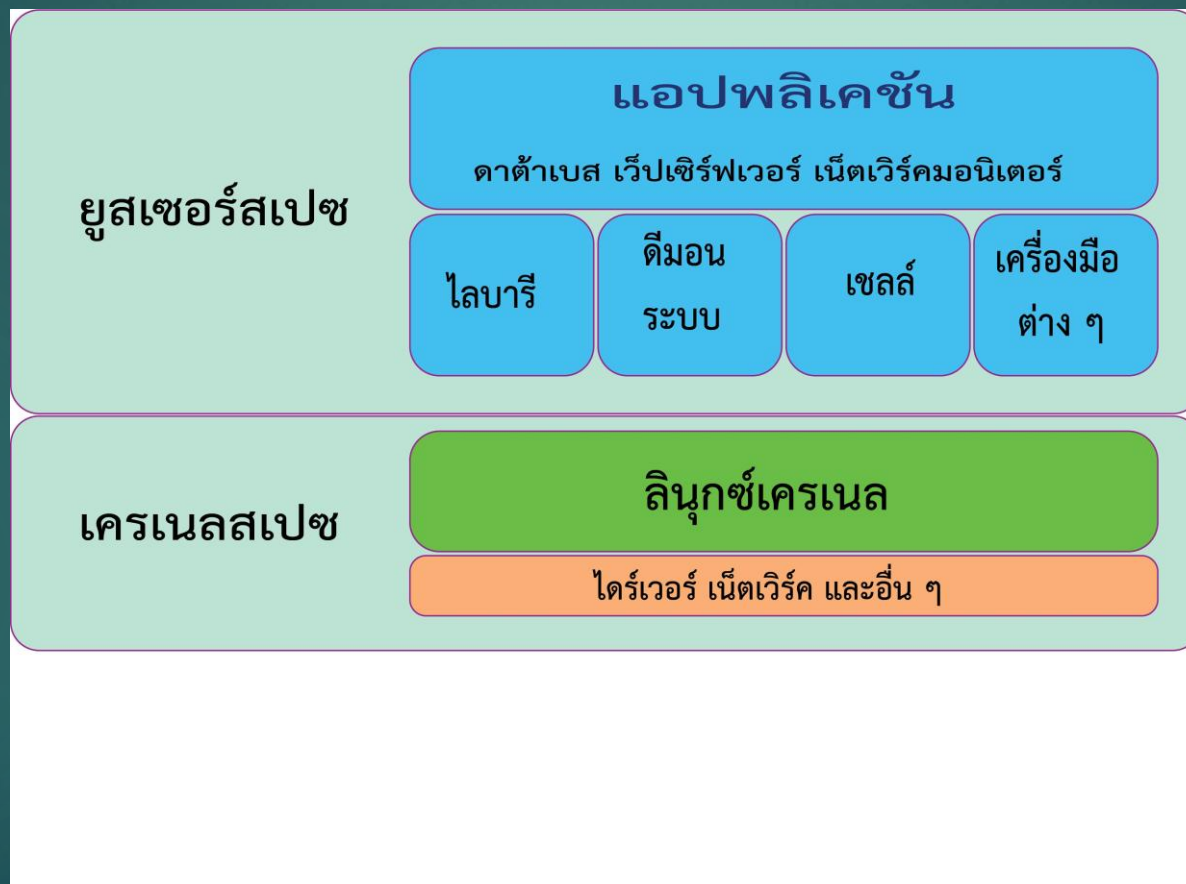




Basic Linux Command and networking

Linux System



Linux 101

- ▶ Install Vmware (VirtualBox, WSLv2 or..)
- ▶ Install Ubuntu (Version 22.04)
 - ▶ Run (update and upgrade)

```
modern@ubuntu:~$ sudo apt-get -y update
[sudo] password for modern:
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
      [88.7 kB]
Hit:2 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic
      InRelease
...
Get:30 http://us.archive.ubuntu.com/ubuntu bionic-backports/
      universe
amd64 DEP-11 Metadata [7,984 B]
Fetched 8,159 kB in 28s (287 kB/s)
Reading package lists... Done
```

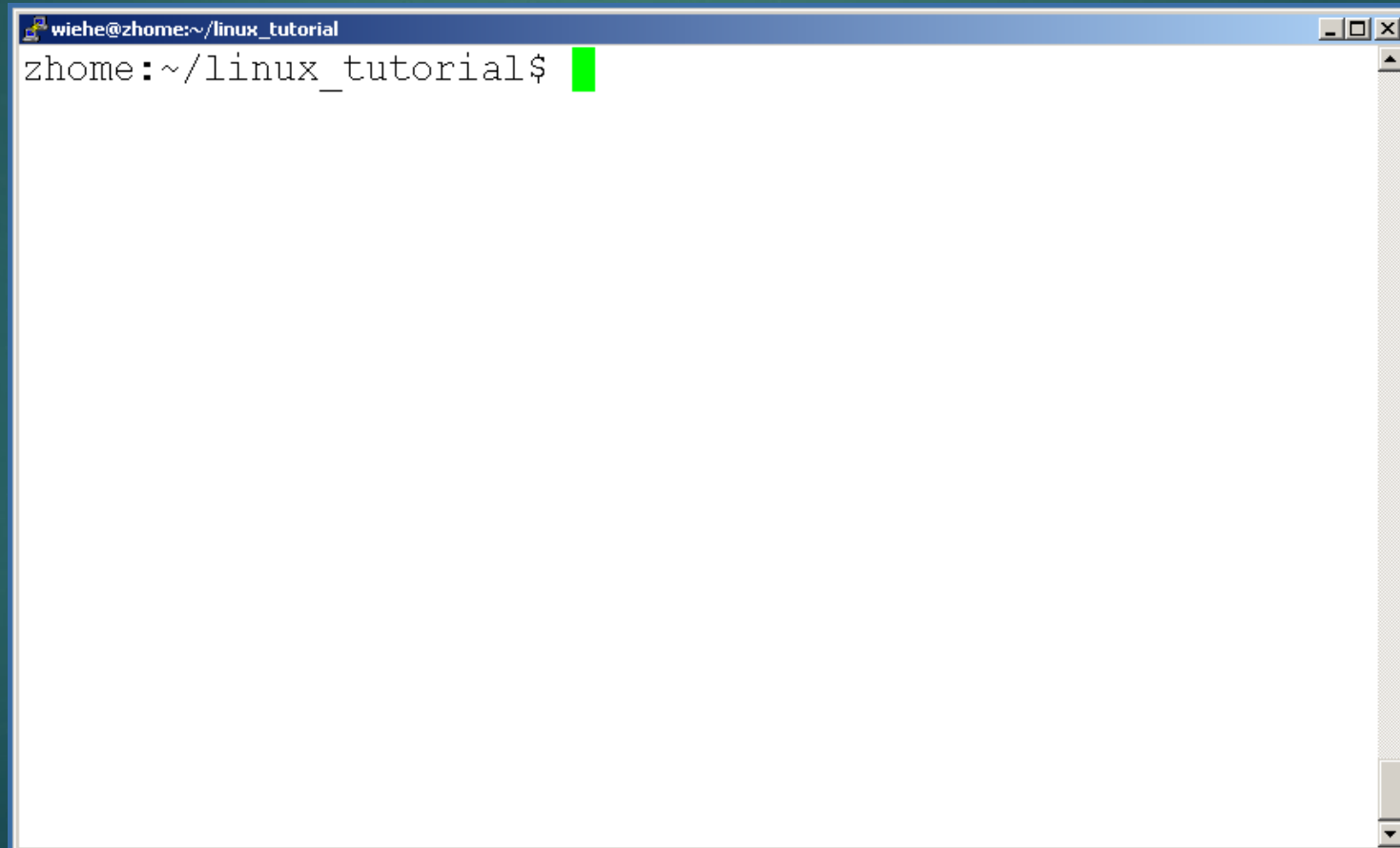
```
modern@ubuntu:~$ sudo apt-get -y upgrade
[
```

Substitute User

```
modern@ubuntu:~$ sudo su  
[sudo] password for modern:  
root@ubuntu:/home/modern# exit  
exit  
modern@ubuntu:~$
```

Connecting to a Unix/Linux system

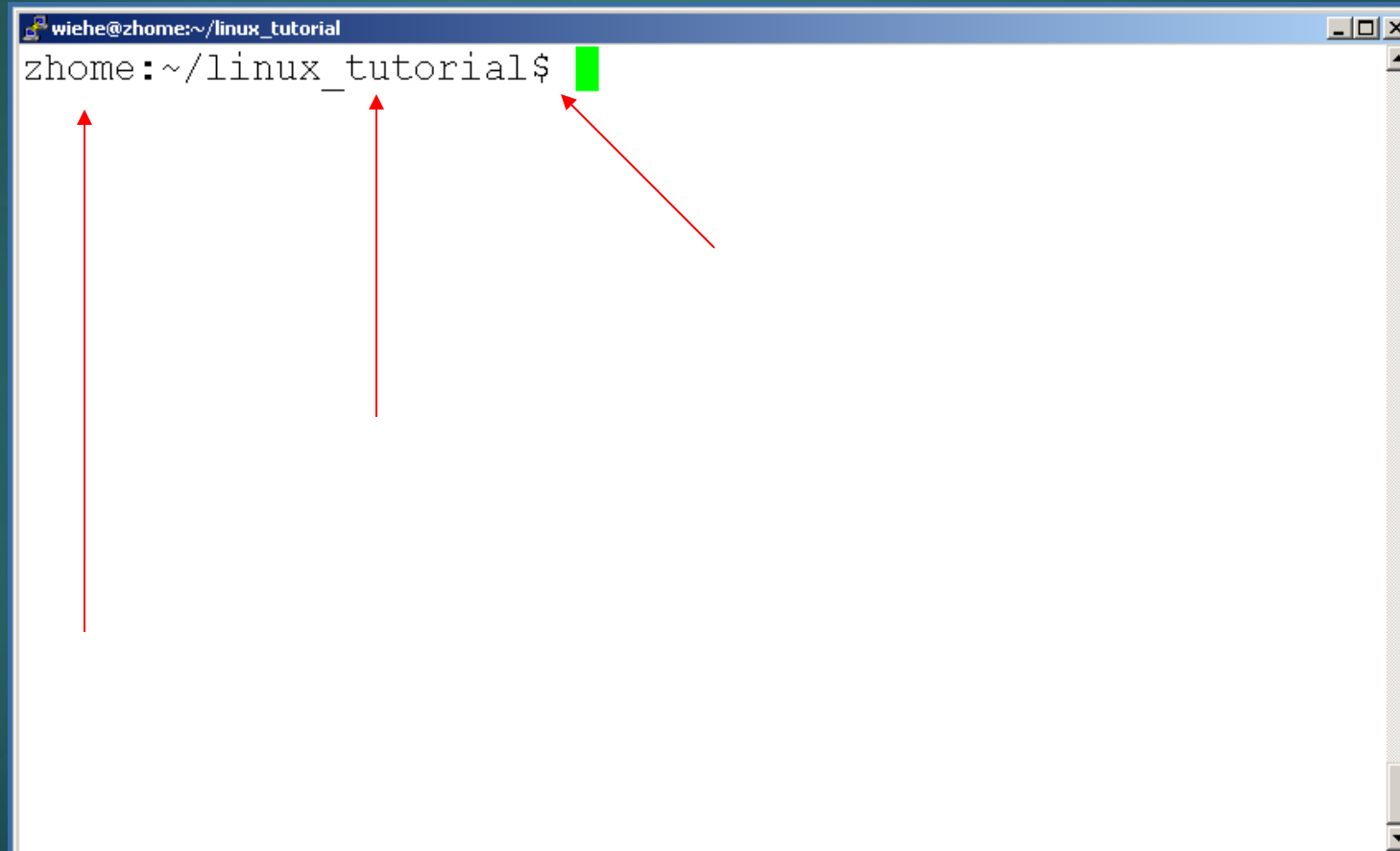
- Open up a terminal:

A screenshot of a terminal window. The title bar at the top reads 'wiehe@zhome:~/linux_tutorial'. The main area of the terminal shows the prompt 'zhome:~/linux_tutorial\$' followed by a green cursor. The window has standard Linux window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$
```

Connecting to a Unix/Linux system

- Open up a terminal:



A terminal window titled "wiehe@zhome:~/linux_tutorial" is shown. The prompt is "zhome:~/linux_tutorial\$". A green cursor is positioned at the end of the prompt. Three red arrows point to the prompt: one to the username "zhome", one to the directory path "~/linux_tutorial", and one to the dollar sign "\$".

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$
```

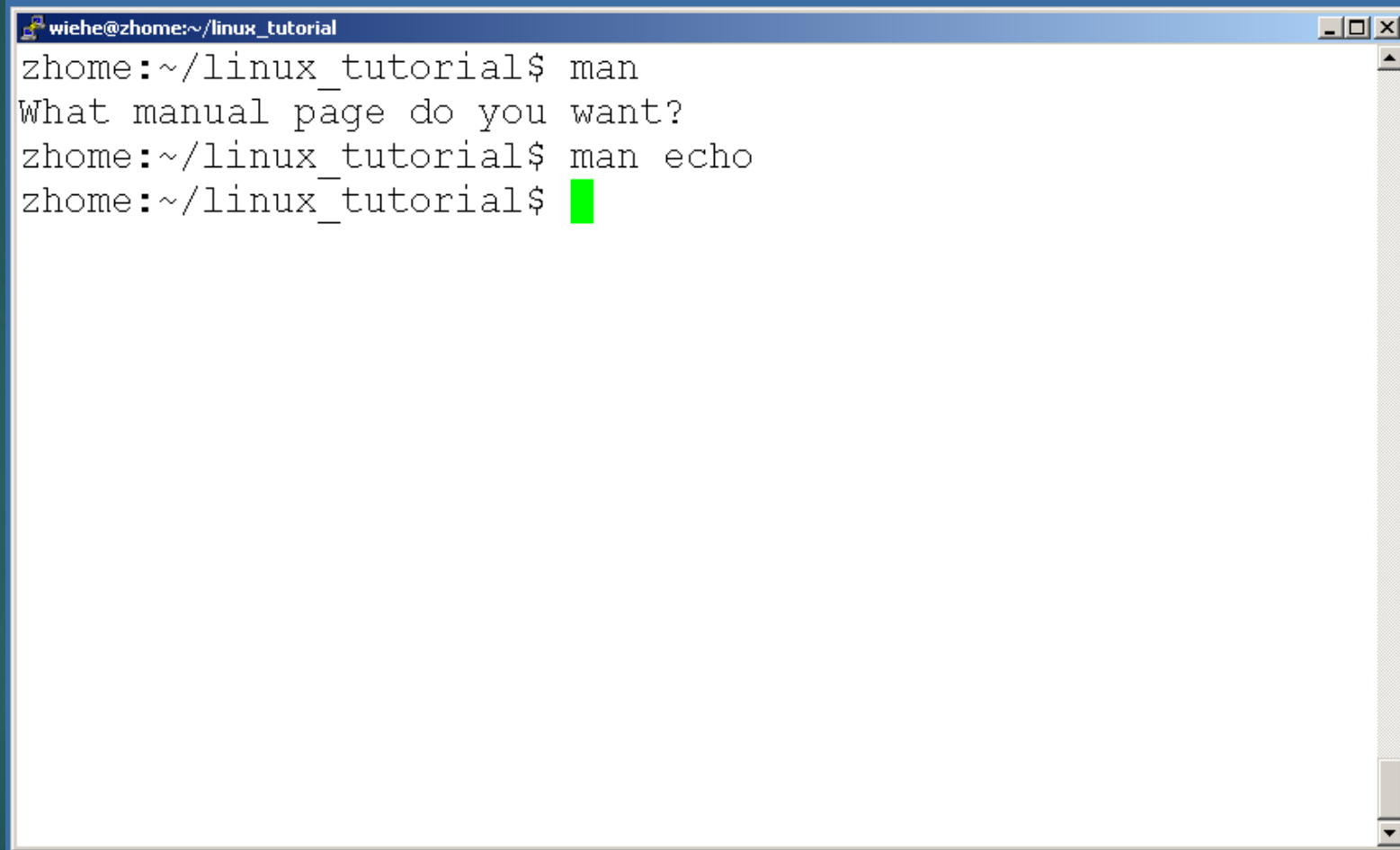
What exactly is a “shell”?

- ▶ After logging in, Linux/Unix starts another program called the **shell**
- ▶ The shell interprets commands the user types and manages their execution
 - ▶ The shell communicates with the internal part of the operating system called the **kernel**
 - ▶ The most popular shells are: tcsh, csh, korn, and bash
- ▶ Shell commands are **CASE SENSITIVE!**

Help!

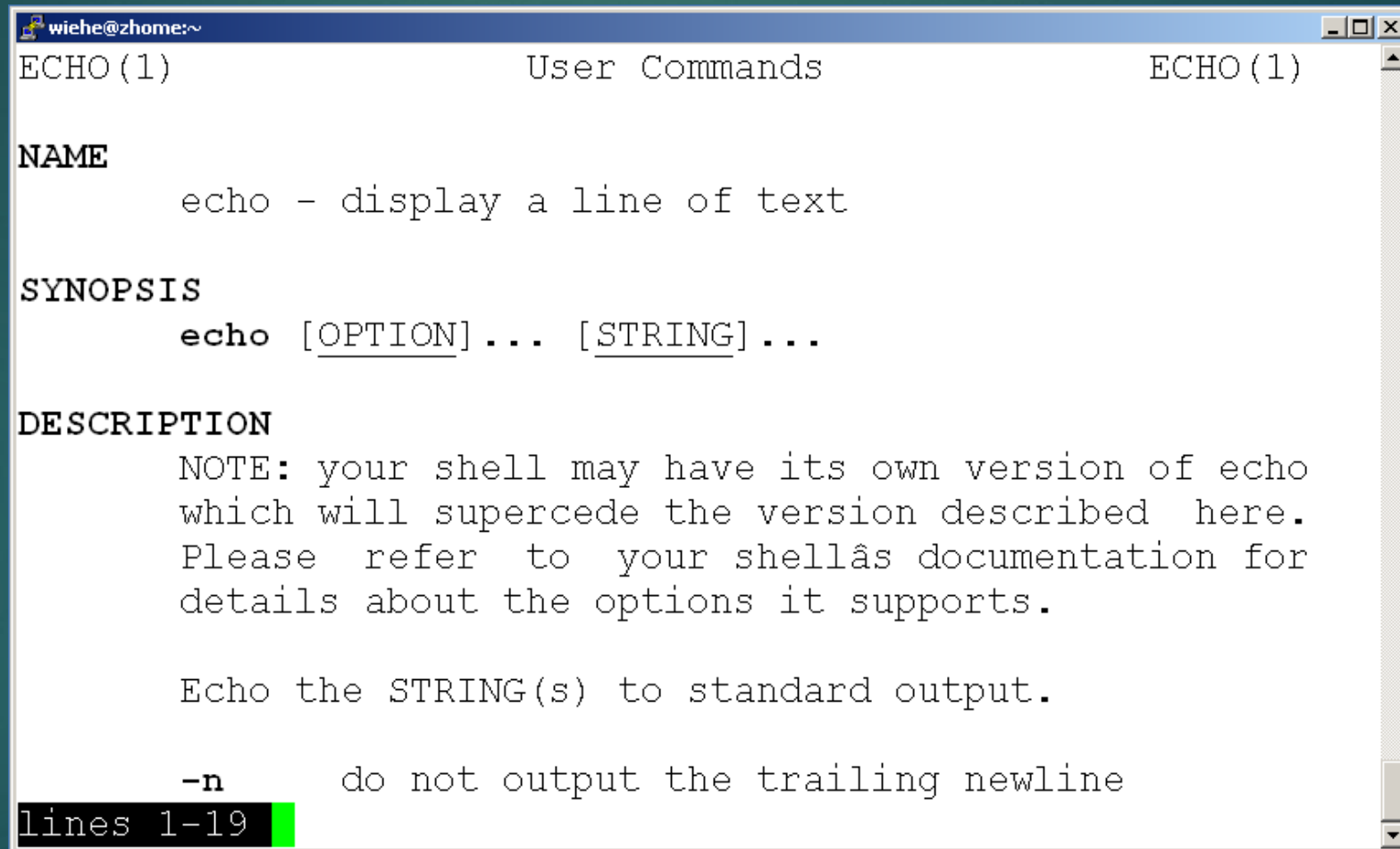
- ▶ Whenever you need help with a command type “**man**” and the command name

Help!

