Unix And Linux Interview Questions

April 15, 2024

1 What is shell?

A shell in Unix-like operating systems is a user interface for access to the operating system's services. It can be either a command-line interface (CLI) or a graphical user interface (GUI).

In the context of a CLI, a shell is a program that interprets text commands and sends them to the operating system to execute. It provides a way for users to interact with the system by typing commands into a terminal. The shell processes these commands and communicates with the operating system to perform the requested operations.

There are several types of shells in Unix and Linux, each with its own set of features and capabilities. Some of the most common ones include the Bourne Shell (sh), the C Shell (csh), the Korn Shell (ksh), the Bourne Again Shell (bash), and the Z Shell (zsh).

In addition to executing system commands, shells also provide a programming environment and allow users to write shell scripts. These scripts are files containing a sequence of commands, and they can be used to automate repetitive tasks, perform complex operations, and manage system resources.

2 Name the commonly used shells on different UNIX/Linux variants.

There are several different shells that are commonly used on Unix and Linux systems. Here are some of the most popular ones:

1. **Bourne Shell (sh)**: The original Unix shell, developed by Stephen Bourne at AT&T's Bell Labs. It's known for its simplicity and portability across different Unix systems.
2. **Bourne Again Shell (bash)**: A superset of the Bourne Shell that adds many features, including command line editing, job control, and shell functions. It's the default shell on many Linux distributions and on macOS.
3. **C Shell (csh)**: Developed by Bill Joy at the University of California, Berkeley. Its syntax is similar to the C programming language, and it introduced many features that are now common in other shells, such as history and command line editing.
4. **TC Shell (tcsh)**: An enhanced version of the C Shell that adds features like command line completion and command line editing.
5. **Korn Shell (ksh)**: Developed by David Korn at AT&T's Bell Labs. It combines features from both the Bourne Shell and the C Shell, and it adds some unique features of its own.

6. **Z Shell (zsh)**: A shell designed to be a comprehensive replacement for both the Bourne Shell and the C Shell. It includes many features for interactive use and scripting, and it's highly customizable.
7. **Fish (friendly interactive shell)**: A shell that aims to be user-friendly and interactive, with features like autosuggestions and a web-based configuration interface.

Each of these shells has its own strengths and weaknesses, and the best one to use often depends on the specific needs and preferences of the user.

3 What is soft link/symbolic link and which command can be used to create one?

A symbolic link, also known as a soft link, is a type of file that is a reference to another file or directory in the form of an absolute or relative path. Unlike a hard link, a symbolic link can point to a file or directory on a different filesystem, and it remains valid even if the target file is moved or deleted.

You can create a symbolic link using the `ln` command with the `-s` option in Unix-like operating systems. Here's the general syntax:

[ ]:
```
ln -s target link_name
```

In this command, `target` is the file or directory that you want to link to, and `link_name` is the name of the symbolic link that you want to create.

For example, to create a symbolic link named `my_link` that points to a file named `my_file`, you would use the following command:

[ ]:
```
ln -s my

_file

 my_link
```

After running this command, `my_link` will be a symbolic link to `my_file`. If you use `ls -l`, you'll see that `my_link` points to `my_file`.

Note that if the target file is deleted, the symbolic link will still exist but it will be broken, i.e., it will not point to anything.

4 What is Sun micro systems UNIX called as?

Sun Microsystems' Unix operating system is called Solaris. It's a highly scalable and robust OS known for its reliability and security features. It's widely used in enterprise environments and supports both SPARC and x86 hardware. Sun Microsystems was acquired by Oracle Corporation in 2010, and the operating system is now known as Oracle Solaris.

5 What is the role of kernel?

The kernel is the core component of an operating system. It acts as a bridge between applications and the actual data processing done at the hardware level. The kernel's responsibilities include:

1. **Process Management**: The kernel is responsible for creating, scheduling, and terminating processes. It manages process communication and coordination through mechanisms like semaphores and message queues.
2. **Memory Management**: The kernel handles memory allocation for processes and manages virtual memory, providing a larger virtual address space than the physical memory available.
3. **Device Management**: The kernel contains drivers for interacting with hardware devices. It manages these devices and provides a standardized interface for the system to interact with them.
4. **File System Management**: The kernel implements the file system and manages file and directory operations.
5. **System Calls and Security**: The kernel provides system calls, which are interfaces that user-level applications can use to request services from the operating system. It also manages access rights and privileges for processes and users, ensuring system security.
6. **Networking**: The kernel handles network communications, implementing protocols and managing sockets and connections.

In summary, the kernel is the part of the operating system that interacts directly with the hardware, providing a platform for running all other software on the system.

6 Which program is responsible to bring the login prompt?

The program responsible for bringing up the login prompt in Unix-like operating systems is called `getty`. The name `getty` stands for "get teletype", referring to the old teletype terminals that were used in the early days of Unix.

`getty` is responsible for protecting the system from unauthorized access. It sets up the terminal lines (or console), sets the login process's controlling terminal, and then waits for a user to log in. When a user logs in, `getty` runs the `login` program, which prompts the user for their username and password.

In some modern Linux distributions, `getty` has been replaced by similar programs like `agetty` or `mingetty`, or by more complex terminal managers like `systemd-logind` (part of the `systemd` init system). But the basic function is the same: to set up the terminal and bring up the login prompt.

7 What is the name of the command to change the password?

The command to change the password in Unix-like operating systems is `passwd`.

Here's how you can use it:

[ ]:
```
passwd
```

When you run this command, you'll be prompted to enter your current password, then the new password, and finally to confirm the new password by entering it again.

If you're a superuser (root), you can change the password for any user by specifying the username as an argument:

[ ]:
```
passwd username
```

In this command, replace `username` with the actual username of the account for which you want to change the password. You'll be prompted to enter the new password and confirm it.

8 What is the second column designates in the long listing of ls command's output?

The second column in the long listing (`ls -l`) of the `ls` command's output represents the number of hard links to the file or directory.

A hard link is essentially a reference to the physical data on the disk. For files, the number of hard links is usually 1, unless more hard links have been created using the `ln` command without the `-s` option.

For directories, the number of hard links is usually the number of immediate subdirectories plus 2. The extra two links come from the `.` (dot) directory link, which points to the directory itself, and the `..` (dot-dot) directory link, which points to the parent directory.

9 What are the various ways that you can disconnect/logout with the UNIX server from your console terminal?

There are several ways to disconnect or log out from a Unix server from your console terminal:

1. **exit command**: You can type `exit` and press Enter. This will end the current shell session.

   ```
   exit
   ```

2. **logout command**: If you're in a login shell, you can use the `logout` command to end the session.

   ```
   logout
   ```

3. **Ctrl+D**: You can press `Ctrl+D`. This sends an EOF (End Of File) character, which signals the end of the input and causes the shell to exit.

4. **kill command**: If you know the process ID of the shell you want to terminate, you can use the `kill` command. For example, to kill shell with process ID 1234:

   ```
   kill -9 1234
   ```

5. **pkill command**: If you want to kill all instances of a particular shell, you can use the `pkill` command. For example, to kill all bash shells:

   ```
   pkill bash
   ```

Remember, it's important to properly disconnect from a Unix server to ensure that all your processes are terminated correctly and your work is saved.

10 Name a command which can be used to shut down the system?

The `shutdown` command is used to shut down or restart a Unix-like operating system. Here's how you can use it:

To shut down the system immediately:

```
[ ]: shutdown -h now
```

In this command, `-h` stands for halt, which means to shut down the system, and `now` means to do it immediately.

To schedule a shutdown, you can replace `now` with a specific time. For example, to shut down the system in 30 minutes:

```
[ ]: shutdown -h +30
```

To cancel a scheduled shutdown:

```
[ ]: shutdown -c
```

Note: You need to have superuser (root) privileges to run the `shutdown` command. If you're not logged in as root, you'll need to use `sudo`:

```
[ ]: sudo shutdown -h now
```

Remember to save all your work before shutting down the system to prevent data loss.

11 Is UNIX/Linux case sensitive from the console terminal end.

Yes, Unix and Linux are case sensitive. This means that the names of files, directories, variables, and commands are all case sensitive. For example, a file named `File.txt` is different from a file named `file.txt`. Similarly, the command `ls` is different from the command `LS`.

This case sensitivity extends to the console terminal as well. When you're typing commands in the terminal, you need to use the correct case. If you don't, you might get an error message, or you might end up running a completely different command than the one you intended.

12 Name a command which can be used to restart the machine?

The `reboot` command is used to restart a Unix-like operating system. Here's how you can use it:

```
[ ]: reboot
```

Alternatively, you can use the `shutdown` command with the `-r` option, which stands for reboot:

```
[ ]: shutdown -r now
```

In this command, `-r` stands for reboot, and `now` means to do it immediately.

