Linux Commands and Tools

Practical Class 2

Systems and Storage Lab.

Department of Computer Science and Engineering

Chung-Ang University

Linux Commands and Tools

Linux Shell or Terminal

- A shell is a program that receives commands from the user and gives it to the Linux kernel, and it shows the output
- This shell interact with users in a terminal window
- To open the terminal, press Ctrl + Alt + T in Ubuntu

```
[root@caladan ~]# ssh reader@localhost -p 10022
reader@localhost's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jun 22 14:20:02 UTC 2018

  System load:  0.13              Processes:             89
  Usage of /:   42.0% of 9.78GB   Users logged in:       1
  Memory usage: 12%               IP address for enp0s3: 10.0.2.15
  Swap usage:   0%


0 packages can be updated.
0 updates are security updates.


Last login: Fri Jun 22 14:19:29 2018 from 10.0.2.2
reader@ubuntu:~$
```

Top 50 Linux Commands

Top 50 Linux Commands you must know

1. is	1. clear	1. diff	1. kill and killall	1. apt, pacman, yum, rpm
2. pwd	2. echo	2. cmp	2. df	2. sudo
3. cd	3. less	3. comm	3. mount	3. cal
4. mkdir	4. man	4. sort	4. chmod	4. alias
5. mv	5. unman	5. export	5. chown	5. dd
6. cp	6. whoami	6. zip	6. ifconfig	6. whereis
7. rm	7. tar	7. unzip	7. traceroute	7. whatis
8. touch	8. grep	8. ssh	8. wget	8. top
9. in	9. head	9. service	9. ufw	9. useradd
10. cat	10. tail	10. ps	10. iptables	10. passwd

Top 50 Linux Commands (1)

Commands	Description
ls	The most frequently used command in Linux to list directories
pwd	Print working directory command in Linux
cd	Command to navigate through directories
mkdir	Command used to create directories in Linux
mv	Move or rename files in Linux
cp	Similar usage as mv but for copying files in Linux
rm	Delete files or directories
touch	Create blank/empty files
In	Create symbolic links (shortcuts) to other files
cat	Display file contents on the terminal
clear	Clear the terminal display
echo	Print any text that follows the command
less	Linux command to display paged outputs in the terminal
man	Access manual pages for all Linux commands
uname	Linux command to get basic information about the OS

Top 50 Linux Commands (2)

Commands	Description
whoami	Get the active username
tar	Command to extract and compress files in Linux
grep	Search for a string within an output
head	Return the specified number of lines from the top
tail	Return the specified number of lines from the bottom
diff	Find the difference between two files
cmp	Allows you to check if two files are identical
comm	Combines the functionality of diff and cmp
sort	Linux command to sort the content of a file while outputting
export	Export environment variables in Linux
zip	Zip files in Linux
unzip	Unzip files in Linux
ssh	Provides a secure encrypted connection between two hosts
service	Linux command to start and stop services
ps	Display active processes

Top 50 Linux Commands (3)

Commands	Description
kill and killall	Kill active processes by process ID or name
df	Display disk file system information
mount	Mount file systems in Linux
chmod	Command to change file permissions
chown	Command for granting ownership of files or folders
Ifconfig	Display network interfaces and IP addresses
traceroute	Trace all the network hops to reach the destination
wget	Direct download files from the internet
ufw	Firewall command
iptables	Base firewall for all other firewall utilities to interface with
apt, pacman, yum, rpm	Package managers depending on the distro
sudo	Command to escalate privileges in Linux

Top 50 Linux Commands (4)

Commands	Description
cal	View a command-line calendar
alias	Create custom shortcuts for your regularly used commands
dd	dd reads one block of input file and process it and write in into an outpuf file
whereis	Locate the binary, source, and manual pages for a command
whatis	Find what a command is used for
top	View active processes live with their system usage
useradd and usermod	Add new user or change existing users data
passwd	Create or update passwords for existing users

Basic Linux Commands – Change Directory

- **Change directory (cd) command**
  - The cd is a command used to change the current working directory in various operating systems

```
$ cd path
```

  - Ex) $ cd Desktop

    $ cd /home/user_name/Desktop

Basic Linux Commands - list

List (ls) command

- It is a command to list files in current working directory
- Depending on the parameters, various results can be checked
- Parameters
  - ✓ l : detailed listing view (e.g., owner, date, permission)
  - ✓ a : to include hidden files

```
$ ls [parameters]
```

