Template Week 6 – Networking

Student number: 547201

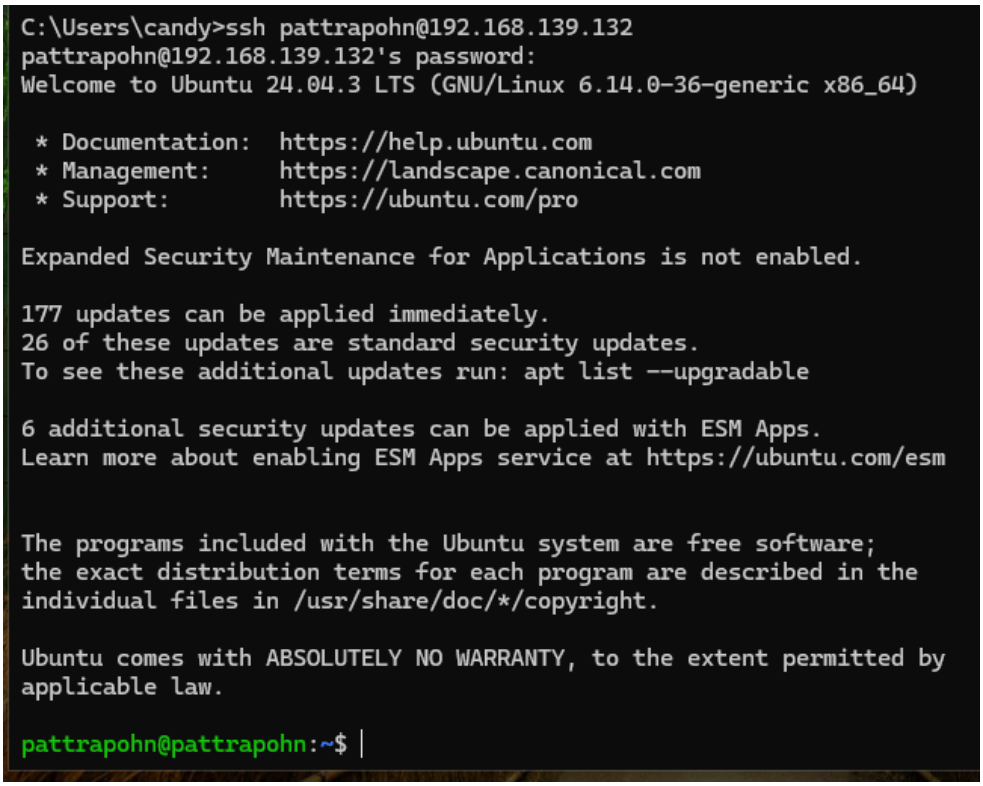
Assignment 6.1: Working from home

Screenshot installation openssh-server:

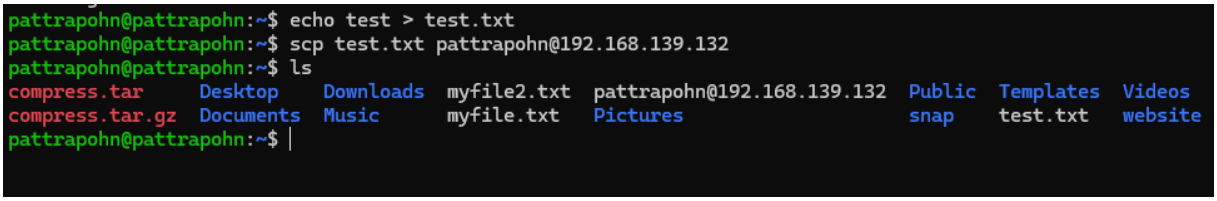
```
pattrapohn@pattrapohn:~$ sudo apt update
[sudo] password for pattrapohn:
Hit:1 http://nl.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:3 http://nl.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://nl.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
150 packages can be upgraded. Run 'apt list --upgradable' to see them.
pattrapohn@pattrapohn:~$ sudo aapt install openssh-server -y
sudo: aapt: command not found
pattrapohn@pattrapohn:~$ sudo apt install openssh-server -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssh-server is already the newest version (1:9.6p1-3ubuntu13.14).
0 upgraded, 0 newly installed, 0 to remove and 150 not upgraded.
pattrapohn@pattrapohn:~$ sudo systemctl enable --now ssh
Synchronizing state of ssh.service with SysV service script with /usr/lib/system
d/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh
pattrapohn@pattrapohn:~$
```

```
pattrapohn@pattrapohn: ~
d/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh
Created symlink /etc/systemd/system/sshd.service → /usr/lib/systemd/system/ssh.s
ervice.
Created symlink /etc/systemd/system/multi-user.target.wants/ssh.service → /usr/l
ib/systemd/system/ssh.service.
pattrapohn@pattrapohn:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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 noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:91:0b:28 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.139.132/24 brd 192.168.139.255 scope global dynamic noprefixrou
te ens33
       valid_lft 1696sec preferred_lft 1696sec
    inet6 fe80::20c:29ff:fe91:b28/64 scope link
       valid_lft forever preferred_lft forever
pattrapohn@pattrapohn:~$
```

