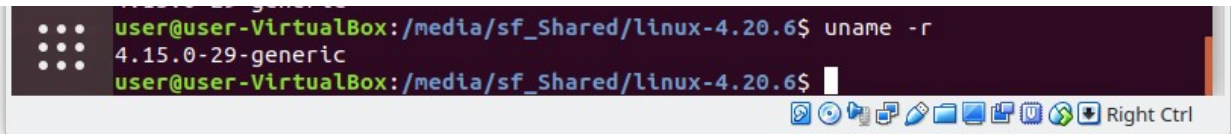


Assignment 1

OS : Kubuntu

Downloaded the linux kernel

A terminal window from a Kubuntu virtual machine. The prompt is 'user@user-VirtualBox:/media/sf_Shared/linux-4.20.6\$'. The command 'uname -r' has been executed, and the output is '4.15.0-29-generic'. The prompt is now 'user@user-VirtualBox:/media/sf_Shared/linux-4.20.6\$' with a cursor at the end. The terminal has a dark background and standard Linux icons on the left. A 'Right Ctrl' button is visible in the bottom right corner of the terminal window.

```
user@user-VirtualBox:/media/sf_Shared/linux-4.20.6$ uname -r
4.15.0-29-generic
user@user-VirtualBox:/media/sf_Shared/linux-4.20.6$
```

Version 4.20.6 was the stable version was chosen.

Command

```
wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.20.6.tar.xz
unxz -v linux-4.20.6.tar.xz
```

To generate the config file.
`make defconfig`

```
make -j $(nproc)
```

Question 2

What is built-in?

When the kernel is booted up the kernel automatically inserts these drivers into the kernel. This makes the kernel size larger. The module will be loaded inside the memory whenever the kernel is loaded. And the benefits are the all these module are not fragmented. We should only use built-in on the modules that are necessary for the user.

What is excluded?

A excluded module is not added to the kernel at compile time.

What is module?

These modules are loaded at runtime. These are also known as Loadable Kernel Module.

In practice.

This config means.

`Driver_A=y`

The driver_A is a built-in module. Hence it will be loaded during the kernel bootup.

While, This config means.

`Driver_B=m`

The driver_b is a Loadable Kernel Module. A LKM is a module that gets loaded dynamically.

To view a list of the LKM Modules we can use the command.

lsmod

Module	Size	Used by
ccm	20480	6
rfcomm	77824	16
pci_stub	16384	1

.

.

.

Since LKM are loaded inside the kernel process. This is a potential security issue. As if a user might load a malicious LKM.

Which are the ones that will appear in the kernel image?

Only the built-in drivers appear in the kernel. And the module drivers will be under the folder `/lib/modules/` or `/lib/modules/[Kernel Version]`

What is `make defconfig`?

Defconfig use the basic set of needed driver needed to run the server. It uses the `x86_64_defconfig`.

