CS5250 - Assignment 1

Daniel Alfred Widjaja - A0184588J

Running virtual box

- Followed the instruction and the first problem I found is
- Before doing anything, this is the kernel version and MAC address.

```
daniel-alfred@monmouth:~$ uname -a
Linux monmouth 5.8.0-41-generic #46-Ubuntu SMP Mon Jan 18 16:48:44 UTC 2021 x86
_64 x86_64 x86_64 GNU/Linux
```

```
daniel@monmouth:~/work/linux-5.10.6$ ip a
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: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 10
00
    link/ether 08:00:27:95:55:39 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 84828sec preferred_lft 84828sec
    inet6 fe80::1563:c176:fcf3:2b6e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
```

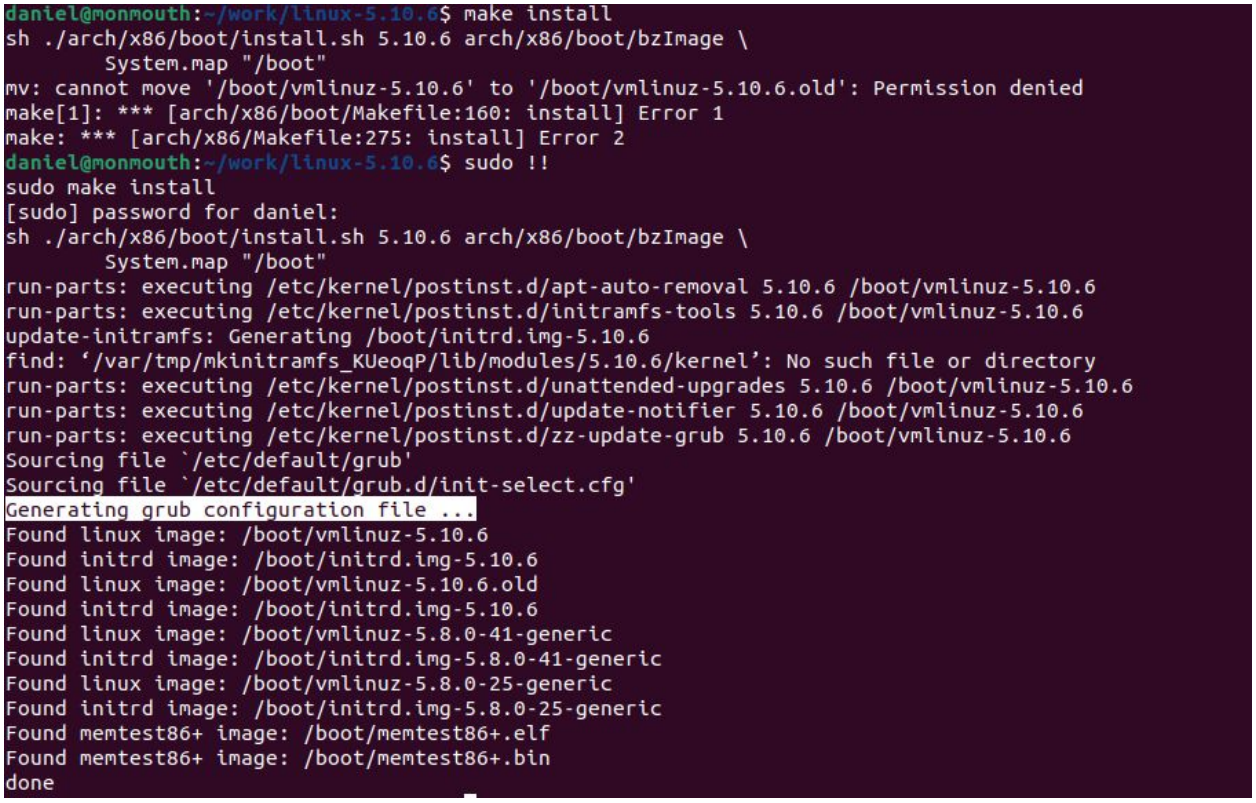
- **Task 2**: When running ‘make menuconfig’ there’s 3 options which are built-in, exclude, or module.
    - built-in means installed directly to the kernel
    - module means it’s installed as a module which can be removed if you wish
    - while exclude means it’s not installed at all
- The one that appears in the kernel image is built-in only.
- I did not change anything in /boot/grub/grub.cfg since it’s a generated file

```
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by grub-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#
```

- I reboot and the kernel is updated

```
daniel@monmouth:~$ uname -a
Linux monmouth 5.10.6 #1 SMP Mon Feb 8 08:10:26 UTC 2021 x86_64 x86_64 x86_64 G
NU/Linux
```

- This happens because ‘make install’ already changes the grub.cfg for us as we can see in the image below.