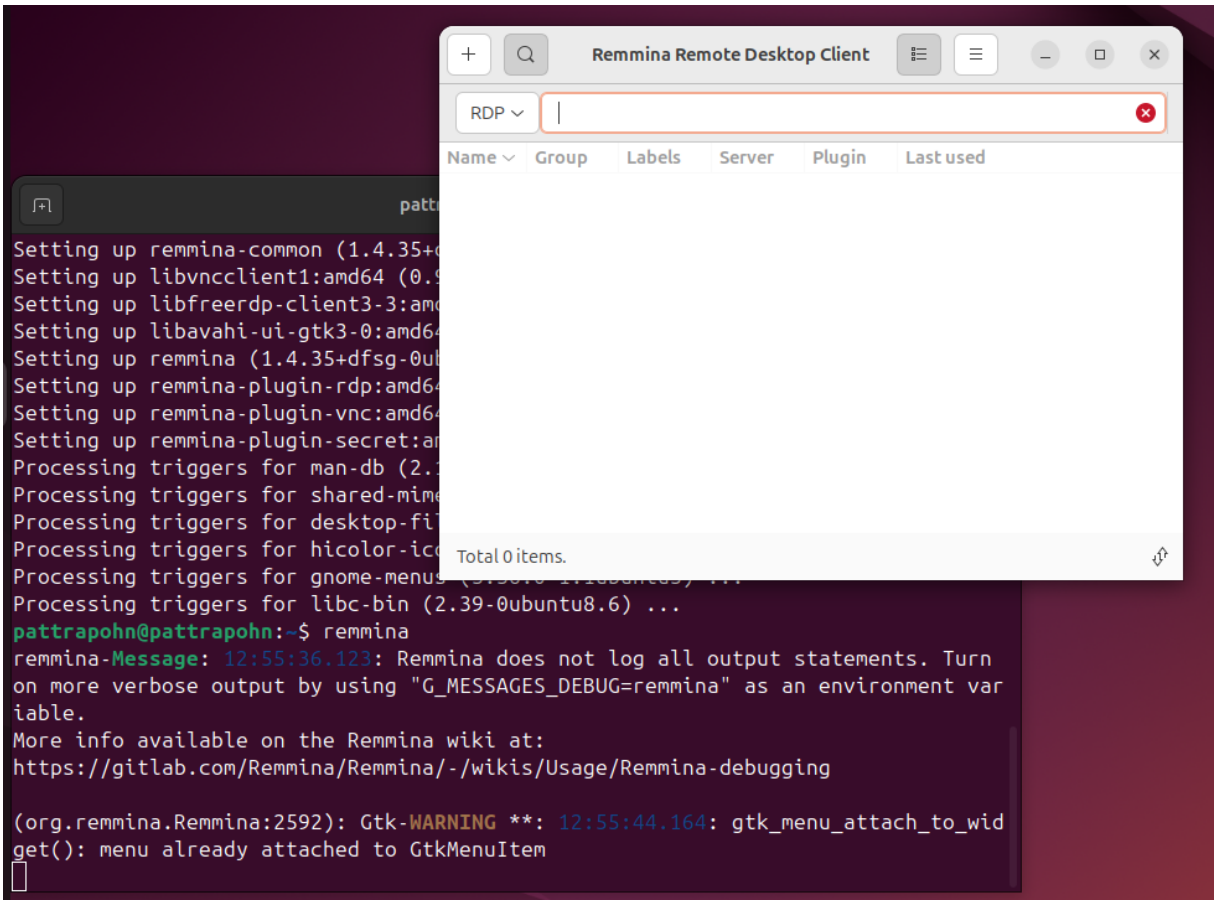
Screenshot successful SSH command execution:



Screenshot successful execution SCP command:



Screenshot remmina:



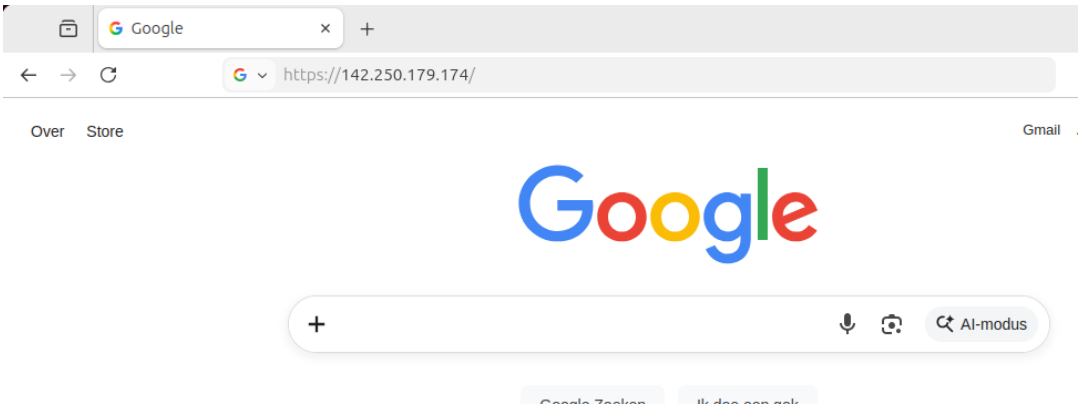
Assignment 6.2: IP addresses websites

Relevant screenshots nslookup command:

```
pattrapohn@pattrapohn:~$ nslookup
> google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.179.174
Name:   google.com
Address: 2a00:1450:400e:801::200e
> exit
```

Screenshot website visit via IP address:



Assignment 6.3: subnetting

How many IP addresses are in this network configuration 192.168.110.128/25?

- 128 IP addresses

What is the usable IP range to hand out to the connected computers?

- HostMin 192.168.110.129 – HostMax 192.168.110.254

Check your two previous answers with this Linux command: `ipcalc 192.168.110.128/25`

