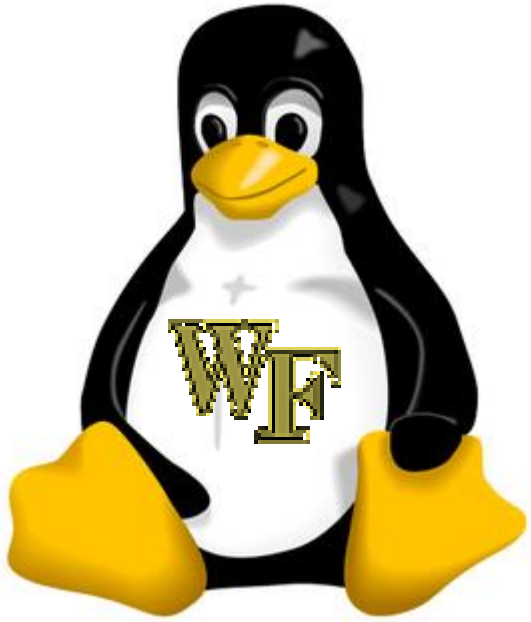
Quick Guide to Linux



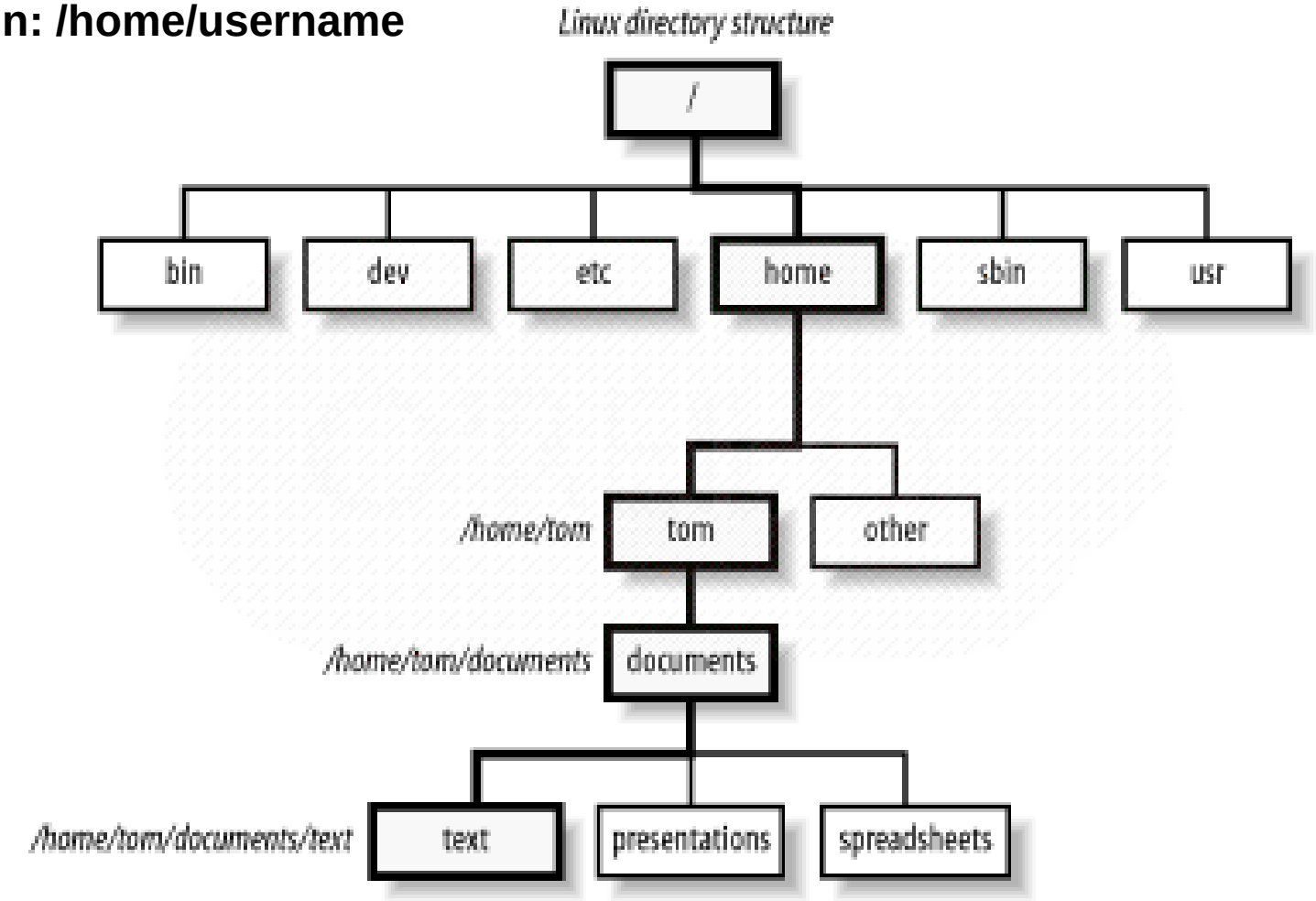
Topics

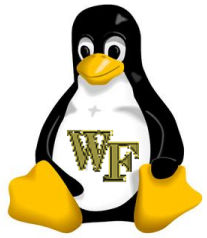
- **Linux Tree Structure**
- **The Command Line**
- **File Manipulation & Wildcards**
- **Text Editors**
- **Permissions**
- **Grep and Regular Expressions**
- **Piping and Redirection**
- **Scripting**

Practice will make you….at least better at it

Linux Tree Structure

- **"/" aka ROOT**
- **When you log in: /home/username**





How to get a Terminal Client

- If you are using Windows:
  - You will need SSH client: http://www.putty.org/
  - FTP: http://winscp.net/eng/download.php
