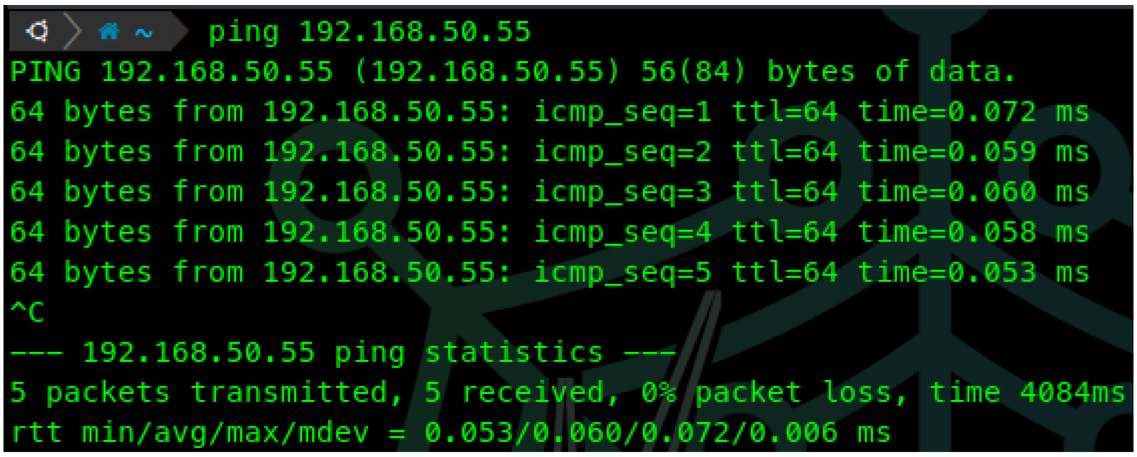
Scripting with Bash

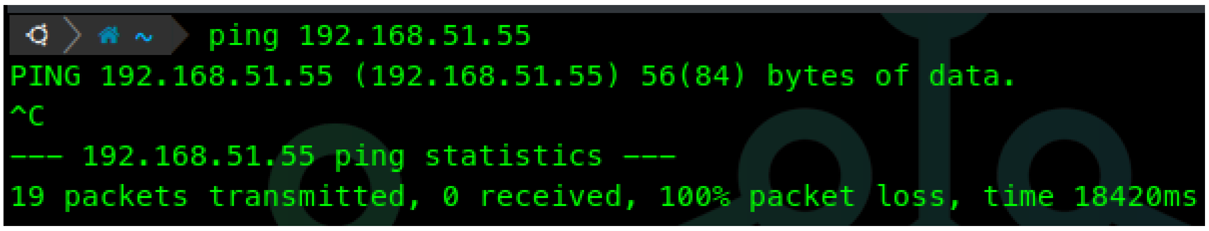
Making a Ping/IP Sweeper.
First we grab our IP address with `ip a` or `ifconfig`. Mine is 192.168.50.55 in this example.

Then ping an IP address, I'm using my own(192.168.50.55) for this example



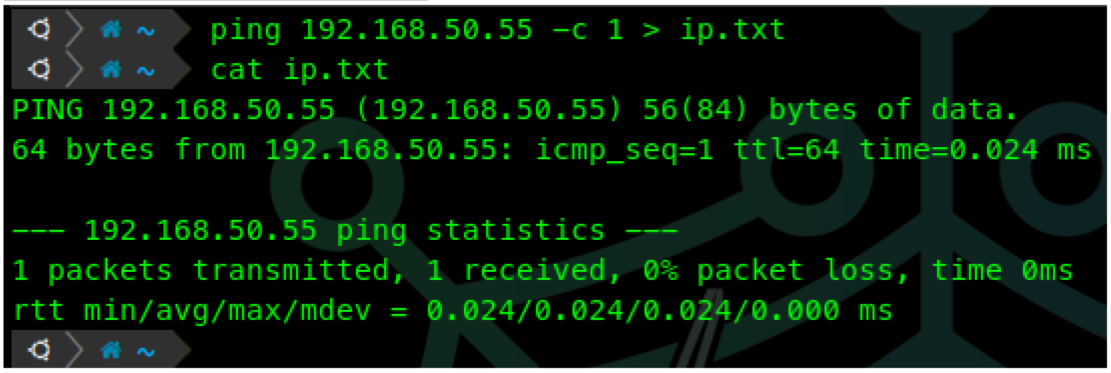
```
ping 192.168.50.55
PING 192.168.50.55 (192.168.50.55) 56(84) bytes of data.
64 bytes from 192.168.50.55: icmp_seq=1 ttl=64 time=0.072 ms
64 bytes from 192.168.50.55: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 192.168.50.55: icmp_seq=3 ttl=64 time=0.060 ms
64 bytes from 192.168.50.55: icmp_seq=4 ttl=64 time=0.058 ms
64 bytes from 192.168.50.55: icmp_seq=5 ttl=64 time=0.053 ms
^C
--- 192.168.50.55 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4084ms
rtt min/avg/max/mdev = 0.053/0.060/0.072/0.006 ms
```

