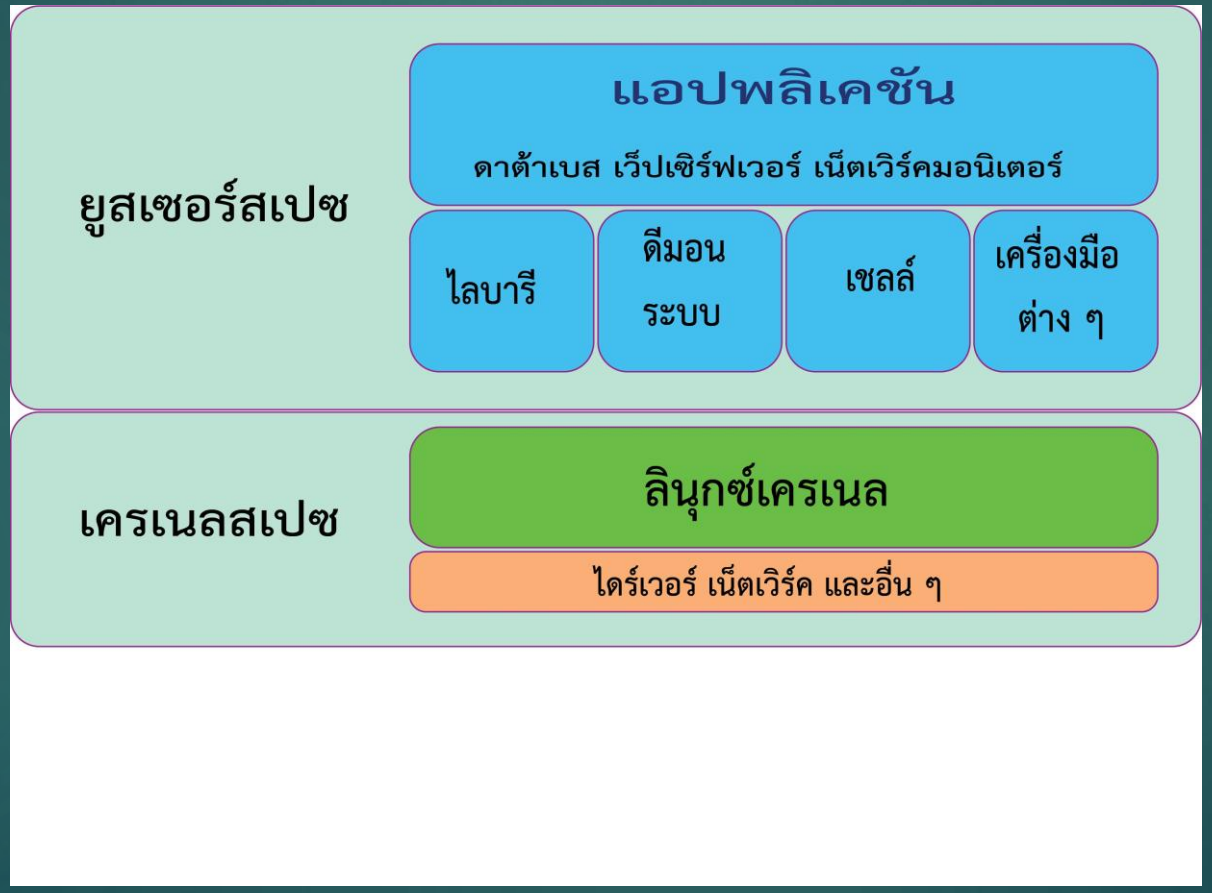
Basic Linux Command and networking

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
```

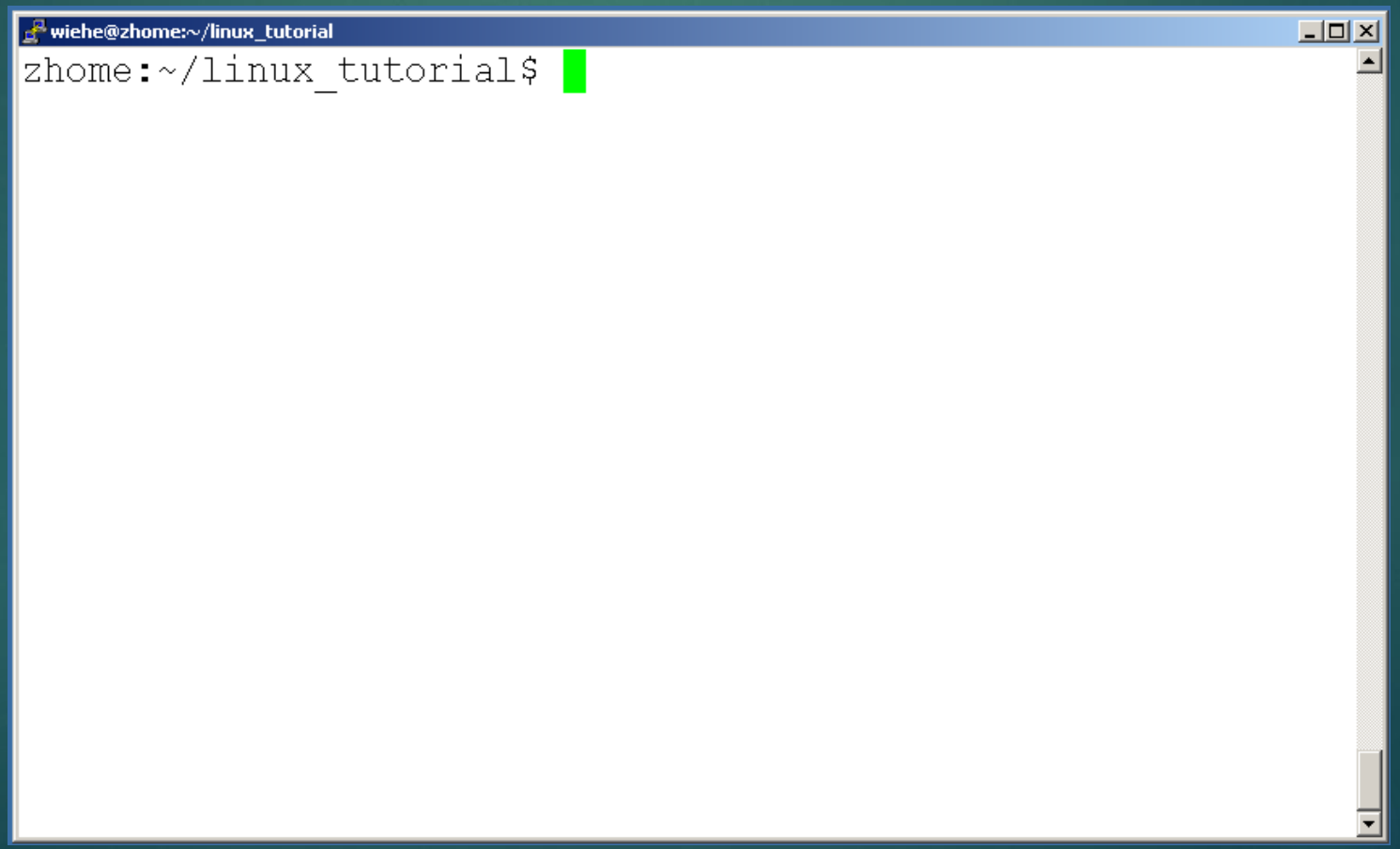
```
modern@ubuntu:~$ sudo apt-get -y upgrade
[
```

Substitute User

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

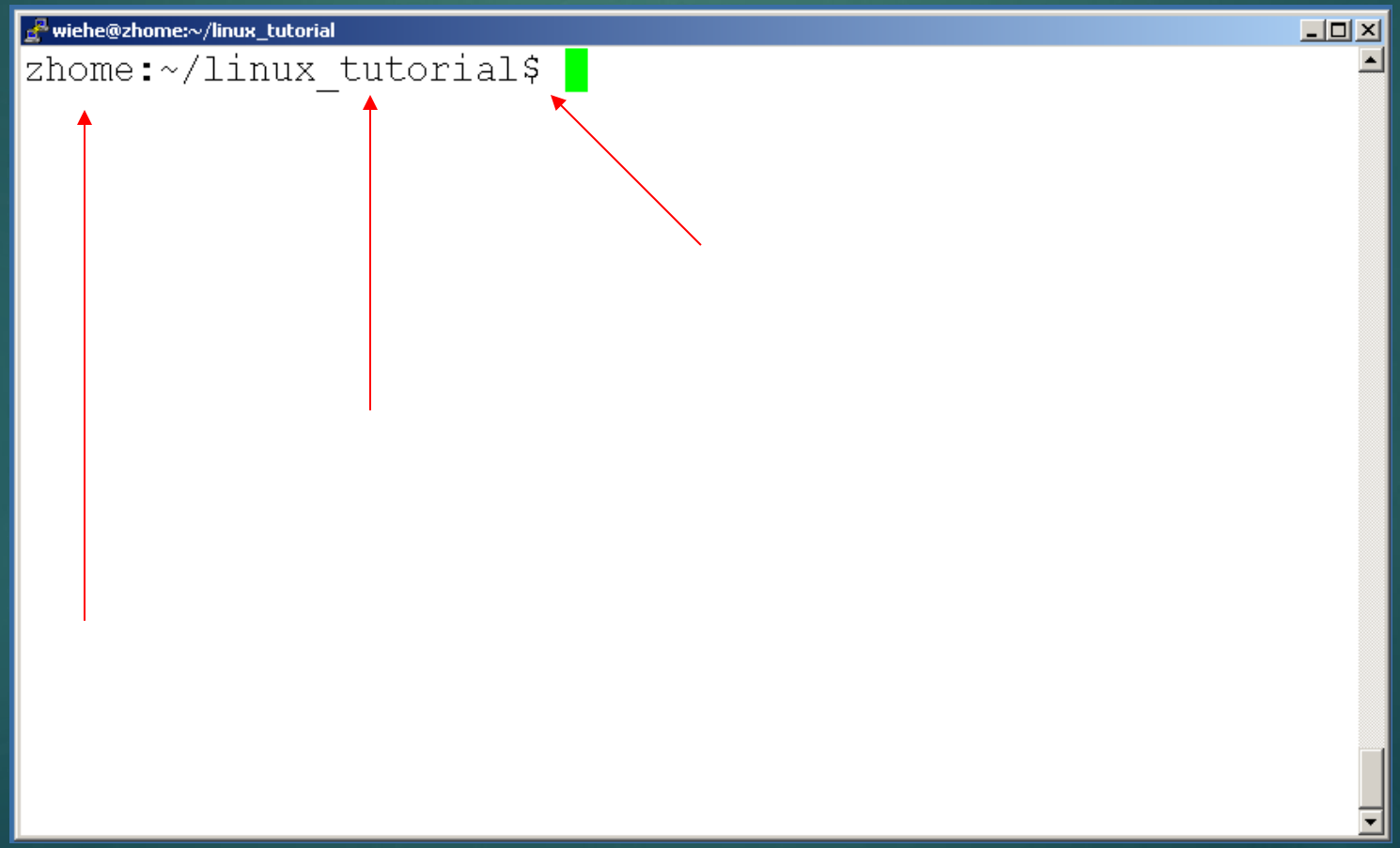
Connecting to a Unix/Linux system

- Open up a terminal:



Connecting to a Unix/Linux system

- Open up a terminal:



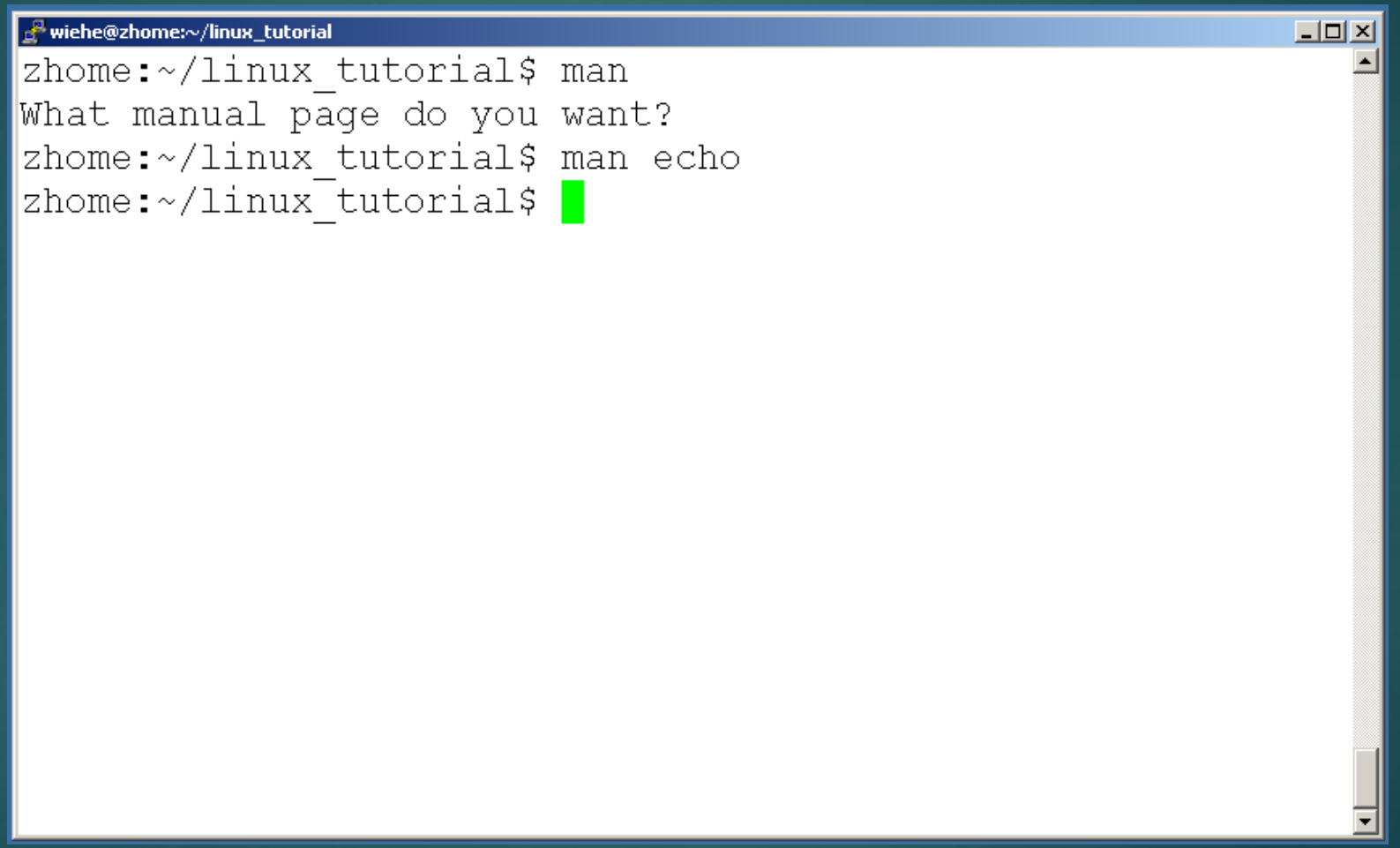
What exactly is a “shell”?

- 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!**

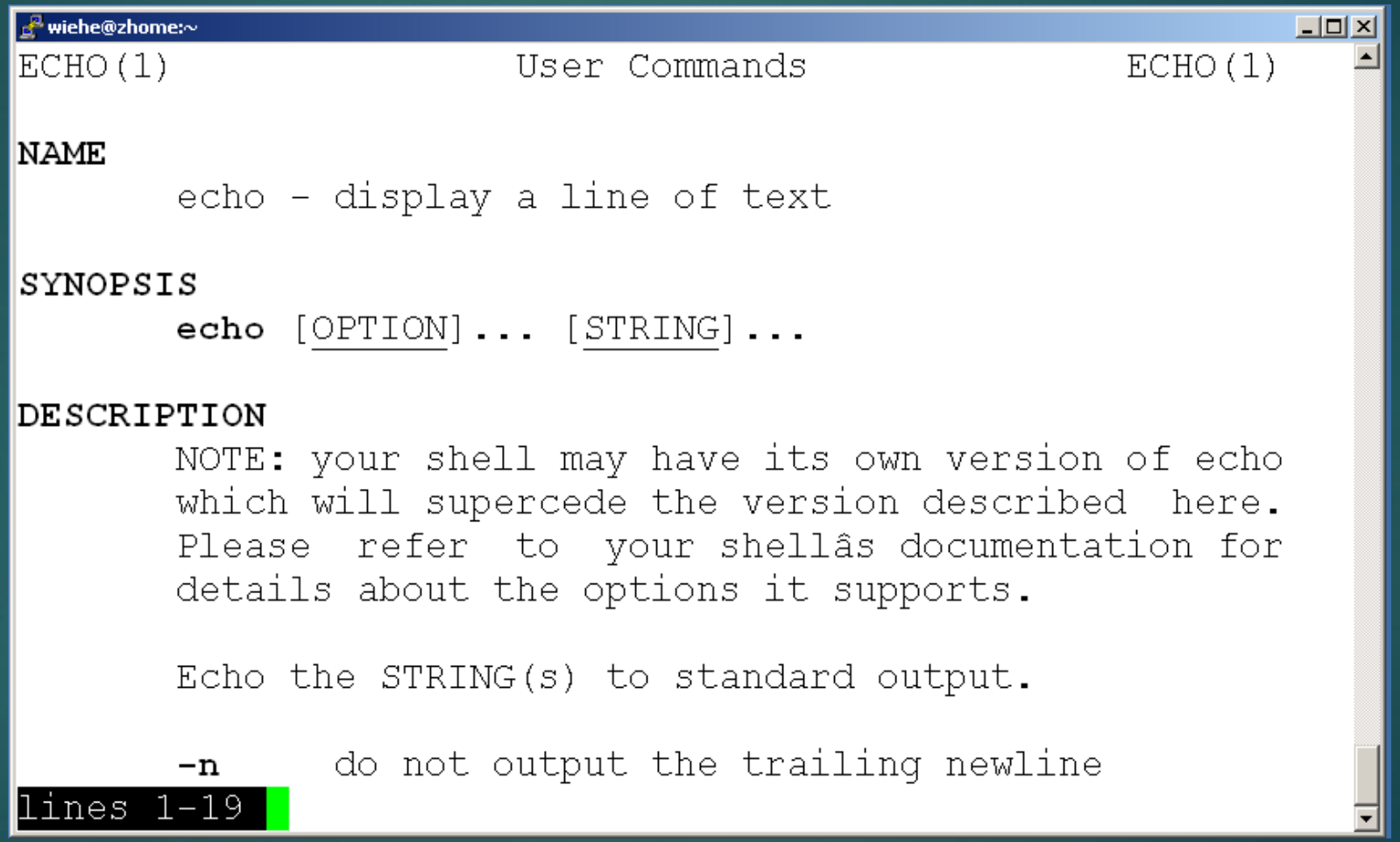
Help!

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

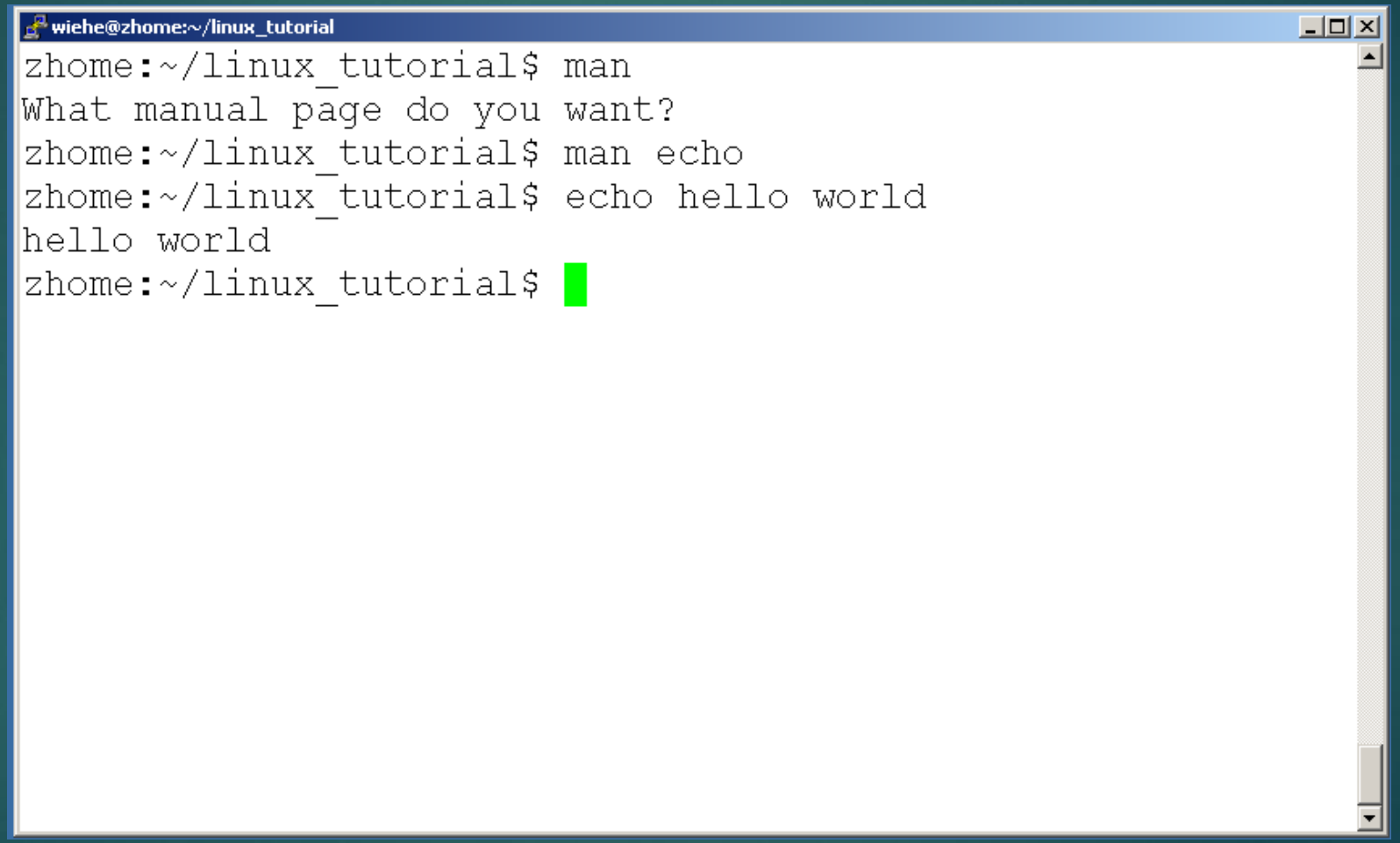
Help!


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

Help!

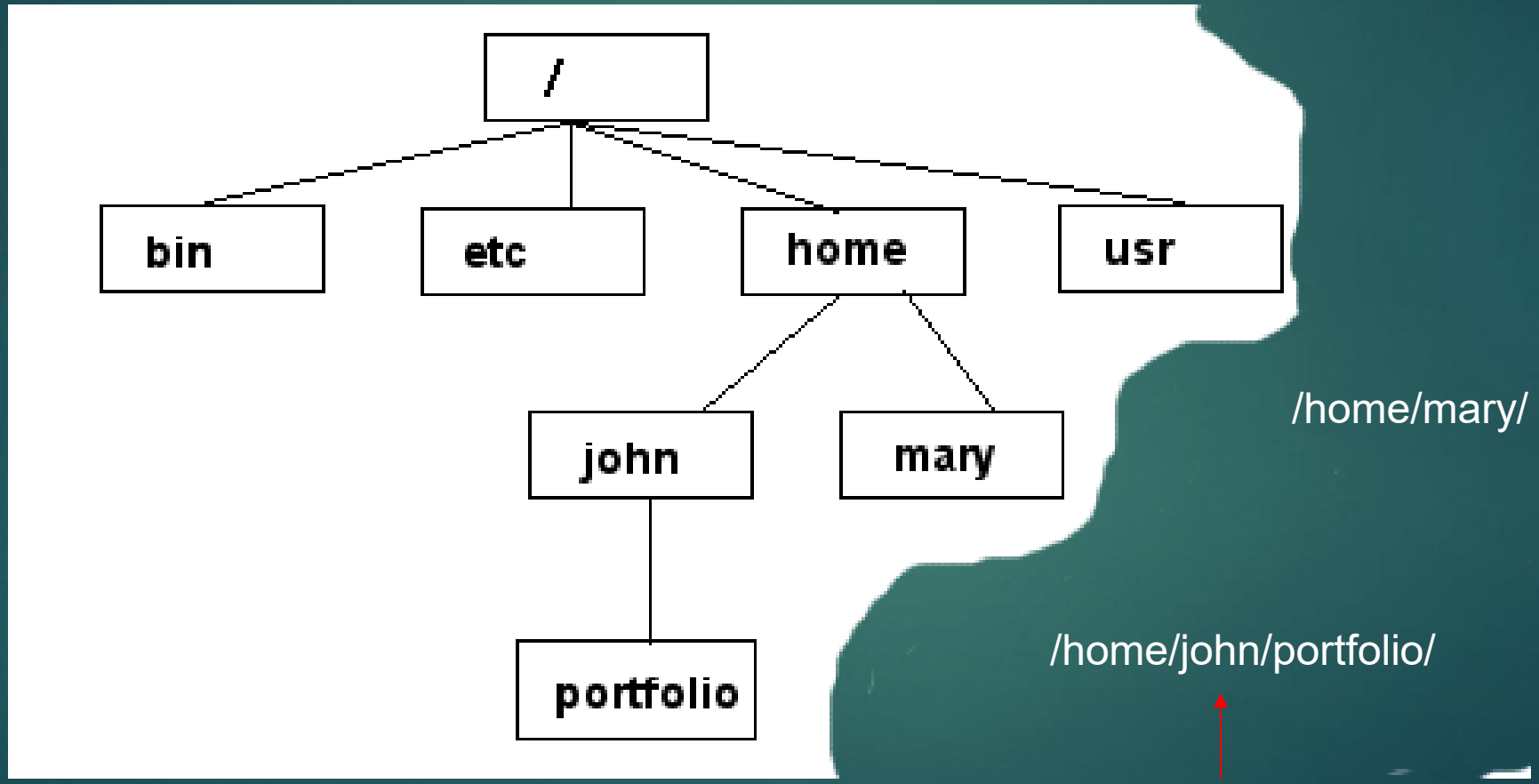

wiehe@zhome:~
ECHO(1) User Commands ECHO(1)
NAME
echo - display a line of text
SYNOPSIS
echo [OPTION]... [STRING]...
DESCRIPTION