Note: You need to have superuser (root) privileges to run the `reboot` or `shutdown` command. If you're not logged in as root, you'll need to use `sudo`:

[ ]:
```
sudo reboot
```

or

[ ]:
```
sudo shutdown -r now
```

Remember to save all your work before restarting the system to prevent data loss.

13 Is UNIX open source?

The original Unix operating system, developed by AT&T's Bell Labs in the 1970s, is not open source. It is proprietary software, and its source code is owned by Novell.

However, over the years, many open source operating systems have been developed based on the concepts and design principles of Unix. These include the BSD family of operating systems (FreeBSD, OpenBSD, NetBSD), and, indirectly, Linux.

Linux is not a Unix system, but it is Unix-like, and it was designed to be compatible with Unix. The source code of Linux is open source, and it is freely available for anyone to view, modify, and distribute under the terms of the GNU General Public License (GPL).

There's also an open source version of Unix itself called Illumos, which is a continuation of the open source release of Sun Microsystems' Solaris operating system, known as OpenSolaris.

So while Unix itself is not open source, many Unix-like or Unix-derived systems are.

14 Name 5 variants of linux?

There are many variants of Linux, known as distributions or distros. Here are five popular ones:

1. **Ubuntu**: One of the most popular Linux distributions, known for its user-friendly interface and strong community support. It's based on Debian and comes with a wide range of software pre-installed.
2. **Fedora**: A community-driven project sponsored by Red Hat. It's known for its commitment to innovation and features the latest software applications.
3. **Debian**: A Unix-like operating system composed entirely of free software. It's known for its robustness and reliability.
4. **CentOS**: A free and open-source community distribution that aims to be functionally compatible with its upstream source, Red Hat Enterprise Linux (RHEL).
5. **Arch Linux**: Known for its simplicity and full control to the end user. It uses a rolling release model, which means updates are continuous, and users don't need to reinstall the entire operating system to keep it up-to-date.

Each of these distributions has its own strengths and is suited to different types of users, from beginners to advanced users, and different use cases, from desktop use to servers.

15 What is a hidden file?

In Unix-like operating systems, a hidden file is a file that is not normally visible when you list the files in a directory. Hidden files are often used to store configuration settings for applications.

A file is made hidden by starting its name with a dot (`.`). For example, `.myhiddenfile` would be a hidden file.

You can view hidden files by using the `-a` option with the `ls` command:

[ ]:
```
ls -a
```

In this command, `-a` stands for "all", which means to show all files, including hidden ones.

Remember that just because a file is hidden doesn't mean it's secure. Hidden files are not a form of security, but rather a way to reduce clutter and prevent users from accidentally modifying or deleting important files.

16 Who designed the UNIX operating system?

The Unix operating system was designed by Ken Thompson and Dennis Ritchie at AT&T's Bell Labs in the United States. The development started in the late 1960s and the first version was released in 1971.

Ken Thompson primarily designed and implemented the original Unix operating system. Dennis Ritchie, who is also the creator of the C programming language, significantly contributed to the development of Unix, especially its rewriting in C which made it portable and changed the history of operating systems.

Other notable contributors to early versions of Unix include Brian Kernighan, Douglas McIlroy, and Joe Ossanna.

17 Which programming language is used in the design of UNIX OS.

The Unix operating system was originally written in assembly language. However, in 1973, Unix was re-written in the C programming language. This was a revolutionary step at the time, as operating systems were typically written in assembly language.

The re-write in C was performed by Dennis Ritchie and Ken Thompson, the original designers of Unix. The use of C made Unix portable, meaning it could be used on different types of hardware with minimal changes. This was a major factor in the widespread adoption of Unix.

Today, Unix and its derivatives (like Linux and BSD) are still largely written in C, with some parts written in assembly language for hardware-specific operations. Some newer components may also be written in other languages, but C remains the primary language for system-level programming.

18 At which company UNIX is designed first.

The Unix operating system was first designed and developed at AT&T's Bell Labs in the United States. The development started in the late 1960s and the first version was released in 1971. The original designers of Unix were Ken Thompson and Dennis Ritchie.

19 What does the second column signifies in the output generated by 'who' command?

The `who` command in Unix-like operating systems provides information about users who are currently logged into the system.

The output of the `who` command typically consists of several columns:

1. Username: The name of the logged-in user.
2. **Terminal**: The terminal line the user is logged in from.
3. Date and time: When the user logged in.
4. Remote host: The host name or IP address of the remote machine the user logged in from (if applicable).

So, the second column in the output of the `who` command signifies the terminal line from which the user is logged in. This could be a physical terminal on the machine, or a virtual terminal established by a remote login session.

20 Which command can give the manual of a command?

The `man` command is used to display the manual of a command in Unix-like operating systems. Here's how you can use it:

```
[ ]: man command_name
```

In this command, replace `command_name` with the name of the command you want to learn about. For example, to view the manual for the `ls` command, you would type:

```
[ ]: man ls
```

The `man` command provides detailed information about the command, including a description of what it does, the options it accepts, and examples of how to use it. It's a great way to learn about new commands or to refresh your memory about commands you've used before.

21 Which command can be used to know about the directory path of a command?

The `which` command is used in Unix-like operating systems to identify the directory path of a command. Here's how you can use it:

```
[ ]: which command_name
```

In this command, replace `command_name` with the name of the command you want to find the path for. For example, to find the path of the `ls` command, you would type:

[ ]:
```
which ls
```

The `which` command searches the directories in your PATH environment variable for the specified command, and prints the full path of the first match it finds.

If the command is not found in any of the directories in your PATH, or if the command is a shell built-in, the `which` command will not return any output.

22 What are the meta-characters which represent to match all and single chracters respectively?

In Unix-like operating systems, the following meta-characters are commonly used in pattern matching:

1. **Asterisk (*)**: This meta-character is used to match zero or more characters. For example, `*.txt` would match any file with a `.txt` extension.
2. **Question mark (?)**: This meta-character is used to match exactly one character. For example, `?.txt` would match any file with a `.txt` extension and a single character name like `a.txt`, `1.txt`, etc.

These meta-characters can be used with various Unix commands, like `ls`, `cp`, `mv`, etc., to perform operations on multiple files at once. They are also used in regular expressions to define search patterns.

23 What is the file descriptor which represents the STDIN.

In Unix-like operating systems, the standard input (STDIN) is represented by the file descriptor 0.

A file descriptor is a non-negative integer that is used to access a file or a stream. The operating system uses file descriptors to keep track of open files and I/O streams.

In addition to STDIN (0), there are two other standard streams represented by file descriptors:

1. Standard output (STDOUT), represented by file descriptor 1.
2. Standard error (STDERR), represented by file descriptor 2.

These three file descriptors (0, 1, 2) are automatically opened by the operating system when a process is started.

24 What is difference between absolute and relative path?

An absolute path and a relative path are two ways to specify the location of a file or a directory in a filesystem.

1. **Absolute Path**: An absolute path is the full path to a file or directory from the root of the filesystem. It starts with a slash (`/`). For example, `/home/user/documents/file.txt` is an absolute path.

2. **Relative Path**: A relative path is a path to a file or directory relative to the current working directory. It does not start with a slash. For example, if your current working directory is `/home/user/documents`, you could refer to `file.txt` in this directory with the relative path `file.txt`.

The main difference between the two is that an absolute path shows the exact path from the root directory to the file or directory, while a relative path shows the path from the current directory to the file or directory.

25 What is the meaning of execute permission on a directory?

In Unix-like operating systems, permissions control the ability of the users to read, write, and execute files or directories. When it comes to directories, the execute permission (`x`) has a specific meaning:

- **Execute Permission on a Directory**: If the execute permission is set on a directory, it means you have the permission to access (i.e., `cd` into) and list the contents of that directory. If the execute permission is not set, you cannot enter the directory or access its contents, even if you have read permission on it.

It's important to note that the execute permission on a directory is often combined with the read permission (`r`). The read permission allows you to see the names of the files in the directory, while the execute permission allows you to enter the directory and access these files. Without both permissions, you may not be able to fully access and navigate a directory.

26 Name the environment variable which can be used to change the primary shell prompt.

The environment variable that can be used to change the primary shell prompt in Unix-like operating systems is `PS1`.

`PS1` stands for "Prompt String 1". It defines the primary prompt string which is displayed whenever the shell is ready to read a command.

You can customize the `PS1` variable to change the appearance of your shell prompt. For example, you can include the current directory, the user name, the host name, or other information in your prompt.

Here's an example of how to set `PS1`:

[ ]:
```
PS1="\u@\h:\w\$ "
```

In this example, `\u` represents the username, `\h` represents the hostname, `\w` represents the current working directory, and `\$` represents the prompt symbol (`$` for regular users, `#` for the root user). The changes will apply to the current session only.

