



2023 Edge Cloud Lab

Edge 팀: 윤성빈, 김태현

Goal: Edge 컴퓨팅에서 Ransomware로부터 데이터 보호를 위한 스토리지 시스템 구현하기

Details:

- 1~2주차: Linux kernel module 만들어 보기
- 3~4주차: Linux KVM의 virtio 해킹
- 5~6주차: virtio 모듈을 수정하여 가상머신의 Disk IO를 interpose하여 surreal DB위에 time-series 데이터로 저장하고 주어진 timestamp 시간 이전으로 virtual disk 상태를 되돌리는 기능 구현
- 7~8주차: VM 안에 샘플 ransomware를 설치하여 실제로 공격 감행 후 ransomware 공격으로부터 회복하는 기능을 통해 복구하는 데모하기, 결과 리포트 작성 및 작성한 코드 github에 업로드

Contents

- Linux Kernel
- Linux Dual Boot
- Linux Kernel Compile
 - Trouble Shooting
- Directory Contents
- Linux Kernel Module Build
- Character Device Driver
 - Trouble Shooting
- Linux KVM
- Linux KVM Installation
 - Trouble Shooting
- VM Installation

- ioctl Function
 - Advanced Character Device Driver
 - Trouble Shooting
 - Block Device Driver
 - Trouble Shooting
 - Source
-

Linux Kernel

OS(Operating System)

운영체제, hardware와 software resource를 관리하는 프로그램

OS(Operating System)의 3대 요소

Kernel

- OS의 핵심, 하드웨어를 통제하는 프로그램

Shell

- User interface

System Program

- Kernel과 Shell을 작동하게 해주는 프로그램

Linux는 OS(운영체제)인가?

대충 말하면 Yes. 엄밀히 따지자면

“Linux = Kernel”

⇒ Linux Kernel + 우리가 만든 Shell & System Program으로 새로 만들어진 배포판 = Linux Distribution

⇒ 그 예로 Redhat 계열의 CentOS, Fedora / Debian 계열의 Ubuntu, Gentoo 등이 있다.

⇒ Linux를 잘한다 = Linux 시스템을 잘 다룬다 = 운영 체제를 잘 안다

Linux Kernel?

리눅스 운영체제의 주요 구성 요소이자 컴퓨터 하드웨어와 프로세스를 잇는 핵심 인터페이스. 이 두가지 관리 리소스 사이에서 최대한 효과적으로 통신하게함.

커널의 기능?

1. 메모리 관리: 메모리가 어디에서 무엇을 저장하는데 얼마나 사용되는지 추적
2. 프로세스 관리: 어느 프로세스가 CPU를 언제 얼마나 오랫동안 사용할지를 결정
3. 장치 드라이버: 하드웨어와 프로세스 사이에서 중재자/인터프리터의 역할을 수행
4. 시스템 호출 및 보안: 프로세스의 서비스 요청을 수신

OS 내에서 커널의 위치?

리눅스 시스템은 다음 3개 Layer로 구성되어 있음.

1. 하드웨어
2. Linux 커널
3. 사용자 프로세스

Linux Dual Boot

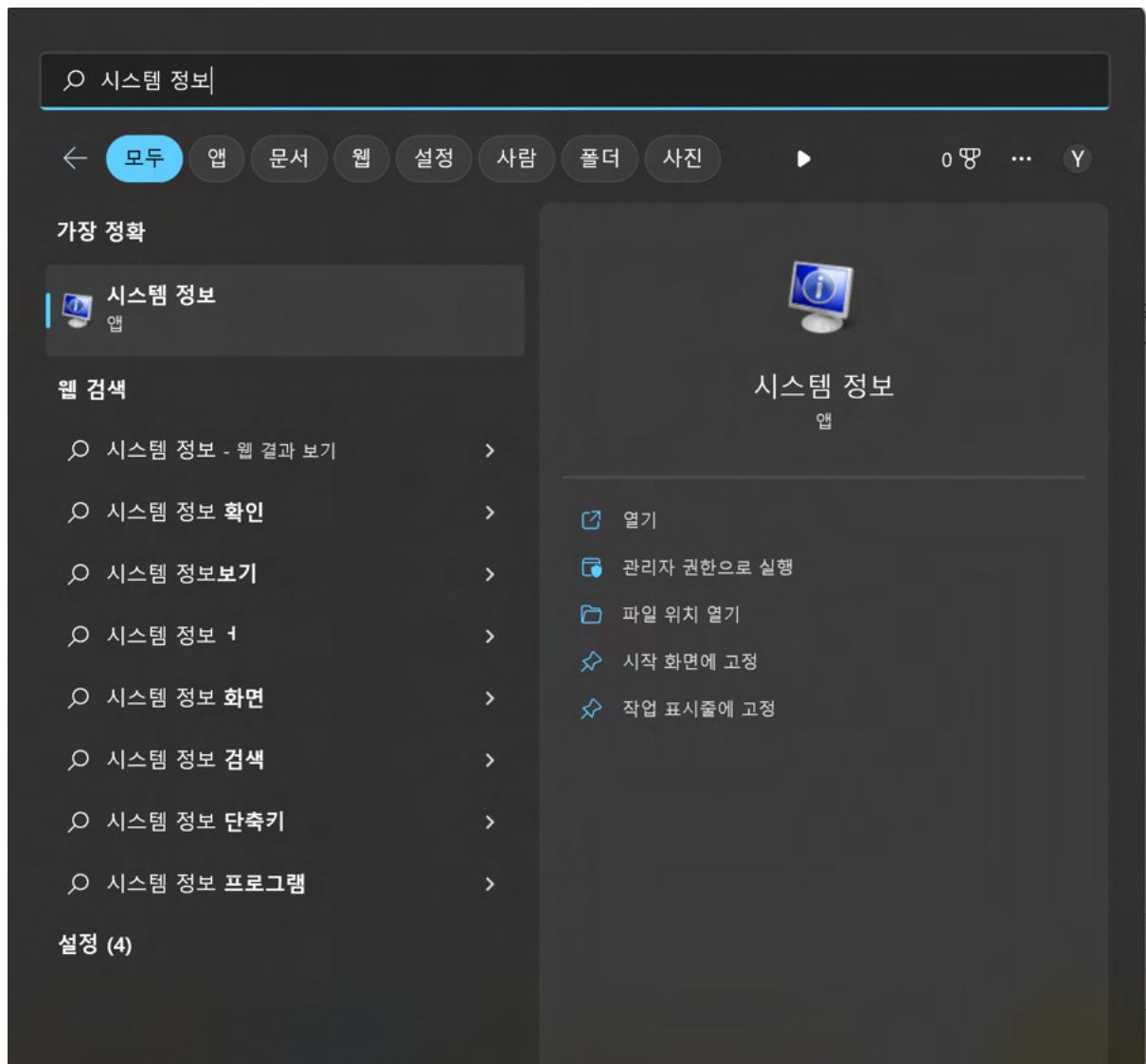
설치 전, 준비할 요소가 있다.

1. 최소 4GB 이상의 USB
2. 최소 40GB 이상의 드라이브 용량 (이 이상 넉넉하게 잡아야 한다. 추후 설명할 Linux Kernel 컴파일에 기본적으로 20GB 이상 쓰이기 때문이다.)
3. Rufus 프로그램
4. Ubuntu 22.04 LTS.iso 파일
5. Wifi가 아닌 LAN(이더넷) 연결(Ubuntu가 Wifi를 못잡는 경우가 있다.)

Step 1)

설정에서 BitLocker 암호화가 설정되어 있다면 해제한다.(설정되어 있으면 설치 후 Windows로 들어갈 경우 복구 키를 요구하며 잠겨버린다.)

PC의 시스템 정보(system information)에 들어간다.



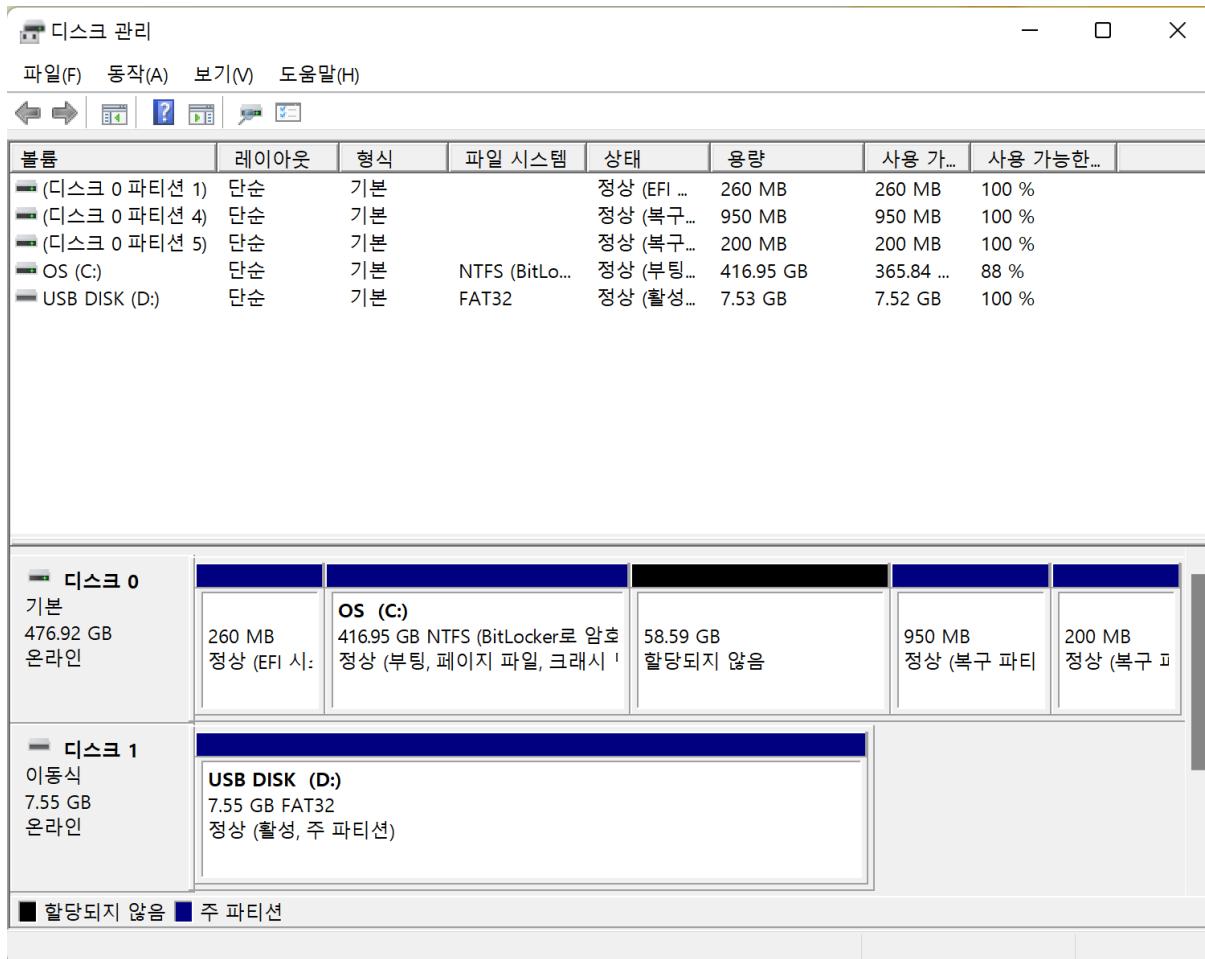
BIOS 모드가 UEFI로 되어있는지 확인한다. (UEFI인지 아닌지에 따라 추후 방법이 다르므로 확인 필수)

BIOS 모드

UEFI

확인 후, 디스크 관리에 들어가 C:에 오른쪽 클릭 후, 볼륨 축소하기 → 할당할 용량을 적는다. 60GB의 경우 60000.

이 후 아래와 같이 할당되지 않은 파티션이 생긴 것을 볼 수 있다.



Ubuntu 홈페이지에서 ubuntu 22.04 LTS iso 파일을 다운로드 받는다.

Get Ubuntu | Download | Ubuntu | Rufus - 간편하게 부팅 가능한 USB 드라이브 | +

ubuntu.com/download#download

CANONICAL

ubuntu Enterprise Developer Community Download Search Q Sign in

Ubuntu Desktop ›
Download Ubuntu desktop and replace your current operating system whether it's Windows or Mac OS, or, run Ubuntu alongside it.
22.04 LTS 22.10

Ubuntu Server ›
The most popular server Linux in the cloud and data centre, you can rely on Ubuntu Server and its five years of guaranteed free upgrades.
Get Ubuntu Server

Ubuntu for IoT ›
Are you a developer who wants to try snappy Ubuntu Core or classic Ubuntu on an IoT board?
Raspberry Pi Intel IoT platforms Intel NUC KVM Qualcomm Dragonboard 410c Intel IEI TANK 870 AMD-Xilinx Evaluation kits & SOMs RISC-V platforms

Ubuntu Cloud ›
Use Ubuntu optimised and certified server images on most major clouds.
Get started on Amazon AWS, Microsoft Azure, Google Cloud Platform and more...
Download cloud images for local development and testing

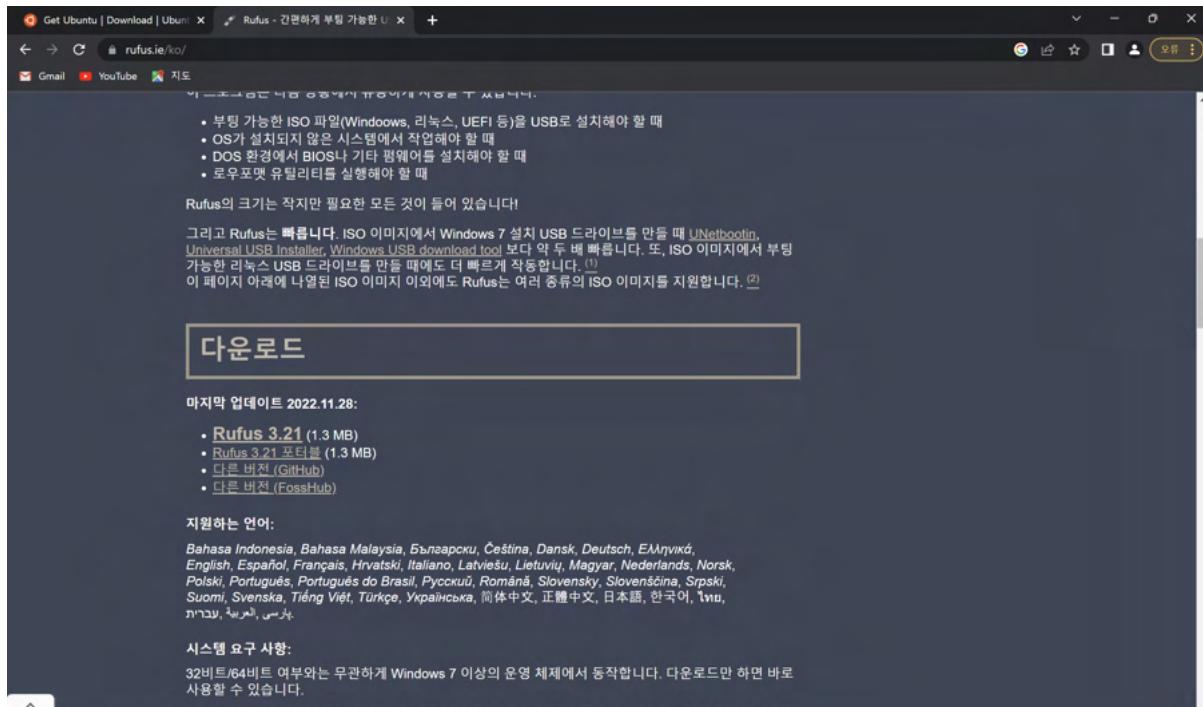
TUTORIALS
If you are already running Ubuntu - you can upgrade with the Software Updater
Burn a DVD on Ubuntu, macOS, or Windows. Create a bootable USB stick on Ubuntu, macOS, or Windows
Installation guides for Ubuntu Desktop and Ubuntu Server

READ THE DOCS
Read the official docs for Ubuntu Desktop, Ubuntu Server, and Ubuntu Core

OTHER WAYS TO DOWNLOAD
Ubuntu is available via BitTorrents and via a minimal network installer that allows you to customise what is installed, such as additional languages. You can also find older releases.

UBUNTU FLAVOURS
Find new ways to experience Ubuntu, each with their own choice of default applications and settings.
Kubuntu Ubuntu MATE
Lubuntu Ubuntu Studio
Ubuntu Budgie Ubuntuuntu Kylin

Rufus 홈페이지에서 Rufus 3.21 프로그램을 다운로드 받아 설치한다.



USB를 PC에 꽂고 Rufus 프로그램을 실행한다.