Change it up to an IP address we will not receive a response from, 192.168.51.55



```
ping 192.168.51.55
PING 192.168.51.55 (192.168.51.55) 56(84) bytes of data.
^C
--- 192.168.51.55 ping statistics ---
19 packets transmitted, 0 received, 100% packet loss, time 18420ms
```

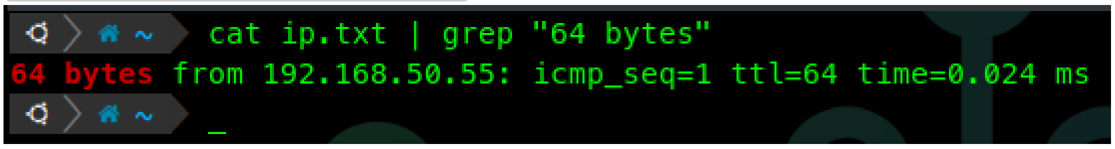
Ping our first IP and output the text to a file with just a count of 1 so we only ping it one time. `ping 192.168.50.55 -c 1> ip.txt`



```
ping 192.168.50.55 -c 1 > ip.txt
cat ip.txt
PING 192.168.50.55 (192.168.50.55) 56(84) bytes of data.
64 bytes from 192.168.50.55: icmp_seq=1 ttl=64 time=0.024 ms

--- 192.168.50.55 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.024/0.024/0.024/0.000 ms
```

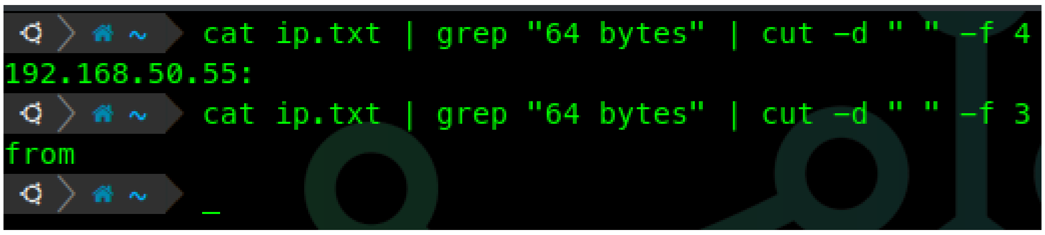
Now we use `grep` to pull one line: `64 bytes from 192.168.50.55: icmp_seq=1 ttl=64 time=0.024 ms` from it.
`cat ip.txt | grep "64 bytes"`



```
cat ip.txt | grep "64 bytes"
64 bytes from 192.168.50.55: icmp_seq=1 ttl=64 time=0.024 ms
_
```

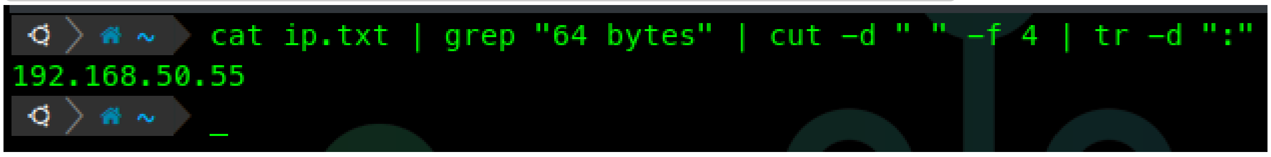
Since we want to pull every IP from this subnet, we will ping all of them once to see if they are alive. Since we can do this with ping and ping the whole subnet, but we want to pull just the hosts that are alive, we will make a script that will do this, and pull just the relevant information out.

