FileSystemBasics

April 15, 2024

1 File System Basics?

In shell scripting, you often need to interact with the file system to read from or write to files, change directories, check if a file exists, etc. Here are some basic concepts and commands:

1. **Directories**: Directories, also known as folders, are used to organize files. The `cd` command changes the current directory. The `pwd` command prints the current directory. The `mkdir` command creates a new directory.

   ```
   cd /path/to/directory  # Change to a specific directory
   pwd  # Print the current directory
   mkdir new_directory  # Create a new directory
   ```

2. **Files**: Files hold data. The `touch` command creates a new file. The `cp` command copies files. The `mv` command moves or renames files. The `rm` command deletes files.

   ```
   touch new_file.txt  # Create a new file
   cp source_file.txt destination_file.txt  # Copy a file
   mv old_name.txt new_name.txt  # Rename a file
   rm file_to_delete.txt  # Delete a file
   ```

3. **File Contents**: The `cat` command prints the contents of a file. The `>`, `>>`, and `<` operators redirect output to a file or input from a file. The `grep` command searches for a pattern in a file.

   ```
   cat file.txt  # Print the contents of a file
   echo "Hello, world!" > file.txt  # Write to a file (overwriting existing contents)
   echo "Another line" >> file.txt  # Append to a file
   grep "pattern" file.txt  # Search for a pattern in a file
   ```

4. **File Permissions**: Files and directories have permissions that control who can read, write, and execute them. The `chmod` command changes permissions. The `chown` command changes the owner.

   ```
   chmod 755 script.sh  # Change permissions of a file (read, write, execute for owner, read
   chown user:group file.txt  # Change the owner and group of a file
   ```

5. **File Tests**: You can check certain properties of files with test commands. For example, `-f` checks if a file exists and is a regular file. `-d` checks if a directory exists.

   ```
   if [ -f file.txt ]; then echo "File exists"; fi  # Check if a file exists
   if [ -d /path/to/directory ]; then echo "Directory exists"; fi  # Check if a directory exi
   ```

These are just the basics. The shell provides many more commands and features for working with the file system.

2 Directory Structure?

In Unix-like operating systems, such as Linux, the file system is organized in a hierarchical directory structure. Here's a brief overview of the standard directories and their purposes:

1. **`/` (Root)**: This is the top level of the file system. All other directories are subdirectories of the root directory.
2. **`/bin`**: This directory contains essential binary executables (commands) that need to be available in single user mode, such as `ls`, `cat`, `mv`, `rm`, etc.
3. **`/boot`**: This directory contains the files needed to boot the system, such as the Linux kernel and the boot loader.
4. **`/dev`**: This directory contains device files, which represent hardware and software devices.
5. **`/etc`**: This directory contains system-wide configuration files.
6. **`/home`**: This directory contains the home directories of individual users.
7. **`/lib`**: This directory contains shared library files that are required by system programs.
8. **`/opt`**: This directory is for optional or add-on software packages.
9. **`/sbin`**: This directory contains system binaries, i.e., commands essential for booting, restoring, recovering, and/or repairing the system in addition to the commands in `/bin`.
10. **`/tmp`**: This directory is used for temporary files. It is often cleared out on reboot.
11. **`/usr`**: This directory contains user utilities and applications. It's a secondary hierarchy for read-only user data and contains the majority of (multi-)user utilities and applications.
12. **`/var`**: This directory contains variable data files such as logs, databases, websites, and email, among others.
13. **`/proc`**: This is a virtual filesystem that provides process and kernel information to userspace.
14. **`/sys`**: This is another virtual filesystem that provides a view into the kernel's view of the system's hardware.

These are the standard directories in a Unix-like file system. Depending on the specific distribution and the installed packages, a system may have more directories.

3 Navigating the File System?

Here are some of the most commonly used commands for navigating the file system in Unix-like operating systems:

1. **`cd` (Change Directory)**: Changes the current directory.

   ```
   cd /path/to/directory  # Change to a specific directory
   cd ..  # Go up to the parent directory
   ```

```
cd ~  # Go to the home directory
cd -  # Go to the previous directory
cd  # Without any arguments, cd takes you to your home directory
```

2. **`ls` (List)**: Lists files and directories in the current directory.

   ```
   ls  # List files in the current directory
   ls /path/to/directory  # List files in a specific directory
   ls -l  # List files in long format, showing detailed information
   ls -a  # List all files, including hidden ones
   ```

3. **`pwd` (Print Working Directory)**: Prints the path of the current directory.

   ```
   pwd  # Print the current directory
   ```

4. **`pushd` and `popd`**: These commands are used to manipulate the directory stack. This is a list of directories you can push onto and pop off, which can be useful for quickly navigating between directories.

   ```
   pushd /path/to/directory  # Push a directory onto the stack and cd to it
   popd  # Pop a directory off the stack and cd to it
   ```

5. **`find`**: Searches for files and directories based on a given expression and performs an action on each matched file.

