Scripting & Computer Environments

File Operations II

IIIT-H

Aug 7, 2013

...Previously & Today...

Previously: `Linux` basics

- The genesis, whats & whys
- Architecture and file system
- Basic commands

  `pwd, ls, cd, mkdir, rmdir, cp, mv, rm...`

Today: More file operations

- Securing
- Compressing/archiving
- Remote-accessing
- Editing

Brainstorm

- Shell *&* Kernel ??
- How the shell locates commands? Where are they?
- What goes on behind-the-scene when:
  - files and directories are created/removed?
  - such commands as `cp`, `mv` and `rm` are executed?
- When `cd` goes awry

Brainstorm

- Shell *&* Kernel ??
- How the shell locates commands? Where are they?
- What goes on behind-the-scene when:
  - files and directories are created/removed?
  - such commands as `cp`, `mv` and `rm` are executed?
- When `cd` goes awry

Brainstorm

- Shell & Kernel ??
- How the shell locates commands? Where are they?
- What goes on behind-the-scene when:
  - files and directories are created/removed?
  - such commands as `cp`, `mv` and `rm` are executed?
- When `cd` goes awry

- Linux is a multi-user OS. Implications?
- Major security goals - the CIA triad
  - Confidentiality
  - Integrity
  - Authorization
- Different user accounts with different file access privileges.
- File attributes maintained in inode.
  - file type and permissions, links, user and group ownerships, size, timestamp (LMT)

- Three-tiered file protection system

Format

`[type]rwxrwxrwx`

- [type]= - (ordinary) or d (directory) or l (link)
- user's permissions
- Group's permissions
- Others' (world's) permissions
- r=read, w=write, x=execute

`chmod` (change mode)

```
chmod [-R] <mode> <file>
```

`<mode>` has three fields:

- *user category*: `u, g, o or a`
- *operation* : `+, - or =`
- *permissions*: any/combination of `r, w or x`
- Can be done using octal numbers too (read=4, write=2, execute=1)
- The `umask` command reveals default permissions. But it can be set!

- Relative vs Absolute permission assignment
- Directory permissions

chgrp (change group)

```
chgrp <group> <file>
```

- Changes the group ownership of `<file>` to a new group, `<group>`.

chown (change owner)

```
chown <user>[:group] <file>
```

- Assigns to `<user>` the ownership of `<file>` (`[group]` is optional).
- For the root/super user

`chgrp` (change group)

```
chgrp <group> <file>
```

- Changes the group ownership of `<file>` to a new group, `<group>`.

`chown` (change owner)

```
chown <user>[:group] <file>
```

- Assigns to `<user>` the ownership of `<file>` (`[group]` is optional).
- For the root/super user

`tar` (tape archiver)

```
tar [options] <archive name> <files>
```

- A utility to archive multiple files together.
- No compression!
- Common options: `-c` (create), `-x` (extract), `-t` (list), `-f` (filename)

```
tar -cvf  Myarchive.tar file1 file2 file3   (creates)
tar -xvf  Myarchive.tar                     (extracts)
tar -tvf  Myarchive.tar                     (displays contents)
```

gzip/gunzip, bzip/bunzip , bzip2/bunzip2

`gzip [options] <file>`

- Compression/decompression tools.
- `gzip` outputs: a compressed file of extension `.gz` ; original file removed.

```
gzip hello.c hello.html hello.sh
gzip .
gzip -l hello.html.gz                    (amount of compression)
```

Decompression: `gzip -d <file.gz>` or `gunzip <file.gz>`

```
gzip -d hello.c.gz hello.html.gz
gunzip  .
```

1. Compressed Archives using `zip/unzip`

zip/unzip

```
zip <output-file> <files-to-be-compressed>
```

- First argument of `zip` be the compressed file name.
- Doesn't overwrite existing compressed file but updates/appends.

```
zip lectures.zip lecture1.pdf lecture2.pdf
zip -r backup.zip .                          (recursive compression)
unzip lectures.zip
unzip -v lectures.zip                        (view compressed archive)
```

2. Compressed Archives using `tar`.
   - With `-z` option, `tar` compresses using `gzip` (tar -cvzf file.tar.gz)
   - With `-j` option, `tar` compresses using `bzip2` (tar -cvjf file.tar.gz)

1. Compressed Archives using `zip/unzip`

zip/unzip

```
zip <output-file> <files-to-be-compressed>
```

- First argument of `zip` be the compressed file name.
- Doesn't overwrite existing compressed file but updates/appends.

```
zip lectures.zip lecture1.pdf lecture2.pdf
zip -r backup.zip .                          (recursive compression)
unzip lectures.zip
unzip -v lectures.zip                        (view compressed archive)
```

2. Compressed Archives using `tar`.
   - With `-z` option, `tar` compresses using `gzip` (tar -cvzf file.tar.gz)
   - With `-j` option, `tar` compresses using `bzip2` (tar -cvjf file.tar.gz)

ssh (secure shell)

```
ssh [options] [username@]<remote-machine-name/IP address>
```

- `ssh` daemon (`sshd`) must be listening on some port (often port 22).
- Remote machine be configured to accept incoming `SSH` connections.
- Can be used to execute remote commands.
- Common Options: `-X /-Y` (imports `X11` - graphical window), `-f` (puts `ssh` into the background before executing the remote command).

```
ssh somebody@somecompany.com
ssh -X somebody@somecompany.com firefox     (run Firefox remotely)
```

scp (secure copy)

```
scp [-r] <file> username@remote machine:          (a)
scp username@remote machine:<file> <target>       (b)
```

- (a) copies `<file>` *to* the remote machine over an encrypted channel.
- Notice the colon (:) It is necessary.
- (b) copies `<file>` *from* the remote machine to `<target>`.

```
scp -r MyDocuments me@abc.org:                    (export)
scp me@abc.org:myfile .                           (import)
```

sftp (Secure File Transfer Protocol)

```
sftp username@remote machine
```

- Transfers files between local and remote machines securely.
- Uses an interactive console.
- Same connection settings as `ssh`.