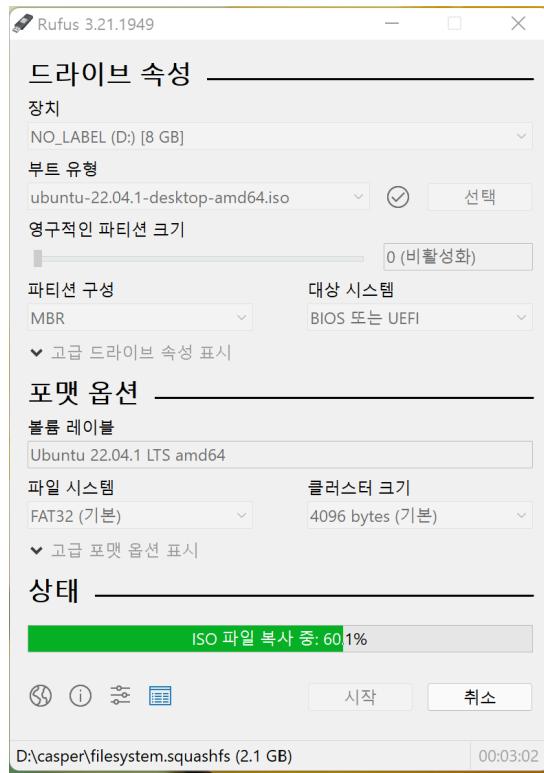
USB 장치와 부트 유형에 iso파일을 선택한다.

BIOS 모드가 UEFI이므로 파티션 구성의 경우 GPT, 대상 시스템은 UEFI(CSM 지원 안함)를 선택한다.

(BIOS 모드가 Legacy인 경우에는 MBR를 선택한다)

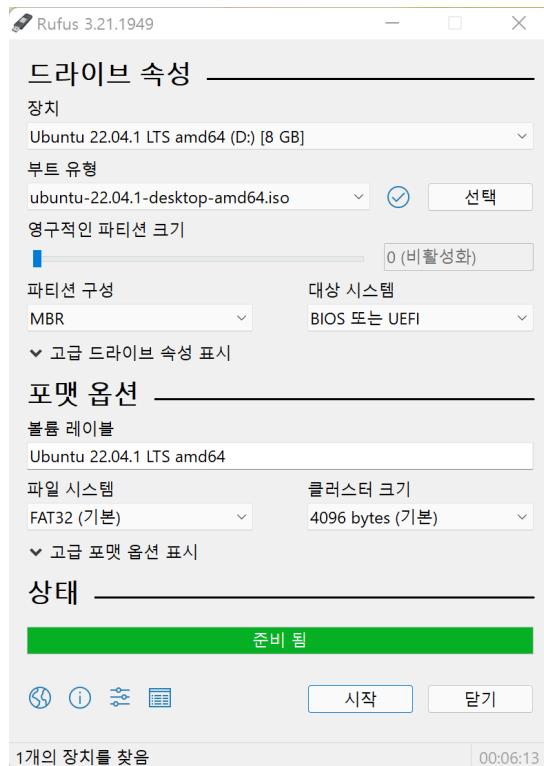
나는 설치 당시 어떤 글을 보고 MBR로 설치를 했는데, BIOS 모드가 UEFI이므로 GPT로 하는 것이 맞다.(어떻게 성공한건지 모르겠다.)

파일 시스템은 FAT32, 클러스터 크기는 기본으로 두고 시작 버튼을 누른다.



ISO 이미지 모드로 쓰기(권장)이라는 메세지가 뜨면 확인을 누르고 진행한다.

USB의 전송 속도에 따라 진행 속도는 다를 수 있다.

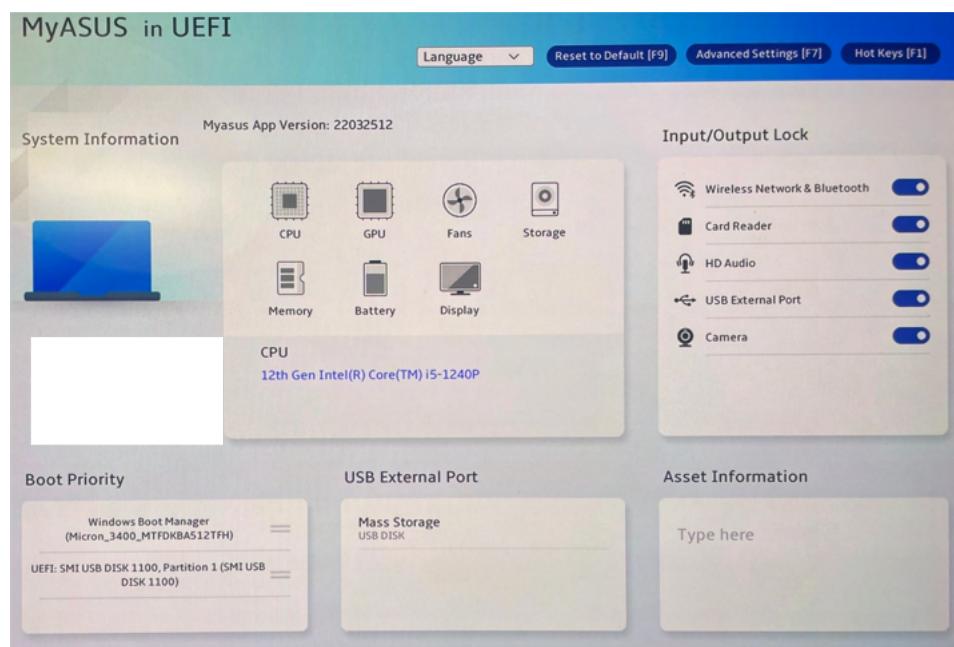


완료되면 닫기를 누른다.

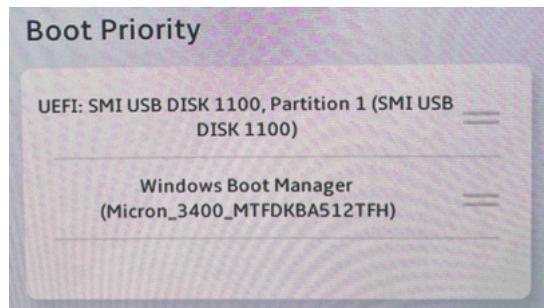
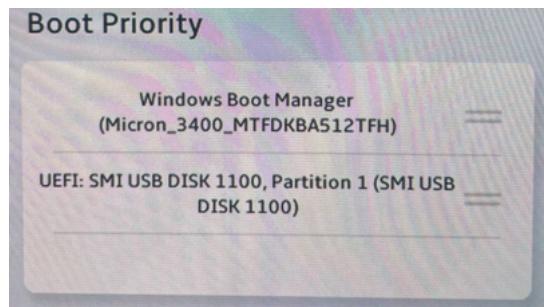
Step2)

창을 모두 닫고 재부팅을 한다.

PC가 다시 켜질 때 BIOS 창으로 진입한다.(ASUS의 경우 부팅될 때 F2를 연속으로 누른다. 키의 경우, 제조사마다 약간 다를 수 있다.)



Boot Priority의 부팅 순서를 아래와 같이 USB가 먼저 오도록 끌어올려 변경해준다.



이 때, Secure boot가 설정되어 있다면 해제해주는 것이 좋다. 해제해야 외부에서 받은 인증되지 않은 OS를 설치하는 것이 가능하다.

Advanced Settings로 들어가 Security에 있는 Secure Boot을 Enabled → Disabled로 변경해준다.

Save & Exit를 해준다.

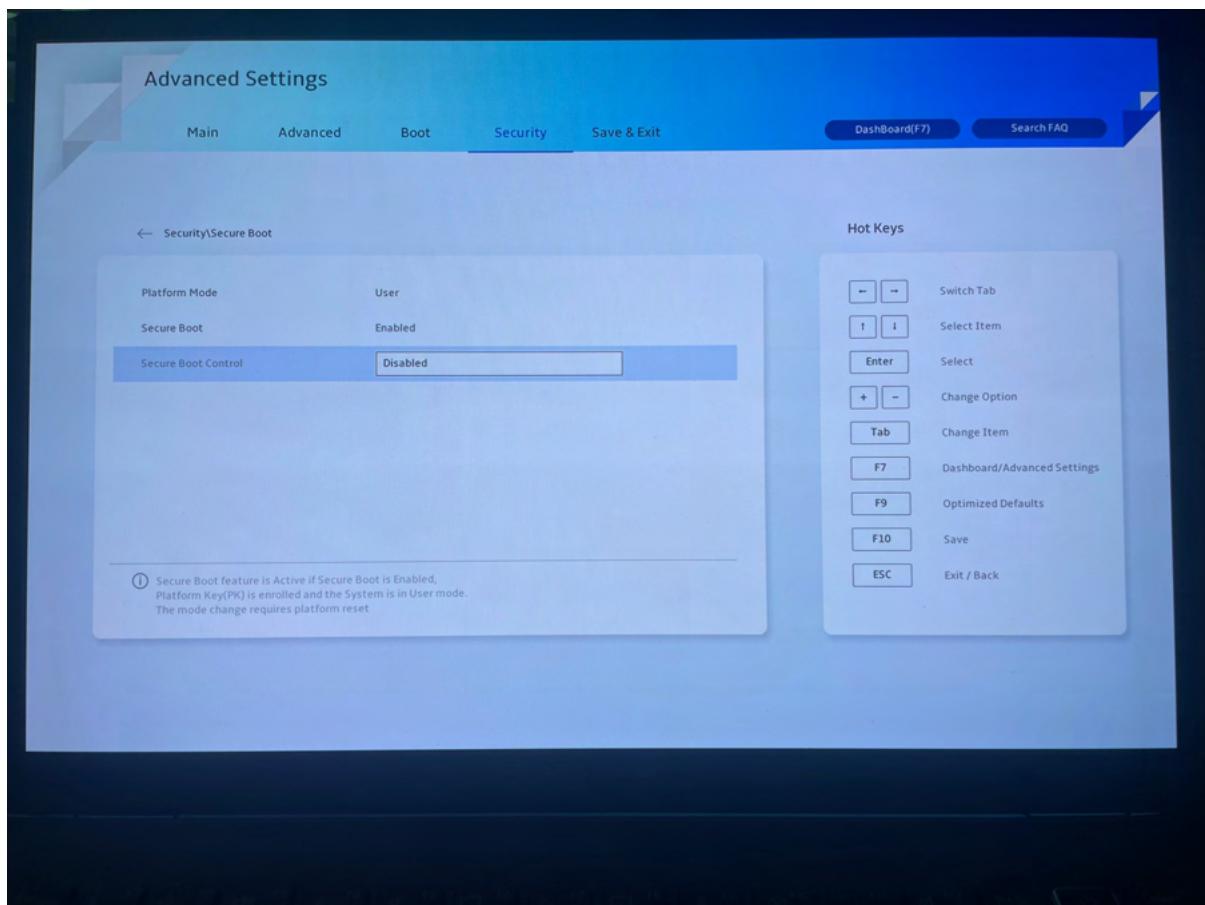
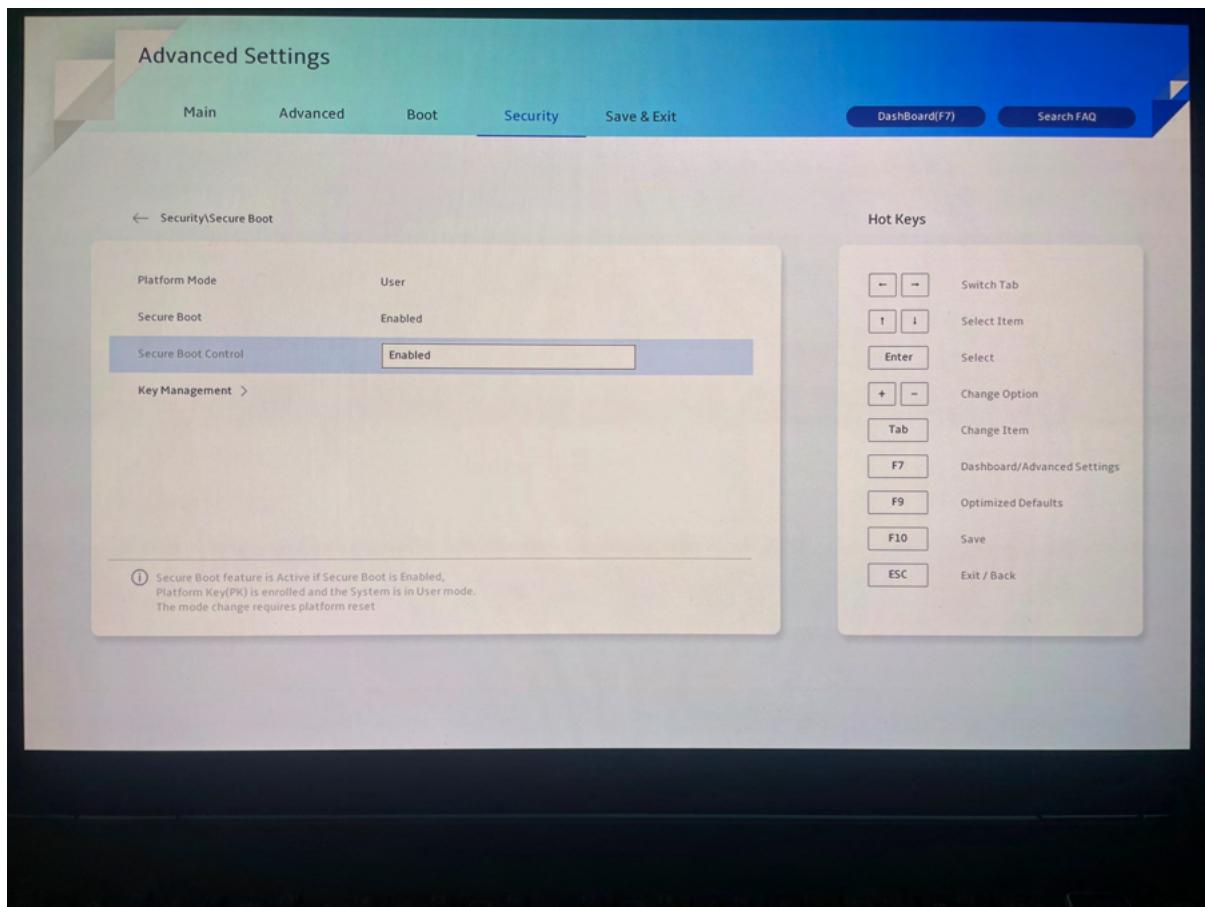
The screenshot shows the 'Advanced Settings' interface with the 'Security' tab selected. On the left, there's a list of password-related settings:

- Setup Utility Password
- Administrator Password Status: NOT INSTALLED
- User Password Status: NOT INSTALLED
- Administrator Password -Not Installed
- User Password -Not Installed
- I/O Interface Security >
- Secure Boot >

A small note at the bottom says: ⓘ Secure Boot configuration.

On the right, there's a 'Hot Keys' sidebar with the following mappings:

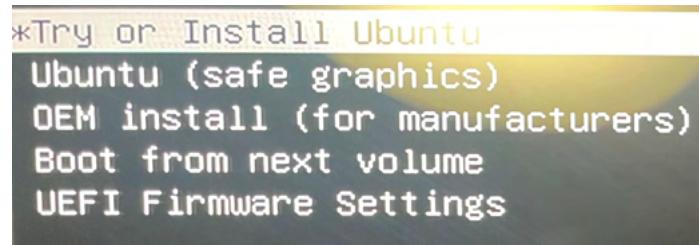
Hot Key	Action
[←] [→]	Switch Tab
f	Select Item
Enter	Select
[+]	Change Option
Tab	Change Item
F7	Dashboard/Advanced Settings
F9	Optimized Defaults
F10	Save
ESC	Exit / Back



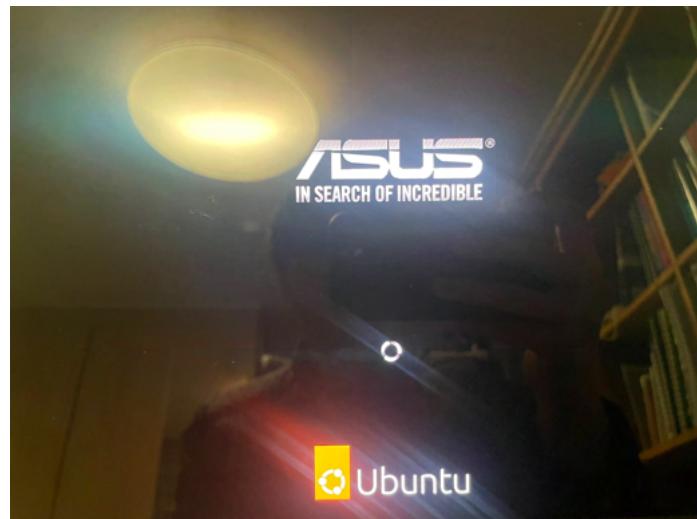
Save & Exit를 해주면 자동으로 재부팅이 된다.