   ```
   find . -name "*.txt"  # Find all .txt files in the current directory and its subdirectorie
   ```

6. **`file`**: Determines the type of a file.

   ```
   file myfile.txt  # Determine the type of myfile.txt
   ```

7. **`locate`**: Finds files by name.

   ```
   locate myfile.txt  # Find files named myfile.txt
   ```

Remember, you can always use the `man` command to get more information about any command, like `man cd` or `man ls`.

Navigating the file system in a Unix-like operating system is done primarily through the `cd` (change directory) command. Here are some basic commands and concepts:

1. **Change to a Specific Directory**: You can change to a specific directory by providing the path to the `cd` command. Paths can be absolute (starting from the root directory, `/`) or relative (starting from the current directory).

   ```
   cd /path/to/directory  # Change to a specific directory
   ```

2. **Home Directory**: The `~` character represents the home directory of the current user. If you run `cd` without any arguments, it will take you to your home directory.

   ```
   cd ~  # Change to the home directory
   cd  # Also changes to the home directory
   ```

3. **Parent Directory**: The `..` symbol represents the parent of the current directory. You can use this to move up in the directory hierarchy.

   ```
   cd ..  # Change to the parent directory
   ```

4. **Current Directory**: The `.` symbol represents the current directory. This is often used when running scripts or binaries located in the current directory.

   ```
   ./script.sh  # Run a script in the current directory
   ```

5. **Previous Directory**: The `-` symbol represents the previous directory. You can use this to quickly switch back and forth between two directories.

   ```
   cd -  # Change to the previous directory
   ```

6. **List Files**: The `ls` command lists the files in the current directory. You can provide a path to list the files in a different directory.

   ```
   ls  # List files in the current directory
   ls /path/to/directory  # List files in a specific directory
   ```

7. **Print Working Directory**: The `pwd` command prints the path of the current directory.

   ```
   pwd  # Print the current directory
   ```

These commands and concepts are fundamental to navigating the file system in a Unix-like operating system.

4 `df` and `du` command?

`df` and `du` are two important commands in Unix-like operating systems for disk usage analysis.

1. **`df` (Disk Free)**: This command displays the amount of disk space used and available on the file system. By default, `df` displays the disk space in 1K blocks. However, you can change the format to make it more readable.

   ```
   df  # Display disk usage of all mounted filesystems
   df -h  # Display disk usage in human-readable format (e.g., KB, MB, GB)
   df /path/to/directory  # Display disk usage of the filesystem containing the specified dir
   ```

2. **`du` (Disk Usage)**: This command estimates and displays the disk space used by files, directories, or the entire file system. Unlike `df`, which gives the usage of entire file systems, `du` can give the size of individual files/directories.

   ```
   du /path/to/directory  # Display the disk usage of a specific directory
   du -h /path/to/directory  # Display the disk usage in a human-readable format
   du -sh /path/to/directory  # Display a summary of the total size in a human-readable forma
   du -a /path/to/directory  # Display the disk usage of all files and directories recursivel
   ```

In both commands, the `-h` option stands for "human-readable", and it displays the sizes in human-readable format (KB, MB, GB, etc.).

5 Mounting System ?

In Unix-like operating systems, mounting is the process by which the operating system makes files and directories on a storage device (such as hard drive, SSD, CD-ROM, or network share) available for users to access via the system's directory tree.

Here's a basic overview of how to mount and unmount file systems:

1. **Mount a File System**: The `mount` command is used to mount a file system. You need to specify the device to mount and the mount point, which is the directory where the file system will be accessed.

   ```
   mount /dev/sdb1 /mnt/my_drive  # Mount the device /dev/sdb1 at the directory /mnt/my_drive
   ```

   You can also specify the file system type with the `-t` option, and additional options with the `-o` option.

   ```
   mount -t ext4 /dev/sdb1 /mnt/my_drive  # Mount an ext4 file system
   mount -o ro /dev/sdb1 /mnt/my_drive  # Mount a file system as read-only
   ```

2. **Unmount a File System**: The `umount` command is used to unmount a file system. You need to specify the mount point or the device to unmount.

   ```
   umount /mnt/my_drive  # Unmount the file system at /mnt/my_drive
   umount /dev/sdb1  # Unmount the device /dev/sdb1
   ```

3. **View Mounted File Systems**: The `mount` command without any arguments displays all mounted file systems. The `df` command also displays mounted file systems along with their disk usage.

   ```
   mount  # Display all mounted file systems
   df -h  # Display disk usage of all mounted file systems
   ```

4. **Automatically Mount File Systems**: File systems can be automatically mounted at boot by adding them to the `/etc/fstab` file. Each line in this file represents a file system to be mounted, with fields for the device, mount point, file system type, options, dump, and pass.