A terminal window with a blue title bar containing the text 'wiehe@zhome:~/linux_tutorial'. The window has standard window controls (minimize, maximize, close) on the right. The terminal content shows a sequence of commands and their outputs: 'man' is entered, followed by the prompt 'What manual page do you want?'. Then 'man echo' is entered. Finally, the prompt 'zhome:~/linux_tutorial\$' is shown with a green cursor block.

```
wiehe@zhome:~/linux_tutorial$ man
What manual page do you want?
zhome:~/linux_tutorial$ man echo
zhome:~/linux_tutorial$
```

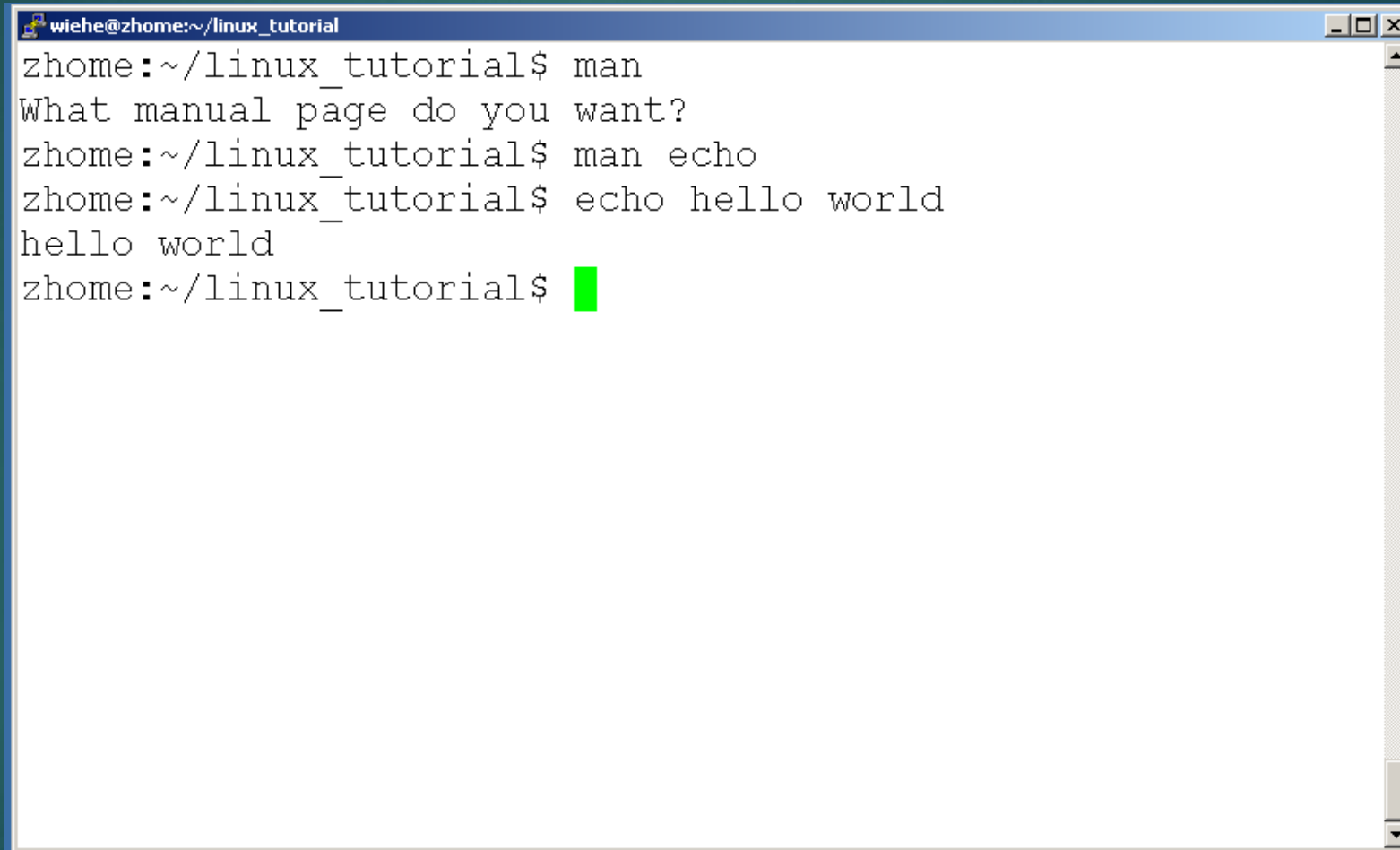
Help!



A terminal window titled 'wiehe@zhome:~' displays the help for the 'echo' command. The window has a title bar with standard Linux window controls. The content is organized into sections: 'NAME', 'SYNOPSIS', and 'DESCRIPTION'. The 'NAME' section describes 'echo' as a command to display a line of text. The 'SYNOPSIS' section shows the command syntax: 'echo [OPTION]... [STRING]...'. The 'DESCRIPTION' section includes a note about shell-specific versions of 'echo' and a brief explanation of the command's function. At the bottom, the option '-n' is described as 'do not output the trailing newline'. A status bar at the very bottom indicates 'lines 1-19'.

```
wiehe@zhome:~  
ECHO (1)                                User Commands                                ECHO (1)  
  
NAME  
    echo - display a line of text  
  
SYNOPSIS  
    echo [OPTION]... [STRING]...  
  
DESCRIPTION  
    NOTE: your shell may have its own version of echo  
    which will supercede the version described here.  
    Please refer to your shell's documentation for  
    details about the options it supports.  
  
    Echo the STRING(s) to standard output.  
  
    -n      do not output the trailing newline  
lines 1-19
```

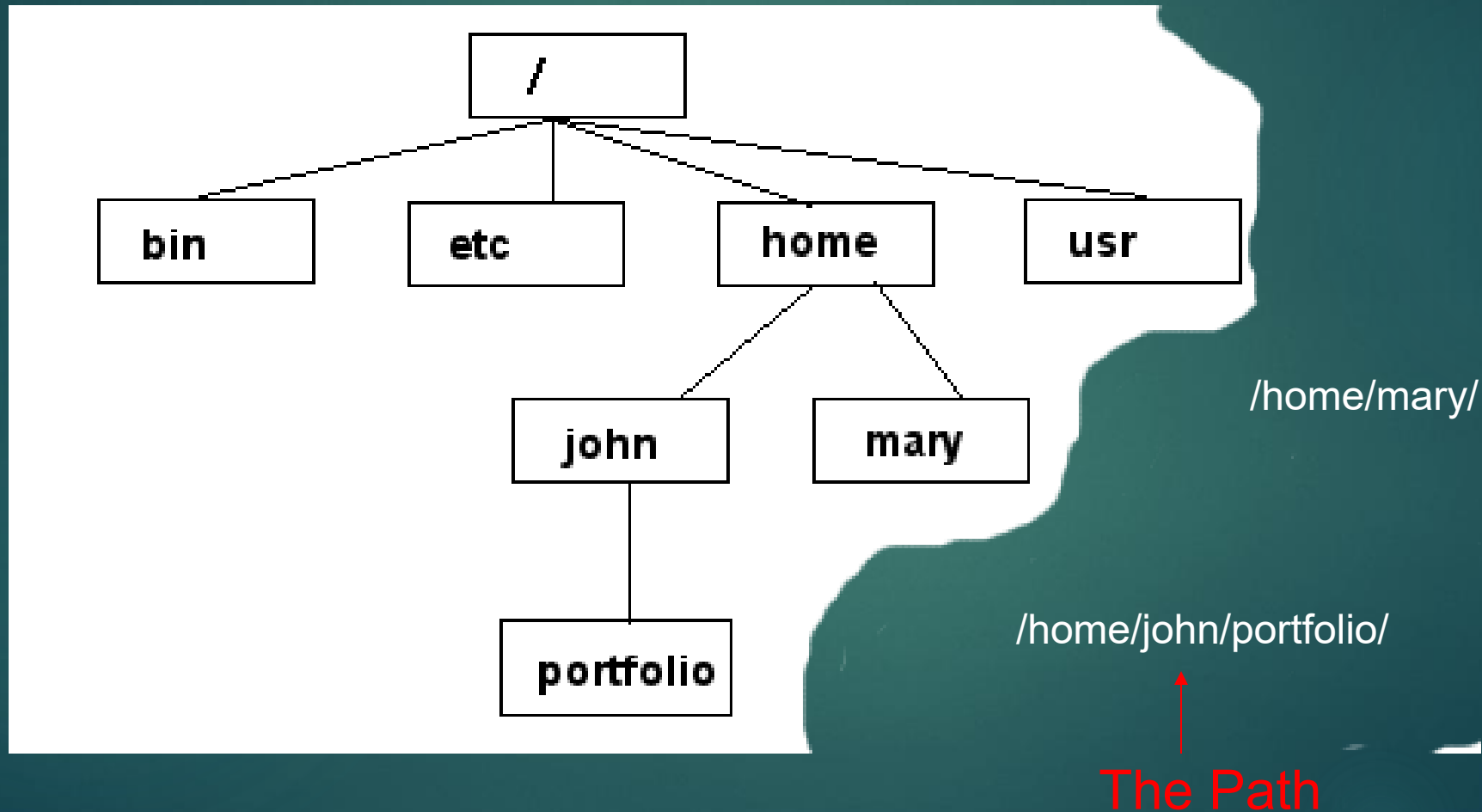
Help!



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ man
What manual page do you want?
zhome:~/linux_tutorial$ man echo
zhome:~/linux_tutorial$ echo hello world
hello world
zhome:~/linux_tutorial$
```

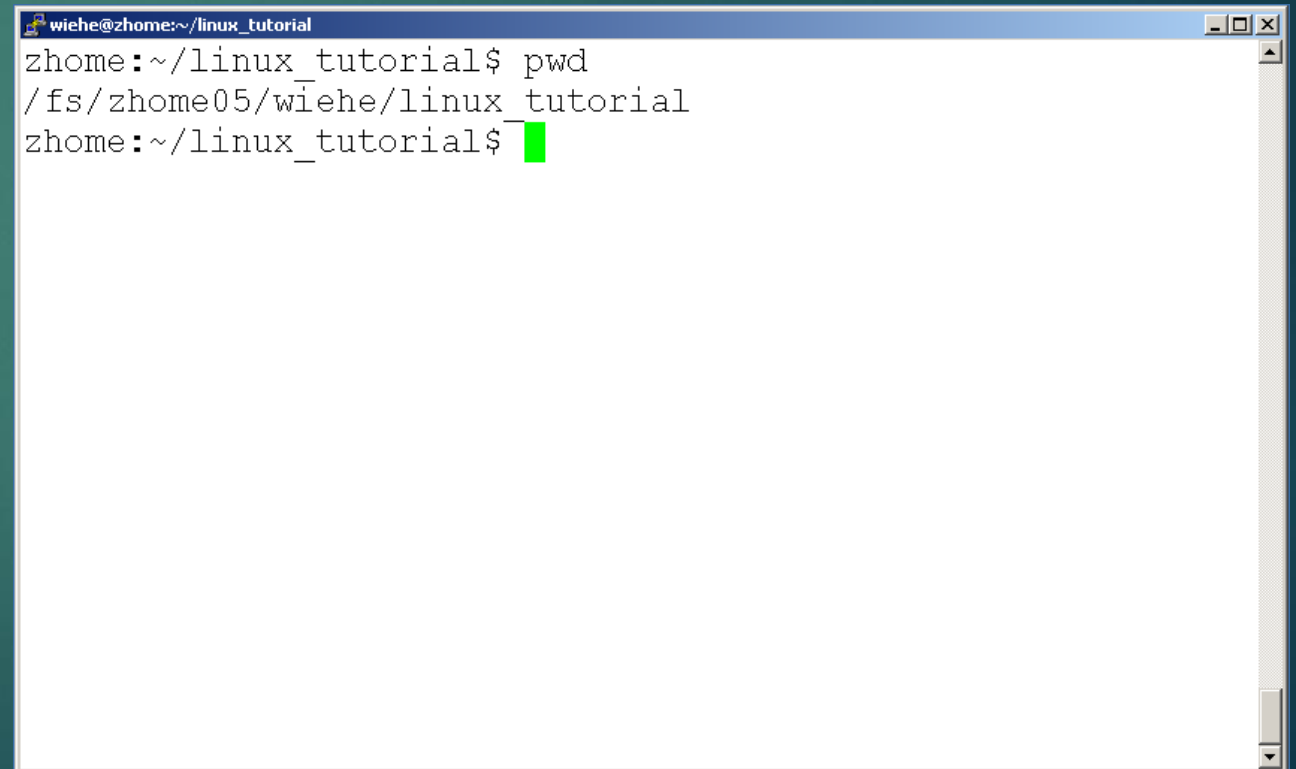
Unix/Linux File System

NOTE: Unix file names
are **CASE SENSITIVE!**



Command: pwd

- ▶ To find your current path use “pwd”
- ▶ (pwd = Print working directory)

A terminal window with a blue title bar containing the text 'wiehe@zhome:~/linux_tutorial'. The terminal has a white background and shows the following text: 'zhome:~/linux_tutorial\$ pwd' followed by the output '/fs/zhome05/wiehe/linux_tutorial' on the next line. The prompt 'zhome:~/linux_tutorial\$' is followed by a green cursor block.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ pwd
/fs/zhome05/wiehe/linux_tutorial
zhome:~/linux_tutorial$
```

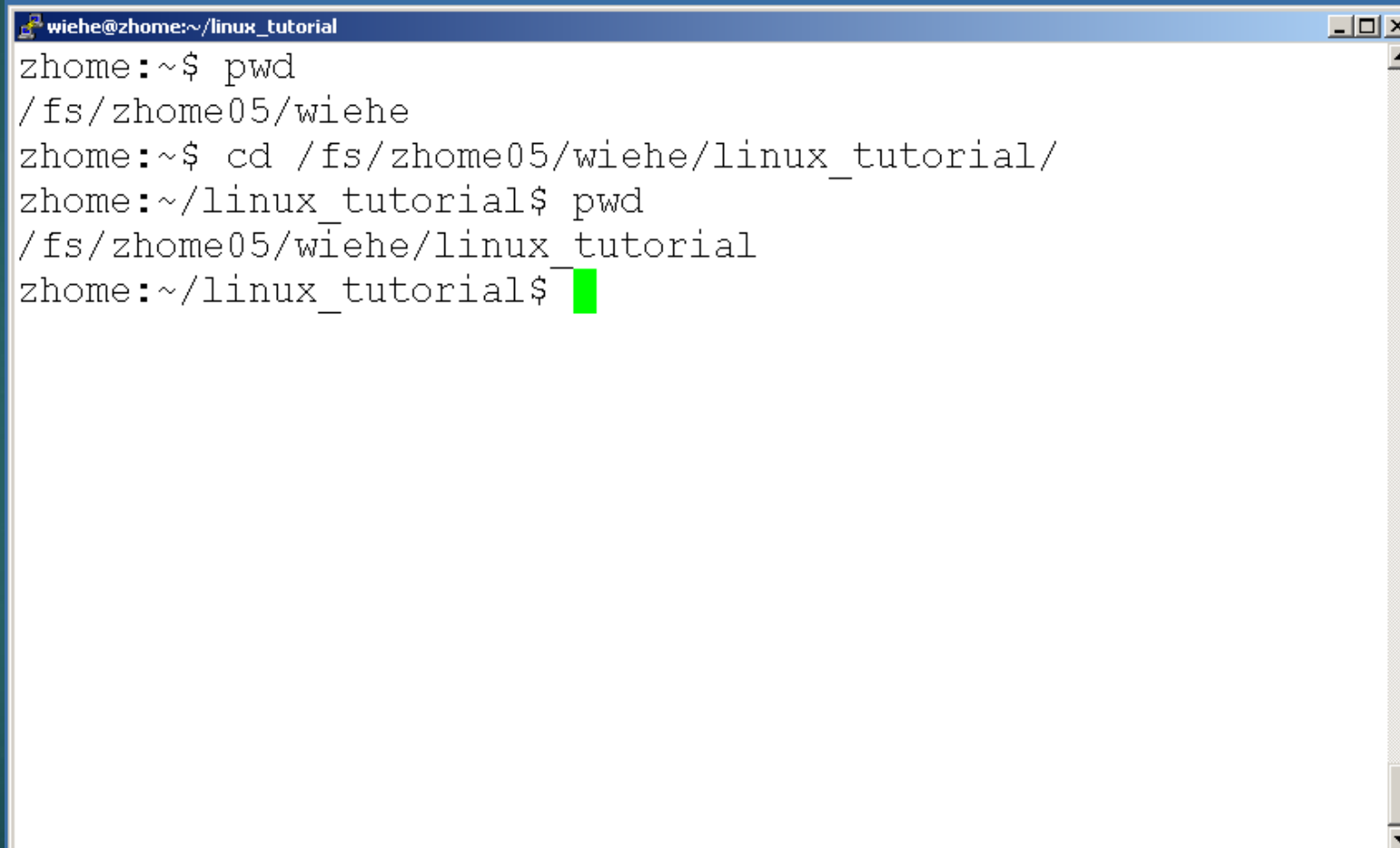
Touch

- ▶ Create a new file with no content

```
root@chatchaiasusPC:~# touch config.txt
root@chatchaiasusPC:~# ls
config.txt
root@chatchaiasusPC:~# |
```

Command: cd

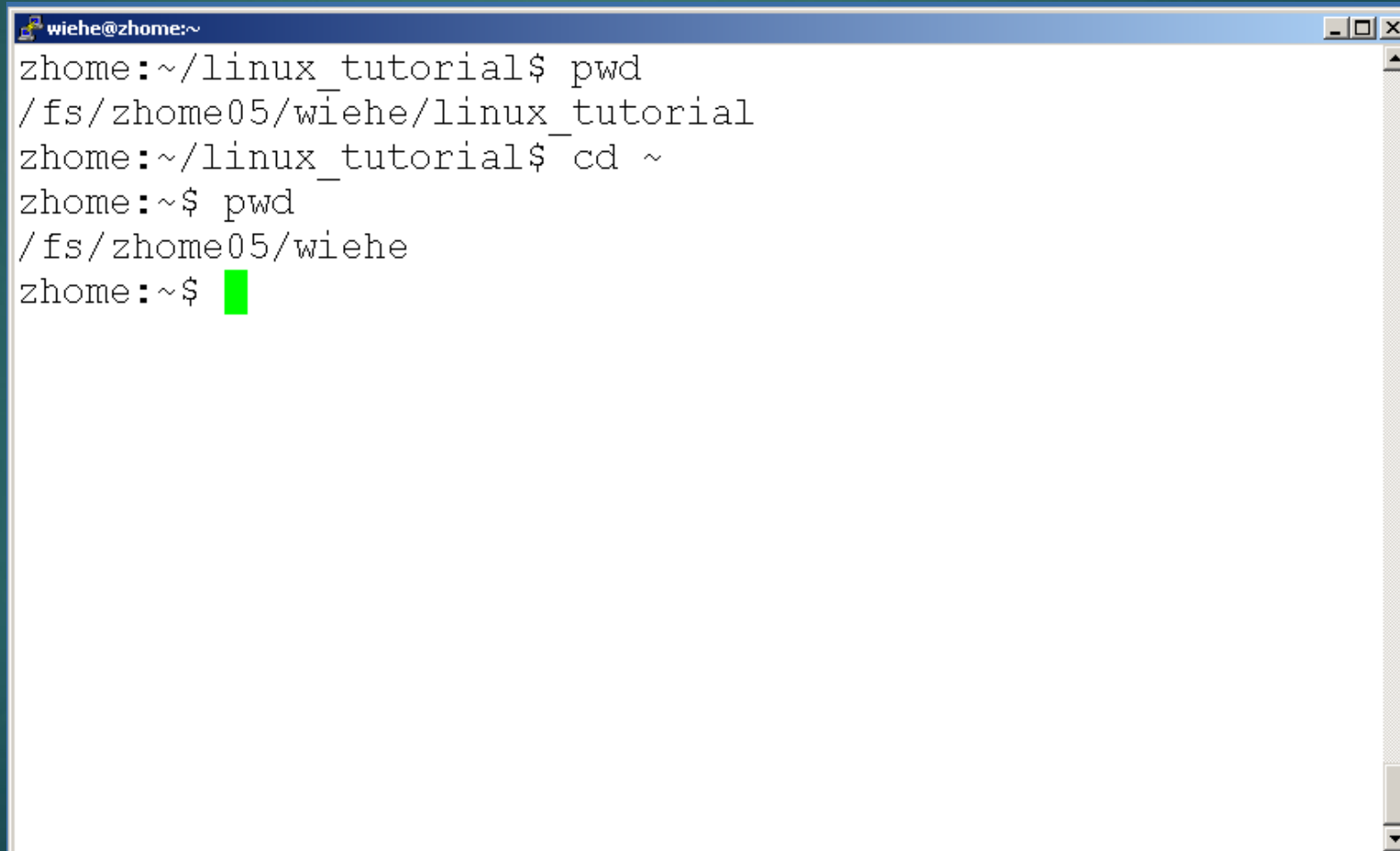
- ▶ To change to a specific directory use “cd”