이 때, grub 창이 뜬다.

Try or Install Ubuntu를 선택해준다.



Ubuntu 로고가 뜨면서 설치가 로딩이 이어진다.



Step3)

화면이 바뀌며 Try Ubuntu와 Install Ubuntu가 뜨는데, Install 클릭 → 언어설정(English로 할 것) →

Normal Installation,

Download updates while installing ubuntu,

Install third-party software for graphics and wifi hardware and additional media formats에 체크(비밀번호 부분은 체크 해제) → Continue

설치타입 설정 → Something else 선택

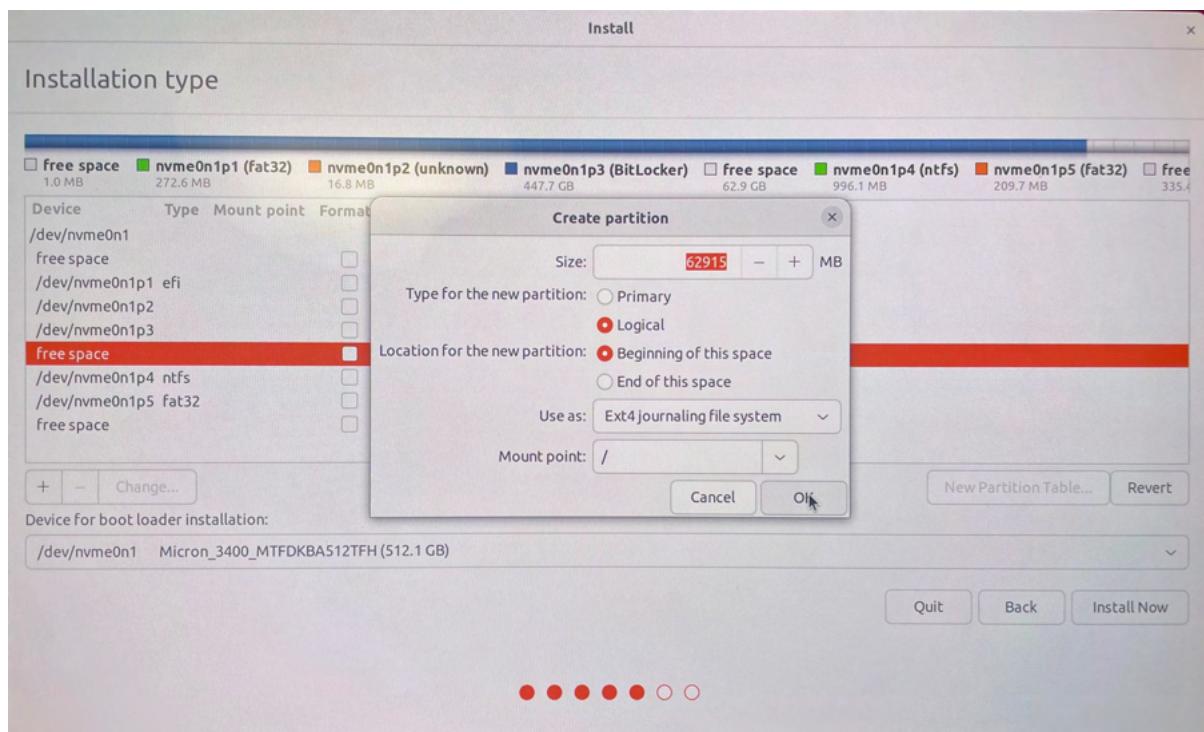
Swap partition 만들기

RAM 용량이 충분하면 굳이 설정할 필요 없다고는 하지만, 커널 컴파일 시(!! Trouble Shooting 문서 참조) RAM 용량을 크게 잡아먹는 일이 생길 수 있어 설정해주는 것이 좋을 듯 하다...

free space → + 클릭 → Size는 PC의 RAM 크기의 두배 권장 → Logical 선택 → Beginning of this space 선택
→ Swap area 선택 → Ok

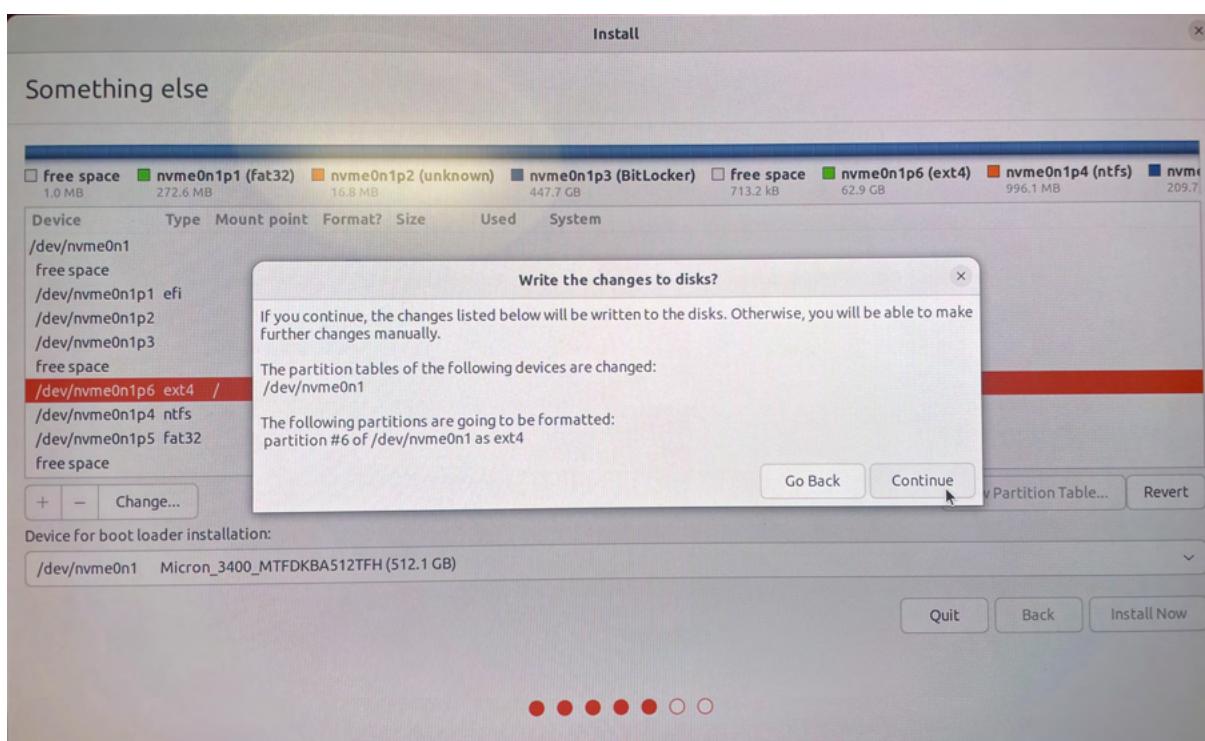
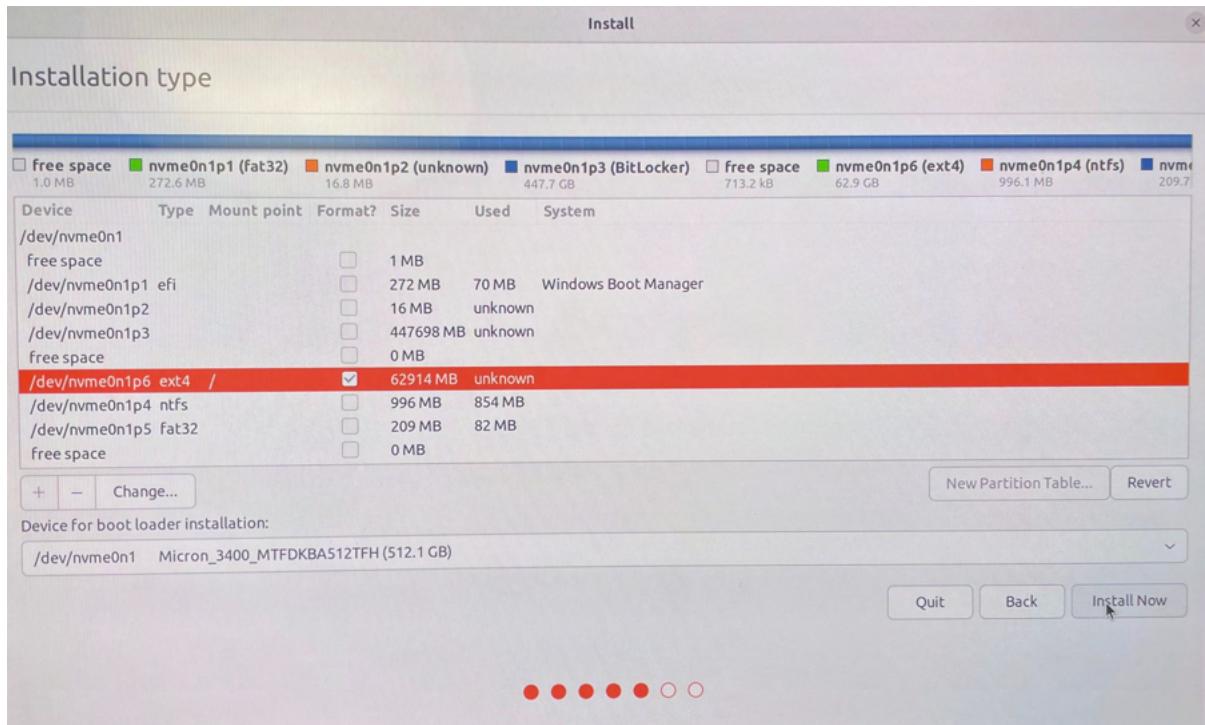
Root partition 만들기

free space → + 클릭 → Size 그대로 → 주파티션이 다 쓰이고 있다면 Logical, 남은 경우에 Primary 선택 → Beginning of this space 선택 → Ext4 journaling file system → / 선택 → Ok

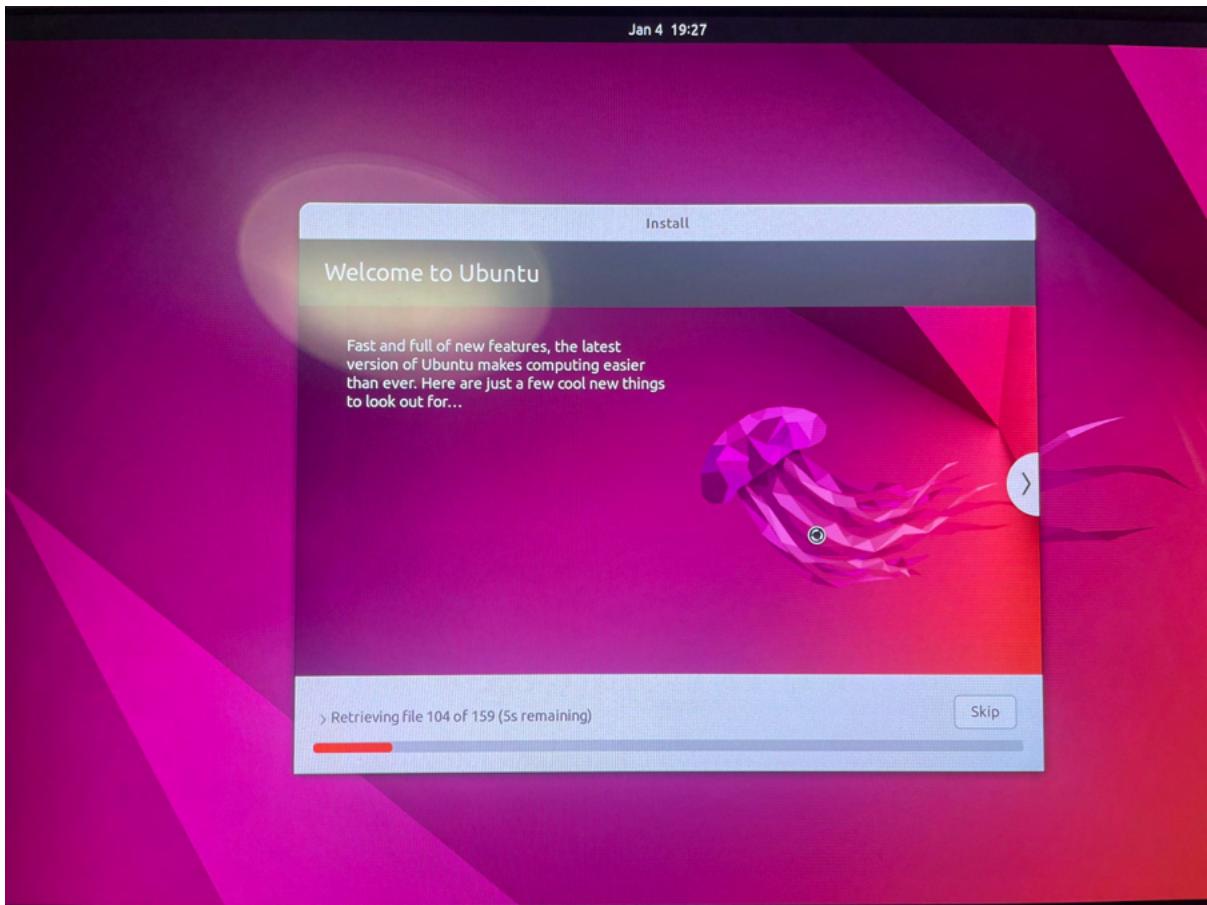


아래와 같이 space가 할당된 것을 볼 수 있다.

체크하고 Install Now를 클릭하고 Continue를 클릭한다.



Install 창이 뜨면서 설치가 진행된다.

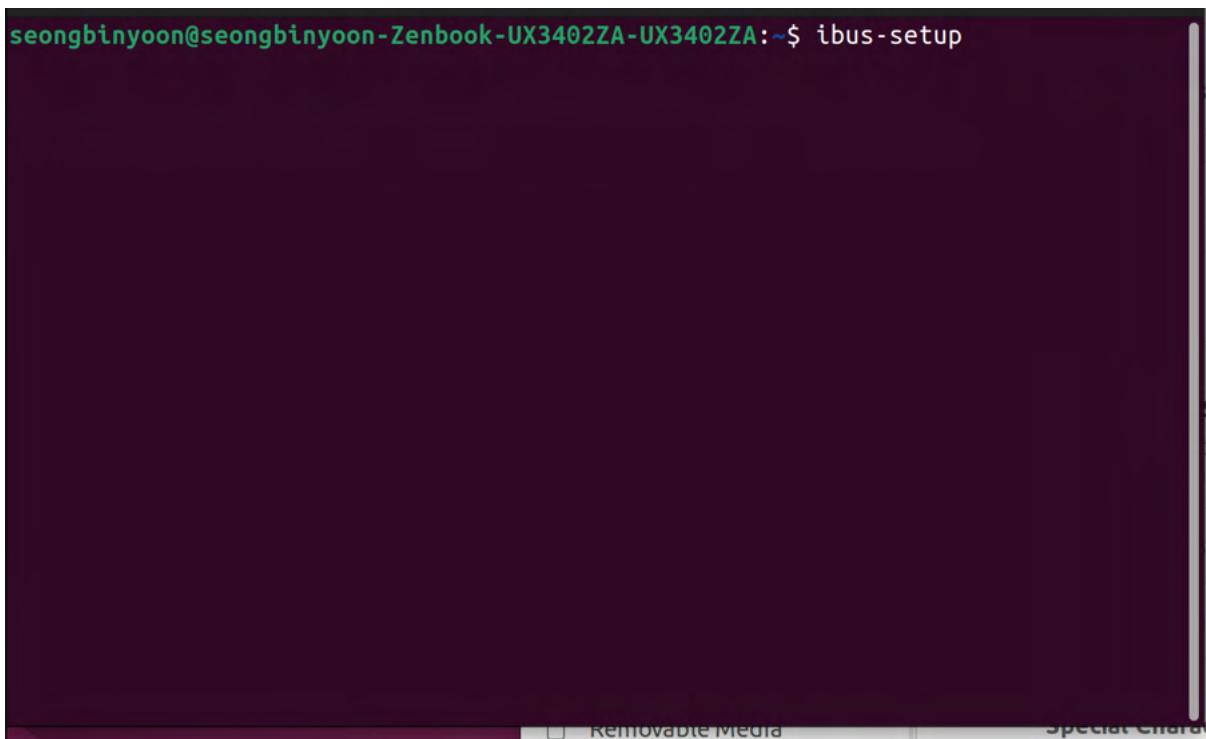


Step4)

설치가 완료된 ubuntu는 키보드에서 한글을 사용할 수 없다. 사용자가 설정해 주어야 한다.

