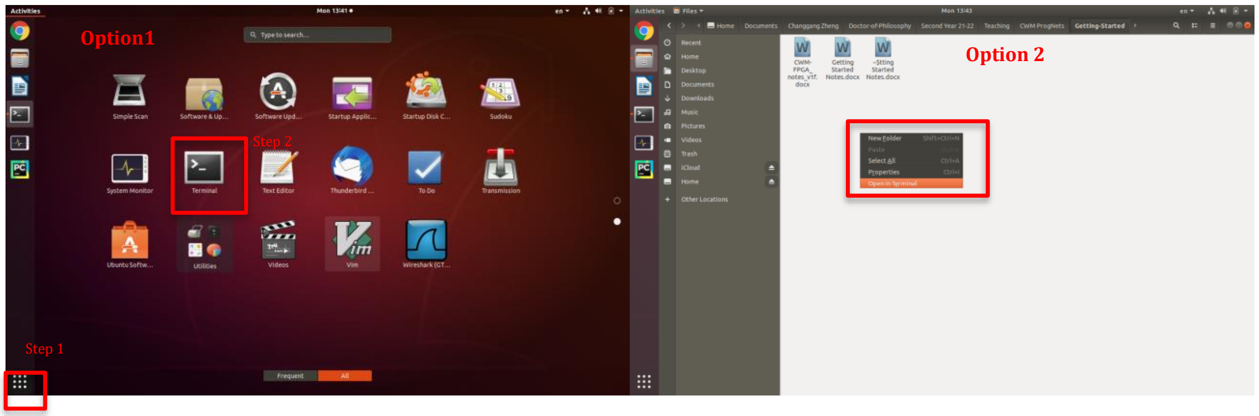
1 Working on Linux

The development environment used as part of this CWM is Linux-based[1]. Specifically, using an operating system called Ubuntu. The discussion of the operating system is beyond the scope of this CWM.

1.1 The Terminal

While Linux provides a graphic user interface that can be used for many purposes, during this CWM we will take advantage of the command line interface (CLI). To access the CLI, a *Terminal* program can be used. The terminal is largely similar to the command prompt on Window.

Go to Terminal:



Option 1: click “Show Application” ® click “Terminal”

Option 2: right-click in a folder blank space or in home page ® click “Open in Terminal”

While folders on Windows are usually located under C:\ (or disk drive with a similar notation), in Linux access to folders uses a different notation. When

[1] Linux is a family of Unix-like operating systems, based on the Linux operating system kernel

you open the terminal, by default (Option 1) it will open under your home directory, e.g., /home/sso1234/.

1.2 Common Commands

A few commands will have you work and navigate within and between directory:

- `ls` – lists the files in a folder
  - You can use options such as *-l* to get more information
- `pwd` – shows the current directory path
- `less` – view a file
- `cat` – print a file on screen
- `cd` – change directory, for example:
  - `cd CWM-ProgNets/assignment1/` – change directory to current directory's subdirectory called assignment1/CWM-ProgNets
  - `cd ..` – go one directory up
  - `cd ../..` – go two directories up
- `cp` – copy file, for example:
  - `cp send.py send_test.py` – create a copy of the file `send.py` called `send_test.py`
  - `cp ../assignment4/calc.p4 <.>` – copy the file `calc.p4` that is under the assignment4 directory to the current directory (not the "`<.>`" Indicating the current location)
- `mv` – move file, for example:
  - `mv send_test.py send_test2.p4` –rename the file `send_test.p4` to `send_test2.p4`

- *rm* – delete file (WARNING – there is no "undelete"
    - *rm send_test2.py* – delete the file send_test2.py
    - *rm -r* – delete subdirectory, e.g. *rm -r send_test2.py* will delete the subdirectory example
- *find* – find a file, for example:
    - *find -name send.py* – search for the file send.py in the current folder and all folders under it.
- *grep* – search for text within a file. The grep command is very useful when looking for errors, but has many options! These are a few:
    - *grep iface send.py -* search the word "iface" in the file send.py
    - *grep -r iface assignemnt1/* - search the word " iface " under the folder assignmemt1. *-r* indicates "recursive", looking in subdirectories.
    - *grep -c iface send.py -* count the number of times the word "iface" appears in the file send.py. *-c* is used for counting.
    - *grep -i iface send.py -* search the word "iface", while ignoring capitalization.
- *ps -* list of running user processes.
- *kill <process id> -* aborts a process with the given process id.
- *sudo -* run programs with security privileges.

The CLI supports using regular expressions using an asterisk, for *.* indicating "all file", *.txt indicating "all text files" or "ca*.p4" indicating all the P4(.p4) files starting with the prefix "ca". This can be used in conjunction with any of the commands listed above, e.g., *ls *.txt* will list all the text files.

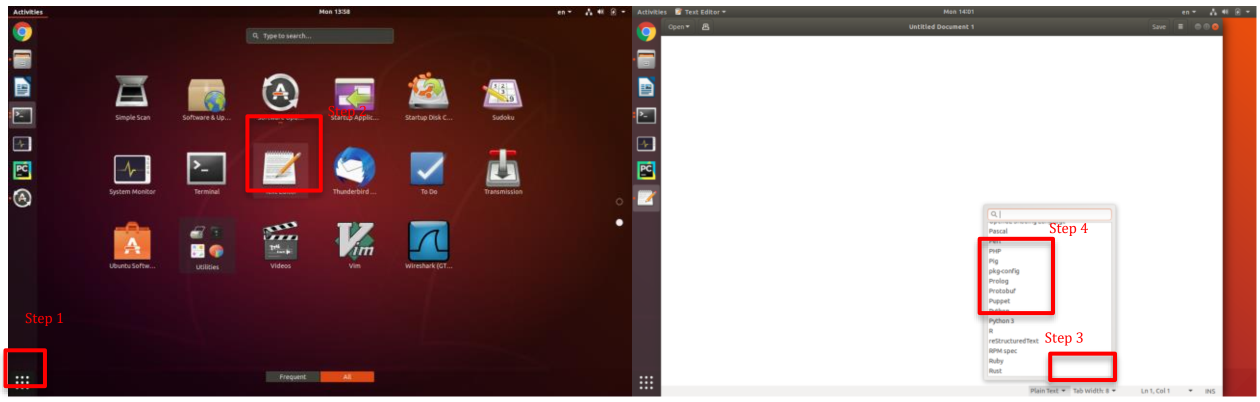
More commands and information are available at:
https://en.wikibooks.org/wiki/A_Quick_Introduction_to_Unix

1.3 Text Editors

There are several popular text editors that can be used on the development machines as part of the CWM. People with no previous Unix/Linux experience may find that gedit, a graphic editor, is the easiest to work with. It can be invoked from the terminal using the command:

```
gedit &
```

It can also be opened from the graphic user interface (GUI):



Option 1: click “Show Application” ® click “Terminal”

Language syntax highlighting can be enabled from the bottom bar, by changing Plain Text to Python/C. You can enable line numbers in a similar manner (bottom bar, right hand side button).