Building smaller kernel

- By seeing ‘make help’ we can see there is ‘make tinyconfig’ which allows us to create the tiniest possible kernel.

```
Configuration targets:
  config          - Update current config utilising a line-oriented program
  nconfig         - Update current config utilising a ncurses menu based program
  menuconfig      - Update current config utilising a menu based program
  xconfig         - Update current config utilising a Qt based front-end
  gconfig         - Update current config utilising a GTK+ based front-end
  oldconfig       - Update current config utilising a provided .config as base
  localmodconfig  - Update current config disabling modules not loaded
                    except those preserved by LMC_KEEP environment variable
  localyesconfig  - Update current config converting local mods to core
                    except those preserved by LMC_KEEP environment variable
  defconfig       - New config with default from ARCH supplied defconfig
  savedefconfig   - Save current config as ./defconfig (minimal config)
  allnoconfig     - New config where all options are answered with no
  allyesconfig    - New config where all options are accepted with yes
  allmodconfig    - New config selecting modules when possible
  alldefconfig    - New config with all symbols set to default
  randconfig      - New config with random answer to all options
  yes2modconfig   - Change answers from yes to mod if possible
  mod2yesconfig   - Change answers from mod to yes if possible
  listnewconfig   - List new options
  helpnewconfig   - List new options and help text
  olddefconfig    - Same as oldconfig but sets new symbols to their
                    default value without prompting
  tinyconfig      - Configure the tiniest possible kernel
  testconfig      - Run Kconfig unit tests (requires python3 and pytest)
```

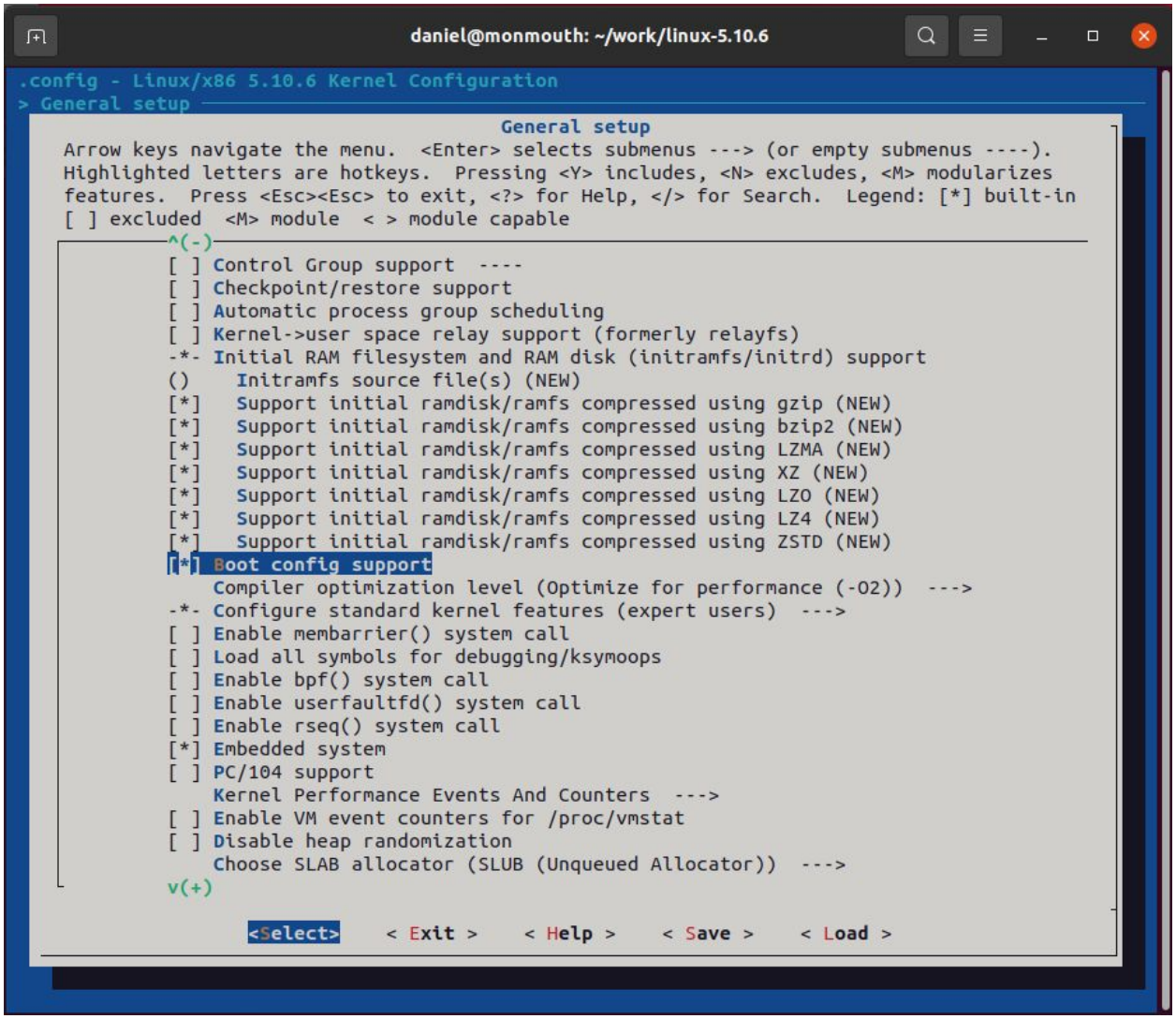
- Interesting thing happens when ‘make modules_install’ and it doesn’t install any modules.

```
daniel@monmouth:~/work/linux-5.10.6$ sudo make modules_install
[sudo] password for daniel:

The present kernel configuration has modules disabled.
Type 'make config' and enable loadable module support.
Then build a kernel with module support enabled.

make: *** [Makefile:1458: modules_install] Error 1
```