터미널을 열고 ~\$ ibus-setup을 입력

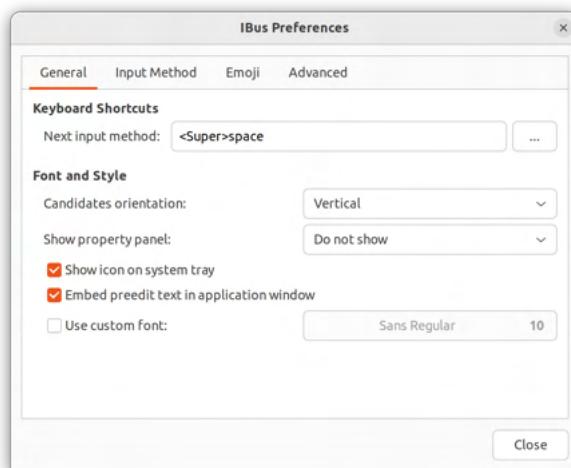
```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ ibus-setup
```

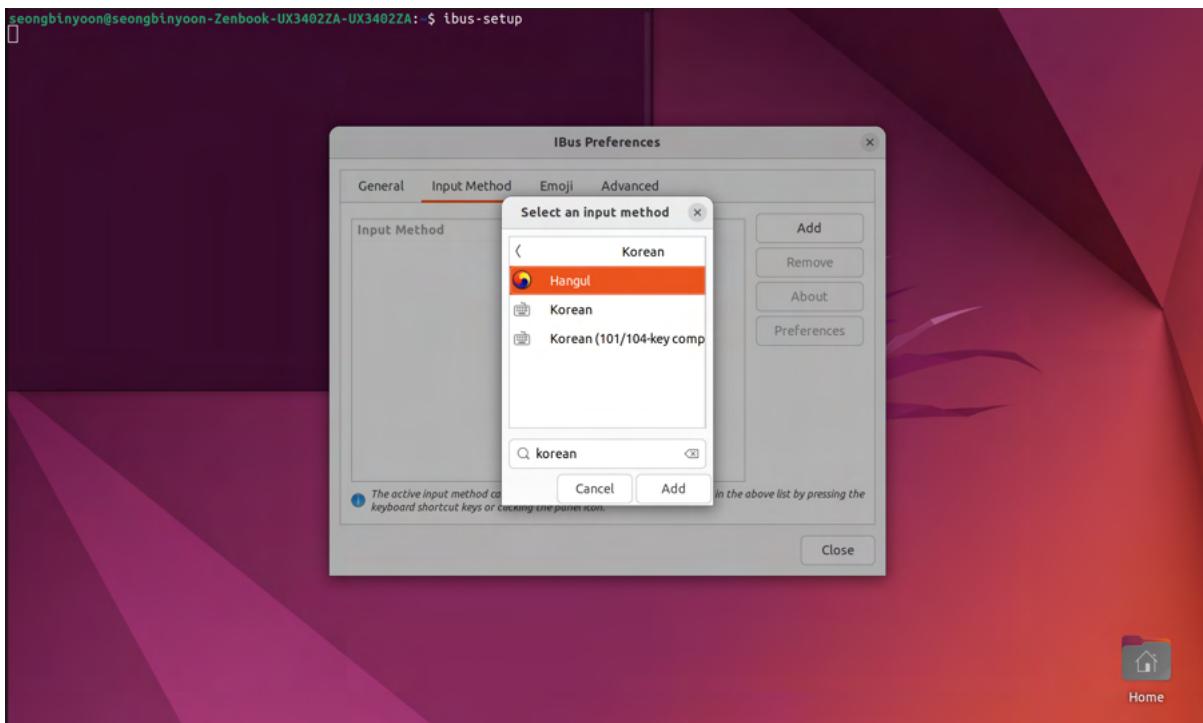


iBus Preferences 창이 뜨는데, Input Method → Add → Korean → Hangul → Add 선택

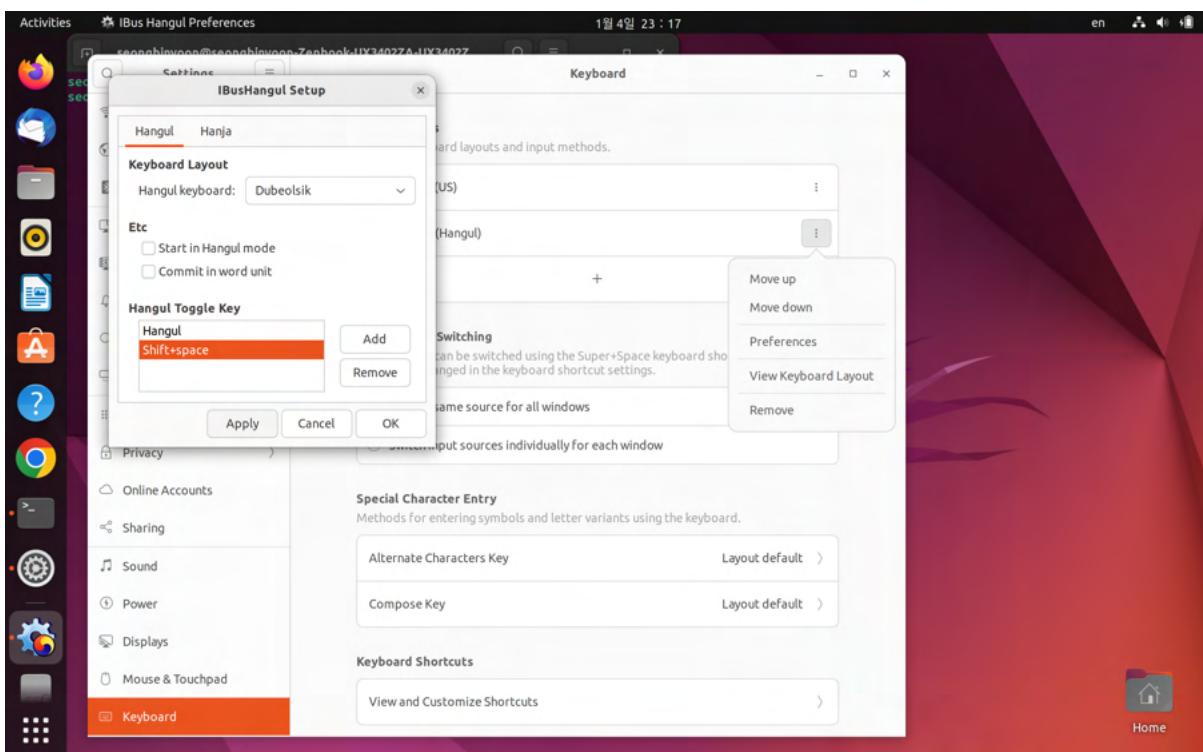
(Hangul 외의 korean은 키보드 한글을 지원하지 않는다.)

만약 Hangul이 뜨지 않는다면 터미널에서 ~\$ sudo apt-get update를 통해 패키지 업데이트 및 재부팅을 해준다.





설정의 Keyboard로 가서 English(US)를 삭제하고 Korean(Hangul)의 오른쪽을 클릭 → Preferences → 원하는 키를 설정 → OK



Step5)

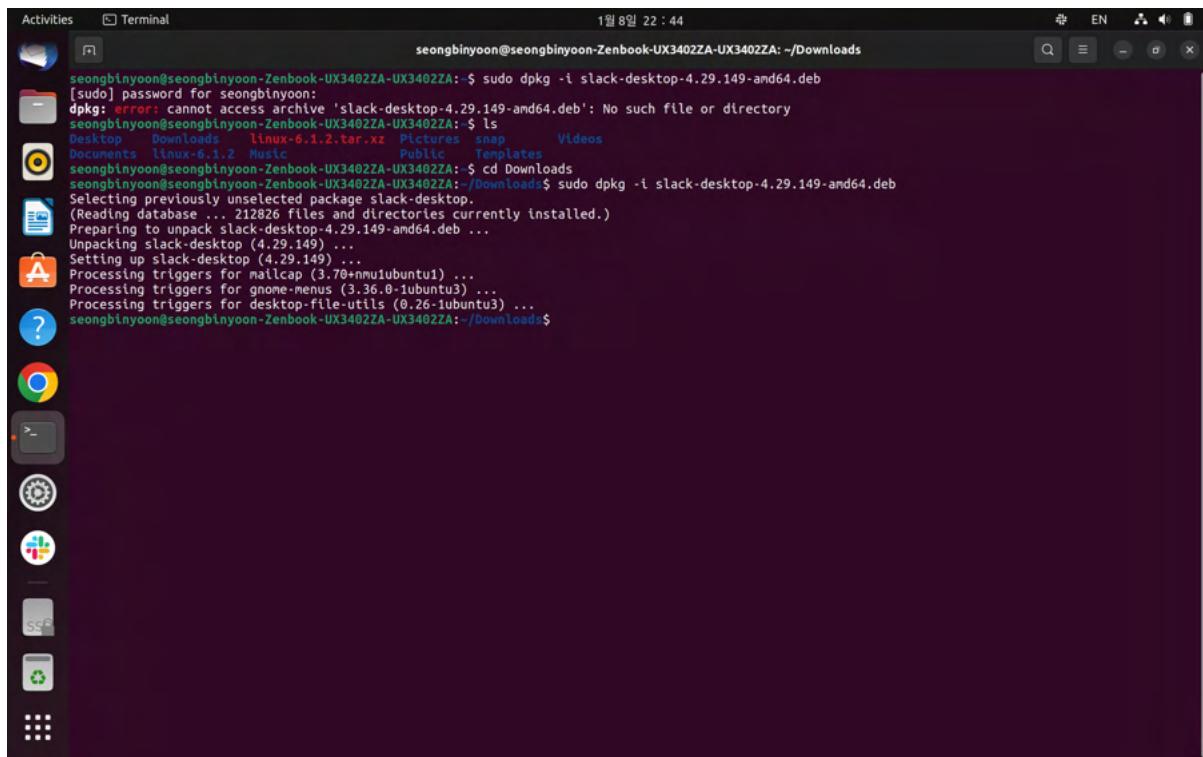
추후 필요한 패키지를 적절한 명령어를 통해 설치

ex) Slack 설치 방법

1. Chrome에 들어가 리눅스용 deb 파일을 다운로드 받는다.
2. 다운로드 디렉토리에 들어가 다음 명령어를 입력한다.

```
~$ sudo dpkg -i slack-desktop-4.29.149.amd64.deb
```

3. 메뉴에서 실행 또는 터미널에 slack 입력

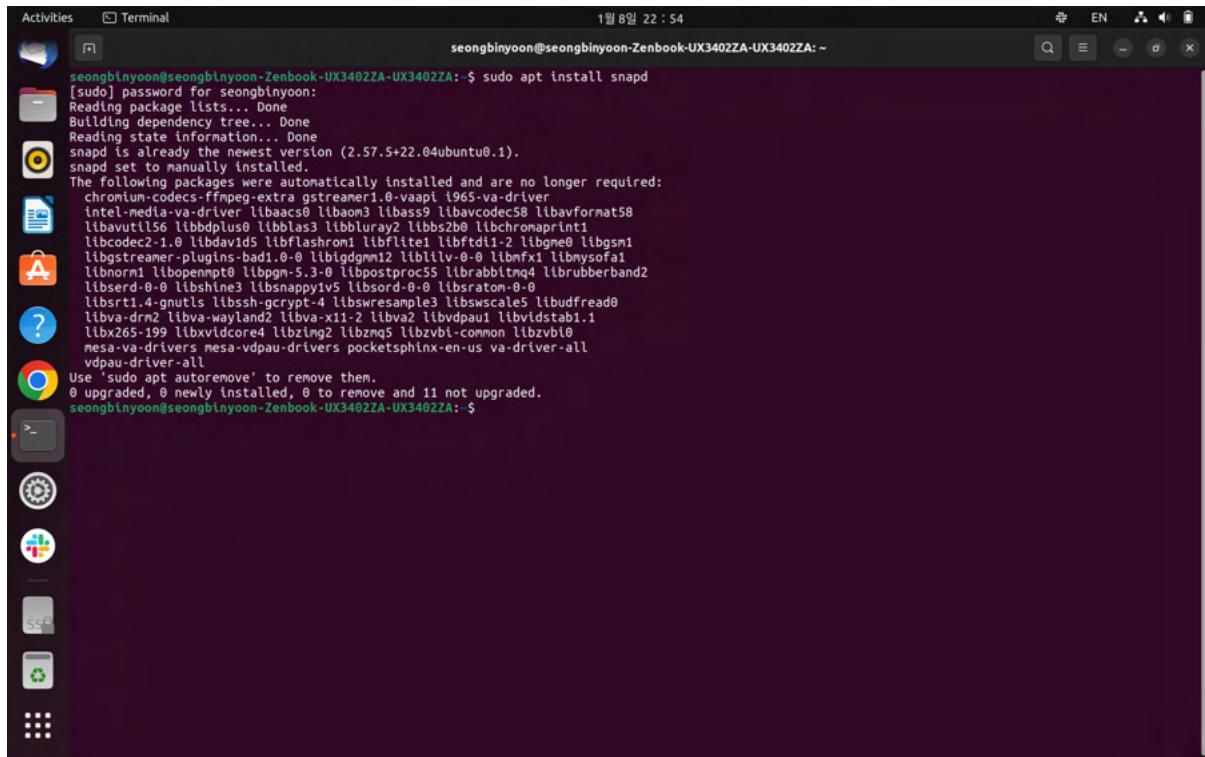


ex) Notion 설치 방법

Notion은 아직 Linux에서 지원하지 않는다. 따라서 패키지 설치 프로그램 snap을 이용해 install한다.

1. 패키지 설치 프로그램 snap을 설치한다.

```
~$ sudo apt install snapd
```



Activities Terminal 1월 8일 22:54 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~

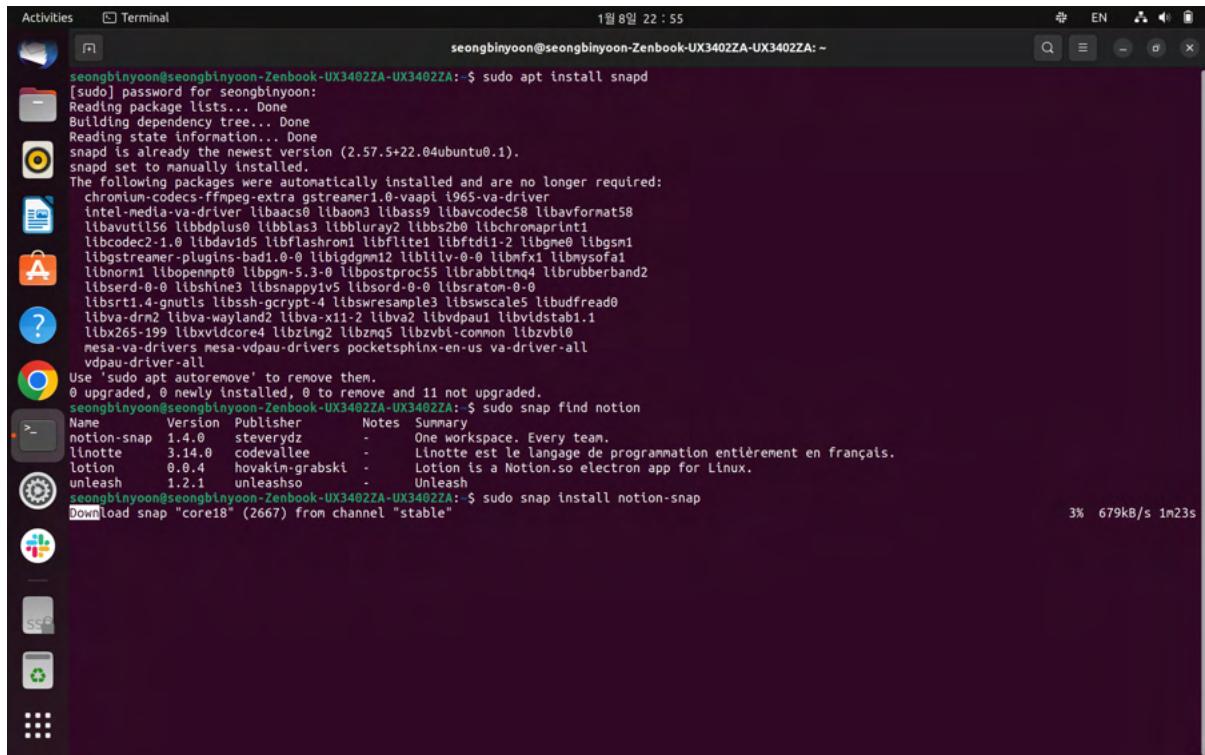
```
[sudo] password for seongbinyoon:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
snapd is already the newest version (2.57.5+22.04ubuntu0.1).  
snapd set to manually installed.  
The following packages were automatically installed and are no longer required:  
chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver  
intel-media-va-driver libaaacs0 libao0 libass9 libavcodec58 libavformat58  
libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1  
libcodec2-1.0 libdavids libflashrom1 libflite1 libftdi1-2 libgme0 libgsf1  
libgstreamer-plugins-bad1.0-0 libigdmm12 libiliiv-0-0 libmfx1 libmysofai  
libnorm1 libopenmp70 libppm-5.3-0 libpostproc55 librabbitmq4 librubberband2  
libserd-0-0 libshine3 libsnappy1v5 libsrord-0-0 libsratom-0-0  
libstl-4-gnutls libssh-gcrypt-4 libswresample3 libwscale5 libudfread0  
libva-drm2 libva-wayland2 libva-x11-2 libva2 libvdpa1 libvidstab1.1  
libx65-199 libxvidcore4 libzimg2 libzmq5 libzvbi-common libzvbi0  
mesa-va-drivers mesa-vdpau-drivers pocketphinx-en-us va-driver-all  
vdpau-driver-all  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.  
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$
```

2. snap 명령어로 snap에서 notion을 찾는다.

```
~$ sudo snap find notion
```

3. snap 명령어로 찾은 notion-snap을 설치한다.

