Making a Directory

mkdir

The **mkdir** command creates new directories in your file system.

```
mkdir [options] <Directory>

...

pwd

cd

mkdir BINF_6410

ls
```

When we supply a directory in the above command we are actually supplying a specified relative or absolute path.

More examples:

```
mkdir /home/lukens/Classes

mkdir ./Research

mkdir ../Dir1

mkdir ~/BINF_6410/Practice
```

Useful options

The first one is **-p** which tells **mkdir** to make parent directories as needed.

The second one is **-v** which makes **mkdir** tell us what it is doing

```
mkdir -p BINF_6410/UNIX/Seq_data/DNA

mkdir -pv BINF_6410/UNIX/Seq_data/DNA
```

If any of the parent directories BINF_6410, UNIX, Seq_data, and DNA do not already exist, they will automatically be created.

Removing a Directory

Removing or deleting an empty directory is as easy as creating a directory.

Important Note: there is no undo when it comes to the command line on unix!

The command to remove a directory is **rmdir**, short for remove directory.

```
rmdir [options] <Directory>

(example)

pwd

cd BINF_6410/UNIX/Seq_data/

ls

rmdir DNA

ls
```

rmdir supports the **-v** and **-p** options.

A directory must be empty before it may be removed. You can’t remove a directory that you are in.

Creating a Blank File

touch is an easy way to create empty files.

```
touch [options] <filename>

...

pwd

cd ..

ls

touch readme.txt
```

```
ls
```

Important Concepts to Remember:

Everything is a file under Unix

Even directories.

Unix is an Extensionless System

Files can are free to have any extension they like.

- file.exe - an executable file, or program.
- file.txt - a plain text file.
- file.png, file.gif, file.jpg - an image.

Some versions of Unix are Case Sensitive

Avoid naming files with only a letter difference.

Spaces in names

Spaces in file and directory names should not be used.

```
cd BINF_6410/UNIX/

mkdir Interesting_code
```

Do Not Forget to Use Tab

tab tab tab ....

Copying a File or Directory

```
cp [options] <source> <destination>

...

cp readme.txt ./Seq_data
cp –r Seq_data ../. #-r means “recursive.” Seq_data is a directory.
```

Keep in mind source and target locations. Source is the location of the file (or directory) that we want to move; the target is the location where we want to put it.

Moving a File or Directory

```
mv [options] <source> <destination>

...

mv readme.txt ./Seq_data

mv *.txt Seq_data. #the * wild card matches one or more of anything

mv *ea* Seq_data

mv ea* Seq_data  #here bear.txt won’t move; earwig.txt will.

(the question mark is another common wild card character)

mv –r Seq_data ../
```

Renaming Files and Directories

```
mv readme.txt README

mv Seq_data Seq_data2 #if Seq_data2 does not exist, it is created

my Seq_data Seq_data2 #if Seq_data2 does exist, Seq_data is put into it
```

Note that if a file exists in your directory that has the same name as the one you are moving, it gets replaced without warning.

Removing a File (and non empty Directories)

```
rm [options] <file>

(WARNING) rm -r Directory_name. #removes the directory and everything in
subdirectories too

rm -ir The “i” option asks you to confirm
```

```
“wildcards” * will match everything, so rm * deletes everything in your
directory.
```

Permissions!

Permissions specify what a particular person may or may not do with respect to a file or directory.

So what are they?

Unix permissions dictate 3 things you may do with a file, read, write and execute. They are referred to by a single letter each.

- **r** read - you may view the contents of the file.
- **w** write - you may change the contents of the file.
- **x** execute - you may execute or run the file, e.g. the file is a program or script.

For every file we define 3 sets of people for whom we may specify permissions.

