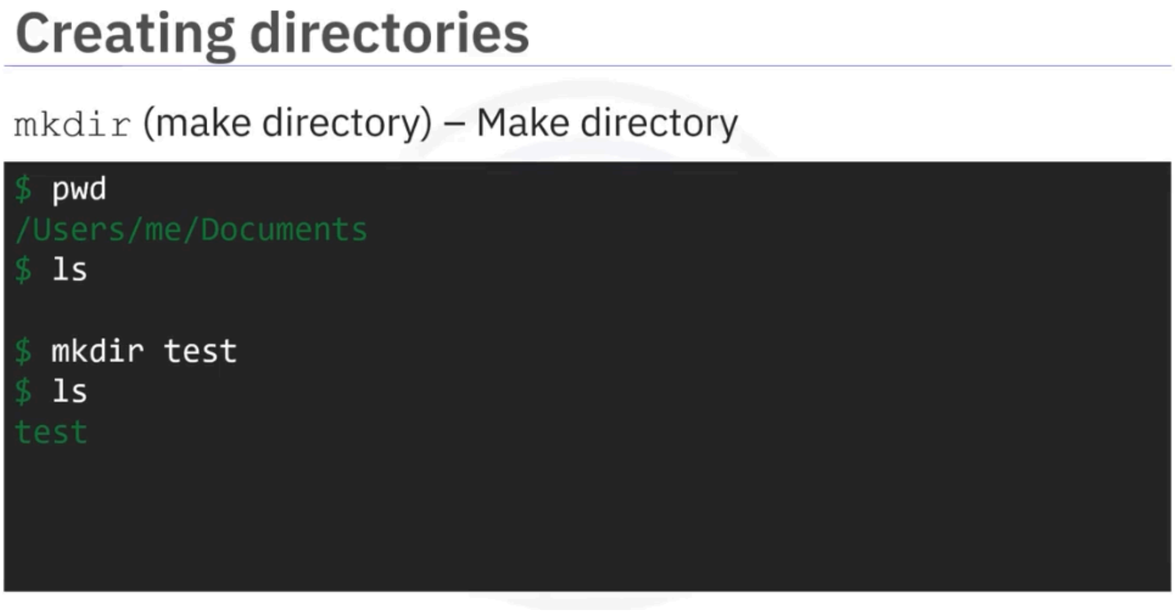
File and Directory Management Commands

File and directory management commands are like the tools in a toolbox that help you organize and control your files and folders on a computer. Imagine your computer is a big library, and each file is a book. You need to know how to create new books (files), remove old ones, and even move them around to different shelves (directories). For example, if you want to create a new folder to keep your school projects, you would use the command `mkdir school_projects`. This is like saying, "Please create a new shelf for my school projects."