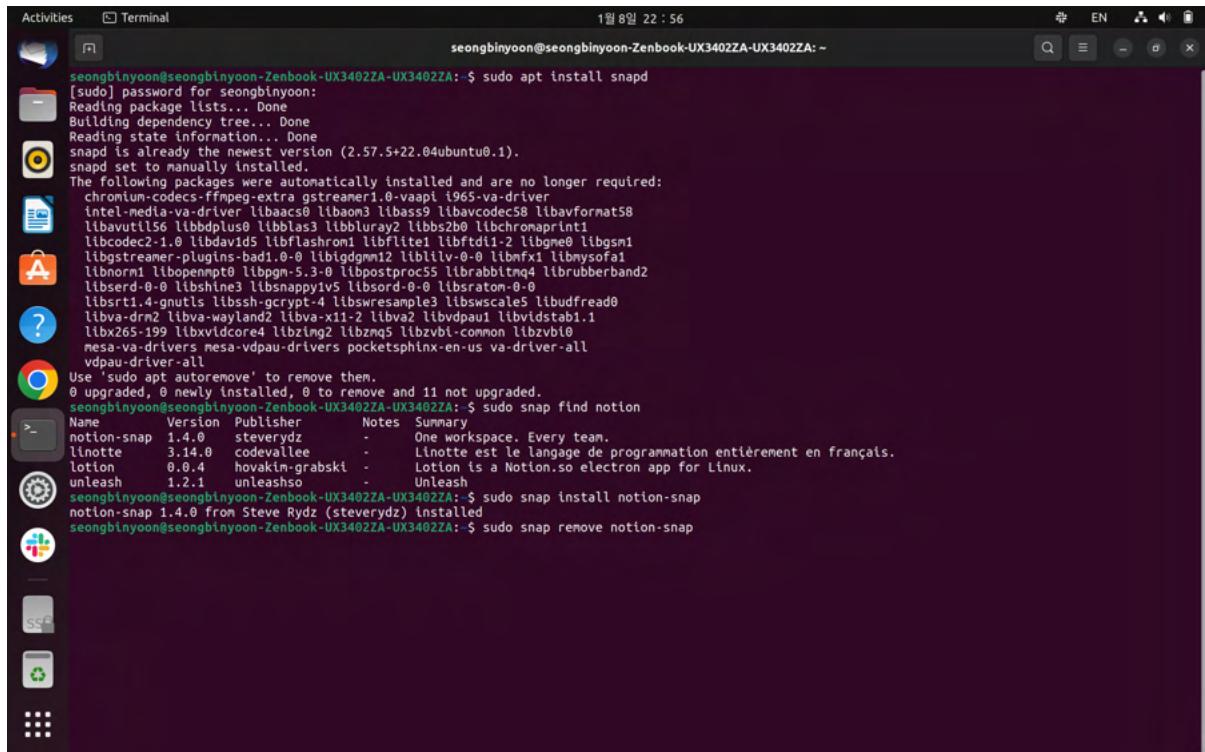
```
~$ sudo snap install notion-snap
```



```
Activities Terminal 1월 8일 22:55
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ sudo apt install snap
[sudo] password for seongbinyoon:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
snapd is already the newest version (2.57.5+22.04ubuntu0.1).
snapd set to manually installed.
The following packages were automatically installed and are no longer required:
chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver
intel-media-va-driver libaaacs0 libaaom3 libass9 libavcodec58 libavformat58
libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1
libcodec2-1.0 libdavids libflashrom1 libflite1 libftdi1-2 libgme0 libgsm1
libgstreamer-plugins-bad1.0-0 libigdmm12 libiliiv-0.0 libmfx1 libmysof1
libnorm1 libopenmp8 libppm-5.3-0 libpostproc55 librabbitmq4 librubberband2
libserd-0-0 libsshine3 libsnappy1v5 libsrord-0-0 libsratom-0-0
libstti.4-gnutls libssh-gcrypt-4 libswresample3 libwscale5 libudfread0
libva-drm2 libva-wayland2 libva-x11-2 libva2 libvdpaui libvidstab1.1
libx265-199 libxvidcore4 libzimg2 libzmq5 libzvbi-common libzvbi0
mesa-va-drivers mesa-vdpau-drivers pocketphinx-en-us va-driver-all
vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ sudo snap find notion
Name Version Publisher Notes Summary
notion-snap 1.4.0 steverydz - One workspace. Every team.
linotte 3.14.0 codevallee - Linotte est le langage de programmation entièrement en français.
lotion 0.0.4 hovakim-grabski - Lotion is a Notion.so electron app for Linux.
unleash 1.2.1 unleashso - Unleash
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ sudo snap install notion-snap
Download snap "core18" (2667) from channel "stable"
3% 679kB/s 1m23s
```

4. 오류가 생기거나 문제가 생기면 아래 명령어로 삭제 후 다시 설치한다.

```
~$ sudo snap remove notion-snap
```



```
Activities Terminal 1월 8일 22:56
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ sudo apt install snap
[sudo] password for seongbinyoon:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
snapd is already the newest version (2.57.5+22.04ubuntu0.1).
snapd set to manually installed.
The following packages were automatically installed and are no longer required:
chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver
intel-media-va-driver libaaacs0 libaaom3 libass9 libavcodec58 libavformat58
libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1
libcodec2-1.0 libdavids libflashrom1 libflite1 libftdi1-2 libgme0 libgsm1
libgstreamer-plugins-bad1.0-0 libigdmm12 libiliiv-0.0 libmfx1 libmysof1
libnorm1 libopenmp8 libppm-5.3-0 libpostproc55 librabbitmq4 librubberband2
libserd-0-0 libsshine3 libsnappy1v5 libsrord-0-0 libsratom-0-0
libstti.4-gnutls libssh-gcrypt-4 libswresample3 libwscale5 libudfread0
libva-drm2 libva-wayland2 libva-x11-2 libva2 libvdpaui libvidstab1.1
libx265-199 libxvidcore4 libzimg2 libzmq5 libzvbi-common libzvbi0
mesa-va-drivers mesa-vdpau-drivers pocketphinx-en-us va-driver-all
vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ sudo snap find notion
Name Version Publisher Notes Summary
notion-snap 1.4.0 steverydz - One workspace. Every team.
linotte 3.14.0 Codevallee - Linotte est le langage de programmation entièrement en français.
lotion 0.0.4 hovakim-grabski - Lotion is a Notion.so electron app for Linux.
unleash 1.2.1 unleashso - Unleash
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ sudo snap install notion-snap
notion-snap 1.4.0 from Steve Rydz (steverydz) installed
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ sudo snap remove notion-snap
```

Linux Kernel Compile

참고한 소스는 다음과 같다.

How To Build Linux Kernel {Step-By-Step} | phoenixNAP KB

Introduction The Linux Kernel is the foundation of all the Linux distributions. The kernel is responsible for communication between hardware and software and the allocation of available resources. All Linux distributions are based on a predefined kernel. But, if you

🌐 <https://phoenixnap.com/kb/build-linux-kernel>

phoenixNAP
GLOBAL IT SERVICES

How to Build Linux Kernel

/* 커널 버전 업그레이드만 해당된다. 다운그레이드는 시도해본 결과 커널 패닉의 가능성성이 다분하다. */

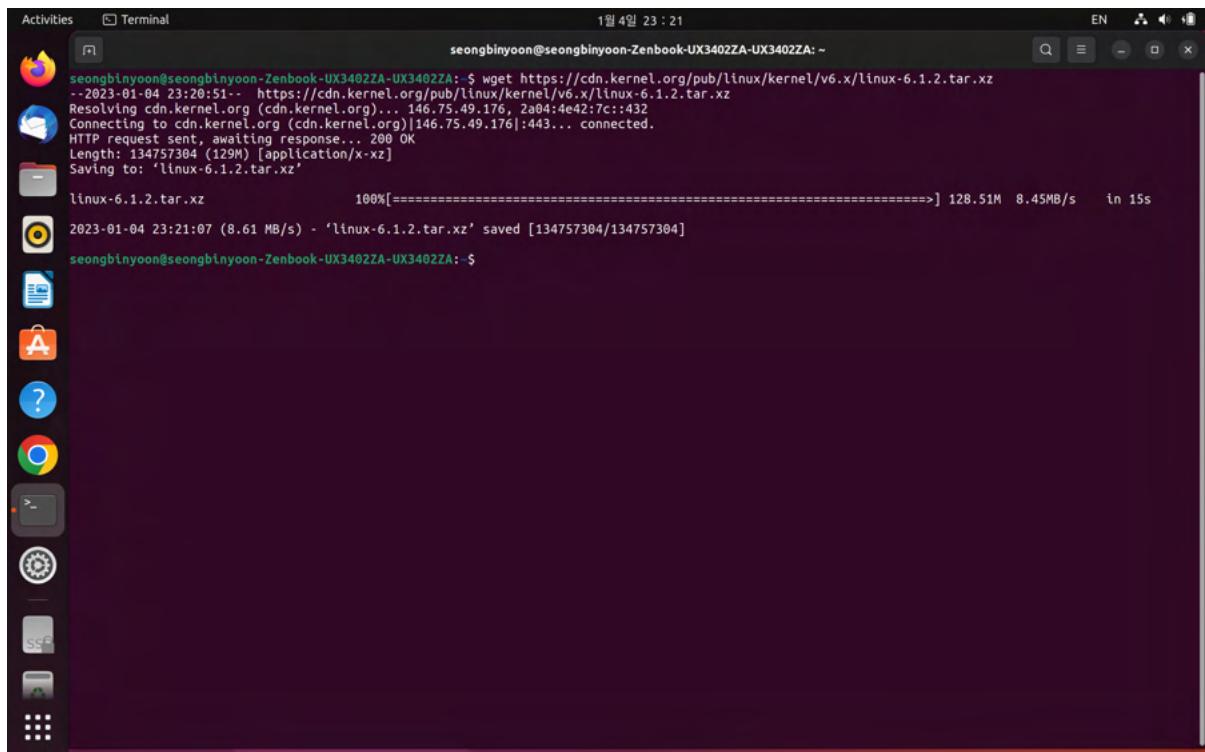
Step1)

wget 명령어를 사용하여 다음과 같이 kernel.org 홈페이지에서 리눅스 커널을 다운로드 받는다.

이 때, 받고자 하는 버전 숫자를 정확히 작성한다.

나의 경우 linux 6.1.2를 받았다.

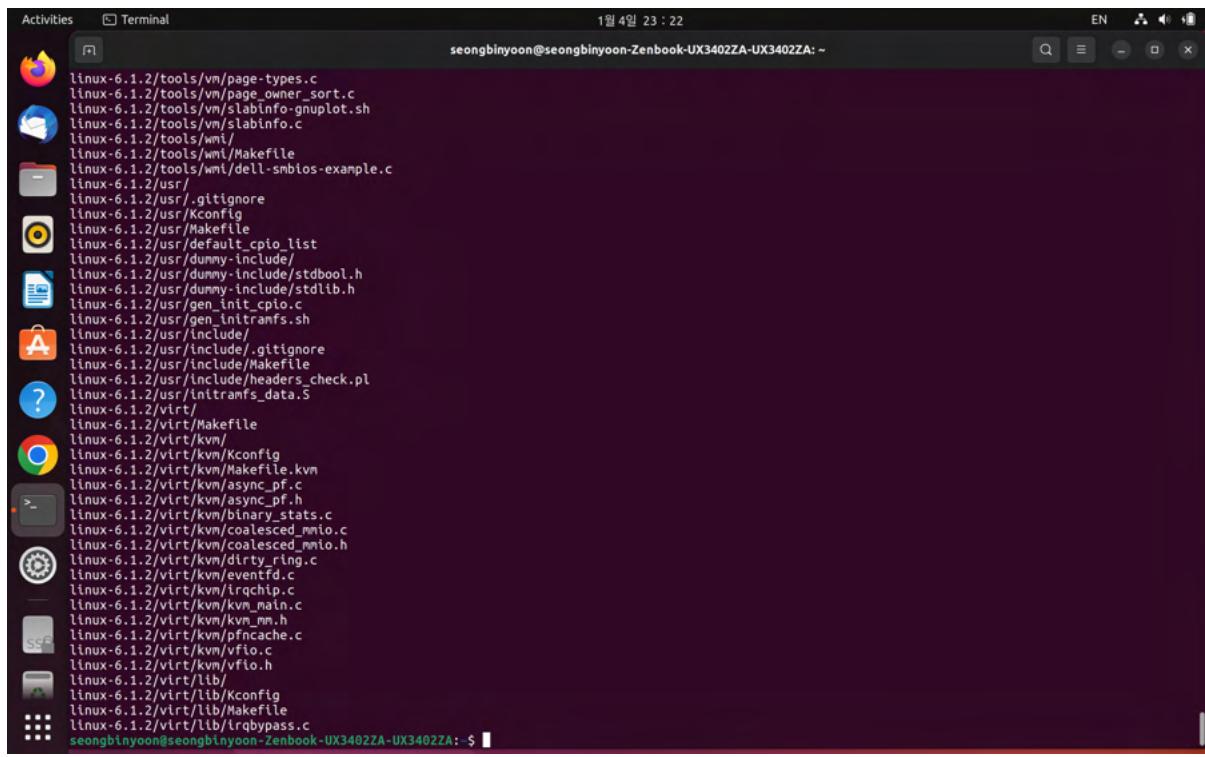
```
~$ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1.2.tar.xz
```



Step2)

tar 명령어를 사용해 소스코드를 추출한다.

```
~$ tar xvf linux-6.1.2.tar.xz
```



Step3)

커널을 설치하는데 필요한 패키지를 설치한다.(자세한 패키지 설명은 위 링크 참조)

```
~$ sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison
```

```
Activities Terminal 1월 4일 23 : 23 EN
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~
linux-6.1.2/tools/vm/page_owner_sort.c
linux-6.1.2/tools/vm/slabInfo-gnuplot.sh
linux-6.1.2/tools/vm/slabInfo.c
linux-6.1.2/tools/wmi/
linux-6.1.2/tools/wmi/Makefile
linux-6.1.2/tools/wmi/dell-smbios-example.c
linux-6.1.2/usr/
linux-6.1.2/usr/.gitignore
linux-6.1.2/usr/Kconfig
linux-6.1.2/usr/Makefile
linux-6.1.2/usr/default_cpio.list
linux-6.1.2/usr/dummy-include/
linux-6.1.2/usr/dummy-include/stdbool.h
linux-6.1.2/usr/dummy-include/stdcuh.h
linux-6.1.2/usr/gen_init_cpio.c
linux-6.1.2/usr/gen_initramfs.sh
linux-6.1.2/usr/include/
linux-6.1.2/usr/include/.gitignore
linux-6.1.2/usr/include/Makefile
linux-6.1.2/usr/include/headers_check.pl
linux-6.1.2/usr/intrtrans_data.S
linux-6.1.2/virt/
linux-6.1.2/virt/Makefile
linux-6.1.2/virt/kvm/
linux-6.1.2/virt/kvm/Kconfig
linux-6.1.2/virt/kvm/Makefile.kvm
linux-6.1.2/virt/kvm/async_pf.c
linux-6.1.2/virt/kvm/async_pf.h
linux-6.1.2/virt/kvm/binary_stats.c
linux-6.1.2/virt/kvm/coalesced_mmio.c
linux-6.1.2/virt/kvm/coalesced_mmio.h
linux-6.1.2/virt/kvm/dirty_ring.c
linux-6.1.2/virt/kvm/eventfd.c
linux-6.1.2/virt/kvm/irqchip.c
linux-6.1.2/virt/kvm/kvm_main.c
linux-6.1.2/virt/kvm/kvm_mn.h
linux-6.1.2/virt/kvm/pfnCache.c
linux-6.1.2/virt/kvm/vfio.c
linux-6.1.2/virt/kvm/vfio.h
linux-6.1.2/virt/lib/
linux-6.1.2/virt/lib/Kconfig
linux-6.1.2/virt/lib/Makefile
linux-6.1.2/virt/lib/irqbypass.c
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~$ sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison
```

Step4)