You can also use more classic unix text editors, such as `vi`, `vim`, `nano` or `emacs`. These text editors have no graphic interface.

1.4 Short cuts and special commands

Some keyboard combinations and special commands provide a better user experience:

- Ctrl+C – end the current program and return to prompt.
- Ctrl+R – search commands history. Start writing the command you are searching, and it will be auto-completed.
- Ctrl+Z – suspend a program (the program “sleeps” and can be continued)
    - `fg` – resume a suspended program in foreground
    - `bg` – resume a suspended program in background
- Tab – command or filename autocomplete
- `&` - when added at the end of a command, the command will run in the background (and not foreground).

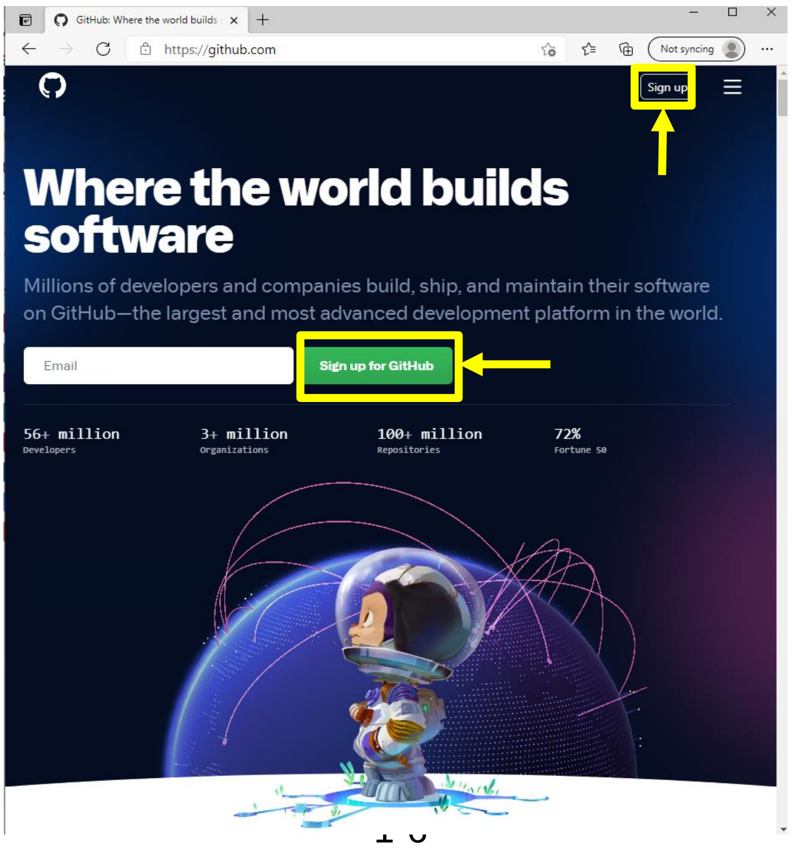
2 Version Control

Version control, also known as source control or source code management, refers to a class of systems that allow to save and manage changes to files. These files may vary in nature, from source code and documents to media files.

Saved versions are typically enumerated, and associated with a timestamp and the person making the change. Using versions, you can revert to previous versions in case of a bug, merge work from multiple contributors, easily track changes in a code, and more.

The version control system used in this CWM is Git, a free and open source version control software. Specifically, we will be hosting our version-controlled project on Github, a popular Git-based version control service (nowadays owned by Microsoft).

If you don't already have a Github user, you will need to create one. Sign up (free) at https://github.com:



Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

Version controlled projects are stored in *repositories.* A repository, and called a Git project, is the collection of files and folders associated with all of their history, and is a self contained unit. A repository may be public, accessible to everyone, or private, with access rights only to selected users.
Not all the files that are part of a project are version controlled (also called *tracked*). For example, generated files, such as log files and compilation outputs are often untracked. These files can be defined in a file called .gitignore.

2.1 Common Git Commands

While there are a lot of Git commands, the following ones will be the most useful ones:

- `git clone` – Create a local copy of a repository stored somewhere else. The repository will be created in a directory under the current directory. For example:
  - `git clone https://github.com/myuser/my-repo` - will create a local copy of the repository my-repo created by the user myuser. The copy will be placed under the folder my-repo.
  - `git clone https://github.com/myuser/my-repo my-folder` - will create a local copy of the repository my-repo created by the user myuser. The copy will be placed under the folder my-folder.
- `git pull` – update the local copy of a repository with all the changes made to the hosted (remote) repository. The command needs to be called from within the repository's folder, for example:

  -
    ```
    cd my-repo
    git pull
    ```

- `git status` – shows a list of all the files that were modified, staged for commit (update) or that are not tracked.
- `git add` – sets the list of changed local files that need to be updated in the remote repository. For example:
  - `git add result.txt`
- `git commit` – saves a snapshot of the files staged for update (using git add). A commit is usually accompanied by a comment describing the change, for example:
  - `git commit –m "A solution to exercise 1"`

  Note that a commit takes a *local* snapshot of the changes, and does not make any changes in the remote repository.
- `git push` – updates the remote repository with the commits made locally.
- `git diff` – shows the differences between the remote and the local project.
- `git merge` – merges two copies of a project, usually used to combine changes made by two users or in two branches of the project.

  During the CWM, it is best to avoid using merge.

See https://git-scm.com/docs for more commands.

2.2 Forking a Project

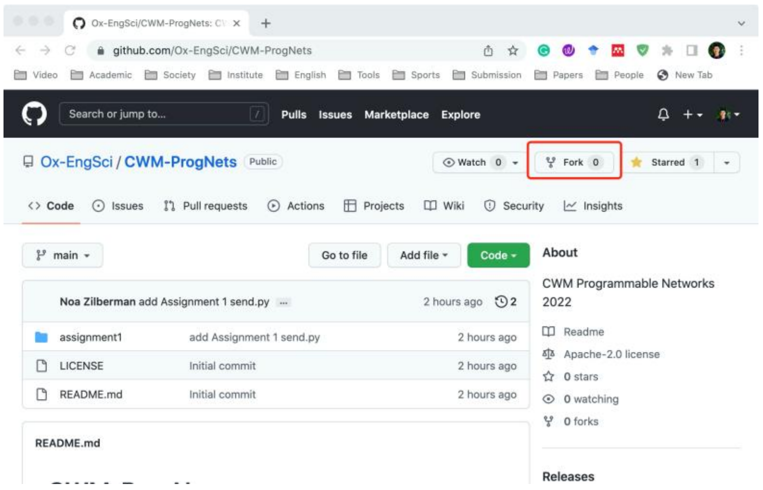
While users can create their own projects, a common use case it to start from an existing project, create a copy of the project under the user's personal account, and then make changes to the project. Creating such a copy of a project under a user's account is called *forking.*

Changes that are made under a forked project can be pushed back to the original project using a *pull request*. The pull request allows the owner of the original project to review the changes and decide whether to merge them or not into the original project.

As part of the CWM, you need to fork the module's repository to your own account. All the solutions to the exercises will be saved in this forked version of the project, and allow each student to work independently of others.

2.3 Getting started with your project

1. Fork the repository https://github.com/Ox-EngSci/CWM-ProgNets to your user:



github.com/Ox-EngSci/CWM-ProgNets
Ox-EngSci / CWM-ProgNets Public
Watch 0
Fork 0
Starred 1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
Go to file
Add file
Code
Noa Zilberman add Assignment 1 send.py
2 hours ago
assignment1 add Assignment 1 send.py 2 hours ago
LICENSE Initial commit 2 hours ago
README.md Initial commit 2 hours ago
README.md
About
CWM Programmable Networks 2022
Readme
Apache-2.0 license
0 stars
0 watching
0 forks
Releases

2. Open a terminal
3. Create a local copy of the repository:

   `git clone https://github.com/myuser/CWM-ProgNets` (make sure to change `myuser` to your Github username)

   `cd CWM-ProgNets`
4. Define the course's source repository as upstream:

   `git remote add upstream https://github.com/Ox-EngSci/CWM-ProgNets.git`
5. Fetch any updates from the main repository:

   `git fetch upstream`

6. Check out your fork's local master branch:

   ```
   git checkout master
   ```

7. Merge changes from main repository (upstream/master) into your local master branch (you will not lose local changes):

   ```
   git merge upstream/master
   ```

8. You are ready to start Assignment 1!

2.4 Saving your work

You should back up your work frequently. It is recommended to do so every time you complete writing a module and when you fix a bug. As a rule of thumb, make sure to push your updates to the repository at least twice a day. To push your updates to the cloud:

1. Open a terminal
2. Change the working directory to your CWM directory:

   ```
   cd CWM-ProgNets
   ```

3. List the changes to the github repository since the last update:

   ```
   git status
   ```

4. From the list of changed files (listed by the previous step), Select the files that need to be pushed to the repository. For example:

   ```
   git add assignment1/result.txt assignment1/switch.p4
   assignment1/report.pdf
   ```

   You should only save source (code) and report files, and not any generated outputs. Example file formats are *.py, *.p4, *.pdf.
   Files that should be ignored can be defined in the *.gitignore* file.

5. Commit your changes, and add a message describing them, for example:

   ```
   git commit –m “A complete solution to Assignment1”.
   ```

6. Push your update to the repository:

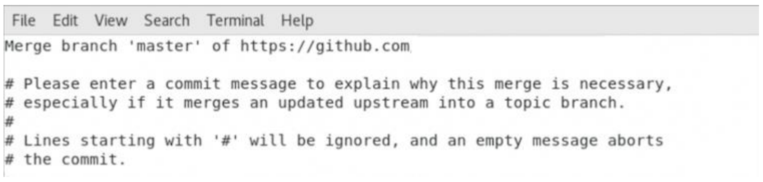
   `git push`

   You may be prompted to enter your github username and password.

2.5 Handling Merge Conflicts

Merge conflicts happen when trying to update a file in a local/remote repository, and both the local and the remote copies of the files include changes. The best way to handle merge conflicts is to avoid them, by making sure changes are made only in one place at a time.

If an automatic merge is successful (changes are not conflicted), you may get the following screen:



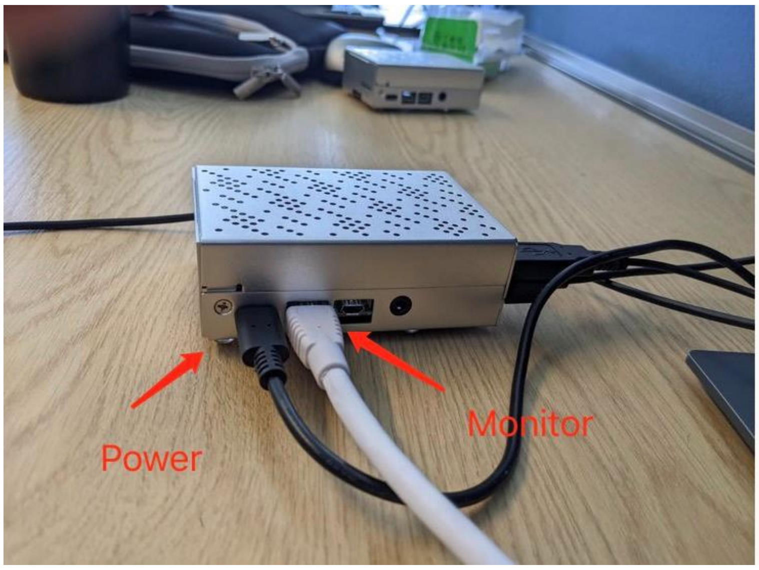
```
Merge branch 'master' of https://github.com