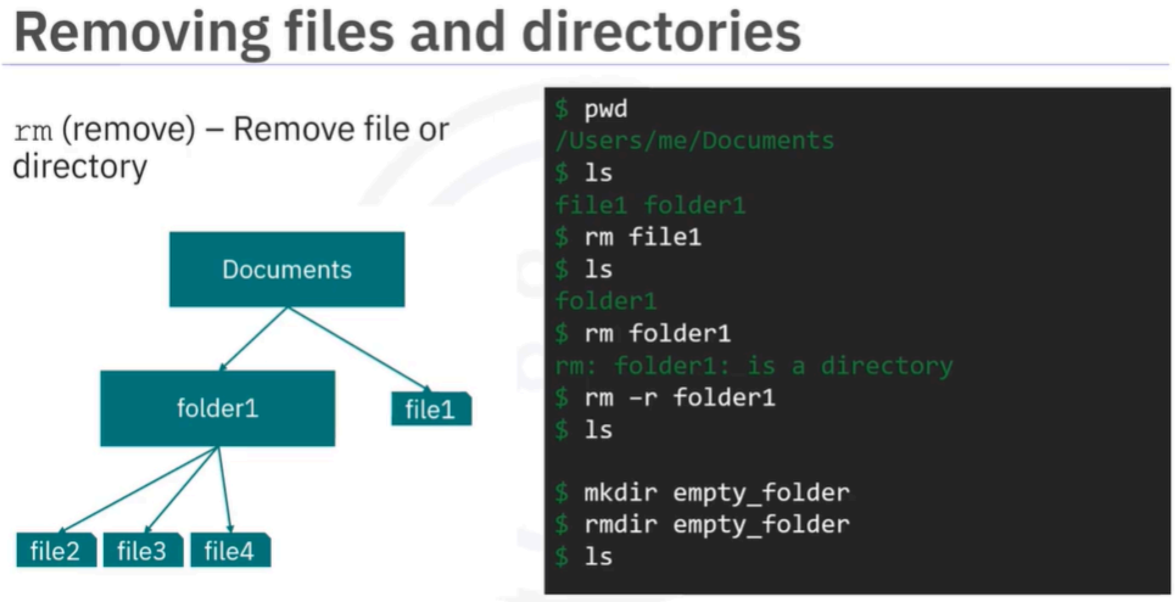
Creating directories

mkdir (make directory) – Make directory

```
$ pwd
/Users/me/Documents
$ ls

$ mkdir test
$ ls
test
```

- The make directory command is used to create directories. Let’s say you are in your documents folder, which is currently empty. To create a folder called “test”, you simply enter “mkdir test.” Now your current directory has a subfolder called “test”.



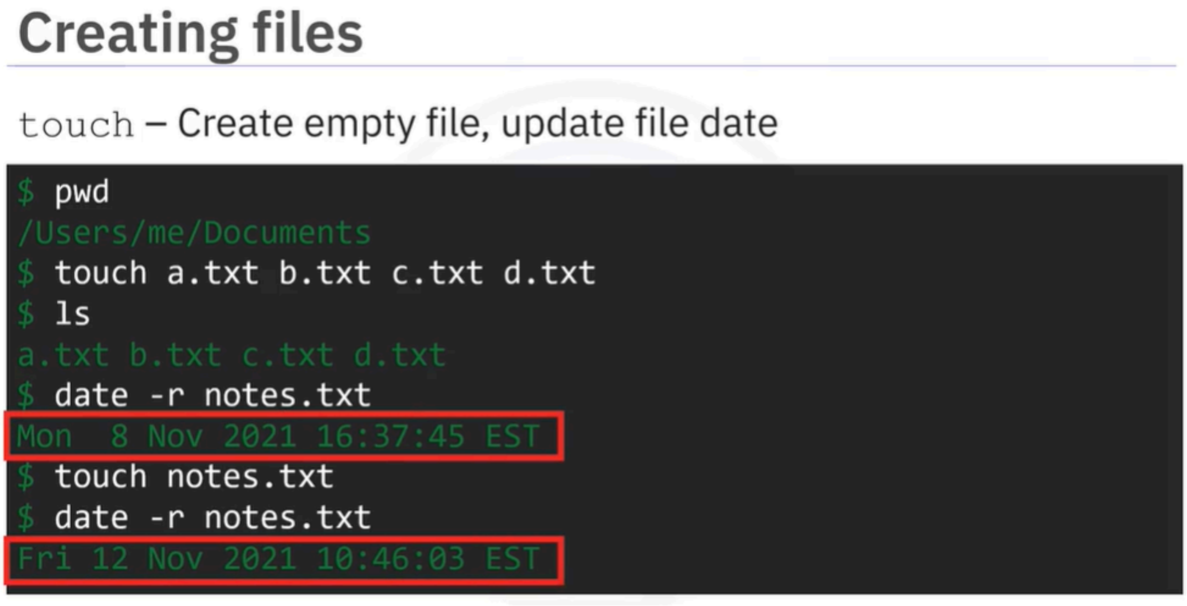
Removing files and directories

rm (remove) – Remove file or directory

```
$ pwd
/Users/me/Documents
$ ls
file1 folder1
$ rm file1
$ ls
folder1
$ rm folder1
rm: folder1: is a directory
$ rm -r folder1
$ ls

$ mkdir empty_folder
$ rmdir empty_folder
$ ls
```