When we pipe(|) a command, we're saying run this command but within this command run this command. So we're going to add another pipe.
`cat ip.txt | grep "64 bytes" | cut -d " " -f 4`
This command is saying we want to "cut" this line we're getting back, using the delimeter of of a space, and pulling the 4th "field" which is -f, which is the IP address.
To help clarify, If we did it with a 3, it would pull the work "from"



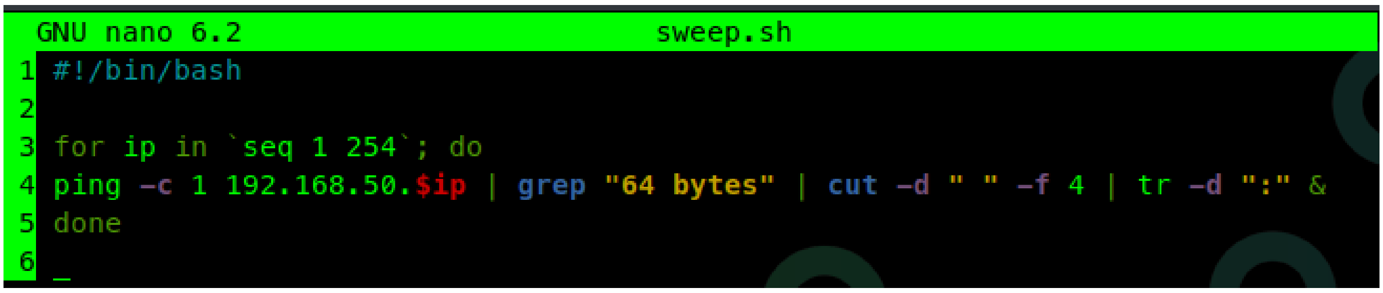
```
cat ip.txt | grep "64 bytes" | cut -d " " -f 4
192.168.50.55:
cat ip.txt | grep "64 bytes" | cut -d " " -f 3
from
_
```

But this is still pulling the colon(:) from the 4th field. So now we are going to "translate".
`cat ip.txt | grep "64 bytes" | cut -d " " -f 4 | tr -d ":"`



```
cat ip.txt | grep "64 bytes" | cut -d " " -f 4 | tr -d ":"
192.168.50.55
_
```

To make this part of a script
we make a script, I will call mine `sweep.sh`. I will be using nano for my script. Each bash script should start with a shbang (#!/bin/bash). SInce we are tying to ping IP addresses to see what is alive, we want to make a 'for' loop.

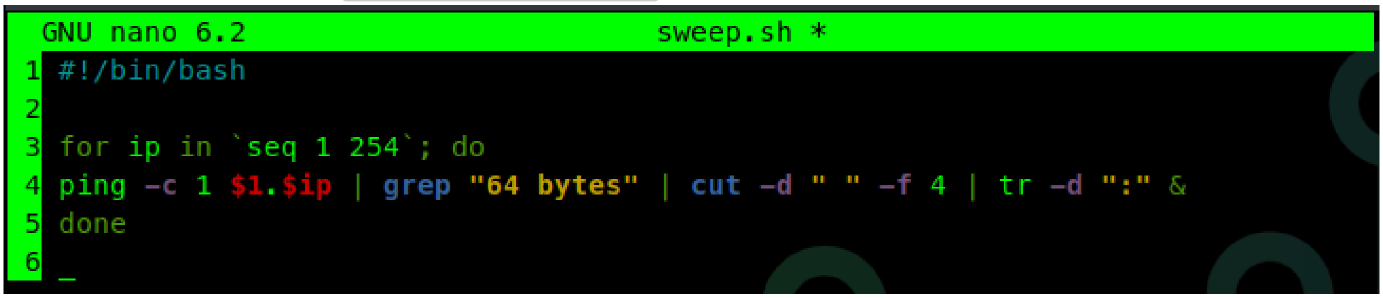


```
GNU nano 6.2                    sweep.sh
#!/bin/bash

for ip in `seq 1 254`; do
ping -c 1 192.168.50.$ip | grep "64 bytes" | cut -d " " -f 4 | tr -d ":" &
done
_
```

The 'ip' in the for loop, is a variable, it can be anything you want. Thr 'seq' is counting everything from number 1 to 254. The for loop in combination with the seq is saying it will run the command on every IP address, in our case, 192.168.50.$ until this is done.

