Operating System Fundamentals

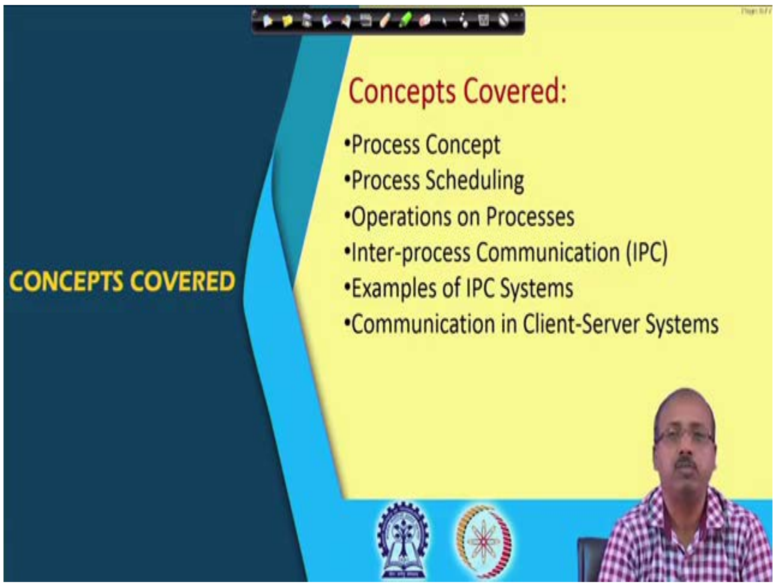
Prof. Santanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 12
Processes

Next we shall be looking into one of the very important concept in operating system design which is known as Processes. So, these are the fundamental blocks that we have in any operating system. So, the programs that are executing so, they are ultimately in terms of some processes. So, processes; the definition of a process is something like this a very working definition is a program that is under execution. So, whenever we are running a program so, from an operating system angle so, it is look as a process.

So, whenever we are talking about managing the user programs we are essentially talking about managing the processes that are running in the system. So, we try to differentiate between programs and processes like from a users perspective so we are writing one program. Now that single program it can have a single process in the system or it can have multiple processes in the system. Similarly, a number of programs developed by different users that may be combined into a single process so, both are possible. So, from the operating system angle so we will be looking into this executable units as processes.

(Refer Slide Time: 01:35)



CONCEPTS COVERED

Concepts Covered:

- Process Concept
- Process Scheduling
- Operations on Processes
- Inter-process Communication (IPC)
- Examples of IPC Systems
- Communication in Client-Server Systems

So, the topics that we are going to cover in this discussion is the concept of process then how the scheduling is done like there may be a number of processes in my system. So, which process will be executed text by the processor so, that is the scheduling one.

And similarly there may be multiple users so, maybe if we are following the policy of giving priority between the users; so, that maybe one possibility. Then we can also have some sort of round robin type of approach where we give some small time quantum to every user or every process so that is also a type of scheduling policies. In this we will see that there can be different types of scheduling policies.

Then what are the operations that we can do on processes. A typical situation; typical example is creating a new process like we may in a within a program may be we try to create another process which will be executing another program. So, that is a quite common like we may start for example, if we are doing some database operation may be a new request has come for get retrieving some data. So, for that we create a new process that retrieves that data. And in the meantime another request comes for retrieving another data so for that we may try to create another process. So, this way we can have different type of creates.