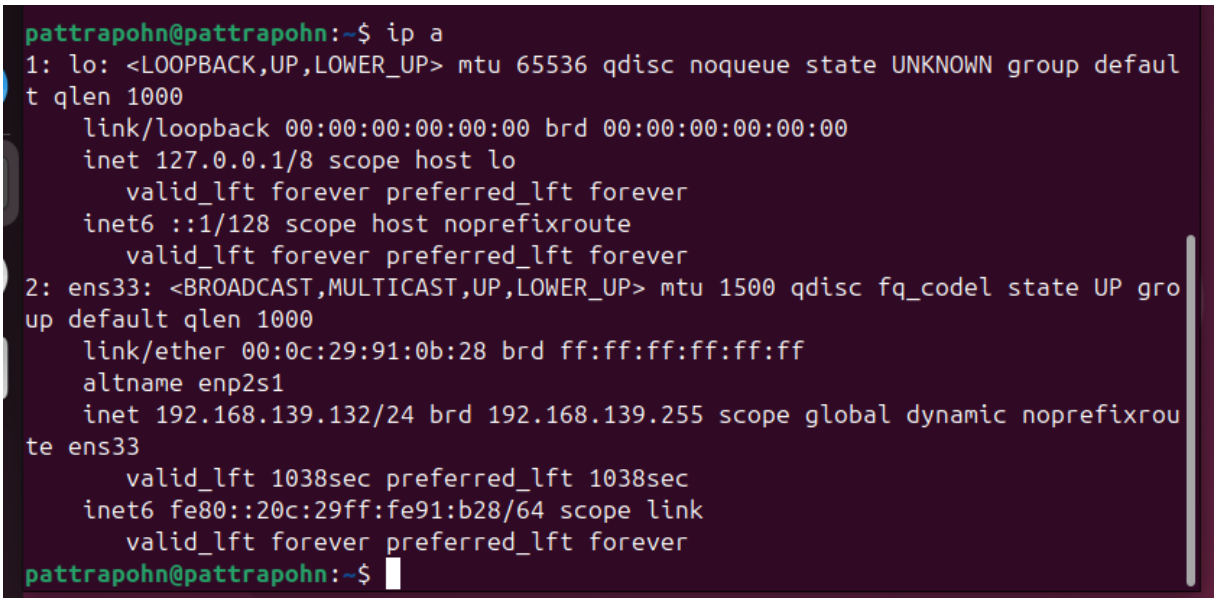
```
pattrapohn@pattrapohn:~$ ipcalc 192.168.110.128/25
Address:   192.168.110.128       11000000.10101000.01101110.1 0000000
Netmask:   255.255.255.128 = 25  11111111.11111111.11111111.1 0000000
Wildcard:  0.0.0.127             00000000.00000000.00000000.0 1111111
=>
Network:   192.168.110.128/25    11000000.10101000.01101110.1 0000000
HostMin:   192.168.110.129       11000000.10101000.01101110.1 0000001
HostMax:   192.168.110.254       11000000.10101000.01101110.1 1111110
Broadcast: 192.168.110.255       11000000.10101000.01101110.1 1111111
Hosts/Net: 126                    Class C, Private Internet
```

Explain the above calculation in your own words.

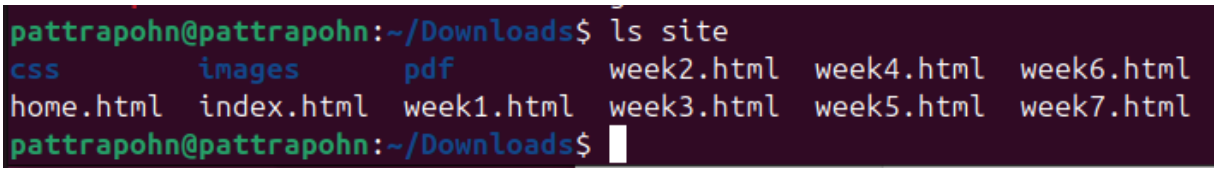
There is 25 bits used for network and 7 bits remain for hosts, so 2^7 = 128 total IP addresses. The first address is the network address and the last is the broadcast address, so 126 addresses are usable for hosts - states in the last line.

Assignment 6.4: HTML

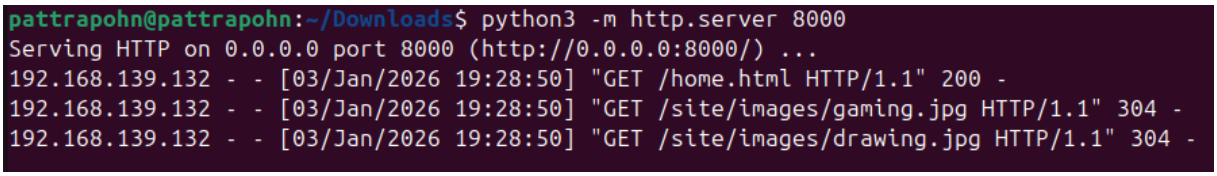
Screenshot IP address Ubuntu VM:



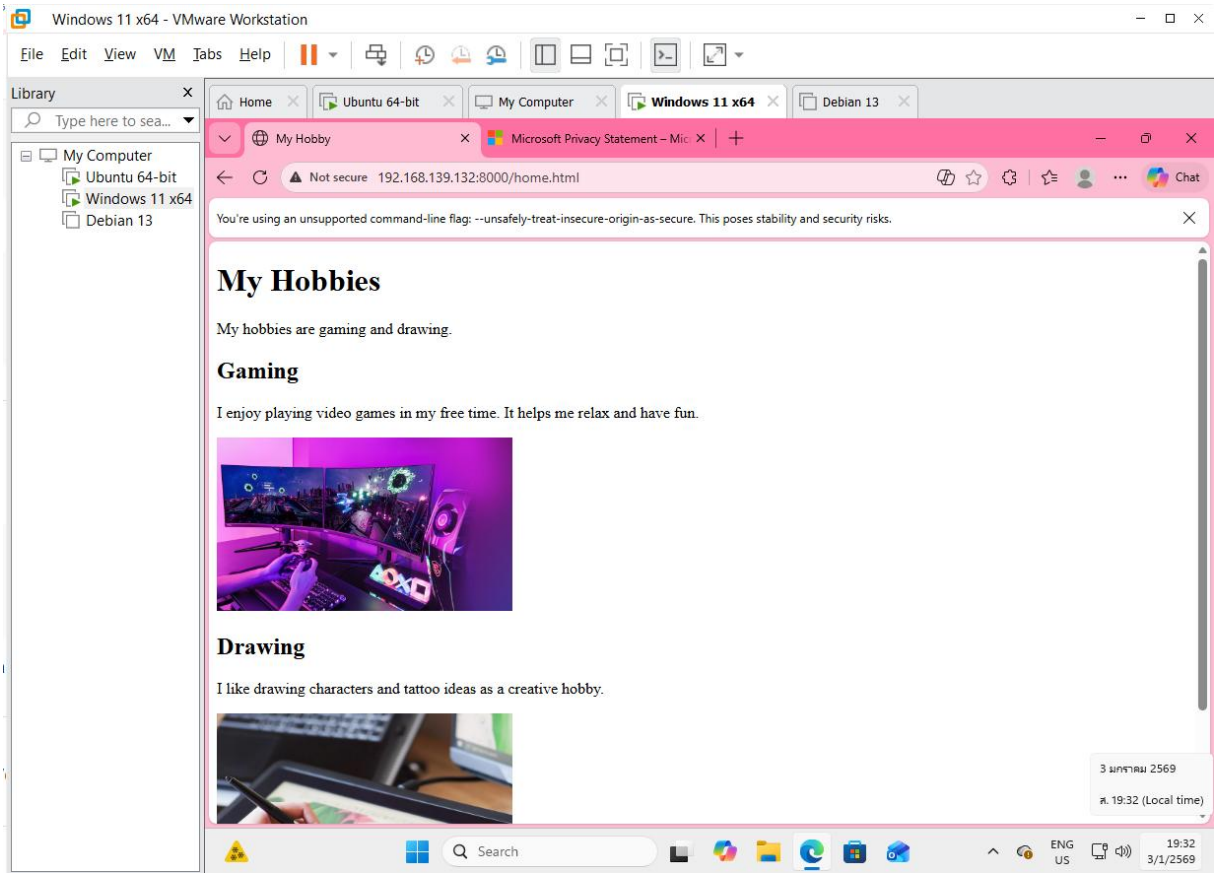
Screenshot of Site directory contents:



Screenshot python3 webserver command:



Screenshot web browser visits your site



Assignment 6.5: Network segment

Remember that bitwise java application you've made in week 2? Expand that application so that you can also calculate a network segment as explained in the PowerPoint slides of week 6. Use the bitwise & AND operator. You need to be able to input two Strings. An IP address and a subnet.