다음 명령어를 사용해 커널 디렉토리로 이동

```
~$ cd linux-6.1.2
```

디렉토리의 configuration 파일을 .config 파일로 복사한다.

```
~$ cp -v /boot/config-$(uname -r) .config
```

```

Activities Terminal 1월 4일 23:26
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2
Setting up libssl1-dev:amd64 (3.0.2-0ubuntu1.7) ...
Setting up libfl2:amd64 (2.6.4-8build2) ...
Setting up libdpkg-perl (1.21.1ubuntu2.1) ...
Setting up libubsan1:amd64 (12.1.0-2ubuntu1-22.04) ...
Setting up libnsl-dev:amd64 (1.3.0-2build2) ...
Setting up libcrypt-dev:amd64 (1:4.4.27-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.5) ...
Setting up libbinutils:amd64 (2.38-4ubuntu2.1) ...
Setting up libc-dev-bin (2.35-0ubuntu3.1) ...
Setting up libalgorithm-diff-xs-perl (0.04-6build3) ...
Setting up libcc1-0:amd64 (12.1.0-2ubuntu1-22.04) ...
Setting up liblsan0:amd64 (12.1.0-2ubuntu1-22.04) ...
Setting up libitm1:amd64 (12.1.0-2ubuntu1-22.04) ...
Setting up libc-devtools (2.35-0ubuntu3.1) ...
Setting up libalgorithm-merge-perl (0.08-3) ...
Setting up libtsan0:amd64 (11.3.0-1ubuntu1-22.04) ...
Setting up libctf0:amd64 (2.38-4ubuntu2.1) ...
Setting up m4 (1.4.18-Subuntu2)
Setting up git (1:2.34.1-1ubuntu1.5) ...
Setting up libgcc-11-dev:amd64 (11.3.0-1ubuntu1-22.04) ...
Setting up bison (2:3.8.2+dfsg-1build1) ...
update-alternatives: using /usr/bin/bison.yacc to provide /usr/bin/yacc (yacc) in auto mode
Setting up libc6-dev:amd64 (2.35-0ubuntu3.1) ...
Setting up binutils-x86-64-linux-gnu (2.38-4ubuntu2.1) ...
Setting up libncurses-dev:amd64 (6.3-2) ...
Setting up binutils (2.38-4ubuntu2.1) ...
Setting up libfl-dev:amd64 (2.6.4-8build2) ...
Setting up dpkg-dev (1.21.1ubuntu2.1) ...
Setting up libstdc++-11-dev:amd64 (11.3.0-1ubuntu1-22.04) ...
Setting up zlib1g-dev:amd64 (1:1.2.11.dsfg-2ubuntu9.2) ...
Setting up gcc-11 (11.3.0-1ubuntu1-22.04) ...
Setting up g++-11 (11.3.0-1ubuntu1-22.04) ...
Setting up gcc (4:11.2.0-1ubuntu1) ...
Setting up libelf-dev:amd64 (0.186-1build1) ...
Setting up g++ (4:11.2.0-1ubuntu1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for install-info (6.8-4build1) ...
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: $ cd linux-6.1.2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-56-generic' -> '.config'
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$
```

configuration 파일에 변경 사항을 적용한다.

```
~$ make menuconfig
```

```

Activities Terminal 1월 4일 23:27
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2
Setting up build-essential (12.9ubuntu3) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for install-info (6.8-4build1) ...
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: $ cd linux-6.1.2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-56-generic' -> '.config'
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$ make menuconfig
HOSTCC scripts/basic/flddep
UPD scripts/kconfig/mconf-cfg
HOSTCC scripts/kconfig/mconf.o
HOSTCC scripts/kconfig/lxdialog/checklist.o
HOSTCC scripts/kconfig/lxdialog/inputbox.o
HOSTCC scripts/kconfig/lxdialog/menubox.o
HOSTCC scripts/kconfig/lxdialog/textbox.o
HOSTCC scripts/kconfig/lxdialog/util.o
HOSTCC scripts/kconfig/lxdialog/yesno.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX scripts/kconfig/lexer.lex.c
YACC scripts/kconfig/parser.tab.[ch]
HOSTCC scripts/kconfig/lexer.lex.o
HOSTCC scripts/kconfig/menu.o
HOSTCC scripts/kconfig/parser.tab.o
HOSTCC scripts/kconfig/preprocess.o
HOSTCC scripts/kconfig/symbol.o
HOSTCC scripts/kconfig/util.o
HOSTLD scripts/kconfig/mconf
.config:435:warning: symbol value 'm' invalid for I8K
.config:2001:warning: symbol value 'm' invalid for MCTP
.config:8861:warning: symbol value 'm' invalid for VIDEO_ZORAN_DC30
.config:8862:warning: symbol value 'm' invalid for VIDEO_ZORAN_ZR36060
.config:8863:warning: symbol value 'm' invalid for VIDEO_ZORAN_BUZ
.config:8864:warning: symbol value 'm' invalid for VIDEO_ZORAN_DC10
.config:8865:warning: symbol value 'm' invalid for VIDEO_ZORAN_LML33
.config:8866:warning: symbol value 'm' invalid for VIDEO_ZORAN_LML33RYES
.config:9958:warning: symbol value 'm' invalid for ANDROID_BINDER_IPC
.config:9959:warning: symbol value 'm' invalid for ANDROID_BINDERFS

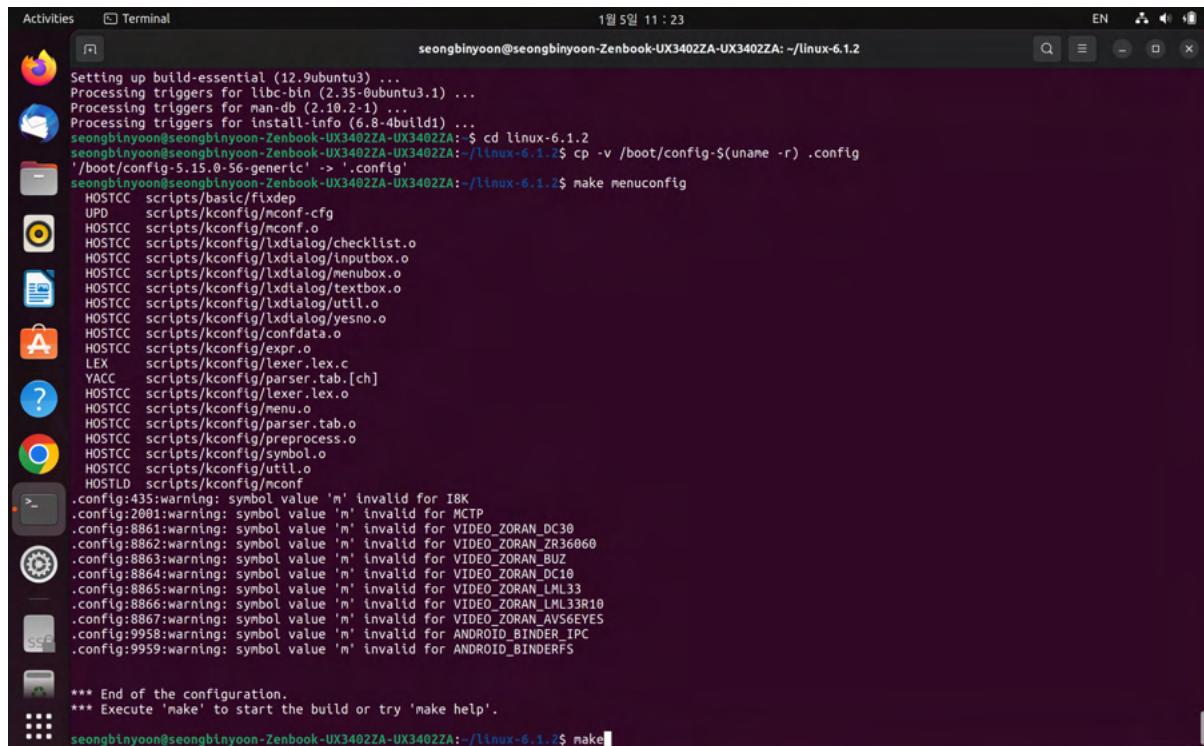
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$
```

setup 창이 뜨는데, 일단 Save → Exit 선택
(해당 단계에서 불필요한 file을 줄여 최대한 컴팩트한 커널을 만들 수 있다.)

Step5)

커널을 빌드한다.

```
~$ make
```



```
Setting up build-essential (12.9ubuntu3) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for install-info (6.8-4build1)
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/linux-6.1.2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/linux-6.1.2$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-56-generic' -> '.config'
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/linux-6.1.2$ make menuconfig
HOSTCC scripts/basic/fixdep
UPD  scripts/kconfig/mconf-cfg
HOSTCC scripts/kconfig/mconf.o
HOSTCC scripts/kconfig/lxdialog/checklist.o
HOSTCC scripts/kconfig/lxdialog/inputbox.o
HOSTCC scripts/kconfig/lxdialog/menubox.o
HOSTCC scripts/kconfig/lxdialog/textbox.o
HOSTCC scripts/kconfig/lxdialog/util.o
HOSTCC scripts/kconfig/lxdialog/yesno.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX   scripts/kconfig/lexer.lex.c
YACC  scripts/kconfig/parser.tab.[ch]
HOSTCC scripts/kconfig/lexer.lex.o
HOSTCC scripts/kconfig/menu.o
HOSTCC scripts/kconfig/parser.tab.o
HOSTCC scripts/kconfig/preprocess.o
HOSTCC scripts/kconfig/symbol.o
HOSTCC scripts/kconfig/util.o
HOSTLD scripts/kconfig/mconf
      .config:43:warning: symbol value 'm' invalid for ISK
      .config:2001:warning: symbol value 'm' invalid for MCTP
      .config:8861:warning: symbol value 'm' invalid for VIDEO_ZORAN_DC30
      .config:8862:warning: symbol value 'm' invalid for VIDEO_ZORAN_ZR36060
      .config:8863:warning: symbol value 'm' invalid for VIDEO_ZORAN_BUZ
      .config:8864:warning: symbol value 'm' invalid for VIDEO_ZORAN_DC10
      .config:8865:warning: symbol value 'm' invalid for VIDEO_ZORAN_LML33
      .config:8866:warning: symbol value 'm' invalid for VIDEO_ZORAN_LML33R10
      .config:8867:warning: symbol value 'm' invalid for VIDEO_ZORAN_AV56EYES
      .config:9958:warning: symbol value 'm' invalid for ANDROID_BINDER_IPC
      .config:9959:warning: symbol value 'm' invalid for ANDROID_BINDERFS
      *** End of the configuration.
      *** Execute 'make' to start the build or try 'make help'.
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/linux-6.1.2$ make
```

아래와 같이 No rule to make target 'debian/canonical-certs.pem'... 오류가 뜨면 다음 명령어를 입력하여 충돌되는 security certificates를 해제

```
~$ scripts/config --disable SYSTEM_TRUSTED_KEYS
```

```
~$ scripts/config --disable SYSTEM_REVOCATION_KEYS
```

```
Activities Terminal 1월 5일 11:28
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2
CC kernel/kexec_core.o
CC kernel/kexec.o
CC kernel/kexec_file.o
CC kernel/compat.o
CC kernel/utsname.o
CC kernel/user_namespace.o
CC kernel/pid_namespace.o
CC kernel/stop_machine.o
CC kernel/audit.o
CC kernel/auditfilter.o
CC kernel/auditsc.o
CC kernel/audit_watch.o
CC kernel/audit_fsnofity.o
CC kernel/audit_tree.o
CC kernel/kprobes.o
CC kernel/hung_task.o
CC kernel/watchdog.o
CC kernel/watchdog_hld.o
CC kernel/seccomp.o
CC kernel/relay.o
CC kernel/utsname_sysctl.o
CC kernel/delayacct.o
CC kernel/taskstats.o
CC kernel/tssact.o
CC kernel/tracepoint.o
CC kernel/irq_work.o
CC kernel/static_call.o
CC kernel/static_call_inline.o
CC kernel/user-return-notifier.o
CC kernel/padata.o
CC kernel/crash_dump.o
CC kernel/jump_label.o
CC kernel/context_tracking.o
CC kernel/iomen.o
CC kernel/seq.o
CC kernel/watch_queue.o
AR kernel/built-in.a
CHK kernel/kheaders_data.tar.xz
GEN kernel/kheaders_data.tar.xz
CC [M] kernel/kheaders.o
CC certs/system_keyring.o
make[2]: *** No rule to make target 'debian/canonical-certs.pem', needed by 'certs/x509_certificate_list'. Stop.
make[1]: *** [scripts/Makefile.build:500: certs] Error 2
make: *** [Makefile:1992: .] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$
```

```
Activities Terminal 1월 5일 11:29
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2
CC kernel/kexec_file.o
CC kernel/compat.o
CC kernel/utsname.o
CC kernel/user_namespace.o
CC kernel/pid_namespace.o
CC kernel/stop_machine.o
CC kernel/audit.o
CC kernel/auditfilter.o
CC kernel/auditsc.o
CC kernel/audit_watch.o
CC kernel/audit_fsnofity.o
CC kernel/audit_tree.o
CC kernel/kprobes.o
CC kernel/hung_task.o
CC kernel/watchdog.o
CC kernel/watchdog_hld.o
CC kernel/seccomp.o
CC kernel/relay.o
CC kernel/utsname_sysctl.o
CC kernel/delayacct.o
CC kernel/taskstats.o
CC kernel/tssact.o
CC kernel/tracepoint.o
CC kernel/irq_work.o
CC kernel/static_call.o
CC kernel/static_call_inline.o
CC kernel/user-return-notifier.o
CC kernel/padata.o
CC kernel/crash_dump.o
CC kernel/jump_label.o
CC kernel/context_tracking.o
CC kernel/iomen.o
CC kernel/seq.o
CC kernel/watch_queue.o
AR kernel/built-in.a
CHK kernel/kheaders_data.tar.xz
GEN kernel/kheaders_data.tar.xz
CC [M] kernel/kheaders.o
CC certs/system_keyring.o
make[2]: *** No rule to make target 'debian/canonical-certs.pem', needed by 'certs/x509_certificate_list'. Stop.
make[1]: *** [scripts/Makefile.build:500: certs] Error 2
make: *** [Makefile:1992: .] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$ scripts/config --disable SYSTEM_TRUSTED_KEYS
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$ scripts/config --disable SYSTEM_REVOCATION_KEYS
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$
```

곧바로 make 명령어 입력 후 Y 및 엔터

Activities Terminal 1월 5일 11:29 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2

```

CC kernel/watchdog.o
CC kernel/watchdog_hld.o
CC kernel/seccomp.o
CC kernel/relay.o
CC kernel/utsname_sysctl.o
CC kernel/delayacct.o
CC kernel/taskstats.o
CC kernel/tscacct.o
CC kernel/tracepoint.o
CC kernel/irq_work.o
CC kernel/static_call.o
CC kernel/static_call_inline.o
CC kernel/user-return-notifier.o
CC kernel/padata.o
CC kernel/crash_dump.o
CC kernel/jump_label.o
CC kernel/context_tracking.o
CC kernel/lmem.o
CC kernel/seq.o
CC kernel/watch_queue.o
AR kernel/built-in.a
CHK kernel/kheaders_data.tar.xz
GEN kernel/kheaders_data.tar.xz
CC [M] kernel/kheaders.o
CC certs/system_keyring.o
make[2]: *** No rule to make target 'debian/canonical-certs.pem', needed by 'certs/x509_certificate_list'. Stop.
make[1]: *** [scripts/Makefile:500: certs] Error 2
make: *** [Makefile:1992: .] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/linux-6.1.2$ scripts/config --disable SYSTEM_TRUSTED_KEYS
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/linux-6.1.2$ scripts/config --disable SYSTEM_REVOCATION_KEYS
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/linux-6.1.2$ make
SYNC include/config/auto.conf.cmd
*
* Restart config...
*
*
* Certificates for signature checking
*
File name or PKCS#11 URI of module signing key (MODULE_SIG_KEY) [certs/signing_key.pem] certs/signing_key.pem
Type of module signing key to be generated
> 1. RSA (MODULE_SIG_KEY_TYPE_RSA)
2. ECDSA (MODULE_SIG_KEY_TYPE_ECDSA)
choice[1-2]: 1
Provide system-wide ring of trusted keys (SYSTEM_TRUSTED_KEYRING) [Y/?] y
Additional X.509 keys for default system keyring (SYSTEM_TRUSTED_KEYS) [] (NEW)