NOTE: your shell may have its own version of echo which will supercede the version described here. Please refer to your shellâs documentation for details about the options 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!**



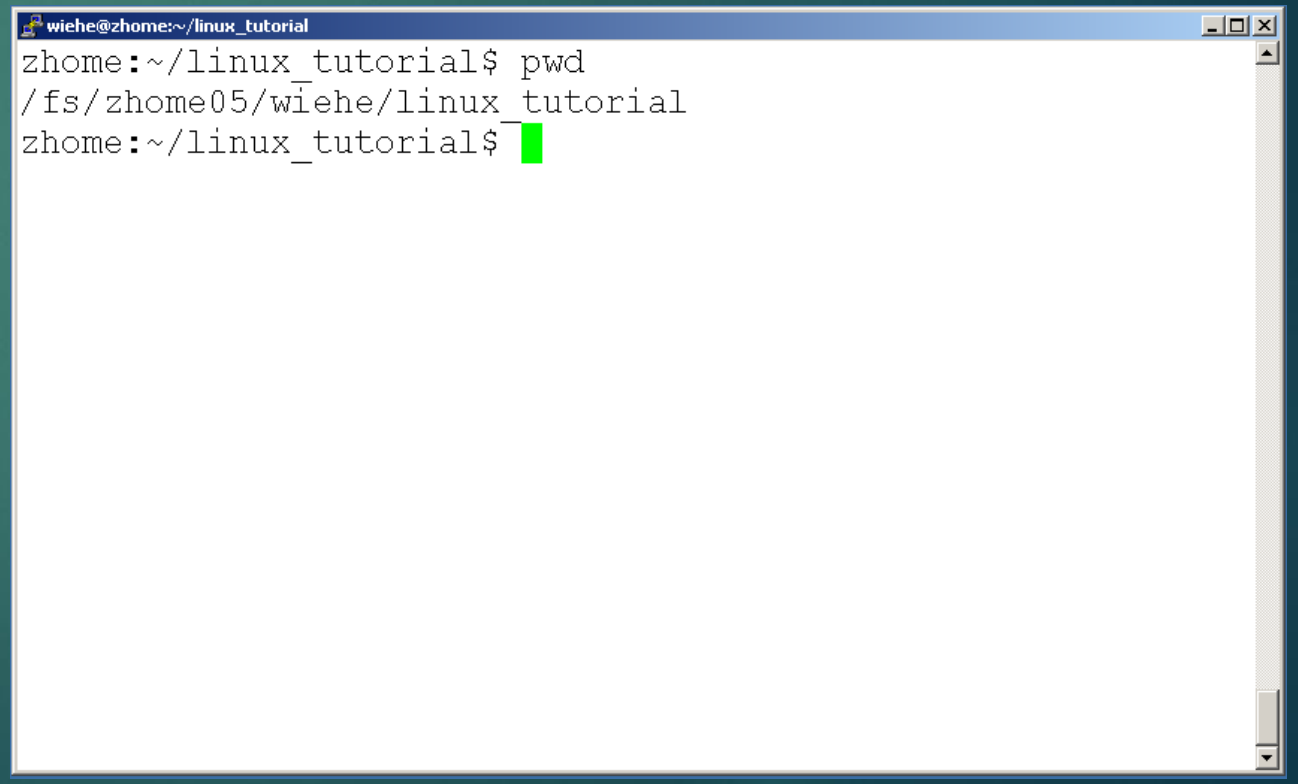
/home/mary/

/home/john/portfolio/

The Path

Command: pwd

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



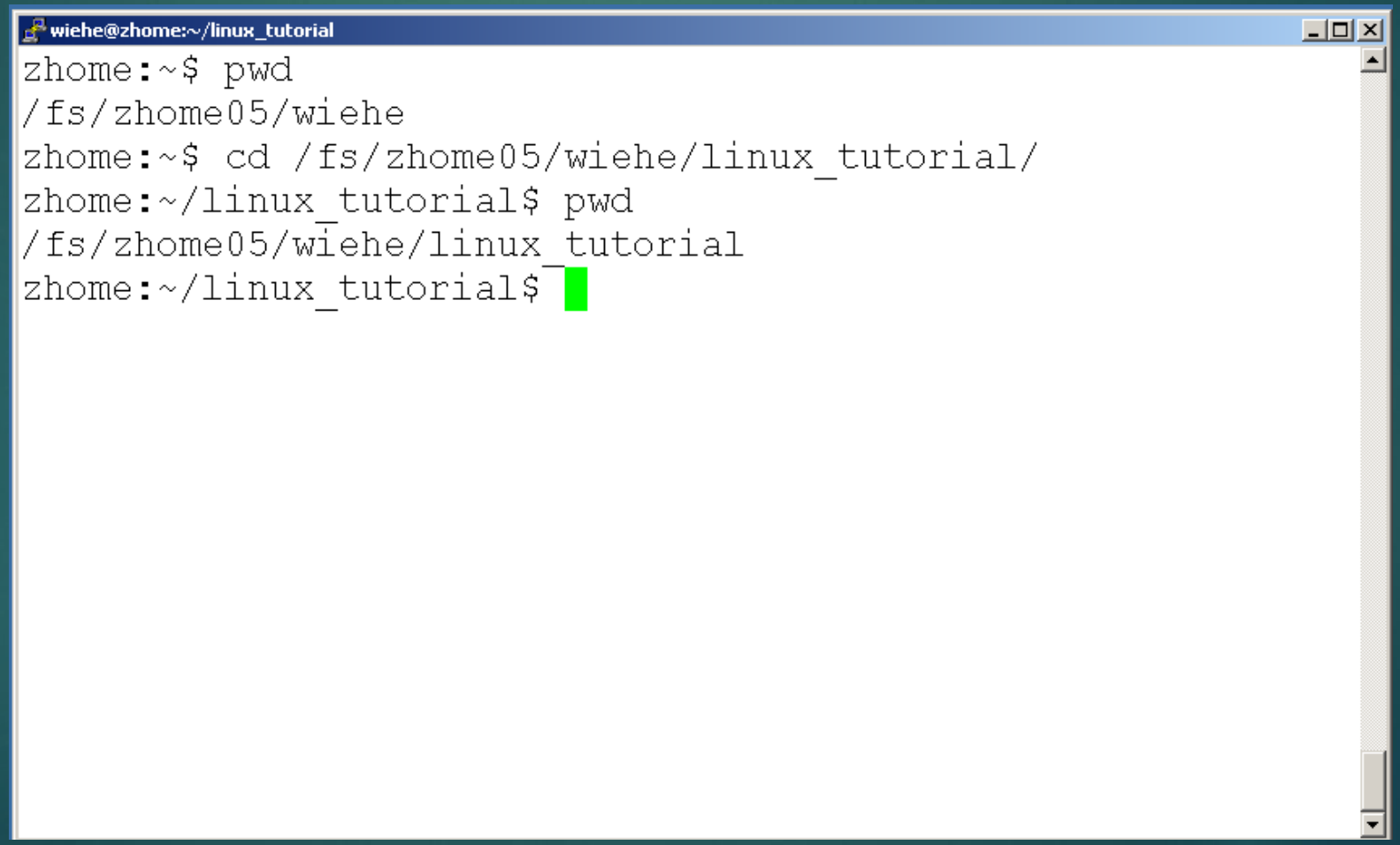
Touch

- Create a new file with no content

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

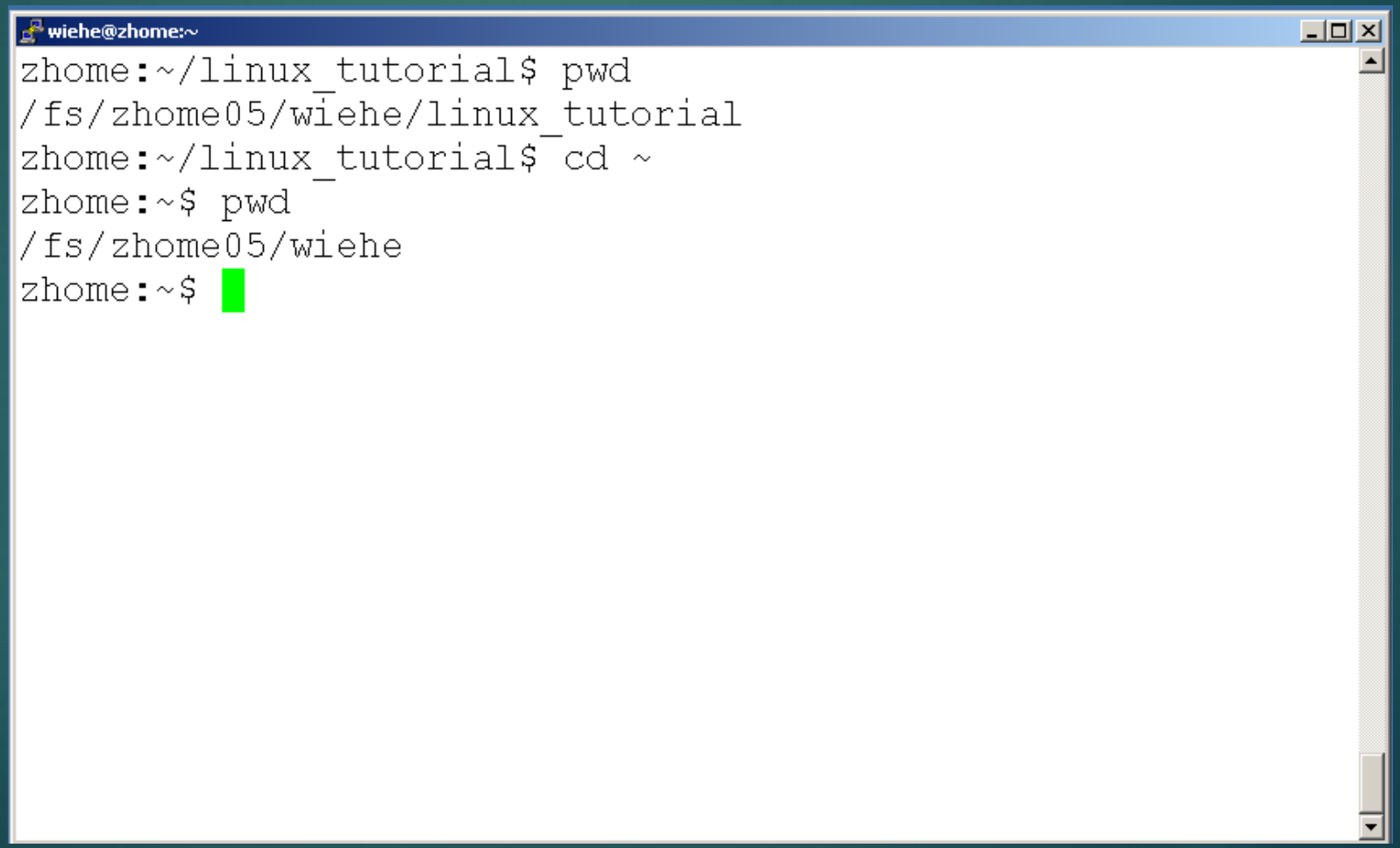
Command: cd

- To change to a specific directory use “cd”



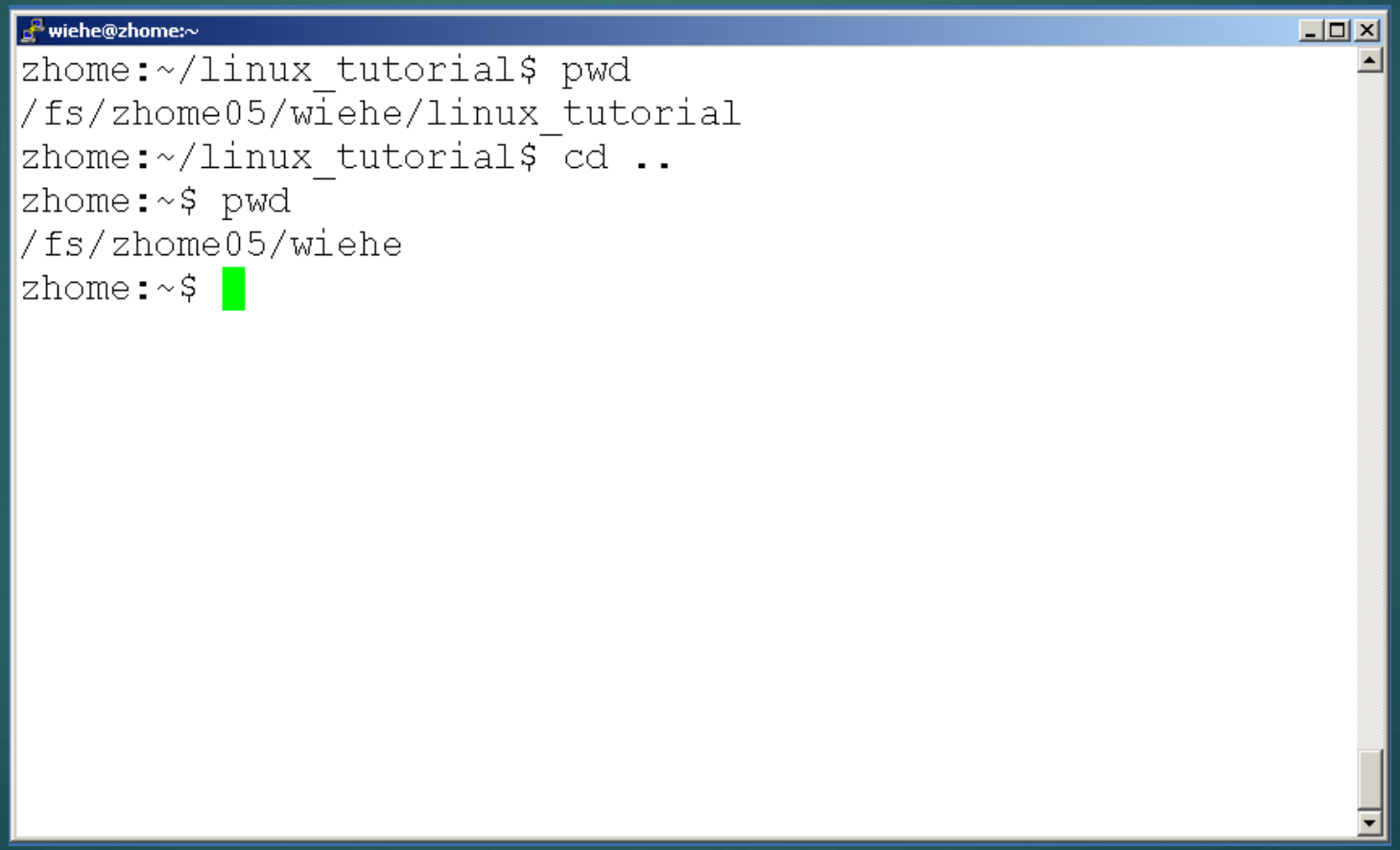
Command: cd

- "~" is the location of your home directory



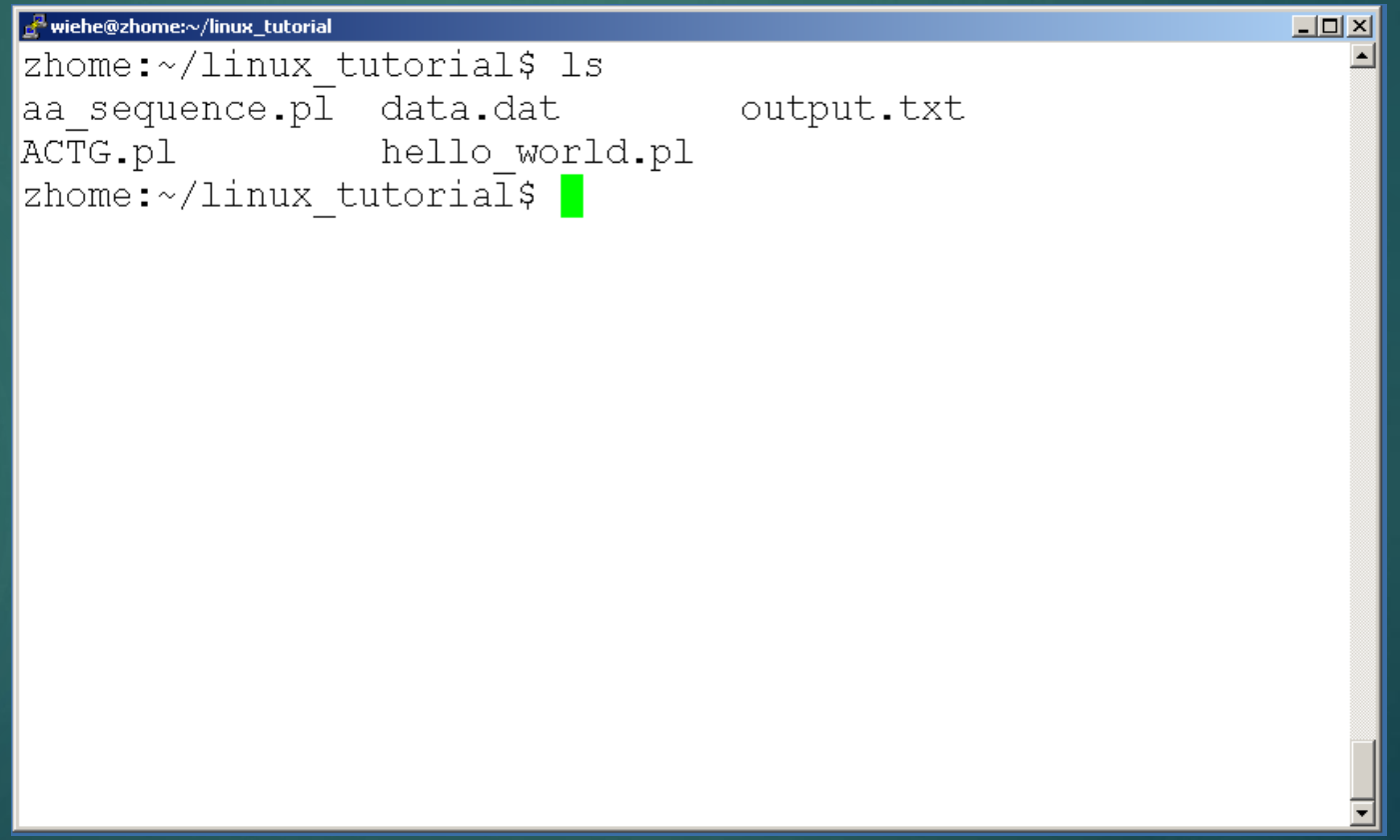
Command: cd

- ".." is the location of the directory below current one



Command: ls

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

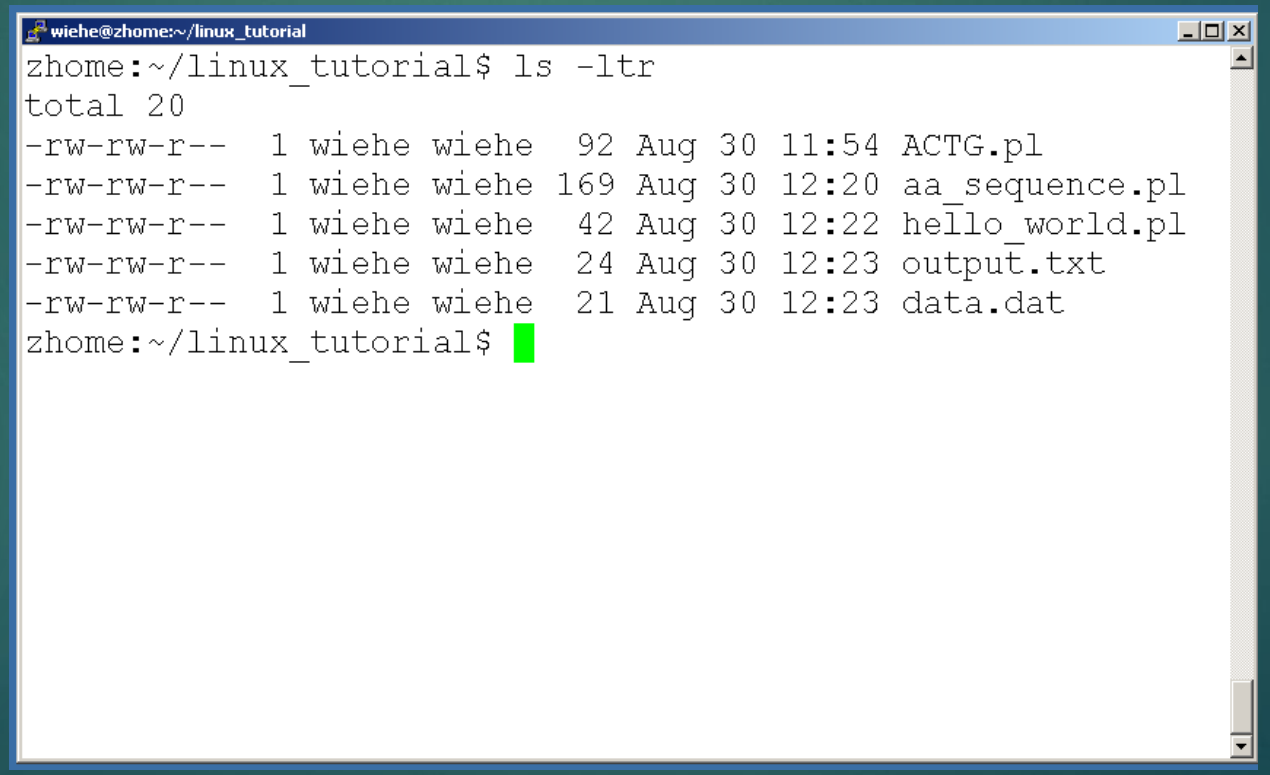


Command: ls

- 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"

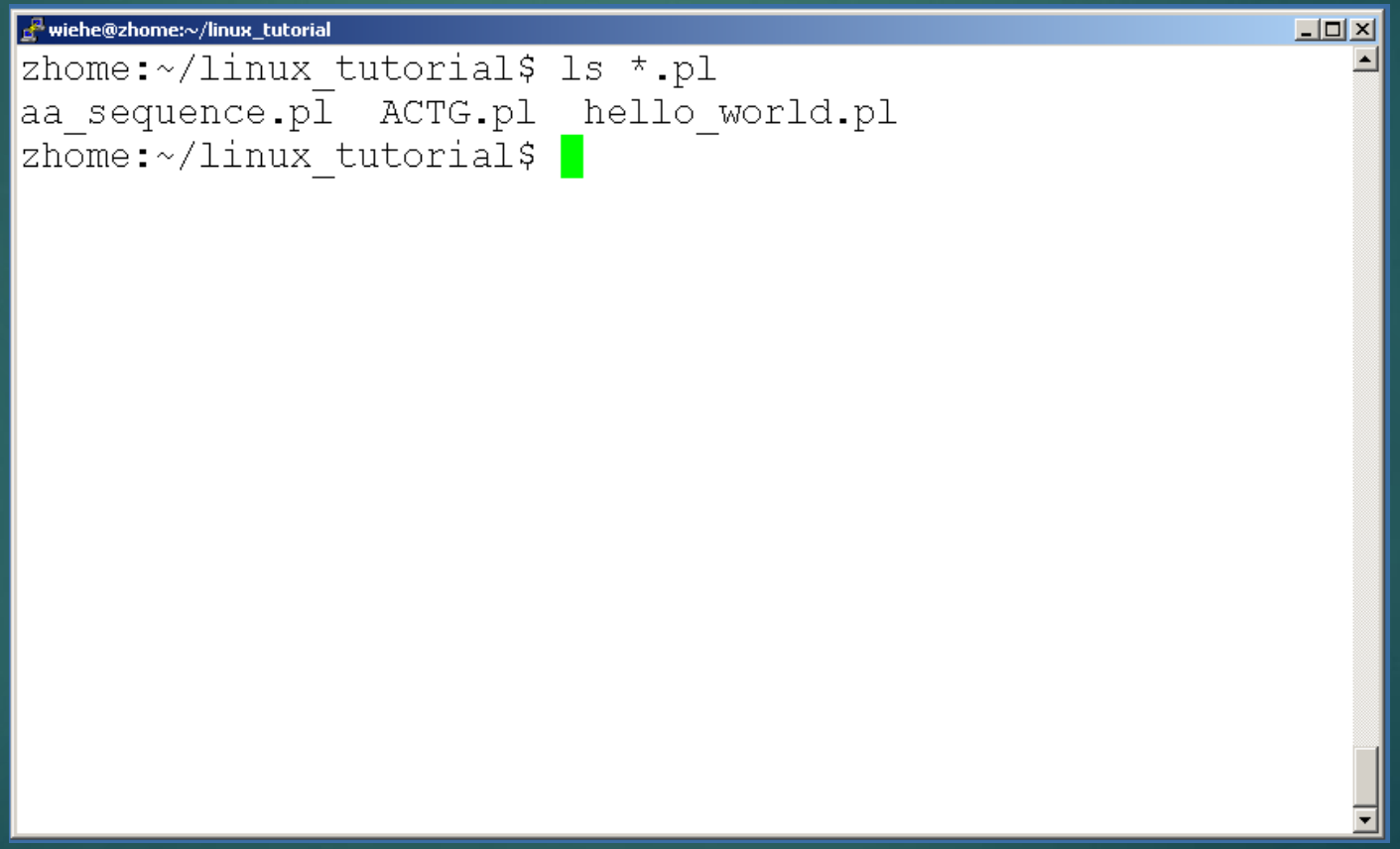
Command: ls -ltr

