system as CPU is not running any process during context switching.