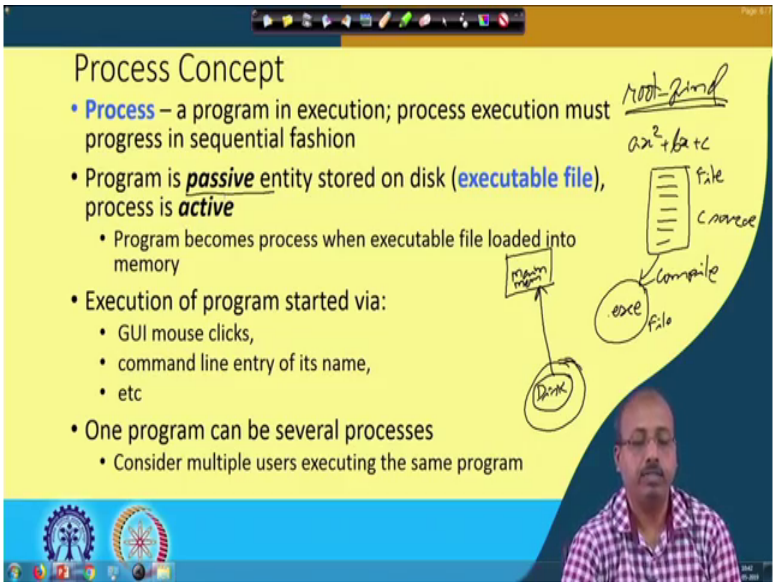
Similarly maybe there are some processes in the system which are ill-behaving or misbehaving like it may be using a lot of memory. So, the system administrator may decide that this process be terminated because otherwise the other processes in the system so, they are suffering so the system throughput is going down. So, that way they administrator may want to terminate that process or kill that process, the administrator may try to reduce the priority of a process. So, these are certain operations that we may like to do on some processes.

Another important point that we have is inter process communication. So, processes when they are executing in a system so, they are actually part of some job. So, as a result they may need to interact between themselves. So, this inter process communication is another very important issue. So, we will be looking into some examples of IPC systems, then communication in client server system. So, this is a very special type of computing platform which is common in data base then this web browsing and all.

So, where it is basically an event driven approach or event driven system like the server that we have the database server or the web server. So, it after it initialises so, it actually waits for some client request. For example, the database server will wait for some query and similarly the web browsers so, it will be waiting for some user asking to access a particular URL. So, like that so they will be waiting for that and when that request comes as a from a client process so it will initiate the corresponding action.

There may be multiple clients whose requests have arrived. So, as a result the server may want to do them parallely or sequentially depending upon the design. So, whatever it is so, this communication is very important in client server system. So, these are the concepts that we are going to cover.

(Refer Slide Time: 04:57)



Process Concept

- **Process** – a program in execution; process execution must progress in sequential fashion
- Program is ***passive*** entity stored on disk (**executable file**), process is ***active***
  - Program becomes process when executable file loaded into memory
- Execution of program started via:
  - GUI mouse clicks,
  - command line entry of its name,
  - etc
- One program can be several processes
  - Consider multiple users executing the same program

To start with we will try to define like what is a process? A process is a program in execution. So, this is a very working definition any program that is under execution we can call it a process and process execution must progress in sequential fashion. So, this is the basic assumption that any in a computer system so, any software that you have. So, that executes in a sequential fashion. So, one instruction is executed, then the next instruction is executed and like that it goes on. So, a process is a program in execution and its execution will go in a sequential fashion.

A program is a passive entity stored on disk and process is active. So, why do we say so, is like this. So, suppose we have got some program to calculate roots of a quadratic equation $a_x^2 + b_x + c$. Now, we have written a piece of program here. So, this is the c source file and then this c source file so, we compile; we compile this code and get the corresponding dot exe file executable version of the file. But this file that we have so, this is also nothing, but a file only so, this is a file and this is also a file. So, there is as such there is no difference excepting that this is an exe file, this is a binary files, apart from that there is not much difference.

So, this files so both this files so, they are existing; they will be existing in the disk and when this files are existing in the disk so, they are not going to be executed. For; now if the user wants that the corresponding program so, let us call the program name as root find. So, this root find program is called; so, then the first thing is that this from the disk this program has to be copied into the main memory. So, this is the main memory so, it has to be copied into main memory.