A terminal window with a blue title bar containing the text 'wiehe@zhome:~/linux_tutorial'. The window shows a series of commands and their outputs. The first command is 'pwd', which outputs '/fs/zhome05/wiehe'. The second command is 'cd /fs/zhome05/wiehe/linux_tutorial/', which outputs 'zhome:~/linux_tutorial\$'. The third command is 'pwd', which outputs '/fs/zhome05/wiehe/linux_tutorial'. The fourth command is 'zhome:~/linux_tutorial\$' followed by a green cursor block.

```
wiehe@zhome:~/linux_tutorial
zhome:~$ pwd
/fs/zhome05/wiehe
zhome:~$ cd /fs/zhome05/wiehe/linux_tutorial/
zhome:~/linux_tutorial$ pwd
/fs/zhome05/wiehe/linux_tutorial
zhome:~/linux_tutorial$ █
```

Command: cd

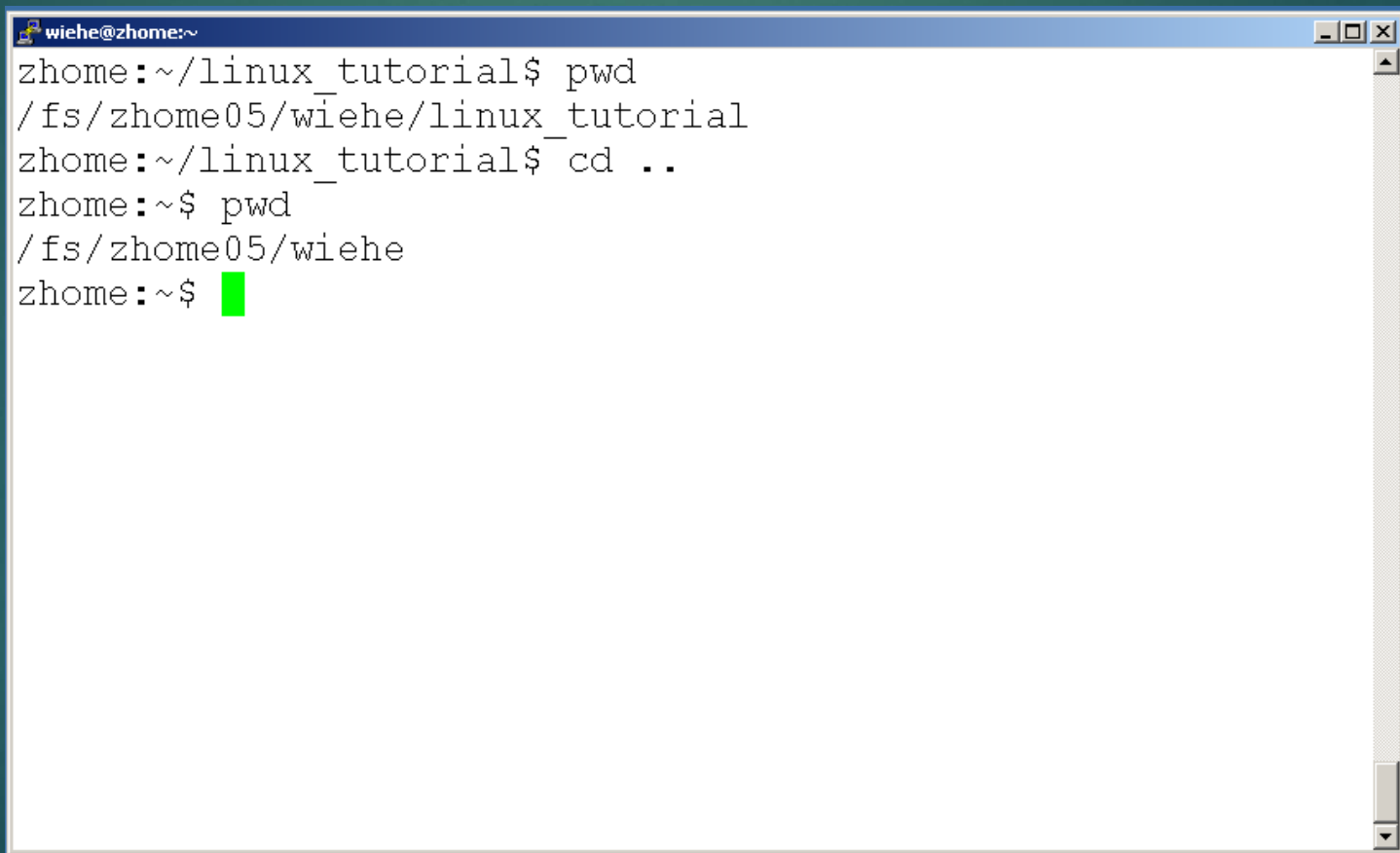
- ▶ “~” is the location of your home directory



```
wiehe@zhome:~  
zhome:~/linux_tutorial$ pwd  
/fs/zhome05/wiehe/linux_tutorial  
zhome:~/linux_tutorial$ cd ~  
zhome:~$ pwd  
/fs/zhome05/wiehe  
zhome:~$
```


Command: cd

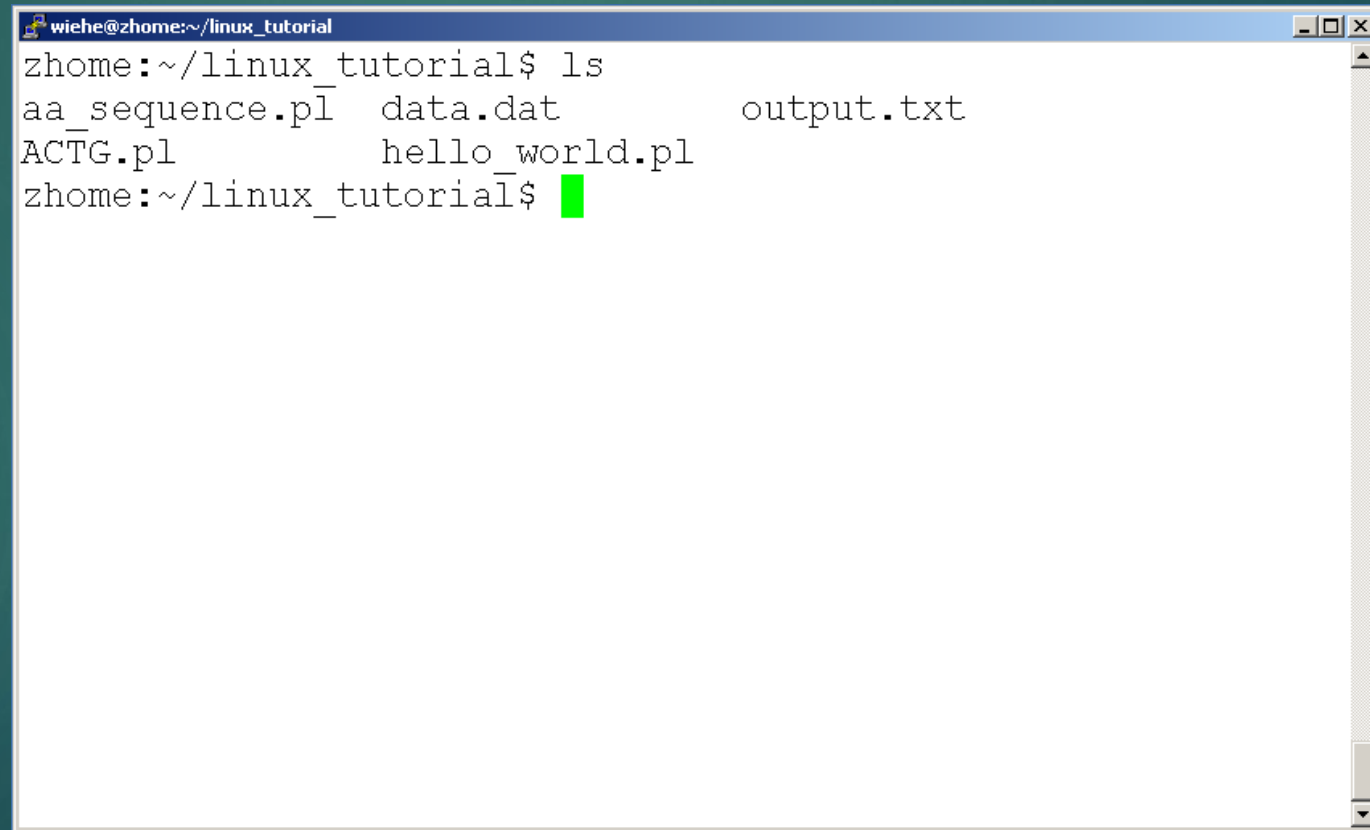
- ▶ “..” is the location of the directory below current one

A terminal window titled 'wiehe@zhome:~' with standard window controls. It shows a sequence of commands and their outputs: 'pwd' returns '/fs/zhome05/wiehe/linux_tutorial', 'cd ..' changes the directory, and a second 'pwd' returns '/fs/zhome05/wiehe'. A green cursor is visible at the end of the final prompt.

```
wiehe@zhome:~  
zhome:~/linux_tutorial$ pwd  
/fs/zhome05/wiehe/linux_tutorial  
zhome:~/linux_tutorial$ cd ..  
zhome:~$ pwd  
/fs/zhome05/wiehe  
zhome:~$
```

Command: ls

- To list the files in the current directory use “ls”

A terminal window titled 'wiehe@zhome:~/linux_tutorial' with standard window controls. The prompt is 'zhome:~/linux_tutorial\$'. The command 'ls' has been entered, and the output is displayed on the next line: 'aa_sequence.pl', 'data.dat', and 'output.txt' on the first line, and 'ACTG.pl' and 'hello_world.pl' on the second line. The prompt 'zhome:~/linux_tutorial\$' is followed by a green cursor.

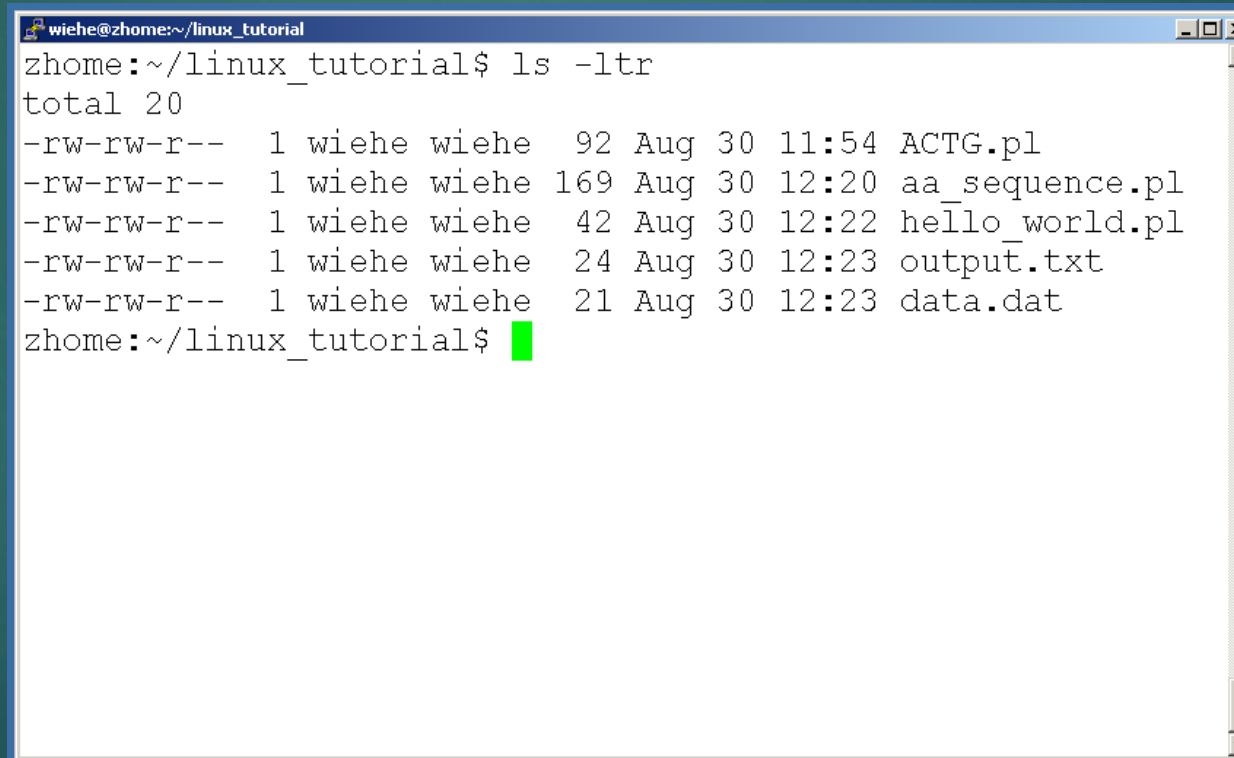
```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      output.txt
ACTG.pl        hello_world.pl
zhome:~/linux_tutorial$
```

Command: ls

- ▶ **ls** has many options
 - ▶ **-l** long list (displays lots of info)
 - ▶ **-t** sort by modification time
 - ▶ **-S** sort by size
 - ▶ **-h** list file sizes in human readable format
 - ▶ **-r** reverse the order
- ▶ “**man ls**” for more options
- ▶ Options can be combined: “ls -ltr”

Command: ls -ltr

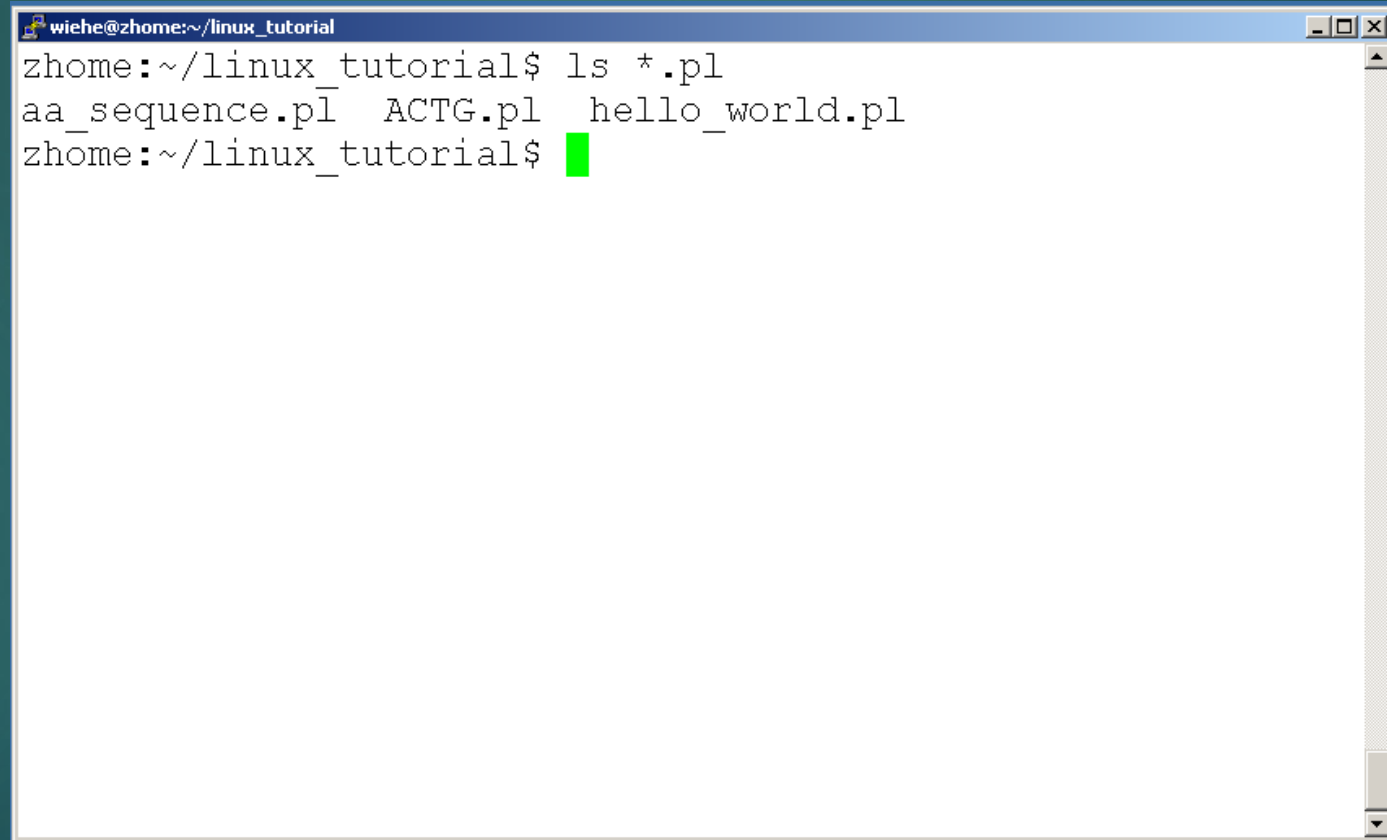
- List files by time in reverse order with long listing

A terminal window titled 'wiehe@zhome:~/linux_tutorial' showing the output of the 'ls -ltr' command. The output lists five files in reverse chronological order: ACTG.pl, aa_sequence.pl, hello_world.pl, output.txt, and data.dat. Each line shows permissions, file size, owner, group, date, time, and filename.

```
wiehe@zhome:~/linux_tutorial$ ls -ltr
total 20
-rw-rw-r-- 1 wiehe wiehe  92 Aug 30 11:54 ACTG.pl
-rw-rw-r-- 1 wiehe wiehe 169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r-- 1 wiehe wiehe  42 Aug 30 12:22 hello_world.pl
-rw-rw-r-- 1 wiehe wiehe  24 Aug 30 12:23 output.txt
-rw-rw-r-- 1 wiehe wiehe  21 Aug 30 12:23 data.dat
wiehe@zhome:~/linux_tutorial$
```

