Managing Files and Software

EE 107S: Introduction to Linux

Lecture 6

File archives: `zip`

- First parameter is archive name, remaining parameters are files/directories
- Create a zip archive from multiple files
- `-r` can be used for directories
- `unzip` extracts zip archive

File archives: `tar`

- Tarball is an uncompressed concatenation of multiple files
- Typically using with `gzip` to compress
  - `gzip` can only compress a single file

Tiers of permissions

1. Permissions for the owner of the file
2. Permissions for the group the file belongs to
   - Groups are composed of users
3. Permissions for the others (not the owner and not in the group the file belongs to)

Types of permissions

- Read: content of files can be read
- Write: file content can be changed
- Execute: file can be executed as a script
  - To `cd` to a directory, that directory must have execute permissions

Reading permissions

- Using `ls -l` command:

```
drwxrwxr-x 2 chirag chirag ... lec1
-rw-rw-r-- 1 chirag chirag ... FileA
```

- Leftmost column is string of permissions; 3 letters for each of user, group, and others
  - First letter represents directory or file
  - Each set of three letters represents read (r), write (w), or execute (x)
- Following two columns are owner and group that the file belongs to

Changing permissions: `chmod`

- Treat permissions field as 3 digit octal number
  - `rwxrwxrwx` = 777 (all users can read, write, or execute)
  - `rw-r-----` = 640 (owner can read/write, group can read, others can't access)
- First parameter is new permissions to assign, parameters after that are files to change
- Use `-R` to recursively change

Alternate syntax

- Specify tier with user (u), group (g), other (o)
- Specify whether to add (+) or subtract (-)
- Specify permission (r, w, x)
- Examples
  - ug+x: add execute permission to user and group
  - o-w: remove write permission for others

Exercise

1. Create a file `temp` in your home directory and see what the default permissions are
2. Remove all group and other permissions
   – Very important if you're doing group projects on a shared machine!

Possible solution

```
touch file
ls -l file
chmod go-rwx file
```

Possible solution

```
touch file
ls -l file
chmod 500 file
```

Superuser privileges: sudo

- sudo and then a command
  - Runs the command as a superuser
- Superuser privileges supersede all file permissions (i.e. can read, write, execute all normal files)

Exercise

- Create a file in `/root` (home directory of the superuser) and see the default permissions are
- Change the permissions so that any user can modify the file

Attempt #1

```
sudo touch /root/file
sudo ls -l /root/file
sudo chmod o+w /root/file
echo "hi" >> /root/file
```

Possible solution

```
sudo touch /root/file
sudo ls -l /root/file
sudo chmod o+w /root/file
sudo chmod o+rw /root
echo "hi" >> /root/file
```

Managing packages: apt

- Think of repositories like an app store
- Used to add/remove and update software
- Must be superuser to change software
  - apt has some features that don't require superuser

Adding software: `apt install`

- Parameters are list of packages to install
- Automatically installs all necessary dependencies

Removing software: `apt remove`

- Parameters are list of packages to remove
- Only removes packages, but residual unused dependencies are marked as “removable”
  - `apt autoremove`

Upgrading software: `apt upgrade`

- One command away from updating all software on a machine!
- Includes kernel upgrades
- Upgrades are done in place, which means no restarting (unless it's a system update)

Updating repository: `apt update`

- Packages are listed in a live repository (i.e. website), which is cached locally
- Sometimes the cache gets out of date, so you need to synchronize it with the repository
  - If out of date, you might get strange errors like "No packages found"

Where is software installed?

- `/usr/bin`: binary executables
- `/usr/lib`: shared libraries
- `/usr/include`: header files
- May be some alternative prefix (e.g. `/usr/local/*`)

Build systems

- Whenever working on or creating a new project, you need some way to generate the executable
- This process is called building, and there are many tools that can help you make building easier

make

- Looks for a file called `Makefile`
- `Makefile` specifies build targets and a “script” for how to compile the target
  - Targets may include “release”, “debug”, “install”
- Can specify flags to `make`
  - `-j` enables parallel builds (e.g. `-j4` uses 4 threads)
  - `-D` adds C++ #define (e.g. `-DENABLE_FEATURE_A=1`)

`configure`

- Projects will often come with a configure script that creates a `Makefile`
- Configure script can set up `Makefile` with certain flags
  - One flag is `--prefix`, which specifies where `make install` should copy the files

Installing tmux from source

```
git clone https://github.com/tmux/tmux.git
mkdir ~/sw
cd tmux
sh autogen.sh
./configure --prefix=/home/chirag/sw
make -j4
make install
~/sw/tmux
```

Resources

- Advanced file permissions
- Writing a Makefile (we'll learn the basics in lab)
- Build systems for C++