Ex) $ ls

$ ls –l

$ ls –a

$ ls –al

```
pungki@dev-machine:~$ ls -l
total 508
drwxr-xr-x 2 pungki pungki   4096 Des 10 16:36 Desktop
drwxr-xr-x 2 pungki pungki   4096 Des 10 19:52 Documents
drwxr-xr-x 5 pungki pungki   4096 Jan  2 00:27 Downloads
drwxrwxr-x 6 pungki pungki   4096 Des 28 09:53 lynis-1.3.8
-rw-r----- 1 root   root   363167 Des 28 09:23 lynis.log
-rw-r----- 1 root   root   115339 Des 28 09:23 lynis-report.dat
drwxr-xr-x 2 pungki pungki   4096 Des  1 10:48 Music
drwxr-xr-x 2 pungki pungki   4096 Des  1 10:48 Pictures
lrwxrwxrwx 1 pungki pungki     38 Des  1 11:14 PlayOnLinux's virtual drives -> /
home/pungki/.PlayOnLinux//wineprefix/
drwxr-xr-x 2 pungki pungki   4096 Des  1 10:48 Public
drwxr-xr-x 2 pungki pungki   4096 Des  1 10:48 Templates
drwxr-xr-x 2 pungki pungki   4096 Des  1 10:48 Videos
pungki@dev-machine:~$
```

Basic Linux Commands - move

- Move (mv) is a command that moves one or more files or directories from one place to another
  - Rename the file, you “move” it into a new file with the new name
  - File move and rename action could have been achieved in one step

```
$ mv original_filepath target_filepath
```

Ex) $ mv /Documents/Ukulele/Apache.pdf /test/ (file move)
$ mv Apache.pdf move_Apache.pdf (rename)
$ mv /Documents/Apache.pdf /test/move_Apache.pdf (filemove + rename)

Basic Linux Commands - copy

- Copy (cp) is a command to copy files or directories from one directory to another directory
  - File copy and rename action could have been achieved in one step

```
$ cp original_filepath target_filepath
```

Ex) $ cp /Documents/Apache.pdf /test/Apache2.pdf

Basic Linux Commands – sudo

- sudo is a command that allows users to run programs with the security privileges
  - It stands for “superuser do”
  - The sudo command is required when performing actions that require root permissions

```
$ sudo “Linux command”
```

Ex) $ sudo passwd root (set password for root)

$ sudo su (change the user to root)

$ apt-get install “something”

Basic Linux Commands - pwd

- pwd command (print working directory) write the full pathname of the current working directory to the standard output

```
$ pwd
```

Basic Linux Commands – mkdir

- Create new directories in the file system. You must provide the name of the new directory to mkdir

```
$ mkdir new_folder_name
```

Ex)

$ mkdir ysson

$ mkdir linux_system

Basic Linux Commands – touch

- Create new file in the file system. You must provide the name of the new file name to touch

```
$ touch new_file_name
```

Ex) $ touch example.py

$ touch syslab.c

Basic Linux Commands – shutdown

- The shutdown command lets you shut down or reboot your Linux system.
  - You must have administrator privileges to run it.

```
$ shutdown
```

Ex) $ sudo shutdown

Basic Linux Commands – passwd

- Change the password for a user.
- Just type passwd to change your own password.

```
$ passwd
```

- Another user account
  - You can also change the password of another user account, but you must use sudo

```
$ sudo passwd username
```

Basic Linux Commands – ps

- ps (process status) command is used to provide information about the currently running processes, including their process identification numbers (PIDs)

```
$ ps
```

  - Parameters

    -A : print all processes

    -f : show in full format (UID, PID, etc.)

Ex) $ ps -Af

Basic Linux Commands – grep

- grep is Linux command to search text and string in a given file

```
$ grep “text” [file]
```

Search for a string in the target file

Ex) $grep printf filename

```
root@syslab-server:/home/syslab/ysson/test# grep printf ./structer_pointer.c
        printf("%s, %d, %s\n", __func__, ts1->val, ts1->str);
        printf("%s, %d, %s\n", __func__, ts->val, ts->str);
        printf("%s, %d, %s\n", __func__, ts->val, ts->str);
        printf("%s, sizeof: %d\n", __func__, sizeof(p1));
        printf("%s, sizeof: %d\n", __func__, parr[0]);
        printf("before %s\n", p2);
        printf("after %s\n", p2);
```