  - FTP: https://filezilla-project.org/

- If you are using Mac:
  - Open up **Utilities** -> click on **Applications** -> Select **Terminal**
  - FTP: https://update.cyberduck.io/Cyberduck-4.5.2.zip
  - FTP: https://filezilla-project.org/

How to use the Terminal Client

- If you are using PUTTY:
  - Host Name (or IP address) field: **rhel6head3.deac.wfu.edu**
  - Use your *username* & *temporary password* to log in

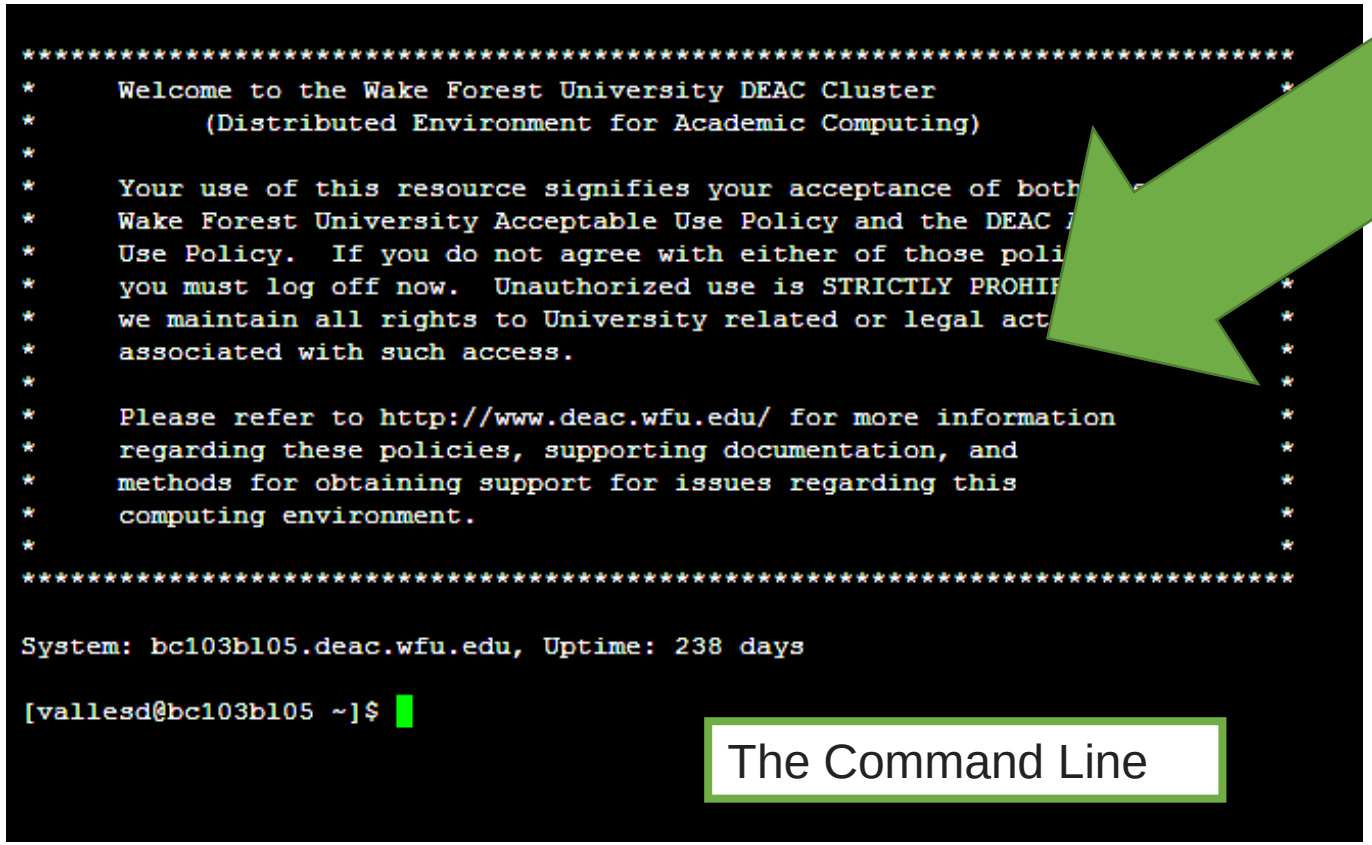
- If you are using Mac's Terminal:
  - Type command: **ssh *username*@rhel6head3.deac.wfu.edu**
  - Use your *temporary password* to log in