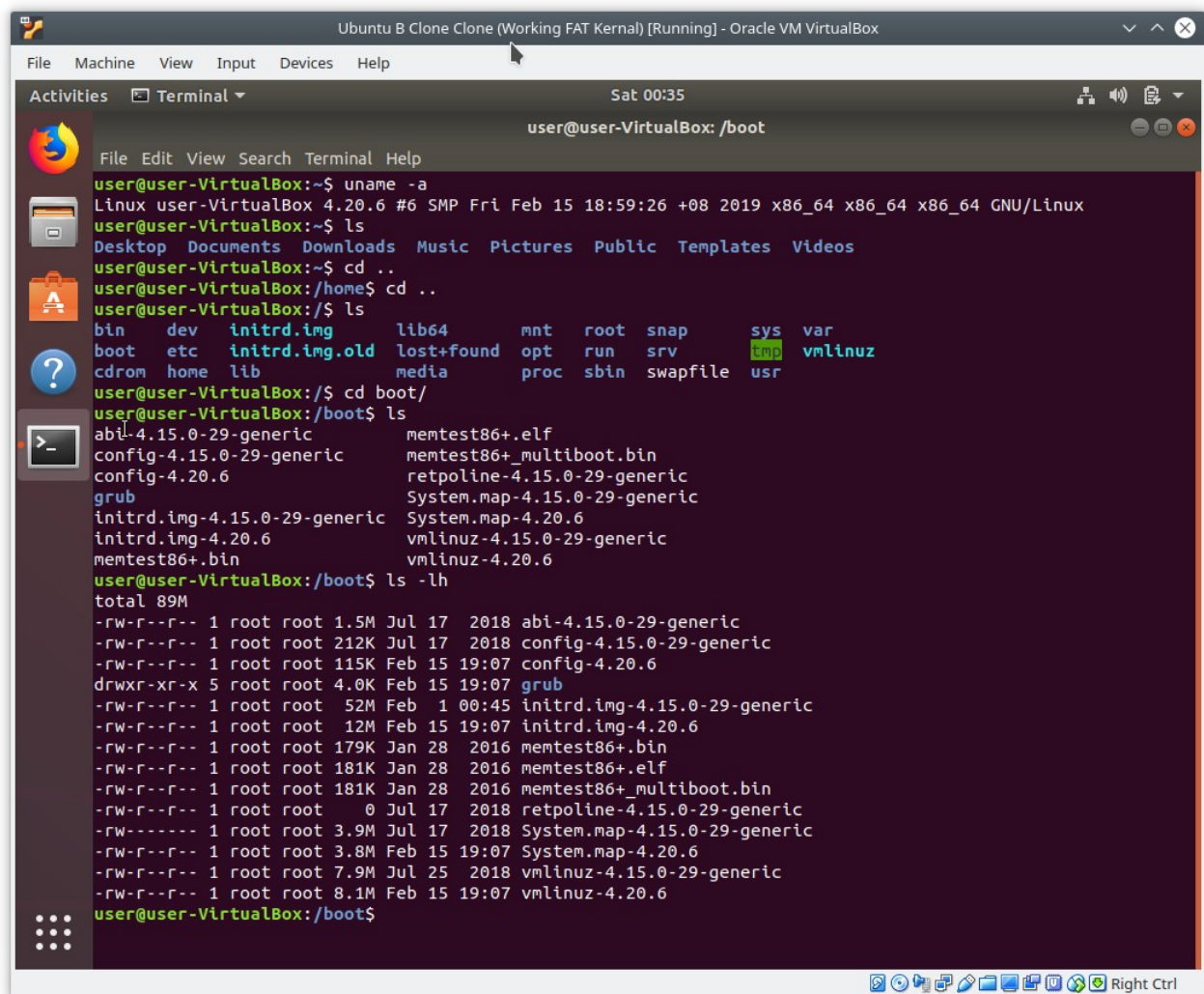
Problems encountered

The compile time is long, to speed it up. To make the compile multi-threaded.

We use the command `make -j 4` to speed up the compile time.

Size of the kernel when using defconfig.

System is 8261 kB



The screenshot shows a terminal window titled "Ubuntu B Clone Clone (Working FAT Kernel) [Running] - Oracle VM VirtualBox". The terminal output shows the user navigating through the file system to the `/boot` directory and listing files. The output of `ls -lh` in `/boot` is as follows:

```
total 89M
-rw-r--r-- 1 root root 1.5M Jul 17 2018 abi-4.15.0-29-generic
-rw-r--r-- 1 root root 212K Jul 17 2018 config-4.15.0-29-generic
-rw-r--r-- 1 root root 115K Feb 15 19:07 config-4.20.6
drwxr-xr-x 5 root root 4.0K Feb 15 19:07 grub
-rw-r--r-- 1 root root 52M Feb 1 00:45 initrd.img-4.15.0-29-generic
-rw-r--r-- 1 root root 12M Feb 15 19:07 initrd.img-4.20.6
-rw-r--r-- 1 root root 179K Jan 28 2016 memtest86+.bin
-rw-r--r-- 1 root root 181K Jan 28 2016 memtest86+.elf
-rw-r--r-- 1 root root 181K Jan 28 2016 memtest86+_multiboot.bin
-rw-r--r-- 1 root root 0 Jul 17 2018 retpoline-4.15.0-29-generic
-rw-r--r-- 1 root root 3.9M Jul 17 2018 System.map-4.15.0-29-generic
-rw-r--r-- 1 root root 3.8M Feb 15 19:07 System.map-4.20.6
-rw-r--r-- 1 root root 7.9M Jul 25 2018 vmlinuz-4.15.0-29-generic
-rw-r--r-- 1 root root 8.1M Feb 15 19:07 vmlinuz-4.20.6
```