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
```

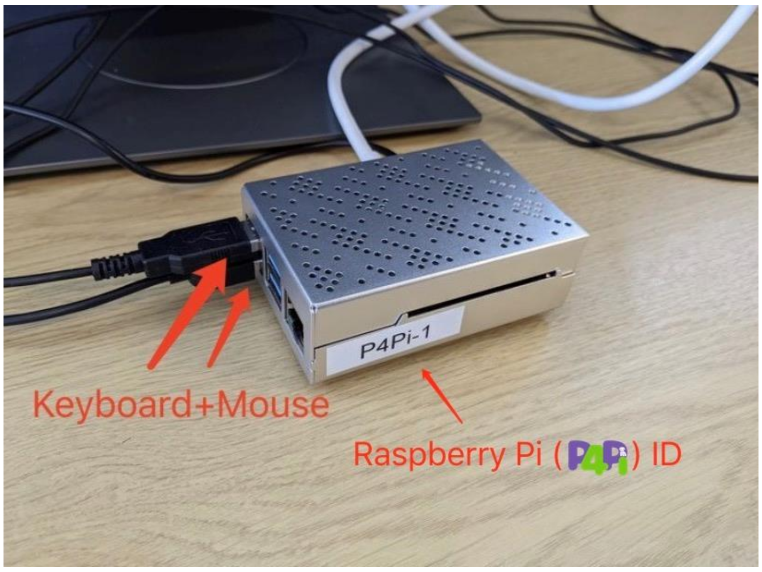
Type `:q` to exit the window. You will need to push the merged changes back to the repository (`git push`).

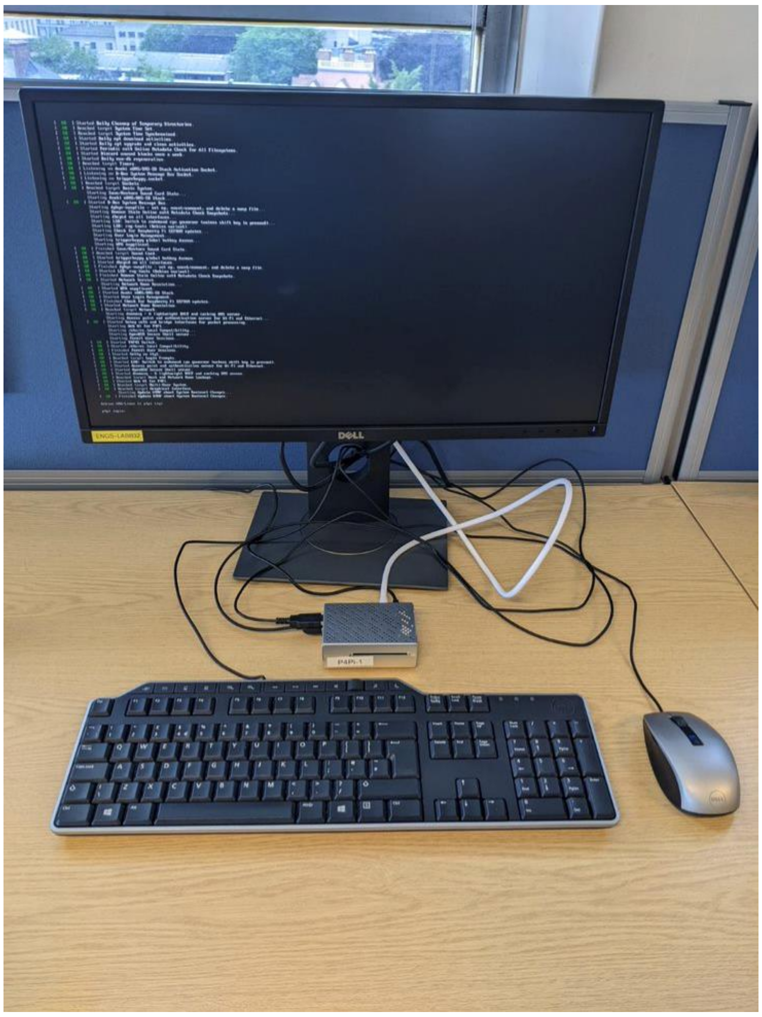
Detailed instructions for resolving merge conflicts are provided here (choose your operating system).

3 Connecting to Raspberry Pi

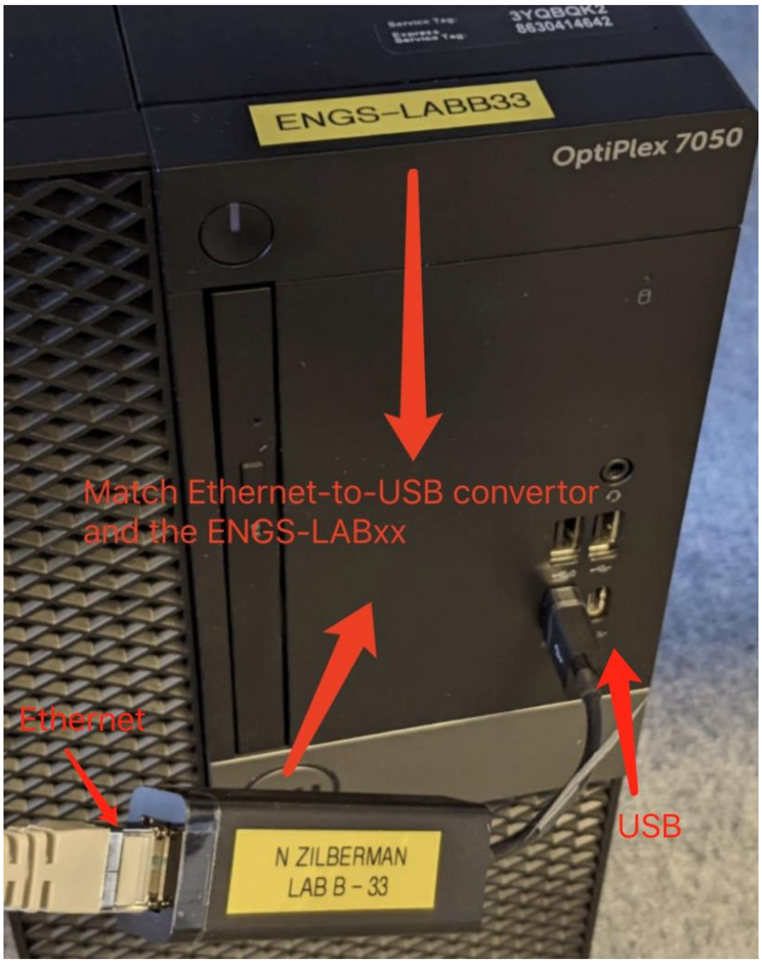
1. Check that monitor, mouse, keyboard, and power supply are connected.



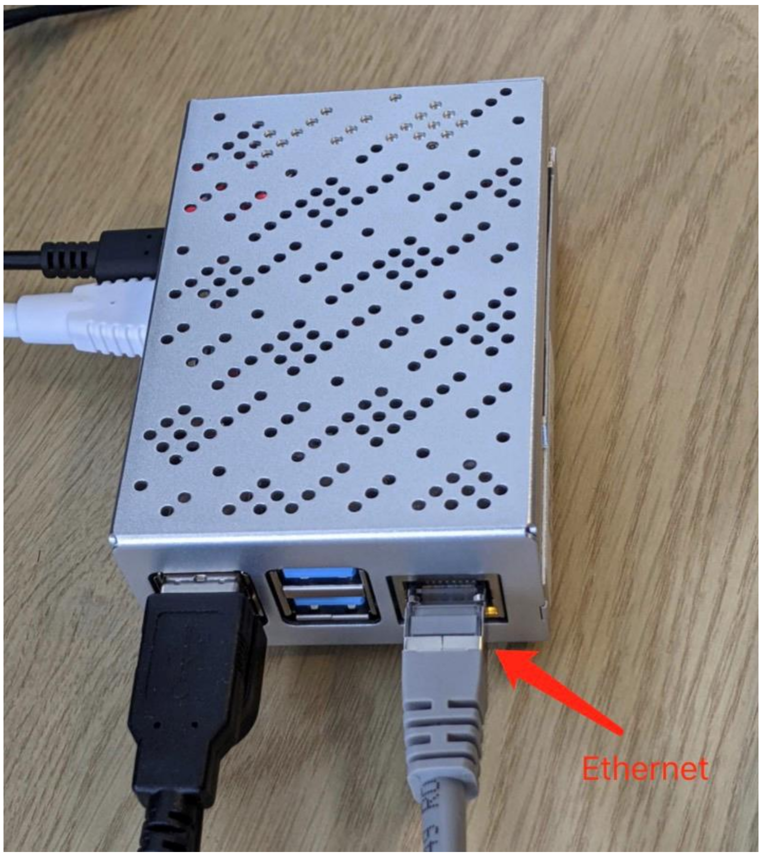




2. Use the ethernet cable to connect Pi and ENGS-LABxx. Please make sure to use the correct Ethernet to USB converter before connecting to ENGS-LABxx.

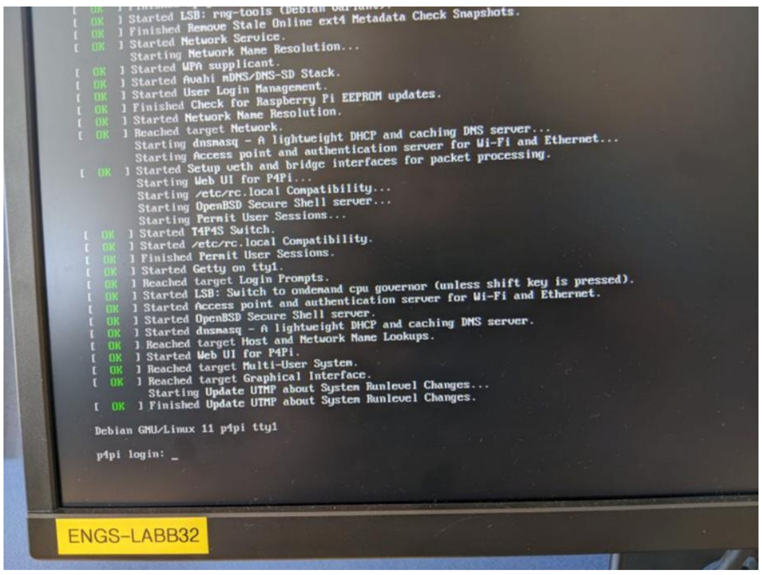


3YQBQK2
8630414642
ENGS–LABB33
OptiPlex 7050
Match Ethernet-to-USB convertor and the ENGS-LABxx
Ethernet
N ZILBERMAN LAB B – 33
USB



Ethernet

3. After all the cables are connected, insert the password and start Pi (P4Pi).



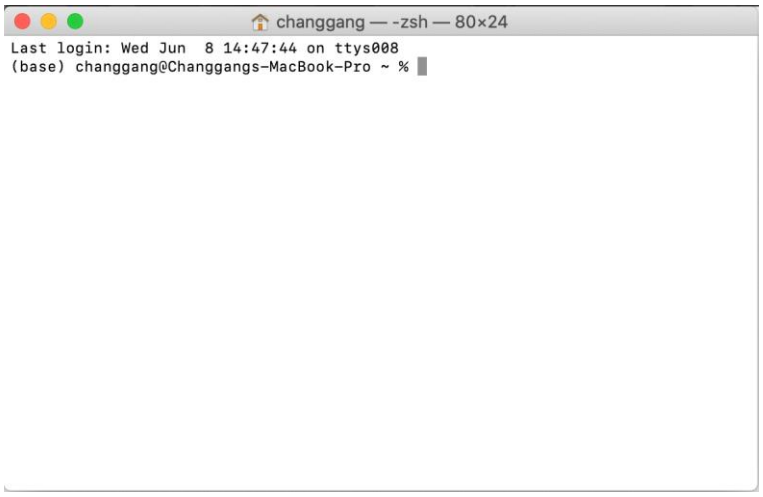
```
[  OK  ] Started LSB: rng-tools (Debian variant).
[  OK  ] Finished Remove Stale Online ext4 Metadata Check Snapshots.
[  OK  ] Started Network Service.
         Starting Network Name Resolution...
[  OK  ] Started WPA supplicant.
[  OK  ] Started Avahi mDNS/DNS-SD Stack.
[  OK  ] Started User Login Management.
[  OK  ] Finished Check for Raspberry Pi EEPROM updates.
[  OK  ] Started Network Name Resolution.
[  OK  ] Reached target Network.
         Starting dnsmasq - A lightweight DHCP and caching DNS server...
         Starting Access point and authentication server for Wi-Fi and Ethernet...
         Starting Setup veth and bridge interfaces for packet processing.
[  OK  ] Started Setup veth and bridge interfaces for packet processing.
         Starting Web UI for P4Pi...
         Starting /etc/rc.local Compatibility...
         Starting OpenBSD Secure Shell server...
         Starting Permit User Sessions...
[  OK  ] Started T4P4S Switch.
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Finished Permit User Sessions.
[  OK  ] Started Getty on tty1.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started LSB: Switch to ondemand cpu governor (unless shift key is pressed).
[  OK  ] Started Access point and authentication server for Wi-Fi and Ethernet.
[  OK  ] Started OpenBSD Secure Shell server.
[  OK  ] Started dnsmasq - A lightweight DHCP and caching DNS server.
[  OK  ] Reached target Host and Network Name Lookups.
[  OK  ] Started Web UI for P4Pi.
[  OK  ] Reached target Multi-User System.
[  OK  ] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Finished Update UTMP about System Runlevel Changes.