To make the changes permanent, you can add the `PS1` setting to a shell startup file like `~/.bashrc` or `~/.bash_profile`.

27 Which command can be used to print a document on the line printer typically?

The `lpr` command is typically used in Unix-like operating systems to print a document on a line printer. Here's how you can use it:

```
[ ]: lpr filename
```

In this command, replace `filename` with the name of the file you want to print. For example, to print a file named `document.txt`, you would type:

```
[ ]: lpr document.txt
```

The `lpr` command sends the file to the printer's queue and the printing happens in the background, allowing you to continue using the terminal for other tasks.

Please note that the `lpr` command requires a correctly configured printer. The specifics of setting up a printer can vary depending on the operating system and the printer model.

28 What is a process?

A process is an instance of a program that is being executed. It contains the program code and its current activity. Each process in a Unix-like operating system has a unique process ID (PID).

When a program is run, the operating system creates a new process and assigns it a unique PID. This process contains the information about the program's execution, including the program counter that keeps track of the next instruction to be executed, and the process's variables, buffers, and other data.

Processes can be in one of several 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), or zombie (the process is dead but has not been removed from the process table).

In Unix-like operating systems, you can use commands like `ps` and `top` to view information about the currently running processes, and commands like `kill` and `nice` to control the execution of processes.

29 How to open a file in read only mode using vi editor?

To open a file in read-only mode using the `vi` editor, you can use the `-R` option with the `vi` command. Here's how you can do it:

```
[ ]: vi -R filename
```

In this command, replace `filename` with the name of the file you want to open. For example, to open a file named `document.txt` in read-only mode, you would type:

```
[ ]: vi -R document.txt
```

When a file is opened in read-only mode, you can view the contents of the file, but you cannot make changes to it. If you try to modify the file, `vi` will display a warning message. To save changes, you would need to open the file in normal mode (without the `-R` option).

30 Which alphabet keys can move the cursor equivalent to arrow keys?

In the `vi` and `vim` editors, you can use the following alphabet keys to move the cursor around, similar to the arrow keys:

- `h`: Move the cursor to the left (equivalent to the left arrow key).
- `j`: Move the cursor down (equivalent to the down arrow key).
- `k`: Move the cursor up (equivalent to the up arrow key).
- `l`: Move the cursor to the right (equivalent to the right arrow key).

These keys allow you to navigate through your file without moving your hands away from the home row on the keyboard, which can be faster and more efficient once you get used to it. Note that these keys work in command mode, not in insert mode.

31 What is yanking w.r.t vi editor?

In the context of the `vi` or `vim` editor, "yanking" refers to copying text.

When you yank text, it is stored in a buffer and can be pasted elsewhere in your document. The yank command in `vi` or `vim` is `y`.

Here are a few examples of how to use the yank command:

- `yy` or `Y`: Yank (copy) the current line.
- `y$`: Yank from the cursor to the end of the line.
- `yw`: Yank from the cursor to the end of the current word.
- `y2w`: Yank the next two words.
- `2yy`: Yank the next two lines.

After yanking the text, you can paste it with the `p` command to paste after the cursor, or `P` to paste before the cursor.

32 How can you display line numbers in vi editor?

To display line numbers in the `vi` or `vim` editor, you can use the `:set number` command. Here's how you can do it:

1. Open `vi` or `vim` and open your file.
2. Press `:` to enter command-line mode.
3. Type `set number` and press Enter.

This will display line numbers on the left side of the window.

If you want to hide the line numbers, you can use the `:set nonumber` command in the same way.

To make `vi` or `vim` always display line numbers when you open a file, you can add the `set number` command to your `vimrc` file, which is the configuration file for `vim`. The `vimrc` file is located in your home directory and is named `.vimrc`.

33 What is a shebang line?

A shebang line in Unix-like operating systems is the first line of a script file that tells the system what interpreter to use to execute the script. It starts with the characters `#!` followed by the path to the interpreter.

Here are a few examples of shebang lines:

- `#!/bin/sh`: This shebang line tells the system to execute the script using the Bourne shell (`sh`).
- `#!/bin/bash`: This shebang line tells the system to execute the script using the Bash shell.
- `#!/usr/bin/python3`: This shebang line tells the system to execute the script using Python 3.
- `#!/usr/bin/env node`: This shebang line tells the system to execute the script using Node.js. The `env` command is used to find the location of the `node` executable in the system's `PATH`.

The shebang line must be the very first line in the script. If the script is made executable (using a command like `chmod +x scriptname`), you can run it just like a binary, and the system will use the specified interpreter to execute the script.

34 Which command can be used to alter the file access permissions?

The `chmod` command is used in Unix-like operating systems to change the access permissions of a file or a directory. Here's how you can use it:

[ ]:
```
chmod permissions filename
```

In this command, replace `permissions` with the permissions you want to set, and `filename` with the name of the file or directory you want to change permissions for.

Permissions can be specified in several ways:

- Using symbolic notation: `u` for user, `g` for group, `o` for others, `a` for all. `+` to add a permission, `-` to remove a permission, `=` to set a permission. `r` for read, `w` for write, `x` for execute. For example, `chmod u+x filename` adds execute permission for the user.
- Using octal notation: `4` for read, `2` for write, `1` for execute. The permissions are added up to get a number from 0 to 7 for each category (user, group, others). For example, `chmod 755 filename` sets read, write, and execute permission for the user, and read and execute permission for the group and others.

Remember that only the owner of a file (or the root user) can change the permissions of a file.

35 How can we retrieve the value of the shell variable?

In Unix-like operating systems, you can retrieve the value of a shell variable by prefixing the variable name with a dollar sign (`$`). Here's how you can do it:

```
[ ]: echo $VARIABLE_NAME
```

In this command, replace `VARIABLE_NAME` with the name of the variable you want to retrieve. For example, to retrieve the value of a variable named `PATH`, you would type:

```
[ ]: echo $PATH
```

This will print the value of the `PATH` variable to the terminal.

Note that variable names are case-sensitive, so `PATH` and `Path` would refer to two different variables. Also, when referencing a variable, do not include a space after the dollar sign.

36 Which command can be used to delete a shell variable?

In Unix-like operating systems, you can delete a shell variable using the `unset` command. Here's how you can do it:

```
[ ]: unset VARIABLE_NAME
```

In this command, replace `VARIABLE_NAME` with the name of the variable you want to delete. For example, to delete a variable named `MY_VARIABLE`, you would type:

```
[ ]: unset MY_VARIABLE
```

After running this command, `MY_VARIABLE` will be removed from the list of defined shell variables. If you try to access its value with `echo $MY_VARIABLE`, nothing will be printed because the variable no longer exists.

Remember that shell variables are case-sensitive, so `MY_VARIABLE` and `my_variable` would refer to two different variables.

37 Which shell variable holds the shell script file name?

The shell variable that holds the name of the current shell script being executed is `$0`.

Here's how you can use it:

```
[ ]: echo $0
```

This command will print the name of the current shell script. If you run this command in a shell script, it will print the name of that script. If you run this command in a terminal session, it will print the name of the shell (for example, `-bash` or `/bin/bash`).

Remember that `$0` only gives the name of the script, not the full path. If you need the full path of the script, you can use a command like `readlink -f $0` or `realpath $0` in the script.

38 What is the value returned by a command after its successful execution?

After a command is executed in Unix-like operating systems, it returns a value known as the exit status. The exit status is a number that indicates whether the command was successful or not.

- If the command was successful, it returns an exit status of `0`.
- If the command was not successful, it returns a non-zero exit status. The specific non-zero value can indicate different types of errors, depending on the command.

You can retrieve the exit status of the last command executed in the shell with the special variable `$?`. Here's how you can do it:

[ ]:
```
echo $?
```

This command will print the exit status of the last command. If the last command was successful, this will print `0`. If the last command was not successful, this will print the non-zero exit status returned by the command.

39 How can we get the execution status of the last executed command?

In Unix-like operating systems, you can get the execution status of the last executed command using the special variable `$?`. Here's how you can do it:

[ ]:
```
echo $?
```

This command will print the exit status of the last command. If the last command was successful, this will print `0`. If the last command was not successful, this will print the non-zero exit status returned by the command.

Remember to use this command immediately after the command for which you want to check the exit status. If you run another command before checking `$?`, the exit status will be overwritten with the exit status of the most recent command.

40 Which command can be used to perform arithmetical computations?

In Unix-like operating systems, you can use the `expr` command to perform basic arithmetic computations. Here's how you can use it:

[ ]:
```
expr operand1 operator operand2
```

In this command, replace `operand1` and `operand2` with the numbers you want to compute, and `operator` with the operation you want to perform. For example, to add 5 and 3, you would type:

[ ]:
```
expr 5 + 3
```

This will print `8` to the terminal.