General Syntax: *

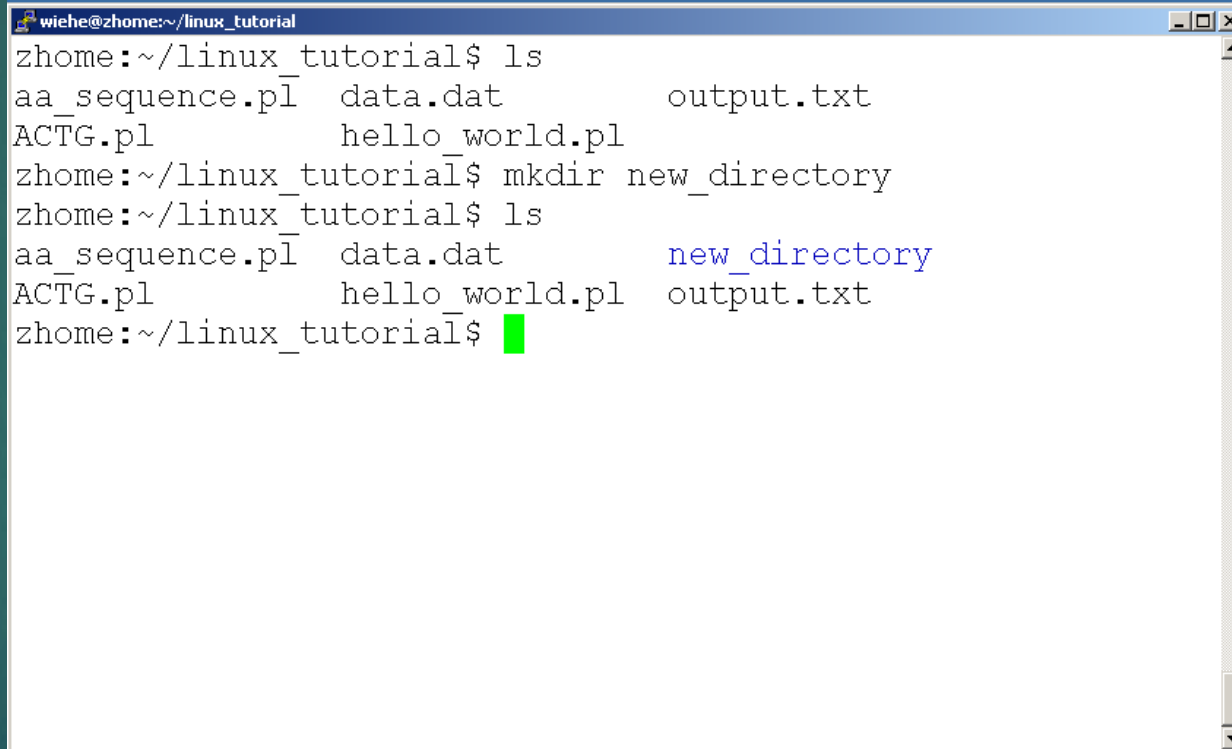
- ▶ “*” can be used as a wildcard in unix/linux

A terminal window titled 'wiehe@zhome:~/linux_tutorial' showing a command prompt. The user enters 'ls *.pl' and the output is 'aa_sequence.pl ACTG.pl hello_world.pl'. The prompt then returns to 'zhome:~/linux_tutorial\$' with a green cursor.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls *.pl
aa_sequence.pl  ACTG.pl  hello_world.pl
zhome:~/linux_tutorial$
```

Command: mkdir

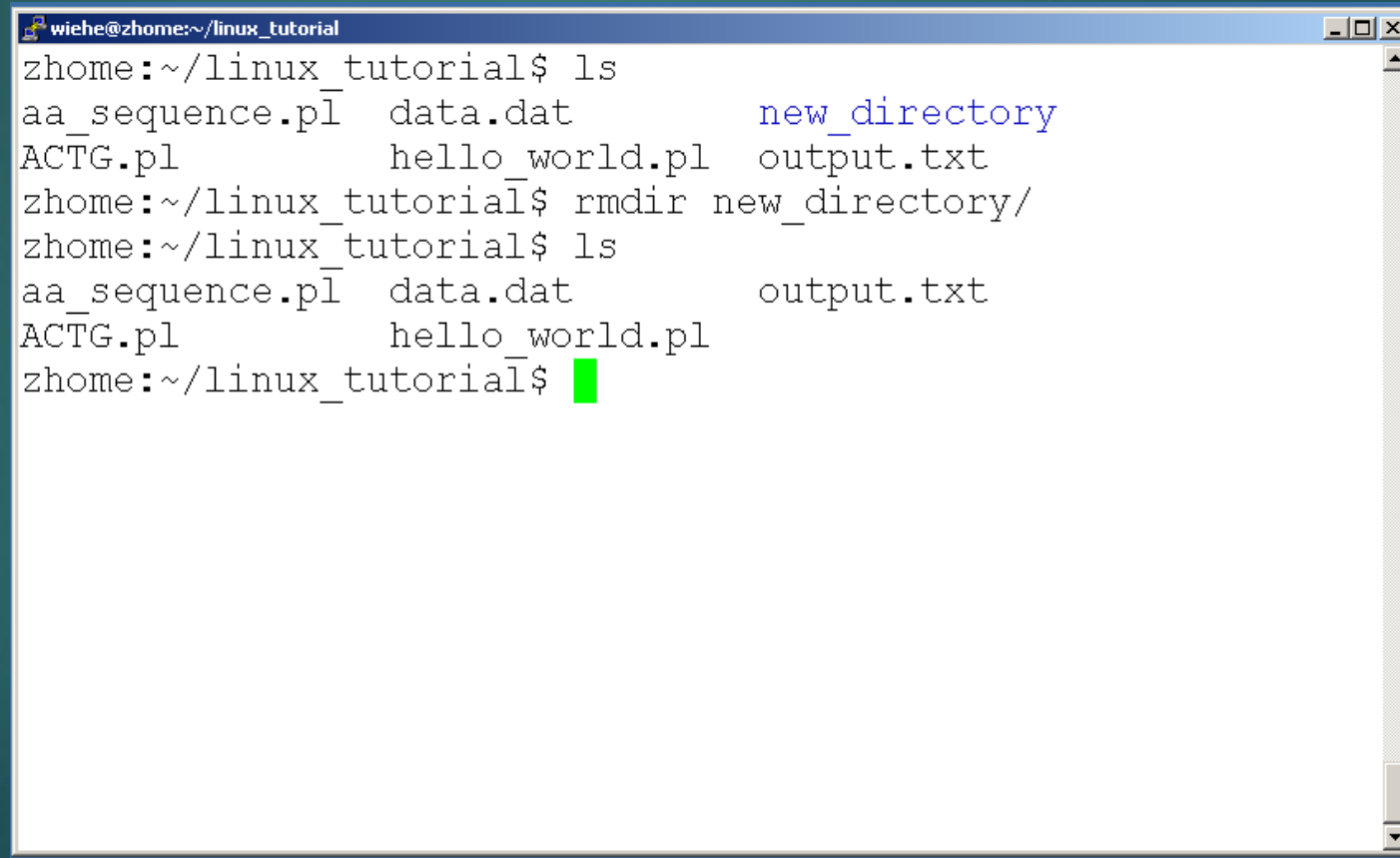
- ▶ To create a new directory use “mkdir”

A terminal window titled 'wiehe@zhome:~/linux_tutorial' showing the execution of the 'mkdir' command. The window has a blue title bar and standard window controls. The terminal text shows the initial directory listing, the 'mkdir new_directory' command being entered and executed, and a second directory listing that confirms the new directory's creation.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      output.txt
ACTG.pl        hello_world.pl
zhome:~/linux_tutorial$ mkdir new_directory
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      new_directory  output.txt
ACTG.pl        hello_world.pl
zhome:~/linux_tutorial$
```

Command: rmdir

- ▶ To remove an empty directory use “rmdir”

A terminal window titled 'wiehe@zhome:~/linux_tutorial' showing a sequence of commands and their outputs. The user first lists the contents of the directory, which includes 'aa_sequence.pl', 'data.dat', 'new_directory', 'ACTG.pl', 'hello_world.pl', and 'output.txt'. Then, the user runs 'rmdir new_directory/'. Finally, the user lists the directory contents again, and 'new_directory' is no longer present. The prompt is a green cursor.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      new_directory
ACTG.pl        hello_world.pl output.txt
zhome:~/linux_tutorial$ rmdir new_directory/
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      output.txt
ACTG.pl        hello_world.pl
zhome:~/linux_tutorial$
```

Displaying a file

- ▶ Various ways to display a file in Unix
 - ▶ cat
 - ▶ less
 - ▶ head
 - ▶ tail

Command: cat

- ▶ Dumps an entire file to standard output
- ▶ Good for displaying short, simple files

Command: less

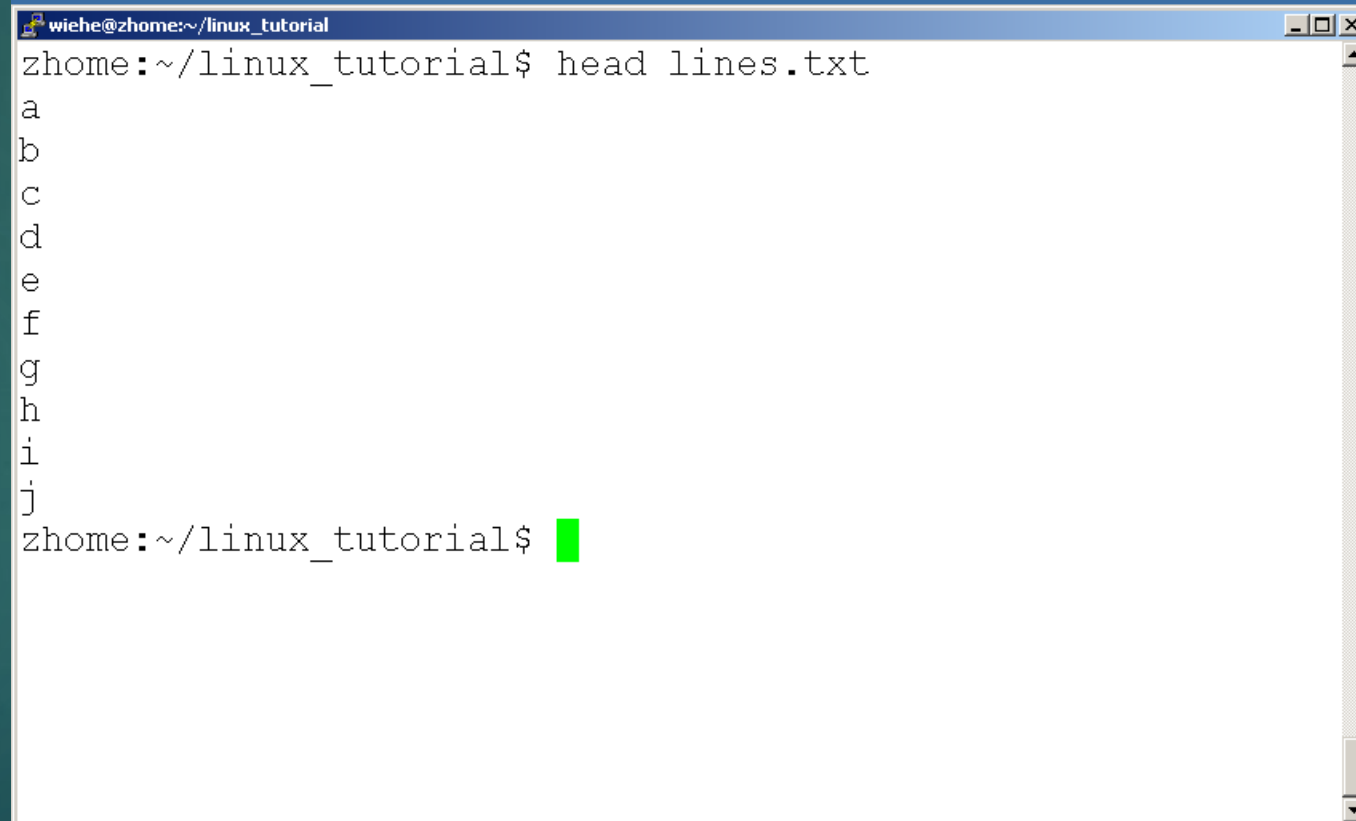
- ▶ “less” displays a file, allowing forward/backward movement within it
 - ▶ return scrolls forward one line, space one page
 - ▶ y scrolls back one line, b one page
- ▶ use “/” to search for a string
- ▶ Press **q** to quit

Command: head

- ▶ “head” displays the top part of a file
- ▶ By default it shows the first 10 lines
- ▶ -n option allows you to change that
- ▶ “head -n50 file.txt” displays the first 50 lines of file.txt

Command: head

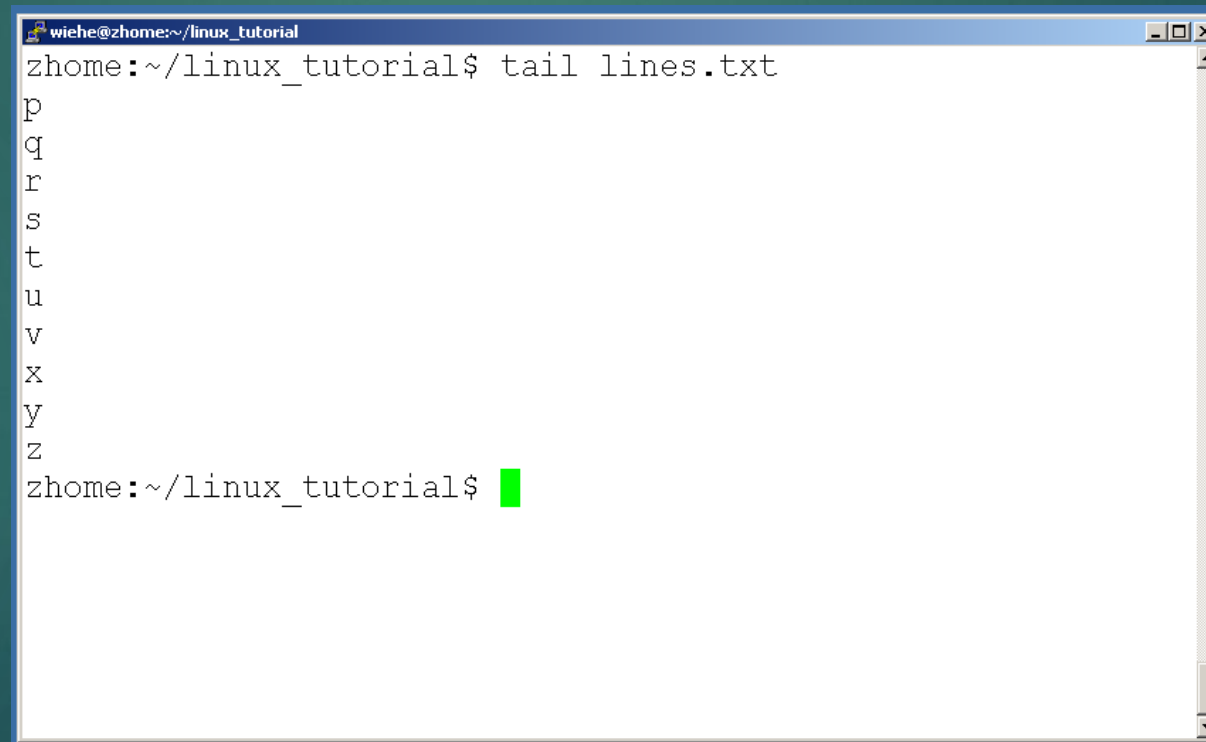
- ▶ Here's an example of using "head":

A terminal window with a blue title bar containing the text 'wiehe@zhome:~/linux_tutorial'. The terminal shows the command 'head lines.txt' being executed, which outputs the first ten lines of a file: 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', and 'j'. The prompt 'zhome:~/linux_tutorial\$' is shown again with a green cursor.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ head lines.txt
a
b
c
d
e
f
g
h
i
j
zhome:~/linux_tutorial$
```

Command: tail

- ▶ Same as head, but shows the last lines

A terminal window with a blue title bar containing the text 'wiehe@zhome:~/linux_tutorial'. The terminal shows the command 'tail lines.txt' being executed. The output consists of the letters 'p', 'q', 'r', 's', 't', 'u', 'v', 'x', 'y', and 'z' on separate lines. Below the output, the prompt 'zhome:~/linux_tutorial\$' is followed by a green cursor.

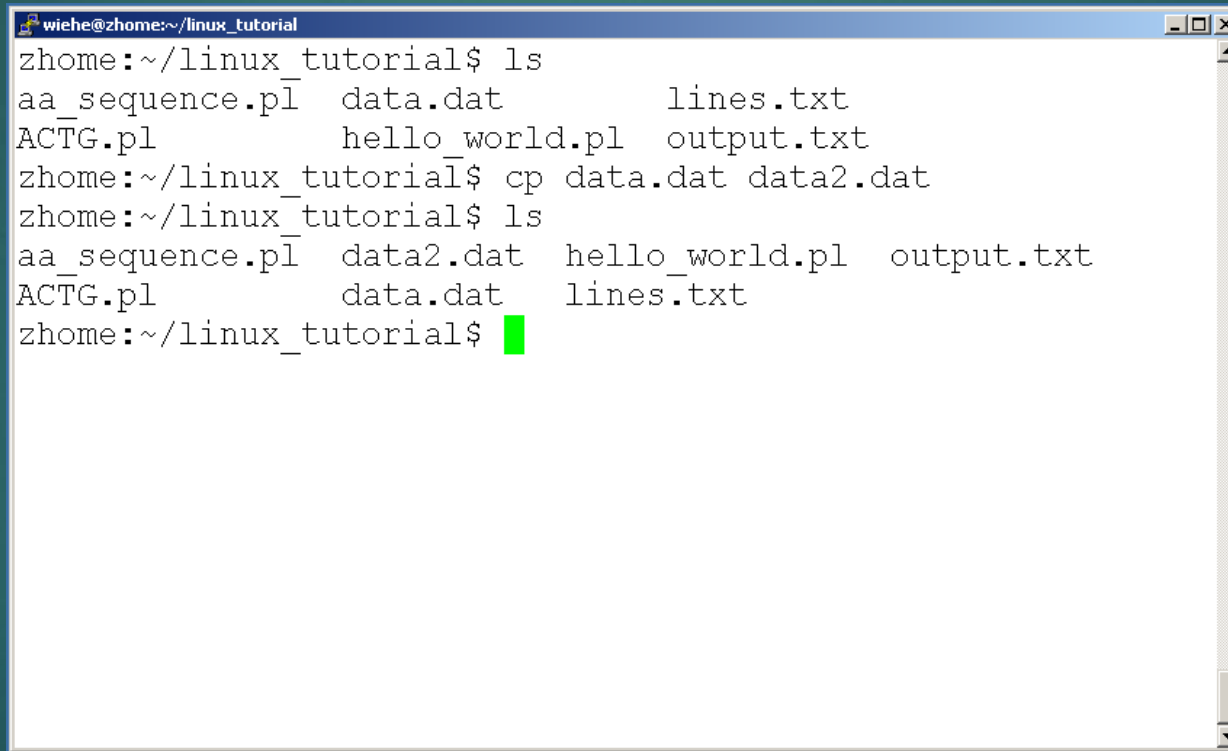
```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ tail lines.txt
p
q
r
s
t
u
v
x
y
z
zhome:~/linux_tutorial$
```

File Commands

- ▶ Copying a file: `cp`
- ▶ Move or rename a file: `mv`
- ▶ Remove a file: `rm`

Command: cp

- ▶ To copy a file use “cp”

A terminal window titled 'wiehe@zhome:~/linux_tutorial' showing a sequence of commands and their outputs. The user first lists files in the current directory, then uses the 'cp' command to copy 'data.dat' to 'data2.dat', and lists the files again to confirm the copy.

```
wiehe@zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      lines.txt
ACTG.pl        hello_world.pl output.txt
zhome:~/linux_tutorial$ cp data.dat data2.dat
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data2.dat    hello_world.pl  output.txt
ACTG.pl        data.dat     lines.txt
zhome:~/linux_tutorial$
```

Command: mv

- ▶ To move a file to a different location use “mv”

```
wiehe@zhome:~/linux_tutorial/new_directory
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data2.dat  hello_world.pl  output.txt
ACTG.pl        data.dat   lines.txt
zhome:~/linux_tutorial$ mkdir new_directory
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data2.dat  hello_world.pl  new_directory
ACTG.pl        data.dat   lines.txt       output.txt
zhome:~/linux_tutorial$ mv data2.dat ./new_directory/
zhome:~/linux_tutorial$ cd new_directory/
zhome:~/linux_tutorial/new_directory$ ls
data2.dat
zhome:~/linux_tutorial/new_directory$
```

