Coccoc Training Course

Linux Fundamentals

Nguyen Trung Dung
(dungw@coccoc)

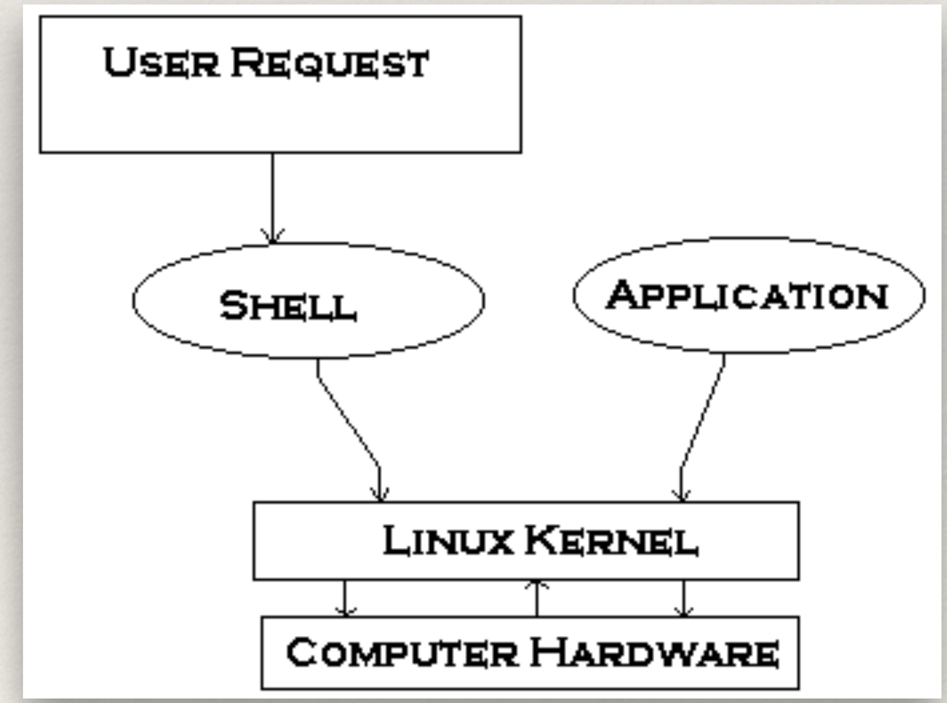
Commands and Scripting

Outline

- Linux Shell and BASH
- Basic commands
- Pipes and Redirections
- Bash Scripting
- Other useful languages

Linux Shells

- Translates instructions in human-readable languages into kernel commands
- bash, ksh, zsh, csh, tcsh…
- Shells are only different in built-in functions and syntax
- `# echo $SHELL`



BASH

- **B**ourne-**a**gain **Sh**ell
- Most popular systems' default shell
- Emphasize programmer efficiency and code clarity rather than computational efficiency -> usually extremely slow on large inputs
- `/bin/sh` and `/bin/bash`

Linux is sexy

```
who | grep -i blonde | date; cd ~; unzip; \
touch; strip; finger; mount; gasp; \
yes; uptime; umount; sleep
```

if you know what i mean

xd4rker

Common commands

- Help: man info whatis apropos
- Text editor: vim emacs nano
- Read: cat tac tail head less more
- Write: echo printf
- Filter: grep cut sort uniq
- Find: find
- Count: wc
- Text manipulation: sed awk
- HTTP: curl wget

Redirections

- Special File descriptors (fd):
  - 0: STDIN
  - 1: STDOUT
  - 2: STDERR
- Special Device: /dev/null

Redirections

- `# echo "100" > /tmp/message 2>&1`
- `# echo $(date) >> /var/log/date.log`
- `# mail -s "salary increase request" boss@coccoc.com < ~/salaryrequest`
- ">": stdout, output to, create file if not exist
- "<": stdin, input from
- ">>": stdout, append to, create file if not exist

Pipes

- Connect one command's STDOUT to another's STDIN
- `# cat ~/report | mail -s "my report" ...`
- `# find /var/log/ -type f -name "*nginx*" | grep -v "gz"`

Xargs

- Use one command's STDIN as another's ARGS
- Can be used to run parallel processes
- `# ls removed* | xargs rm -vf`
- `#`
- `find /var/log -type f -name "*nginx*" | xargs -I{} -n1 -P8 rm -vf {}`

Return Codes

- `# echo $?`
- command ; command
- command || command
- command && command

Bash Scripting

- Shebang: #!/bin/bash
- Variables
  - var="/tmp/log"   # mind the space
    grep 20170723 $var
- Backquotes ($(command) preferred)
  - `# echo "Today is `date`. Hello, I am $(whoami)"`

Function

```
function show_help(){
    local m=1
    echo "$0 PID [interval]"
    echo "First argument: $1"
    echo "Number of arguments: $#"
    echo "m is $m"
}

show_help 1234 10
echo "m is $m"
```

Branching

```
n=10
if [[ "$n" -gt 9 ]]; then     # mind the spaces
    echo "n is $n"
fi

case "$n" in
    "9" | "10" ) echo "n is $n";;
    * ) echo "unexpected";;
esac
```

Comparison Operators

String	Numeric	True If
x = y	x -eq y	x is equal to y
x != y	x -ne y	x is not equal to y
x < y	x -lt y	x is less than y
x <= y	x -le y	x is less than or equal to y
x > y	x -gt y	x is greater than y
x >= y	x -ge y	x is greater than or equal to y
-z x		x is null
-n x		x is not null

- Question: How does String comparison work?

Loops

```
for n in 1 2 3 4; do
    echo "n is $n"
done

for n in $(ls /); do
    echo "n is $n"
done

m=1
while [[ "$m" -le 10 ]]; do
    echo "m is $m"; m=$(expr $m + 1)
    if [[ "$m" -eq 5 ]]; then break fi
done
```

Other languages

- Golang
  - DevOps favs
  - High performant / parallelism
  - Syntax is neat
- Python
  - Soft learning curve
  - Built-in on most systems
  - Can be used on many other purposes: web api, data science…
  - More elegant for complicated scripts

Practices

- Common commands with pipes, xargs
- Basic scripting

Exercises

- Print the name of all users in the system (Hint: /etc/passwd)
- Check to see if python is installed in your server, what's its version
- Find and remove all directories named "remove_me" inside /tmp
- How many processes named "xxx" are running in the system? (Hint: ps aux)

Exercises (a bit more complicated)

- Rename all *.txt inside a folder into *.text
- Remove all files named "xxx" and contains "Shitty stuff" in /tmp
- Given a http log file, print all the lines containing bad status codes (>399)
- Given a http log file, find the most accessed urls, get the total number of accesses on top 5 urls
- Create `./check_process.sh PID [interval]`