The `expr` command supports the following operators: `+` for addition, `-` for subtraction, `*` for multiplication, `/` for division, and `%` for modulus. Note that you need to escape the `*` operator with a backslash (`\*`) to prevent it from being interpreted as a wildcard by the shell.

Alternatively, you can use the `$((...))` syntax for arithmetic operations, which is more flexible and easier to use:

[ ]:
```
echo $((5 + 3))
```

This will also print `8` to the terminal. The `$((...))` syntax supports more operators and does not require you to escape the `*` operator.

41 Which switch can be used in test expression to determine a file as a regular file?

In Unix-like operating systems, you can use the `-f` switch in a test expression to check if a file is a regular file. Here's how you can do it:

[ ]:
```
if [ -f filename ]; then
    echo "filename is a regular file"
else
    echo "filename is not a regular file"
fi
```

In this script, replace `filename` with the name of the file you want to check. The `-f` test returns true if `filename` is a regular file. If `filename` is a directory, a symbolic link, a device file, or any other type of special file, the `-f` test returns false.

Remember to put spaces around the brackets in the test expression. If you omit the spaces, the shell will interpret `[` or `]` as part of the filename, which is not what you want.

42 How can we capture commands displayable output in a shell variable?

In Unix-like operating systems, you can capture the output of a command in a shell variable using command substitution. Here's how you can do it:

[ ]:
```
variable=$(command)
```

In this command, replace `variable` with the name of the variable where you want to store the output, and `command` with the command you want to run. For example, to store the output of the `date` command in a variable named `current_date`, you would type:

[ ]:
```
current_date=$(date)
```

After running this command, `current_date` will hold the current date and time. You can print the value of `current_date` with `echo $current_date`.

Command substitution can be done with either `$(command)` or `` `command` `` (backticks), but `$(command)` is preferred because it is easier to read and can be nested.

43 What is a here document?

A "here document" (also known as a "heredoc") is a type of redirection that allows you to pass a block of input to a command. In shell scripting, it's often used when a script requires multiple lines of input.

Here's a basic example of a here document:

[ ]:
```
cat << EOF
This is line 1.
This is line 2.
This is line 3.
EOF
```

In this example, `cat` is the command, and `<< EOF` starts the here document. The lines that follow are the input to the `cat` command. The here document ends when the shell encounters `EOF` on a line by itself. The `EOF` can be any word you choose, as long as it's the same at the start and end of the here document.

The `cat` command will print the lines of input to the terminal. The here document allows you to specify multiple lines of input without having to echo each line separately.

Here documents can also be used with other commands that accept standard input, and they can include shell variables and command substitution.

44 Which command can be used to throw/return a value from the function?

In Unix-like operating systems, you can use the `return` command in a shell function to return a value. Here's how you can do it:

[ ]:
```
function_name() {
    # some commands
    return value
}
```

In this function, replace `function_name` with the name of your function, `# some commands` with the commands you want to run in the function, and `value` with the value you want to return. The `value` must be an integer between 0 and 255.

For example, here's a function that returns the value `5`:

[ ]:
```
my_function() {
    # some commands
    return 5
}
```

After calling this function with `my_function`, you can get the return value with `$?`:

```
[ ]: my

_function
echo $?
```

This will print `5` to the terminal.

Note that the `return` command in a shell function is similar to the `exit` command in a script, but `return` only ends the function, not the entire script. Also, the `return` command is not used to return a result like in other programming languages. To return a result from a function, you typically use command substitution or output parameters.

45 Is it compulsory to have the extension 'sh' for a shell script?

No, it is not compulsory to have the `.sh` extension for a shell script in Unix-like operating systems. The shell determines how to execute a file based on the file's shebang line (`#!/bin/sh`, `#!/bin/bash`, etc.), not its extension.

However, it is common practice to use the `.sh` extension for shell scripts to indicate that the file is a shell script. This makes it easier for other people (and for you) to understand what the file is just by looking at its name.

Remember that to run a shell script, you need to make it executable with the `chmod` command:

```
[ ]: chmod +x scriptname
```

Then you can run it with `./scriptname`. If the script is in your `PATH`, you can run it with just `scriptname`.

46 How can we execute a shell script if execute bit is off for a shell script?

Even if the execute bit is off for a shell script, you can still execute it by invoking the shell program directly with the script as an argument. Here's how you can do it:

```
[ ]: bash scriptname
```

In this command, replace `bash` with the name of the shell you want to use (it could be `sh`, `bash`, `ksh`, `zsh`, etc.), and `scriptname` with the name of your script.

This command will start a new shell process, which will read and execute the commands in the script. The script does not need to be executable for this to work.

However, remember that if the script contains a shebang line (`#!/bin/sh`, `#!/bin/bash`, etc.), you should use the same shell specified in the shebang line to execute the script. If you use a different shell, the script may not work as expected because different shells have different features and syntax.

47 What is the full form of grep?

The full form of `grep` is "Global Regular Expression Print".

This command-line utility is used in Unix-like operating systems for searching plain-text data sets for lines that match a regular expression. Its name comes from the `ed` command `g/re/p`, which has the same effect: doing a global search with a regular expression and printing all matching lines.

48 Which filter can be used to extract a/specific/range of character from the lines of text?

The `cut` command in Unix-like operating systems can be used to extract a specific range of characters from the lines of text. Here's how you can use it:

```
[ ]: cut -c start-end filename
```

In this command, replace `start` with the position of the first character you want to extract, `end` with the position of the last character you want to extract, and `filename` with the name of the file you want to process. For example, to extract characters 5 to 10 from each line of a file named `file.txt`, you would type:

```
[ ]: cut -c 5-10 file.txt
```

This command will print the extracted characters to the terminal.

The `cut` command can also extract fields separated by a delimiter with the `-f` option, and change the delimiter with the `-d` option. For example, to extract the first field from a CSV file, you would type:

```
[ ]: cut -d ',' -f 1 file.csv
```

This command will print the first field (column) of each line in `file.csv` to the terminal.

49 Command 'cat' basically does _____

The `cat` command in Unix-like operating systems is used to concatenate and display the content of files. The name `cat` is short for "concatenate".

Here's how you can use it:

```
[ ]: cat filename
```

In this command, replace `filename` with the name of the file you want to display. The `cat` command will print the content of the file to the terminal.

You can also use `cat` to concatenate multiple files into one:

```
[ ]: cat file1 file2 > file3
```

This command will concatenate `file1` and `file2` and write the result to `file3`. If `file3` already exists, it will be overwritten. If you want to append to `file3` instead of overwriting it, use `>>` instead of `>`.

In addition, `cat` can be used to create a new file by taking input from the terminal:

[ ]:
```
cat > filename
```

In this command, replace `filename` with the name of the file you want to create. After running this command, you can type the content of the file at the terminal. Press Ctrl+D to end the input.

50 What is a process id?

A Process ID (PID) is a unique identifier that is assigned by a Unix-like operating system to each process when it is created. Processes include all running instances of programs, whether they were started by the user or by the system.

The PID is used by the system to track the process's execution state, memory allocation, and other resources. It's also used by system calls and commands that need to interact with the process, such as `kill`, `nice`, and `top`.

PIDs are integers. The PID of the first process, `init` or `systemd`, is always 1. Other PIDs are assigned in ascending order as processes are created, up to a maximum value that depends on the system. When the maximum value is reached, the PID counter wraps around and starts from a low value again.

You can find the PID of a process with the `ps` command or in the output of the `top` command. The PID of the last background process can be retrieved with the special variable `$!` in the shell.

51 Which is the command used to create a new user on the system?

In Unix-like operating systems, you can use the `useradd` or `adduser` command to create a new user. The specific command and its syntax can vary between different systems.

Here's how you can use `useradd`:

[ ]:
```
sudo useradd -m username
```

In this command, replace `username` with the name of the user you want to create. The `-m` option tells `useradd` to create a home directory for the user.

After creating the user, you should set a password for the user with the `passwd` command:

[ ]:
```
sudo passwd username
```

This command will prompt you to enter a password for the user.

On some systems, `adduser` is a more user-friendly command that does the same thing:

[ ]:
```
sudo adduser username
```

This command will prompt you for the password and other information such as the full name of the user, and it will create the home directory automatically.

Remember that you need to have sudo privileges to create a new user.

52 Command 'telnet' is used for _____

The `telnet` command is used in Unix-like operating systems to interact with another host using the Telnet protocol. It's primarily used for remote command-line login and execution.