- However, after I reboot, it does not load.
- So I restart everything (because I forgot to take a snapshot)



- I started with tinyconfig again and include boot config support. It runs into grub but still can’t log in.

make localmodconfig

- So instead of making from tinyconfig, I start with the default config and disable the not needed modules. make localmodconfig allows us to disable modules not loaded.
    - The disabling module part is a little bit trial and error because I tried to disable more and more modules and if it doesn’t boot, I will revert to the previous version.
- Additionally we can disable certain stuff like:
    - several networking support
    - security option
    - use XZ compression

- etc.
- The excluded modules above can make the kernel image size smaller as can be seen below .

```
daniel@monmouth:~/work/linux-5.10.6$ du -h arch/x86/boot/bzImage
3.3M    arch/x86/boot/bzImage
```

- The kernel works and only cost 3.3MB.

```
daniel@monmouth:~$ uname -a
Linux monmouth 5.10.6 #1 Tue Feb 9 12:39:04 +08 2021 x86_64 x86_64 x86_64 GNU/L
inux
```

Part B

1. 4d 29 44 7a a7
    a. 4d is a prefix (0100 1101), W = 1, R = 1, X = 0, B = 1
    b. 29 is the opcode that means subtract
    c. 44 is the ModRM byte which is 0100 0100
        i. mod = 01 (with register, [r/m + disp8] is used
        ii. R + reg/opcode = 1000 (r8)
        iii. B + R/M = 1100
    d. 7a is the SIB byte which is 0111 1010
        i. scale = 01 = 2
        ii. X + Index = 0111 (rdi)
        iii. B + base = 1010 (r10), based on the table this makes [base + index * s]
        iv. so [r10 + rdi * 2]
    e. a7 is the displacement which is 1010 0111
        i. That is the binary of -0x59
        ii. 1010 0111 --inverse--> 0101 1000 --(+1)--> 0101 1001 = 0x59
    f. `sub    %r8, -0x59(%r10, %rdi, 2)`
2. `addl 8(%esp), %ecx`
    a. `addl = 03 opcode because we want to add 32 bits to 32 bits`
    b. `4c (0100 1100) is the ModRM, this is for %ecx`
        i. `mod = 01 (1 byte displacement)`
        ii. `reg ecx = 001`
        iii. `RM = 100 by default`
    c. `24 (0010 0100) is the SIB bytes, this is for %esp`
        i. `mod = 00 (no multiplier)`
        ii. `index illegal = 100`

iii. base esp = 100

d. 08 is the displacement

i. +8 = 0x08

e. 03 4c 24 08

3.
```
void unknown_func(char *c) {
        int arr[10];
        arr[7] = 0;
        while (*c != 0) {
             if (*c == 101) break;
             arr[7]++;
             *c++;
        }
        arr[0] = arr[7];
        printf("%d\n", arr[7]);
  }
```