Command Line

- **Where you start**
- **Where things end**
- **Where everything operates**

/home/vallesd

(Home directory)



The Command Line

Command Line

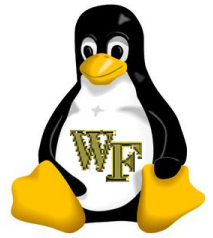
- **Simple Linux commands**
  - **ls** (list) – lists files in current path
  - **pwd** (print working directory) – prints in where you are in the directory tree
  - **cd** (change directory) – move up/down as many levels as desired [Navigation]
  - **cp** (copy) – copy file or directory
  - **mv** (move) – to move file/directory or rename a file/directory
  - **rm** (remove) – delete file/directory
  - **mkdir** (make directory) – creates directory
  - **chmod** (change mode) – change permissions for file/directory
  - **chown** (change of ownership) – change ownership of file/directory
  - **passwd** (password change) – asks for current, then new and then confirm
  - **man** (manual) – read manual for a command

File Manipulation – Copy Command

Command	Results
cp *file1 file2*	Copies the contents of *file1* into *file2*. If *file2* does not exist, it is created; otherwise, *file2* is overwritten with the contents of *file1*.
cp -i *file1 file2*	Like above however, since the "-i" (interactive) option is specified, if *file2* exists, the user is prompted before it is overwritten with the contents of*file1*.
cp *file1 dir1*	Copy the contents of *file1* (into a file named *file1*) inside of directory *dir1*.
cp -R *dir1 dir2*	Copy the contents of the directory *dir1*. If directory *dir2* does not exist, it is created. Otherwise, it creates a directory named *dir1* within directory*dir2*.

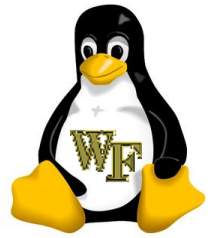
File Manipulation – Move Command

Command	Results
mv *file1 file2*	If *file2* does not exist, then *file1* is renamed *file2*. If *file2* exists, its contents are replaced with the contents of *file1*.
mv -i *file1 file2*	Like above however, since the "-i" (interactive) option is specified, if *file2* exists, the user is prompted before it is overwritten with the contents of *file1*.
mv *file1 file2 file3 dir1*	The files *file1, file2, file3* are moved to directory *dir1*. *dir1* must exist or mv will exit with an error.
mv *dir1 dir2*	If *dir2* does not exist, then *dir1* is renamed *dir2*. If *dir2* exists, the directory *dir1* is created within directory *dir2*.