- List files by time in reverse order with long listing



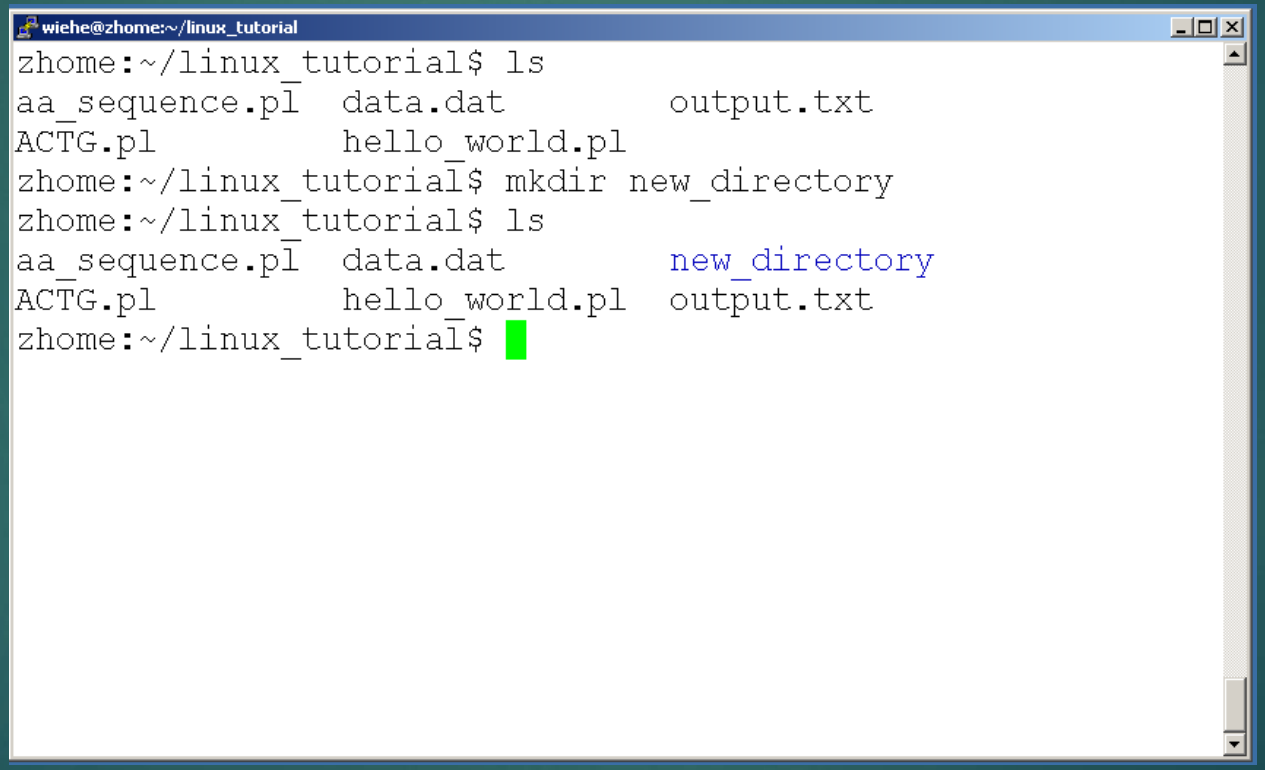
General Syntax: *

- "*" can be used as a wildcard in unix/linux



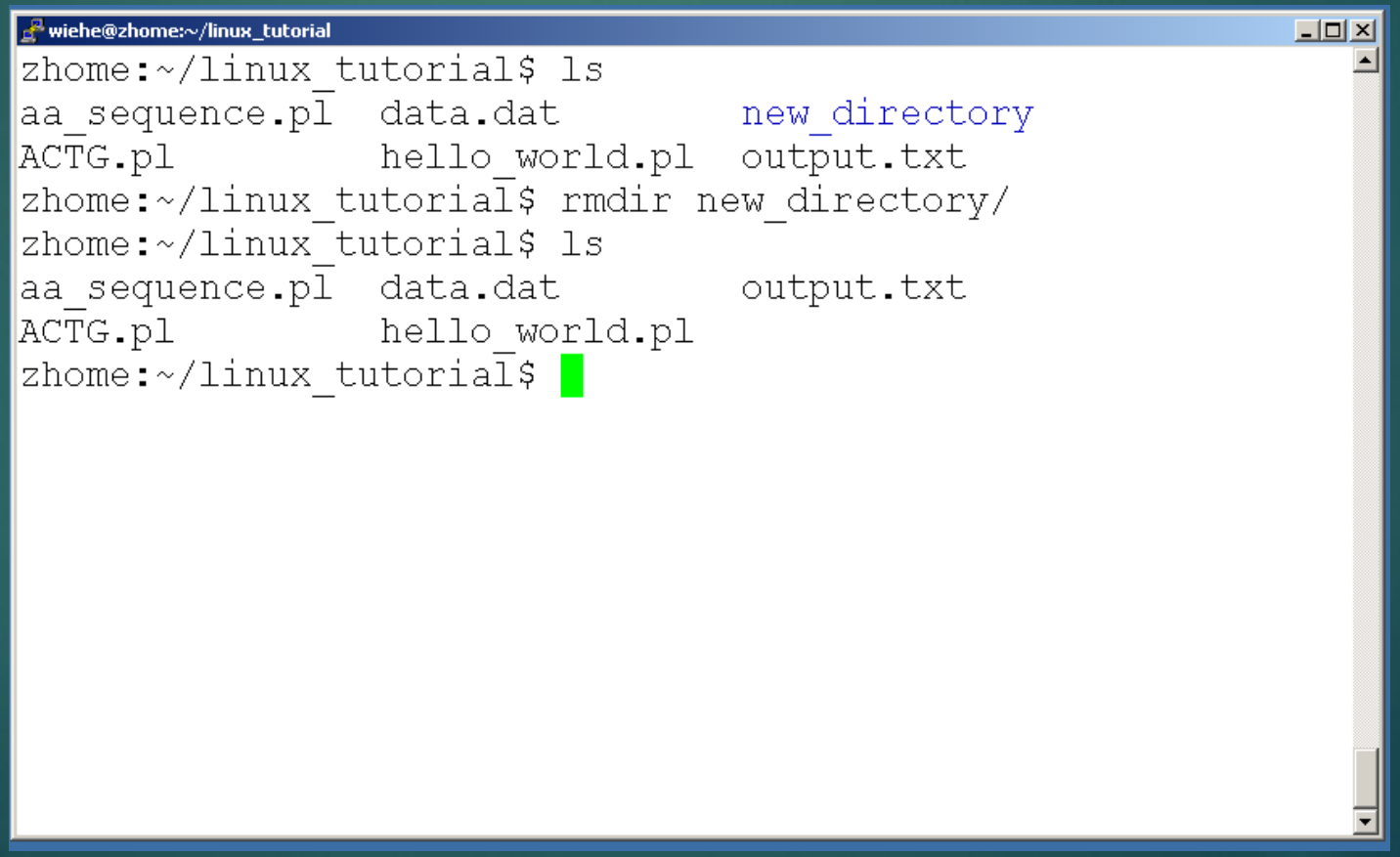
Command: mkdir

- To create a new directory use “mkdir”



Command: rmdir

- To remove and empty directory use “rmdir”



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

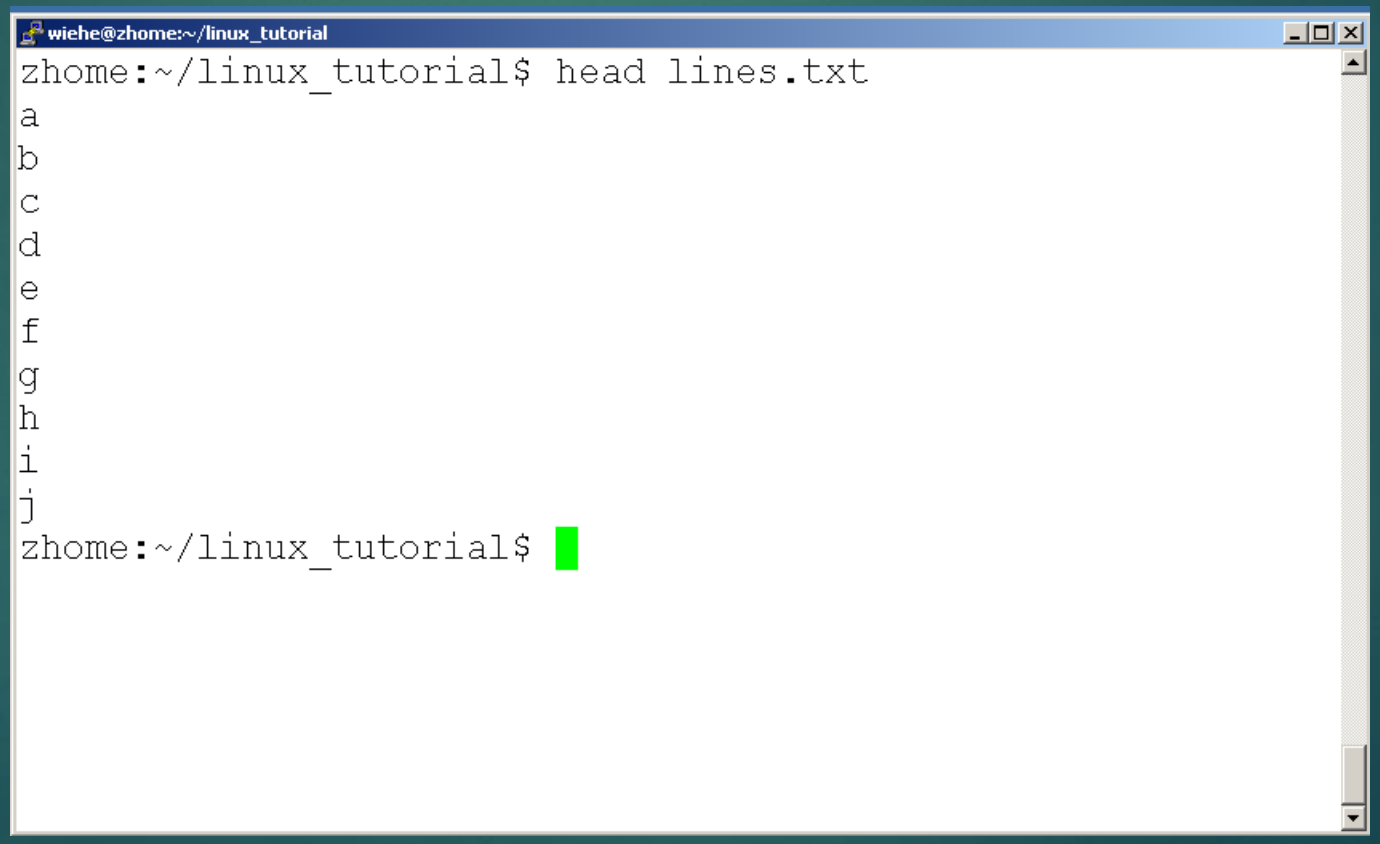
- "less" displays a file, allowing forward/backward movement within it
  - return scrolls forward one line, space one page
  - y scrolls back one line, b one page
- use "/" to search for a string
- Press q to quit

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

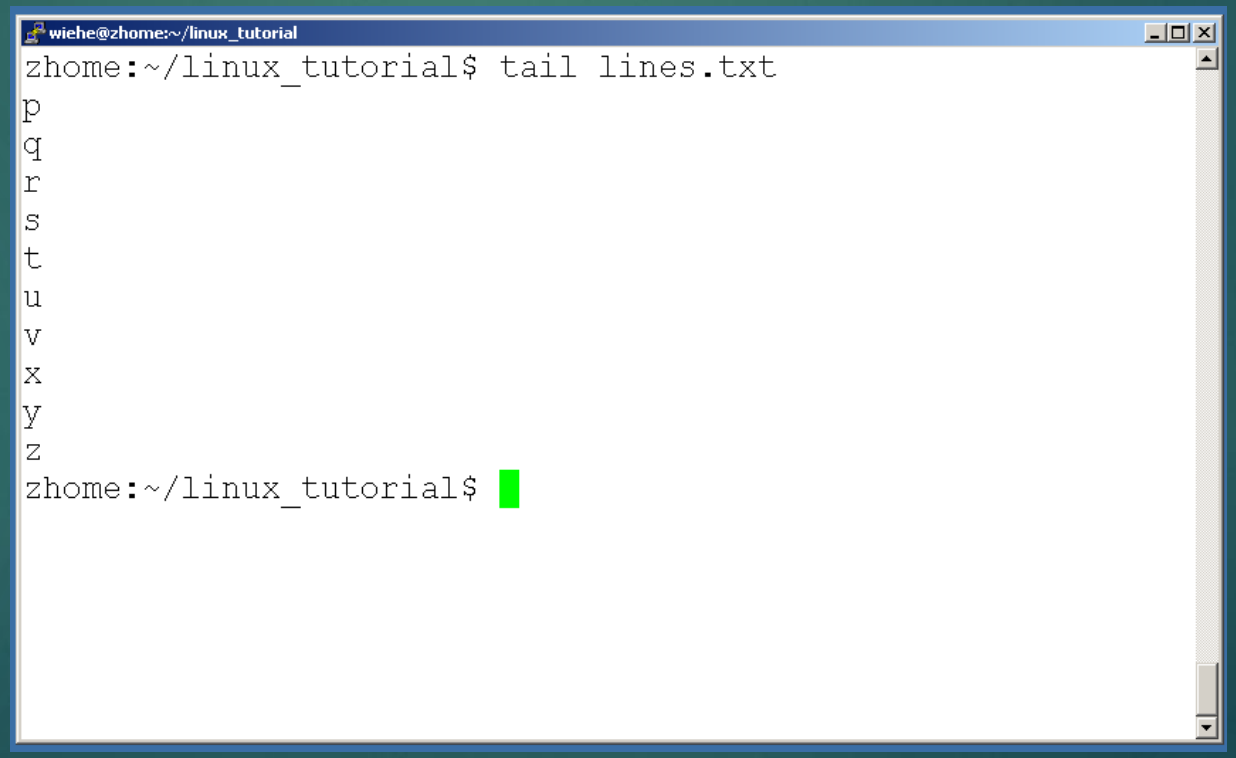
Command: head

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



Command: tail

- Same as head, but shows the last lines

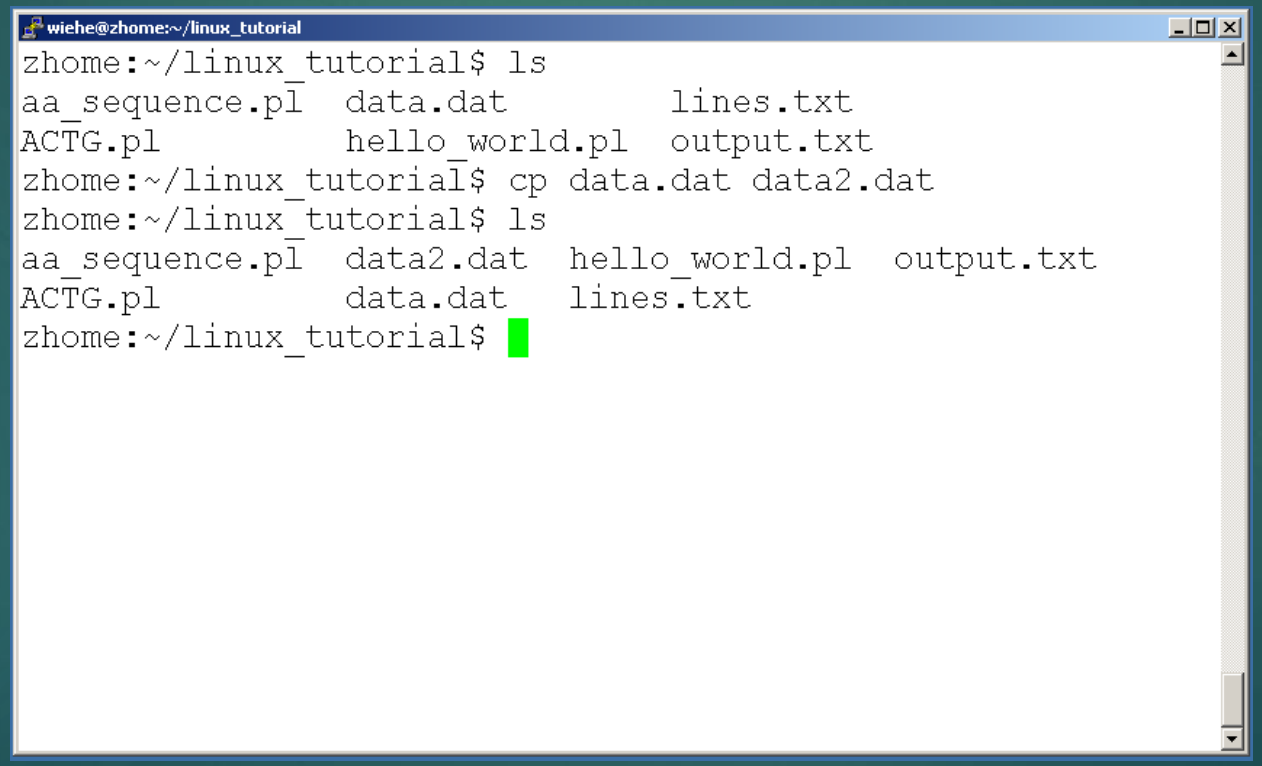


File Commands

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

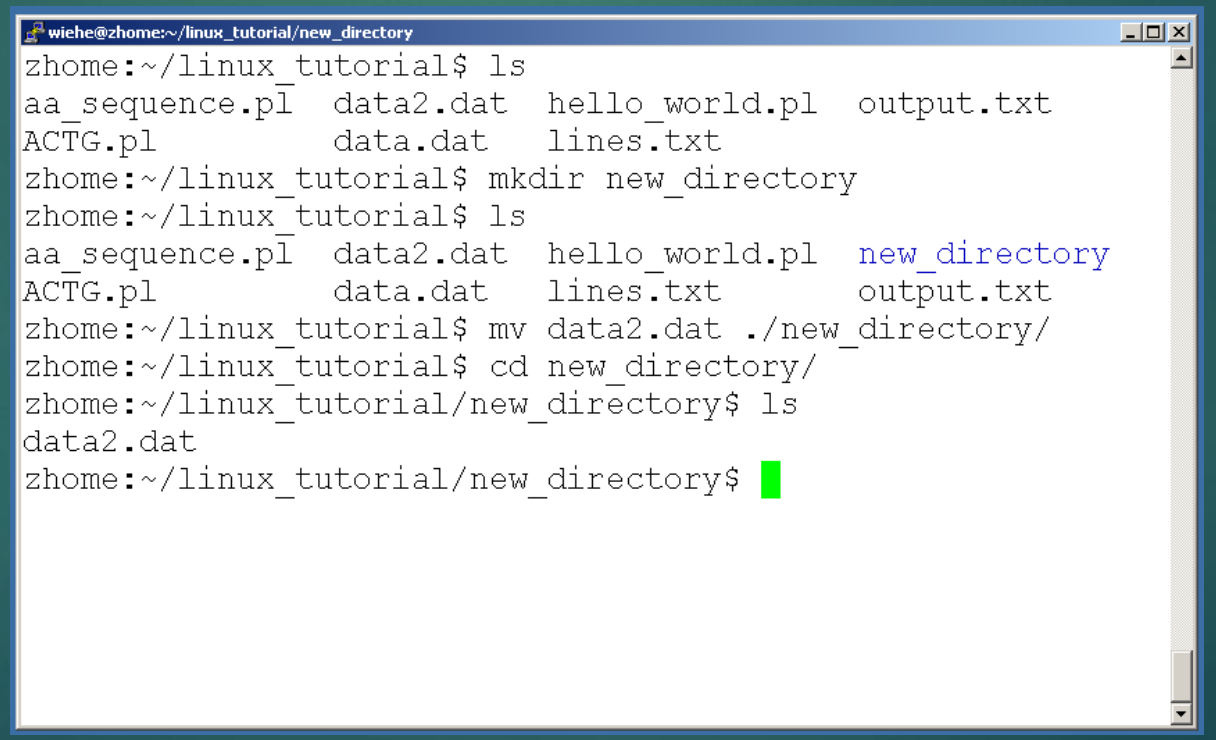
Command: cp

- To copy a file use “cp”