   ```
   /dev/sdb1 /mnt/my_drive ext4 defaults 0 0  # Add this line to /etc/fstab to mount /dev/sdb
   ```

Remember, mounting and unmounting file systems usually requires root privileges, so you may need to use `sudo` with these commands. Always ensure to safely unmount a file system before physically disconnecting a device to prevent data loss.

6 User and Group Quotas?

In Unix-like operating systems, quotas are a standard feature used to limit the amount of disk space a user or group can use. They help manage resources in a multi-user environment, preventing a single user or group from consuming all the disk space.

There are two types of quotas:

1. **User Quotas**: These apply to specific users. Each user has a limit on the amount of disk space they can use.

2. **Group Quotas**: These apply to all members of a group. The total disk space used by all members of the group is limited.

Quotas work by setting two limits:

- **Soft Limit**: This is the maximum amount of disk space a user or group can use. If this limit is reached, the user is only given a grace period to reduce their disk usage.

- **Hard Limit**: This is the absolute limit of disk space a user or group can use. Once this limit is reached, no further disk space can be used.

Here's a basic overview of how to set up quotas:

1. **Install Quota Tools**: Most distributions have a package for quota tools that can be installed with the package manager.

   ```
   sudo apt-get install quota  # On Debian/Ubuntu
   sudo yum install quota  # On CentOS/RHEL
   ```

2. **Enable Quotas on File System**: Edit the `/etc/fstab` file to enable quotas on a file system. Add `usrquota` for user quotas, `grpquota` for group quotas, or both.

   ```
   /dev/sda1 /home ext4 defaults,usrquota,grpquota 0 1  # Enable user and group quotas on /ho
   ```

   Then remount the file system for the changes to take effect.

   ```
   sudo mount -o remount /home
   ```

3. **Create Quota Database**: The `quotacheck` command scans a file system for disk usage and creates a quota database.

   ```
   sudo quotacheck -cug /home  # Create a user and group quota database for /home
   ```

4. **Set Quotas**: The `edquota` command is used to edit user or group quotas.

   ```
   sudo edquota username  # Set a quota for a user
   sudo edquota -g groupname  # Set a quota for a group
   ```

5. **Check Quotas**: The `quota` command displays the current quota usage and limits for a user or group.

   ```
   quota -u username  # Check the quota for a user
   quota -g groupname  # Check the quota for a group
   ```

6. **Enforce Quotas**: The `quotaon` command is used to enable quota enforcement.

   ```
   sudo quotaon /home  # Enable quotas on /home
   ```

Remember, managing quotas usually requires root privileges, so you may need to use `sudo` with these commands.

7 How can I set user and group quotas for disk space usage?

To set user and group quotas for disk space usage in Unix-like operating systems, you need to follow these steps:

1. **Install Quota Tools**: Most distributions have a package for quota tools that can be installed with the package manager.

   ```
   sudo apt-get install quota  # On Debian/Ubuntu
   sudo yum install quota  # On CentOS/RHEL
   ```

2. **Enable Quotas on File System**: Edit the `/etc/fstab` file to enable quotas on a file system. Add `usrquota` for user quotas, `grpquota` for group quotas, or both.

```
/dev/sda1 /home ext4 defaults,usrquota,grpquota 0 1  # Enable user and group quotas on /ho
```

Then remount the file system for the changes to take effect.

```
sudo mount -o remount /home
```

3. **Create Quota Database**: The `quotacheck` command scans a file system for disk usage and creates a quota database.

```
sudo quotacheck -cug /home  # Create a user and group quota database for /home
```

4. **Set Quotas**: The `edquota` command is used to edit user or group quotas. This command opens an editor where you can set the soft and hard limits for the number of disk blocks and inodes.

```
sudo edquota username  # Set a quota for a user
sudo edquota -g groupname  # Set a quota for a group
```

In the editor, you might see something like this:

```
Disk quotas for user username (uid 1001):
  Filesystem                   blocks       soft       hard     inodes     soft     hard
  /dev/sda1                     16          0          0          4        0        0
```

You can then modify the soft and hard limits for blocks (disk space) and inodes (number of files). For example, to set a soft limit of 10000 blocks and a hard limit of 12000 blocks, you would change it to:

```
Disk quotas for user username (uid 1001):
  Filesystem                   blocks       soft       hard     inodes     soft     hard
  /dev/sda1                     16          10000      12000      4        0        0
```

5. **Check Quotas**: The `quota` command displays the current quota usage and limits for a user or group.

```
quota -u username  # Check the quota for a user
quota -g groupname  # Check the quota for a group
```

6. **Enforce Quotas**: The `quotaon` command is used to enable quota enforcement.

```
sudo quotaon /home  # Enable quotas on /home
```

Remember, managing quotas usually requires root privileges, so you may need to use `sudo` with these commands.

8 Thank You!