LINUX KERNEL PROCESS SCHEDULING INTERVIEW QUESTIONS AND ANSWERS

PDF prepared by embeddedshiksha

Q.1 : What is Process Scheduling and why it's needed ?

As we discussed in last chapter about process , Linux has a complex operating system which can have thousand of processes available to run at a time . But we have limited CPUs or processor, and hence we have to choose among those runnable process which process is going to execute and which process is going to wait.

In simple words , process scheduling decides which process to run when and for what time . When allocated execution time elapsed for a running process , scheduler terminate that process and give chance to another process .

Process scheduling play an important role in overall system performance as schedular has to distribute CPU times to processes considering how important a process for system performance . If an important process had to wait for long to execute then it's obvious it will slow down system performance .

`Q.2 : What is scheduler .

Scheduler is an important component of our linux kernel which takes all the pain of scheduling of processes available in system . Scheduler has to take lot of critical decision for system by deciding which process to give chance to execute and which process has to wait . Schedular has to best utilise the CPU time to give good system performance .

Schedular works on scheduling policy which enable scheduler to take decision which process is going to run and which has to wait . We are going to talk lot more about scheduling policies going ahead in this chapter .

Q.3 : **What is scheduling policy**

Scheduling policy Is the set of rule which helps scheduler to decide which process to choose for running and which process to choose for waiting .

In linux we have difference scheduling algorithm available to schedule the process which has their own scheduling policy , we are going to talk about later in this chapter .

4**. What is input/output bound and Processor bound processes ?**

Input bound processes are those which spend most of the time waiting for some input/output event to happen . For example , any graphical user interface like text editor which has direct user interaction and will need processing only when user has some interaction with this text editor . Till that time this process will be blocked for user interaction .

On the other hand Processor bound processes are those who spend most of the time executing on processor , they has very minimal interaction with user Input/output event . For example some complex calculation programs which has to spend spend lot of CPU time for performing calculations .

Q.5: **What role input/output bound and Processor bound processes play in decisions for scheduling ?**

Processor bound process once scheduled will continue executing for quite sometime till the time another high priority task preempt it . On other hand Input/Output bound process will be scheduled frequently and will execute for small period of Time and then again will go waiting for another input/output event . As Input / Output bound process mostly has direct interaction with user hence it play very important role in system responsiveness , that's why linux kernel gives priority to Input/ Output Bound process over Processor bound process .

Q.6 What is preemption ?

The act of stopping currently running process from execution and giving chance to another high priority task for running is called preemption . For example a process A is currently executing and suddenly a process B become ready to run which has high priority than currently executing process A , Linux kernel will stop process A from executing and allow process B to execute . This process of stopping process A and giving chance to execute another process B is called Preemption .

Q.7 **What is Process Priority ?**

Process priority is the ranking given to process depending upon importance of process. Higher the priority means process is important and lower the priority means process is relatively less important . Process priority play an important role in process scheduling , using process priority linux kernel can take better decision and schedule high priority process on time which further improve overall system performance .

Q.8 Tell me about different types of process priority linux kernel uses.

Linux kernel mainly used two types of process priority

1) Real time process Priority
2) Nice Value

1) **Real time process priority** : Linux kernel define the process priority from range of 0 to 99 with default value of 0 for Real time process . High the real time priority value , higher the priority of that process in system and lower the priority means lower the importance of process in system .
2) **Nice Value :** Nice value is the priority range defined from -20 to +19 , its just opposite to Real time priority , higher the nice value means process is less important in system . Lower the nice value means process is process is relatively more important .

Q.9 when to use nice value and when to use real time priority in linux kernel ?

Nice value should be used for non critical process which doesn't play much role in system responsive , for example some background process may be flush process which doesn't have any user interaction .

On the other hand if your process is time critical and play important role in system responsiveness then I suggest you to go for real time priority .

Q.10 : What is timeslice .