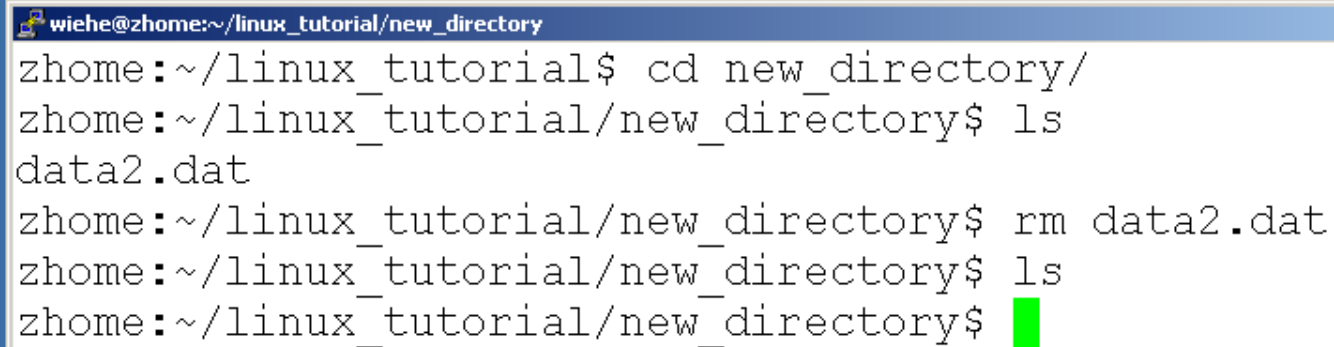

Command: mv

- ▶ mv can also be used to rename a file

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      lines.txt     output.txt
ACTG.pl        hello_world.pl new_directory
zhome:~/linux_tutorial$ mv output.txt input.txt
zhome:~/linux_tutorial$ ls
aa_sequence.pl  data.dat      input.txt     new_directory
ACTG.pl        hello_world.pl lines.txt
zhome:~/linux_tutorial$
```

Command: rm

- ▶ To remove a file use “rm”



```
wiehe@zhome:~/linux_tutorial/new_directory
zhome:~/linux_tutorial$ cd new_directory/
zhome:~/linux_tutorial/new_directory$ ls
data2.dat
zhome:~/linux_tutorial/new_directory$ rm data2.dat
zhome:~/linux_tutorial/new_directory$ ls
zhome:~/linux_tutorial/new_directory$
```

A terminal window with a blue title bar containing the text "wiehe@zhome:~/linux_tutorial/new_directory". The window shows a series of commands and their outputs. The user navigates to the "new_directory" folder, lists its contents (showing "data2.dat"), and then uses the "rm" command to remove "data2.dat". A subsequent "ls" command confirms the file has been removed. The prompt ends with a green cursor.

Command: rm

- ▶ To remove a file “recursively”: `rm -r`
- ▶ Used to remove all files and directories
- ▶ Be very careful, deletions are permanent in Unix/Linux

File permissions

- ▶ Each file in Unix/Linux has an associated permission level
- ▶ This allows the user to prevent others from reading/writing/executing their files or directories
- ▶ Use “`ls -l filename`” to find the permission level of that file

Permission levels

- ▶ “r” means “read only” permission
- ▶ “w” means “write” permission
- ▶ “x” means “execute” permission
 - ▶ In case of directory, “x” grants permission to list directory contents

File Permissions

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls -l
total 28
-rw-rw-r-- 1 wiehe wiehe 169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r-- 1 wiehe wiehe 92 Aug 30 11:54 ACTG.pl
-rw-rw-r-- 1 wiehe wiehe 21 Aug 30 12:23 data.dat
-rw-rw-r-- 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
-rw-rw-r-- 1 wiehe wiehe 24 Aug 30 12:23 input.txt
-rw-rw-r-- 1 wiehe wiehe 50 Aug 30 13:13 lines.txt
drwxrwxr-x 2 wiehe wiehe 4096 Aug 30 13:19 new_directory
zhome:~/linux_tutorial$
```

User (you)

File Permissions

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls -l
total 28
-rw-rw-r-- 1 wiehe wiehe 169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r-- 1 wiehe wiehe 92 Aug 30 11:54 ACTG.pl
-rw-rw-r-- 1 wiehe wiehe 21 Aug 30 12:23 data.dat
-rw-rw-r-- 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
-rw-rw-r-- 1 wiehe wiehe 24 Aug 30 12:23 input.txt
-rw-rw-r-- 1 wiehe wiehe 50 Aug 30 13:13 lines.txt
drwxrwxr-x 2 wiehe wiehe 4096 Aug 30 13:19 new_directory
zhome:~/linux_tutorial$
```

Group

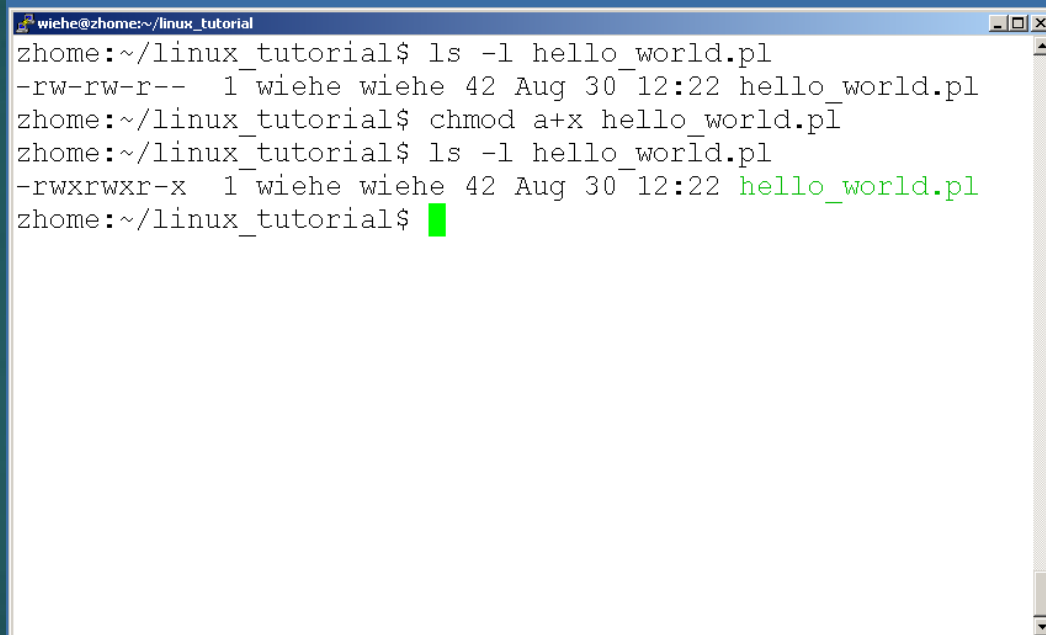
File Permissions

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls -l
total 28
-rw-rw-r-- 1 wiehe wiehe 169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r-- 1 wiehe wiehe 92 Aug 30 11:54 ACTG.pl
-rw-rw-r-- 1 wiehe wiehe 21 Aug 30 12:23 data.dat
-rw-rw-r-- 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
-rw-rw-r-- 1 wiehe wiehe 24 Aug 30 12:23 input.txt
-rw-rw-r-- 1 wiehe wiehe 50 Aug 30 13:13 lines.txt
drwxrwxr-x 2 wiehe wiehe 4096 Aug 30 13:19 new_directory
zhome:~/linux_tutorial$
```

“The World”

Command: chmod

- ▶ If you own the file, you can change it's permissions with “chmod”
 - ▶ Syntax: `chmod [user/group/others/all]+[permission] [file(s)]`
 - ▶ Below we grant execute permission to all:



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls -l hello_world.pl
-rw-rw-r-- 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
zhome:~/linux_tutorial$ chmod a+x hello_world.pl
zhome:~/linux_tutorial$ ls -l hello_world.pl
-rwxrwxr-x 1 wiehe wiehe 42 Aug 30 12:22 hello_world.pl
zhome:~/linux_tutorial$
```

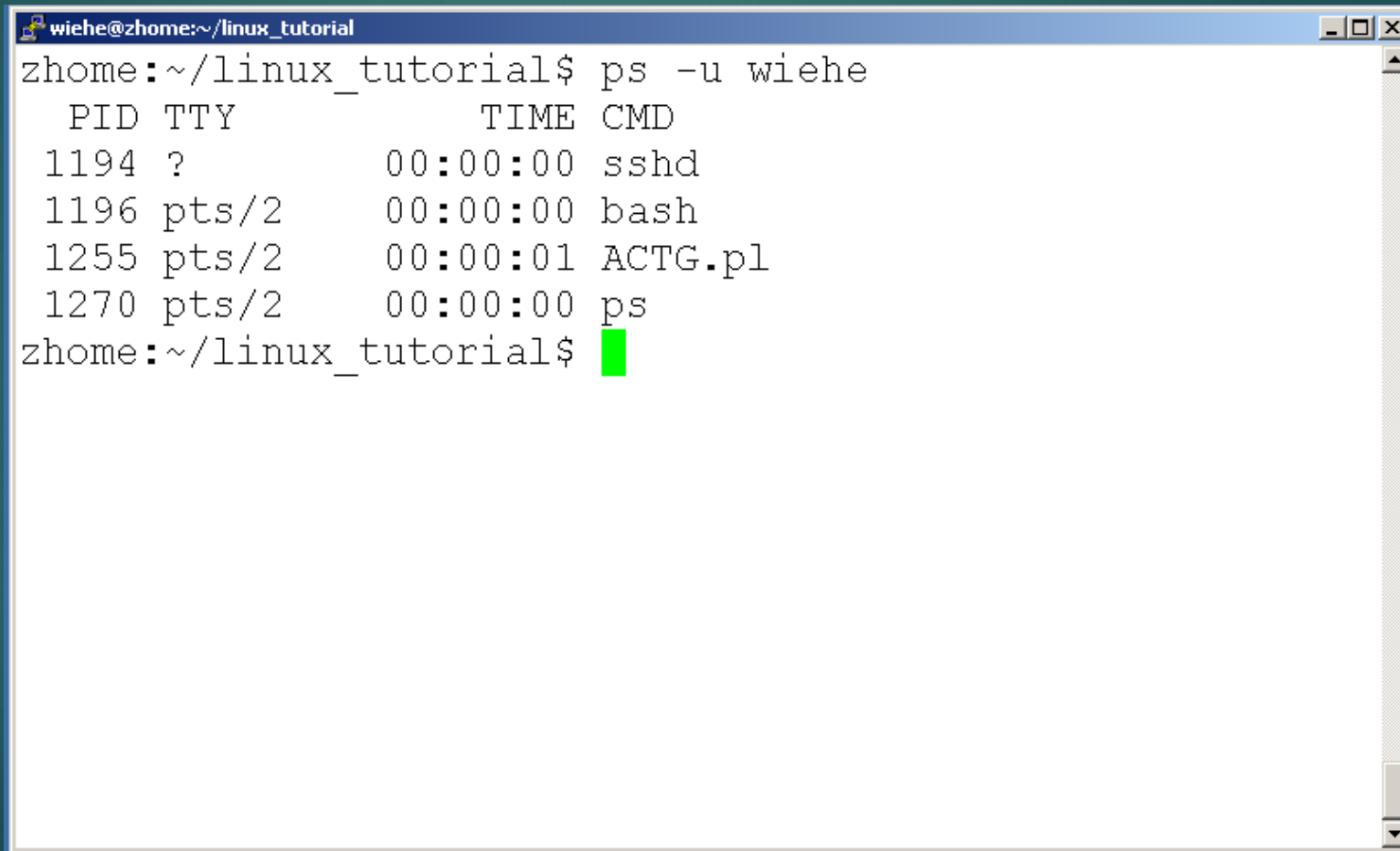
Permission

644 (user = read + write, group = read, others = read)
755 (user = read + write + execute, group = read + execute,
others = read + execute)
600 (user = read + write, group = none, others = none)
620 (user = read + write, group = write, others = none)

644 = rw-r--r--
755 = rwxr-xr-x
600 = rw-----
620 = rw--w----

Command: ps

- ▶ To view the processes that you're running:

A terminal window titled 'wiehe@zhome:~/linux_tutorial' showing the output of the 'ps -u wiehe' command. The output is a table with four columns: PID, TTY, TIME, and CMD. The processes listed are sshd (PID 1194), bash (PID 1196), ACTG.pl (PID 1255), and ps (PID 1270). The terminal prompt is 'zhome:~/linux_tutorial\$' followed by a green cursor.

```
wiehe@zhome:~/linux_tutorial$ ps -u wiehe
  PID TTY          TIME CMD
 1194 ?            00:00:00 sshd
 1196 pts/2        00:00:00 bash
 1255 pts/2        00:00:01 ACTG.pl
 1270 pts/2        00:00:00 ps
zhome:~/linux_tutorial$
```

Command: top

- To view the CPU usage of all processes:

```
wiehe@zhome:~/linux_tutorial
top - 13:46:33 up 50 days,  4:26,  2 users,  load avera
Tasks:  total,      running,      sleeping,      stoppe
Cpu(s):    us,       sy,          ni,          id,          w
Mem:       total,          used,          free,
Swap:      total,          used,          free,

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM
3403	root	15	0	0	0	0	S	0.7	0.0
1	root	16	0	1604	324	292	S	0.0	0.0
2	root	RT	0	0	0	0	S	0.0	0.0
3	root	34	19	0	0	0	S	0.0	0.0
4	root	RT	0	0	0	0	S	0.0	0.0
5	root	34	19	0	0	0	S	0.0	0.0
6	root	RT	0	0	0	0	S	0.0	0.0
7	root	34	19	0	0	0	S	0.0	0.0
8	root	RT	0	0	0	0	S	0.0	0.0
9	root	34	19	0	0	0	S	0.0	0.0

Command: kill

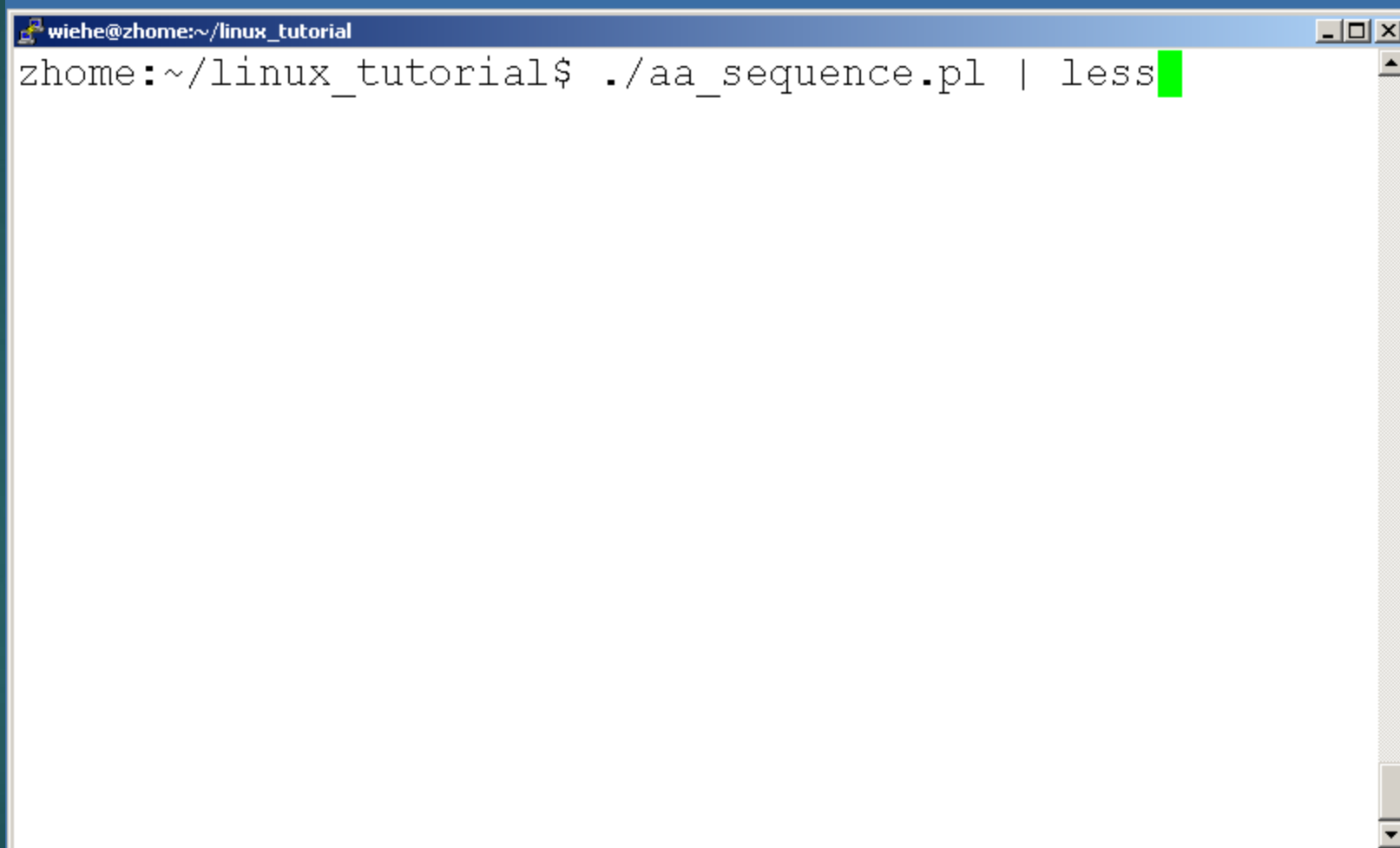
- ▶ To terminate a process use “kill”

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ps -u wiehe
  PID TTY          TIME CMD
 1194 ?            00:00:00 sshd
 1196 pts/2        00:00:00 bash
 1255 pts/2        00:00:01 ACTG.pl
 1287 pts/2        00:00:00 ps
zhome:~/linux_tutorial$ kill -9 1255
[1]+  Killed                  ./ACTG.pl
zhome:~/linux_tutorial$ ps -u wiehe
  PID TTY          TIME CMD
 1194 ?            00:00:00 sshd
 1196 pts/2        00:00:00 bash
 1289 pts/2        00:00:00 ps
zhome:~/linux_tutorial$
```

Input/Output Redirection (“piping”)

- ▶ Programs can output to other programs
- ▶ Called “piping”
- ▶ “program_a | program_b”
 - ▶ program_a's output becomes program_b's input
- ▶ “program_a > file.txt”
 - ▶ program_a's output is written to a file called “file.txt”
- ▶ “program_a < input.txt”
 - ▶ program_a gets its input from a file called “input.txt”

A few examples of piping

A terminal window with a blue title bar containing the text 'wiehe@zhome:~/linux_tutorial'. The terminal has a white background and shows the command 'zhome:~/linux_tutorial\$./aa_sequence.pl | less' followed by a green cursor. The window includes standard Linux window controls (minimize, maximize, close) in the top right and a vertical scrollbar on the right side.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ./aa_sequence.pl | less
```

