UNIX Operating System Interview Questions And Answers Guide.

Global Guideline.

https://globalguideline.com/

UNIX Operating System Job Interview Preparation Guide.

Question # 1

How are devices represented in UNIX OS?

Answer:-

All devices are represented by files called special files that are located in/dev directory. Thus, device files and other files are named and accessed in the same way. A 'regular file' is just an ordinary data file in the disk. A 'block special file' represents a device with characteristics similar to a disk (data transfer in terms of blocks). A 'character special file' represents a device with characteristics similar to a keyboard (data transfer is by stream of bits in sequential order).

Read More Answers.

Question # 2

What is inode in UNIX OS?

Answer:-

All UNIX files have its description stored in a structure called inode. The inode contains info about the file-size, its location, time of last access, time of last modification, permission and so on. Directories are also represented as files and have an associated inode. In addition to descriptions about the file, the inode contains pointers to the data blocks of the file. If the file is large, inode has indirect pointer to a block of pointers to additional data blocks (this further aggregates for larger files). A block is typically 8k.
Inode consists of the following fields:
File owner identifier
File type
File access permissions
File access times
Number of links
File size
Location of the file data

Read More Answers.

Question # 3

Brief about the directory representation in UNIX OS?

Answer:-

A Unix directory is a file containing a correspondence between filenames and inodes. A directory is a special file that the kernel maintains. Only kernel modifies directories, but processes can read directories. The contents of a directory are a list of filename and inode number pairs. When new directories are created, kernel makes two entries named . (refers to the directory itself) and .. (refers to parent directory).
System call for creating directory is mkdir (pathname, mode).

Read More Answers.

Question # 4

What are the Unix system calls for I/O?

Answer:-

open(pathname,flag,mode) - open file
creat(pathname,mode) - create file
close(filedes) - close an open file
read(filedes,buffer,bytes) - read data from an open file
write(filedes,buffer,bytes) - write data to an open file
lseek(filedes,offset,from) - position an open file
dup(filedes) - duplicate an existing file descriptor
dup2(oldfd,newfd) - duplicate to a desired file descriptor
fcntl(filedes,cmd,arg) - change properties of an open file
ioctl(filedes,request,arg) - change the behaviour of an open file
The difference between fcntl anf ioctl is that the former is intended for any open file, while the latter is for device-specific operations.

Read More Answers.

Question # 5

How do you change File Access Permissions in UNIX OS?

Answer:-

Every file has following attributes:
owner's user ID (16 bit integer)
owner's group ID (16 bit integer)
File access mode word
'r w x -r w x- r w x'
(user permission-group permission-others permission)
r-read, w-write, x-execute
To change the access mode, we use chmod(filename,mode).
Example 1:
To change mode of myfile to 'rw-rw-r--' (ie. read, write permission for user - read,write permission for group - only read permission for others) we give the args as:
chmod(myfile,0664) .
Each operation is represented by discrete values
'r' is 4
'w' is 2
'x' is 1
Therefore, for 'rw' the value is 6(4+2).
Example 2:
To change mode of myfile to 'rwxr--r--' we give the args as:
chmod(myfile,0744).

Read More Answers.

Question # 6

What are links and symbolic links in UNIX OS file system?

Answer:-

A link is a second name (not a file) for a file. Links can be used to assign more than one name to a file, but cannot be used to assign a directory more than one name or link filenames on different computers.
Symbolic link 'is' a file that only contains the name of another file.Operation on the symbolic link is directed to the file pointed by the it.Both the limitations of links are eliminated in symbolic links.
Commands for linking files are:
Link ln filename1 filename2
Symbolic link ln -s filename1 filename2

Read More Answers.

Question # 7

What is a FIFO in UNIX OS?

Answer:-

FIFO are otherwise called as 'named pipes'. FIFO (first-in-first-out) is a special file which is said to be data transient. Once data is read from named pipe, it cannot be read again. Also, data can be read only in the order written. It is used in interprocess communication where a process writes to one end of the pipe (producer) and the other reads from the other end (consumer).

Read More Answers.

Question # 8

How do you create special files like named pipes and device files in UNIX OS?