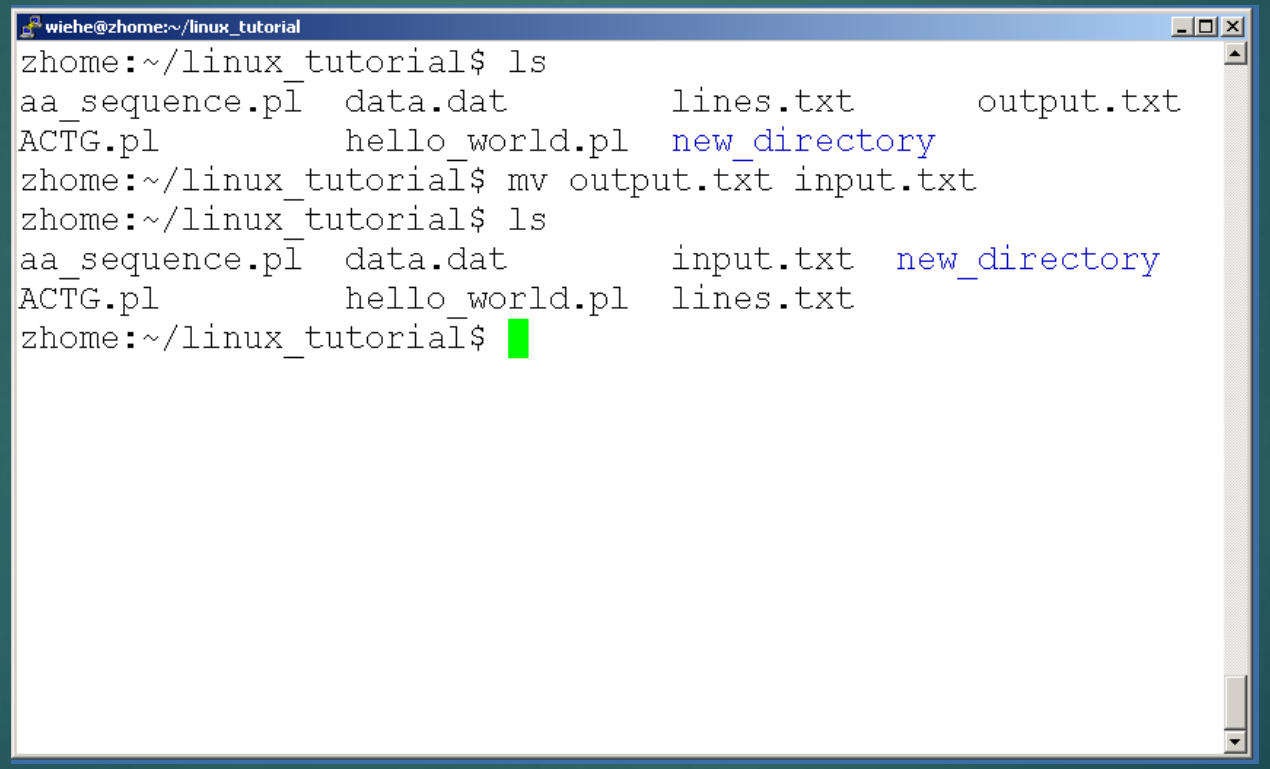
Command: mv

- To move a file to a different location use "mv"



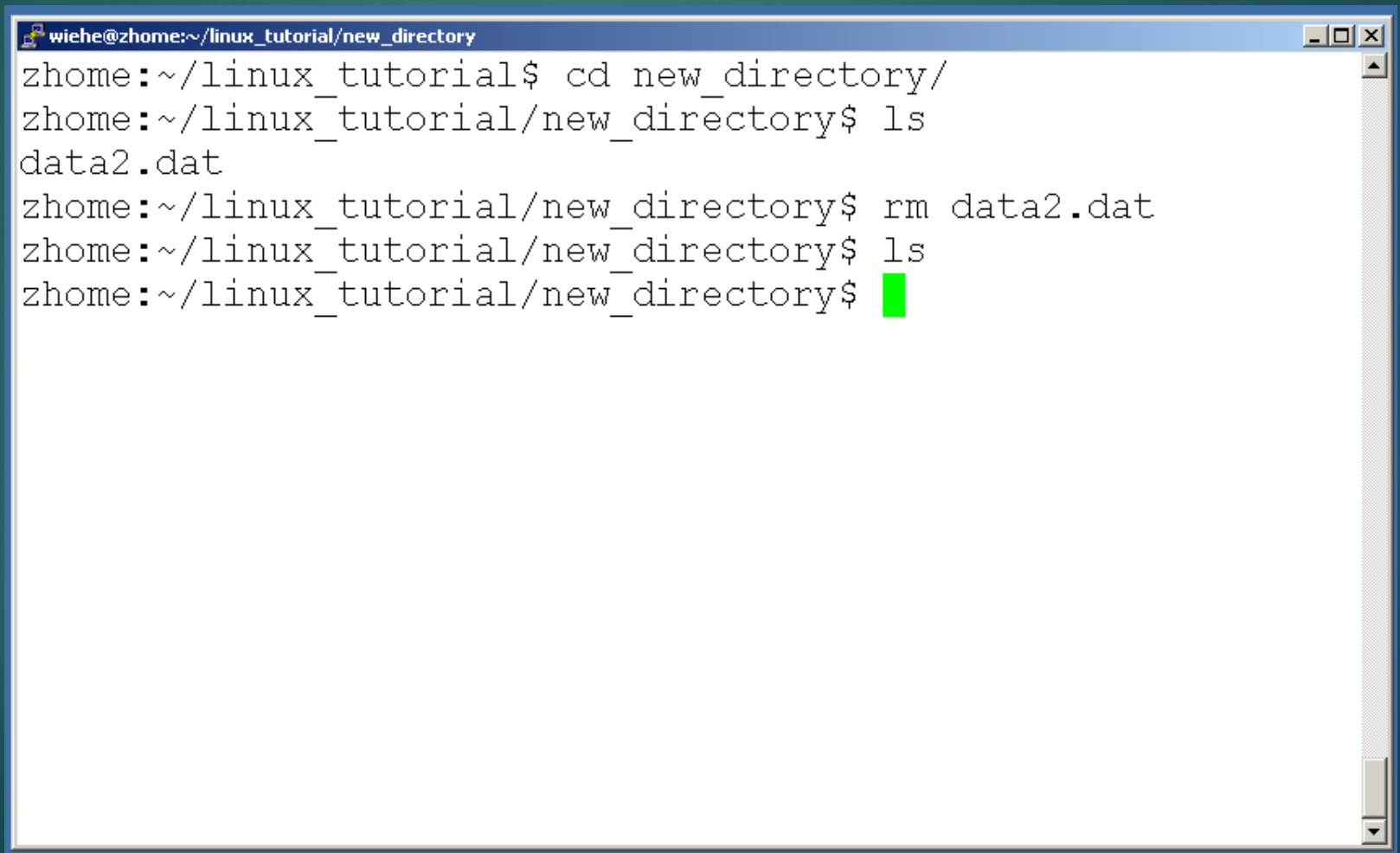
Command: mv

- mv can also be used to rename a file



Command: rm

- To remove a file use “rm”



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

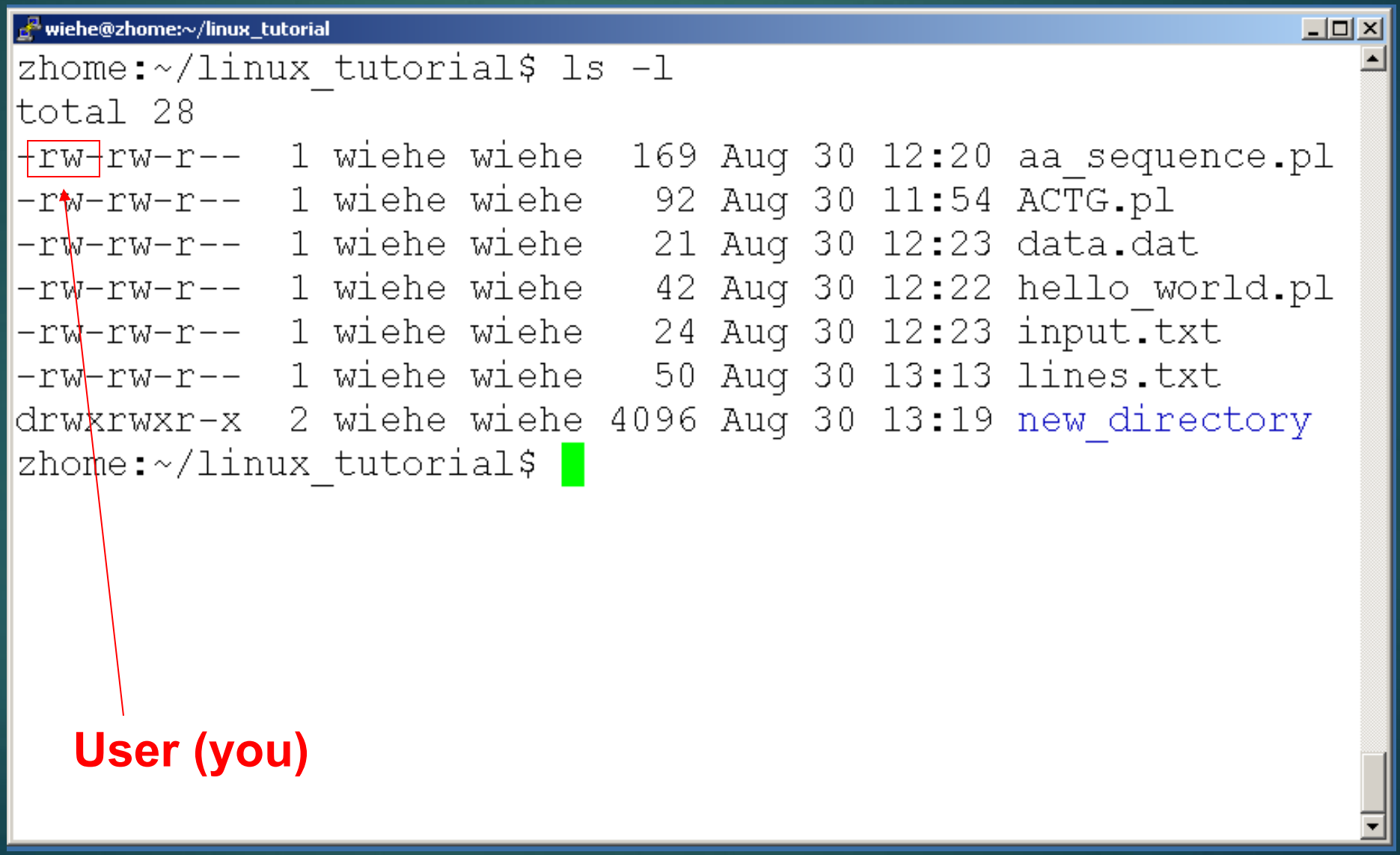
File permissions

- 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

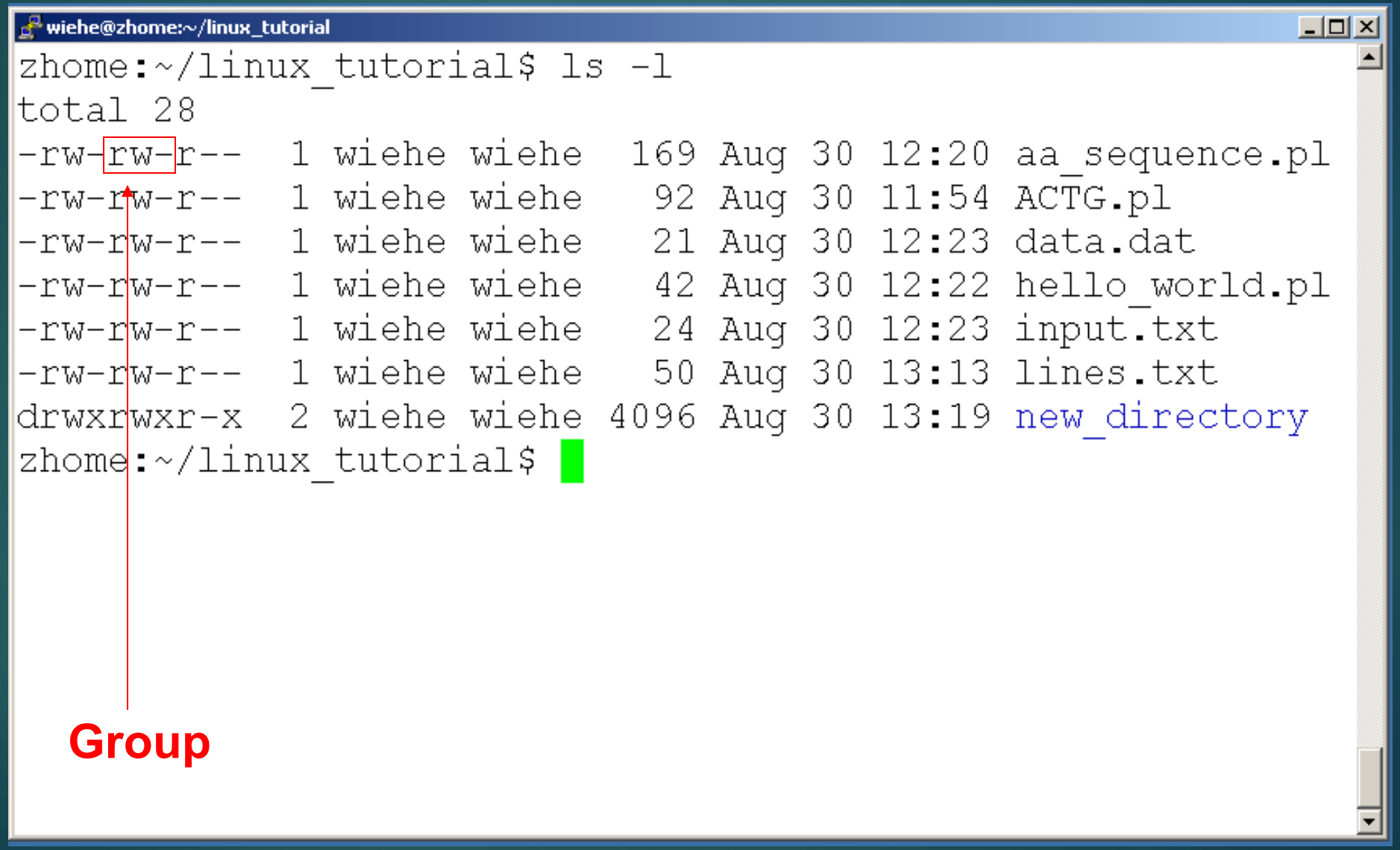
Permission levels

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

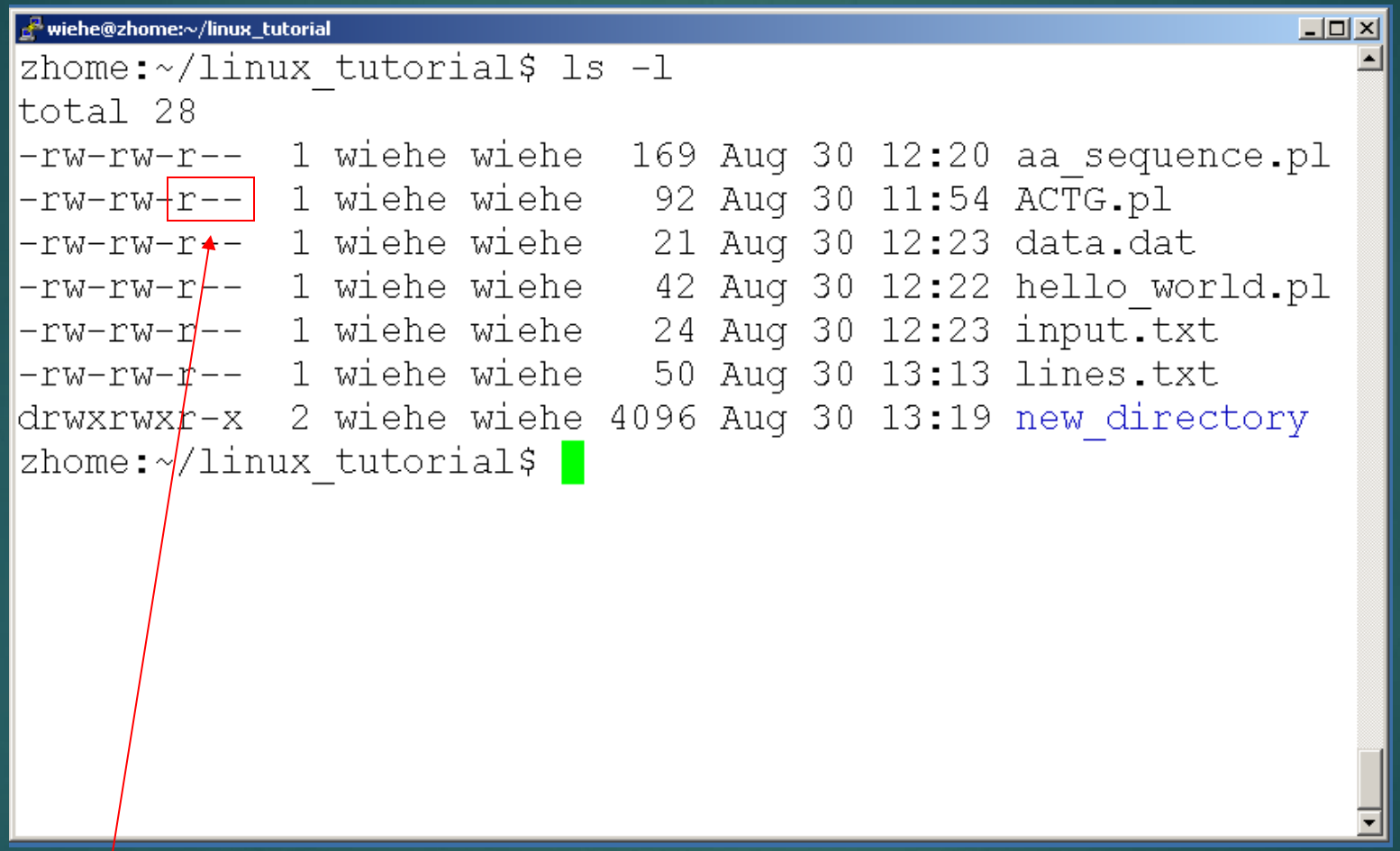
File Permissions



File Permissions



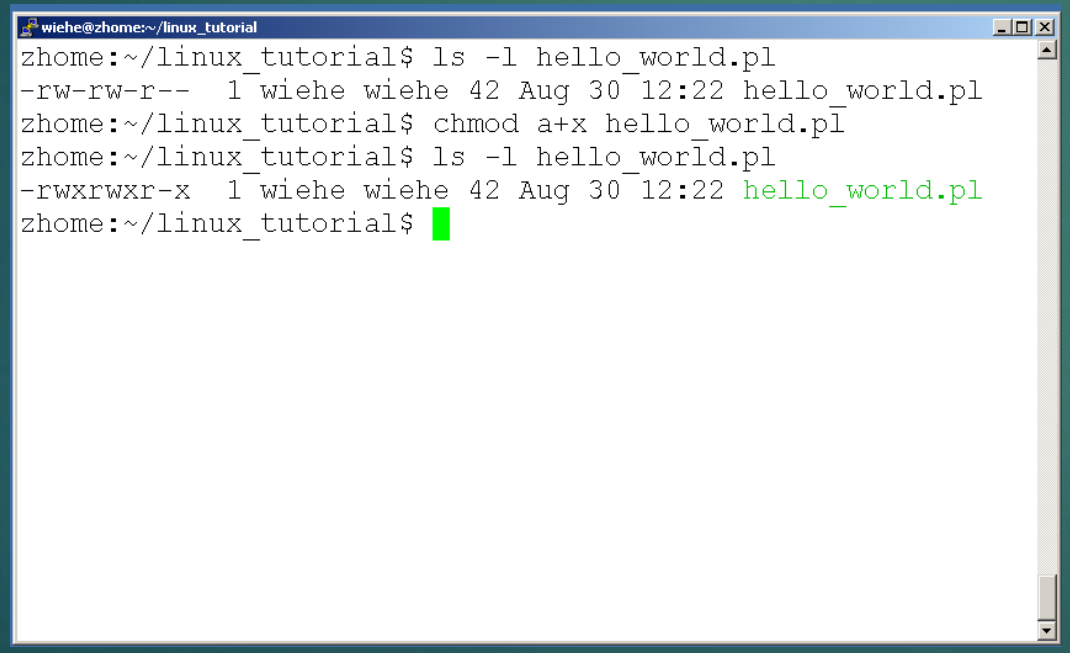
File Permissions



“The World”

Command: chmod

- If you own the file, you can change it's permissions with "chmod"
  - Syntax: chmod [**u**ser/**g**roup/**o**thers/**a**ll]+[permission] [file(s)]
  - Below we grant execute permission to all:



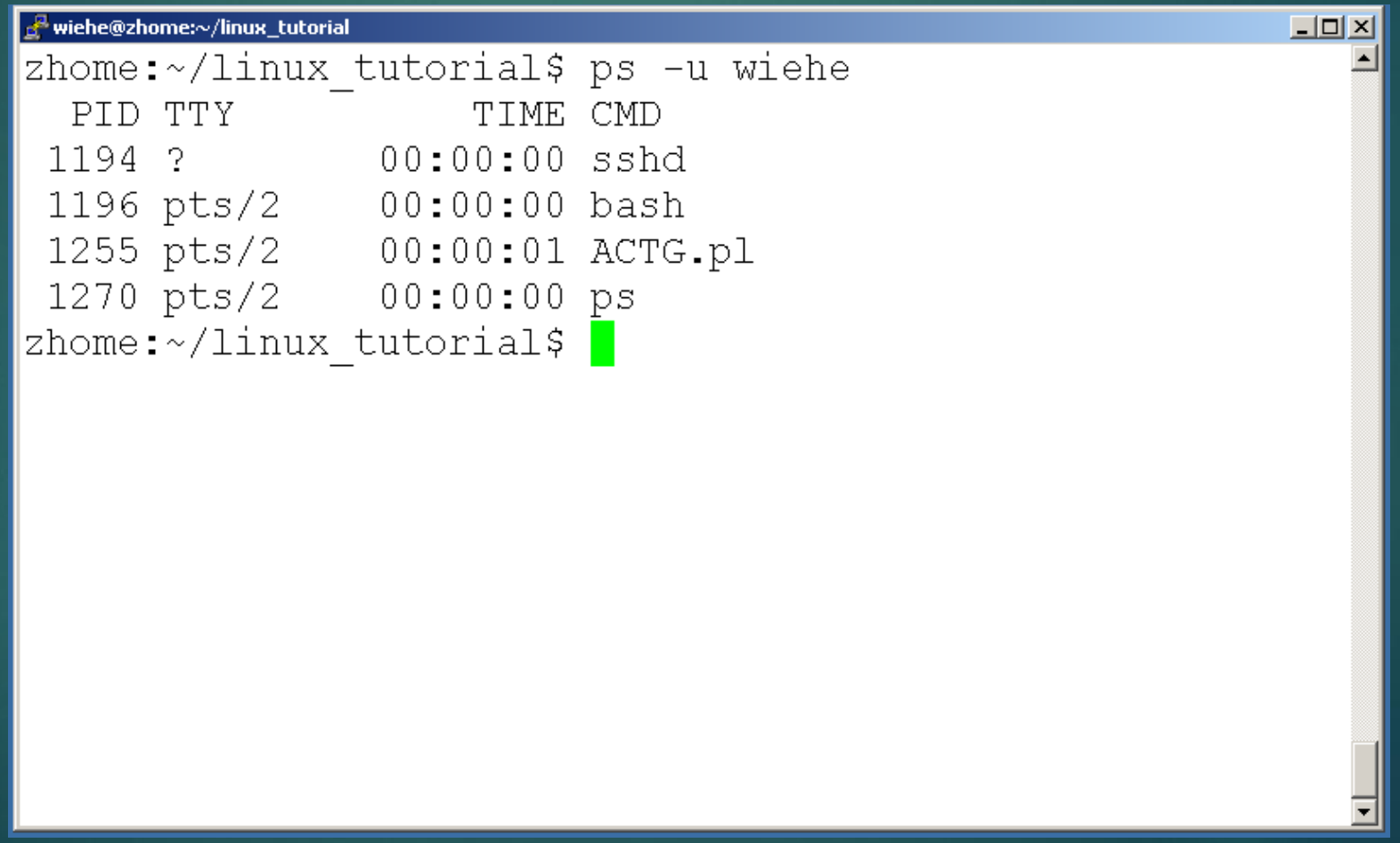
Permission

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----