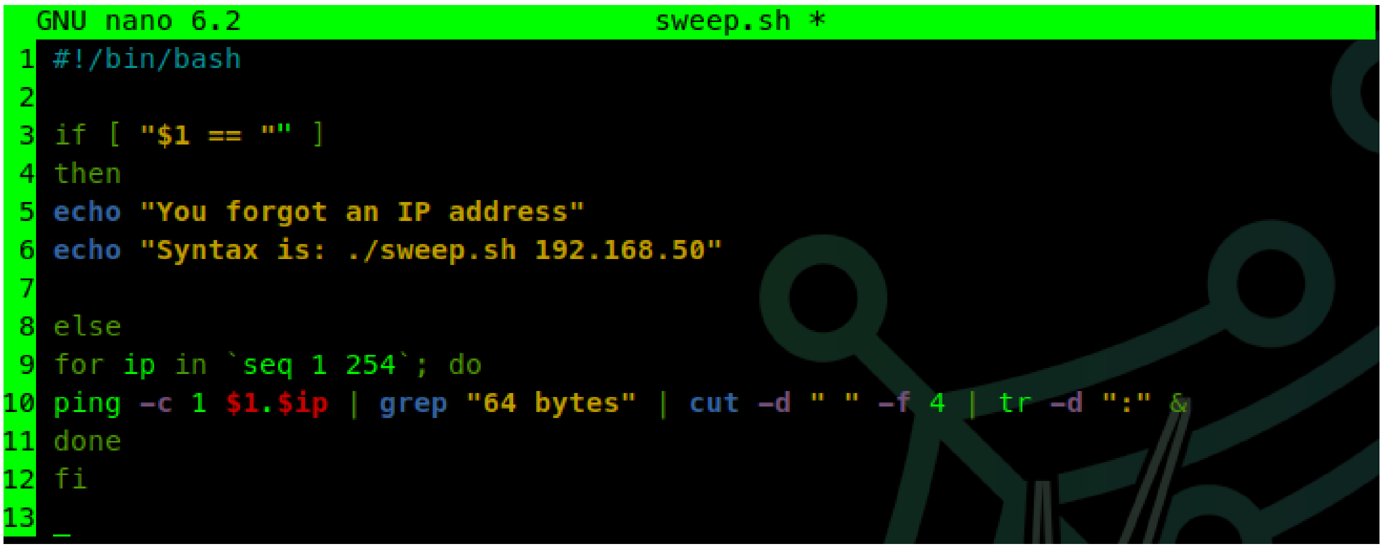
Now, this script will work as is but we can make it better. Right now it on'y scans IP addresses that being with 192.168.50, but we can make it call an argument to scan a given IP range and change the command of the script to `sweep.sh 192.168.50`.



```
GNU nano 6.2                    sweep.sh *
#!/bin/bash

for ip in `seq 1 254`; do
ping -c 1 $1.$ip | grep "64 bytes" | cut -d " " -f 4 | tr -d ":" &
done
_
```

Now we are going to add in an 'if statement', meaning, that "if" this condition is met, do X, if it's not, do Y. So we modify this to say: if argument $1 doesn't equal anything, it will give us an error and tell us the syntax. If it does equal something, it will run our for loop. The & at the end will run it multiple times at once instead of just one at a time.

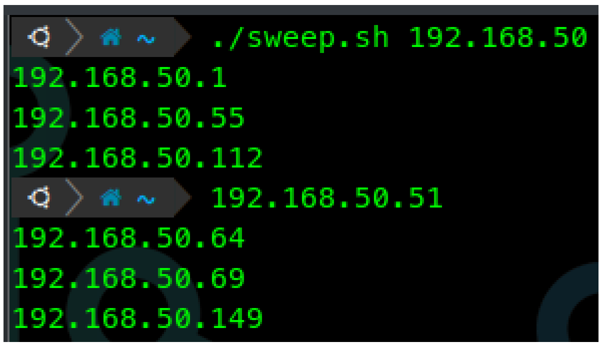


```
GNU nano 6.2                    sweep.sh *
#!/bin/bash

if [ "$1 == "" ]
then
echo "You forgot an IP address"
echo "Syntax is: ./sweep.sh 192.168.50"

else
for ip in `seq 1 254`; do
ping -c 1 $1.$ip | grep "64 bytes" | cut -d " " -f 4 | tr -d ":" &
done
fi
_
```

We can run this and see that it works:



```
./sweep.sh 192.168.50
192.168.50.1
192.168.50.55
192.168.50.112
192.168.50.51
192.168.50.64
192.168.50.69
192.168.50.149
```

We can output this to a file as well, `./sweep.sh 192.168.50 > ip.txt`. We can also now take this file and run an nmap scan against it:
`for ip in $(cat ip.txt); do nmap$ip; done`

Full script of **sweep.sh**:

```
#!/bin/bash
if [ "$1" == "" ]
then
echo "You forgot an IP address!"
echo "Syntax: ./sweep.sh 192.168.1"

else
for ip in `seq 1 254`; do
ping -c 1 $1.$ip | grep "64 bytes" | cut -d " " -f 4 | tr -d ":" &
done
fi
```