MultiProcessing

1. Where are the function arguments and variables stored?

Parameter values to functions are stored **on the stack** as well, pushed immediately before the return address. Everything what lives on the stack (local variables, parameters etc.)

1. Where are global variables stored?

Global variables are stored **in the data section**. Unlike the stack, the data region does not grow or shrink — storage space for globals persists for the entire run of the program.

1. What are the resources assigned to a process?

The process needs certain resources such as **CPU and memory** to perform the tasks

1. How are processes identified?
2. Who selects the process for execution?

**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?

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

1. List atleast 5 scheduling algorithms

* irst-Come, First-Served (FCFS) Scheduling.
* Shortest-Job-Next (SJN) Scheduling.
* Priority Scheduling.
* Shortest Remaining Time.
* Round Robin(RR) Scheduling.

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

A processor that has more than one core is called Multicore Processor while one with single core is called Unicore Processor or Uniprocessor

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

You must have more than one processing core to execute **two processes** in parallel.

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

Loader loads the executable module to the main memory for execution. Linker takes the object code generated by an assembler, as input. Loader takes executable module generated by a linker as input. Linker combines all the object modules of a source code to generate an executable module.

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

Running − In this state the instructions are being executed. Waiting − The process is in waiting state until an event occurs like I/O operation completion or receiving a signal. Ready − The process is waiting to be assigned to a processor. Terminated − the process has finished execution

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

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?

Context Switching **involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier**. This is a feature of a multitasking operating system and allows a single CPU to be shared by multiple processes.

1. What does the term concurrency and parallelism mean?

Concurrency means multiple tasks which start, run, and complete in overlapping time periods, in no specific order. Parallelism is when multiple tasks OR several parts of a unique task literally run at the same time, e.g. on a multi-core processor.

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

Every process requires a certain amount of system resources, like central processing unit (CPU) time and random access memory (RAM), to be able to perform its tasks. Each process is assigned a process priority, which **determines how much CPU or processor time is allocated to it for execution**.

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

The **ps (process statuses)** command produces a snapshot of all running processes.

1. Which command is used to view process tree with pid details?
2. Which command is used to get pid, ppid and process group id?

**Type the simply “pstree” command with the “-p” option in the terminal** to check how it displays all running parent processes along with their child processes and respective PIDs. It shows the parent ID along with the child processes IDs.

1. Which process starts all processes in the system?

**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; it manages all other processes on the system. It is started by the kernel itself, so in principle it does not have a parent process. The init process always has process ID of 1.

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

A new process can be created **by the fork() system call**. The new process consists of a copy of the address space of the original process. fork() creates new process from existing process. Existing process is called the parent process and the process is created newly is called child process

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

A **process control block (PCB)** is a data structure used by computer operating systems to store all the information about a process.

1. What happens on exit()?

On many computer operating systems, **a computer process terminates its execution** by making an exit system call. More generally, an exit in a multithreading environment means that a thread of execution has stopped running. For resource management, the operating system reclaims resources (memory, files, etc.)

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

The key difference between exit() and \_exit() function lies in the fact that **exit() function performs some cleanup tasks before terminating the program**. These include clearing the buffer, terminating the connection, etc. On the other hand \_exit() does not perform any such cleaning operation.

1. Does \_exit close open fds?
2. Does \_exit flush open streams?

The exit() function shall then **flush all open streams with unwritten buffered data**, close all open streams, and remove all files created by tmpfile().

1. What happens when you press Ctrl+C?

Turned out the way Ctrl-c works is quite simple — it's just a shortcut key for **sending the interrupt (terminate) signal SIGINT to the current process running in the foreground**. Once the process gets that signal, it's terminating itself and returns the user to the shell prompt

1. What happens when you press Ctrl+Z?

ctrl+z **stops the process and returns you to the current shell**. You can now type fg to continue process, or type bg to continue the process in the background

1. What is the use of an fd? How is it different from FILE \*?
2. How many fd’s are created for every process? What are they?

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

Get the file descriptor from a FILE pointer (e.g. file ) in C on Linux: **int fd = fileno(file);** More details can be found in the man page of fileno : fileno manual

1. If a process creates a child sub process, how can it detect exit of a child?
2. Which process reaps the exit code of orphan child?

The act of reading that exit code is called "**reaping**" the child. Between the time a child exits and is reaped, it is called a zombie.

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

A child process inherits most of its attributes, such as **file descriptors**, from its parent. In Unix, a child process is typically created as a copy of the parent, using the fork system call. The child process can then overlay itself with a different program (using exec) as required.