

MultiProcessing Assignment1 :-

Answers :-

- 1) Local variables and function parameters are usually stored on the thread's stack. Some optimizers will put a few of them in CPU registers, but it's hard to control that. Global variables and static variables are stored in the process's static data or BSS segment.
- 2) **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. Finally, the heap portion of memory is the part of a program's address space associated with dynamic memory allocation.
- 3) The process needs certain resources such as CPU and memory to perform the tasks. Also there are certain limits by default for each process on the resources, and if required the limits can be enhanced to accommodate the application requirements.
- 4) Process identification is a set of activities aiming to systematically define the set of business processes of a company and establish clear criteria for prioritizing them. The output of process identification is a process architecture, which represents the business processes and their interrelations.
- 5) Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions. Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

6) The objective/principle which should be kept in view while selecting a scheduling processes are the following –

- a. **Fairness** – All processes should be treated the same. No process should suffer indefinite postponement.
- b. **Maximize throughput** – Attain maximum throughput. The largest possible number of processes per unit time should be serviced.
- c. **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.
- d. **Maximum resource usage** – The system resources should be kept busy. Indefinite postponement should be avoided by enforcing priorities.
- e. **Controlled Time** – There should be control over the different times.
 - (a) Response time
 - (b) Turnaround time
 - (c) Waiting time

7) **CPU Scheduling** is a process of determining which process will own CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution. The selection process will be carried out by the CPU scheduler. It selects one of the processes in memory that are ready for execution.

Types of CPU scheduling Algorithm :-

1. First Come First Serve (FCFS)
2. Shortest-Job-First (SJF) Scheduling
3. Shortest Remaining Time
4. Priority Scheduling
5. Multilevel Queue Scheduling

8) **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 is an integrated circuit that 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) Multiple processes can run simultaneously (without context-switching) in multi-core processors. If all processes are single threaded as you ask then two processes can run simultaneously in a dual core processor.

10)

Step 1: Fetch instruction. Execution cycle starts with fetching instruction from main memory. ...

Step 2: Decode instruction. ...

Step 3: Perform ALU operation. ...

Step 4: Access memory. ...

Step 5: Update Register File. ...

Step 6: Update the PC (Program Counter)

- 11) Attributes of a process are :-

- Process ID
- Program counter
- Process state
- Priority
- General Purpose Registers
- List of open files
- List of open devices

“ps” command is used to show some attributes of a process. This command reads through the kernel's data structures and process tables to fetch the characteristics of a process.

 nikki@INLEN6239004542: ~

```
nikki@INLEN6239004542:~$ ps
  PID TTY          TIME CMD
  548 tty1        00:00:01 bash
   807 tty1        00:00:00 ps
nikki@INLEN6239004542:~$
```

12) The five states that are being used in this process model are:

- (a) **Running:** It means a process that is currently being executed. Assuming that there is only a single processor in the below execution process, so there will be at most one processor at a time that can be running in the state.
- (b) **Ready:** It means a process that is prepared to execute when given the opportunity by the OS.
- (c) **Blocked/Waiting:** It means that a process cannot continue executing until some event occurs like for example, the completion of an input-output operation.
- (d) **New:** It means a new process that has been created but has not yet been admitted by the OS for its execution. A new process is not loaded into the main memory, but its process control block (PCB) has been created.
- (e) **Exit/Terminate:** A process or job that has been released by the OS, either because it is completed or is aborted for some issue.

13) **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.

14) **A context switch** occurs when a computer's CPU switches from one process or thread to a different process or thread. Context switching allows for one CPU to handle numerous processes or threads without the need for additional processors.

15) **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.