Debian GNU/Linux 11 p4pi tty1

p4pi login: _
```

User: pi

Password: ProgNets

4 Working on Raspberry Pi

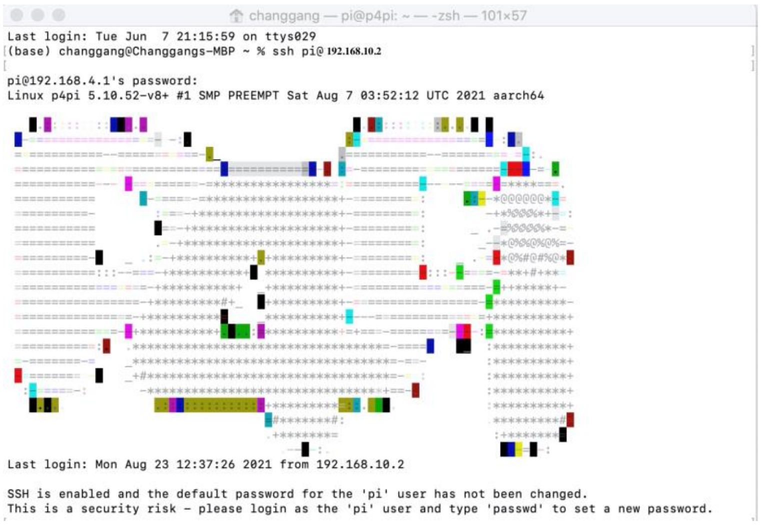
1. Open a terminal