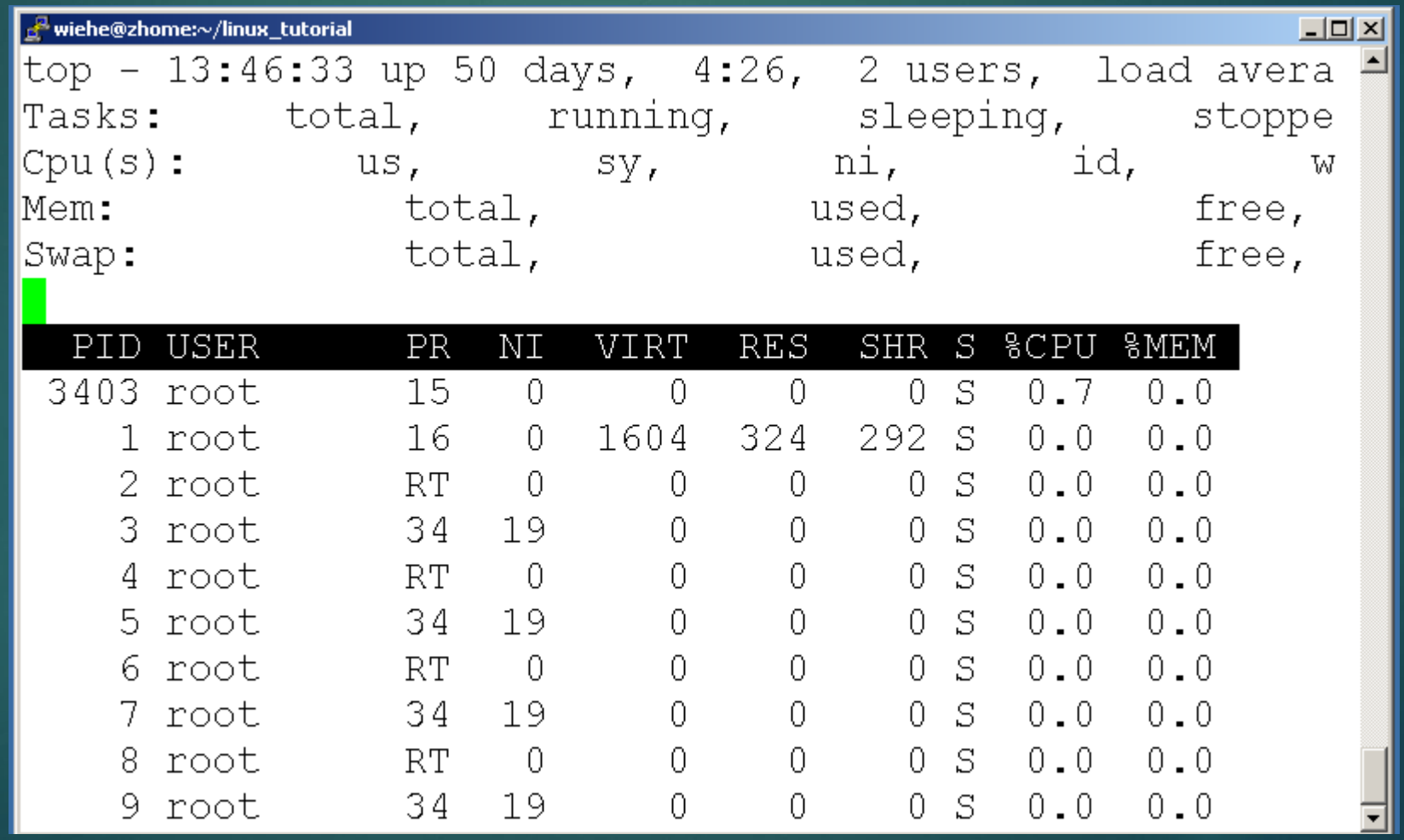
Command: ps

- To view the processes that you're running:



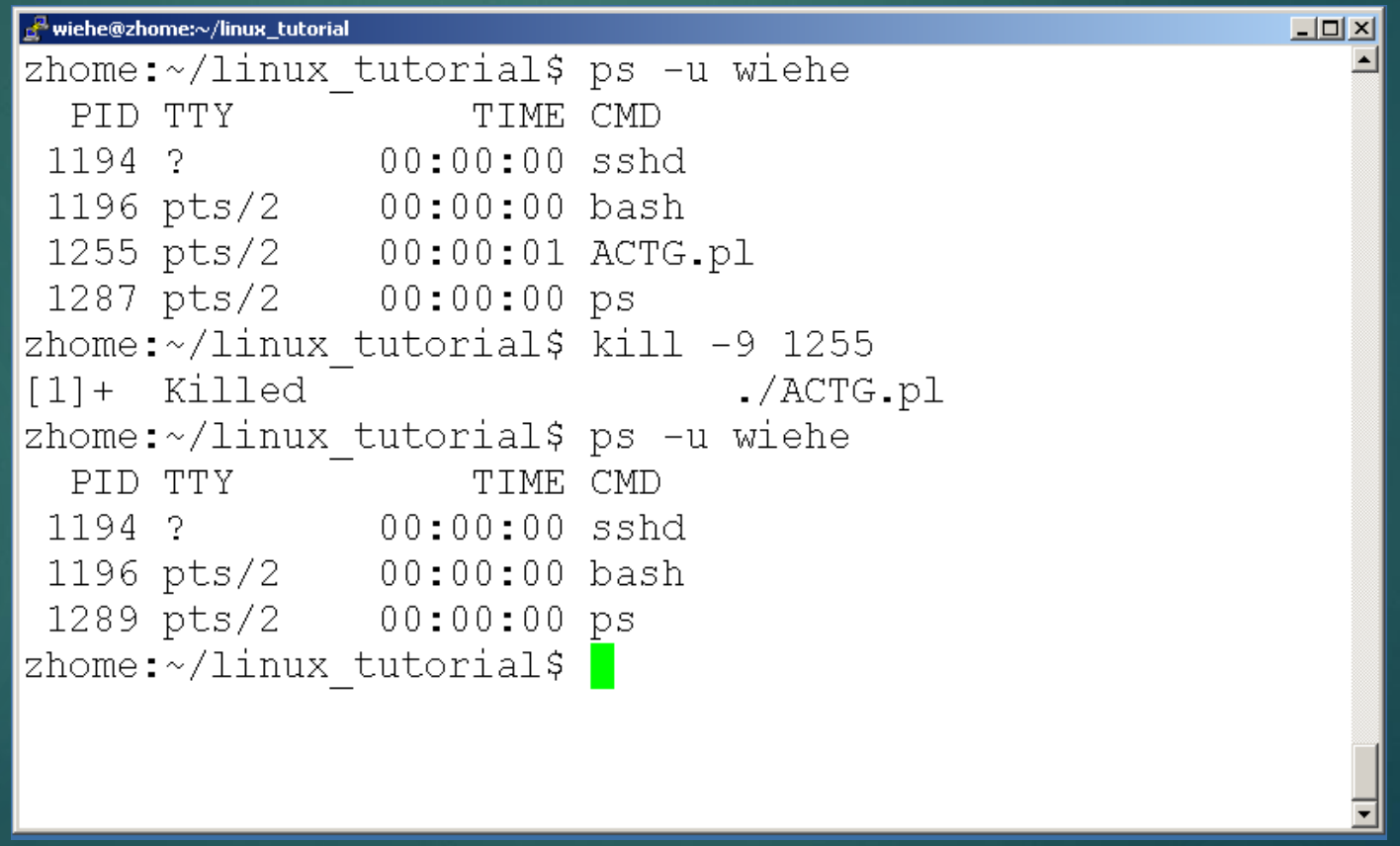
Command: top

- To view the CPU usage of all processes:



Command: kill

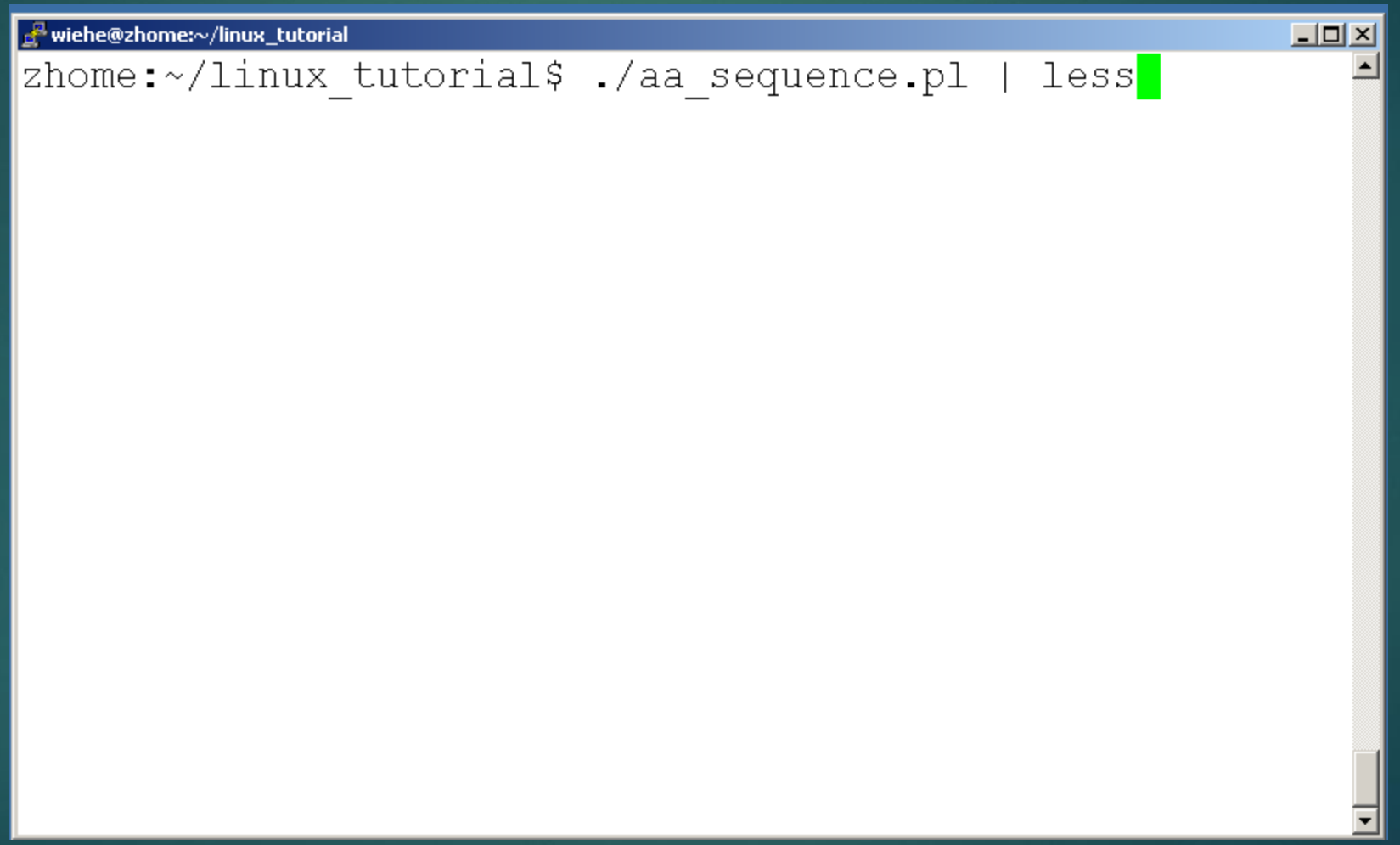
- To terminate a process use “kill”



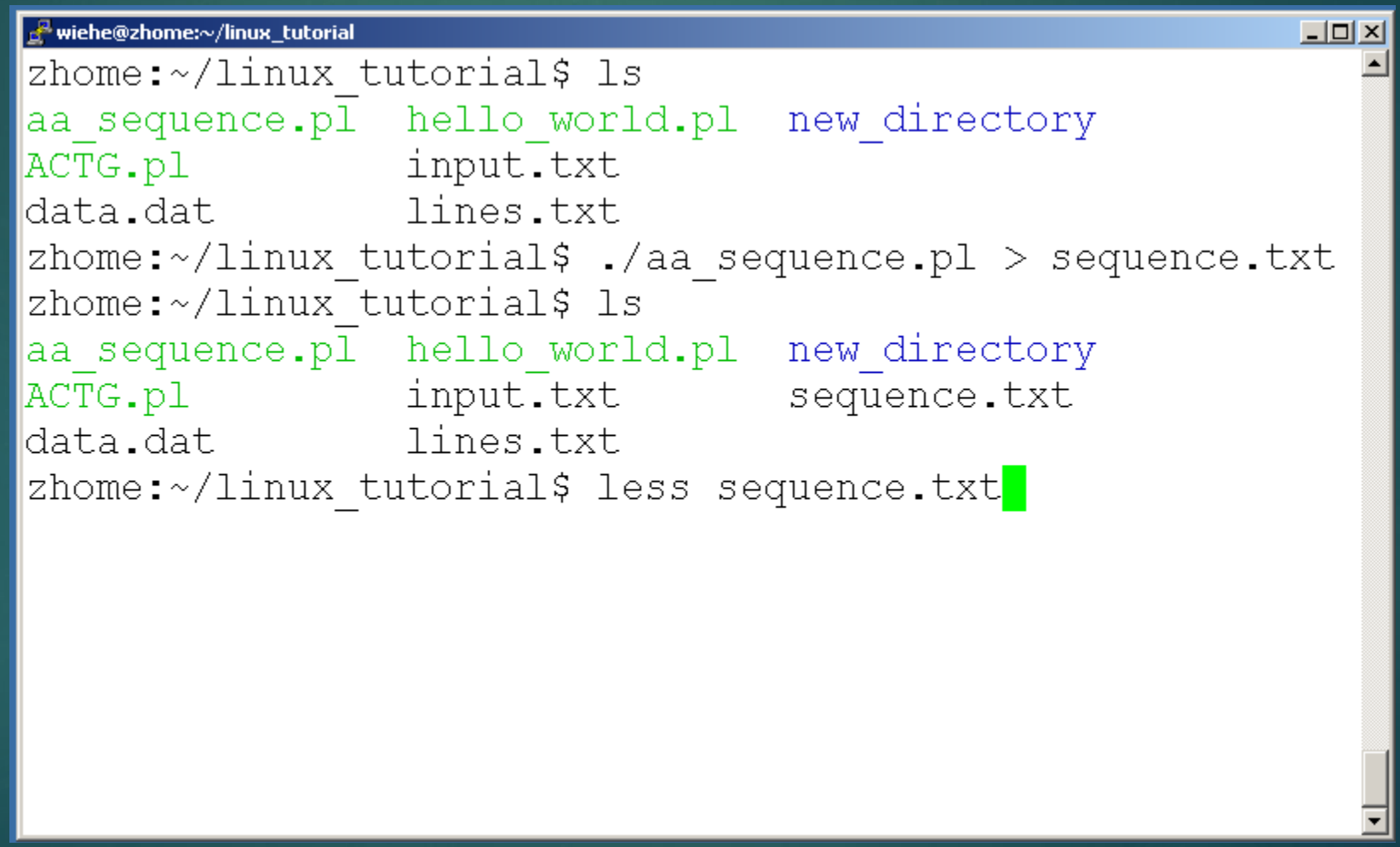
Input/Output Redirection (“piping”)

- 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



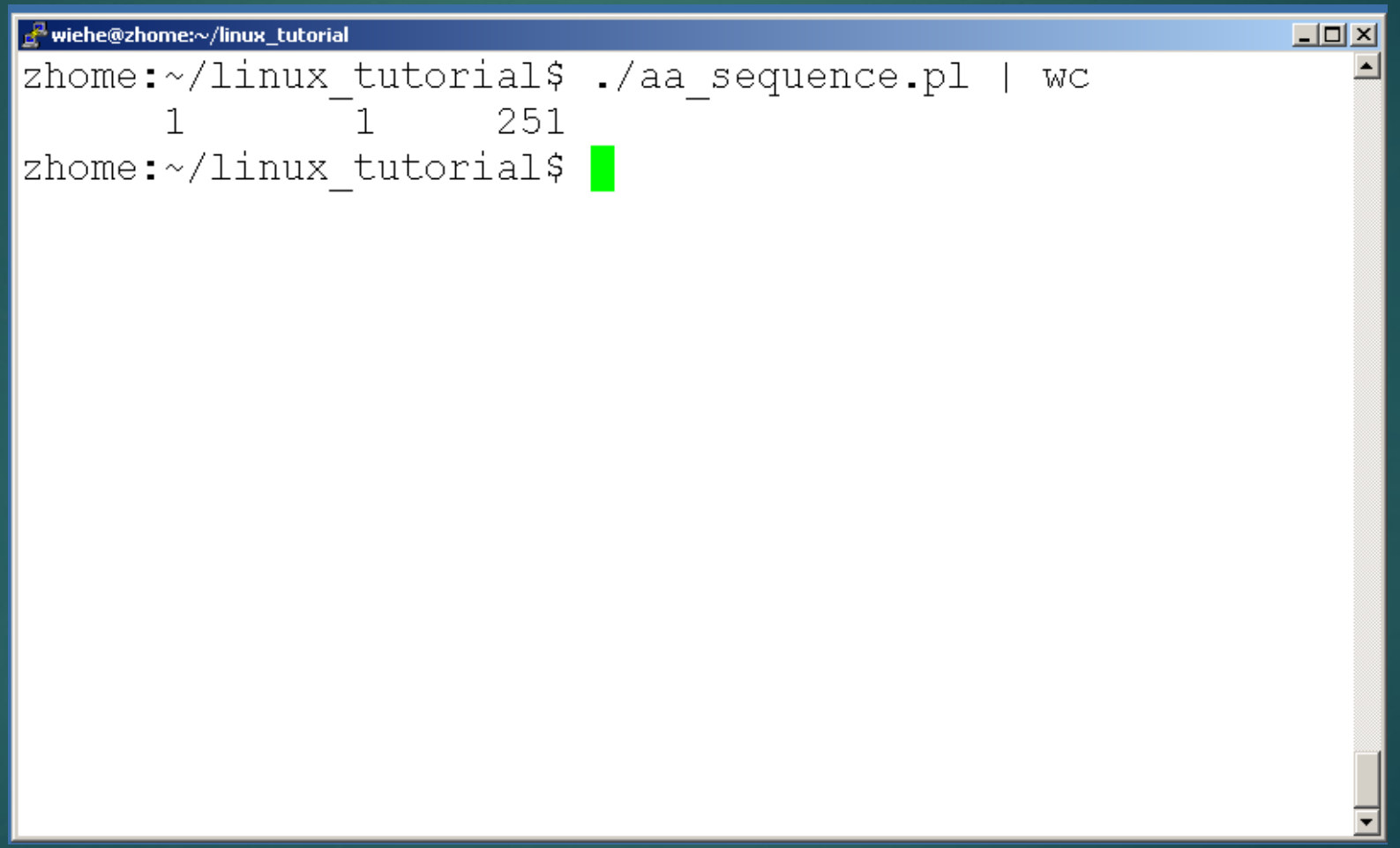
A few examples of piping



Command: wc

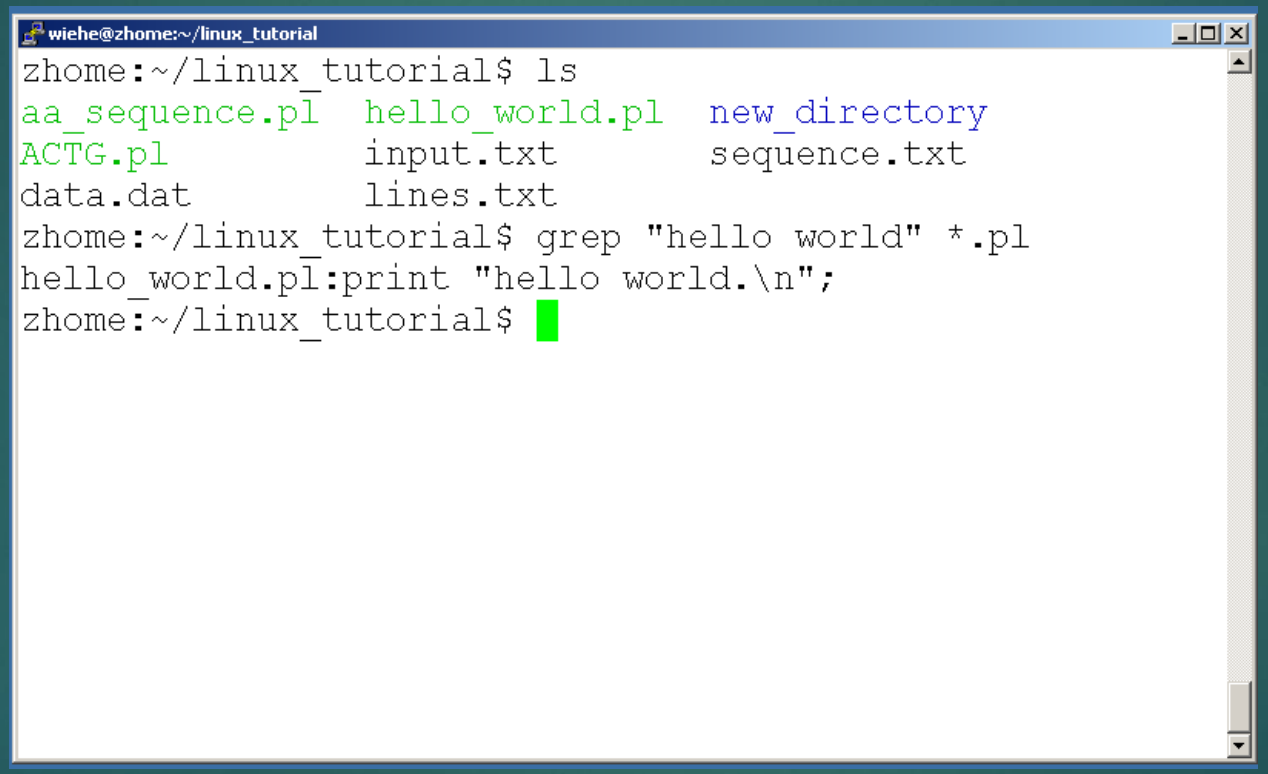
- To count the characters, words, and lines in a file use “wc”
- The first column in the output is lines, the second is words, and the last is characters

A few examples of piping



Command: grep

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



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

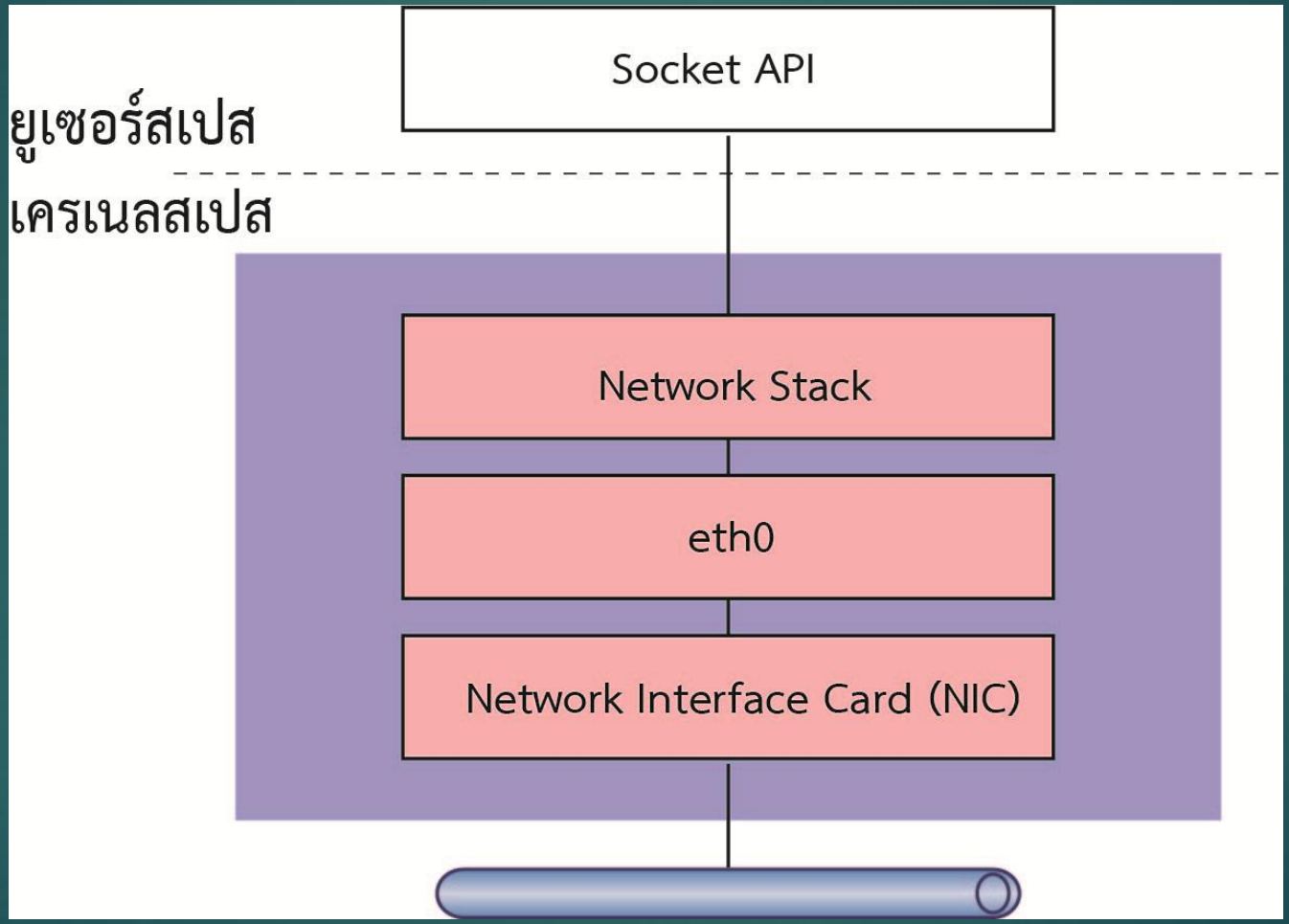
- **systemctl start** *service-name*
- **systemctl stop** *service-name*
- **systemctl restart** *service-name*
- **systemctl status** *service-name*

