SHOW NOTES

@mjkabir Notes

https://shownotes.app/show/linux-basics

Basic Linux Cheatsheet 101

This is a list of basic Linux commands that beginning Linux devs and users should know by heart. My 9th-grader son is learning this as part of his cybersecurity education at home :)

AI REVIEW PASSED.

Basic File Manipulation Commands

Show current directory

$ pwd

Example:
/Users/kabir

Show files and sub-directories of the current directory

$ ls

Show files and sub-directories with more details

$ ls -l

Example:
drwxr-xr-x@ 9 kabir staff   288 Feb 6 13:46 jan
drwxr-x--- 106 kabir staff 3392 Mar 11 09:03 logs

Here, the first column shows file/dir permissions. It also shows which user (kabir) and group (staff) owns the file or directory. The size and create/modify time is also shown.

If you want to see all files (including the files and directories starting with dot), run:

$ ls -al

To go to a different directory

$ cd Downloads
$ pwd
/Users/kabir/Downloads

To return to the current user's home directory, they can run:

$ cd

Running cd command without any destination parameter will return the user to their home directory. This is a shortcut.

To move a file or directory to another name (rename)

$ mv file1 file2
$ mv dir1 dir2
$ mv file1 /some/place/else/

In UNIX/LINUX, files and directories are treated the same way.

To delete a file or directory:

$ rm filename
$ rm -rf dir

The **-rf** is needed to remove everything inside the dir, including other sub-directories.
The **-r** is for recursive delete, and **-f** forces the delete without asking to confirm every time.

To view the content of a text file:

$ cat filename

Never use cat on a binary file. If you cat a binary file on a shell terminal, you can accidentally execute some weird commands that the shell interprets from the content of the binary file and do great damage to your system.

To copy a file:
$ cp filename newfilename

To copy a direcotry:

$ cp -ra dir1 dir2

The -r is for recursive copy and -a is for maintaining the existing file ownership and permission.

To find the type of a file:
$ file filename

Example 1:
$ file README.TXT
README.TXT: ASCII text

Example 2:
$ file /usr/bin/nano
/usr/bin/nano: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV)

To create a directory:
$ mkdir newdir

To create a directory tree (with sub-directories)
$ mkdir -p dir1/subdir1/subdir2

Example 2:
$ file /usr/bin/nano
/usr/bin/nano: ELF 64-bit LSB pie executable, x86-64...
300 days 2 hrs ago

Special Files / Directories

There are two special files in Linux system that you need to know:

```
$ ls -a
. ..  .bash_history  .bash_logout  .bash_profile  .bashrc  .cshrc  .ssh  .tcshrc
```

Here, the first two files are very special. The single dot represents the current directory. The double dot represents the parent directory.

For example, if you want to go back to the parent directory,

```
$ cd ..
```

This will take you back to the parent level. Example:

```
 $ pwd
/root

$ cd ..

$ pwd
/
```

Here, the user was first in the /root directory, and when they ran the change directory (cd) command with the double dot as the destination, the shell took the user to the parent directory, which in this case is the actual root directory represented by the single / character.

Now, if you run the cd command with the single dot file, as shown below, you will not change to another directory as the single dot represents the current directory.

```
$ pwd
/Users/kabir
$ cd .
$ pwd
/Users/kabir
```

There are lots of good uses of the single dot directory, for example:

```
$ ./myprogram
```

This runs the program called **my program from the current directory,** as the ./ path indicates.
301 days 13 hrs ago

Basic File Editing from Terminal

Basic File Editing:

You have to use terminal-based file editors such as vi or improved vi called vim or nano editor, which are typical for macOS and Linux systems.

To edit a file using vim:

```
$ vim filename
```

This will open the file if it exists in the current directory. If you want to open the file from another directory:

```
$ vim /path/to/file
```

For example, to create a new file called readme.txt:

```
$ vim readme.txt
```

Mode Basics

Vim has several modes; the most commonly used are:

- Normal Mode: For navigating and manipulating the text
- Insert Mode: For inserting text
- Command Mode: For running commands

Switching Modes

- To enter Insert Mode from Normal Mode, press i
- To return to Normal Mode from Insert Mode, press Esc

Basic Editing Commands

- i - Enter insert mode to edit the text.
- Esc - Exit insert mode to go back to normal mode.
- :w - Save the file without exiting.
- :wq or ZZ - Save the file and exit Vim.
- :q - Quit Vim. If you've made changes, Vim will warn you.
- :q! - Quit without saving changes.