Here's how you can use it:

[ ]:
```
telnet hostname port
```

In this command, replace `hostname` with the name or IP address of the host you want to connect to, and `port` with the port number. If you omit the port number, `telnet` will use the default Telnet port, which is 23.

Once connected, you can type commands at the `telnet` prompt, and they will be executed on the remote host. The output of the commands will be displayed in your terminal.

Note that Telnet is not secure for transmitting sensitive information because it does not encrypt the data. It's recommended to use SSH (Secure Shell) instead of Telnet for remote login and execution. SSH provides the same functionality as Telnet, but it encrypts the data to prevent eavesdropping.

53 Which command can be used to stop the current executing process in the background?

In Unix-like operating systems, you can use the `kill` command to stop a process that is running in the background. Here's how you can do it:

[ ]:
```
kill PID
```

In this command, replace `PID` with the Process ID of the process you want to stop. The `kill` command will send the `TERM` signal to the process, which will cause the process to terminate.

If the process does not respond to the `TERM` signal, you can use the `KILL` signal, which cannot be ignored:

[ ]:
```
kill -9 PID
```

You can find the PID of a process with the `ps` command or in the output of the `top` command. The PID of the last background process can be retrieved with the special variable `$!` in the shell.

Remember that you need to have the necessary permissions to kill a process. If you are not the owner of the process, you may need to use `sudo`:

[ ]:
```
sudo kill PID
```

This command will prompt you for your password, and then it will kill the process.

54 What is the basic purpose of 'find' command?

The `find` command in Unix-like operating systems is used to search for files and directories in a directory hierarchy based on different criteria such as name, type, size, and modification time.

Here's a basic usage of the `find` command:

```
[ ]: find path -name pattern
```

In this command, replace `path` with the directory you want to search in, and `pattern` with the name of the file or directory you are looking for. The `-name` option tells `find` to search by name. The `pattern` can include wildcards such as `*` and `?`.

For example, to find all `.txt` files in the current directory and its subdirectories, you would type:

```
[ ]: find . -name "*.txt"
```

This command will print the paths of all `.txt` files to the terminal.

The `find` command has many other options for searching by type, size, modification time, and other criteria. It can also execute commands on the files it finds with the `-exec` option.

55 Which filter can be used to display first 10 lines of a file?

The `head` command in Unix-like operating systems can be used to display the first 10 lines of a file. Here's how you can use it:

```
[ ]: head filename
```

In this command, replace `filename` with the name of the file you want to display. The `head` command will print the first 10 lines of the file to the terminal.

If you want to display a different number of lines, you can use the `-n` option followed by the number of lines. For example, to display the first 5 lines of a file, you would type:

```
[ ]: head -n 5 filename
```

This command will print the first 5 lines of the file to the terminal.

56 What is the first operating system designed using a high level programming language?

The first operating system designed using a high-level programming language was Multics (Multiplexed Information and Computing Service). It was developed in the 1960s and was written in PL/I (Programming Language One). Multics was a significant innovation in the development of operating systems and influenced many systems that came after it, including Unix.

57 Explain the use of SUID permission bit on a file.

The SUID (Set User ID) permission bit on a file in Unix-like operating systems allows the file to be executed with the permissions of the file's owner, rather than the permissions of the user who is executing the file.

This is useful for programs that need to perform tasks that require higher privileges than those of the user. For example, the `passwd` command, which is used to change a user's password, needs to modify the `/etc/shadow` file that is owned by root and not writable by regular users. By setting the SUID bit on the `passwd` program, the system allows it to write to the `/etc/shadow` file even when it's invoked by a regular user.

You can set the SUID bit with the `chmod` command by using the `u+s` option:

[ ]:
```
chmod u+s filename
```

In this command, replace `filename` with the name of the file you want to set the SUID bit on.

To view the SUID bit, you can use the `ls -l` command. The SUID bit is displayed as an `s` in the owner's execute field. For example, a file with the SUID bit set would have permissions like `rwsr-xr-x`.

Remember that the SUID bit can be a security risk if it's set on a program that is not secure, because it allows the program to perform actions with the permissions of the owner, which is often root. Therefore, it should be used carefully and sparingly.

58 What is a zombie process?

A zombie process, in Unix-like operating systems, is a process that has completed execution (via the exit system call) but still has an entry in the process table. This is due to the parent process not having read the child's exit status via the wait system call.

The term "zombie" is used because the process is dead as it has completed execution, but it still has a presence in the process table.

Zombie processes don't consume any resources (like memory or CPU time), but they do take up a process ID (PID) which could eventually lead to exhaustion of PIDs in a system with many zombies.

The parent process is responsible for removing the zombie with a wait call. If the parent process doesn't do this, the zombie will be adopted by the init process (PID 1) which periodically calls wait to clean up its child processes, including zombies.

59 Are arrays supported in shell scripts?

Yes, arrays are supported in shell scripts, specifically in Bash (Bourne Again SHell) scripts, which is a common shell used in various Unix-like operating systems.

Here's an example of how you can define and use an array in a Bash script:

[ ]:
```bash
# Define an array
fruits=("apple" "banana" "cherry")

# Print all elements
echo "${fruits[@]}"

# Print first element
echo "${fruits[0]}"

# Print number of elements
echo "${#fruits[@]}"

# Update an element
fruits[1]="blueberry"
```

In this script, `fruits` is an array that contains three elements. The elements are accessed with the syntax `${fruits[index]}`, where `index` is the index of the element. The first element has an index of 0.

The special syntax `${fruits[@]}` is used to access all elements of the array, and `${#fruits[@]}` is used to get the number of elements in the array.

Note that not all shell scripting languages support arrays. For example, the original Bourne shell (sh) does not support arrays. If you need to use arrays in a shell script, make sure to use a shell that supports them, such as Bash, and start your script with the appropriate shebang, such as `#!/bin/bash`.

60 What is the role of book block?

It contains a program called MBR (Master boot record) which load the kernel as the part of the system booting

61 Which command can be used to create alternate name for an existing command?

In Unix-like operating systems, you can use the `alias` command to create an alternate name (alias) for an existing command. Here's how you can do it:

[ ]:
```bash
alias newname='command'
```

In this command, replace `newname` with the alias you want to create, and `command` with the command you want to alias. For example, to create an alias `l` for the command `ls -l`, you would type:

[ ]:
```bash
alias l='ls -l'
```

After running this command, you can type `l` in the terminal to execute `ls -l`.

Note that aliases created this way are temporary and will not persist after the current shell session ends. To create permanent aliases, you need to add the `alias` commands to your shell's startup file, such as `~/.bashrc` or `~/.bash_profile` for the Bash shell.

To remove an alias, you can use the `unalias` command:

```
[ ]: unalias newname
```

This command will remove the alias `newname`.

62 How can you suppress trailing new line for echo command?

In Unix-like operating systems, you can suppress the trailing newline in the `echo` command by using the `-n` option. Here's how you can do it:

```
[ ]: echo -n "Hello, world"
```

This command will print "Hello, world" to the terminal without a trailing newline, so the cursor will stay on the same line.

Note that the behavior of `echo` can vary between different shells and operating systems. In some shells, you may need to use the `printf` command to suppress the trailing newline:

```
[ ]: printf "Hello, world"
```

This command will also print "Hello, world" to the terminal without a trailing newline. The `printf` command is more portable and consistent than `echo`, so it's often recommended for complex scripts.

63 Distinguish between cmp and diff commands?

Both `cmp` and `diff` are Unix/Linux commands used for comparing files, but they serve slightly different purposes and provide different types of output.

1. `cmp` Command: The `cmp` command is used to compare two files byte by byte. It's a simpler tool that's useful when you just want to know if the files are identical or not. If the files are different, `cmp` will tell you the byte and line number where the first difference is found, but it won't tell you what the actual differences are.

   Here's an example of using `cmp`:

   ```
   cmp file1.txt file2.txt
   ```

2. `diff` Command: The `diff` command, on the other hand, is used to show the differences between two files line by line, in various formats. It's a more complex tool that's useful when you want to know what the specific differences are. `diff` can output the differences in a format that can be used as input to the `patch` command, which can apply the differences to one file to make it identical to the other.

   Here's an example of using `diff`:

   ```
   diff file1.txt file2.txt
   ```

In summary, use `cmp` when you just want to know if two files are identical or not, and use `diff` when you want to know what the specific differences are.

64 How can we redirect the output of one command to the another command?

In Unix-like operating systems, you can use a pipe (`|`) to redirect the output of one command to another command. This is known as "piping". Here's how you can do it:

[ ]:
```
command1 | command2
```

In this command, `command1` is the command whose output you want to redirect, and `command2` is the command you want to redirect the output to.

For example, to list the files in the current directory and then pass the list to the `grep` command to search for a specific file, you would type:

[ ]:
```
ls | grep filename
```

This command will print the names of all files that contain `filename` in their name.