Answer:-

The system call mknod creates special files in the following sequence.
1. kernel assigns new inode,
2. sets the file type to indicate that the file is a pipe, directory or special file,
3. If it is a device file, it makes the other entries like major, minor device numbers.
For example:
If the device is a disk, major device number refers to the disk controller and minor device number is the disk.

Read More Answers.

Question # 9

Discuss the mount and unmount system calls in UNIX OS?

Answer:-

The privileged mount system call is used to attach a file system to a directory of another file system; the unmount system call detaches a file system. When you mount another file system on to your directory, you are essentially splicing one directory tree onto a branch in another directory tree. The first argument to mount call is the mount point, that is , a directory in the current file naming system. The second argument is the file system to mount to that point. When you insert a cdrom to your unix system's drive, the file system in the cdrom automatically mounts to /dev/cdrom in your system.

Read More Answers.

Question # 10

How does the inode map to data block of a file in UNIX OS?

Answer:-

Inode has 13 block addresses. The first 10 are direct block addresses of the first 10 data blocks in the file. The 11th address points to a one-level index block. The 12th address points to a two-level (double in-direction) index block. The 13th address points to a three-level(triple in-direction)index block. This provides a very large maximum file size with efficient access to large files, but also small files are accessed directly in one disk read.

Read More Answers.

Question # 11

What is a UNIX OS shell?

Answer:-

A shell is an interactive user interface to an operating system services that allows an user to enter commands as character strings or through a graphical user interface. The shell converts them to system calls to the OS or forks off a process to execute the command. System call results and other information from the OS are presented to the user through an interactive interface. Commonly used shells are sh,csh,ks etc.

Read More Answers.

Question # 12

Brief about the initial process sequence while the UNIX system boots up.

Answer:-

While booting, special process called the 'swapper' or 'scheduler' is created with Process-ID 0. The swapper manages memory allocation for processes and influences CPU allocation. The swapper inturn creates 3 children:
the process dispatcher,
vhand and
dbflush
with IDs 1,2 and 3 respectively.
This is done by executing the file /etc/init. Process dispatcher gives birth to the shell. Unix keeps track of all the processes in an internal data structure called the Process Table (listing command is ps -el).

Read More Answers.

Question # 13

What are various IDs associated with a process in UNIX OS?

Answer:-

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'. Every process is associated 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.
getpid() -process id
getppid() -parent process id
getuid() -user id
geteuid() -effective user id

Read More Answers.

Question # 14

Explain UNIX fork() system call.

Answer:-

The `fork()' used to create a new process from an existing process. The new process is called the child process, and the existing process is called the parent. We can tell which is which by checking the return value from `fork()'. The parent gets the child's pid returned to him, but the child gets 0 returned to him.

Read More Answers.

Question # 15

Predict the output of the following program code in UNIX?

Answer:-

```
main()
{
fork();
printf("Hello World!");
}
```
Answer:
Hello World!Hello World!
Explanation:
The fork creates a child that is a duplicate of the parent process. The child begins from the fork().All the statements after the call to fork() will be executed twice.(once by the parent process and other by child). The statement before fork() is executed only by the parent process.

Read More Answers.

Question # 16

Predict the output of the following program code?

Answer:-

```
main()
{
fork(); fork(); fork();
printf("Hello World!");
}
```
Answer:
"Hello World" will be printed 8 times.
Explanation:
2^n times where n is the number of calls to fork()

Read More Answers.

Question # 17

List the system calls used for process management in UNIX?

Answer:-

System calls Description
fork() To create a new process
exec() To execute a new program in a process
wait() To wait until a created process completes its execution
exit() To exit from a process execution
getpid() To get a process identifier of the current process
getppid() To get parent process identifier
nice() To bias the existing priority of a process
brk() To increase/decrease the data segment size of a process

Read More Answers.

Question # 18

How can you get/set an environment variable from a program in UNIX?

Answer:-

Getting the value of an environment variable is done by using `getenv()'. Setting the value of an environment variable is done by using `putenv()'.

Read More Answers.

Question # 19

How can a parent and child process communicate?