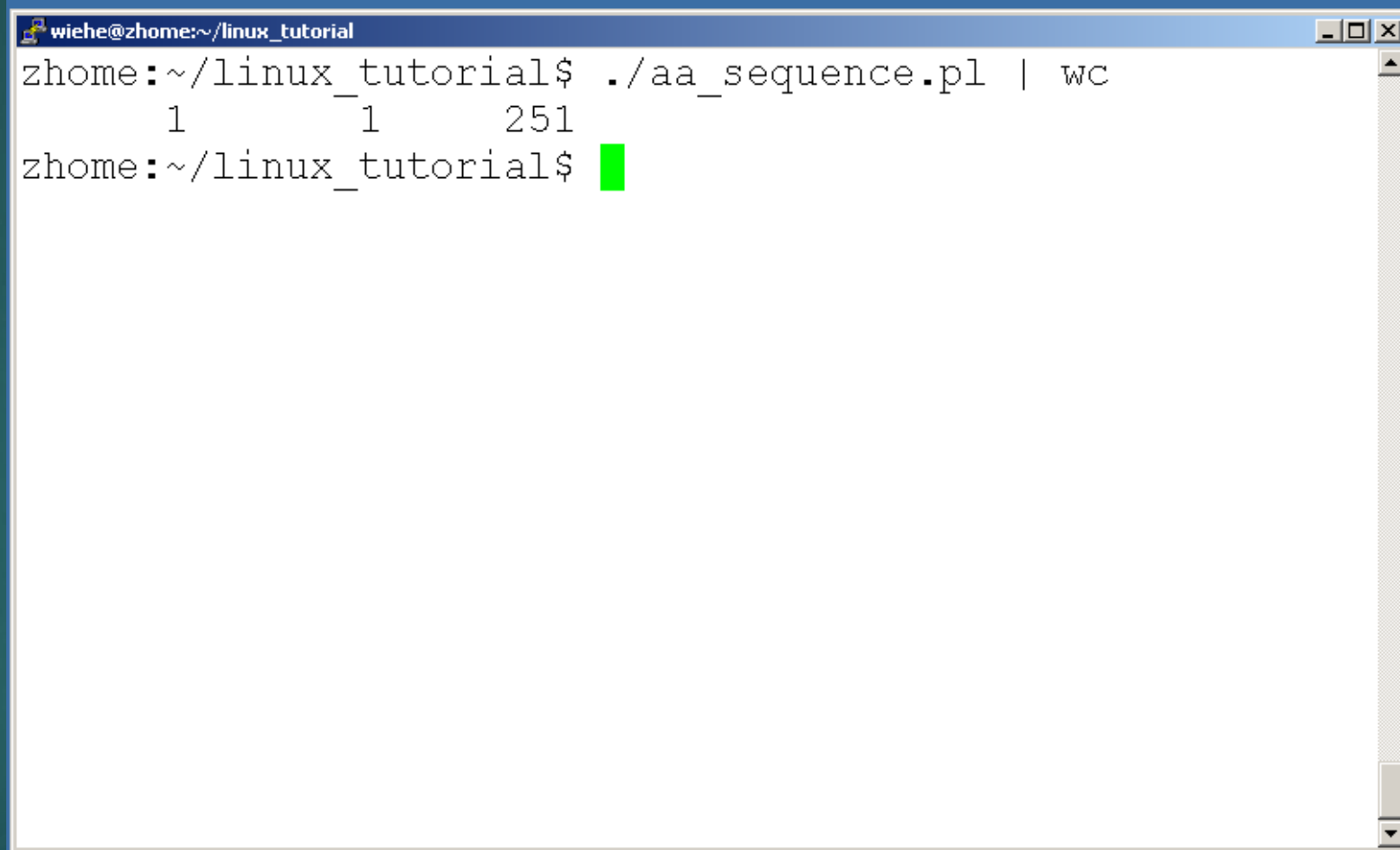
A few examples of piping

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  hello_world.pl  new_directory
ACTG.pl        input.txt
data.dat       lines.txt
zhome:~/linux_tutorial$ ./aa_sequence.pl > sequence.txt
zhome:~/linux_tutorial$ ls
aa_sequence.pl  hello_world.pl  new_directory
ACTG.pl        input.txt       sequence.txt
data.dat       lines.txt
zhome:~/linux_tutorial$ less sequence.txt
```


Command: wc

- ▶ To count the characters, words, and lines in a file use “wc”
- ▶ The first column in the output is lines, the second is words, and the last is characters

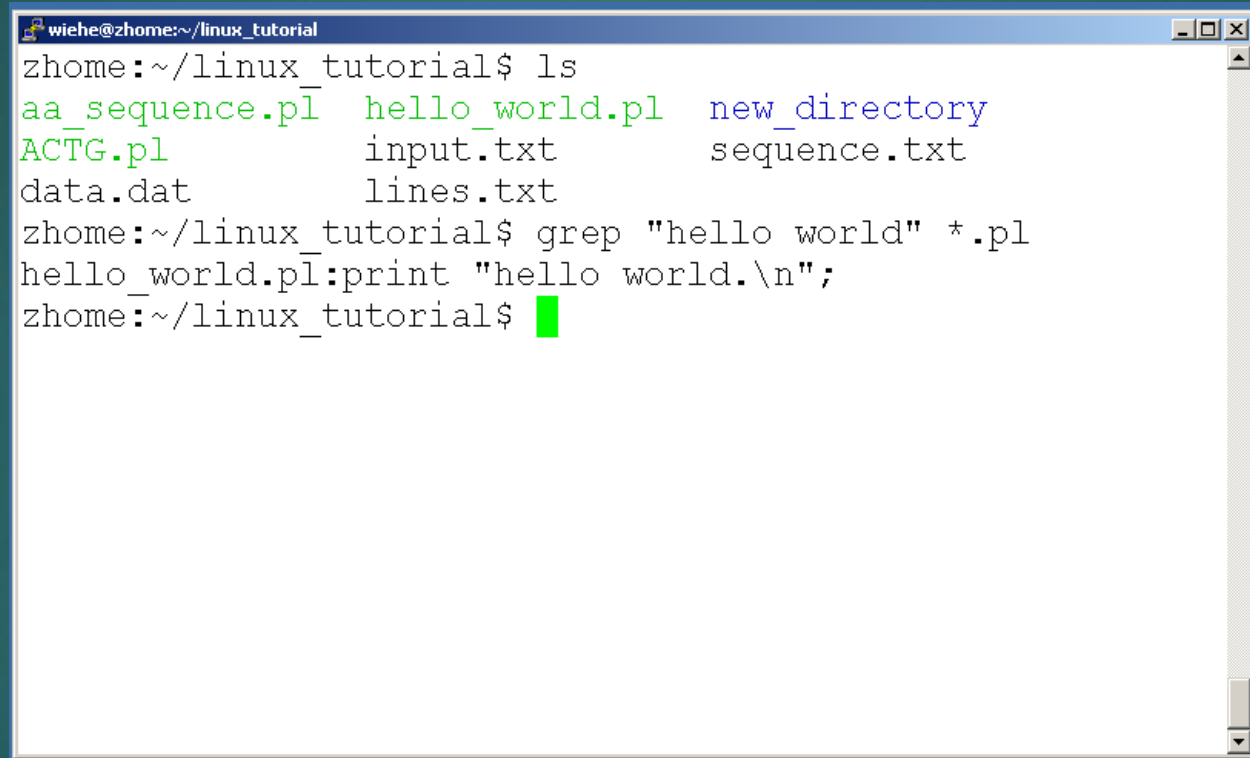
A few examples of piping

A terminal window with a blue title bar containing the text 'wiehe@zhome:~/linux_tutorial'. The terminal shows a command being executed and its output. The command is './aa_sequence.pl | wc'. The output is '1 1 251'. The prompt 'zhome:~/linux_tutorial\$' is shown twice, once before the command and once after the output. A green cursor is visible after the second prompt.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ./aa_sequence.pl | wc
      1      1     251
zhome:~/linux_tutorial$
```

Command: grep

- ▶ To search files in a directory for a specific string use “grep”

A terminal window titled 'wiehe@zhome:~/linux_tutorial' showing a series of commands and their outputs. The user first runs 'ls', which lists files in the directory: 'aa_sequence.pl', 'hello_world.pl', 'new_directory', 'ACTG.pl', 'input.txt', 'sequence.txt', 'data.dat', and 'lines.txt'. Then, the user runs 'grep "hello world" *.pl', which outputs 'hello_world.pl:print "hello world.\n";'. The terminal window has a standard Linux look with a blue title bar and a scrollable text area.

```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ls
aa_sequence.pl  hello_world.pl  new_directory
ACTG.pl         input.txt       sequence.txt
data.dat        lines.txt
zhome:~/linux_tutorial$ grep "hello world" *.pl
hello_world.pl:print "hello world.\n";
zhome:~/linux_tutorial$
```

Command: diff

- ▶ To compare to files for differences use “diff”
 - ▶ Try: `diff /dev/null hello.txt`
 - ▶ `/dev/null` is a special address -- it is always empty, and anything moved there is deleted

Working with Daemons

- ▶ In the Linux world, we use the term **daemon** to refer to a process that runs in the background.
 - ▶ You may also see the term **service** used to describe these types of background processes.)

Starting, stopping, and restarting background services

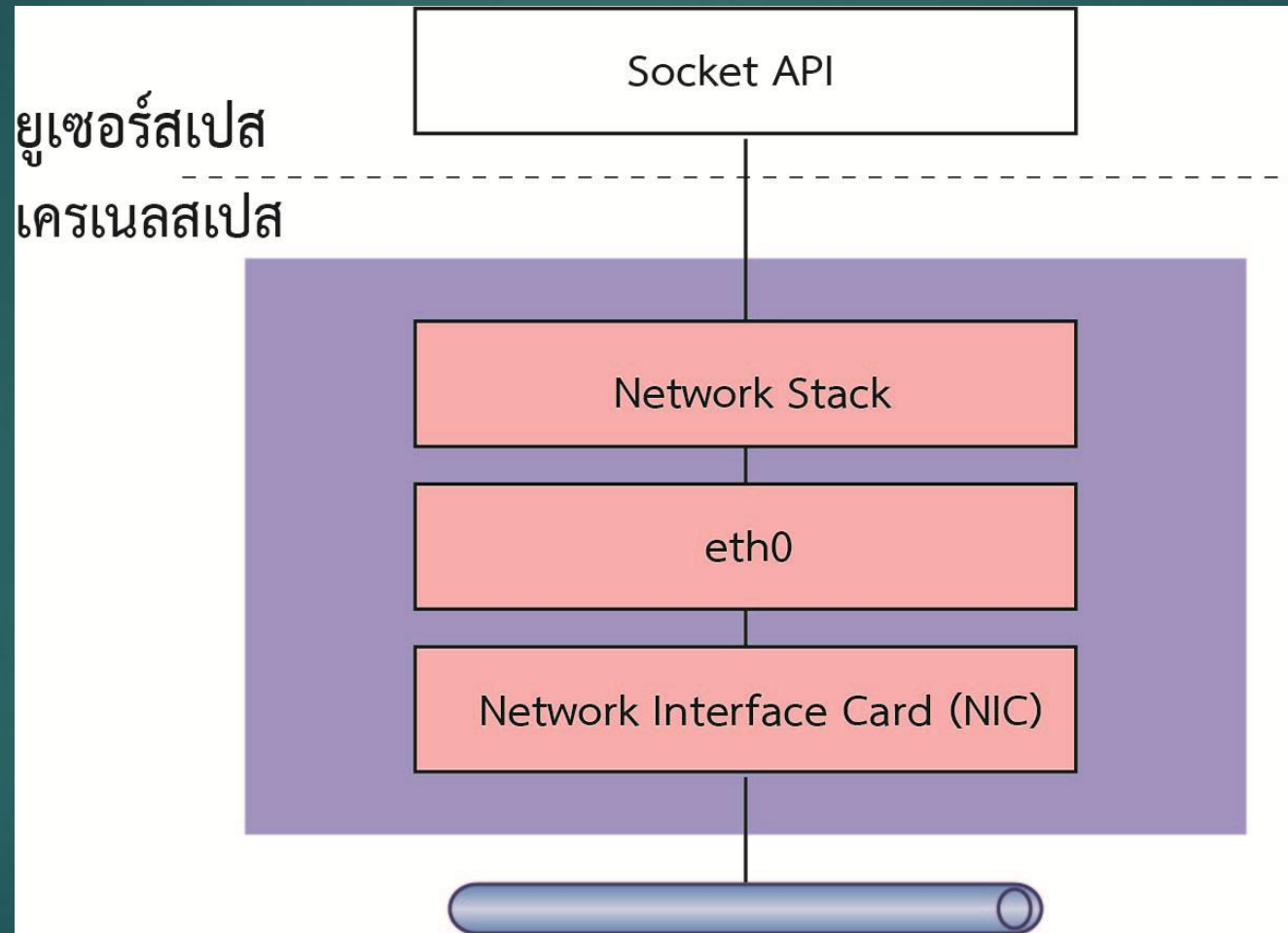
- ▶ **systemctl start** *service-name*
- ▶ **systemctl stop** *service-name*
- ▶ **systemctl restart** *service-name*
- ▶ **systemctl status** *service-name*
- ▶ *Note show config file*
- ▶ **systemctl cat** *service-name*

```
root@chatchaiasusPC:~# systemctl status mysql.service
● mysql.service - MySQL Community Server
   Loaded: loaded (/usr/lib/systemd/system/mysql.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-11-27 11:14:08 +07; 4h 34min ago
     Main PID: 403 (mysqld)
    Status: "Server is operational"
       Tasks: 37 (limit: 23763)
      Memory: 387.4M (peak: 444.4M)
         CPU: 1min 52.069s
        CGroup: /system.slice/mysql.service
                └─403 /usr/sbin/mysqld
```

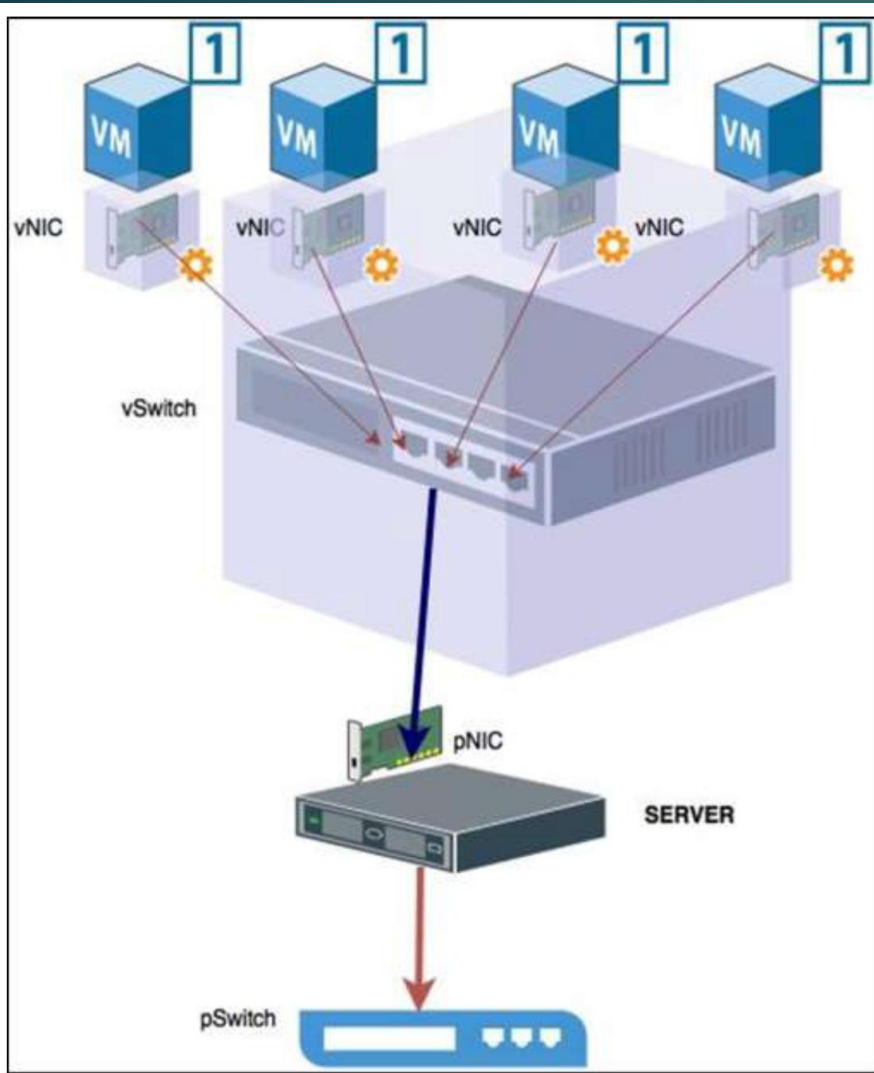
```
root@chatchaiasusPC:~# systemctl cat mysql.service
# /usr/lib/systemd/system/mysql.service
# MySQL systemd service file

[Unit]
Description=MySQL Community Server
After=network.target
```

Linux Interface



VM machine



ประเภทของ Virtual Interface

https://developers.redhat.com/blog/2018/10/22/introduction-to-linux-interfaces-for-virtual-networking#bonded_interface

Linux Basic Command

► Listing interfaces

```
[vagrant@centos ~]$ ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    mode DEFAULT qlen 1000
    link/ether 00:0c:29:d7:28:17 brd ff:ff:ff:ff:ff:ff
3: ens33: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc pfifo fast state UP
```

- UP: Indicates that the interface is enabled (activated)
- LOWER_UP: Indicates that interface link is up.
- NO_CARRIER (not shown): The interface is enabled, but there is no link.
- BROADCAST indicates that interface is configured to handle broadcast packets, which is required for obtaining IP address via DHCP.

ping

- Standard network tool to test connectivity and to measure the round trip time between two nodes
- Implements ICMP Echo Request – Response protocol (RFC 792)

```
$ ping -c 3 localhost
PING localhost(localhost (:::1)) 56 data bytes
64 bytes from localhost (:::1): icmp_seq=1 ttl=64 time=0.027 ms
64 bytes from localhost (:::1): icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from localhost (:::1): icmp_seq=3 ttl=64 time=0.088 ms

--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2044ms
rtt min/avg/max/mdev = 0.027/0.058/0.088/0.024 ms
```

Example: Ping the localhost

iproute2

iproute2 is a suite of tools used to display and manipulate network devices, interfaces, and routing information in a Linux system.

We limit the focus to:

- `ip addr`
- `ip link`
- `ip route`

ip addr

- Tool to perform operations on the Layer 3 (networking) of the OSI stack
- Helpful in both monitoring and manipulation

```
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:c2:be:11 brd ff:ff:ff:ff:ff:ff
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
valid_lft 64793sec preferred_lft 64793sec
inet6 fe80::a00:27ff:fec2:be11/64 scope link
valid_lft forever preferred_lft forever
```

Example: Display the addresses of all interfaces

ip addr

```
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:c2:be:11 brd ff:ff:ff:ff:ff:ff
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
valid_lft 64793sec preferred_lft 64793sec
inet6 fe80::a00:27ff:fec2:be11/64 scope link
valid_lft forever preferred_lft forever
```

Example: Display the addresses of all a specific interface

ip addr

```
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:c2:be:11 brd ff:ff:ff:ff:ff:ff
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
valid_lft 64793sec preferred_lft 64793sec
inet6 fe80::a00:27ff:fec2:be11/64 scope link
valid_lft forever preferred_lft forever
```

Example: Display the addresses of all a specific interface

ip addr

```
$ sudo ip addr add dev eth0 10.0.8.1/24
$ ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:c2:be:11 brd ff:ff:ff:ff:ff:ff
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
valid_lft 64072sec preferred_lft 64072sec
inet 10.0.8.1/24 scope global eth0
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fec2:be11/64 scope link
valid_lft forever preferred_lft forever
```

Example: Add an address to an interface

ip addr

```
$ sudo ip addr del dev eth0 10.0.8.1/24
$ ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:c2:be:11 brd ff:ff:ff:ff:ff:ff
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
valid_lft 63953sec preferred_lft 63953sec
inet6 fe80::a00:27ff:fec2:be11/64 scope link
valid_lft forever preferred_lft forever
```

Example: Delete an address from an interface

► Show Ip address

```
vagrant@trusty:~$ ip addr list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    group default qlen 1000
    link/ether 00:0c:29:33:99:f6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.70.205/24 brd 192.168.70.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe33:99f6/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    group default qlen 1000
    link/ether 00:0c:29:33:99:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.11/24 brd 192.168.100.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe33:9900/64 scope link
        valid_lft forever preferred_lft forever
vagrant@trusty:~$
```

ip link

- Operates on the layer 2 (link layer) of the OSI stack
- Commonly used to manage the interface properties like adding virtual links, VLAN tags, bridging, etc.

```
$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default
qlen 1000
link/ether 08:00:27:c2:be:11 brd ff:ff:ff:ff:ff:ff
```

Example: Show the link properties of all interfaces

► Enabling/disabling an interface

```
[vagrant@centos ~]$ ip link set ens33 down
[vagrant@centos ~]$ ip link list ens33
3: ens33: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT
    qlen 1000
    link/ether 00:0c:29:d7:28:21 brd ff:ff:ff:ff:ff:ff
[vagrant@centos ~]$
```

► Set interface up

```
[vagrant@centos ~]$ ip link set ens33 up
[vagrant@centos ~]$ ip link list ens33
3: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    mode DEFAULT qlen 1000
    link/ether 00:0c:29:d7:28:21 brd ff:ff:ff:ff:ff:ff
[vagrant@centos ~]$
```