Pipes can be chained together to redirect the output through multiple commands. For example, to list the files in the current directory, search for a specific file, and then count the number of matching files, you would type:

[ ]:
```
ls | grep filename | wc -l
```

This command will print the number of files that contain `filename` in their name.

65 What is the effect of 'cd ..' if your current working directory is root (/) ?

The `cd ..` command in Unix-like operating systems changes the current working directory to the parent directory of the current directory.

However, the root directory (`/`) is the topmost directory in the filesystem hierarchy, and it does not have a parent directory. Therefore, if your current working directory is the root directory, `cd ..` will have no effect: the current working directory will remain the root directory.

You can verify this by running the following commands:

[ ]:
```
cd /
cd ..
pwd
```

The `pwd` command prints the current working directory. After running these commands, it will print `/`, indicating that the current working directory is still the root directory.

66 What is a mount point?

A mount point in Unix-like operating systems is a directory in the filesystem where additional storage, such as a physical device or a filesystem, is attached.

When a storage device is mounted to a mount point, the filesystem on the storage device becomes accessible at the location of the mount point. This allows the system to read and write files on the storage device as if they were part of the system's own filesystem.

For example, if you have a USB drive with a filesystem on it, you could mount it at the directory `/mnt/usb`. After mounting, the files on the USB drive would be accessible at paths starting with `/mnt/usb`.

Mounting is performed with the `mount` command, and the current mount points can be viewed with the `mount` command with no arguments or the `df` command.

Here's an example of mounting a USB drive:

```
[ ]: mount /dev/sdb1 /mnt/usb
```

In this command, `/dev/sdb1` is the device file for the USB drive, and `/mnt/usb` is the mount point. After running this command, the files on the USB drive are accessible at `/mnt/usb`.

Remember that only root or users with appropriate permissions can mount and unmount filesystems.

67 Which is the environment variable which holds your home directory information?

The environment variable that holds your home directory information in Unix-like operating systems is `HOME`.

You can print the value of the `HOME` variable with the `echo` command:

```
[ ]: echo $HOME
```

This command will print the path to your home directory. For example, it might print `/home/username` for a user named `username`.

Environment variables like `HOME` are used by the shell and other programs to store information about the system and the user's preferences. The `HOME` variable is often used in shell scripts and commands to refer to the user's home directory.

68 How can you list hidden files too with ls command?

In Unix-like operating systems, you can use the `-a` or `--all` option with the `ls` command to list all files, including hidden files. Here's how you can do it:

```
[ ]: ls -a
```

This command will list all files in the current directory, including those whose names start with a dot (`.`), which are hidden files in Unix-like systems.

If you want to list all files with details like file permissions, owner, group, size, and modification time, you can use the `-l` option in addition to `-a`:

[ ]:
```
ls -la
```

This command will list all files in the current directory in long format, including hidden files.

69 Which command you can use to find the remaining disk free space?

In Unix-like operating systems, you can use the `df` command to display disk space usage for all mounted filesystems. Here's how you can do it:

[ ]:
```
df -h
```

The `-h` option makes the output human-readable by displaying sizes in KB, MB, GB, etc., instead of blocks.

The `df` command displays information such as the filesystem name, total space, used space, available space, percentage of space used, and the mount point of each filesystem.

If you want to check the disk space usage of a specific directory, you can use the `du` command:

[ ]:
```
du -sh /path/to/directory
```

In this command, replace `/path/to/directory` with the path to the directory you want to check. The `-s` option makes `du` display the total disk space usage of the directory, and the `-h` option makes the output human-readable.

70 What is the purpose of 'at' command?

The `at` command in Unix-like operating systems is used to schedule commands to be executed once at a later time.

Here's an example of how you can use the `at` command:

[ ]:
```
echo "ls -l" | at midnight
```

In this command, `ls -l` is the command to be executed, and `midnight` is the time when the command will be executed. The `echo` command is used to pass the command to `at` through a pipe.

The `at` command reads commands from standard input or a specified file, and it schedules them to be executed in a separate shell at the specified time.

The time can be specified in various formats, such as `now + 1 hour`, `noon`, `midnight`, `teatime` (which is 4 PM), or a specific date and time.

The `atq` command can be used to list the user's pending jobs, and the `atrm` command can be used to remove jobs.

Note that the `at` command may not be installed or enabled on all systems, and its use may be restricted by system policies.

71 If you are not logged in and the scheduled command produces a displayable output, how would you see it?

When you schedule a command using the `at` command in Unix-like operating systems, the output of the command is not displayed on the screen. Instead, it is mailed to the user who scheduled the command.

By default, the `at` command sends an email to the user's local mail account with the output of the command. You can check this mail using a command-line mail client such as `mail` or `mutt`, or a graphical mail client that supports local mailboxes.

Here's how you can check your mail with the `mail` command:

```
[ ]: mail
```

This command will start the `mail` program, which will display a list of messages in your mailbox. You can then read the messages with the commands provided by `mail`.

If you want the output to be sent to a different email address, you can redirect the output of the command to the `mail` command. For example:

```
[ ]: echo "ls -l | mail -s 'Output of ls -l' user@example.com" | at midnight
```

In this command, `ls -l` is the command to be executed, `user@example.com` is the email address where the output will be sent, and `midnight` is the time when the command will be executed. The `-s` option to `mail` specifies the subject of the email.

72 What is LILO w.r.t linux?

LILO stands for Linux Loader. It's a boot loader for Linux and was one of the first boot loaders capable of booting the Linux kernel.

A boot loader is a small program that runs whenever a computer is started and is responsible for loading the operating system into memory.

LILO does not depend on a specific file system, can boot an operating system from any disk that the BIOS can access, and can boot other operating systems as well.

One of the characteristics of LILO is that it's a static boot loader. This means that it needs to be run (or re-run) whenever the configuration file or the files it points to (like the Linux kernel) are changed. This is in contrast to other boot loaders like GRUB, which are dynamic and can read their configuration file at boot time.

While LILO was widely used in the early days of Linux, it has largely been replaced by more modern and flexible boot loaders like GRUB and systemd-boot. However, understanding LILO can

still be useful for understanding the boot process and for dealing with older systems that still use it.

73 What is the recommended size of swap space for a typical OS installation?

The recommended size of swap space can vary depending on the system's RAM, the workloads running on the system, and whether the system will use features like hibernation. However, here are some general guidelines:

- For systems with 2GB of RAM or less, a swap space equal to the amount of RAM is usually recommended.
- For systems with 2GB to 8GB of RAM, a swap space equal to the amount of RAM or half the amount of RAM might be sufficient.
- For systems with 8GB to 64GB of RAM, a swap space of at least 4GB is usually recommended.
- For systems with more than 64GB of RAM, the swap space might be determined by the system's workload and the amount of disk space available.

If the system will use hibernation, the swap space should be at least as large as the system's RAM, because the contents of RAM are saved to swap space during hibernation.

Remember that these are just guidelines, and the optimal swap space can vary greatly depending on the specific use case. It's also possible to add more swap space later if needed, either by resizing the swap partition or by adding a swap file.

74 How can you list all the directories using ls command?

In Unix-like operating systems, you can use the `ls` command with the `-d` and `*/` options to list all directories in the current directory. Here's how you can do it:

[ ]:
```
ls -d */
```

In this command, the `-d` option makes `ls` list directories themselves, not their contents, and `*/` is a pattern that matches all directories.

If you want to list all directories with details like directory permissions, owner, group, size, and modification time, you can use the `-l` option in addition to `-d`:

[ ]:
```
ls -ld */
```

This command will list all directories in the current directory in long format.

If you want to list all directories, including those in subdirectories, you can use the `find` command:

[ ]:
```
find . -type d
```

In this command, `.` is the starting directory (the current directory), `-type d` specifies that you want to find directories, and `find` lists all directories under the starting directory.

75 Which command can be used to display basic information about your server?

In Unix-like operating systems, you can use the `uname` command to display basic information about your server. Here's how you can do it:

```
[ ]: uname -a
```

The `-a` option makes `uname` display all system information, including the kernel name, hostname, kernel release, kernel version, machine hardware name, processor type, hardware platform, and operating system.

If you want to display more detailed information about the system's hardware, you can use the `lshw` command:

```
[ ]: sudo lshw
```

The `lshw` command lists hardware on your system, including information about the CPU, RAM, disk drives, network interfaces, and more. Note that `lshw` must be run as root to display all information.

If you want to display information about the system's CPU, you can use the `lscpu` command:

```
[ ]: lscpu
```

The `lscpu` command displays information about the system's CPU architecture, including the number of CPUs, cores, threads, CPU family, model, and more.

Note that these commands may not be installed or may have different options on all systems.