Search for a string in all files in the current directory

Ex) $grep printf *

```
root@syslab-server:/home/syslab/ysson/test# grep printf *
double_pointer.c:       printf("%s, %d\n", __func__, arr[0]);
double_pointer.c:       printf("%s, %d\n", __func__, arr[0]);
double_pointer.c:       printf("%s, %d\n", __func__, arr[1]);
double_pointer.c:       printf("%s, %d\n", __func__, arr[2]);
long_max_printf.c:      printf("~0UL: %lu\n", (unsigned long)~0UL);
macro.c:        printf("%s, _AC: %lu\n", __func__, _AC(1, UL)<< PAGE_SHIFT);
malloc_pointer.c:       printf("%s, %d\n", __func__, *a);
malloc_pointer.c:       printf("%s, %d\n", __func__, *a);
malloc_pointer.c:       printf("%s, %d\n", __func__, *a);
printf_test22.c:        printf("Hello Linux\n");
string_pointer.c:       printf("set_string test: %s\n", *str);
string_pointer.c:       printf("set_string str address: %p\n", str);
string_pointer.c:       printf("set_string *str address: %p\n", *str);
string_pointer.c:       printf("set_string address of str: %p\n", &str);
string_pointer.c:       printf("set_string test: %s\n", *str);
string_pointer.c:       printf("set_string str address: %p\n", str);
string_pointer.c:       printf("set_string address of str: %p\n", &str);
```

Basic Linux Commands – kill

- Terminate a process from the command line

```
$ kill PID
```

Ex) $ ps -A | grep process_name //process_name's PID: 1692

$ kill -9 1692

Basic Linux Commands – apt

- The apt is a command which works with Ubuntu's Advanced Packaging Tool (APT)
  - It performs installation of new software packages, upgrade of existing software packages, updating of the package list index, and even upgrading the entire Ubuntu system.

```
$ apt options
```

Ex) $ sudo apt update

$ sudo apt upgrade

$ sudo apt install “something”

$ sudo apt remove “something”

Basic Linux Commands – ssh (secure shell)

- Make a connection to a remote Linux computer and log into user account

```
$ ssh user_account@server_domain or IP_address
```

Ex) $ ssh linux@192.168.10.109

Basic Linux Commands – tar

- tar is a command to create an archive and extract files

```
$ tar –options file_path
```

  - Common options
    - ✓ cvf : when compressing
    - ✓ xvf : when extracting

Ex) $ tar -cvf foo.tar files

$ tar -xvf foo.tar

Basic Linux Commands – top

- top command is used to show the Linux processes. It provides a dynamic real-time view of the running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.

```
$ top
```

  - First line shows you the time and how long your computer has been running
  - Second line shows the number of tasks and their states: running, stopped, sleeping and zombie
  - Third line shows CPU information
  - Forth line shows the total amount of memory, and how much is free and used
  - Fifth line shows the total amount of swap memory, and how much is free and used

Basic Linux Commands – uname

- uname (short for unix name) is a computer program in Unix and Unix-like computer operating systems that prints the name, version and other details about the current machine and the operating system running on it

```
$ uname
```

  - Common options.
    - ✓ a : see everything
    - ✓ r : see the kernel release

Ex) $ uname -r

Basic Linux Commands – history

- History command lists the commands you have previously issued on the command line

```
$ history
```

Basic Linux commands – less

- View files without opening an editor

```
$ less file_path
```

Ex) $ less core.c

$ less example.py

$ less pci.c

Basic Linux Commands – man

- Displays the “manual pages” for a command

```
$ man [linux_command]
```

Ex) $ man ps

$ man top

Useful Tools – ctags

- Ctags is a programming tool that generates an index (or tag) file of names found in source and header files
  - We can find or move to where certain functions and variables are declare or defined
  - Install

    $ sudo apt install ctags
  - Usage

    $ ctags -R

Useful Tools – cscope

- cscope is a programming tool used on very large projects to find source code, functions, declarations, definitions and regular expressions

  - Install

    $ sudo apt install cscope

  - Usage

    $ cscope

Useful Tools – cscope + ctags

❖ Finding vfs_read() function

- vfs_read() function → a function in the virtual file system

1) Run cscope

```
root@test1:/home/syslab/ysson/linux-5.2.11# cscope
```

2) Enter vfs_read

```
Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string: vfs_read
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

3) Choose vfs_read which you want

Useful Tools – cscope + ctags

❖ Finding vfs_read() function

- vfs_read() function → a function in the virtual file system

3) Choose vfs_read which you want