```

Activities Terminal 1월 5일 11:24 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2

```

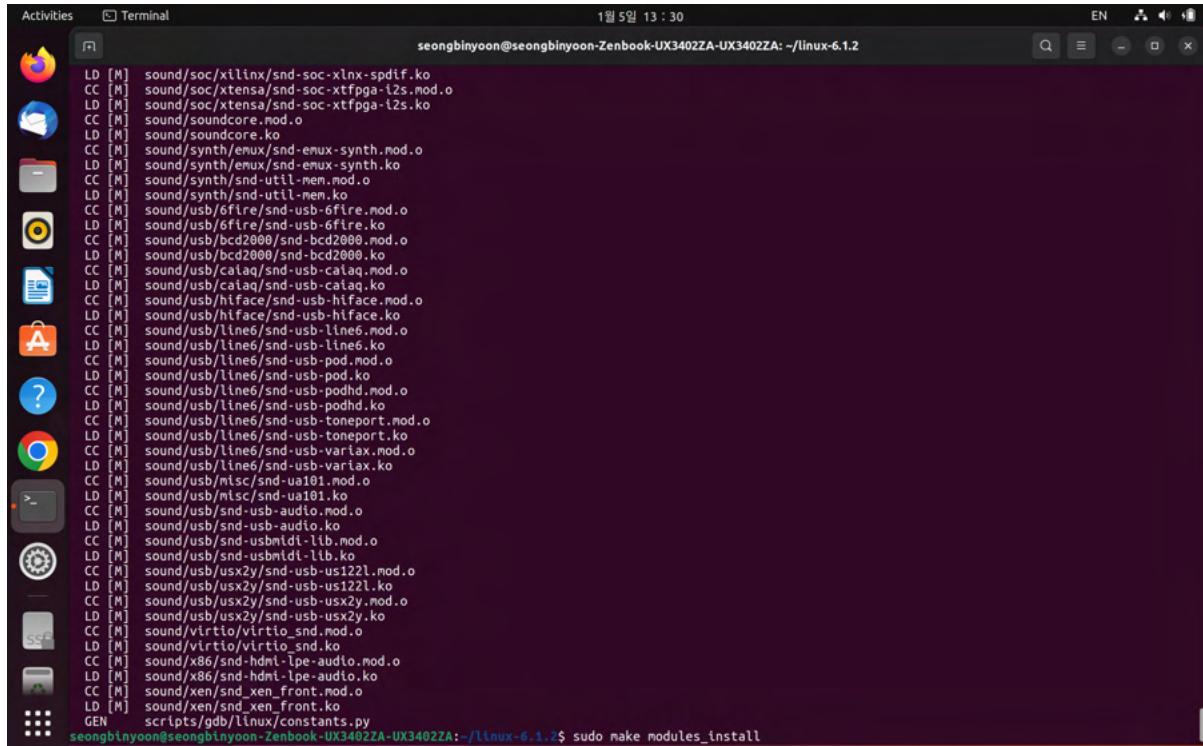
CC arch/x86/kernel/cpu/sgx/main.o
CC arch/x86/kernel/cpu/sgx/virt.o
AR arch/x86/kernel/cpu/sgx/built-in.a
CC arch/x86/kernel/cpu/cacheinfo.o
CC arch/x86/kernel/cpu/scattered.o
CC arch/x86/kernel/cpu/topology.o
CC arch/x86/kernel/cpu/common.o
CC arch/x86/kernel/cpu/drando.o
CC arch/x86/kernel/cpu/match.o
CC arch/x86/kernel/cpu/bugs.o
CC arch/x86/kernel/cpu/aperfmpf.o
CC arch/x86/kernel/cpu/cpuid-deps.o
CC arch/x86/kernel/cpu/uwait.o
CC arch/x86/kernel/cpu/proc.o
MKCAP arch/x86/kernel/cpu/capflags.c
CC arch/x86/kernel/cpu/capflags.o
CC arch/x86/kernel/cpu/powerflags.o
CC arch/x86/kernel/cpu/feat_ctl.o
CC arch/x86/kernel/cpu/intel.o
CC arch/x86/kernel/cpu/intel_pconfig.o
CC arch/x86/kernel/cpu/tlsx.o
CC arch/x86/kernel/cpu/intel_epb.o
CC arch/x86/kernel/cpu/and.o
CC arch/x86/kernel/cpu/hygon.o
CC arch/x86/kernel/cpu/centaur.o
CC arch/x86/kernel/cpu/zhaoxin.o
CC arch/x86/kernel/cpu/perfcctr-watchdog.o
CC arch/x86/kernel/cpu/vmware.o
CC arch/x86/kernel/cpu/hypervtsr.o
CC arch/x86/kernel/cpu/mshyperv.o
CC arch/x86/kernel/cpu/acrn.o
AR arch/x86/kernel/cpu/built-in.a
CC arch/x86/kernel/api/boot.o
CC arch/x86/kernel/api/sleep.o
AS arch/x86/kernel/api/wakeup_64.o
CC arch/x86/kernel/api/apei.o
CC arch/x86/kernel/api/cppc.o
CC arch/x86/kernel/api/cstate.o
AR arch/x86/kernel/api/built-in.a
CC arch/x86/kernel/api/apic.o
CC arch/x86/kernel/api/apic_common.o
CC arch/x86/kernel/api/apic_noop.o
CC arch/x86/kernel/api/ipi.o
CC arch/x86/kernel/apic/vector.o

```

Step6)

모듈을 설치한다.

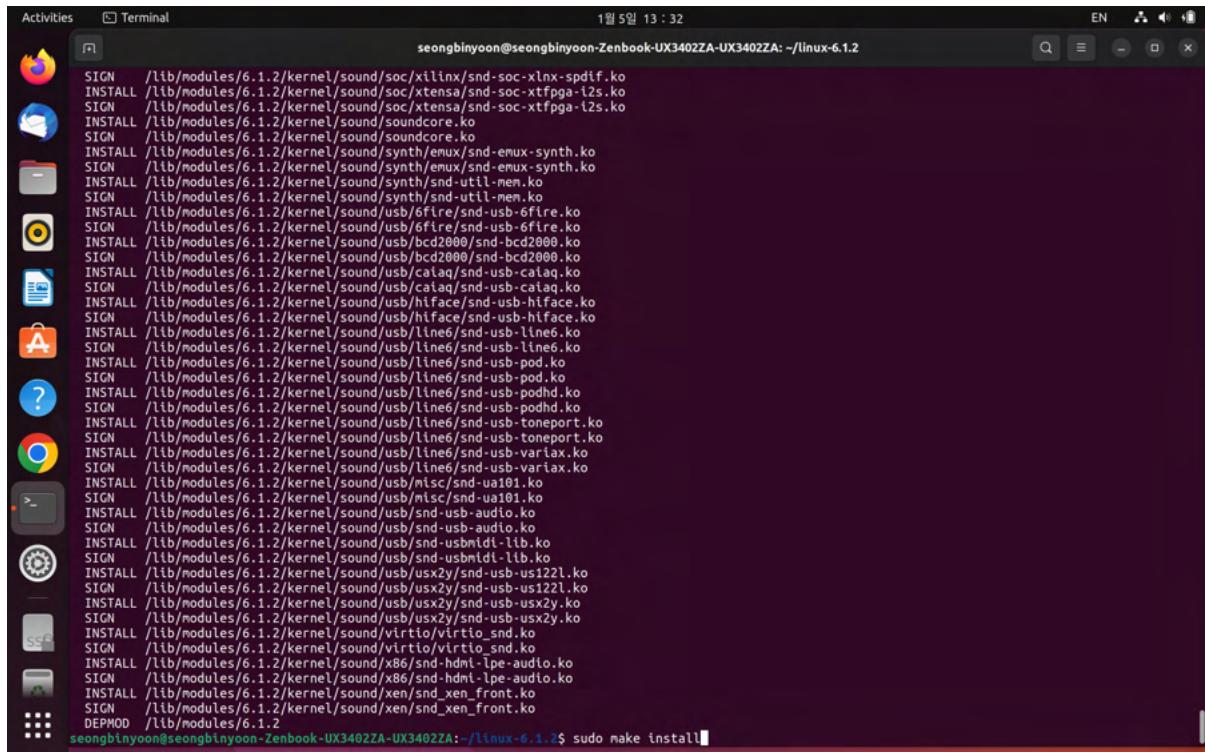
```
~$ sudo make modules_install
```



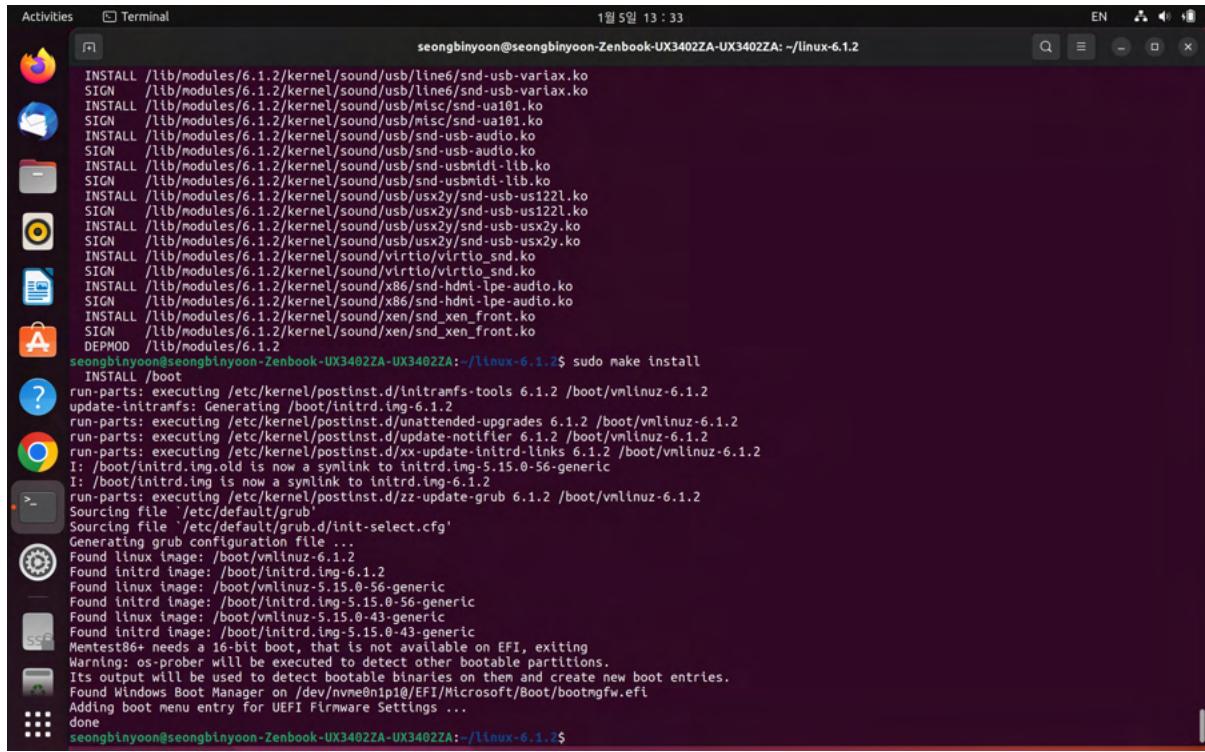
```
Activities Terminal 1월 5일 13 : 30
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2
LD [M] sound/soc/xilinx/snd-soc-xlnx-spdif.ko
CC [M] sound/soc/xtensa/snd-soc-xtfpqa-lzs.mod.o
LD [M] sound/soc/xtensa/snd-soc-xtfpqa-lzs.ko
CC [M] sound/soundcore.mod.o
LD [M] sound/soundcore.ko
CC [M] sound/synth/emu3/snd-emu3-synth.mod.o
LD [M] sound/synth/emu3/snd-emu3-synth.ko
CC [M] sound/synth/snd-util-mem.mod.o
LD [M] sound/synth/snd-util-mem.ko
CC [M] sound/usb/6fire/snd-usb-6fire.mod.o
LD [M] sound/usb/6fire/snd-usb-6fire.ko
CC [M] sound/usb/bcd2000/snd-bcd2000.mod.o
LD [M] sound/usb/bcd2000/snd-bcd2000.ko
CC [M] sound/usb/calaq/snd-usb-calaq.mod.o
LD [M] sound/usb/calaq/snd-usb-calaq.ko
CC [M] sound/usb/htface/snd-usb-htface.mod.o
LD [M] sound/usb/htface/snd-usb-htface.ko
CC [M] sound/usb/line6/snd-usb-line6.mod.o
LD [M] sound/usb/line6/snd-usb-line6.ko
CC [M] sound/usb/line6/snd-usb-pod.mod.o
LD [M] sound/usb/line6/snd-usb-pod.ko
CC [M] sound/usb/line6/snd-usb-podhd.mod.o
LD [M] sound/usb/line6/snd-usb-podhd.ko
CC [M] sound/usb/line6/snd-usb-toneport.mod.o
LD [M] sound/usb/line6/snd-usb-toneport.ko
CC [M] sound/usb/line6/snd-usb-variax.mod.o
LD [M] sound/usb/line6/snd-usb-variax.ko
CC [M] sound/usb/misc/snd-ua101.mod.o
LD [M] sound/usb/misc/snd-ua101.ko
CC [M] sound/usb/snd-usb-audio.mod.o
LD [M] sound/usb/snd-usb-audio.ko
CC [M] sound/usb/snd-usbmidi1-lib.mod.o
LD [M] sound/usb/snd-usbmidi1-lib.ko
CC [M] sound/usb/usx2y/snd-usb-us122l.mod.o
LD [M] sound/usb/usx2y/snd-usb-us122l.ko
CC [M] sound/usb/usx2y/snd-usb-usx2y.mod.o
LD [M] sound/usb/usx2y/snd-usb-usx2y.ko
CC [M] sound/virtio/virtio_snd.mod.o
LD [M] sound/virtio/virtio_snd.ko
CC [M] sound/x86/snd-hdmi-lpe-audio.mod.o
LD [M] sound/x86/snd-hdmi-lpe-audio.ko
CC [M] sound/xen/snd_xen_front.mod.o
LD [M] sound/xen/snd_xen_front.ko
GEN scripts/gdb/linux/constants.py
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$ sudo make modules_install
```

커널을 설치하여 완료한다.

```
~$ sudo make install
```



```
Activities Terminal 1월 5일 13 : 32
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA: ~/linux-6.1.2$ sudo make install
SIGN /lib/modules/6.1.2/kernel/sound/soc/xilinx/snd-soc-xlnx-spdif.ko
INSTALL /lib/modules/6.1.2/kernel/sound/soc/xtensa/snd-soc-xtfpqa-lzs.ko
SIGN /lib/modules/6.1.2/kernel/sound/soc/xtensa/snd-soc-xtfpqa-lzs.ko
INSTALL /lib/modules/6.1.2/kernel/sound/soundcore.ko
SIGN /lib/modules/6.1.2/kernel/sound/soundcore.ko
INSTALL /lib/modules/6.1.2/kernel/sound/synth/emu32/snd-emu32-synth.ko
SIGN /lib/modules/6.1.2/kernel/sound/synth/emu32/snd-emu32-synth.ko
INSTALL /lib/modules/6.1.2/kernel/sound/synth/snd-utl-mem.ko
SIGN /lib/modules/6.1.2/kernel/sound/synth/snd-utl-mem.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/6fire/snd-usb-6fire.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/6fire/snd-usb-6fire.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/bcd2000/snd-bcd2000.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/bcd2000/snd-bcd2000.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/caiaq/snd-usb-caiaq.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/caiaq/snd-usb-caiaq.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/hiface/snd-usb-hiface.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/hiface/snd-usb-hiface.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-line6.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-line6.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-pod.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-pod.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-podhd.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-podhd.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-toneport.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-toneport.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-variax.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-variax.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/misc/snd-ua101.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/misc/snd-ua101.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/snd-usb-audio.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/snd-usb-audio.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/snd-usbmidi-lib.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/snd-usbmidi-lib.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2l.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2l.ko
INSTALL /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
SIGN /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
INSTALL /lib/modules/6.1.2/kernel/sound/x86/snd-hdmi-lpe-audio.ko
SIGN /lib/modules/6.1.2/kernel/sound/x86/snd-hdmi-lpe-audio.ko
INSTALL /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
SIGN /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
DEPMOD /lib/modules/6.1.2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA: ~/linux-6.1.2$ sudo make install
```



```
Activities Terminal 1월 5일 13 : 33
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA: ~/linux-6.1.2$ sudo make install
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-variax.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-variax.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/misc/snd-ua101.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/misc/snd-ua101.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/snd-usb-audio.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/snd-usb-audio.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/snd-usbmidi-lib.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/snd-usbmidi-lib.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2l.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2l.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
SIGN /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
INSTALL /lib/modules/6.1.2/kernel/sound/x86/snd-hdmi-lpe-audio.ko
SIGN /lib/modules/6.1.2/kernel/sound/x86/snd-hdmi-lpe-audio.ko
INSTALL /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
SIGN /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
DEPMOD /lib/modules/6.1.2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA: ~/linux-6.1.2$ sudo make install
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.1.2 /boot/vmlinuz-6.1.2
update-initramfs: Generating /boot/initrd.img-6.1.2
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 6.1.2 /boot/vmlinuz-6.1.2
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.1.2 /boot/vmlinuz-6.1.2
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 6.1.2 /boot/vmlinuz-6.1.2
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-56-generic
I: /boot/initrd.img is now a symlink to initrd.img-6.1.2
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.1.2 /boot/vmlinuz-6.1.2
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.1.2
Found initrd image: /boot/initrd.img-6.1.2
Found linux image: /boot/vmlinuz-5.15.0-56-generic
Found initrd image: /boot/initrd.img-5.15.0-56-generic
Found linux image: /boot/vmlinuz-5.15.0-43-generic
Found initrd image: /boot/initrd.img-5.15.0-43-generic
Mntest86+ needs a 16-bit boot, that is not available on EFI, exiting
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot entries.
Found Windows Boot Manager on /dev/nvme0n1p1@/EFI/Microsoft/Boot/bootmgfw.efi
Adding boot menu entry for UEFI Firmware Settings ...
done
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA: ~/linux-6.1.2$
```

Step7)

부트로더를 업데이트 한다.

initramfs를 업데이트 한다.

```
~$ sudo update-initramfs -c -k 6.1.2
```

GRUB를 업데이트한다.

```
~$ sudo update-grub
```

Step8)

PC를 재부팅하고 커널 버전을 확인한다.

```
~$ uname -mrs
```

재부팅 후 out of memory error, bad shim signature error 등의 **Kernel Panic**이 일어난다면 Trouble Shooting 참조.

