



# Linux Tutorial

Adopted from Alan Orth

- o **cd** – *change directory*
- o **mkdir** – make directory
- o **mv** – move a file or directory
- o **cp** – copy a file or folder
- o **whoami** – print the name of the current user
- o **who** – print a list of other users who are logged in
- o **date** – print the current date and time
- o **cal** – print a calendar for the current month
- o **echo** – print a text string to the screen



# Command structure

◊ Linux commands come in various forms. Some are simple, and can be used by themselves:

- whoami
  - cal
  - ls
- date

# Command structure

- ...other times you can add “arguments” to change the behavior of the command.

- Arguments are separated by one or more spaces:

`cal 2013`

- Some commands require arguments (they don't make sense to run by themselves, like `mkdir`)



# Commands and their arguments

- Go to your home directory by running: **\$ cd**
- Verify that your working directory is your home directory by running: **\$ pwd**
- Now create a directory named linux: **\$ mkdir linux**
- Change to your results directory: **\$ cd linux**
- make three files by typing: **\$ touch one.txt two.txt three.txt**
- List to see the created files: **\$ ls**

# Commands and their arguments

*ls -lh* ("long" list of files)

*ls -la* ("long" list of hidden files)

*ls -lh one.txt* ("long" list of file)

*mv one.txt four.txt* (rename file1 to file2)

*cp file file* (copy file to filecopy)

*rm four.txt* (delete file)

*rm -I two.txt* (delete file, but ask first)



# Commands and their arguments

*ls -lh* ("long" list of files)

*ls -la* ("long" list of hidden files)

*ls -lh one.txt* ("long" list of file)

*mv one.txt four.txt* (rename file1 to file2)

*cp file file* (copy file to filecopy)

*rm four.txt* (delete file)

*rm -I two.txt* (delete file, but ask first)

# Navigating the file system

o Create some directories and get the hang of moving around them:

```
mkdir first
```

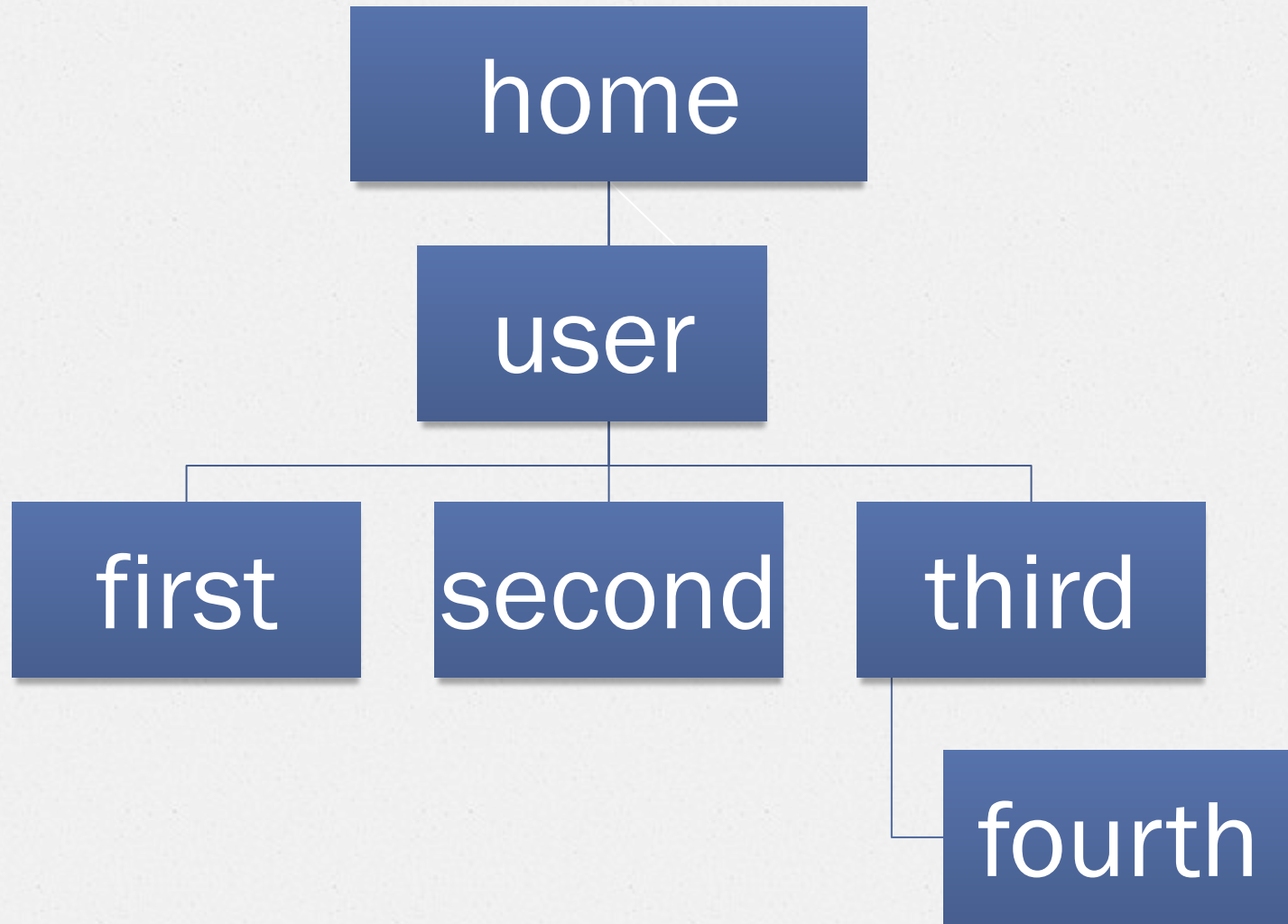
```
mkdir second
```

```
mkdir third/fourth
```

```
cd first
```

How do we get from first to second





## Navigating the file system

o If we want to move to the directory “two” we have to first move back up in the directory hierarchy. Once we move back to “user” we will be able to move into “two.”

```
cd ..
```

```
cd two      or we can
```

```
cd ../two
```

o In Linux “..” means “parent directory



# Your first shell script

A shell script is a text file with a list of commands inside. Shell scripts are good for automating tasks.

We will use the text editor vim for this

Enter the below in a new file, call it script.sh:

```
vim script.sh
```

```
echo "Date and time is:"
```

```
Date
```

```
echo "Your current directory is:"
```

```
pwd
```

To save and exit the text editor type esc :wq

# Your first shell script

🔗 `ls -lh script.sh` (to view the details of the file)

🔗 You can use the below commands to view the contents of a file:

`cat script.sh`

`more script.sh`

`Less script.sh`

🔗 you can thus create files and write into them or edit them.



# Your first shell script

o Run the shell script

```
sh script.sh
```

o It should output something like this

```
“Date and time is:”
```

```
Tue Dec 2 10:00:00 EAT 2014
```

```
“Your current directory is:”
```

```
/home/student1
```

# Input/output redirection

◦ By default, command line programs print to *stdout* (“standard out”). I/O redirection manipulates the input/output of Linux programs, allowing you to capture it or send it somewhere else.

◦ Two main kinds of redirection:

- > to a file

- | to another prog



# Input/output redirection

o stdout lets appreciate this

```
echo "the current date is" date
```

o Results are printed on the screen now let us redirect his output to a file called date.txt

```
echo "the current date is" date > date.txt
```

```
sh script.sh > script.out
```

o Let us pipe the output to another program

```
sh script.sh | less
```