```
  File            Line
0 cache.c          225 static void v9fs_vfs_readpage_complete(struct page *page, void *data,
1 cache.c          252 v9fs_vfs_readpage_complete,
2 cache.c          290 v9fs_vfs_readpage_complete,
3 vfs_addr.c        76 * v9fs_vfs_readpage - read an entire page in from 9P
4 vfs_addr.c        83 static int v9fs_vfs_readpage(struct file *filp, struct page *page)
5 vfs_addr.c        89 * v9fs_vfs_readpages - read a set of pages from 9P
6 vfs_addr.c        98 static int v9fs_vfs_readpages(struct file *filp, struct address_space *mapping,
7 vfs_addr.c       326 .readpage = v9fs_vfs_readpage,
8 vfs_addr.c       327 .readpages = v9fs_vfs_readpages,
9 dax.c            1160 * validated via access_ok() in either vfs_read() or
a exec.c           1003 ssize_t res = vfs_read(file, (void __user *)addr, len, &pos);
b namei.c          4704 * vfs_readlink - copy symlink body into userspace buffer
c namei.c          4713 int vfs_readlink(struct dentry *dentry, char __user *buffer, int buflen)
d namei.c          4742 EXPORT_SYMBOL(vfs_readlink);
e nfs4xdr.c        3648 * XXX: By default, vfs_readlink() will truncate symlinks if they
f nfs4xdr.c        3650 * easy fix is: if vfs_readlink() precisely fills the buffer, assume
g read_write.c      421 ssize_t __vfs_read(struct file *file, char __user *buf, size_t count,
h read_write.c      440 result = vfs_read(file, (void __user *)buf, count, pos);
i read_write.c      446 ssize_t vfs_read(struct file *file, char __user *buf, size_t count, loff_t *pos)
j read_write.c      461 ret = __vfs_read(file, buf, count, pos);
k read_write.c      587 ret = vfs_read(f.file, buf, count, ppos);
l read_write.c      639 ret = vfs_read(f.file, buf, count, &pos);
m read_write.c      987 ssize_t vfs_readv(struct file *file, const struct iovec __user *vec,
n read_write.c     1034 ret = vfs_readv(f.file, vec, vlen, ppos, flags);
o read_write.c     1089 ret = vfs_readv(f.file, vec, vlen, &pos, flags);
p splice.c          359 res = vfs_readv(file, (const struct iovec __user *)vec, vlen, &pos, 0);
q stat.c            411 error = vfs_readlink(path.dentry, buf, bufsiz);
r xfs_ioctl.c       294 error = vfs_readlink(dentry, hreq->ohandle, olen);
s fs.h             1885 extern ssize_t __vfs_read(struct file *, char __user *, size_t, loff_t *);
t fs.h             1886 extern ssize_t vfs_read(struct file *, char __user *, size_t, loff_t *);
u fs.h             1888 extern ssize_t vfs_readv(struct file *, const struct iovec __user *,
v fs.h             3222 extern int vfs_readlink(struct dentry *, char __user *, int);
w sysctl_binary.c   923 result = vfs_read(file, oldval, oldlen, &pos);
x iint.c            200 ret = __vfs_read(file, buf, count, &offset);
```

Useful Tools – cscope + ctags

❖ Finding vfs_read() function

- vfs_read() function → a function in the virtual file system

4) Enter “ Ctrl +] ” on vfs_read function

```
ssize_t ksys_pread64(unsigned int fd, char __user *buf, size_t count,
                     loff_t pos)
{
        struct fd f;
        ssize_t ret = -EBADF;

        if (pos < 0)
                return -EINVAL;

        f = fdget(fd);
        if (f.file) {
                ret = -ESPIPE;
                if (f.file->f_mode & FMODE_PREAD)
                        ret = vfs_read(f.file, buf, count, &pos);
                fdput(f);
        }

        return ret;
}
```

Useful tools – cscope + ctags

❖ Finding vfs_read() function

- vfs_read() function → a function in the virtual file system

5) You can find body of vfs_read() function