```
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
```

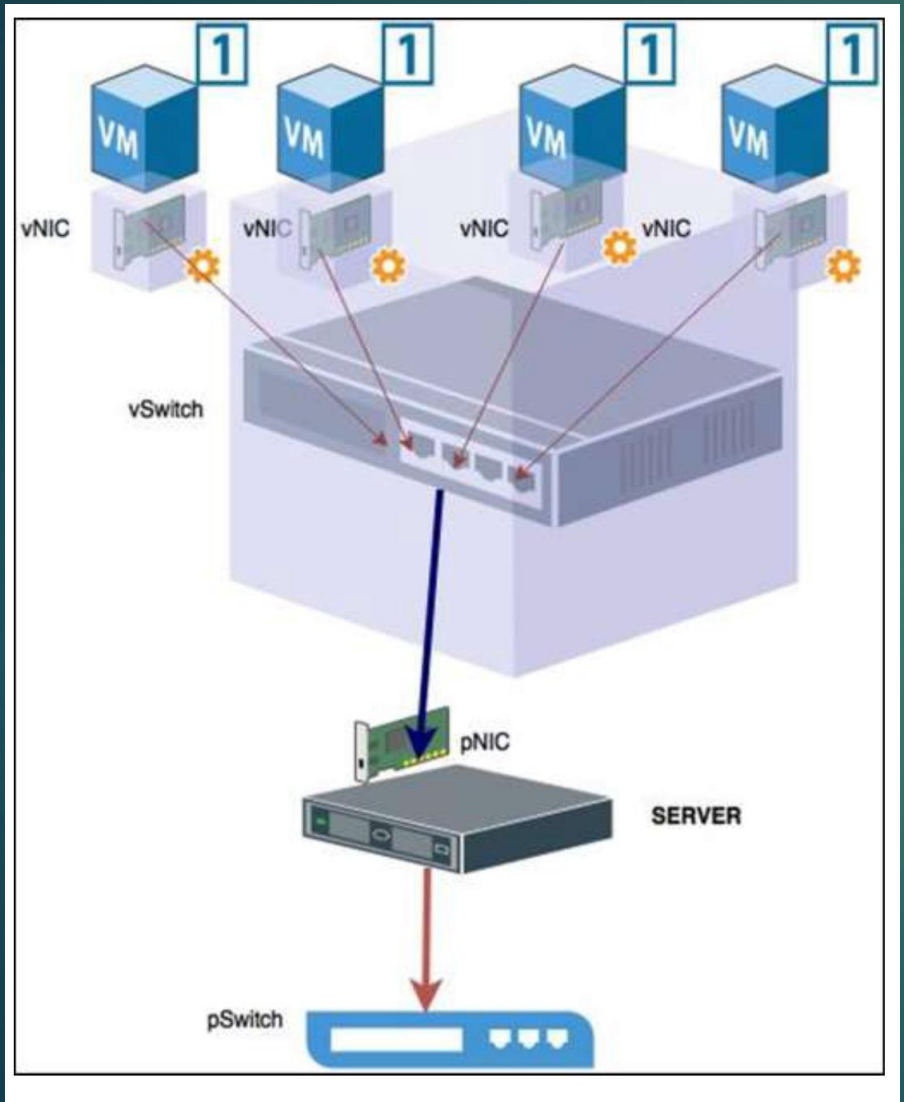
- *Note show config file*
- **systemctl cat** *service-name*