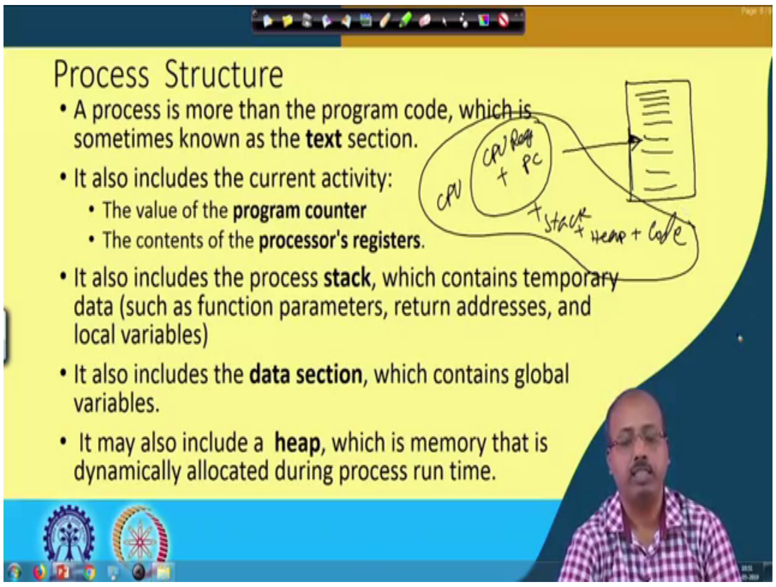
So, after it has copied into main memory so, then only the program can start executing. So, as long as it is residing in the disk so, we cannot do anything ok. So, this program is a passive entity that that is why it is called a passive entity that is stored in the disk in the form of an executable file, but the process when the program has been loaded into the main memory now the program can be executed. So, it can really find the roots of some quadratic equation now, given it will read the values of abc and find the roots.

So, this program becomes process when executable file is loaded into memory. So, this is the difference between programs and process; so, program is a passive entity, process is an active entity. So, execution of a program has started via GUI mouse clicks, where most of the operating systems now we have got this graphical user interface. So, it is using that interface or it may be it is combined line entry of it is name interface it is name so, this is also there. So, there are some combined line interfaces so, using that also the user may type the name of the program or process to be executed.

So, this is another possibility and there are certain programs that are automatically loaded by the system and they are executed. So, particularly the operating system processes so, that is also execution of some program. So, it is a inherently done by the operating system. So, execution may start via different means and one program can be of several processes. So, we have got multiple users executing the same program for example, there maybe multiple people doing the editing job. So, each student in a laboratory class maybe writing their own programs and all of them connected to a server and they are all using the editor software of the server.

So, as a result multiple users are executing the same editor program. So, that way one possibility is that if there are 10 users doing the editing job. So, we create 10 copies of that editor software on to the main memory, but that is definitely a big wastage of space. So, what is normally done is that this processes they are designed in such a fashion we have got only one piece of editor code available in the main memory, but there are 10 different data segments, each data segment holding the editing data of one user. So, that way we can have the say, multiple users executing the same program and one program can have can lead to several processes.

(Refer Slide Time: 09:44)

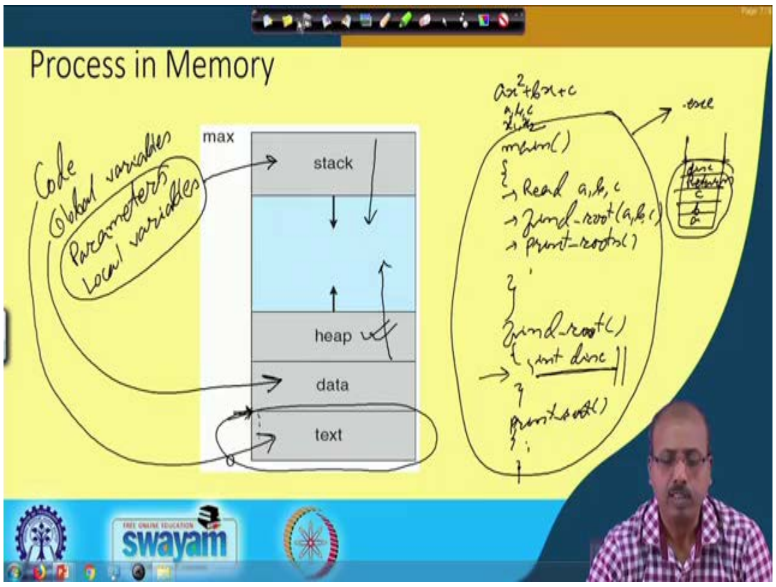


Process Structure

- A process is more than the program code, which is sometimes known as the **text** section.
- It also includes the current activity:
  - The value of the **program counter**
  - The contents of the **processor's registers**.
- It also includes the process **stack**, which contains temporary data (such as function parameters, return addresses, and local variables)
- It also includes the **data section**, which contains global variables.
- It may also include a **heap**, which is memory that is dynamically allocated during process run time.

