Getting Started with Linux Commands

Network Startup Resource Center
www.nsrc.org

The Format of a Command

```
command [options] parameters
```

"Traditionally, UNIX command-line options consist of a dash, followed by one or more lowercase letters. The GNU utilities added a double-dash, followed by a complete word or compound word."

Two very typical examples are:

`-h`

`--help`

and

`-v`

`--version`

Command Parameters

The parameter is what the command acts on.

Often there are multiple parameters.

In Unix UPPERCASE and lowercase are different options and parameters matter.

Spaces ___ are ___ critical ___ .

“-- help” is wrong.

“--help” is right.

Some Command Examples

Let's start simple:

Display a list of files:

```
ls
```

Display a list of files in a long listing format:

```
ls -l
```

Display a list of all files in a long listing format with human-readable file sizes:

```
ls -alh
```

Some Command Examples

Some equivalent ways to do "`ls -alh`":

```
ls -lah
ls -l -a -h
ls -l -all --human-readable
```

Note that there is no double-dash option for "`-l`". You can figure this out by typing:

```
man ls
```

Or by typing:

```
ls --help
```

Where's The Parameter

We typed the "`ls`" command with several options, but no parameter. Do you think "`ls`" uses a parameter?

What is the parameter for "`ls -l`"?

It is "." -- our current directory.

"`ls -l`" and "`ls -l .`" are the same.

We'll discuss files and directories later.

A Disconcerting Linux Feature

If a command executes successfully there is no output returned from the command execution. *this is normal*.

That is, if you type:

```
cp file1 file2
```

The result is that you get your command prompt back. *Nothing means success*.

Let's give this a try...

A Disconcerting Linux Feature

Try doing the following on your machine:

```
$ cd                    [cd = change dir]
$ touch file1           [touch = create/update]
$ cp file1 file2        [cp = copy]
```

- The "$" indicates the command prompt for a normal user.
- A "#" usually means you are the *root* user.

Using Pipes

In Unix it is very easy to use the result of one command as the input for another.

To do this we use the pipe symbol “|”. For example:

```
ls -l /sbin | sort
ls -l /sbin | sort | more
```

What will these commands do?

Stopping Command Output

Stopping commands with continuous output:

Terminate foreground program: CTRL+C

```
$ ping yahoo.com
PING yahoo.com (67.195.160.76): 56 data bytes
64 bytes from 67.195.160.76: icmp_seq=0 ttl=45 time=221.053 ms
64 bytes from 67.195.160.76: icmp_seq=1 ttl=45 time=224.145 ms
^C      ← here press CTRL + C
```

Terminate paging like “less <filename>”

```
$ less /etc/passwd
sysadm:x:1000:1000:System Administrator,,,:/home/sysadm:/bin/bash
postfix:x:104:113::/var/spool/postfix:/bin/false
mysql:x:105:115:MySQL Server,,,:/var/lib/mysql:/bin/false
(END)      ← press the “q” key
```

Proper Command Line Use

The command line in Unix is *much more powerful* than what you may be used to in Windows. ***You can...***

...easily edit long commands

...find and recover past commands

...quickly copy and paste commands.

...auto-complete commands using the tab key (in *bash* shell).

Edit Long Commands

Stop! Don't touch that keyboard!
Arrow keys are slooooooow...

Use *Home* and *End* instead (ctrl-a, shift-a)

Delete with *Backspace* not *Delete*.

Press <ENTER> when *the command is correct*.
You don't need to go to end of the command.

Use "`history | grep string`", then
!*num* instead of lots of up-arrows.

Find and Recover Past Commands

As noted on the previous slide. Use:

```
$ history | grep "command string"
```

Find command number in resulting list.

Execute the command by typing: `$!`*`number`*

To find any command you typed in the past:

```
$ history | grep command
```

Find and Recover Past Commands

For last few commands use the up-arrow.

Don’t re-type a long command if you just typed it.

Instead use the up arrow and adjust the command.

Copy and Paste Commands

In Unix/Linux when you highlight something it is copied.

To copy/paste in Linux/Unix do:

- Highlight text with left mouse cursor. It is now copied (like *ctrl-c* in Windows).
- Move mouse/cursor where you want (any window), and press the *middle* mouse button. This is paste (like *ctrl-v*).

In Windows / Mac use the traditional ctrl-c / ctrl-v

Copy and Paste Commands

Good system administrator

==

Lazy Person

Don’t type a long command

if you can copy / paste it instead!

Auto Complete Commands with Tab

Very, very, very powerful

"The tab key is good", "the tab key is my friend", "press the tab key", "press it again" - This is your mantra.

Tab works in the *bash* shell. Note, the *root* user might not use the *bash* shell by default.

Auto Complete Commands with Tab

Type something and press TAB.

If nothing happens, press TAB twice.

If text was unique text will auto-complete.

- Commands, directory names, file names, and command parameters will all complete

If not unique, press TAB twice. All possibilities will be displayed.

Works with file types based on command!

Auto Complete Commands with Tab

We'll do this now:

```
$ cat /etc          (TAB twice quickly)
$ cat /etc/netw               (TAB)
$ cat /etc/network/in         (TAB)
```

Viewing Files

Several ways to view a file:

1. `cat <filename>`
2. `more <filename`
3. `less <filename>`

- `cat` is short for *conCATenate*
- "`less` is `more`"

Obtaining Help

To get help explaining commands you can do:

- `man <command>`
- `<command> --help`

man stands for “man”ual.

More on “man”

- `man man`

More on Linux directory structure:

- `man hier`

Your Mission

Pay close attention to options and parameters.

Use "`man command`" or "`command --help`" to figure out how each command works.

Use command line magic to save lots and lots and lots and lots of time.

A command acts upon its parameters based on the options you give to the command...