Navigating Text

- h - Move left
- j - Move down
- k - Move up
- l - Move right

Editing Text

- dd - Delete a line.
- yy - Yank (copy) a line.
- p - Paste below the cursor.
- u - Undo the last operation.
- Ctrl + r - Redo the last undo.

This is just a fundamental overview of Vim. There are countless commands and features, such as searching, replacing, file management, and more, which can enhance your editing efficiency. You can explore more advanced features as you get comfortable with the basics.
301 days 13 hrs ago

Working with Files and Directories

To find all files and directories inside a dir:

$ find directory

Example:
$ find /etc

To list the details of each of the files inside a directory using find:

$ find directory -ls

Example:
$ find /etc/ -ls

To find all the files and directories that you have permission to see:

$ find / -ls

To find a specific file in the entire system as root:

$ find / -type f -name "name of the file"

Example:
$ find / -type f -name "passwd"

This will find the files with the word "passwd" in their names. Example output:

/usr/bin/passwd
/etc/passwd
/etc/pam.d/passwd
/sys/fs/selinux/class/passwd/perms/passwd

To find a directory called shm you can:

$ find / -type d -name "shm"

Here is the output:
$ find / -type d -name "shm"
/sys/fs/selinux/class/shm
/dev/shm
300 days 2 hrs ago

File and Directory Permissions

Every file and directory (which are special files) in Unix/Linux/MacOS has three sets of permissions:

[owner]-[group]-[public]

Each set has three types of access:
r - read
w - write
x - execute

Example 1:
rwxr-----

Here, the owner has read, write, execute, the group has only read permission, and the public has no permission.

Example 2:
rwxrwx---

Here, the owner and the group have both read, written, and executed, while the public has none. Each file or directory can be owned by a single user or group.

For example:

drwxr-xr-x 3 ethan root 4096 Apr 27 00:58 python

Here, the Python directory is owned by a user called 'ethan' and a group called 'root'. Users are defined in /etc/passwd file. Groups are defined in /etc/group file. Only a root-privileged user can modify ownership. To change a file's ownership, the root user can run the following:

```
$ chown newuser file1
```

This will change file1's ownership to the new user.

To change all files and sub-directory ownership:

```
$ chown -R newuser  dir1
```

This will change the ownership of dir1 and its contents (files and sub-directories) to newuser. Any file/dir owner can change access permission for their files. For example:

```
$ chmod 777 README.TXT
```

This makes everyone in the system able to read, write, and execute the README.TXT file, which is a really bad idea.

In UNIX (Linux/MacOS), file/dir permissions can be understood as an octal number such as 000...777, where the most significant digit represents the owner's permission, the 2nd significant (middle) digit represents the group's permission, and the last or least significant digit represents the public or world permission.

For example:

$ chmod 700 filename

This will set the owner's read, write, and execute permissions, and others will have no access. Think of each octal digit as:

BIN. OCT PERMISSIONS
000 0 = no permission
001 1 = --x
010 2 = -w-
011 3 = -wx
100 4 = r--
101 5 = r-x
110 6 = rw-
111 7 = rwx

300 days 2 hrs ago

Manipulating Files using Tools

You can use cut to cut a piece of the file data. For example, say you have a file that has the following lines:

A B C D 1000
A B C D 2000
A B C D 3000
A B C D 4444
A B C D 9999
A B C D 1000

To get the 4th field (numbers) extracted (aka cut) from the file, you can run
$cat filename | cut -d' ' -f5
1000
2000
3000
4444
9999

To sort data in a text file, you can run:

$ sort filename

To sort and find uniq records in a file:

$ sort filename | uniq

To find how many duplicates are in a file:

$ sort filename | uniq -c

300 days 2 hrs ago