MultiProcessing

1. Where are the function arguments and variables stored?

**Ans: Local variables and function parameters are usually stored on the thread's stack. Arguments beyond four are stored on the stack as though they had been pushed onto the stack in right-to-left order.**

1. Where are global variables stored?

**Ans: Global variables are stored in the data section**

1. What are the resources assigned to a process?

**Ans: The information of processes (waiting to run, sleeping, runnable processes, etc.), memory (virtual memory information such as free, used, etc.), swap area, IO devices, system information (number of interrupts, context switches) and CPU (user, system and idle time).**

1. How are processes identified?

**Ans: Each process is named by a process ID number, a value of type pid\_t. A process ID is allocated to each process when it is created. Process IDs are reused over time. The lifetime of a process ends when the parent process of the corresponding process waits on the process ID after the process has terminated.**

**In the GNU C Library implementation running on Linux, the process ID is the thread group ID of all threads in the process. You can get the process ID of a process by calling getpid . The function getppid returns the process ID of the parent of the current process (this is also known as the parent process ID).**

1. Who selects the process for execution?

**Ans: CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.**

1. What are the guiding principles used by scheduler to select a process?

**Ans:**

**Fairness** − All processes should be treated the same. No process should suffer indefinite postponement.

**Maximize throughput** − Attain maximum throughput. The largest possible number of processes per unit time should be serviced.

**Predictability** − A given job should run in about the same predictable amount of time and at about the same cost irrespective of the load on the system.

**Maximum resource usage** − The system resources should be kept busy. Indefinite postponement should be avoided by enforcing priorities.

**Controlled Time** − There should be control over the different times −

* 1. Response time
  2. Turnaround time
  3. Waiting time

1. List atleast 5 scheduling algorithms

**Ans:**

**1) First Come First Serve (FCFS),**

**2) Shortest-Job-First (SJF) Scheduling,**

**3) Shortest Remaining Time,**

**4) Priority Scheduling,**

**5) Round Robin Scheduling,**

**6) Multilevel Queue Scheduling**

1. What do you mean by single and multi core?

**Ans: A single-core processor is a microprocessor with a single core on its die. It performs the fetch-decode-execute cycle once per clock-cycle, as it only runs on one thread. A computer using a single core CPU is generally slower than a multi-core system.**

**A multicore processor is an integrated circuit that has two or more processor cores attached for enhanced performance and reduced power consumption.**

1. How many processes can a N core CPU run parallely?

**Ans: There must be more than one processing core to execute two processes in parallel.**

1. How is a program executed internally? What are the steps involved?

**Ans:**

**The following steps are involved in the execution of a program:  
-Fetch: The control unit is given an instruction.  
-Decode: The control unit then decodes the newly received instruction.  
-Execute: During the execution the Control unit first commands the correct part of hardware to take action. Once that is found out the control is handed over to the hardware. Now the task is performed.  
-Store: Once the task is saved successfully the end result is stored.  
 After the cycle is complete the Control unit is again handled the control.**

1. What are the various attributes of a process? Mention atleast one command to view process attributes

**Ans:**

**1.Process ID**

**2.Program counter.**

**3.Process State.**

**4.Priority.**

**5.General Purpose Registers.**

**6.List of open files.**

**7.List of open devices.**

1. What are the different states of a process?

**Ans:**

**1.New.**

**2.Ready**

**3.Ready Suspended.**

**4.Running.**

**5.Blocked.**

**6.Blocked Suspended.**

**7.Terminated.**

1. How do we run multiple processes using a single CPU?

**Ans: Single CPU systems use scheduling and can achieve multi-tasking because the time of the processor is time-shared by several processes so allowing each process to advance in parallel. So a process runs for some time and another waiting gets a turn.**

1. What do you mean context switch? When does it happen?

**Ans: A context switching is a process that involves switching of the CPU from one process or task to another. In this phenomenon, the execution of the process that is present in the running state is suspended by the kernel and another process that is present in the ready state is executed by the CPU.**

1. What does the term concurrency and parallelism mean?

**Ans: Concurrency is the task of running and managing the multiple computations at the same time.** **While parallelism is the task of running multiple computations simultaneously**.

1. Why do we need to assign priorities to processes?

**Ans: Priorities should be assigned to the processes so that the tasks that needs immediate attention can be given higher priority and executed first and then the rest.**

1. Which command is used to view process status in realtime?

**Ans: ps command**

1. Which command is used to view process tree with pid details?

**Ans: ps -ef command**

1. Which command is used to get pid, ppid and process group id?

**Ans: ps -el command**

1. Which process starts all processes in the system?

**Ans: Init process is the mother (parent) of all processes on the system, it's the first program that is executed when the Linux system boots up**

1. How to create a new process from within a program?

**Ans: A new process can be created by the fork() system call**

1. Where the process information maintained? What is the name of the data structure used to hold process information?

**Ans: The process control block is kept in a memory area that is protected from the normal user access. The symbol table is a data structure that is used to hold information about source code during the compilation process.**

1. What happens on exit()?

**Ans: exit() In C, exit() terminates the calling process without executing the rest code which is after the exit() function.**

1. What is the difference between exit() and \_exit()? Which will cause quick exit?

**Ans: exit() and \_Exit() in C/C++ are very similar in functionality. However, there is one difference between exit() and \_Exit() and it is that exit() function performs some cleaning before termination of the program like connection termination, buffer flushes etc.**

1. Does \_exit close open fds?

**Ans: \_exit() does close open file descriptors**, **and this may cause an unknown delay, waiting for pending output to finish.**

1. Does \_exit flush open streams?

**Ans: Yes,  \_exit() does flush open streams**

1. What happens when you press Ctrl+C?

**Ans: It is used to kill the process.**

1. What happens when you press Ctrl+Z?

**Ans: It is used to pause the process. It will not terminate the program, it will keep your program in background.**

1. What is the use of an fd? How is it different from FILE \*?

**Ans: A file descriptor (FD) is a small non-negative integer that helps in identifying an open file within a process while using input/output resources like network sockets or pipes.**

**File descriptor is an int whereas a FILE \* is a file pointer. The main difference is that the latter is buffered while the former is not. A file pointer ( FILE\* ) typically contains more information about the stream such as current location, end of file marker, errors on the stream etc.**

1. How many fd’s are created for every process? What are they?

**Ans: Linux systems limit the number of file descriptors that any one process may open to 1024 per process.**

1. Name the call to get an fd for a file

**Ans: int fd = fileno(file);**

1. If a process creates a child sub process, how can it detect exit of a child?

**Ans: To get the exit status of the child via the first argument of wait() , or the second argument of waitpid() , and then using the macros WIFEXITED and WEXITSTATUS with it. waitpid() will block until the process with the supplied process ID exits.**

1. Which process reaps the exit code of orphan child?

**Ans: Parent process.**

1. What all does a child inherit from its parent?

**Ans: A child class inherits its parent's fields and methods,**