The structure of a process a process is more than the program code which is sometimes known as the text part. So, it also includes the current activity which is the program counter and the contents of processor registers. Also includes process stack which contains temporary data and the data section which has got global variables and it has got some local variables in the some dynamically created local variables in the heap.

(Refer Slide Time: 10:15)



So, what we mean is like this; so, if you look into this diagram it explains it in a better fashion. Coming back to the same say example that is the program to find roots of that quadratic equation sorry, roots of that quadratic equation a x square plus b x plus c.

So, here in this while writing this program I may have a main routine. In the main routine I read the values of a b c; it reads the values of a b c then it calls a function called find root a b c and then it will be doing some printing print roots etcetera and then this find root function is somewhere here, then this print root is there. So, this may be the program is organised like this.

So, when you translate this program; when you translate to compile this program so, you get the corresponding dot exe file. Now when this program is executed there are certain parts in it so, one thing is that I should have this program lines. So, these all these lines that I should have the corresponding code for that plus I should have these variables abc ok; I should have these variables abc, I should have some corresponding memory location, similarly if the roots are say x 1 and x 2, then I should have some location for them.

And I should also have some so, this within this find root function I can define some local variable may be the discriminate and all. So, this is a local variable. So, we have got certain things one thing is the program code and one another part is the global variables ok, we have the global variables, then we have got the parameters that are passed for a function and the local variables. So, these are the components of a program when it is executing.

So if you look into the; if you take a snapshot of the memory where the program has been loaded and it is running. So, you can find that it has got some sections like this, one part of that memory it will contain the program code so that is the text part.

So the when the program counter is initialised to this and then it goes on executing line by line here depending upon if there is a fall to a functions. So it will call it and all that so, that is there, but it will be executing those routines those statements. Then the global variable that we have so, they will come to this data segment part. So, this memory as far as the process is concerned. So it can be think about to be consisting of different segments we have got code segment that holds the program text, we have got the data segment which has got the global variables.

Now this parameters and this local variable so, these two so, they are created in to the stack so, they go to the stack. So whenever this find root function is called so, in the in the stack we put the variable values a b and c and then it also saves the return address it also saves the return address and then the control branches to this routine and coming to this find the root in.

So, it will find that and other local variable at discriminant is there; so, this discriminant variable is also created in the stack so, this is also allocated space in the stack. Now when this part of the program is executing so, it will be using these variables that we have here ok. So, it will be utilising these variables.

So, that way the stack is there and sometimes in the program so, we can have some dynamic memory allocation. So, by say for example, in C program; in C language we have got the malloc system call. So like that it creates some dynamic space; so, those dynamic space so, they are located from this heap space ok. So, this stack part is for this local variables and parameters that we are passing and this heap is for the dynamics variables that are created so, this way it goes on.

So, this normally what is done is that this stack and heap so, these two instead of making them two separate segments. So, we take a big chunk of memory and from one end the stack starts to grow and from the other end the heap starts to grow. The idea is that some program may be using less of subroutine call so, sub program calls. So, the stack will be less, but it may utilising lot of dynamic memory. So, that heap space will be utilised more.

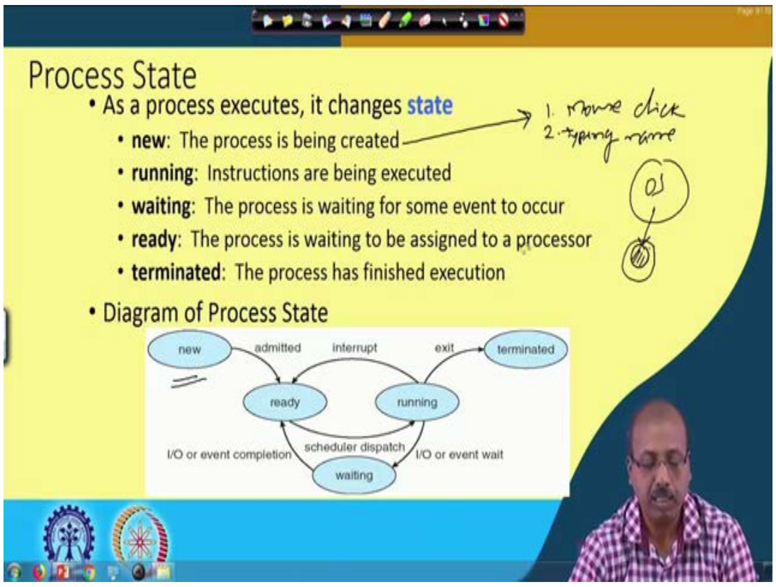
On the other hand some program may be using a less of dynamic memory, but it may have lot of recursive calls as a result it creates, it utilizes the stack a lot. So, to maximize the utilisation of this part of the space so, this stack and heap so, they are located from that ends of the same memory block.