- The rm command allows you to remove a file or directory. Suppose you have a file structure as shown on the left. And, let’s say you are in the documents folder, and you wish to remove “file1”. You can do this by simply typing “rm file1”. Now you can see only “folder1” is left. You cannot simply remove “folder1” since it may contain other files. However, you can easily get around this by entering “rm” with the -r option. The -r option means you want to remove the directory along with all its child file objects. Your Documents folder is now empty. Accordingly, you should always be careful when using rm with the -r option. It is very easy to accidentally remove folders with important data. Now, suppose you create an empty directory with the mkdir command and then you decide to remove it. The rm option rf command is not recommended. Instead you should use the rmdir command which is used solely to remove empty directories. This guarantees that you’ll never accidentally delete any important files or directories. Entering “ls” shows that the current directory is indeed empty.

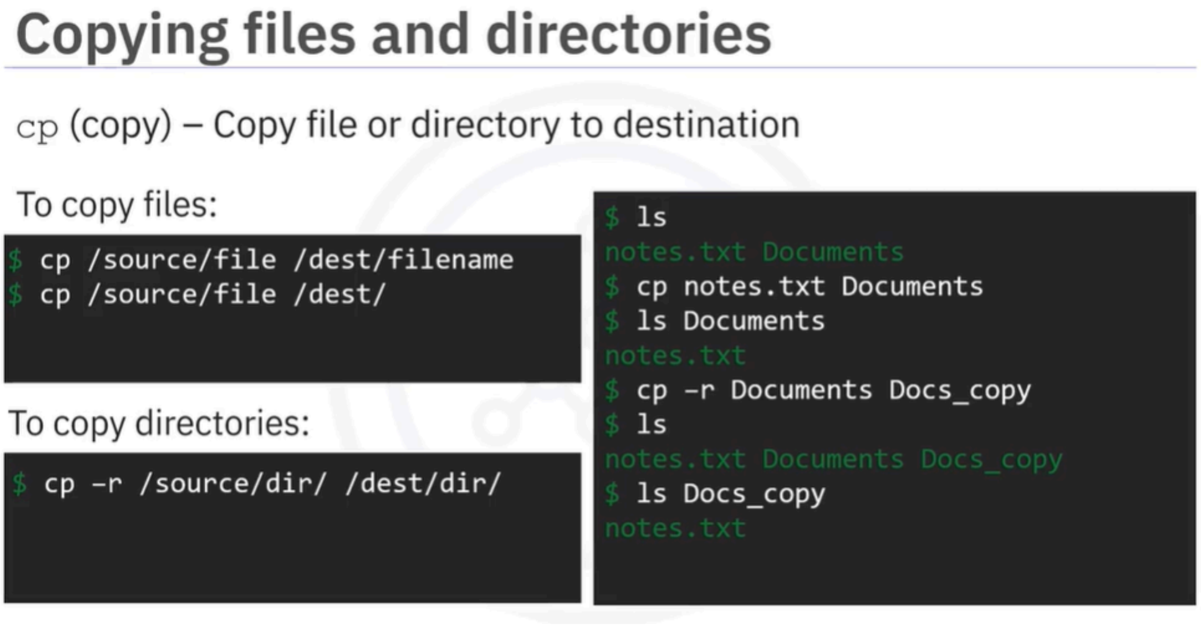


Creating files

`touch` – Create empty file, update file date

```
$ pwd
/Users/me/Documents
$ touch a.txt b.txt c.txt d.txt
$ ls
a.txt b.txt c.txt d.txt
$ date -r notes.txt
Mon  8 Nov 2021 16:37:45 EST
$ touch notes.txt
$ date -r notes.txt
Fri 12 Nov 2021 10:46:03 EST
```