```
Last login: Wed Jun  8 14:47:44 on ttys008
(base) changgang@Changgangs-MacBook-Pro ~ %
```

2. Connect to Pi by using *ssh* command

   ```
   ssh pi@192.168.10.2
   ```

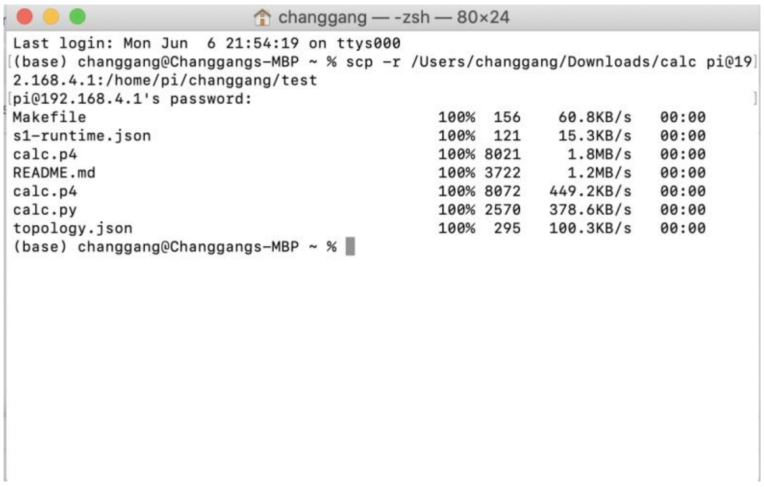


```
Last login: Tue Jun  7 21:15:59 on ttys029
(base) changgang@Changgangs-MBP ~ % ssh pi@ 192.168.10.2

pi@192.168.4.1's password:
Linux p4pi 5.10.52-v8+ #1 SMP PREEMPT Sat Aug 7 03:52:12 UTC 2021 aarch64

Last login: Mon Aug 23 12:37:26 2021 from 192.168.10.2

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.
```

   Password: ProgNets

3. Copy the repository to Pi by using *scp* command from ENGS-LABxx.

   ```
   scp -r <local file> pi@<remote location>:<remote
   folder>
   ```

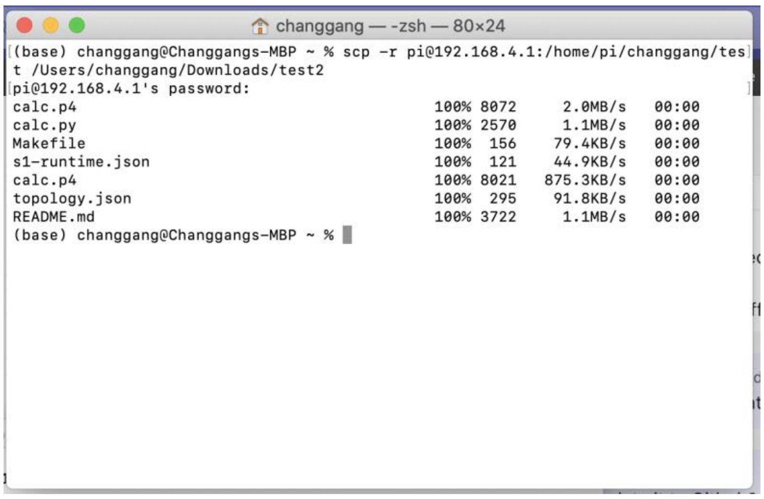