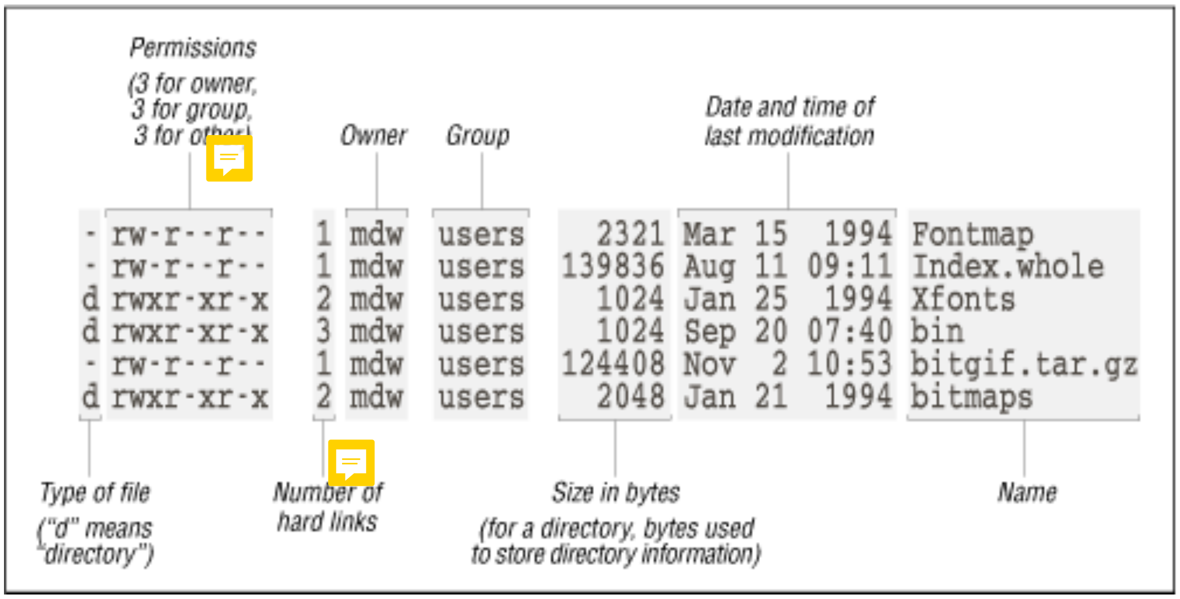
- **owner** - a single person who owns the file. (typically the person who created the file but ownership may be granted to some one else by certain users)
- **group** - every file belongs to a single group.(e.g. lab group, staff, etc..)
- **other** - everyone else who is not in the group or the owner.

If you make a file, it will automatically gain read and write permissions; execute permissions need to be added.

Directories must have executable permissions for one to navigate into them.

Windows won’t respect file permissions of Unix. From a PC to Unix, all permissions might be turne on. Likewise for usb files.

```
ls -l
```



```
- rw-r--r--  1 mdw  users    2321 Mar 15  1994 Fontmap
- rw-r--r--  1 mdw  users  139836 Aug 11 09:11 Index.whole
d rwxr-xr-x  2 mdw  users    1024 Jan 25  1994 Xfonts
d rwxr-xr-x  3 mdw  users    1024 Sep 20 07:40 bin
- rw-r--r--  1 mdw  users  124408 Nov  2 10:53 bitgif.tar.gz
d rwxr-xr-x  2 mdw  users    2048 Jan 21  1994 bitmaps
```

Change Permissions

The command name **chmod** stands for "change mode", and it is used to define the way a file can be accessed.

```
chmod [permissions] [path]

chmod u+x file.txt #add executable permission to user level

chmod g-r file.txt #remove read permission at group level

chmod o+w

chmod u=rwx,g=rx,o=r myfile

chmod 754 myfile
```

Here the digits **7**, **5**, and **4** each individually represent the permissions for the user, group, and others, in that order. Each digit is a combination of the numbers **4, 2, 1,** and **0**:

- 4 stands for "read",
- 2 stands for "write",
- 1 stands for "execute", and

- 0 stands for "no permission."

So **7** is the combination of permissions **4+2+1** (read, write, and execute), **5 is 4+0+1**(read, no write, and execute), and **4 is 4+0+0** (read, no write, and no execute).

The root user

"Root privileges are the powers that the root account has on the system. The root account is the most privileged on the system and has absolute power over it (i.e., complete access to all files and commands)."

http://www.linfo.org/root.html

What should you use

Your home directory is your own personal space on the system. Normally, you should not give either the group or others write access to your home directory, but execute and read can come in handy sometimes.