PRE-LAB

1. What command will show you which groups you are a member of?
The command ‘groups’ will.

2. What does the environmental variable “$?” hold? (Hint: the command ‘echo $?’ will show you this on your screen)
It holds the exit status of the last executed command.

3. What key combo will suspend a running process and place it as a background process?
‘CTRL+Z bg’ will do so.

4. With what command/arguments can you find out your kernel version and the “nodename”?
Use the command ‘uname’ with ‘-r’ for kernel version and ‘-n’ for nodename.

5. What is the difference between the paths “.”, “..”, and “~”? What does the path “/” refer to when not preceded by anything?
‘.’ refers to the current directory, ‘..’ refers to the parent directory, ‘~’ refers to the home directory, and ‘/’ refers to the root directory.

6. What is a pid? Which command would you use to find the “pid” for a running process?
‘pid’ is the process identifier. The command ‘ps’ will show the pid of a running process.

7. Write a single command that will return every user’s default shell.
“awk -F: '{print $1, $NF}' /etc/passwd” will do so.

8. What is the difference between “sudo” and “su root”?
‘sudo’ allows a permitted user to execute a command as the superuser or another user, while su root switches to the root user's environment, requiring the root password.

9. How would you tell your computer to run a program or script on a schedule or set interval on Linux? E.g. Run this program once every 30 minutes.
You would use crontab. Run ‘crontab -e’ to edit the crontab file, and add “*/30 * * * * /path/to/your/program” to your file to run every 30 minutes.

10. Write a shell script that only prints even numbered lines of files in the current directory. The output should be filename: line. You do not need to print line numbers.

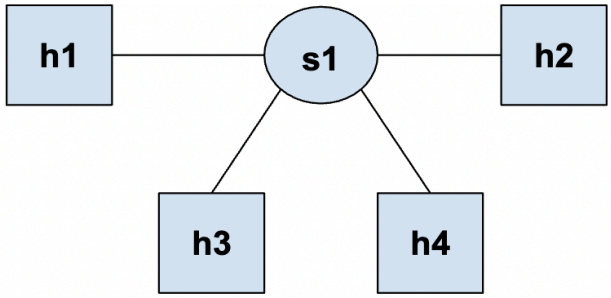
```
#!/bin/bash
for file in *; do
    if [ -f "$file" ]; then
        echo "File: $file"
        awk 'NR % 2 == 0 {print FILENAME": "$0}' "$file"
    fi
done
```

LAB

1.

```python
#!/usr/bin/python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.cli import CLI
class MyTopology(Topo):
  def __init__(self):
    Topo.__init__(self)
    switch = self.addSwitch('s1')
    host1 = self.addHost('h1')
    host2 = self.addHost('h2')
    host3 = self.addHost('h3')
    host4 = self.addHost('h4')
    self.addLink(host1, switch)
    self.addLink(host2, switch)
    self.addLink(host3, switch)
    self.addLink(host4, switch)
if __name__ == '__main__':
  topo = MyTopology()
  net = Mininet( topo=topo )
  net.start()
  CLI(net)
  net.stop()
```

We create the network topology below with this python script. We create a switch (s1) and 4 hosts (h1, h2, h3, h4) and link each host to the switch. We call our user defined topology and pass it into Mininet. We start the Mininet, call CLI to open the command line interface, and have a stop call following that.

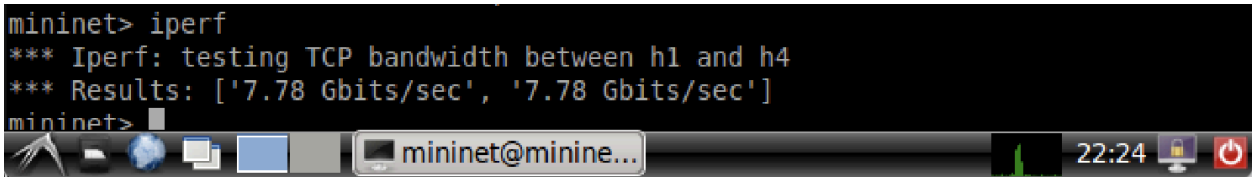


2.

```
mininet@mininet-vm:~/Desktop$ sudo ./cadenroberts-topo.py
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1678>
<Host h2: h2-eth0:10.0.0.2 pid=1682>
<Host h3: h3-eth0:10.0.0.3 pid=1684>
<Host h4: h4-eth0:10.0.0.4 pid=1686>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None
pid=1691>
<Controller c0: 127.0.0.1:6633 pid=1671>
```