```
changgang — -zsh — 80×24
Last login: Mon Jun  6 21:54:19 on ttys000
(base) changgang@Changgangs-MBP ~ % scp -r /Users/changgang/Downloads/calc pi@19
2.168.4.1:/home/pi/changgang/test
pi@192.168.4.1's password:
Makefile                                      100%  156    60.8KB/s   00:00
s1-runtime.json                               100%  121    15.3KB/s   00:00
calc.p4                                       100% 8021     1.8MB/s   00:00
README.md                                     100% 3722     1.2MB/s   00:00
calc.p4                                       100% 8072   449.2KB/s   00:00
calc.py                                       100% 2570   378.6KB/s   00:00
topology.json                                 100%  295   100.3KB/s   00:00
(base) changgang@Changgangs-MBP ~ %
```

(The example uses the wireless connection, where IP is 192.168.4.1. In the lab, please use IP 192.168.10.2)

4. Copy the results back from Pi.

   scp -r pi@<remote location>:<remote folder> <local location>



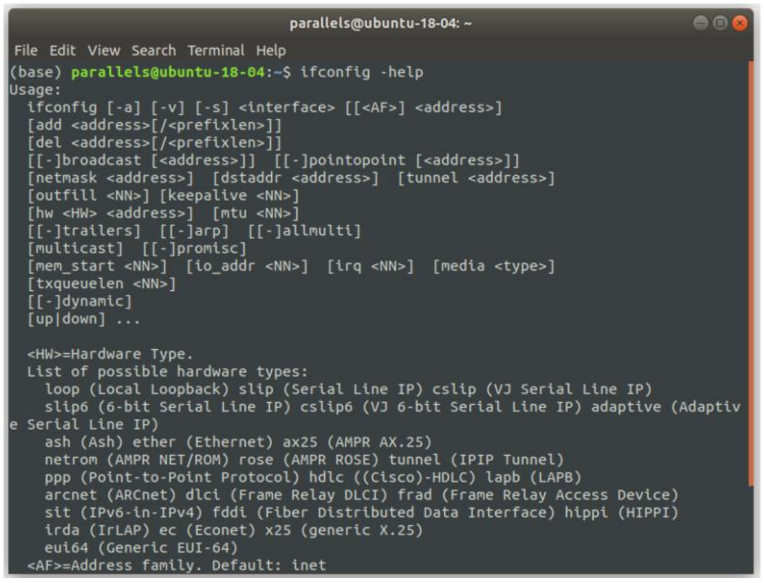
```
changgang — -zsh — 80×24
(base) changgang@Changgangs-MBP ~ % scp -r pi@192.168.4.1:/home/pi/changgang/tes
t /Users/changgang/Downloads/test2
pi@192.168.4.1's password:
calc.p4                                       100% 8072     2.0MB/s   00:00
calc.py                                       100% 2570     1.1MB/s   00:00
Makefile                                      100%  156    79.4KB/s   00:00
s1-runtime.json                               100%  121    44.9KB/s   00:00
calc.p4                                       100% 8021   875.3KB/s   00:00
topology.json                                 100%  295    91.8KB/s   00:00
README.md                                     100% 3722     1.1MB/s   00:00
(base) changgang@Changgangs-MBP ~ %
```

(The example uses the wireless connection, where IP is 192.168.4.1. In the lab, please use IP 192.168.10.2)

5 Exercises on Raspberry Pi

1. Network Interface Configuration:

   `ifconfig` (short for interface config) is a system administration utility in Unix-like operating systems for network interface configuration. Link: https://en.wikipedia.org/wiki/Ifconfig



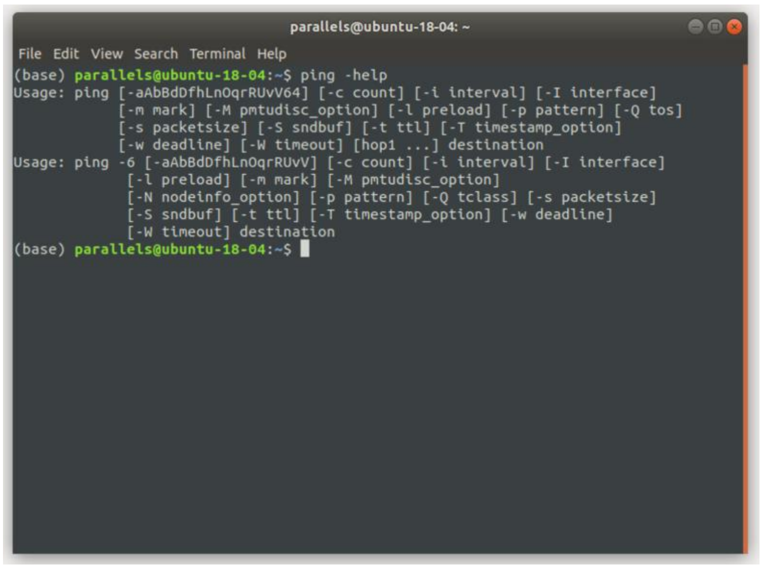
```
parallels@ubuntu-18-04: ~
File  Edit  View  Search  Terminal  Help
(base) parallels@ubuntu-18-04:~$ ifconfig -help
Usage:
  ifconfig [-a] [-v] [-s] <interface> [[<AF>] <address>]
  [add <address>[/<prefixlen>]]
  [del <address>[/<prefixlen>]]
  [[-]broadcast [<address>]]  [[-]pointopoint [<address>]]
  [netmask <address>]  [dstaddr <address>]  [tunnel <address>]
  [outfill <NN>] [keepalive <NN>]
  [hw <HW> <address>]  [mtu <NN>]
  [[-]trailers]  [[-]arp]  [[-]allmulti]
  [multicast]  [[-]promisc]
  [mem_start <NN>]  [io_addr <NN>]  [irq <NN>]  [media <type>]
  [txqueuelen <NN>]
  [[-]dynamic]
  [up|down] ...

  <HW>=Hardware Type.
  List of possible hardware types:
    loop (Local Loopback) slip (Serial Line IP) cslip (VJ Serial Line IP)
    slip6 (6-bit Serial Line IP) cslip6 (VJ 6-bit Serial Line IP) adaptive (Adaptiv
e Serial Line IP)
    ash (Ash) ether (Ethernet) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) rose (AMPR ROSE) tunnel (IPIP Tunnel)
    ppp (Point-to-Point Protocol) hdlc ((Cisco)-HDLC) lapb (LAPB)
    arcnet (ARCnet) dlci (Frame Relay DLCI) frad (Frame Relay Access Device)
    sit (IPv6-in-IPv4) fddi (Fiber Distributed Data Interface) hippi (HIPPI)
    irda (IrLAP) ec (Econet) x25 (generic X.25)
    eui64 (Generic EUI-64)
  <AF>=Address family. Default: inet
```

Try the following commands:

```
ifconfig veth0 down
ifconfig veth0 up
ifconfig veth 192.168.4.4
```

2. Ping Test:

   `Ping` is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network. Link: https://en.wikipedia.org/wiki/Ping_(networking_utility)



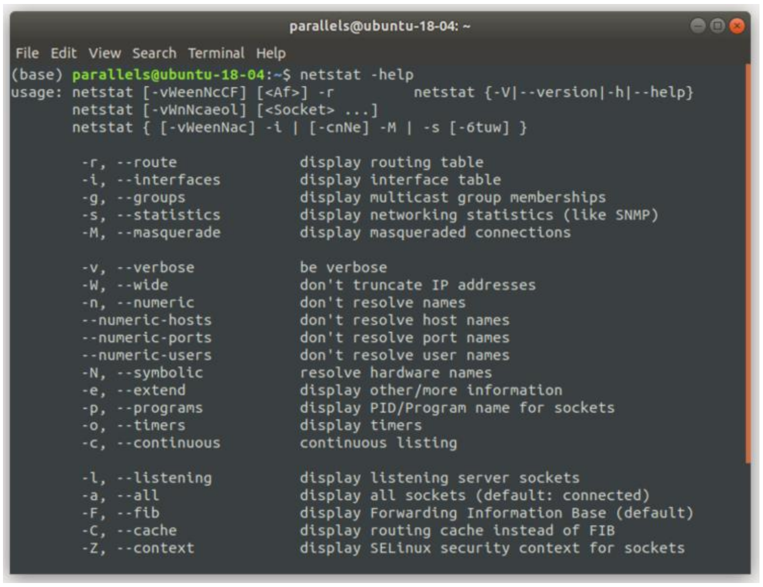
```
(base) parallels@ubuntu-18-04:~$ ping -help
Usage: ping [-aAbBdDfhLnOqrRUvV64] [-c count] [-i interval] [-I interface]
            [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
            [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
            [-w deadline] [-W timeout] [hop1 ...] destination
Usage: ping -6 [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
             [-l preload] [-m mark] [-M pmtudisc_option]
             [-N nodeinfo_option] [-p pattern] [-Q tclass] [-s packetsize]
             [-S sndbuf] [-t ttl] [-T timestamp_option] [-w deadline]
             [-W timeout] destination
(base) parallels@ubuntu-18-04:~$
```

From the Raspberry Pi try the following command:

`Ping 192.168.10.1`

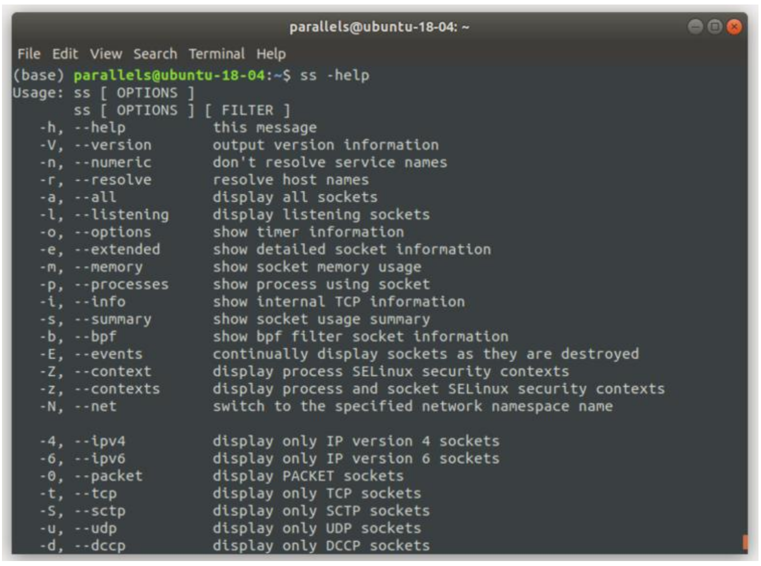
3. Display Network Statistics:

`netstat` (network statistics) is a command-line network utility that displays network connections. Link: https://en.wikipedia.org/wiki/Netstat