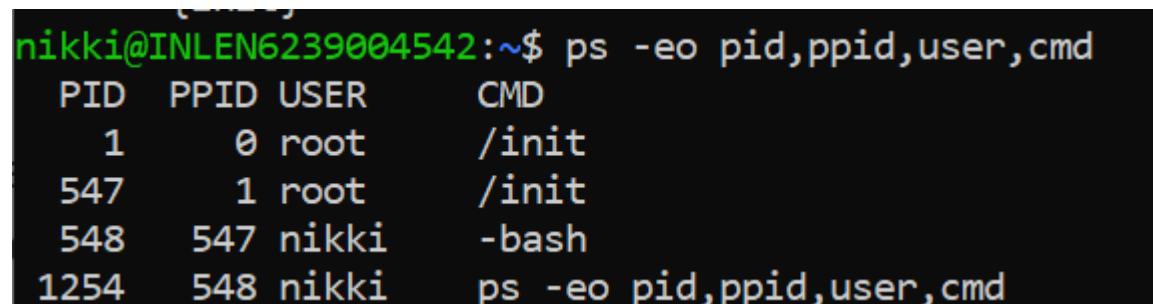
e.g:- On a multi-core processor.

16) Establishing priorities is necessary in order to complete everything that needs to be done. Prioritization 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. A process is the running instance of a program. Each process is assigned a process priority, which determines how much CPU or processor time is allocated to it for execution. There are two types of process priorities: the nice value and real-time priority.

17) The “**ps**” command has several flags that enable you to specify which processes to list and what information to display about each process.

18) “**pstree**” is a Linux command that shows the running processes as a tree. It is used as a more visual alternative to the ps command. The root of the tree is either init or the process with the given pid. It can also be installed in other Unix systems.

19) “\$ **ps -eo pid,ppid,user,cmd**”



```
nikki@INLEN6239004542:~$ ps -eo pid,ppid,user,cmd
  PID  PPID  USER    CMD
    1     0  root    /init
   547     1  root    /init
   548   547  nikki   -bash
  1254   548  nikki   ps -eo pid,ppid,user,cmd
```

20) Whenever a command is issued in Unix/Linux, it creates/starts a new process. For example, pwd when issued which is used to list the current directory location the user is in, a process starts. Through a 5 digit ID number Unix/Linux keeps an account of the processes, this number is called process ID or PID.

21) 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.

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

23) 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.)

24) Difference between `exit()` and `_exit()` function :- `exit()` does cleanup work like closing file descriptor, file stream and so on, while `_exit()` does not.

`_exit()` won't flushes the stdio buffer while `exit()` flushes the stdio buffer prior to exit. `_exit()` can not perform clean-up process while `exit()` can be registered with some function (i.e `on_exit` or `at_exit`) to perform some clean-up process if anything is required before existing the program.

25) `_exit()` does close open file descriptors, and this may cause an unknown delay, waiting for pending output to finish. If the delay is undesired, it may be useful to call functions like `tcflush(3)` before calling `_exit()`.

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

27) Control+C is a common computer command. It is generated by pressing the C key while holding down the Ctrl key on most computer keyboards. In graphical user interface environments that use the control key to control the active program, control+C is often used to copy highlighted text to the clipboard.

28) To reverse your last action, press CTRL+Z. You can reverse more than one action. To reverse your last Undo, press CTRL+Y. You can reverse more than one action that has been undone.

29) A **file descriptor** is a number that uniquely identifies an open file in a computer's operating system. It describes a data resource, and how that resource may be accessed. When a program asks to open a file — or another data resource, like a network socket — the kernel: Grants access.

A **FILE pointer** is a C standard library-level construct, used to represent a file. The FILE wraps the file descriptor, and adds buffering and other features to make I/O easier.

30) Linux systems limit the number of file descriptors that any one process may open to **1024 per process**. (This condition is not a problem on Solaris machines, x86, x64, or SPARC). After the directory server has exceeded the file descriptor limit of 1024 per process, any new process and worker threads will be blocked.

31) 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 .

32) You 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 exits.

33) The parent process reads the exit status of the child process which reaps off the child process entry from the process table. **Orphan Process:** A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an orphan process. An orphan process is a running process whose parent process has finished or terminated. In a Unix-like operating system any orphaned process will be immediately adopted by the special init system process. This operation is called re-parenting and occurs automatically.

34) Following are the things that a derived class inherits from its parent.

- 1) Every data member that is defined in the parent class (although such members may not always be accessible in the derived class!).
- 2) Every ordinary member function of the parent class (although such members may not always be accessible in the derived class!).
- 3) The same initial data layout as of the base class.