```
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.

คำอธิบาย ref: https://goinbigdata.com/demystifying-ifconfig-and-network-interfaces-in-linux/

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 ~]$
```

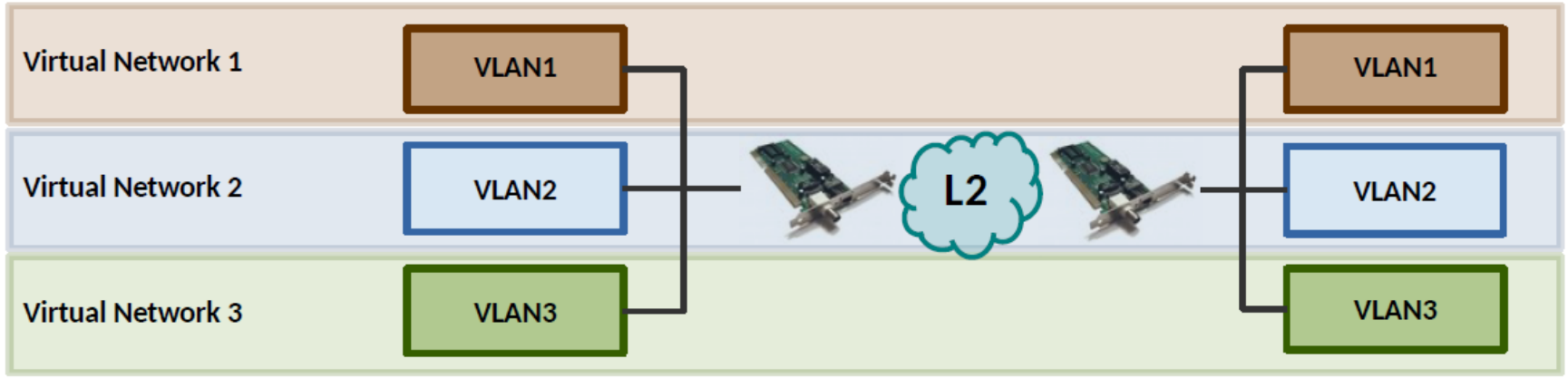
- Setting the MTU of an interface
  - ip link set mtu *MTU interface*

```
[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

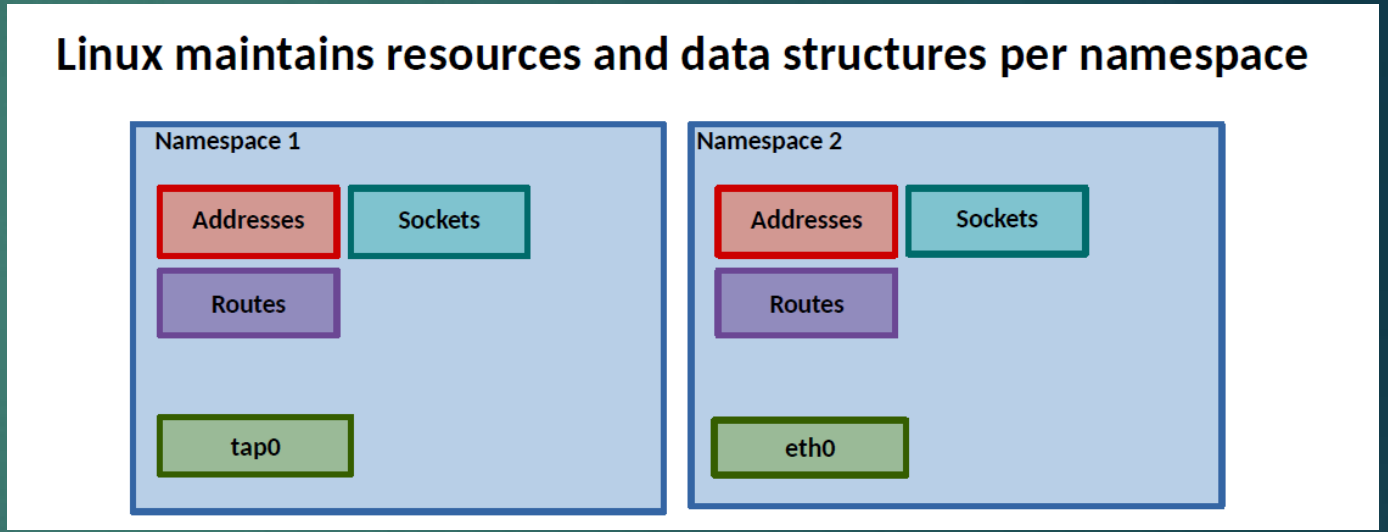
```
ck_bo@chatchaiasusPC:~$ lsns
        NS TYPE   NPROCS   PID USER  COMMAND
4026531834 time       33  1483 ck_bo /bin/bash /usr/local/bin/nginx-entrypoint.sh
4026531835 cgroup      4  2171 ck_bo -bash
4026531837 user       33  1483 ck_bo /bin/bash /usr/local/bin/nginx-entrypoint.sh
4026531840 net         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***



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:~$
```

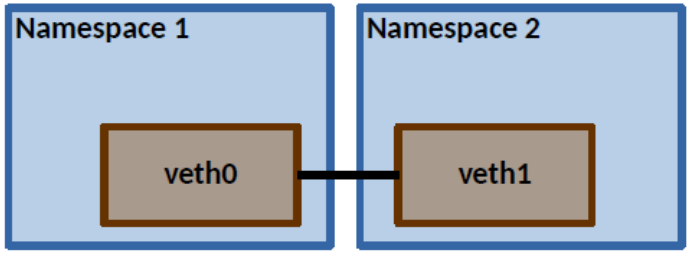
Executing Commands in a Network Namespace (2)

- 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



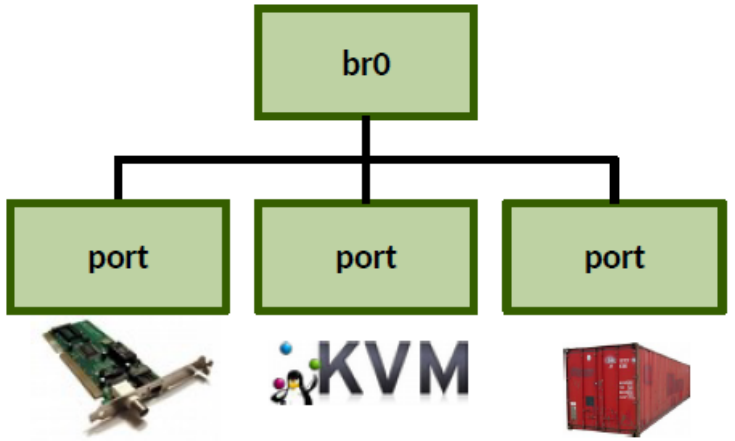
```
$ ip link add veth1 type veth peer name veth2
$ ip link set veth1 netns ns1
$ ip link set veth2 netns ns2
```