Timeslice refers to time a process can execute before its termination . Once timeslice is exhausted , currently running process will be terminated and next process will get chance to execute .

Q. 11 What is consideration while choosing values of timeslice in any process scheduling algorithm ?

A scheduling algorithm has to choose timeslice in such a way to optimise system performance and to avoid any process to monopolise the whole system . A very long timeslice can cause system to behave unresponsive for sometime which will effect overall responsive ness of system . On the other hand if timeslice is too short then switching from one task to another task will cause low of context switching overhead . So scheduling algorithm has to choose timeslice very wisely to improve responsiveness of system and to fairly distribute CPU time to all the available processes .

Q.12 What are scheduler classes and what are different scheduling classes in linux kernel ?

Scheduling classes refers to different scheduling policy or algorithms used to manage the execution of processes In linux kernel . Each scheduler class has its own set of rules for deciding which process will run next and for how long .

Below are main schedular classes in linux kernel

1) Completely Fair Scheduler (CFS)
The completely fair scheduler is the default used scheduling algorithm used in Linux . It's aimed to provide fair distribution of processor time to each process . We will discuss CFS in detail later .

2) **Real time Scheduler :**
Real time Scheduler use real time scheduling policies like Round Robin and FIFO . This schedular class is used to schedule time critical process which execution can not be delayed .

Q.13 What is CFS scheduling in linux kernel ?

Complete far scheduling is the default scheduling policy in Linux kernel . As the name suggest complete fair scheduling ensure

runnable process in system get fair amount of CPU time for execution .

In CFS , Linux kernel instead of directly assigning time slice to a process basic on the nice value , it consider nice value to decide relative weightage of process as compare to another processes available in system . Higher the weightage means , process will get high proportion of CPU time and lower the weightage means low proportion of CPU time .

Lets assume we have two process in system with nice value of 0 and 5 . So relative weightage of nice value 5 process will be 1/3rd . If target latency of system is 20 milliseconds then process with 0 nice value will get 15 millisecond timeslice and other process with nice value will get 5 millisecond of timeslice .

Once timeslice is decided for each processes available in system , linux kernel scheduler has to decide which process will get chance to execute whenever scheduler has to schedule a new process .

CFS keeps the record of Virtual run time of each process . So whenever scheduler wants to schedule a new process it search for a process with lowest virtual run time . The process which has lowest virtual run time get chance to execute first .

Overall, CFS provides an efficient and fair scheduling policy that ensures that all processes get a fair share of CPU time, regardless of their priority or history of CPU usage.

Q.14 What is Virtual runtime ?

Virtual runtime denotes the time for which a process is actually executed till now . Virtual runtime play very important role in deciding which process to schedule next in CFS . CFS select the process which has shortest virtual runtime .

Q,15: How linux kernel handles sleeping off a process ?

Whenever a process wants to wait for some input output event to happen in future it put itself into wait queue and mark itself as non runnable . Now lets understand what is wait queue . Waitqueue is queue of processes which are waiting for some event to happen . When that event happen process waiting in that queue is wake up and moved to runnable state .

Q.16: How waking up from Waitqueue is handled in linux kernel ?

Whenever event for which the processes were waiting happen , linux kernel calls wake_up() for the queue and move all process waiting in that queue to TASK_RUNNING state . All the processes added back to red black tree as now all the processes are ready to execute . Depending upon the priority of processes linux kernel further schedule these processes .

Q.17: What is context switching and how it is handled?

Context switching refers to save the current state of currently executing process and switch execution to another process . To perform context switch below are the main steps kernel has to perform :

1) Save the context of currently executing process in PCB (Process control block). This include Program counter , CPU registers values , stack pointer etc .
2) Now as we have saved the context of previously executing process , kernel further select new process for execution as per scheduling algorithm used and load the context of newly selected process .
3) Once software context of new process is set , its time for actual hardware context switch where hardware will save current

context of process and switch to execute newly selected process .

4) Once context switching is done , new process starts executing .