Multiprocessing

1. **Where are the function arguments and variables stored?**
    - Function arguments are stored onto the stack.
    - Variables are usually stored in RAM, either on heap or on the stack.

2. **Where are global variables stored?**
    - Global variables stored in the data section, that is separate from both heap and stack.

3. **What are the resources assigned to a process?**
    - A process will need certain resources such as CPU time, memory, files, I/O devices, etc., to accomplish its task.
    - These resources are given to the process when it is created.

4. **How are processes identified?**
    - Each process is identified with a unique positive integer called process ID or PID (Process Identification number).
    - Each process is guaranteed a unique PID, which is always a non-negative integer.

5. **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.

6. **What are the guiding principles used by scheduler to select a process?**
    - Fairness.
    - Maximize throughput.
    - Predictability.
    - Maximum resource usage.
    - Controlled Time.

7. **List atleast 5 scheduling algorithms.**
    - First-Come, First-Served (FCFS) Scheduling.
    - Shortest-Job-Next (SJN) Scheduling.
    - Priority Scheduling.
    - Shortest Remaining Time.
    - Multiple-Level Queues Scheduling.

8. **What do you mean by single and multi core?**
    - **Single core:** 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.
    - **Multi core:** A multicore processor has two or more processor cores attached for enhanced performance and reduced power consumption. These processors also

enable more efficient simultaneous processing of multiple tasks, such as with parallel processing and multithreading.

9. **How many processes can a N core CPU run parallely?**
   - N number of processes can run in N core CPU parallely.
   - For example, Today's multicore processors can easily include 12, 24 or even more processor cores.

10. **How is a program executed internally? What are the steps involved?**
    - When the CPU executes a program, that program is stored in the computer's main memory and executed.
    - **Steps:**

      Preprocessor:

      In this stage, the program processes before compilation. Preprocessors include header files and macros.

      Compiler:

      In this stage, the compiler will compile the program, check the errors and generate the object file.

      Linker:

      In this stage, linker links the more than one object files or libraries and generates the executable file.

      Loader:

      In this stage, the loader loads the executable file into the main/primary memory and the program will run.

11. **What are the various attributes of a process? Mention atleast one command to view process attributes.**
    - **Attributes of a Process:**

      Process ID.
      Program Counter.
      Process State.
      Priority.
      General Purpose Register.
      List of Open Files.
      List of Open Devices.
    - **ps command** : to view process attributes.

12. **What are the different states of a process?**
    - New.
    - Running.
    - Waiting.
    - Ready.
    - Terminated.

13. **How do we run multiple processes using a single CPU?**
    - A single core CPU can run two or more threads simultaneously.
    - These threads may belong to the one program, or they may belong to different programs and thus processes.

- The processor can switch execution resources between threads, resulting in concurrent execution.

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

- A context switch is the switching of the CPU from one process or thread to another.
- A context switch occurs when the kernel transfers control of the CPU from an executing process to another that is ready to run.

15. What does the term concurrency and parallelism mean?

- **Concurrency:** It is the execution of multiple instruction sequences at the same time. It happens in the operating system when there are several process threads running in parallel.
- **Parallelism:** It is related to an application where tasks are divided into smaller sub-tasks that are processed seemingly simultaneously or parallel.

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

- Assigning priorities to the process is necessary in order to complete everything that needs to be done.
- Prioritisation is important because it will allow you to give your attention to tasks that are important and urgent so that you can later focus on lower priority tasks.

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

- ps command is used to view process status in realtime.
- ps : process status.

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

- pstree command is used to view process tree with pid details.

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

- ps xao pid,ppid,pgid command is used to get pid, ppid and process group id.

20. Which process starts all processes in the system?

- Init process is parent of all processes on the system.
- It is the first program that is executed when the Linux system boots up.
- It manages all other processes on the system.

21. 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 created newly is called the child process.

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

- The process information is maintained in the process control block where it is kept in a memory area that is protected from normal user access.

- The symbol table is a data structure that is used to hold information about source code during the compilation process.

23. What happens on exit()?

- The exit() function is used to terminate a process or function calling immediately in the program.

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

- **exit()** is used to terminate the calling function immediately without executing further processes. As function calls, it terminates processes.
- **_exit()** is used to terminate the process normally and it returns the control to the host environment. It does not perform any cleanup task.

25. Does _exit close open fds?

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

26. Does _exit flush open streams?

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

27. What happens when you press Ctrl+C?

- Ctrl + C terminates SIGINT event.
- Once the process gets that signal, it's terminating itself and returns the user to the shell prompt.

28. What happens when you press Ctrl+Z?

- Ctrl + Z stops SIGTSTP. (Terminal Stop Signal).
- By default, this causes the process to suspend execution.

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

- fd stands for file descriptor. It is generally used for the application that do frequently random access of file.
- fd is integer whereas a FILE* is a file pointer.
- fd is not buffered, where, FILE* is buffered.

30. How many fd's are created for every process? What are they?

- fd's are generally unique to each process, but they can be shared by child processes created with a fork().
- fd's are,
    - stdin
    - stdout
    - stderr

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

- lsof is used to get an fd for a file.

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

- We can 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 exit.

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

- When a child exits, some process must wait on it to get its exit code.
- That exit code is stored in the process table until this happens.
- 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 process.

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

- From parent, a child inherits,
    - Real and effective user and group IDs.
    - Process group ID.
    - Current working directory.
    - File descriptors.
    - Attached shared memory segments.
    - Signal handling settings.

Assignment No: 01
Done By: Rajesh Kannan M
Employee ID: 46252660