So, ultimately so, depending upon your program that resides in the disks so, you have got only the compiled version of the text, but when you put into the memory and the program is executing the snapshot if you take so, that is quite different from what you have in the disk. So this is actually explained in the previous slide that we had. So, this is actually telling that a process is more than the program code so which is sometimes known as the text section.

So process includes the value of the program counter. So, whenever a program is executing suppose I have got a program so, like this so and it is executing line by line. So at suppose we are looking at this point. So, how do you identify this particular location in the program? One is the value of the program counter that is there plus there are a number of CPU registers that will be holding some of the temporary values for the variables. So this CPU registers so, they are also a part of the current activity.

So, apart from this CPU; as far as the CPU is concerned so, within the CPU you have got the program counter and this processor registers so, they are part of this process. So, apart from that we have got in the memory; we have got the stack of the process the data section the global variable and the heap so, these are part of the things. So, from for a process so, we can say that we have got this CPU registers plus the program counter this is coming from the CPU plus we have got the stack plus heap plus the code segment. So, this whole thing is defining the process when it is executing. So, that way the process structure is much different from the program structure.

(Refer Slide Time: 17:40)



Process State

- As a process executes, it changes **state**
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution
- Diagram of Process State

So, in terms of execution sequence when a process is going through different phases of it is evolution. So, we can say that it goes through a number of states. So, the initially when the process when the user says that I want to execute the program to compute the roots quadratic equation that program find root. So, that way the program the code of the program has to be copied from the disk into the main memory.

So when the process is being created when the user says that I want to execute this particular program. So how the user can say this, one possibility is by giving, some by clicking the some mouse. So one possibility is a mouse click. So, we can click the icon corresponding to a particular program on our desktop. So, that is by mouse click or it may be by means of some giving the name of the program so, typing the name of the program ok.

So, these are; these may be the two options by which you want to tell the system that I want to execute this. So, what the system does is that ultimately if this is the operating system, then it must create a new process which will be taking care of this particular user request. So when this process is just created so we have not yet marked it to a particular operation and all so that is the new state so, processor is just been created.

Then we have got the after that has to be done. So if this process has to execute that program for finding roots of quadratic equation, then the next step is to copy the program from the disk into the main memory. So for that we have to allocate some space in the main memory and copy the program from the secondary storage to the main memory. So after this copy has been done into the main memory now the program is ready for execution.

So, now if the process gets CPU so, it can do the execution. So, this particular transition is known as new to admitted and this transition happens when the program is when the program has been copied onto a main memory and it is ready for execution. Now, as soon as it is ready so it is not required that the process will start executing, because at present the processor or the CPU that we have, so that maybe busy doing something else. And so, this one it has to wait for the this particular routine or this particular program or process it has to wait for getting the CPU.

So after sometime the scheduler takes a decision that now this process should be executed. So when it happens the process makes a transition from ready state to the running state. So, in the running state so, process has got the CPU and it is executing it. Now while executing again there can be a different many different situations that can occur one possibility is that that for example the program that we looked into that we are considering finding roots of quadratic equation, the initially the program needs to read the values of a b and c.

Now reading the values of a b and c so, the user will give the values through keyboard so, it will take some time. So, the system that the CPU should not be wasting its time waiting for the user to enter the value of a b c which is users are much much slower compared to the electronic processors. So, what is done? The process whenever a process wants to do some input output operation or some event wait so, it will it is put on to a waiting state and the next ready process from the queue. So, it is taken by the scheduler and it is put on to the running state.

So, this scheduler it dispatches the next process from ready to running. So, CPU is busy doing the next, handling the next process and this process which was waiting. So, after sometime when the user has entered the values of this a b and c. So, it informs the system that this IO is over ok. So, whichever hardware module is doing this operation for example, if it is done by the keyboard then when the keyboard keys are placed. So, it sends an interrupt to the system and say that way when the interrupt is processed. So, after sometime these values of a b and c have been read and then this process from the waiting state it goes back to the ready state.

