Scripting & Computer Environments

Intro to Linux (II)

IIIT-H

Aug 12, 2015

...Previously & Today...

Previously:

- Intro to GNU/Linux
  - The genesis, whats, whys
  - Architecture (Kernel, Shell)
  - The file system
- Basic commands

  `pwd, cd, ls, cat, touch`

  `mkdir, rm, rmdir, cp, mv ...`

Today:

- Working with files
  - File security, compression/archival, remote access, editing

Brainstorm

1. FOSS? Linux? GNU/Linux? Kernel? Shell?
2. Inode, Inode number? Behind-the-scene of commands such as `mkdir`, `cat`, `ls -l`, `cp`, `mv` and `rm`? How about `ln`?
3. Security - first thing that comes to your mind? Why bother? Linux is "invulnerable", right?

Brainstorm

1. FOSS? Linux? GNU/Linux? Kernel? Shell?

2. Inode, Inode number? Behind-the-scene of commands such as `mkdir`, `cat`, `ls -l`, `cp`, `mv` and `rm` ? How about `ln`?

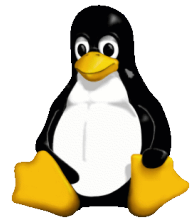
3. Security - first thing that comes to your mind? Why bother? Linux is "invulnerable", right?

Brainstorm

1. FOSS? Linux? GNU/Linux? Kernel? Shell?

2. Inode, Inode number? Behind-the-scene of commands such as `mkdir`, `cat`, `ls -l`, `cp`, `mv` and `rm` ? How about `ln`?

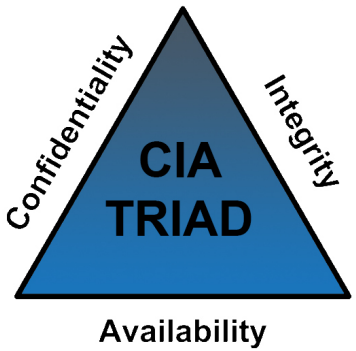
3. Security - first thing that comes to your mind? Why bother? Linux is "invulnerable", right?



Working with Files

1) File Security

- File attributes maintained in inode (aka Index node).
  - Some metadata: file type and permissions, links, user and group ownerships, size, timestamp (LMT), etc. (not file name. Why?)
  - Some commands: `ls -i`, `stat <file>`, `df -i`
- GNU/Linux is a multi-user OS. Implications?
- Major security goals - the CIA triad
  - Confidentiality
  - Integrity
  - Availability



  - Authentication
  - Authorization
  - Accountability

File Permissions

- Different user accounts with different file access privileges/permissions.
- Three-tier file protection system

Format

`[type]rwxrwxrwx`

- [type]= - (ordinary), d (directory), l (link) ...
- user's permissions
- Group's permissions
- Others' (world's) permissions
- r=read, w=write, x=execute
- Notion of `rwx` for ordinary files and directories

Changing Permissions: chmod

Relative vs Absolute permission assignment

chmod (change mode)

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

- `<mode>` has three fields:
  1. *user category*: `u, g, o or a`
  2. *operation* : `+, - or =`
  3. *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!

Example

Suppose you have executed the following:

```
umask 444 ; mkdir MyDir ; cd MyDir ; touch helloWorld
```

- Permissions? What can you do on the file & directory?
- Comment on the following operations:
  1. `ls -l`
  2. `cat helloWorld`
  3. `cp ~/someFile .`
  4. `rm -i helloWorld`
  5. `./helloWorld`

Example

Suppose you have executed the following:

```
umask 444 ; mkdir MyDir ; cd MyDir ; touch helloWorld
```

- Permissions? What can you do on the file & directory?
- Comment on the following operations:

1. `ls -l`
2. `cat helloWorld`
3. `cp ~/somefile .`
4. `rm -rf helloWorld`
5. `./helloWorld`

Example

Suppose you have executed the following:

```
umask 444 ; mkdir MyDir ; cd MyDir ; touch helloWorld
```

- Permissions? What can you do on the file & directory?
- Comment on the following operations:

1. `ls -l`
2. `cat helloWorld`
3. `cp ~/somefile .`
4. `rm -i helloWorld`
5. `./helloWorld`

Example

Suppose you have executed the following:

```
umask 444 ; mkdir MyDir ; cd MyDir ; touch helloWorld
```

- Permissions? What can you do on the file & directory?
- Comment on the following operations:

1. `ls -l`
2. `cat helloWorld`
3. `cp ~/somefile .`
4. `rm -i helloWorld`
5. `./helloWorld`

Example

Suppose you have executed the following:

```
umask 444 ; mkdir MyDir ; cd MyDir ; touch helloWorld
```

- Permissions? What can you do on the file & directory?
- Comment on the following operations:

1. `ls -l`
2. `cat helloWorld`
3. `cp ~/somefile .`
4. `rm -i helloWorld`
5. `./helloWorld`

Example

Suppose you have executed the following:

```
umask 444 ; mkdir MyDir ; cd MyDir ; touch helloWorld
```

- Permissions? What can you do on the file & directory?
- Comment on the following operations:

1. `ls -l`
2. `cat helloWorld`
3. `cp ~/somefile .`
4. `rm -i helloWorld`
5. `./helloWorld`

Changing [Member|Owner]ship

- The `id` command
- The `/etc/passwd`, `/etc/shadow` and `/etc/group` files?

Changing [Member|Owner]ship

- The `id` command
- The `/etc/passwd`, `/etc/shadow` and `/etc/group` files?

`chgrp` (change group)

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

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

Changing [Member|Owner]ship

- The `id` command
- The `/etc/passwd`, `/etc/shadow` and `/etc/group` files?

`chgrp` (change group)

`chgrp <group> <file>`

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

`chown` (change owner)

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

- Assigns to `<user>` the ownership of `<file>` (`[group]` is optional). You can change both owner & group in one go.
- For the root/super user

2) File Compression/Archival

`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)
```

2) File Compression/Archival

`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>
```

```
gzip -d <file.gz> / gunzip <file.gz>
```
(decompression)

- Compression/decompression tools.
- Outputs a compressed file of .gz ext; original file removed.

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

gzip/gunzip, bzip/bunzip, bzip2/bunzip2

`gzip [options] <file>`

`gzip -d <file.gz> / gunzip <file.gz>` (decompression)

- Compression/decompression tools.
- Outputs a compressed file of .gz ext; original file removed.

```
gzip hello.c hello.html hello.sh

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

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

gunzip hello.c.gz hello.html.gz

gzip .                                   (?)
```

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` (the -v flag?)

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.bz2`)

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                          (the -v flag?)
```

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.bz2`)

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                          (the -v flag?)
```

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.bz2)

3) Remote File Access

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 user@example.com

ssh -X user@example.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 user@example.com:              (export)

scp user@example.com: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/diagnostic tools you may find useful:

- `ping`
- `host, dig`
- `traceroute`
- `wget, curl`
- `netstat, ss`

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

4) File Editing

- `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/Command mode* (shortcut key: `esc`)
     - `Vi(m)` starts in this mode.
     - To view the text but not edit it.
     - Also to issue a command.
  2. *Insert/Input mode* (shortcut key: `i`)
     - To type text into the file (buffer)
  3. *Visual mode* (shortcut key: `v`)
     - To highlight text and perform operations on selected text

Vi/Vim Commands

Vi Help

```
:help
```

Vi/Vim Commands

Vi Help

```
:help
```

Save (write) file

```
:w <filename>
```

Vi/Vim Commands

Vi Help

```
:help
```

Save (write) file

```
:w <filename>
```

Open another file

```
:e <filename>
```

Vi/Vim Commands

Vi Help

`:help`

Save (write) file

`:w <filename>`

Open another file

`:e <filename>`

Editing commands

Copy (yank) $\Longrightarrow$ `y`	(try: `yy`, `yw`, {n}`yy`)
delete $\Longrightarrow$ `d`	(try: `dd`, `dw`, {n}`dd`)
Paste $\Longrightarrow$ `p`	{n} is No of lines
undo $\Longrightarrow$ `u`	
redo $\Longrightarrow$ `ctrl + R`	

Moving between lines

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

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)

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)

Modifying Environment

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

Modifying Environment

`: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!

Checkpoint!

- Notion of permission for regular files and directories ?

Assume a system with `umask` value set to 022. Create a file inside a new directory.

1. case 1: -w for the directory only
2. case 2: -w for the file only
3. case 3: -w for both

Now, do `rm`/`mv` on the file. What happens?

Checkpoint!

- Notion of permission for regular files and directories ?

  Assume a system with `umask` value set to 022. Create a file inside a new directory.

  1. case 1: -w for the directory only
  2. case 2: -w for the file only
  3. case 3: -w for both

  Now, do `rm/mv` on the file. What happens?