76 How can you count for a particular pattern occurrences in a file?

In Unix-like operating systems, you can use the `grep` command with the `-c` option to count the occurrences of a particular pattern in a file. Here's how you can do it:

```
[ ]: grep -c 'pattern' filename
```

In this command, replace `'pattern'` with the pattern you want to search for, and `filename` with the name of the file you want to search in.

The `-c` option makes `grep` count the lines that match the pattern. Note that this counts lines, not individual occurrences of the pattern. If a line contains the pattern multiple times, it will still be counted as one.

If you want to count individual occurrences of the pattern, even if multiple occurrences are on the same line, you can use `grep` with the `-o` option and pipe the output to `wc -l`:

```
[ ]: grep -o 'pattern' filename | wc -l
```

The `-o` option makes `grep` print each match on a new line, and `wc -l` counts the lines. This will count each occurrence of the pattern, even if multiple occurrences are on the same line.

77 Explain the purpose of nohup command.

The `nohup` command in Unix-like operating systems is used to run a command or process that continues running after the user has logged out. The name `nohup` stands for "no hangup".

When you start a command in the terminal, it's attached to your session. If you log out or your session is disconnected for some reason (like a network disruption), all the processes attached to your session are usually terminated.

The `nohup` command prevents this from happening. It starts a process that ignores the `SIGHUP` (hangup) signal, which means the process won't be terminated when the session is disconnected.

Here's an example of how you can use the `nohup` command:

[ ]:
```
nohup command &
```

In this command, replace `command` with the command you want to run. The `&` at the end of the command runs the process in the background.

The `nohup` command redirects the standard output and standard error of the command to a file called `nohup.out` in the current directory. If the user doesn't have write permissions for the current directory, the output is redirected to `$HOME/nohup.out`.

The `nohup` command is useful for running long-running processes on remote servers, as it allows the process to continue running even if the user is disconnected.

78 How are devices represented in UNIX/Linux?

In Unix and Linux systems, devices are represented as special files located in the `/dev` directory. Each device is associated with a file in this directory, and interacting with the file is equivalent to interacting with the device.

There are two types of device files:

1. **Character devices**: These devices are accessed as a stream of bytes, one byte at a time. Keyboards and mice are examples of character devices.
2. **Block devices**: These devices are accessed as blocks of bytes. Hard drives, USB drives, and CD-ROMs are examples of block devices.

Each device file has a major number and a minor number associated with it. The major number identifies the driver associated with the device. For example, all SCSI hard drives have the same major number. The minor number is used by the kernel to determine the specific device to access for a given driver.

For example, `/dev/sda` represents the first SCSI hard drive in the system, and `/dev/sda1` represents the first partition on that drive.

You can use the `ls -l` command to list the devices in the `/dev` directory and see their major and minor numbers. For example:

[ ]:
```
ls -l /dev/sda
```

This command will display information about the `/dev/sda` device, including its major and minor numbers.

79 What is inode?

An inode (index node) is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory. Each inode stores the attributes and disk block locations of the object's data.

File-system object attributes may include metadata (times of last change, access, modification), as well as owner and permission data.

Directories are also represented as inodes. A directory contains a list of entry names (files and other directories) and the corresponding inode numbers.

The inode data structure does not include the file name or the directory name, which are linked to the inode via a separate directory entry. There can be multiple directory entries (hard links) associated with a single inode.

You can view the inode of a file using the `-i` option with `ls` command:

[ ]:
```
ls -i filename
```

In this command, replace `filename` with the name of the file you want to check. The command will display the inode number of the file.

80 Which command can be used to execute a command/s repeatedly for the given schedule?

In Unix-like operating systems, the `cron` command is used to execute commands or scripts automatically at a specified time/date.

You can schedule tasks using the `crontab` (cron table) command. The `crontab` command creates a `crontab` file containing entries for each job you want to schedule. Each line of the `crontab` file represents a single job and follows a particular syntax.

Here's an example of how you can schedule a job with `crontab`:

[ ]:
```
crontab -e
```

This command opens the `crontab` file for the current user in the default text editor. You can then add a line for each job you want to schedule. Each line has the following format:

[ ]:
```
*     *     *   *    *        command to be executed
-     -     -   -    -
|     |     |   |    |
|     |     |   |    +----- day of the week (0 - 6) (Sunday=0)
|     |     |   +------- month (1 - 12)
```

```
|     |     +--------- day of the month (1 - 31)
|     +----------- hour (0 - 23)
+------------- min (0 - 59)
```

For example, if you want to run a script at 5 a.m every day, you would add the following line to your `crontab` file:

[ ]:
```
0 5 * * * /path/to/script.sh
```

In this command, replace `/path/to/script.sh` with the path to the script you want to run. The `0 5 * * *` part of the command specifies that the script should be run at 5 a.m. every day.

81 Which command can be used to change file access permission bits?

In Unix-like operating systems, you can use the `chmod` command to change the file access permission bits of a file or directory. Here's how you can do it:

[ ]:
```
chmod permissions filename
```

In this command, replace `permissions` with the permissions you want to set, and `filename` with the name of the file or directory you want to change the permissions of.

The `permissions` can be specified in several ways:

- As a three-digit octal number, where each digit represents the permissions for the user, group, and others, respectively. Each digit is the sum of 4 for read, 2 for write, and 1 for execute. For example, `chmod 755 filename` sets the permissions to `rwxr-xr-x`.
- As a symbolic expression, where `u` represents the user, `g` represents the group, `o` represents others, `a` represents all, `+` adds permissions, `-` removes permissions, and `=` sets permissions. For example, `chmod u+x filename` adds execute permission for the user.

Here are some examples:

[ ]:
```
chmod 755 filename  # Set permissions to rwxr-xr-x
chmod u+x filename  # Add execute permission for the user
chmod go-w filename # Remove write permission for group and others
```

In these commands, replace `filename` with the name of the file or directory you want to change the permissions of.

82 Which command can be used to rename a file/directory?

In Unix-like operating systems, you can use the `mv` command to rename a file or directory. Here's how you can do it:

[ ]:
```
mv oldname newname
```

In this command, replace `oldname` with the current name of the file or directory, and `newname` with the new name you want to give to the file or directory.

The `mv` command moves the file or directory from the old name to the new name, effectively renaming it. If the new name is in a different directory, the file or directory is also moved to that directory.

Here's an example:

[ ]:
```
mv myfile mynewfile
```

This command renames the file `myfile` to `mynewfile`. If `mynewfile` already exists, it will be overwritten, so use this command with caution.

83 What are the respective octal value for the permission bits r, w & x?

In Unix-like operating systems, the permission bits `r` (read), `w` (write), and `x` (execute) are represented by the following octal values:

- `r` (read): 4
- `w` (write): 2
- `x` (execute): 1

These values can be added together to represent multiple permissions. For example, read and write permission is `4 + 2 = 6`, and read, write, and execute permission is `4 + 2 + 1 = 7`.

These octal values are used for each of the three sets of permissions: user (owner), group, and others. For example, the permission set `rwxr-xr-x` can be represented in octal as `755`.

84 Which operator can be used to throw a process into background?

In Unix-like operating systems, you can use the `&` operator to run a process in the background. Here's how you can do it:

[ ]:
```
command &
```

In this command, replace `command` with the command you want to run. The `&` operator at the end of the command runs the process in the background.

When a process is run in the background, the terminal doesn't wait for the process to finish before returning to the command prompt. This allows you to continue using the terminal while the process runs.

Here's an example:

[ ]:
```
sleep 30 &
```

This command runs the `sleep 30` command in the background. The `sleep 30` command pauses for 30 seconds, but because it's run in the background, you can continue using the terminal immediately.

85 How can we list out all currently executing background processes?

In Unix-like operating systems, you can use the `jobs` command to list all currently executing background processes in your current shell session. Here's how you can do it:

```
[ ]: jobs
```

The `jobs` command displays the job number, current state and command of each background process. The state can be `Running` if the job is currently executing, `Stopped` if the job is paused, or `Done` if the job has completed.

If you want to see all processes running in the system, not just those in your current shell session, you can use the `ps` command:

```
[ ]: ps aux
```

The `ps` command displays information about all the running processes. The `aux` option tells `ps` to display all processes in the system (`a`), display processes for all users (`x`), and display additional information (`u`).

86 Which command can be used to know the terminal type?

In Unix-like operating systems, you can use the `tput` command with the `longname` argument to display the terminal type. Here's how you can do it:

```
[ ]: tput longname
```

This command displays a description of the terminal type.

You can also use the `echo` command with the `TERM` environment variable to display the terminal type:

```
[ ]: echo $TERM
```

This command displays the value of the `TERM` environment variable, which is set to the terminal type.

