MultiProcessing :

1] Ans: 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.)

2] Ans: 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] Ans: It reports 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).

4] Ans: Process identification refers to those management activities that aim to **systematically define the set of business processes of an organization and establish clear criteria for selecting specific processes for improvement.**

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

6] Ans: The objective/principle which should be kept in view while selecting a scheduling processes are the following −

* 1. **Fairness −** All processes should be treated the same. No process should suffer indefinite postponement.
  2. **Maximize throughput −** Attain maximum throughput. The largest possible number of processes per unit time should be serviced.
  3. **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.
  4. **Maximum resource usage −** The system resources should be kept busy. Indefinite postponement should be avoided by enforcing priorities.
  5. **Controlled Time −** There should be control over the different times.
     1. Response time
     2. Turnaround time
     3. Waiting time

7] Ans: First-Come, First-Served (FCFS) Scheduling.

* + Shortest-Job-Next (SJN) Scheduling.
  + Priority Scheduling.
  + Shortest Remaining Time.
  + Round Robin(RR) Scheduling.
  + Multiple-Level Queues Scheduling.

8] Ans: A processor that has more than one core is called Multicore Processor while one with single core is called Unicore Processor or Uniprocessor.

9] Ans: You must have more than one processing core to execute **two processes** in parallel. Erlang is built for concurrency and will run concurrent solutions (even with a single CPU). Given multiple execution cores, it can also execute processes in parallel.

10] Ans: **The preprocessor generates an expanded source code.**

- Expanded source code is sent to compiler which compiles the code and converts it into assembly code.

- The assembly code is sent to assembler which assembles the code and converts it into object code.

11] Ans: Attributes of a Process:

-> PID or process ID, an integer.

-> PPID or parent process ID, an integer

-> TTY, the terminal to which the process is connected.

->RUID , or real user ID.

-ps

12] Ans: Different States of the process:

- New ->This is the state when the process has just been created.

- Running ->Instructions are being executed.

- Waiting ->The process is waiting for some event to occur.

- Ready ->The process is waiting to be assigned to a processor.

- Termination ->The process has finished execution.

13] 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.

14] Ans: **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] Ans: **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] Ans: Establishing priorities is necessary in order to complete everything that needs to be done. Prioritization is important because it with 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] Ans: The “**ps”** command has several flags that enable you to specify which processes to list and what information to display about each process.

18] Ans:“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] Ans:“$ ps -eo pid,ppid,user,cmd”

20] Ans: 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 call process ID or PID.

21] Ans: 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] Ans: **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] Ans: 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] Ans:

* 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. It simply terminates the program.
* \_Exit() function does not make any call to object destructors or the functions registered with atexit() or at\_quick\_exit() functions.

25] Ans: \_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] Ans: 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] Ans: 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] Ans: 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] Ans: **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] Ans: 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] Ans:  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] Ans: 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] Ans: 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] Ans: 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.