```
(base) parallels@ubuntu-18-04:~$ netstat -help
usage: netstat [-vWeenNcCF] [<Af>] -r         netstat {-V|--version|-h|--help}
       netstat [-vWnNcaeol] [<Socket> ...]
       netstat { [-vWeenNac] -i | [-cnNe] -M | -s [-6tuw] }

        -r, --route              display routing table
        -i, --interfaces         display interface table
        -g, --groups             display multicast group memberships
        -s, --statistics         display networking statistics (like SNMP)
        -M, --masquerade         display masqueraded connections

        -v, --verbose            be verbose
        -W, --wide               don't truncate IP addresses
        -n, --numeric            don't resolve names
        --numeric-hosts          don't resolve host names
        --numeric-ports          don't resolve port names
        --numeric-users          don't resolve user names
        -N, --symbolic           resolve hardware names
        -e, --extend             display other/more information
        -p, --programs           display PID/Program name for sockets
        -o, --timers             display timers
        -c, --continuous         continuous listing

        -l, --listening          display listening server sockets
        -a, --all                display all sockets (default: connected)
        -F, --fib                display Forwarding Information Base (default)
        -C, --cache              display routing cache instead of FIB
        -Z, --context            display SELinux security context for sockets
```

The command “`ss`” has similar functionality.


parallels@ubuntu-18-04: ~
File Edit View Search Terminal Help

```
(base) parallels@ubuntu-18-04:~$ ss -help
Usage: ss [ OPTIONS ]
       ss [ OPTIONS ] [ FILTER ]
   -h, --help          this message
   -V, --version       output version information
   -n, --numeric       don't resolve service names
   -r, --resolve       resolve host names
   -a, --all           display all sockets
   -l, --listening     display listening sockets
   -o, --options       show timer information
   -e, --extended      show detailed socket information
   -m, --memory        show socket memory usage
   -p, --processes     show process using socket
   -i, --info          show internal TCP information
   -s, --summary       show socket usage summary
   -b, --bpf           show bpf filter socket information
   -E, --events        continually display sockets as they are destroyed
   -Z, --context       display process SELinux security contexts
   -z, --contexts      display process and socket SELinux security contexts
   -N, --net           switch to the specified network namespace name

   -4, --ipv4          display only IP version 4 sockets
   -6, --ipv6          display only IP version 6 sockets
   -0, --packet        display PACKET sockets
   -t, --tcp           display only TCP sockets
   -S, --sctp          display only SCTP sockets
   -u, --udp           display only UDP sockets
   -d, --dccp          display only DCCP sockets
```

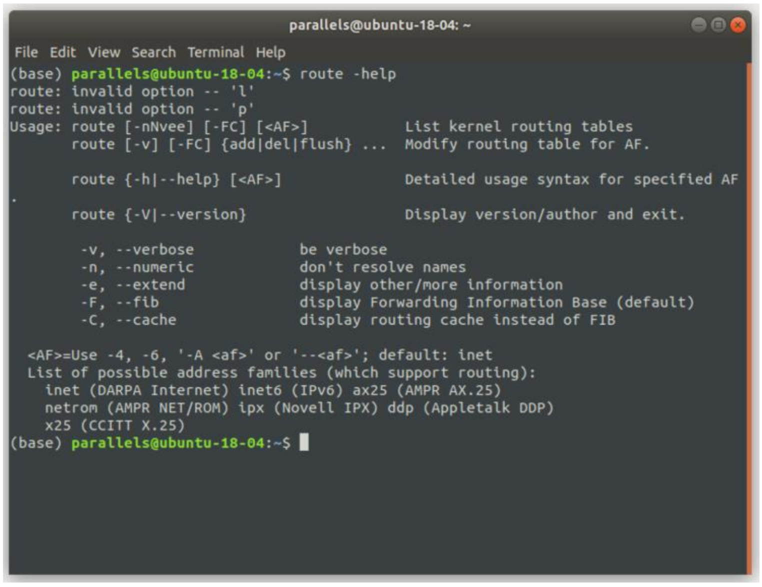
Try the following commands:

```
netstat -rn
```

4. Display Routing Table:

`Route` is a command used to view and manipulate the IP routing table.

Link: https://en.wikipedia.org/wiki/Route_(command)


parallels@ubuntu-18-04: ~
File Edit View Search Terminal Help

```
(base) parallels@ubuntu-18-04:~$ route -help
route: invalid option -- 'l'
route: invalid option -- 'p'
Usage: route [-nNvee] [-FC] [<AF>]           List kernel routing tables
       route [-v] [-FC] {add|del|flush} ...  Modify routing table for AF.

       route {-h|--help} [<AF>]              Detailed usage syntax for specified AF
.
       route {-V|--version}                  Display version/author and exit.

        -v, --verbose            be verbose
        -n, --numeric            don't resolve names
        -e, --extend             display other/more information
        -F, --fib                display Forwarding Information Base (default)
        -C, --cache              display routing cache instead of FIB

  <AF>=Use -4, -6, '-A <af>' or '--<af>'; default: inet
  List of possible address families (which support routing):
    inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)
    x25 (CCITT X.25)
(base) parallels@ubuntu-18-04:~$ 
```

Try the following commands:

```
route -n
arp -a
```

5. Practice exercise:

1. Disable eth0 on P4Pi, then, enable it (not while using ssh!).
2. Show all TCP and UDP connections status by using “netstat”.
3. Show the routing table.
4. Check that the connection between P4Pi and the lab machine works using ping.