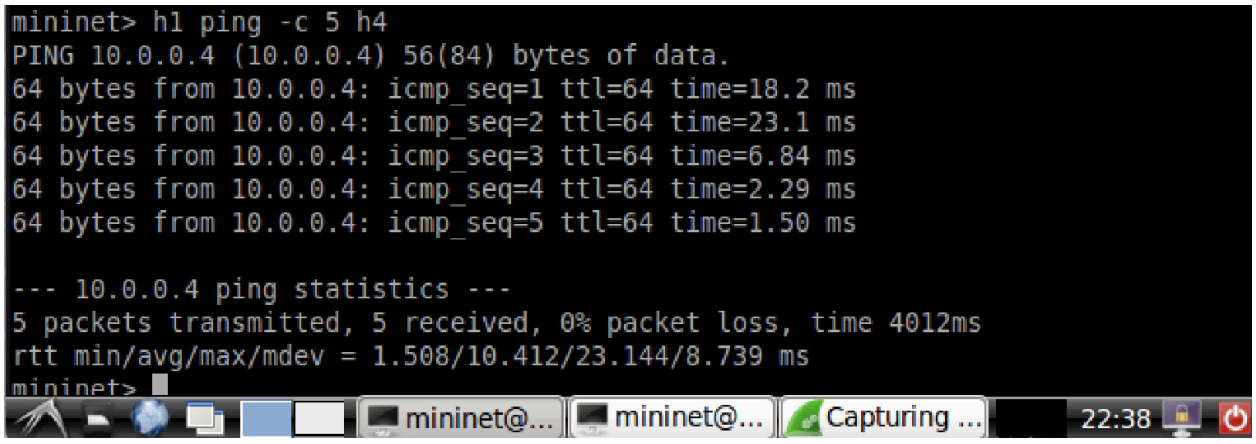
Pingall shows each host pinging all other hosts through the switch. The results show no drops and 12/12 pings received. Dump shows us what IP address and name is assigned to each host, along with the pid for each host. The pid, name, and ip for the switch and controller is also shown.

3.

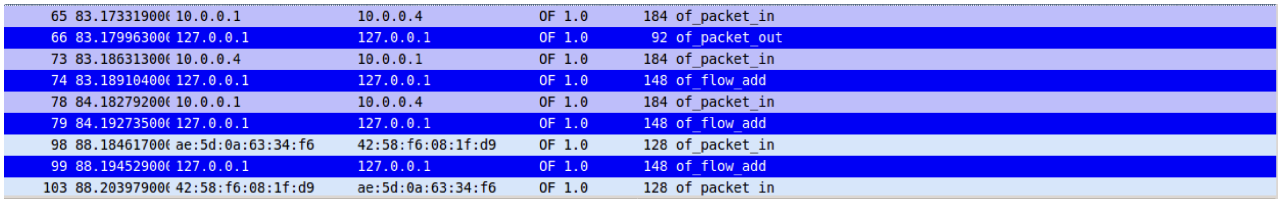


The speed is 7.78 Gbits/sec.

4a.



No. | Time | Source | Destination | Prot.|Len.| Info



5 of_packet_in messages appear.

4b.

For of_packet_in messages we have source 10.0.0.1 and destination 10.0.0.4. We have 1 of_packet_out message, with source 127.0.0.1 and destination 127.0.0.1.

4c.

57 packets are displayed. Echo (ping) request and Echo (ping) reply are displayed.