- The touch command can be used to create empty files. Suppose you are in your Documents folder, which is empty, and you wish to create some empty text files. You can do this by entering “touch” along with some filenames: “a”, “b”, “c”, and “d” with the “.txt“ suffix. Now, you can see that your Documents folder contains the four files you created. You might be wondering what the touch command does to an existing file. Suppose your current directory contains a file called “notes.txt". You can see when it was last modified using 'date -r notes.txt’. If you use the touch command on notes.txt, you can see that its last-modified date updated to the current time.



Copying files and directories

cp (copy) – Copy file or directory to destination

To copy files:

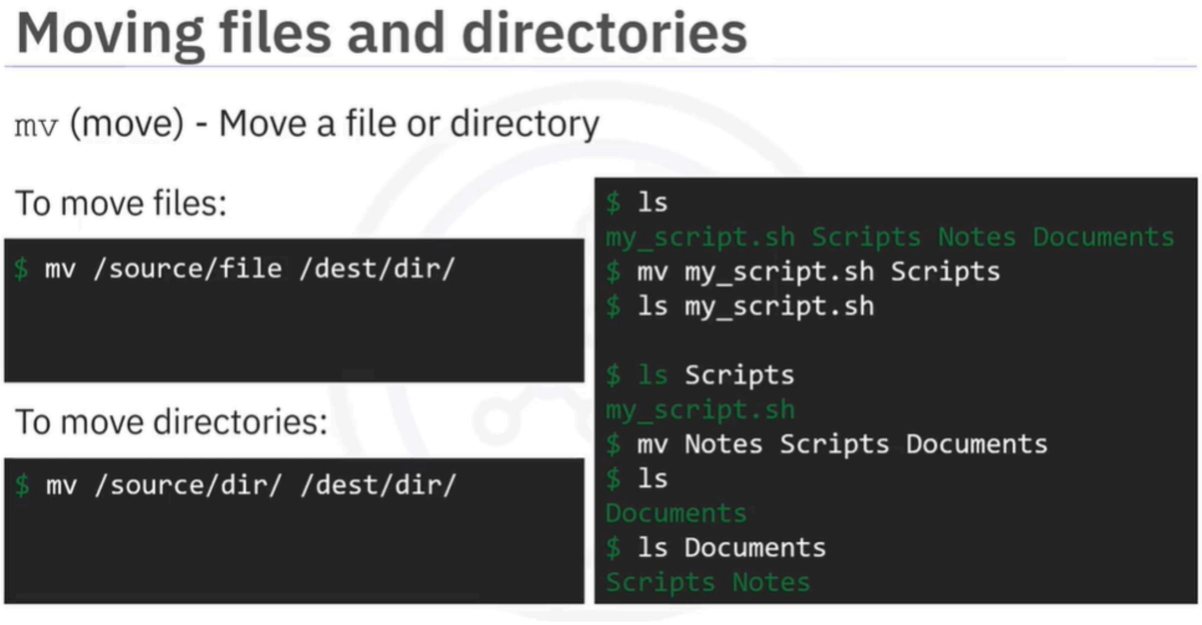
```
$ cp /source/file /dest/filename
$ cp /source/file /dest/
```

To copy directories:

```
$ cp -r /source/dir/ /dest/dir/
```

```
$ ls
notes.txt Documents
$ cp notes.txt Documents
$ ls Documents
notes.txt
$ cp -r Documents Docs_copy
$ ls
notes.txt Documents Docs_copy
$ ls Docs_copy
notes.txt
```

- The cp command allows you to copy a file or directory. To copy files, you can either: copy a file from a source directory and specify the filename for the file in its destination directory, or simply omit the destination filename and default to keeping the same file name. To copy entire directories, you need to give cp the -r option so that it knows to recursively copy all subdirectories and files. Let’s look at some examples: Suppose you have a file called “notes.txt” and a folder called “Documents” in your working directory. You can copy notes.txt to your “Documents” folder using “cp notes.txt Documents”. Now you can see that your Documents folder contains a copy of notes.txt. Notice that you didn’t need to specify a source directory, because cp defaults to your current directory. Next, you can create a copy of your Documents folder, called “Docs_copy”, using the syntax from the left. And as expected, you now have a “Docs_copy ” folder containing the same contents as the original Documents folder.