Answer:-

A parent and child can communicate through any of the normal inter-process communication schemes (pipes, sockets, message queues, shared memory), 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.

Read More Answers.

Question # 20

What is a zombie?

Answer:-

When a program forks and the child finishes before the parent, the kernel still keeps some of its information about the child in case the parent might need it - for example, the parent may need to check the child's exit status. To be able to get this information, the parent calls `wait()'; In the interval between the child terminating and the parent calling `wait()', the child is said to be a `zombie' (If you do `ps', the child will have a `Z' in its status field to indicate this.)

Read More Answers.

Question # 21

What are the process states in Unix?

Answer:-

As a process executes it changes state according to its circumstances. Unix processes have the following states:
Running : The process is either running or it is ready to run .
Waiting : The process is waiting for an event or for a resource.
Stopped : The process has been stopped, usually by receiving a signal.
Zombie : The process is dead but have not been removed from the process table.

Read More Answers.

Question # 22

What Happens when you execute a program?

Answer:-

When you execute a program on your UNIX system, the system creates a special environment for that program. This environment contains everything needed for the system to run the program as if no other program were running on the system. Each process has process context, which is everything that is unique about the state of the program you are currently running. Every time you execute a program the UNIX system does a fork, which performs a series of operations to create a process context and then execute your program in that context. The steps include the following:
Allocate a slot in the process table, a list of currently running programs kept by UNIX.
Assign a unique process identifier (PID) to the process.
iCopy the context of the parent, the process that requested the spawning of the new process.
Return the new PID to the parent process. This enables the parent process to examine or control the process directly. After the fork is complete, UNIX runs your program.

Read More Answers.

Question # 23

What Happens when you execute a command in UNIX OS?

Answer:-

When you enter 'ls' command to look at the contents of your current working directory, UNIX does a series of things to create an environment for ls and the run it:

The shell has UNIX perform a fork. This creates a new process that the shell will use to run the ls program. The shell has UNIX perform an exec of the ls program. This replaces the shell program and data with the program and data for ls and then starts running that new program. The ls program is loaded into the new process context, replacing the text and data of the shell. The ls program performs its task, listing the contents of the current directory.

Read More Answers.

Operating System Most Popular Interview Topics.

1 : Windows Frequently Asked Interview Questions and Answers Guide.

2 : Operating System (OS) Frequently Asked Interview Questions and Answers Guide.

3 : Windows 7 Frequently Asked Interview Questions and Answers Guide.

4 : Solaris Frequently Asked Interview Questions and Answers Guide.

5 : Real-Time Operating System (RTOS) Frequently Asked Interview Questions and Answers Guide.

6 : MAC OS Frequently Asked Interview Questions and Answers Guide.

7 : Solaris Admin Frequently Asked Interview Questions and Answers Guide.

8 : Shell Scripting Frequently Asked Interview Questions and Answers Guide.

9 : Unix Commands Frequently Asked Interview Questions and Answers Guide.

10 : Unix Socket Programming Frequently Asked Interview Questions and Answers Guide.

About Global Guideline.

Global Guideline is a platform to develop your own skills with thousands of job interview questions and web tutorials for fresher's and experienced candidates. These interview questions and web tutorials will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts. Global Guideline invite you to unlock your potentials with thousands of **Interview Questions with Answers** and much more. Learn the most common technologies at Global Guideline. We will help you to explore the resources of the World Wide Web and develop your own skills from the basics to the advanced. Here you will learn anything quite easily and you will really enjoy while learning. Global Guideline will help you to become a professional and Expert, well prepared for the future.

* This PDF was generated from https://GlobalGuideline.com at **November 29th, 2023**

* If any answer or question is incorrect or inappropriate or you have correct answer or you found any problem in this document then don't hesitate feel free and e-mail us we will fix it.

You can follow us on FaceBook for latest Jobs, Updates and other interviews material.
www.facebook.com/InterviewQuestionsAnswers

Follow us on Twitter for latest Jobs and interview preparation guides
https://twitter.com/InterviewGuide

Best Of Luck.

Global Guideline Team
https://GlobalGuideline.com
Info@globalguideline.com