- ▶ Setting the MTU of an interface

- ▶ `ip link set mtu MTU interface`

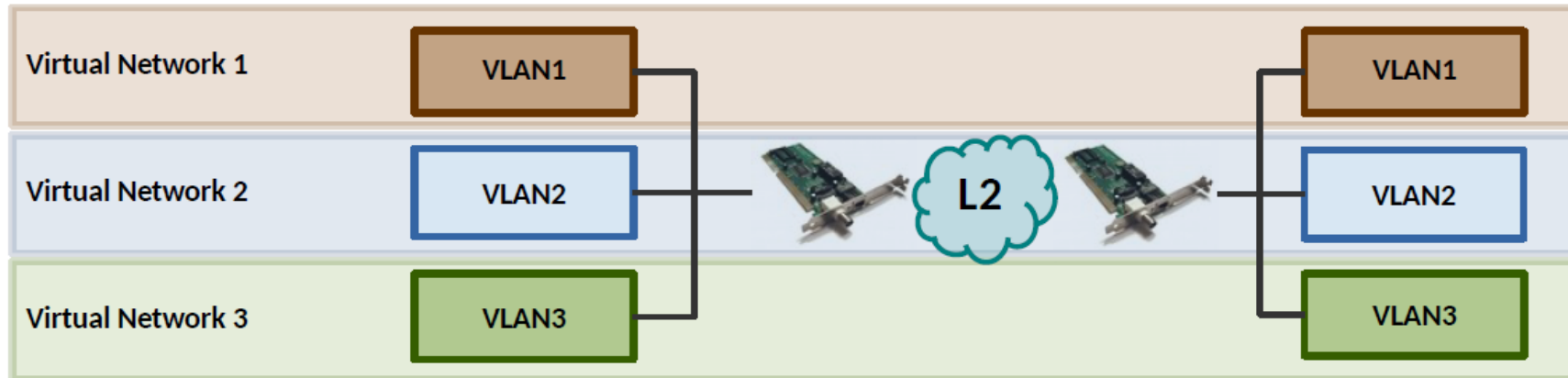
```
[vagrant@centos ~]$ ip link set mtu 9000 ens33  
[vagrant@centos ~]$
```

- ▶ Assigning an IP address to an interface

- ▶ `ip addr add address dev interface`

```
vagrant@jessie:~$ ip addr add 172.31.254.100/24 dev eth1  
vagrant@jessie:~$
```

VLAN



Packet Headers:



```
$ ip link add link em1 vlan1 type vlan id 1
$ ip link set vlan1 up
$ ip link show vlan1
15: vlan1@em1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP [...]
    link/ether 10:c3:7b:95:21:da brd ff:ff:ff:ff:ff:ff
```

Creating, configuring, and deleting VLAN interfaces

```
ip link add link ens3 ens3.150 type vlan id 150
```

- ▶ The **parent-device** is the physical adapter with which the logical VLAN interface is associated. This would be something like eth1 or ens33.
- ▶ The **vlan-device** is the name to be given to the logical VLAN interface; the common convention is to use the name of the parent device, a dot (period), and then the VLAN ID. For a VLAN interface associated with eth1 and using VLAN ID 100, the name would be eth1.100.
- ▶ **vlan-id** is exactly that—the 802.1Q VLAN ID value assigned to this logical interface

Use cases for VLAN interfaces

- ▶ For communicating on multiple VLANs at the same time *and* minimizing the number of switch ports and physical interfaces required
- ▶ Example one VLAN for Web Servers and another VLAN to database servers, using a single physical interface with two logical VLAN interfaces is an ideal solution

ip link

```
$ sudo ip link add link eth0 name eth0.100 type vlan id 100
$ sudo ip link set dev eth0.100 mtu 1450 up
$ sudo ip link show dev eth0.100
3: eth0.100@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP mode DEFAULT group
default qlen 1000
link/ether 08:00:27:c2:be:11 brd ff:ff:ff:ff:ff:ff
```

Example: Add an IEEE802.1q VLAN tagged link onto an interface

ip route

- Used to manage the network routes across the different interfaces managed by the OS

```
$ ip r
default via 10.0.2.2 dev eth0 proto dhcp src 10.0.2.15 metric 100
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15
10.0.2.2 dev eth0 proto dhcp scope link src 10.0.2.15 metric 100
```

Example: Show all active IPv4 routes in the default namespace

Namespaces

- ▶ Provides a wrapper around a global system resource of the kernel and makes the resource appear to the process within the namespace as if they have an isolated instance.
 - ▶ **pid namespace**: Used for process isolation (**PID—Process ID**)
 - ▶ **net namespace**: Used for managing network interfaces (**NET—Networking**)
 - ▶ **ipc namespace**: Used for managing access to IPC resources (**IPC—Inter Process Communication**)
 - ▶ **mnt namespace**: Used for managing mount points (**MNT—Mount**)
 - ▶ **uts namespace**: Used for isolating kernel and version identifiers (**UTS—Unix Time sharing System**)

“lsns”

list information about currently active Linux namespaces

```
ck_bo@chatchaiasusPC:~$ lsns
```

	NS	TYPE	NPROCS	PID	USER	COMMAND
	4026531834	time	33	1483	ck_bo	/bin/bash /usr/local/bin/nginx-entrypoint.sh
	4026531835	cgroup	4	2171	ck_bo	-bash
	4026531837	user	33	1483	ck_bo	/bin/bash /usr/local/bin/nginx-entrypoint.sh
	4026531840	net	4	2171	ck_bo	-bash
	4026532206	ipc	4	2171	ck_bo	-bash
	4026532217	mnt	4	2171	ck_bo	-bash
	4026532218	uts	4	2171	ck_bo	-bash
	4026532219	pid	4	2171	ck_bo	-bash
	4026532365	mnt	22	1483	ck_bo	/bin/bash /usr/local/bin/nginx-entrypoint.sh
	4026532366	uts	22	1483	ck_bo	/bin/bash /usr/local/bin/nginx-entrypoint.sh
	4026532367	ipc	22	1483	ck_bo	/bin/bash /usr/local/bin/nginx-entrypoint.sh
	4026532368	pid	22	1483	ck_bo	/bin/bash /usr/local/bin/nginx-entrypoint.sh
	4026532369	cgroup	22	1483	ck_bo	/bin/bash /usr/local/bin/nginx-entrypoint.sh
	4026532370	net	22	1483	ck_bo	/bin/bash /usr/local/bin/nginx-entrypoint.sh

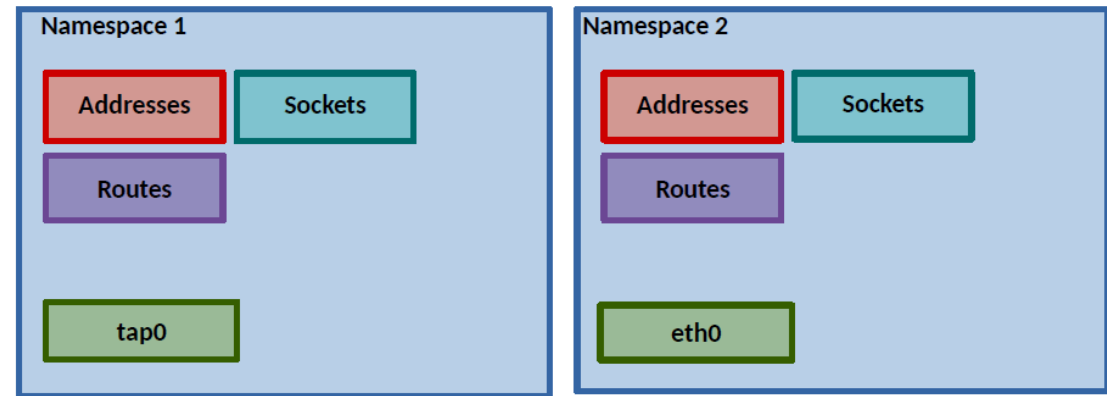
Net namespace

- ▶ provides isolation of the system resources associated with networking.
- ▶ has its own network devices, IP addresses, IP routing tables, /proc/net directory, port numbers, and so on.
- ▶ Every Linux come with default network namespace
- ▶ Use case of network namespace
 - ▶ Config routing on per-process basis
 - ▶ Combine overlay network and/or NAT

Creating and Remove Network namespace

- ▶ Create a network namespace
 - ▶ **ip netns add *namespace-name***
- ▶ Example :add blue namespace
 - ▶ ip netns add blue
- ▶ Show (list) namespace create
 - ▶ **ip netns list**
- ▶ Deleting network namespace
 - ▶ **ip netns del *namespace-name***

Linux maintains resources and data structures per namespace



Placing interface in a Network Namespace

- ▶ by default, a newly created network namespace contains no network interfaces
- ▶ To place an interface into a namespace, use the `ip link` command (obviously this command assumes that the blue namespace has already been created):
 - ▶ **`ip link set`** *interface-name* **`netns`** *namespace-name*

```
vagrant@jessie:~$ ip link set eth1 netns blue
vagrant@jessie:~$
```


Executing Commands in a Network Namespace

- ▶ The general syntax
 - ▶ **ip netns exec** *namespace-name* *command*.
- ▶ to see interface, inside a particular namespace

```
vagrant@jessie:~$ ip netns exec blue ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
   default qlen 1000
   link/ether 00:0c:29:7d:38:9d brd ff:ff:ff:ff:ff:ff
vagrant@jessie:~$
```

- ▶ To enable this interface (currently down) :

```
vagrant@jessie:~$ ip netns exec blue ip link set eth1 up
vagrant@jessie:~$ ip netns exec blue ip link list eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
   mode DEFAULT group default qlen 1000
   link/ether 00:0c:29:7d:38:9d brd ff:ff:ff:ff:ff:ff
vagrant@jessie:~$
```

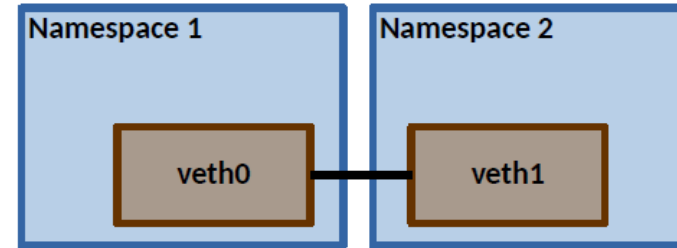
Executing Commands in a Network Namespace (2)

- ▶ assign an IP address and check the namespace's routing table

```
vagrant@jessie:~$ ip netns exec blue ip addr add 192.168.100.10/24 dev eth1
vagrant@jessie:~$ ip netns exec blue ip route list
192.168.100.0/24 dev eth1 proto kernel scope link src 192.168.100.11
vagrant@jessie:~$
```


Virtual Ethernet Cable

- Bidirectional FIFO
- Often used to cross namespaces



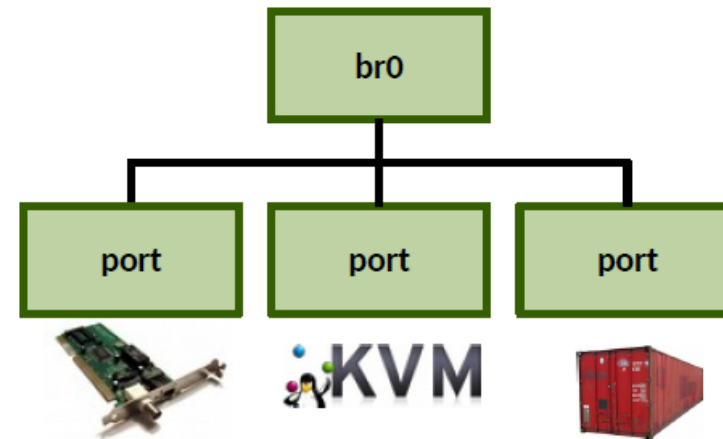
```
$ ip link add veth1 type veth peer name veth2  
$ ip link set veth1 netns ns1  
$ ip link set veth2 netns ns2
```

Connecting Network Namespaces with veth Pairs

- ▶ connect this new namespace with the default namespace
- ▶ *Virtual Ethernet pairs* (more commonly known as *veth pairs*) are
 - ▶ a special kind of logical interface supported by the Linux kernel.
 - ▶ traffic entering one interface in the pair comes out the other interface in the pair.
 - ▶ a veth pair can be assigned to a non-default network namespace—thus enabling users to *connect* network namespaces to each other

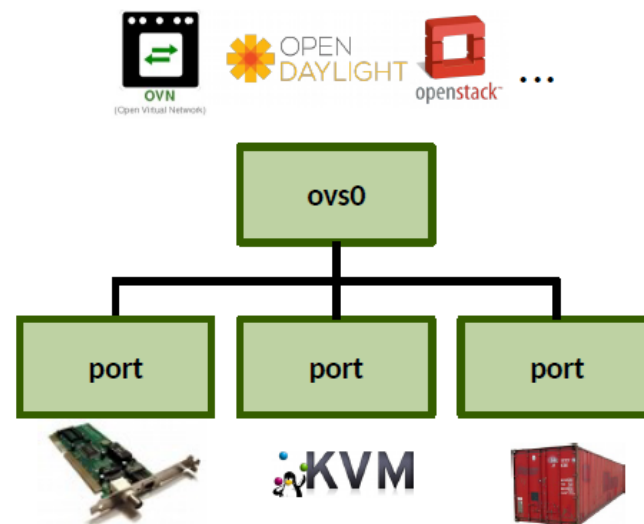
Bridge: Virtual Switch

- **Flooding:** Clone packets and send to all ports.
- **Learning:** Learn who's behind which port to avoid flooding
- **STP:** Detect wiring loops and disable ports
- **Native VLAN integration**
- **Offload:** Program HW based on FDB table



Open vSwitch

- Fully programmable L2-L4 virtual switch with APIs: OpenFlow and OVSDB
- Split into a user and kernel component
- Multiple control plane integrations:
 - OVN, ODL, Neutron, CNI, Docker, ...



```
$ ovs-vsctl add-br ovs0
$ ovs-vsctl add-port ovs0 em1
$ ovs-ofctl add-flow ovs0 in_port=1,actions=drop
$ ovs-vsctl show
a425a102-c317-4743-b0ba-79d59ff04a74
    Bridge "ovs0"
        Port "em1"
            Interface "em1"
[...]
```

Traffic generation

iperf

- A server-client program to generate arbitrary network to test the network performance
- Capable of both TCP and UDP
- Works on both IPv4 and IPv6

```
$ iperf3 -s
-----
Server listening on 5201
-----
Accepted connection from 127.0.0.1, port 50816
[ 5] local 127.0.0.1 port 5201 connected to 127.0.0.1 port 50818
[ ID] Interval      Transfer    Bandwidth
[ 5]  0.00-1.00  sec  2.87 GBytes  24.6 Gbits/sec
[ 5]  1.00-2.00  sec  2.80 GBytes  24.0 Gbits/sec
[ 5]  2.00-3.00  sec  3.08 GBytes  26.5 Gbits/sec
[ 5]  3.00-3.04  sec  89.8 MBytes  18.7 Gbits/sec
-----
[ ID] Interval      Transfer    Bandwidth
[ 5]  0.00-3.04  sec  0.00 Bytes  0.00 bits/sec      sender
[ 5]  0.00-3.04  sec  8.83 GBytes  25.0 Gbits/sec      receiver
```

Server

```
$ iperf3 -c 127.0.0.1 -t 3
Connecting to host 127.0.0.1, port 5201
[ 4] local 127.0.0.1 port 50818 connected to 127.0.0.1 port 5201
[ ID] Interval      Transfer    Bandwidth    Retr  Cwnd
[ 4]  0.00-1.00  sec  2.95 GBytes  25.3 Gbits/sec    0   3.18 MBytes
[ 4]  1.00-2.00  sec  2.79 GBytes  24.0 Gbits/sec    0   3.18 MBytes
[ 4]  2.00-3.00  sec  3.09 GBytes  26.5 Gbits/sec    0   3.18 MBytes
-----
[ ID] Interval      Transfer    Bandwidth    Retr
[ 4]  0.00-3.00  sec  8.83 GBytes  25.3 Gbits/sec    0  sender
[ 4]  0.00-3.00  sec  8.83 GBytes  25.3 Gbits/sec    receiver

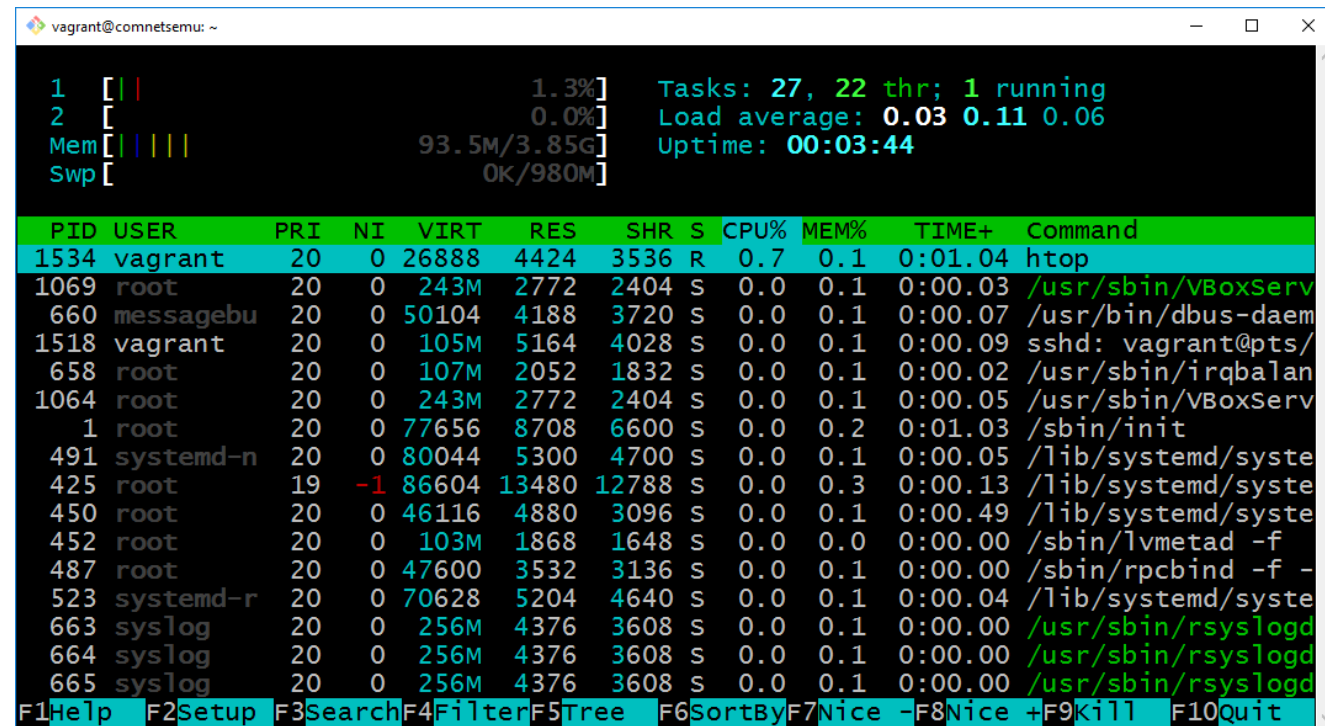
iperf Done.
```

Client

Process Monitoring

htop

- An interactive tool to monitor process performance
- Offers detailed information like individual CPU load, memory load and CPU time used and allows corresponding sorting and filtering