- Common commands include:
  - `help`
  - `get` - download from remote machine
  - `put` - upload to remote machine
  - `cd / pwd / ls` - (on remote machine)
  - `lcd / lpwd / lls` - (on local machine)

Other network-related commands you may find useful:

- Ping
- traceroute
- wget
- curl

Consult `man` for more. Again, make `man` your best friend.

- `Vi/Vim` (Vi improved) is a lightweight but powerful text editor.
- Other common text editors: `pico, nano, emacs, gedit ...`
- Uses 3 modes to speed up editing:
  1. Normal mode (shortcut key: esc)
     - `Vi(m)` starts in this mode.
     - To view the text but not edit it.
     - Also to issue a command.
  2. Visual mode (shortcut key: v)
     - To highlight text and perform operations on selected text
  3. Insert mode (shortcut key: i)
     - To type text into the file (buffer)

File Operations: Vi/Vim Commands

Vi Help

```
:help
```

Save (write) file

```
:w <filename>
```

Open another file

```
:e <filename>
```

Editing commands

Copy (yank) $\Longrightarrow$ `y` (also checkout `yy`, `yw`, `{n}yy`)
delete $\Longrightarrow$ `d` (also checkout `dd`, `dw`, `{n}dd`)
Paste $\Longrightarrow$ `p` (where {n} is number of lines)
undo $\Longrightarrow$ `u`
redo $\Longrightarrow$ `ctrl + R`

File Operations: Vi/Vim Commands

Vi Help

```
:help
```

Save (write) file

```
:w <filename>
```

Open another file

```
:e <filename>
```

Editing commands

Copy (yank) $\Longrightarrow$ `y`	(also checkout `yy`, `yw`, `{n}yy`)
delete $\Longrightarrow$ `d`	(also checkout `dd`, `dw`, `{n}dd`)
Paste $\Longrightarrow$ `p`	(where {n} is number of lines)
undo $\Longrightarrow$ `u`	
redo $\Longrightarrow$ `ctrl + R`	

File Operations: Vi/Vim Commands

Vi Help

```
:help
```

Save (write) file

```
:w <filename>
```

Open another file

```
:e <filename>
```

Editing commands

Copy (yank) $\Longrightarrow$ `y` (also checkout `yy`, `yw`, `{n}yy`)
delete $\Longrightarrow$ `d` (also checkout `dd`, `dw`, `{n}dd`)
Paste $\Longrightarrow$ `p` (where {n} is number of lines)
undo $\Longrightarrow$ `u`
redo $\Longrightarrow$ `ctrl + R`

Vi Help

```
:help
```

Save (write) file

```
:w <filename>
```

Open another file

```
:e <filename>
```

Editing commands

Copy (yank) $\Longrightarrow$ `y`	(also checkout `yy`, `yw`, `{n}yy`)
delete $\Longrightarrow$ `d`	(also checkout `dd`, `dw`, `{n}dd`)
Paste $\Longrightarrow$ `p`	(where {n} is number of lines)
undo $\Longrightarrow$ `u`	
redo $\Longrightarrow$ `ctrl + R`	

File Operations: Vi/Vim Commands (2)

Moving between lines

`0 (zero)`	(beginning of line)
`$`	(end of line)
`<n>|`	(move to the nth column)
`<n>G`	(Go to line number `<n>`)

Searching

`/pattern`	(search forward)
`?pattern`	(search backward)
`n`	(Repeat the last pattern search)

Useful Turn-ons

`:set spell`	(spell check)
`:set number`	(line number)
`:syntax on`	(syntax highlighting)

Moving between lines

`0 (zero)`	(beginning of line)
`$`	(end of line)
`<n>|`	(move to the nth column)
`<n>G`	(Go to line number `<n>`)

Searching

`/pattern`	(search forward)
`?pattern`	(search backward)
`n`	(Repeat the last pattern search)

Useful Turn-ons

`:set spell`	(spell check)
`:set number`	(line number)
`:syntax on`	(syntax highlighting)

Moving between lines

`0 (zero)`	(beginning of line)
`$`	(end of line)
`<n>|`	(move to the nth column)
`<n>G`	(Go to line number `<n>`)

Searching

`/pattern`	(search forward)
`?pattern`	(search backward)
`n`	(Repeat the last pattern search)

Useful Turn-ons

`:set spell`	(spell check)
`:set number`	(line number)
`:syntax on`	(syntax highlighting)

File Operations: Vi/Vim Commands (3)

Modifying Envrironment

`:sp`	(horizontal split)
`:vsp`	(vertical split)
`ctrl+w`	(move around)

Quit

`:q`	
`:q!`	(Quit without saving)
`:wq` or `:x`	(Save and quit)

For more on `Vi(m)`, checkout the built-in vimtutor!

Modifying Envrironment

`:sp`	(horizontal split)
`:vsp`	(vertical split)
`ctrl+w`	(move around)

Quit

`:q`	
`:q!`	(Quit without saving)
`:wq` or `:x`	(Save and quit)

For more on `Vi(m)`, checkout the built-in vimtutor!