Moving files and directories

mv (move) - Move a file or directory

To move files:

```
$ mv /source/file /dest/dir/
```

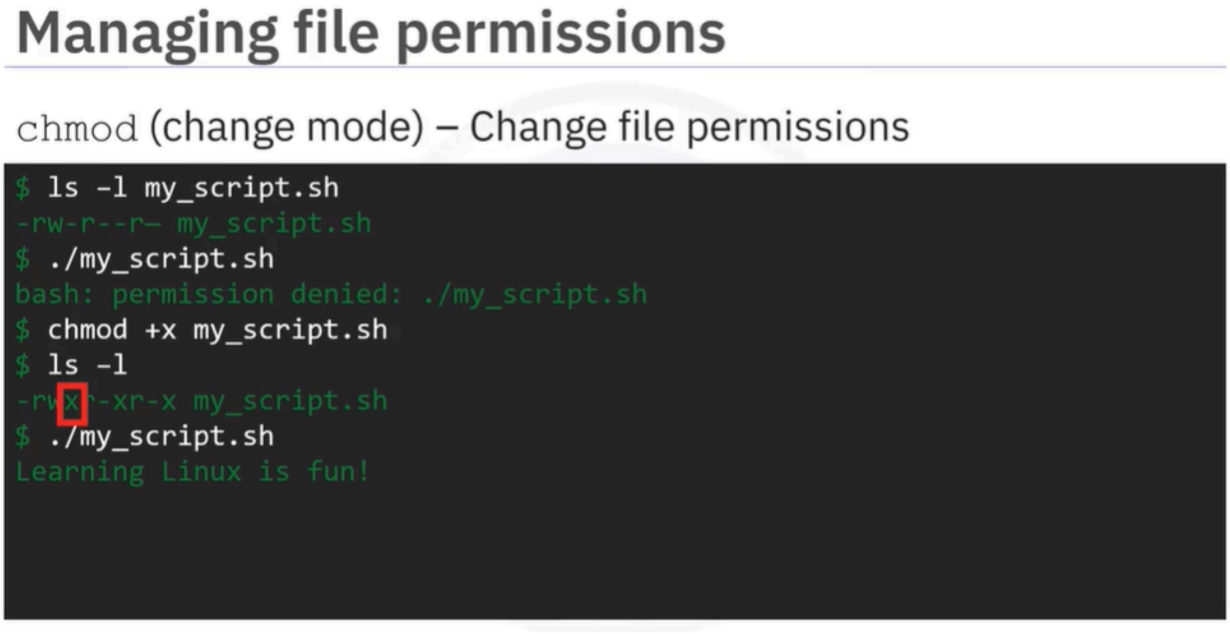
To move directories:

```
$ mv /source/dir/ /dest/dir/
```

```
$ ls
my_script.sh Scripts Notes Documents
$ mv my_script.sh Scripts
$ ls my_script.sh

$ ls Scripts
my_script.sh
$ mv Notes Scripts Documents
$ ls
Documents
$ ls Documents
Scripts Notes
```

- The mv command allows you to move a file or directory. To move files, you can enter “mv” followed by the paths of the files you want to move, followed by the folder you want to move them to. Likewise, to move directories, you type “mv” with the directory’s path to be moved, followed by the path and directory you want to move them to. Let’s look at an example: Suppose you have a file called “my_script.sh” and three folders, called “Scripts”, ”Notes”, and “Documents”. You can move “my_script.sh” to your “Scripts” folder using the syntax from the left. Accordingly, entering “ls my_script.sh” returns nothing, but, entering “ls Scripts” shows that you successfully moved my_script.sh to your Scripts folder. Next, you can move your Notes and Scripts folders to your Documents folder using the syntax on the left. You can see that your directory now just contains your Documents folder and that your Documents folder contains the Scripts and Notes folders that you just moved.