```
ssize_t vfs_read(struct file *file, char __user *buf, size_t count, loff_t *pos)
{
        ssize_t ret;

        if (!(file->f_mode & FMODE_READ))
                return -EBADF;
        if (!(file->f_mode & FMODE_CAN_READ))
                return -EINVAL;
        if (unlikely(!access_ok(buf, count)))
                return -EFAULT;

        ret = rw_verify_area(READ, file, pos, count);
        if (!ret) {
                if (count > MAX_RW_COUNT)
                        count =  MAX_RW_COUNT;
                ret = __vfs_read(file, buf, count, pos);
                if (ret > 0) {
                        fsnotify_access(file);
                        add_rchar(current, ret);
                }
                inc_syscr(current);
        }

        return ret;
}
```

Useful tools – cscope + ctags

❖ Finding vfs_read() function

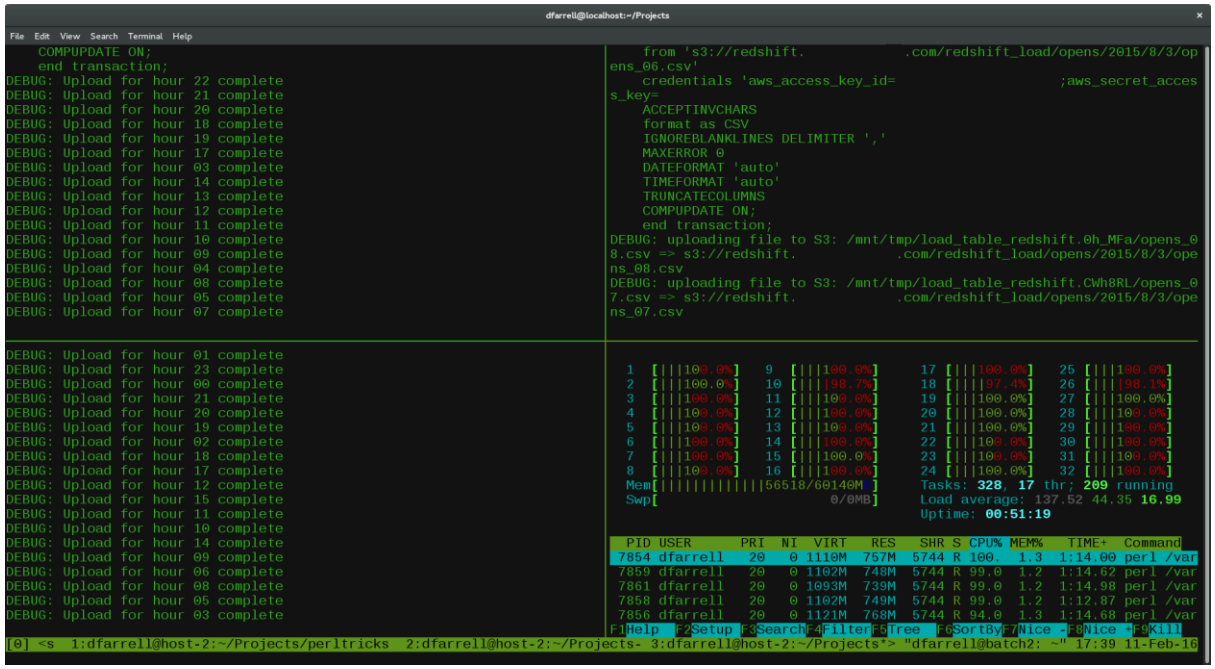
- vfs_read() function → a function in the virtual file system

5) Enter “ Ctrl + t ” → you can back the previous function

```
ssize_t ksys_pread64(unsigned int fd, char __user *buf, size_t count,
                     loff_t pos)
{
        struct fd f;
        ssize_t ret = -EBADF;

        if (pos < 0)
                return -EINVAL;

        f = fdget(fd);
        if (f.file) {
                ret = -ESPIPE;
                if (f.file->f_mode & FMODE_PREAD)
                        ret = vfs_read(f.file, buf, count, &pos);
                fdput(f);
        }

        return ret;
}
```

Tmux

- **Tmux is a tool to manage virtual concoles**
  - It allows multiple terminal sessions to be accessed simultaneously in a single window
  - It is useful for running more than one command-line program at the same time



Tmux

- **Essential tmux commands**
  - tmux: start a new tmux session
  - ctrl-b + %: split a pane vertically
  - ctrl-b + “ : split a pane horizontally
  - ctrl-b + o : move to the next pane
  - ctrl-b + <arrow key> : switch to the pane in whichever direction you press
  - ctrl-b + [: move with cursor
  - ctrl-b + z : zoom (or unzoom) a pane
  - ctrl-b + c: create a new window
  - ctrl-b + N: go to window N (0~9)
  - ctrl-b + d: detach from a session
  - tmux a: attach to an existing session