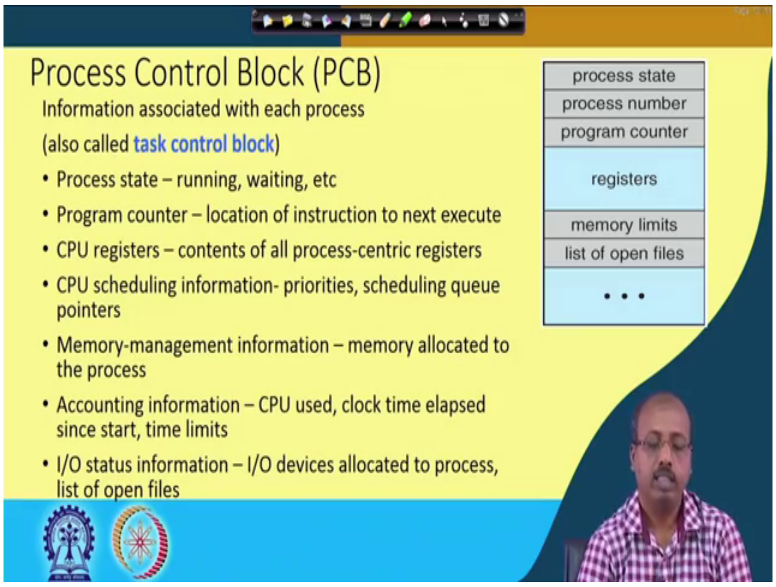
So, again after sometime the process will get chance for execution and it will continue from whichever point it has left it so that way we can have this ready running and waiting sort of thing. Another possibility is that when a process is executing so, it is given some small time quantum and after sometime the time quanta expires; so, when the time quanta expires so, it the processor has to be given to the next process in the queue.

So, the process which was executing here so, it has to be taken back to the ready state and the next process should get a chance. So, this way the interrupt will come and is ready process. So, it will be so, this process comes back to the ready state and the next process gets a chance. So, that way after so, if a process has got several chances for execution in the cyclic fashion then maybe it is over now. So, it will do an exit and the process goes to a state called terminated.

So, these are the different states that we have in the process. So, the new state is the process is the point at which the process is being created, we have got the running state where we have got this instructions are being executed, then this waiting; the process is waiting for some event to occur, then ready the process is waiting for to be assigned to a processor and the terminated when the process has finished execution.

So, we have to understand this state transition process behaviour of a processor very well, because any operating system that we design. So, the processes will be designed particularly following this type of state transitions. So, they have some operating system may have some additional states, some operating systems may not be having it; so, but overall the situation is like this that the process makes state transitions in this fashion.

(Refer Slide Time: 24:20)



Process Control Block (PCB)

Information associated with each process

(also called task control block)

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files

So, every process has got a control block associated with it which is known as the PCB or the Process Control Block. So, here we note down all the important information about the process. So, information associated with each process so, they are stored in this process control block or task control block.

So what are the things that we are remembering, one is the first and foremost is the process state ok. So which or what is the state of the process at present whether the process is running, waiting, ready, etcetera. Then the program counters so, it is the location of instruction to the next execute. So program counter at present what is the program counter value, may be the program that we are that that is executing has got total 1000 bites long. So, out of that at present where is the execution point so that is kept in the program counter.

Then we have got the CPU register values. So, all the CPU registers they will be holding some value at this point of time when the process is executing. So, that way it contains the CPU register value so, processor centric registers. So, there are some like in the CPU there can be different classes of registers, some of them are users user defined sorry some of them are general purpose registers, some are some special function registers. So, depending upon registers which come into the context of program execution so, they are called process centric registers. So, those registers will be copied on to this program control block.

Then we have got the CPU scheduling information like the priorities, then scheduling queue pointers etcetera. So, this scheduling information is necessary because the scheduler has to take the decision. In the memory management information then this accounting information and IO status information. So, memory management information is for memory allocated to the process, accounting information for the system usage and IO status information for these IO devices that we have in the system the open files etcetera. So, they are kept in the PCB.