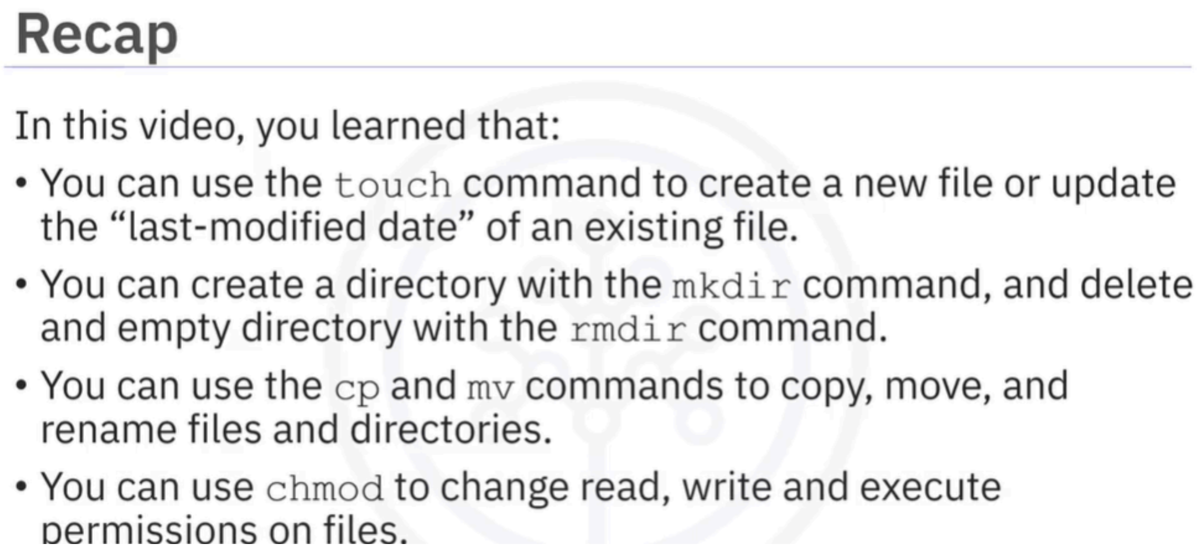


Managing file permissions

chmod (change mode) – Change file permissions

```
$ ls -l my_script.sh
-rw-r--r– my_script.sh
$ ./my_script.sh
bash: permission denied: ./my_script.sh
$ chmod +x my_script.sh
$ ls -l
-rwxr-xr-x my_script.sh
$ ./my_script.sh
Learning Linux is fun!
```

- chmod stands for “change mode” and is used to change read, write, and execute permissions on files. Suppose you have a shell script file in your current directory called “my_script.sh” that echoes “Learning Linux is fun!“ Entering ls -l my_script.sh shows that your shell script has read and write permissions, as indicated by the r and w characters. But if you try to execute the file, you get a permission denied error. In order to make your script executable, you call chmod on my_script.sh with the +x option. Entering ls -l my_script.sh now shows that my_script.sh has executable permissions, as indicated by the x character. Great! Running the script now works.



Recap

In this video, you learned that:

- You can use the `touch` command to create a new file or update the “last-modified date” of an existing file.
- You can create a directory with the `mkdir` command, and delete and empty directory with the `rmdir` command.
- You can use the `cp` and `mv` commands to copy, move, and rename files and directories.
- You can use `chmod` to change read, write and execute permissions on files.