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

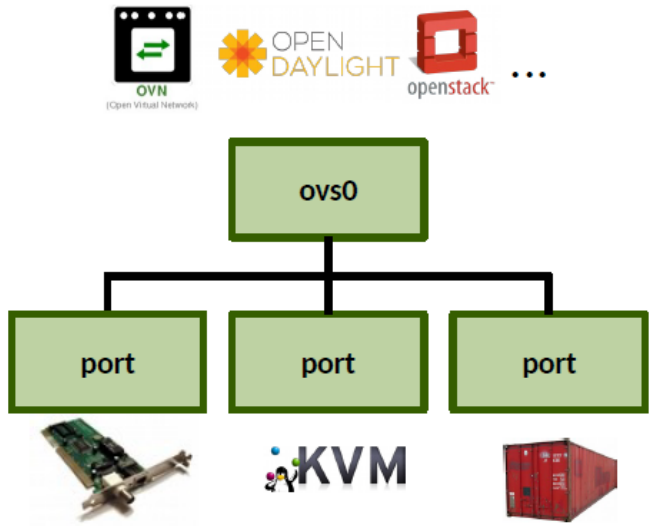
Bridge: Virtual Switch

- **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

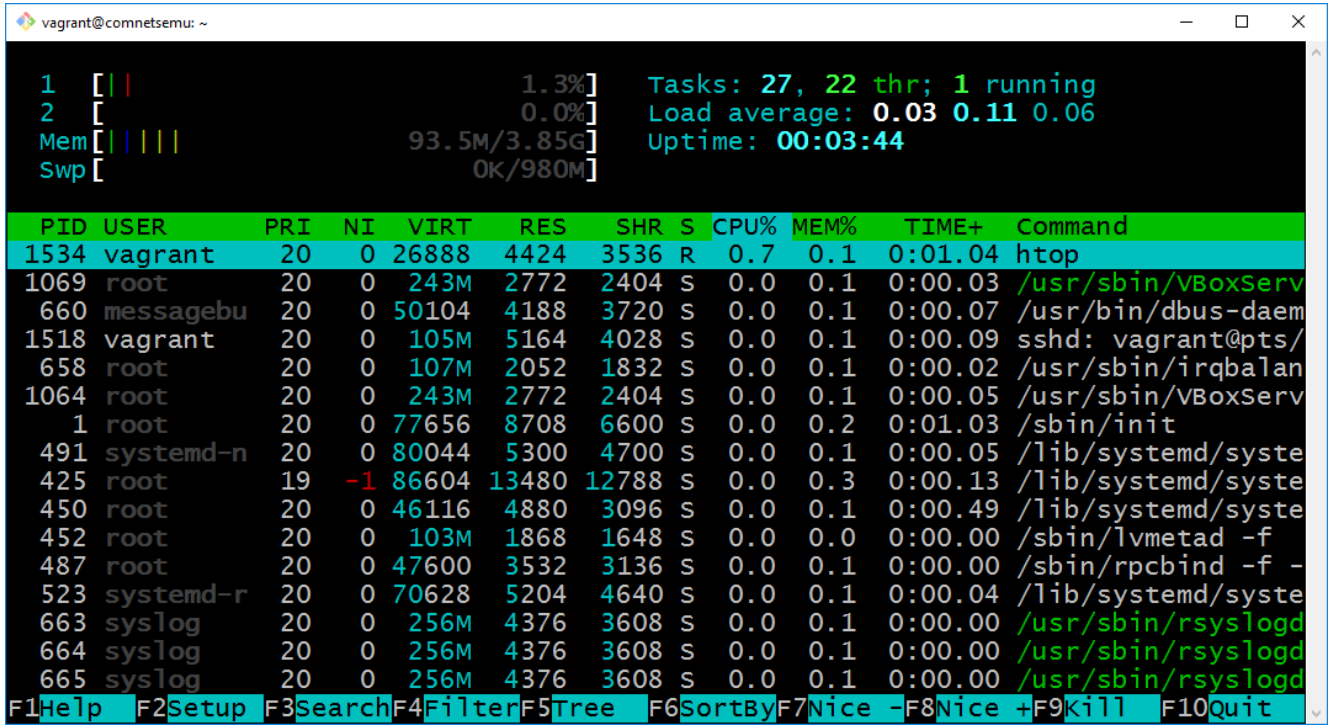
```
$ iperf3 -c 127.0.0.1 -t 3
Connecting to host 127.0.0.1, port 5201
[  4] local 127.0.0.1 port 50818 connected to 127.0.0.1 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  2.95 GBytes  25.3 Gbits/sec    0   3.18 MBytes
[  4]   1.00-2.00   sec  2.79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

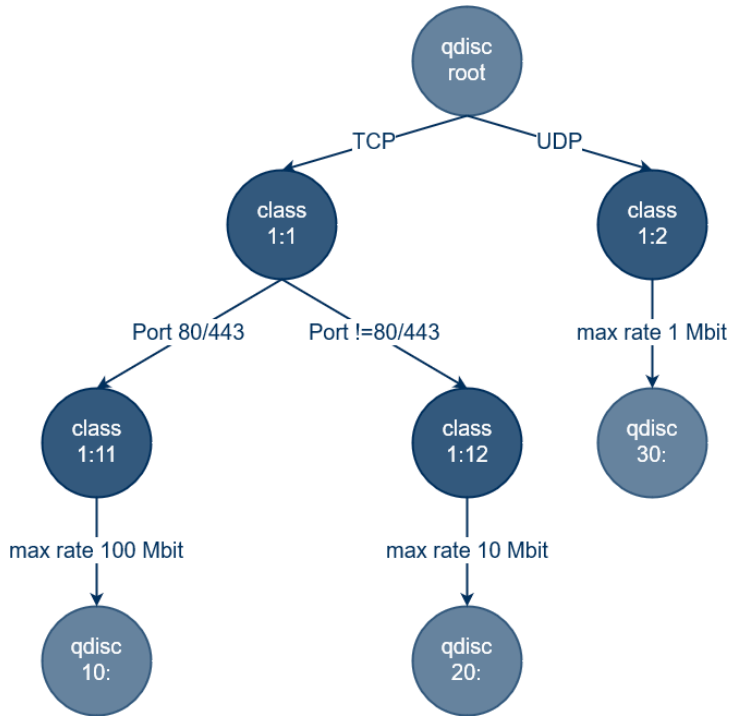


Screenshot of an example htop session

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

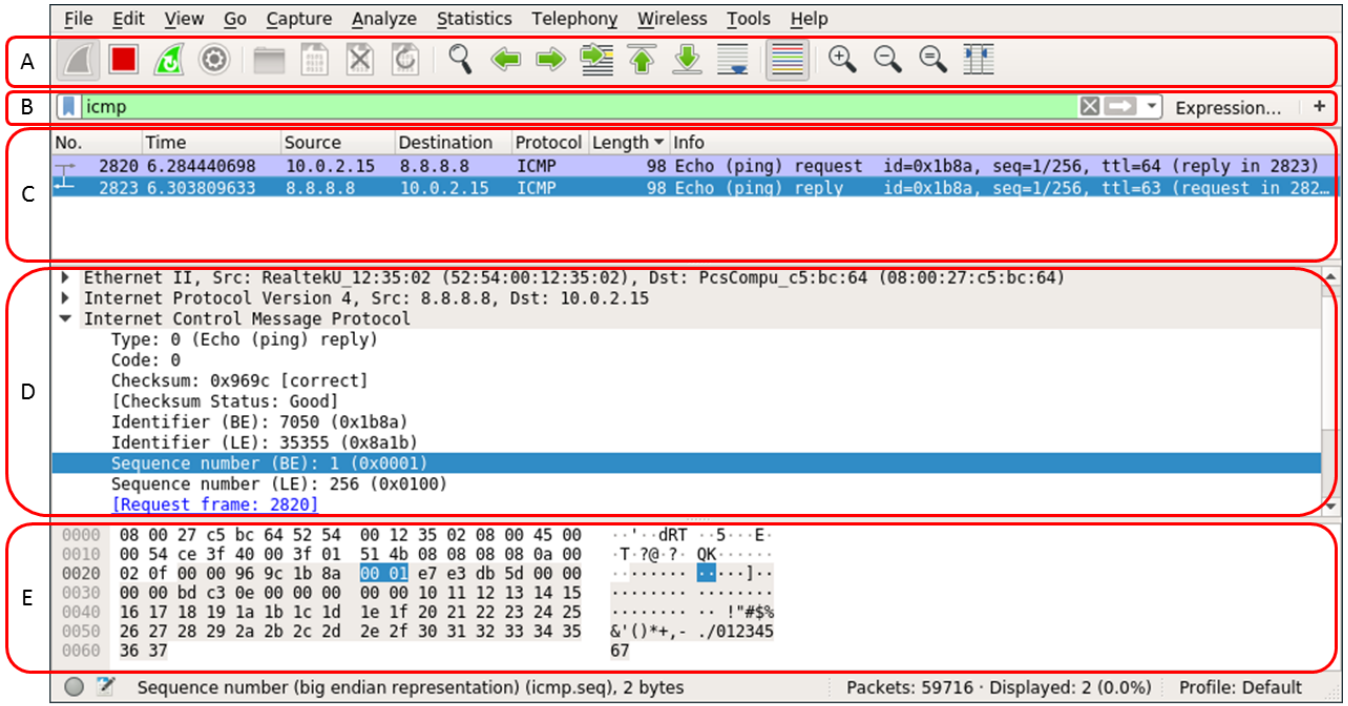
- A command line tool that provides insights into the network traffic on a packet level
- The packets may be filtered using a vast variety of options including typr of transport (TCP /UDP), port number, and IP address

```
$ 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

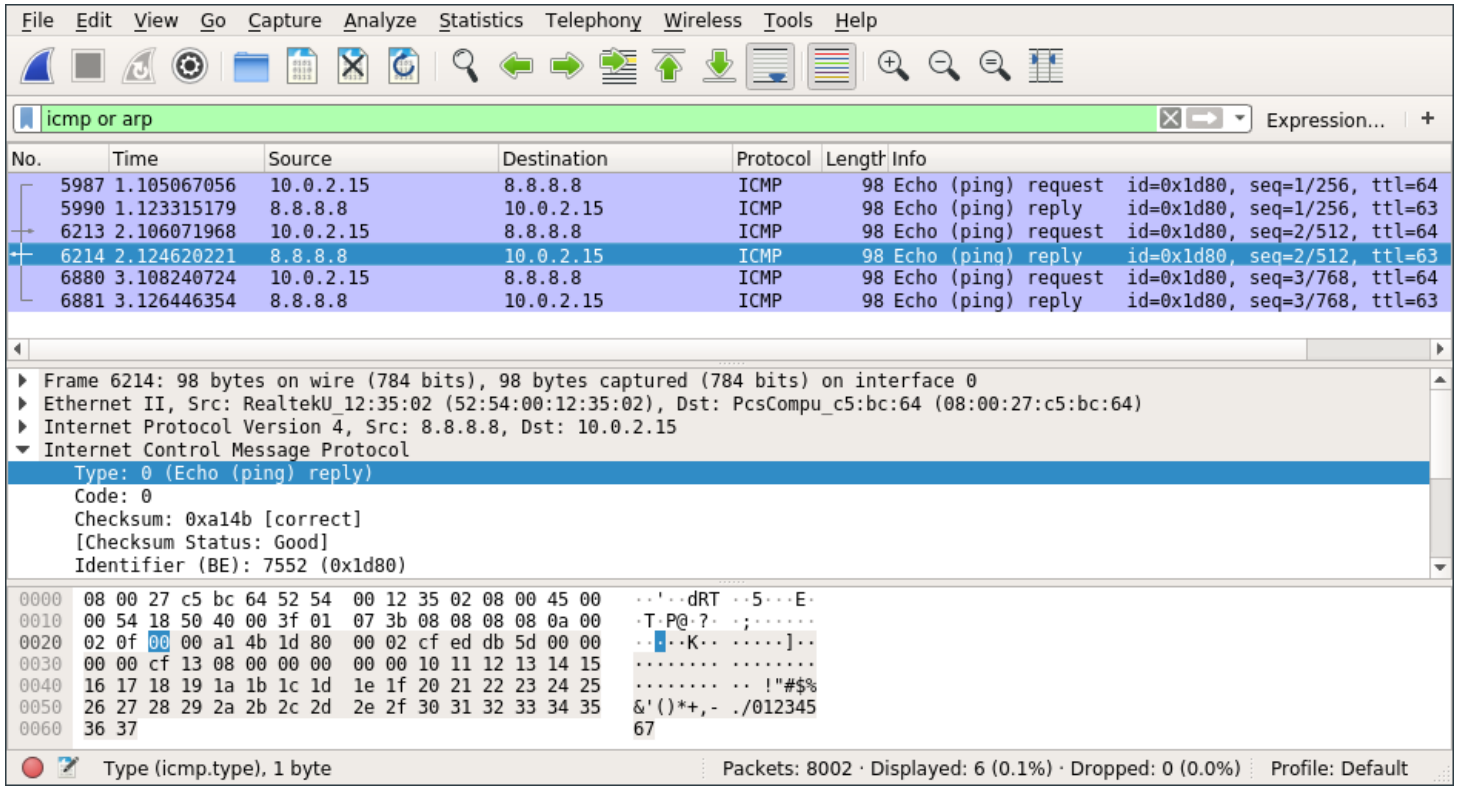
- Similar to tcpdump, but offers a user-friendly GUI
- Provides deeper network inspections using graphing tools and dynamic plots



GUI of Wireshark

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

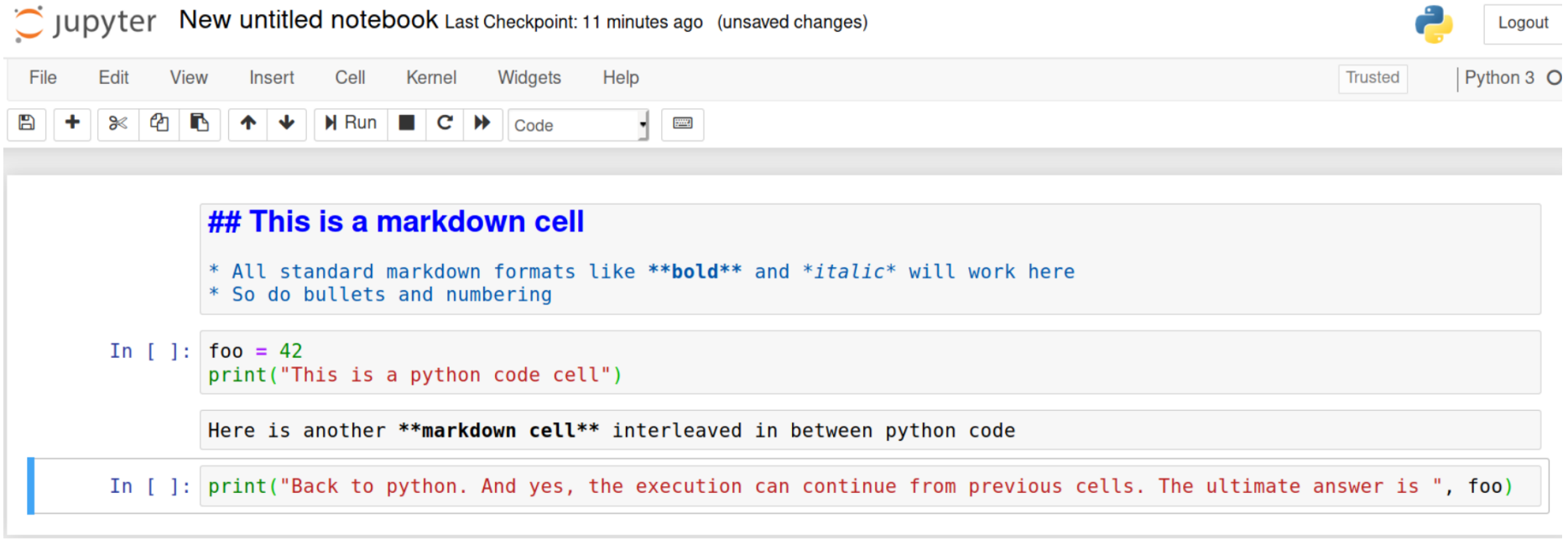
Wireshark



Example: Wireshark output of an ongoing ping

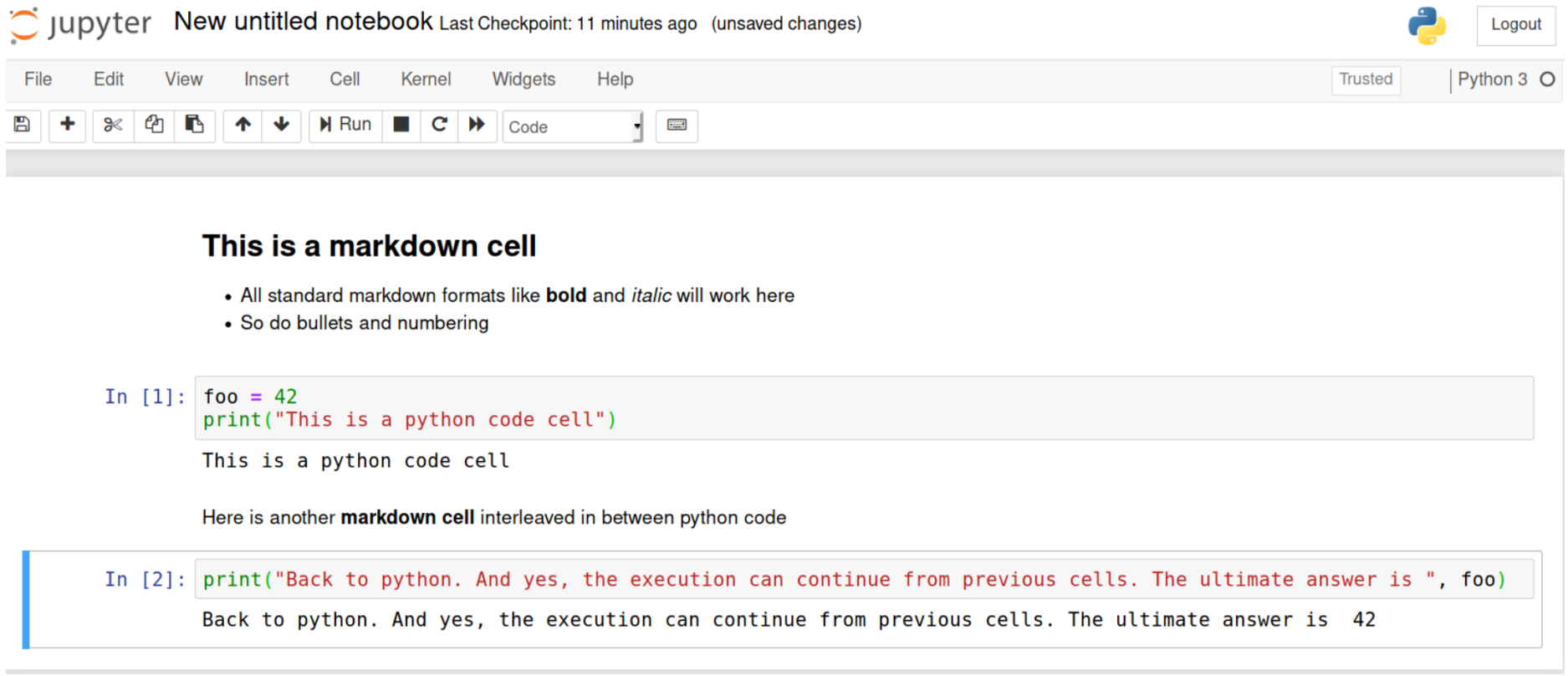
Jupyter

- 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



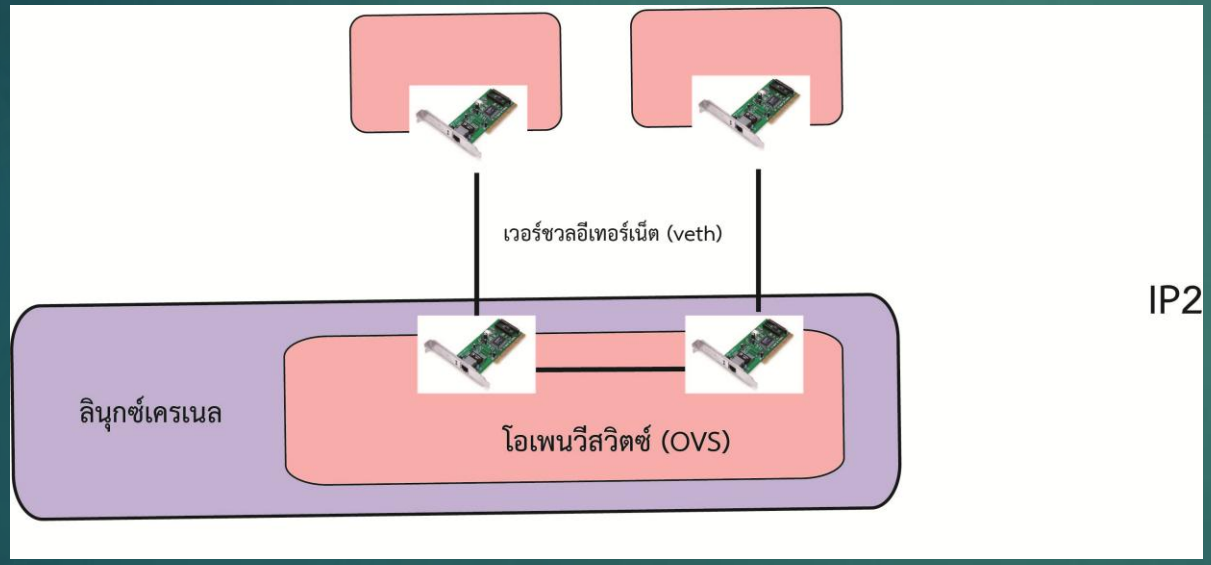
Example python and markdown (unexecuted)

Jupyter



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 พร้อมทั้งส่งผมที่ได้ออกไปยังอีก 1 file ที่มีเฉพาะ ID ตัวเอง พร้อมทั้งแสดงผลที่ได้ออกที่หน้าจอ
- 2. จากรูปที่ 4.17 ในหนังสือ ทำตามขั้นตอนที่ปรากฏ โดยใช้ class B address