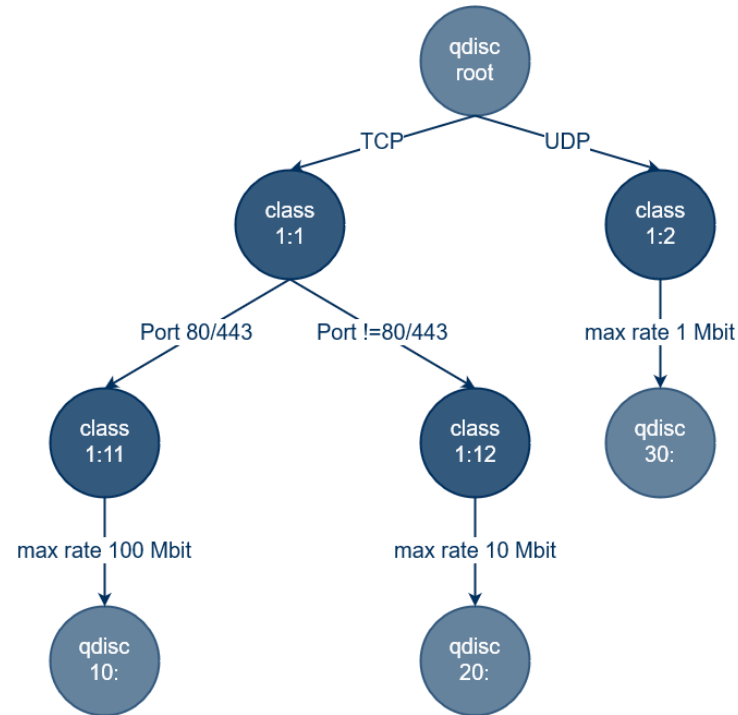
```
vagrant@comnetsemu: ~  
  
 1 [|||] 1.3% Tasks: 27, 22 thr; 1 running  
 2 [ ] 0.0% Load average: 0.03 0.11 0.06  
Mem [||||] 93.5M/3.85G Uptime: 00:03:44  
Swp [ ] 0K/980M  
  
 PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
1534 vagrant 20 0 26888 4424 3536 R 0.7 0.1 0:01.04 htop  
1069 root 20 0 243M 2772 2404 S 0.0 0.1 0:00.03 /usr/sbin/VBoxServ  
660 messagebu 20 0 50104 4188 3720 S 0.0 0.1 0:00.07 /usr/bin/dbus-daem  
1518 vagrant 20 0 105M 5164 4028 S 0.0 0.1 0:00.09 sshd: vagrant@pts/  
658 root 20 0 107M 2052 1832 S 0.0 0.1 0:00.02 /usr/sbin/irqbalan  
1064 root 20 0 243M 2772 2404 S 0.0 0.1 0:00.05 /usr/sbin/VBoxServ  
1 root 20 0 77656 8708 6600 S 0.0 0.2 0:01.03 /sbin/init  
491 systemd-n 20 0 80044 5300 4700 S 0.0 0.1 0:00.05 /lib/systemd/syste  
425 root 19 -1 86604 13480 12788 S 0.0 0.3 0:00.13 /lib/systemd/syste  
450 root 20 0 46116 4880 3096 S 0.0 0.1 0:00.49 /lib/systemd/syste  
452 root 20 0 103M 1868 1648 S 0.0 0.0 0:00.00 /sbin/lvmetag -f  
487 root 20 0 47600 3532 3136 S 0.0 0.1 0:00.00 /sbin/rpcbind -f -  
523 systemd-r 20 0 70628 5204 4640 S 0.0 0.1 0:00.04 /lib/systemd/syste  
663 syslog 20 0 256M 4376 3608 S 0.0 0.1 0:00.00 /usr/sbin/rsyslogd  
664 syslog 20 0 256M 4376 3608 S 0.0 0.1 0:00.00 /usr/sbin/rsyslogd  
665 syslog 20 0 256M 4376 3608 S 0.0 0.1 0:00.00 /usr/sbin/rsyslogd  
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice + F9Kill F10Quit
```

Screenshot of an example htop session

Network traffic manipulation

tc

- Allows manipulation of the network traffic patterns in terms of latency, throughput and losses
- Elemental in all Linux network emulators including Mininet



An example hierarchical tree structure of tc

tc

```
$ ping 8.8.8.8 -c 2
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=50 time=15.0 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=50 time=15.5 ms

$ sudo tc qdisc add dev eth0 root netem delay 200ms

$ ping 8.8.8.8 -c 2
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=50 time=217 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=50 time=216 ms
```

Example: RTT measurement before and after adding delay using tc

tc

```
$ sudo tc qdisc change dev eth0 parent root netem delay 2000ms 1500ms

$ ping 8.8.8.8 -c 5
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=50 time=1730 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=50 time=765 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=50 time=1649 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=50 time=1962 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=50 time=3051 ms
```

Example: Adding jitter to the interface

Traffic Monitoring

tcpdump

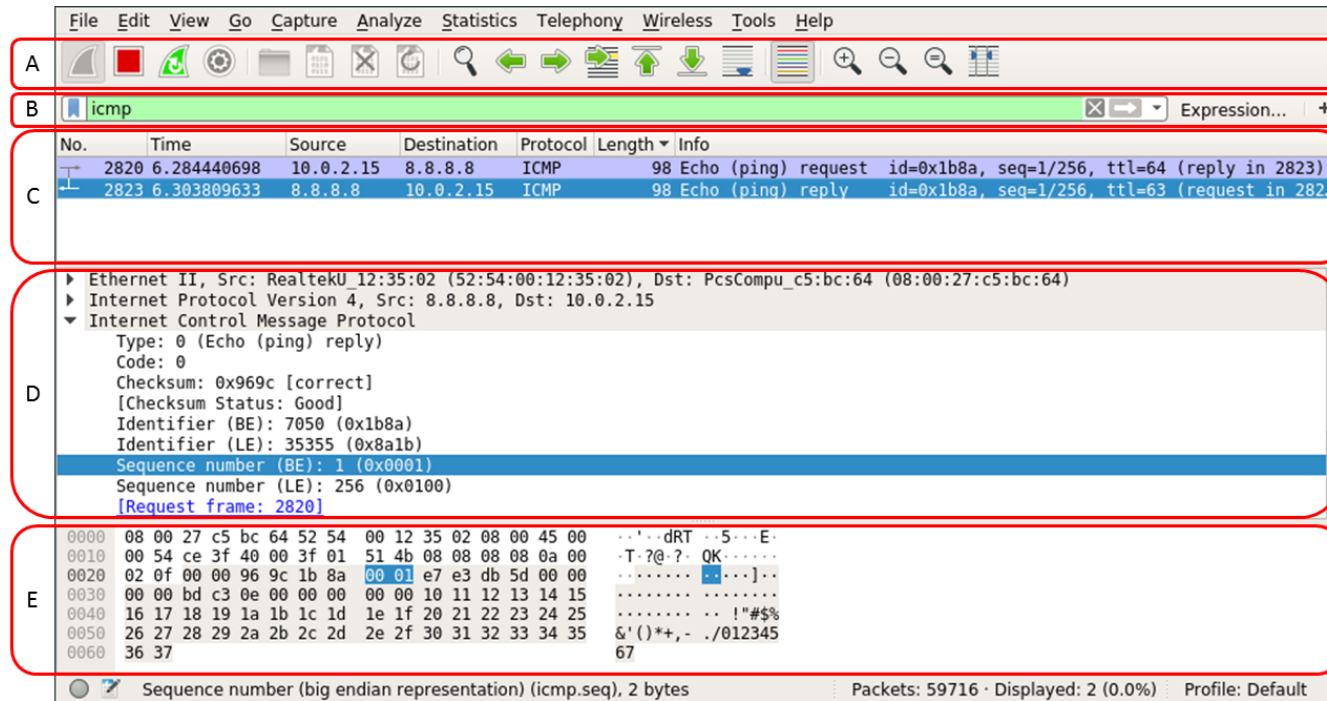
- A command line tool that provides insights into the network traffic on a packet level
- The packets may be filtered using a vast variety of options including type of transport (TCP /UDP), port number, and IP address

```
$ tcpdump -i enp0s31f6 -n -e icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s31f6, link-type EN10MB (Ethernet), capture size 262144 bytes
06:44:34.409654 4c:de:ad:ff:be:ef > 28:de:ad:0b:ee:fc, ethertype 802.1Q (0x8100), length 102: vlan 717, p
    0, ethertype IPv4, 172.31.56.3 > 172.31.56.95: ICMP echo request, id 31996, seq 2, length 64
06:44:34.411150 28:de:ad:0b:ee:fc > 4c:de:ad:ff:be:ef, ethertype 802.1Q (0x8100), length 102: vlan 717, p
    0, ethertype IPv4, 172.31.56.95 > 172.31.56.3: ICMP echo reply, id 31996, seq 2, length 64
```

Example: TCP dump of an ongoing ping

Wireshark

- Similar to tcpdump, but offers a user-friendly GUI
- Provides deeper network inspections using graphing tools and dynamic plots



- A – Main toolbar
- B – Filter toolbar
- C – Packet list pane
- D – Packet filter pane

GUI of Wireshark

Wireshark

The screenshot displays the Wireshark interface with a capture filter of 'icmp or arp'. The packet list shows several ICMP Echo (ping) requests and replies. Packet 6214 is selected, showing details of an ICMP Echo (ping) reply from 8.8.8.8 to 10.0.2.15.

No.	Time	Source	Destination	Protocol	Length	Info
5987	1.105067056	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x1d80, seq=1/256, ttl=64
5990	1.123315179	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x1d80, seq=1/256, ttl=63
6213	2.106071968	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x1d80, seq=2/512, ttl=64
6214	2.124620221	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x1d80, seq=2/512, ttl=63
6880	3.108240724	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request id=0x1d80, seq=3/768, ttl=64
6881	3.126446354	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply id=0x1d80, seq=3/768, ttl=63

Frame 6214: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: RealtekU 12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_c5:bc:64 (08:00:27:c5:bc:64)
Internet Protocol Version 4, Src: 8.8.8.8, Dst: 10.0.2.15
Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0xa14b [correct]
[Checksum Status: Good]
Identifier (BE): 7552 (0x1d80)

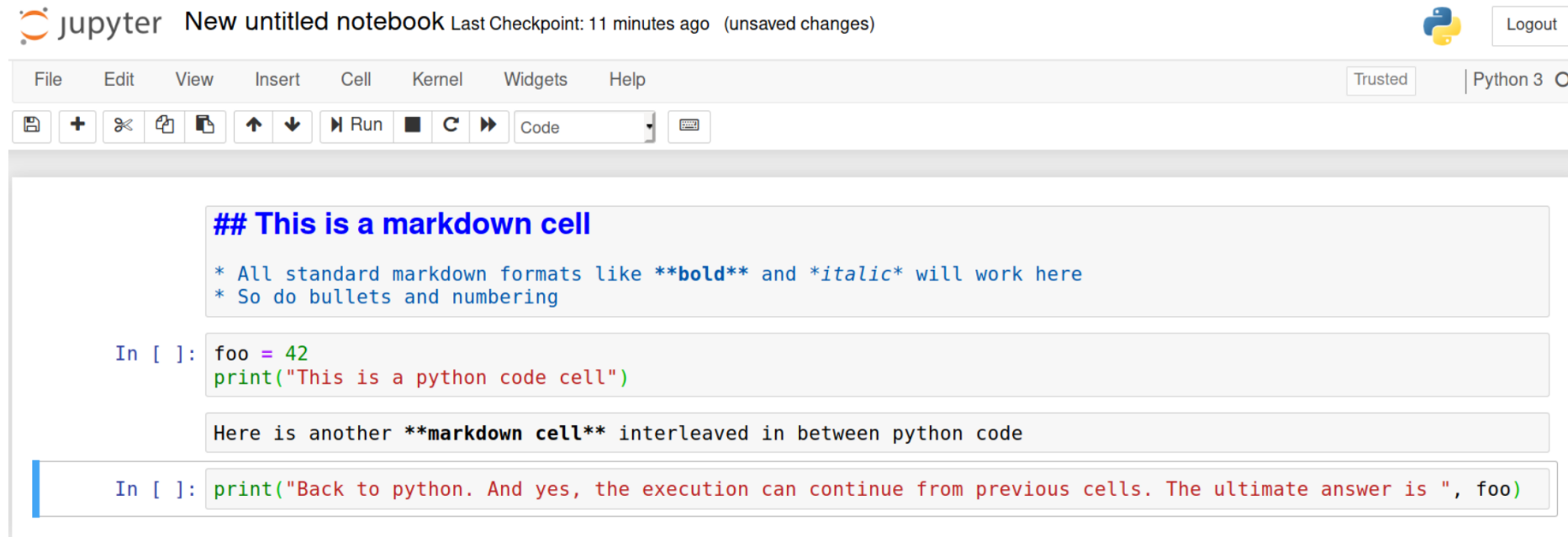
0000 08 00 27 c5 bc 64 52 54 00 12 35 02 08 00 45 00 ..'.dRT --5...E.
0010 00 54 18 50 40 00 3f 01 07 3b 08 08 08 08 0a 00 .T.P@.?.-;.....
0020 02 0f 00 00 a1 4b 1d 80 00 02 cf ed db 5d 00 00 ..K.....]
0030 00 00 cf 13 08 00 00 00 00 00 10 11 12 13 14 15
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 !"#%
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &'()*+,-./012345
0060 36 37 67

Type (icmp.type), 1 byte Packets: 8002 · Displayed: 6 (0.1%) · Dropped: 0 (0.0%) Profile: Default

Example: Wireshark output of an ongoing ping

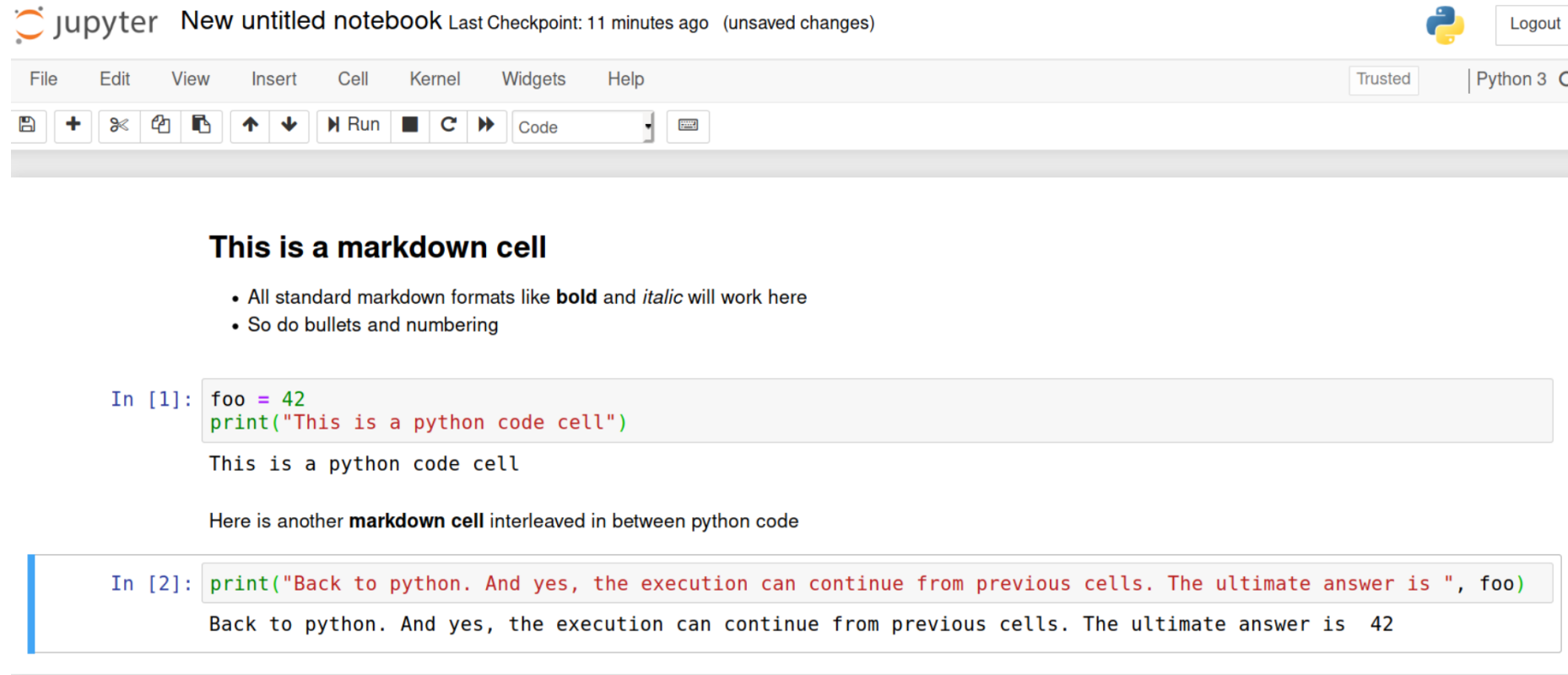
Jupyter

- An interactive browser based IDE to develop and test python scripts
- Allows active code and documentation to coexist in a single sequential structure, making it useful in storyboarding



Example python and markdown (unexecuted)

Jupyter



The screenshot shows a Jupyter Notebook interface. At the top, it says "jupyter New untitled notebook" and "Last Checkpoint: 11 minutes ago (unsaved changes)". There is a "Logout" button and a Python 3 logo. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3" buttons. Below the menu bar is a toolbar with icons for saving, adding, deleting, and running cells, as well as a dropdown menu currently set to "Code".

The notebook content consists of three cells:

- A markdown cell with the heading "This is a markdown cell" and two bullet points:
 - All standard markdown formats like **bold** and *italic* will work here
 - So do bullets and numbering
- A code cell (In [1]:) containing:

```
foo = 42
print("This is a python code cell")
```

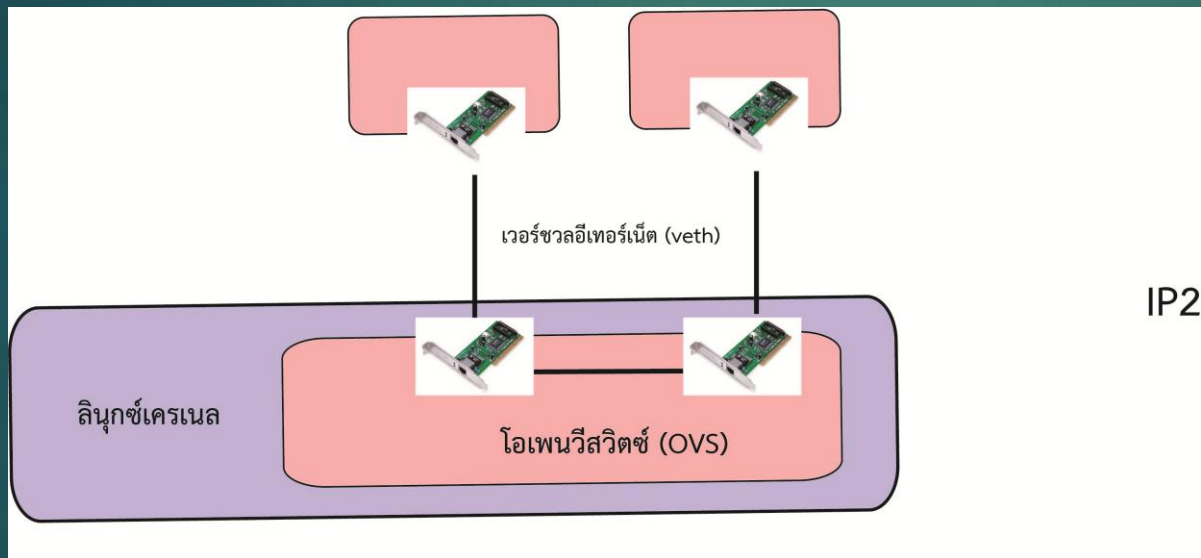
The output of this cell is "This is a python code cell". Below the output, there is a line of text: "Here is another **markdown cell** interleaved in between python code".
- A code cell (In [2]:) containing:

```
print("Back to python. And yes, the execution can continue from previous cells. The ultimate answer is ", foo)
```

The output of this cell is "Back to python. And yes, the execution can continue from previous cells. The ultimate answer is 42".

Example python and markdown (executed)

Lab วันนี้



จากรูปที่ 4.17 ในหนังสือ ทำตามขั้นตอนที่ปรากฏ โดยใช้ class A address ส่งท้าย ชม

Lab ส่งในชั่วโมง

- ▶ 1. จาก Text file list ของ ID นักศึกษา (random_ids) ให้เขียน Shell Script ที่สามารถ run ได้ เพื่อดูว่ามี ID ของตัวเองกี่ครั้ง พร้อมทั้งส่งผลที่ได้ออกไปยังอีก 1 file ที่มีเฉพาะ ID ตัวเอง พร้อมทั้งแสดงผลที่ได้ออกที่หน้าจอ
- ▶ 2. จากรูปที่ 4.17 ในหนังสือ ทำตามขั้นตอนที่ปรากฏ โดยใช้ class B address