File Manipulation – Make Directory Command

Command	Results
mkdir *dirname*	The *dirname* directory is created in the current directory path.



File Manipulation – Remove (Delete) Command

Command	Results
rm *file1 file2*	Delete *file1* and *file2*.
rm -i *file1 file2*	Like above however, since the "-i" (interactive) option is specified, the user is prompted before each file is deleted.
rm -r *dir1 dir2*	Directories *dir1* and *dir2* are deleted along with all of their contents.

Be careful with rm!

Linux does not have an undelete command. Once you delete a file with `rm`, it's gone. You can inflict terrific damage on your system with `rm` if you are not careful, particularly with wildcards.

File Manipulation – Wildcards

Wildcard	Meaning
*	Matches any characters
?	Matches any single character
[*characters*]	Matches any character that is a member of the set *characters*. The set of characters may also be expressed as a *POSIX character class* such as one of the following:
[:alnum:]	Alphanumeric characters
[:alpha:]	Alphabetic characters
[:digit:]	Numerals
[:upper:]	Uppercase alphabetic characters
[:lower:]	Lowercase alphabetic characters
[!*characters*]	Matches any character that is not a member of the set *characters*

Text Editors – Plenty!

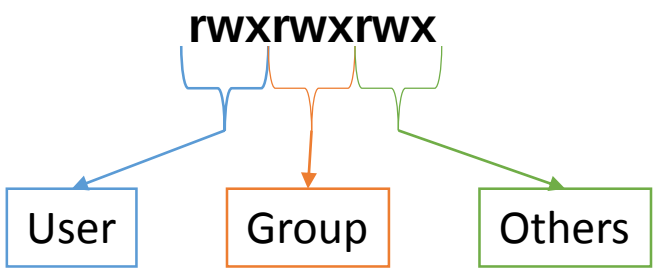
- **Popular ones**
  - **Vi/Vim**
  - **Nano**
  - **Emacs**
  - **gEdit**
- **Vi/Vim is the popular choice in the cluster**
- **Nano is popular to the younger users**
- **Emacs is older and more complex**
- **gEdit is the comfort one – would need X-Window active for the session**

Permissions

- **You can set permission levels for each file and/or directories**
- See permissions on with command: ***ls –al***
- In the first column, you see permission set for each file & directory
- **3 Levels**
  - Users (yourself)
  - Groups
  - Others
- **Permission Types**
  - Read
  - Write
  - Execute

Permissions

- The 3 levels are grouped together:



- Example when **Write** is not set anywhere

r-xr-xr-x

- Indication of a **Directory** as a **READ-ONLY**

dr--r--r--

Permissions

- Setting permissions – CHMOD command
  - “Changing Mode”

- Simple way to change is to use the single letter meaning and +/-
  - User (u)
  - Group (g)
  - Other (o)
  - Read (r)
  - Write (w)
  - Execute (x)

- chmod u=rx file (Give the owner rx permissions, not w)
- chmod go-rwx file (Deny rwx permission for group, others)
- chmod g+w file (Give write permission to the group)
- chmod a+x file1 file2 (Give execute permission to everybody)
- chmod g+rx,o+x file (OK to combine like this with a comma)

Permissions – List all attributes

```
drwxrwxr-x   4  username group     32768    Mar  5 11:21    .matlab
drwxrwxr-x  15 username group     32768    Mar  5 10:56    Matlab
-rw-rw-r--   1 username group         0    Mar  6 06:22    matlab_crash_dump.11415-1
-rw-r--r--   1 username group   3869725  Sep 11  2014      matlab_usage.txt
drwxrwxr-x   4 username group       512 Oct  8  2014      .mgltools
drwxr-xr-x   5 username group     32768  Feb  3 09:56      MoC
drwxr-xr-x   5 username group     32768  Jul 18  2014     .mozilla
drwxr-xr-x.  5 username group     32768  Mar 18 11:19     namd
drwxrwxr-x   3 username group     32768  Jun 20  2014     R
drwxrwxr-x   2 username group     32768  Mar 10 09:22     R_class
```

Permissions

user:group ownership

Size

Date/Time Stamp of last access

Name of File/Directory

GREP & General Expressions

- The GREP command is useful find specific pattern information
- The general use of grep:

Command	Results
grep *pattern file*	The commands outputs to the screen the resultant strings of the inquired *pattern* in the indicated *file*

GREP & General Expressions – Finding String Pattern

test.R file content:

```
library(Rmpi)
mpi.spawn.Rslaves(nslaves=15)
mpi.remote.exec(paste("I am",mpi.comm.rank(),"of",mpi.comm.size()))
mpi.close.Rslaves()
mpi.quit()
```

- Grep for ‘comm’

```
[username@bc103bl05 ~]$ grep comm test.R
mpi.remote.exec(paste("I am",mpi.comm.rank(),"of",mpi.comm.size()))
[username@bc103bl05 ~]$
```

GREP & General Expressions – All kinds

- Grep using the –i flag

[username@bc103bl05 ~]$ grep –i crazy TWEETS.dat

- Resultant output lines that contain:
  - Uppercase: Crazy
  - Lowercase: crazy
  - All Caps: CRAZY
  - Mixture

GREP & General Expressions – Invert Match

- Grep using the –v flag

[username@bc103bl05 ~]$ grep –v crazy TWEETS.dat

- Resultant output lines that contain:
  - Lines that DO NOT contain crazy

GREP & General Expressions – Starts or Ends With

- Grep using the “^string” expression searching lines that starts with the string pattern
- Grep using the “string$” expression searching lines that ends with the string pattern

```
[username@bc103bl05 ~]$ grep "^mpi" test.R
mpi.spawn.Rslaves(nslaves=15)
mpi.remote.exec(paste("I am",mpi.comm.rank(),"of",mpi.comm.size()))
mpi.close.Rslaves()
mpi.quit()
```

- Nothing returned as output

```
[username@bc103bl05 ~]$ grep "mpi$" test.R
[username@bc103bl05 ~]$
```

GREP & General Expressions – Using Wildcards

- Grep using the [:digit:] wildcard expression searching for digit pattern

```
[username@bc103bl05 ~]$ grep [:15:] test.R
mpi.spawn.Rslaves(nslaves=15)
[vallesd@bc103bl05 ~]$
```

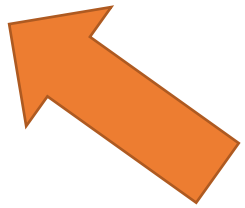
- Grep using the [:upper:] wildcard expression searching for uppercase pattern

```
[username@bc103bl05 ~]$ grep [:R:] test.R
library(Rmpi)
mpi.spawn.Rslaves(nslaves=15)
mpi.close.Rslaves()
```

Pipes & Redirection

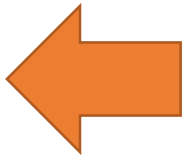
- Pipes help to send data from one program to another

```
[username@bc103bl05 ~]$ ls | head
00.all.20150327094052.xml
4.2.source.tar.gz
5
abinit-7.6.4
ATLAS
atlas3.10.2.tar.bz2
ATLASucs
bin
bowtie2-2.2.4-linux-x86_64.zip
Cav_Dman_2053998.pfile
```



- LS – list of all files in current directory
- | - pipe in which obtains the output of LS and send it to be executed by the second program
- HEAD – List the 10 header files/directories of the current directory

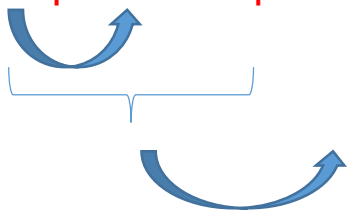
```
[username@bc103bl05 ~]$ ls | head -3
00.all.20150327094052.xml
4.2.source.tar.gz
5
```



- head -3 => list the leading 3 files/directories

Pipes & Redirection

```
[username@bc103bl05 ~]$ ls | head -3 | tail -1
5
```



- TAIL – The tail command displays the trailing part of the command (in this case – LS)

```
[username@bc103bl05 ~]$  ls | tail -3
R_directory
test.R
updates
```

Pipes & Redirection

- Redirection is used mainly to change direction of the program's output

```
[username@bc103bl05 ~]$  ls | tail -3 > file.txt
[username@bc103bl05 ~]$  cat file.txt
R_valles
test.R
Updates
```

- This becomes useful when writing outputs to running programs

Scripting

- A script is a program file that when executed, it performs Linux commands
- Showing examples