Size of the kernal after removing.
 Change zip format.
 Change zip to XZ
 Optimize for size.
 Remove Kernel Hacking mods
 System is 5829 kB

```

user@user-VirtualBox: /boot
File Edit View Search Terminal Help
user@user-VirtualBox:/$ uname -a
Linux user-VirtualBox 4.20.6 #10 SMP Sat Feb 16 00:36:17 +08 2019 x86_64 x86_64
x86_64 GNU/Linux
user@user-VirtualBox:/$ cd boot/
user@user-VirtualBox:/boot$ ls -lh
total 86M
-rw-r--r-- 1 root root 1.5M Jul 17 2018 abi-4.15.0-29-generic
-rw-r--r-- 1 root root 212K Jul 17 2018 config-4.15.0-29-generic
-rw-r--r-- 1 root root 110K Feb 16 2019 config-4.20.6
drwxr-xr-x 5 root root 4.0K Feb 16 2019 grub
-rw-r--r-- 1 root root 52M Feb 1 00:45 initrd.img-4.15.0-29-generic
-rw-r--r-- 1 root root 12M Feb 16 2019 initrd.img-4.20.6
-rw-r--r-- 1 root root 179K Jan 28 2016 memtest86+.bin
-rw-r--r-- 1 root root 181K Jan 28 2016 memtest86+.elf
-rw-r--r-- 1 root root 181K Jan 28 2016 memtest86+_multiboot.bin
-rw-r--r-- 1 root root 0 Jul 17 2018 retpoline-4.15.0-29-generic
-rw-r--r-- 1 root root 3.9M Jul 17 2018 System.map-4.15.0-29-generic
-rw-r--r-- 1 root root 3.3M Feb 16 2019 System.map-4.20.6
-rw-r--r-- 1 root root 7.9M Jul 25 2018 vmlinuz-4.15.0-29-generic
-rw-r--r-- 1 root root 5.8M Feb 16 2019 vmlinuz-4.20.6
user@user-VirtualBox:/boot$

```

Part B
 81 46 f8 83 00 00 00

OpCode	MOD	REG	RM	Displacement	Constant
81		46		f8	83
1000 0001	01	000	110	1111 1000	1000 0011

If opcode high-order bit set to 1, then instruction has an immediate constant.

If opcode lowest-order bit set to 1, then 8-bit operands

REG is 000 is eax

MOD is 01 means One-byte signed displacement follows addressing mode byte(s).

MOD R/M

01 110 = [esi + disp8]

The code will looks like.

add eax, [esi + disp8]

Adding the displacement

add eax, [esi + f8]

Include the constant

add 83, [esi + f8]

What is the 32-bit instruction encoding for the following instruction:

addl \$13, %edx

Size of the operands are 32 bits

OpCode	MOD	REG	R/M	Constant
0	8			13
0000 0000	00	010	000	0001 0011

What would be the 64-bit encoding for the same instruction above?

addl \$13, %rdx

REX Prefix	OpCode	MOD	REG	R/M	Constant
	0	8			13
01001000	0000 0000	00	010	000	0001 0011

Refence websites

https://wiki.osdev.org/X86-64_Instruction_Encoding

<https://www-user.tu-chemnitz.de/~heha/viewchm.php/hs/x86.chm/x86.htm>