No.	Time	Source	Destination	Protocol	Length	Info
7	8.986582000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x0be2, seq=1/256, ttl=64
11	8.997974000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x0be2, seq=1/256, ttl=64
12	8.998005000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x0be2, seq=1/256, ttl=64
13	8.998012000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x0be2, seq=1/256, ttl=64
14	8.998025000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0x0be2, seq=1/256, ttl=64 (reply in 15)
15	8.998164000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0be2, seq=1/256, ttl=64 (request in 14)
18	9.004025000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0be2, seq=1/256, ttl=64
19	9.028496000	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x0be3, seq=1/256, ttl=64
22	9.030589000	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x0be3, seq=1/256, ttl=64
23	9.030606000	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x0be3, seq=1/256, ttl=64
24	9.030611000	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x0be3, seq=1/256, ttl=64
25	9.030614000	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x0be3, seq=1/256, ttl=64 (reply in 26)
26	9.030691000	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0be3, seq=1/256, ttl=64 (request in 25)
29	9.037309000	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0be3, seq=1/256, ttl=64
30	9.064873000	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0x0be4, seq=1/256, ttl=64
33	9.071295000	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0x0be4, seq=1/256, ttl=64
34	9.071323000	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0x0be4, seq=1/256, ttl=64
35	9.071329000	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0x0be4, seq=1/256, ttl=64
36	9.071335000	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0x0be4, seq=1/256, ttl=64 (reply in 37)
37	9.071454000	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0be4, seq=1/256, ttl=64 (request in 36)
40	9.073285000	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0x0be4, seq=1/256, ttl=64
41	9.105812000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) request id=0x0be5, seq=1/256, ttl=64
46	9.116491000	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) request id=0x0be5, seq=1/256, ttl=64 (reply in 47)
47	9.116574000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) reply id=0x0be5, seq=1/256, ttl=64 (request in 46)
50	9.122992000	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) reply id=0x0be5, seq=1/256, ttl=64
51	9.143286000	10.0.0.2	10.0.0.3	ICMP	100	Echo (ping) request id=0x0be6, seq=1/256, ttl=64
54	9.147806000	10.0.0.2	10.0.0.3	ICMP	100	Echo (ping) request id=0x0be6, seq=1/256, ttl=64 (reply in 55)
55	9.147896000	10.0.0.3	10.0.0.2	ICMP	100	Echo (ping) reply id=0x0be6, seq=1/256, ttl=64 (request in 54)
58	9.152655000	10.0.0.3	10.0.0.2	ICMP	100	Echo (ping) reply id=0x0be6, seq=1/256, ttl=64
59	9.169181000	10.0.0.2	10.0.0.4	ICMP	100	Echo (ping) request id=0x0be7, seq=1/256, ttl=64
62	9.175207000	10.0.0.2	10.0.0.4	ICMP	100	Echo (ping) request id=0x0be7, seq=1/256, ttl=64 (reply in 63)
63	9.175296000	10.0.0.4	10.0.0.2	ICMP	100	Echo (ping) reply id=0x0be7, seq=1/256, ttl=64 (request in 62)
66	9.177162000	10.0.0.4	10.0.0.2	ICMP	100	Echo (ping) reply id=0x0be7, seq=1/256, ttl=64
67	9.196760000	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) request id=0x0be8, seq=1/256, ttl=64
70	9.202517000	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) request id=0x0be8, seq=1/256, ttl=64 (reply in 71)
71	9.202610000	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) reply id=0x0be8, seq=1/256, ttl=64 (request in 70)
74	9.204880000	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) reply id=0x0be8, seq=1/256, ttl=64
75	9.229184000	10.0.0.3	10.0.0.2	ICMP	100	Echo (ping) request id=0x0be9, seq=1/256, ttl=64
78	9.237475000	10.0.0.3	10.0.0.2	ICMP	100	Echo (ping) request id=0x0be9, seq=1/256, ttl=64 (reply in 79)
79	9.237562000	10.0.0.2	10.0.0.3	ICMP	100	Echo (ping) reply id=0x0be9, seq=1/256, ttl=64 (request in 78)
82	9.244482000	10.0.0.2	10.0.0.3	ICMP	100	Echo (ping) reply id=0x0be9, seq=1/256, ttl=64
83	9.258125000	10.0.0.3	10.0.0.4	ICMP	100	Echo (ping) request id=0x0bea, seq=1/256, ttl=64
86	9.261838000	10.0.0.3	10.0.0.4	ICMP	100	Echo (ping) request id=0x0bea, seq=1/256, ttl=64 (reply in 87)
87	9.261926000	10.0.0.4	10.0.0.3	ICMP	100	Echo (ping) reply id=0x0bea, seq=1/256, ttl=64 (request in 86)
90	9.264932000	10.0.0.4	10.0.0.3	ICMP	100	Echo (ping) reply id=0x0bea, seq=1/256, ttl=64
91	9.274649000	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) request id=0x0beb, seq=1/256, ttl=64
94	9.277235000	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) request id=0x0beb, seq=1/256, ttl=64 (reply in 95)
95	9.277323000	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) reply id=0x0beb, seq=1/256, ttl=64 (request in 94)
98	9.280027000	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) reply id=0x0beb, seq=1/256, ttl=64
99	9.291600000	10.0.0.4	10.0.0.2	ICMP	100	Echo (ping) request id=0x0bec, seq=1/256, ttl=64
102	9.294543000	10.0.0.4	10.0.0.2	ICMP	100	Echo (ping) request id=0x0bec, seq=1/256, ttl=64 (reply in 103)
103	9.294614000	10.0.0.2	10.0.0.4	ICMP	100	Echo (ping) reply id=0x0bec, seq=1/256, ttl=64 (request in 102)
106	9.298003000	10.0.0.2	10.0.0.4	ICMP	100	Echo (ping) reply id=0x0bec, seq=1/256, ttl=64
107	9.320605000	10.0.0.4	10.0.0.3	ICMP	100	Echo (ping) request id=0x0bed, seq=1/256, ttl=64
110	9.323692000	10.0.0.4	10.0.0.3	ICMP	100	Echo (ping) request id=0x0bed, seq=1/256, ttl=64 (reply in 111)
111	9.323785000	10.0.0.3	10.0.0.4	ICMP	100	Echo (ping) reply id=0x0bed, seq=1/256, ttl=64 (request in 110)
114	9.326884000	10.0.0.3	10.0.0.4	ICMP	100	Echo (ping) reply id=0x0bed, seq=1/256, ttl=64