Process Management in UNIX Questions and Answers

1.(a) How will you run a process in background? How will you bring that into foreground and how will you kill that process?

For running a process in background in command line use "&". For bringing it back in foreground command line use "fg jobid" and for getting job id we use command "jobs", for killing that process find PID and we use kill -9 PID command.

(b) How do you know if a remote host is alive or not?

We can check these by using either telnet or ping command in UNIX.

(c) How do you see command line history in UNIX?

We use history command along with grep command in unix to find any relevant command we have already executed.

(d) How do you copy file from one host to other?

By using "scp" command we can copy file from one host to other. we can also use rsync command .

(e) How do you find which process is taking how much CPU?

By using "top" command in UNIX we can find process is taking how much CPU

(f How do you check how much space left in current drive ?

By using "df" command in UNIX we can check how much space left in current drive

(g) How can we get and set an environment variable from a program?

- By using "getenv()"getting the value of an environment variable is done.
- By using "putenv()"Setting the value of an environment variable is done.

(h) What is "ps" command for?

The "ps" commandis used for printing the process status for some or all of the running processes. The information given are the process identification number (PID),the amount of time that the process has taken to execute so far etc.

(i) How would you kill a process?

The "kill" command takes the PID as one argument; This identifies which process has to terminate. The PID of a process can be got using "ps" command.

(j) What is an advantage of executing a process in background?

The most common reason is to put a process in the background to allow you to do some thing else interactively without waiting for the process to complete. At last of the command you add the special background symbol, &. This symbol will tell your shell to execute the given command in the background.

2. Discuss about the initial process sequence while the system boots up.

When the system boots up, special process 'swapper' or 'scheduler' is created with Process ID 0. The swapper usually manages memory allocation for processes and influences CPU allocation. The swapper creates 3 children:

- the process dispatcher,
- vhand
- dbflush

with IDs 1,2 and 3 respectively.

3. Give a brief idea about various IDs associated with a process?

Unix identifies each process with a unique integer called ProcessID. The process that executes the request for creation of a process is called the 'parent process' whose PID is 'Parent Process ID'. Each and every process is linked with a particular user called the 'owner' who has privileges over the process. The identification for the user is 'UserID'. Owner is the user who executes the process. Process also has 'Effective User ID' which determines the access privileges for accessing resources like files.

4. Write the system calls and its description used for process management:

- fork()   - Create a new process.
- exec()   - Execute a new program in a process.
- wait()   - Wait until a created process completes its execution.
- exit()   - Exit from a process execution.
- getpid()   - Get a process identifier of the current process.
- getppid() - Get parent process identifier.
- nice()   - Bias the existing priority of a process.
- brk()   - Increase/decrease the data segment size of a process.

5. write a short note on Daemon?

It is a process which detaches itself from the terminal and runs, disconnected, in the background, waiting for requests and responding to them. It can also be said as the background process that does not belong to a terminal session. Deamons performs many system functions, including the send mail daemon, which handles mail, the NNTP daemon, which usually handles USENET news. Many other daemons exists.

Some of the most common daemons are:

- init: init takes over the basic running of the system when kernel has finished the boot process.
- inetd: inetd is usually responsible for initiating network services that do not have their own stand-alone daemons. For example, inetd usually takes care of incoming rlogin, telnet, and ftp connections.
- cron: cron is responsible for running repetitive tasks on a regular schedule.

6. What is IPC? What are the various schemes available?

The term IPC describes various ways by which different process running on some operating system communicate between each other. Various schemes available are as follows:

- Pipes: It is the one-way communication scheme through which different process can communicate. The problem is that the two processes should have a common ancestor. However this problem was fixed with the introduction of named-pipes (FIFO).
- Message Queues : It can be used between related and unrelated processes running on a machine.
- Shared Memory: Shared memory is the fastest of all IPC schemes. The memory to be shared is mapped into the address space of the processes The speed achieved is attributed to the fact that there is no kernel involvement. But this scheme needs synchronization.

Various forms of synchronisation are mutexes, condition-variables, read-write locks, record-locks, and semaphores.

7. How do you execute one program from within another?

The system calls used for low-level process creation are "execvp()" and "execlp()". The "execlp()" call overlays the existing program with the new one, runs that and exits. The original program gets back control only when an error occurs.

8. What is the difference between Swapping and Paging?

- Swapping: Here the whole process is moved from the swap device to the main memory for execution. Process size must be less than or equal to the available main memory. It is easier to implement and overhead to the system. Swapping systems can not handle the memory more flexibly as compared to the paging systems.
- Paging: Here only the required memory pages are moved to main memory from the swap device for execution. Process size does not matter. Gives the concept of the virtual memory. It provides greater flexibility in mapping the virtual address space into the physical memory of the machine. Allows more number of processes to fit in the main memory simultaneously. Allows the greater process size than the available physical memory. Demand paging systems handle the memory more flexibly.

9. Write the process communication between parent and child.

A parent and child can communicate through the normal inter-process communication schemes, but also have some special ways to communicate that take advantage of their relationship as a parent and child. One of the most obvious is that the parent can get the exit status of the child.

10. What's the difference between fork() and vfork()?

vfork() and fork() both are system calls, but there are some differences

- vfork() is the lower-overhead version of fork(). fork() is the original system call.
- fork() is used to copy the entire address space of the process whereas, vfork() is used to increase the functionality of fork().