SI 504 Class Notes

Jonas Zhonghan Xie

Basic Concept

Shell is a text-based interface to the system.
TCP/IP: protocols that define how computers communicate over Internet.
DNS: Domain Name System. It translates domain names to IP addresses.
Service: a process that runs on a server that handles a task.
Port: a number that identifies a service on a server.
Web Ports: 80 (HTTP) and 443 (HTTPS).
SSH: Secure Shell. A protocol that allows you to connect to a server securely. Port 22. Defined in document RFC 4253.
GNU: free operating system.

File system

- `~`, `/home/`: user home directory
- `/`: root directory
- `/bin/`: binary files
    - `/usr/bin`: user binaries
    - `/sbin`: system binaries
- `/boot/`: boot files, kernel files
- `/dev/`: device files
- `/lib/`: library files
- `/proc/`: info about systems: `cat /proc/cpuinfo` to get CPU info, `cat /proc/meminfo` to get memory info
- `/etc/`: configuration files
- `/opt/`: install packages from vendors
- `/usr/local/*`: files that are local to this system
    - `/usr/local/bin`: local binaries
    - `/usr/local/lib`: local libraries
    - `/usr/local/etc`: local configuration files
- `/var/`: files that are variables
    - `/var/log`: log files
    - `PATH`: environment variable: `echo $PATH` to see the path

User common commands

`ls`: list files in a directory

- `-l`: long format
- `-r`: reverse order
- `-R`: recursive
- `-h`: human readable size

`cat`: print text files. Not all files can be used with `cat`.

`pwd`: print working directory

`less`: view files. Use `q` to exit.

`nano`: text editor

`file`: tell the file type

`cp`: copy files: `cp <source> <destination>`

- `-r`: recursive
- `cp -r <source dir> <destination dir>` to copy a directory recursively

`mv`: move or rename files: `mv <source> <destination>`

`rm`: remove files: `rm <file>`

- `-r`: recursive, to remove a directory
- `-f`: force, to remove without confirmation

`which`: find the path of a command

`touch`: create a file

`mkdir`: create a directory

`rmdir`: remove a directory, only if it is empty

`sort`: sort lines in a file

`uniq`: remove duplicate lines, but only works on sorted files. `sort <file> | uniq`

Glance a file

`tail`: show the last few lines of a file

- `-f`: follow the file
- `-n <N>`: number of lines

`head`: show the first few lines of a file

- `-n <N>`: number of lines

Download files

`wget`: download files from the web

`curl`: download files from the web and output to the screen. Combined with `>` to save into a file.

Search files

`grep`: search for a pattern in a file

- `-i`: case insensitive
- `-v`: invert match
- `-C`: print number of lines before and after the match
- `-c`: print number of matches
- `cat <file> | grep <pattern>`: search for a pattern in a file

`wc`: word count

- `-l`: line count

Compress Files and Uncompress Files

`zip`: compress files

- `zip -R <file> <dir>`: compress a directory recursively

`unzip`: uncompress files

`tar`: compress files

- `tar -zxvf <file>`: uncompress a `.tar.gz` file
- `tar -cvf <file> <dir>`: compress a directory to a `.tar` file
- `tar -zcvf <file> <dir>`: compress a directory to a `.tar.gz` file
- `tar -xvf <file>`: decompress a `.tar` file

Redirect

`>`: redirect the output to a file

`>>`: append the output to a file

`|`: pipe the output to another command

xargs

take command lines from standard input and execute them

```
cat <file> | xargs <command>
```

```
cat * | grep "mlhess" | wc
```

awk

Typical syntax: `awk -F '<split>' '{print $<col>} <file>`. The `<split>` can only takes one character. `<col>` starts from 1.

sed

Replace strings with `sed`.

Typical syntax: `sed s/<old>/<new>/g <file>`. `g` means replace all.

`sed` does not automatically save the changes. To save the changes, use `sed -i s/<old>/<new>/g <file>`.

For case-insensitive, use `sed -i s/<old>/<new>/gI <file>`.

Logs

`/var/log/` stores the system logs.

`/var/log/auth.log` stores the authentication logs.

`zcat` used to read compressed files (`.gz` but not `.tar.gz`)

IP

Two IP addresses for cloud servers. One for local and one for public (changes every time you restart the server). `curl ipconfig.me` ge the public IP address.

System related

`uptime` shows the system uptime.

`du` shows disk usage: `du -h` for human readable size. `du --max-depth=1` to show the size of the first level directories.

`df` shows disk free space.

`ps -ax`: shows all processes on the system

- `PID`: process ID
- `TTY`: terminal the process is running on
- `TIME`: CPU time, how long the process has been running
- `Stat`: process status: `S/D` for sleeping, `R` for running, `Z` for zombie

`w`: shows who is logged in, current load and uptime

System load: the number of processes that are waiting to run. `top` shows the system load. Shows 1 min, 5 min, and 15 min load. Compare the load to the number of CPUs. The number needs to be less than the number of CPUs.

`kill -9 <PID>`: kill a process.

Keys

Keys hold a pattern which matches a lock. Common "key" to server is a password. Two factor authentication is more secure.

Public key - gives this to other people. It will prove you have access to the private key that it matches with

Private key - keep it secret.

Create a key pair: `ssh-keygen -t rsa`. This will create two files `rsa_id` and `rsa_id.pub` in the `~/.ssh` directory. Should backup two files

Keys are preferred to passwords because they are faster and more secure and can be used for automation.

Sudo and Root

Root is the primary admin. Root has no limits on what it can do.

`sudo` : SuperUserDO: allows to become root for a command.

In most places, you have to be given auth to use `sudo`

Packages

OS level packages manager

- `dpkg -l` lists all packages installed on server
- `apt-get update` updates the package list.
- `apt-cache search <package>` searches for a package.
- `apt-get install <package>` installs a package. It will apply to all users.
- `apt-get remove <package>` removes a package. `dpkg -P <package>` removes a package and its configuration files.

Webservers on linux: `apache2` and `nginx`. `apache2` builds the server in `/var/www/html/`.

Automation Script

- Open a text file and start with `#!/bin/bash` to tell the system it is a bash script.
- `chmod 700 <filename>` to make the file executable
- `./<filename>.sh` to run the script
- `echo` prints to the screen
- Define variable with `VAR=value`. No space around `=`
  - `VAR=command``` to store the output of a command in a variable
- Use `$VAR` to refer to the variable
- Loop

```
for i in `ls`; do
echo $i
done
```

- `if` statement

```
if [ $VAR -eq 1 ]; then
echo "VAR is 1"
fi
```

We can also use `#!/usr/bin/python3` to write a python script.