The terminal type is used by applications to determine how to interact with the terminal. Different terminal types may support different features, so some applications may behave differently depending on the terminal type.

87 What is an internal command?

An internal command is a command that is built into the shell itself. These commands are executed directly by the shell, without launching a separate program.

Internal commands are typically basic commands that are used frequently, such as `cd` to change directories, `echo` to print text, `set` to set environment variables, and `exit` to exit the shell.

Because internal commands are built into the shell, they are typically faster to execute than external commands, which require launching a separate program. They also have access to the internal workings of the shell, so they can do things that external commands can't, like changing the shell's environment or controlling job execution.

You can use the `type` command to determine whether a command is internal or external:

```
[ ]: type command
```

In this command, replace `command` with the name of the command you want to check. The `type` command will tell you whether the command is internal or external.

88 Which command can be used by the administrator to bring the system into single user mode?

In Unix-like operating systems, the `init` or `telinit` command can be used by the administrator to change the runlevel of the system. To bring the system into single user mode, you can use the `init` or `telinit` command with the `1` or `s` argument. Here's how you can do it:

```
[ ]: init 1
```

or

```
[ ]: telinit 1
```

or

```
[ ]: init s
```

or

```
[ ]: telinit s
```

These commands change the runlevel to 1 or 's', which is single user mode. In single user mode, the system boots into a superuser shell without launching any services or network interfaces, which is useful for system maintenance.

Note that you need to have root privileges to use the `init` or `telinit` command. If you're not logged in as root, you can use the `sudo` command to run the `init` or `telinit` command with root privileges:

```
[ ]: sudo init 1
```

or

```
[ ]: sudo telinit 1
```

or

```
[ ]: sudo init s
```

or

```
[ ]: sudo telinit s
```

In these commands, replace `sudo` with the command to run as root, and `init` or `telinit` with the command to change the runlevel.

89 Which command can be used to write onto other currently logged in user's console output terminal.

In Unix-like operating systems, you can use the `write` command to send a message to another user's terminal. Here's how you can do it:

```
[ ]: write username tty
```

In this command, replace `username` with the username of the user you want to send a message to, and `tty` with the name of the terminal you want to send the message to.

After you enter the `write` command, you can type your message. Press `Ctrl+D` or `Ctrl+C` to end the message.

The `write` command sends the message to the specified user's terminal, interrupting whatever they are doing. The user can disable messages from the `write` command by using the `mesg` command with the `n` option:

```
[ ]: mesg n
```

This command disables messages from the `write` command.

Note that you need to have write permission to the other user's terminal to use the `write` command. If you don't have write permission, you can use the `sudo` command to run the `write` command with root privileges:

```
[ ]: sudo write username tty
```

In this command, replace `sudo` with the command to run as root, `write` with the command to send a message, `username` with the username of the user you want to send a message to, and `tty` with the name of the terminal you want to send the message to.

90 What is UMASK?

UMASK, or 'user file-creation mode mask', is a command in Unix and Unix-like operating systems that sets the default permissions for newly created files and directories. It's a three or four digit octal number.

The system starts with a default set of permissions when a file or directory is created and the value of UMASK is used to modify this default set. The UMASK value is subtracted from the system's default permissions to arrive at the default permissions for the new file or directory.

For example, if the system's default permissions are `666` for files (read and write for owner, group, and others) and `777` for directories (read, write, and execute for owner, group, and others), and the UMASK value is `022`, then:

- The default permissions for new files would be `644` (read and write for owner, read for group and others).
- The default permissions for new directories would be `755` (read, write, and execute for owner, read and execute for group and others).

You can use the `umask` command to view or set the UMASK value:

[ ]:
```
umask       # Display the current UMASK value
umask 022   # Set the UMASK value to 022
```

Note that the UMASK value is specific to each user's environment and can be set in the user's shell profile.

91 What is the default value of UMASK?

The default value of UMASK can vary depending on the system and its configuration. However, a common default UMASK value in many Unix-like systems is `022`.

With a UMASK value of `022`, the default permissions for new files will be `644` (read and write for the owner, and read for the group and others), and the default permissions for new directories will be `755` (read, write, and execute for the owner, and read and execute for the group and others).

You can check the current UMASK value in your session by running the `umask` command with no arguments:

[ ]:
```
umask
```

This command will display the current UMASK value.

To set a new UMASK value for the current session, you can use the `umask` command followed by the new value:

[ ]:
```
umask 022
```

This command sets the UMASK value to `022`. Note that this change only affects the current session. To set a new default UMASK value, you would typically add the `umask` command to a shell startup file like `~/.bashrc` or `~/.profile`.

92 Expand UFS in context of file system?

UFS stands for Unix File System. It's a file system used by Unix and Unix-like operating systems. UFS is a derivative of the original Berkeley Fast File System (FFS) from the BSD branch of Unix.

UFS manages the storage and retrieval of data on storage devices, like hard drives. It includes features like metadata, direct and indirect block allocation, and optional journaling.

While UFS is not as commonly used today as it once was, it's still used in some Unix-like systems, including some versions of BSD, and it's the default file system for Solaris, a Unix operating system developed by Sun Microsystems.

93 Expand JFS in context of file system?

JFS stands for Journaled File System. It is a high-performance, 64-bit journaling file system created by IBM. There are actually two types of JFS file systems:

1. JFS (often referred to as JFS1): This was originally developed for IBM's AIX operating system.
2. JFS2 or JFS for Linux: This is a reimplementation of JFS for the Linux operating system, with some enhancements like dynamic inode allocation and extents-based space management.

The key feature of a journaled file system like JFS is that it keeps a log, or journal, of what activity has taken place in the main data areas of the disk. If a crash or power failure occurs, this journal can be used to restore the file system to a consistent state without needing to check the entire file system, which can significantly speed up the recovery time after a crash.

94 Who designed linux OS?

The Linux operating system was designed by Linus Torvalds. He started the development as a personal project while he was a student at the University of Helsinki in Finland. The initial release of the Linux kernel, which is the core of the operating system, was in 1991. Since then, it has been developed and maintained by a community of developers from around the world under the coordination of Torvalds.

95 How can you display n

It seems like your question got cut off. If you're asking how to display the value of a variable `n` in a shell script, you can do it using the `echo` command:

[ ]:
```
n=10
echo $n
```

In this script, the first line sets the value of `n` to 10, and the second line displays the value of `n`.

If you're asking something else, could you please provide more details?

96 Name three looping constructs provided by shell.

Shell scripting provides several constructs for looping, which allow you to repeat a block of commands multiple times. Here are three of them:

1. `for` loop: This loop repeats a block of commands for each item in a list.

[ ]:
```
for i in 1 2 3
do
```

```
    echo $i
done
```

2. `while` loop: This loop repeats a block of commands as long as a certain condition is true.

[ ]:
```
i=1
while [ $i -le 3 ]
do
    echo $i
    i=$((i+1))
done
```

3. `until` loop: This loop repeats a block of commands until a certain condition is true.

[ ]:
```
i=1
until [ $i -gt 3 ]
do
    echo $i
    i=$((i+1))
done
```

In these examples, the `echo` command is used to print the value of the variable `i`. The `for` loop prints the numbers 1, 2, and 3. The `while` loop prints the numbers 1, 2, and 3 as long as `i` is less than or equal to 3. The `until` loop prints the numbers 1, 2, and 3 until `i` is greater than 3.

97 In which directory by default user home directories are created?

In Unix-like operating systems, user home directories are typically created in the `/home` directory by default.

For example, if you have a user named `username`, their home directory would typically be `/home/username`.

This is a convention and it can be changed by system administrators. For example, in some enterprise or server setups, home directories might be located on a separate partition or network share.

For the root user, the home directory is typically `/root`, not `/home/root`.

98 What is the default working directory of administrator?

The default working directory for the administrator, also known as the root user, in Unix-like operating systems is typically `/root`.

When the root user logs in or opens a new shell session, the current working directory is set to `/root` unless configured otherwise. This is separate from the `/home` directory where other users' home directories are typically located.

Please note that it's generally recommended to use the root user sparingly and for administrative tasks only, due to the potential risks associated with its elevated privileges. For regular tasks, it's better to use a standard user account and elevate privileges as needed using tools like `sudo`.

99 What is the login name of the administrator?

The login name of the administrator in Unix-like operating systems is typically `root`.

The `root` user, also known as the superuser, has full administrative privileges and can perform tasks that are restricted for other users, such as modifying system files, installing software, and changing ownership of files.

However, it's generally recommended to use the `root` account sparingly and only for tasks that require administrative privileges, due to the potential risks associated with its elevated privileges. For regular tasks, it's better to use a standard user account and elevate privileges as needed using tools like `sudo`.

100 Thank You!