IP: 192.168.1.100 and subnet: 255.255.255.224 for /27

```
Example: 192.168.1.100/27
Calculate the network segment
IP Address:   11000000.10101000.00000001.01100100
Subnet Mask:  11111111.11111111.11111111.11100000
-----------------------------------------------
Network Addr: 11000000.10101000.00000001.01100000

This gives 192.168.1.96 in decimal as the network address.
For a /27 subnet, each segment (or subnet) has 32 IP addresses (2^5).
The range of this network segment is from 192.168.1.96 to 192.168.1.127.
```

Paste source code here, with a screenshot of a working application.

```
SaxionApp.printLine("=== 8: Network Segment Calculation ===");

   // Input
   SaxionApp.print("Enter IP address (e.g. 192.168.1.100): ");
   String ipString = SaxionApp.readString();

   SaxionApp.print("Enter subnet mask (e.g. 255.255.255.224): ");
   String subnetString = SaxionApp.readString();

   int ip = ipToInt(ipString);
   int subnet = ipToInt(subnetString);

   // Calculate network address
   int network = ip & subnet;

   // Count how many 1-bits in the subnet mask
   int prefixLength = Integer.bitCount(subnet);

   // How many IP addresses in this subnet
   int totalAddress = 1 << (32 - prefixLength);  // bit shift (it'll be 2^host bits to calculate possible
combinations)

   // First and last IP in range
   int firstIp = network;
   int lastIp = network + totalAddress - 1;

   // Output
   SaxionApp.printLine("\nIP Address:   " + toBinary(ip));
   SaxionApp.printLine("Subnet Mask:  " + toBinary(subnet));
   SaxionApp.printLine("-------------------------------------------------");
   SaxionApp.printLine("Network Addr: " + toBinary(network));
   SaxionApp.printLine();

   SaxionApp.printLine("Network address: " + intToIp(network));
   SaxionApp.printLine("Prefix: /" + prefixLength + " (segment has " + totalAddress + " addresses)");
   SaxionApp.printLine("Range: " + intToIp(firstIp) + " to " + intToIp(lastIp));

   SaxionApp.pause();
}

public int ipToInt(String ip) {
   String[] parts = ip.split("\\.");  //split by dots
   return (Integer.parseInt(parts[0]) << 24) |  //shift to its right position
         (Integer.parseInt(parts[1]) << 16) |
         (Integer.parseInt(parts[2]) << 8) |
         Integer.parseInt(parts[3]);
}
```

```
public String intToIp(int value) {
   return ((value >> 24) & 0xFF) + "." +  //shift back to the same position, 0xff for 255 and mask out
everything except the lowest 8 bits
          ((value >> 16) & 0xFF) + "." +
          ((value >> 8) & 0xFF) + "." +
          (value & 0xFF);
}

public String toBinary(int value) {
char[] result = new char[35]; // 8*4 + 3 dots = 35 chars

   int index = 0

   for (int i = 0; i < 4; i++) {
   int octet = (value >> (24 - i * 8)) & 0xFF;
   // Build binary manually by checking each bit
   for (int bit = 7; bit >= 0; bit--) {//ex. 192 = 11000000 >> 7 & 1 = 00000001 & 00000001 = 1 in index 0
       result[index++] = ((octet >> bit) & 1) == 1 ? '1' : '0'; }

if (i < 3) result[index++] = '.'; }

   return new String(result);
```

```
=== 8: Network Segment Calculation ===
Enter IP address (e.g. 192.168.1.100): 192.168.1.100
Enter subnet mask (e.g. 255.255.255.224): 255.255.255.224

IP Address:    11000000.10101000.00000001.01100100
Subnet Mask:   11111111.11111111.11111111.11100000
------------------------------------------------
Network Addr: 11000000.10101000.00000001.01100000

Network address: 192.168.1.96
Prefix: /27 (segment has 32 addresses)
Range: 192.168.1.96 to 192.168.1.127
```