PRCMP PL06
The Unix shell: file utilities

António Barros, Bertil Marques, Luis Ferreira, Nuno Morgado, Nuno Pereira
October 2024

The Unix shell is a powerful command-line interface that allows users to interact directly with the operating system. Unlike graphical user interfaces (GUIs), which rely on windows and icons, the shell uses text commands to perform tasks. This direct communication offers a high degree of control and flexibility, making it popular among developers, system administrators, and power users. Through the shell, you can navigate directories, manipulate files, and run programs with speed and precision.

One of the shell's greatest strengths is its ability to execute complex tasks by combining simple commands. You can chain commands together using pipes and redirection, allowing the output of one command to become the input of another. This makes the shell not only a tool for everyday file management but also a robust platform for automating workflows and processing data. Users can write shell scripts to execute a series of commands in sequence, enabling repetitive tasks to be handled efficiently.

Additionally, the Unix shell supports a variety of built-in utilities for file manipulation, text processing, and system monitoring. Whether you are compressing files, searching for text patterns with 'grep', or checking system resource usage with 'top', the shell provides a suite of tools designed to handle specific tasks with minimal effort. Moreover, different flavours of Unix, like Linux and macOS, offer various shells such as Bash, Zsh, and Fish, each with its own set of features while maintaining core functionalities. Mastering the shell unlocks a range of possibilities for powerful system utilisation.

1 File utilities

Learning Unix file utility commands is crucial for effectively managing files and directories on a Unix-based system. Commands like '`cp`' for copying, '`mv`' for moving, '`rm`' for deleting, and '`ls`' for listing files are fundamental for organising and controlling data. These commands allow users to quickly navigate the file system, manipulate files, and perform essential tasks with precision. By mastering these utilities, users can streamline their workflow, automate simple tasks, and handle large volumes of files efficiently. Understanding these basic commands is a key step in becoming proficient with the Unix shell and gaining more control over the system's file management.

The Unix file utility commands are:

- `ls` – Lists files and directories.
- `cd` – Changes the current directory.
- `pwd` – Displays the current working directory.
- `cp` – Copies files or directories.
- `mv` – Moves or renames files or directories.
- `rm` – Removes (deletes) files or directories.
- `mkdir` – Creates a new directory.
- `rmdir` – Removes an empty directory.
- `touch` – Creates an empty file or updates the timestamp of an existing file.

- `cat` – Concatenates and displays file contents.
- `more` / `less` – Views file contents page by page.
- `head` – Displays the first few lines of a file.
- `tail` – Displays the last few lines of a file.
- `chmod` – Changes file permissions.
- `chown` – Changes file ownership.
- `ln` – Creates links (hard or symbolic) between files.
- `stat` – Displays detailed information about a file or directory.
- `du` – Shows disk usage for files or directories.
- `df` – Displays available disk space on file systems.
- `file` – Determines the type of a file.

Questions

1. Open a terminal and login. Your initial working directory will be your home directory. Use the `pwd` command to see the absolute path to the working directory.
2. The `man` command provides information about the various shell commands. See information on some commands by executing the following commands. Press the 'q' key to exit the `man` program.
   (a) `man ls`
   (b) `man cp`
   (c) `man rm`
   (d) `man man`
3. List the contents of the directory with the `ls` command.
4. Now list all content (including hidden files) in long format.
5. Create a new text file, with the `nano` editor, using the following command: `nano document.txt`
   Type a few lines of text and then press CTRL-X to exit (press 'Y' to save, and ENTER to confirm the file name).
6. Confirm that the new file is saved, and check its disk size.
7. Create a duplicate of the file, named `document.txt.old` and check the result.
8. Open the document.txt file again with the `nano` editor and add a few more lines of text. Exit and check that the two documents are different sizes.
9. Change the name of `document.txt` to `my_text.txt` and check the result.
10. Create a new directory called `docs` in the working directory and check the result with the `ls` command in long format. How does `ls` distinguish files from directories?
11. Create the new `table.csv` file with the nano editor. Write a few lines and save the result.
12. List only files whose name ends in ".txt".
13. List only files whose name contains the expression "txt".
14. Move files whose name ends in ".txt" to the `docs` directory. Check that the file is located in the `docs` directory, without changing the working directory.
15. Change the working directory to the `docs` directory. List the contents of the current working directory.
16. Create two more text files with the nano editor, with whatever names you want.
17. Explain the result of each of these commands:

   (a) `ls *.txt`
   (b) `ls ?.txt`
   (c) `ls [a-c]*`
   (d) `ls [a-z]*`
   (e) `ls [aeiouAEIOU]*`
   (f) `ls ???t*`

18. Change the mode of files whose name ends in ".txt", removing all permissions for all users. Confirm that the permissions have been removed.
19. Run the `cat my_text.txt` command line to view the file contents. How do you explain the result?
20. Assign read permissions to all users, on all files whose name ends in ".txt". Try again to view the contents of the file `my_text.txt`.
21. Change the working directory back to your home directory. Confirm which directory you are in.
22. List the contents of the `docs` directory. Remove the owner's read permission on this directory Try again to list the directory contents. How do you explain what happened?
23. Remove execute permission for the owner of the `docs` directory. Try changing the working directory to `docs`. How do you explain what happened?
24. Delete all files whose name ends in ".old". Confirm the operation result.
25. Select a file and use the `stat` command to display detailed information about it.
26. Select a directory and use the `stat` command to display detailed information about it.
27. Select a file and use the `file` command to identify its type. What are the differences between the `file` and `stat` commands?
28. Check if the `file` command also works with directories. What is the result?
29. The `du` command shows the disk usage for files and directories.
    (a) Select a file and check its disk usage. The output shows how many *blocks* are used by the file.
    (b) Check again the disk usage of the file but in a human-readable format. Does the disk usage equals the size of the file displayed by the `ls` command? Can you speculate about your observation?
    (c) Check the total disk usage of directory `/bin`, in a human-readable format.
    (d) Display the disk usage of each file and subdirectory in `/bin`.
    (e) Display the disk usage for each `.log` file in the `/var/log` directory, along with the total disk usage of those files.

2 Solutions

1. `pwd`
2. (a) `man ls`

   Press the 'q' key to exit the `man` program.

   (b) `man cp`

   (c) `man rm`

   (d) `man man`
3. `ls`
4. `ls -al`
5. `nano document.txt`
6.
7.
8.
9.
10.
11.
12.
13.
14.

15.
    ```
    cd docs
    ls
    ```
16. Create some additional files, using the same process as above, giving arbitrary names.
17. (a) Lists all entries whose name ends in ".txt".

    (b) Lists all entries whose name is strictly one character followed by ".txt".

    (c)

    (d)

    (e)

    (f)

18.
    ```
    chmod 000 *.txt
    ls -l
    ```
19.

20.
    ```
    chmod ugo+r *.txt
    cat my_text.txt
    ```
21.
22. Read permission allows you to view the contents of a directory.
23.
24.
25.

26.

27.

28.

29. (a) `du somefile.txt`

    (b) `du -h somefile.txt`

    (c) `du -h /bin`

    (d) `du -h /bin/*`

    (e) `du -ch /var/log/*.log`