Trouble Shooting

Trouble 1

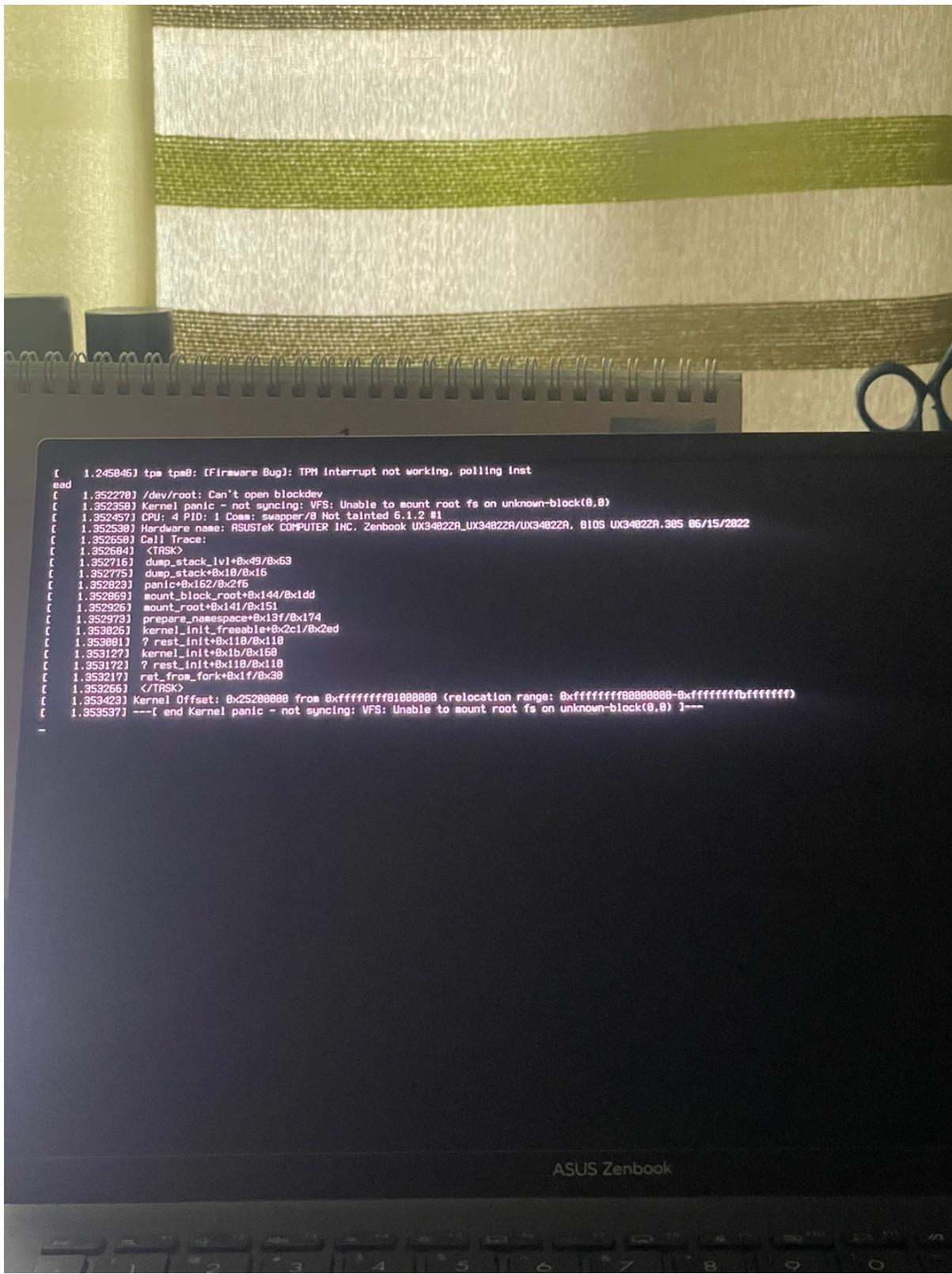
우분투 22.04 LTS 버전의 듀얼부팅은 완료되어 Linux Kernel을 컴파일하는 과정에서 문제가 생겼다.

Linux Kernel Compile 챕터 매뉴얼의 Step7은 optional이라 진행하지 않고 바로 재부팅을 하였다.

GRUB 창에서 Ubuntu를 선택해 부팅을 하였는데 Error가 발생했다.

“Bad shim signature”

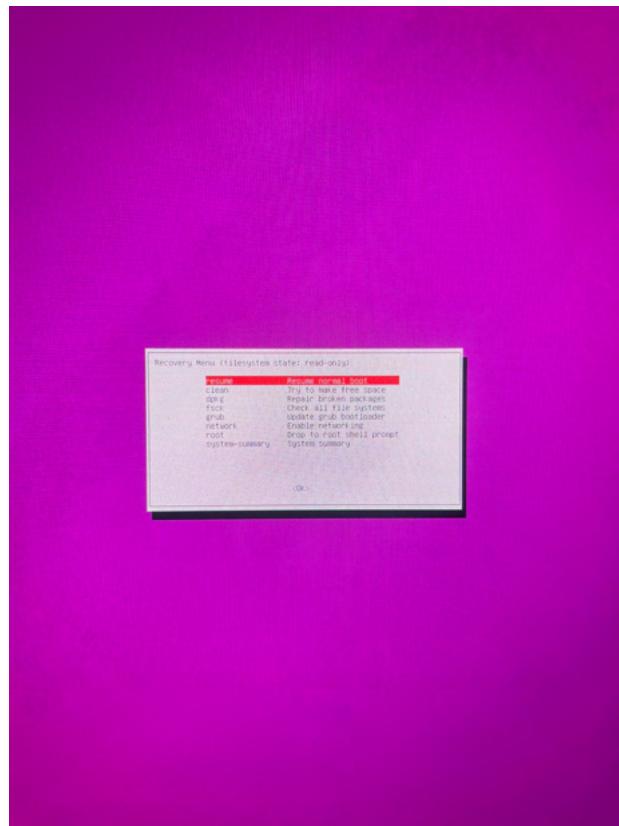
Press any key to continue 에서 엔터를 눌렀더니



Kernel Panic이 일어났다.

reboot 하여 GRUB에 들어가 ubuntu 5.15(이전 버전)의 Recovery mode(복구 모드)로 들어간다.

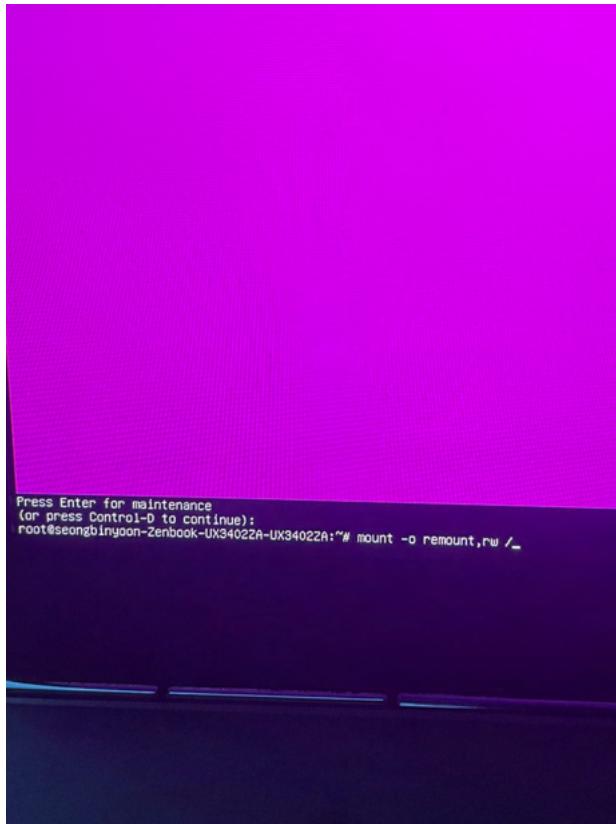
아래의 root의 Drop to root shell prompt를 선택하고 엔터



아래와 같은 프롬프트가 보인다.

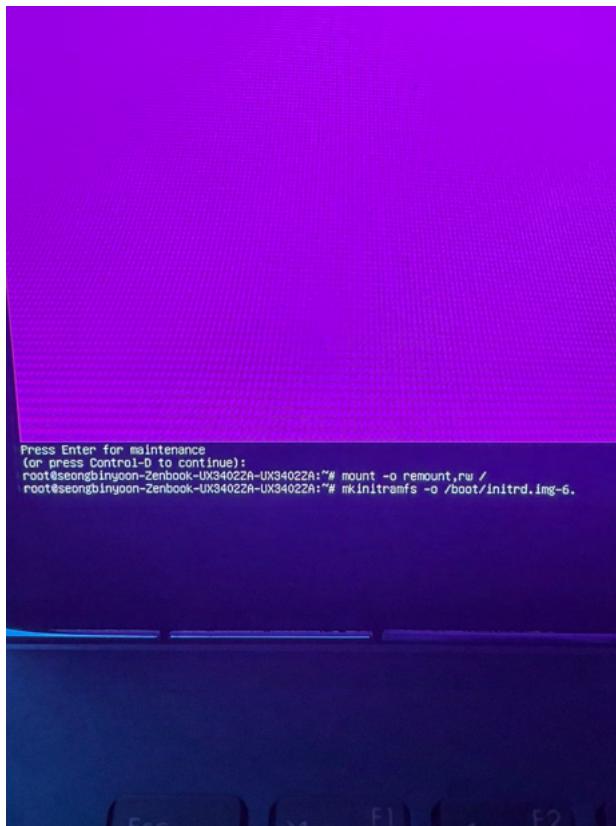
커널을 다시 마운트한다.

```
mount -o remount,rw /
```

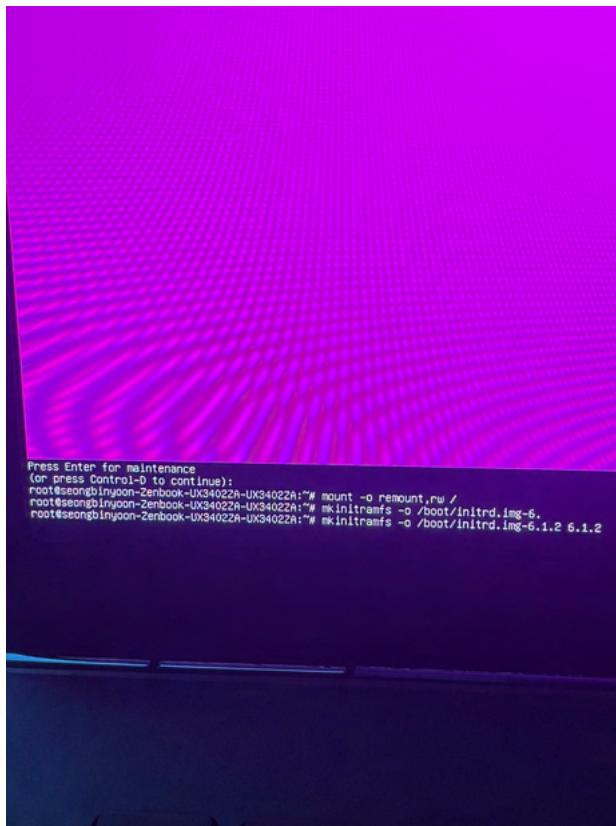


mkinitramfs 명령어를 사용해 initrd.img를 boot에 만들어준다.

```
mkinitramfs -o /boot/initrd.img-6.
```



```
mkinitramfs -o /boot/initrd.img-6.1.2 6.1.2
```



GRUB를 업데이트한다.

```
update-grub
```

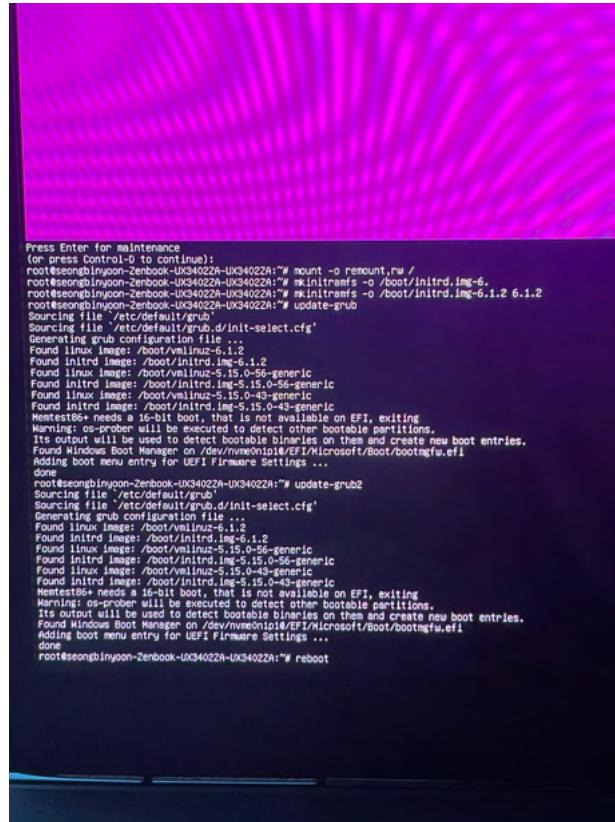
```
update-grub2
```

```
Press Enter for maintenance  
(or press Control-D to continue)  
root@seongbinyoon-Zenbook-UX3402ZA:~# mount -o remount,rw /  
root@seongbinyoon-Zenbook-UX3402ZA:~# rminitramfs -o /boot/initrd.img-6.  
root@seongbinyoon-Zenbook-UX3402ZA:~# rminitramfs -o /boot/initrd.img-6.1.2 6.1.2  
root@seongbinyoon-Zenbook-UX3402ZA:~# update-grub
```

```
Press Enter for maintenance  
(or press Control-D to continue)  
root@seongbinyoon-Zenbook-UX3402ZA:~# mount -o remount,rw /  
root@seongbinyoon-Zenbook-UX3402ZA:~# rminitramfs -o /boot/initrd.img-6.  
root@seongbinyoon-Zenbook-UX3402ZA:~# rminitramfs -o /boot/initrd.img-6.1.2 6.1.2  
root@seongbinyoon-Zenbook-UX3402ZA:~# update-grub  
Sourcing file '/etc/default/grub'  
Sourcing file '/etc/default/grub.d/init-select.cfg'  
Generating grub configuration file ...  
Found initrd Image: /boot/initrd.img-6.1.2  
Found linux Image: /boot/vmlinuz-5.15.0-56-generic  
Found initrd Image: /boot/initrd.img-5.15.0-56-generic  
Found linux Image: /boot/vmlinuz-5.15.0-49-generic  
Found initrd Image: /boot/initrd.img-5.15.0-49-generic  
Warning: os-prober will not be able to find bootable media on UEFI, exiting  
Warning: os-prober will be executed to detect other bootable partitions.  
Its output will be used to detect bootable binaries on them and creates new boot entries.  
Found Windows Boot Manager on /dev/nvme0n1p4/EFI/Microsoft/Boot/bootmgfw.efi  
Adding boot menu entry for UEFI Firmware Settings ...  
done  
root@seongbinyoon-Zenbook-UX3402ZA:~# update-grub2
```

재부팅한다.

```
reboot
```



위와 같이 진행한 결과 커널 5.15는 복구가 되었다.

하지만 커널 6.1.2는 복구가 되지 않았고 다른 오류가 발생하며 커널 패닉이 계속되었다.

“out of memory”

심지어 GRUB에는 존재하는 linux kernel 6.1.2가

```
~$ dpkg -list | grep linux-image
```

위 명령어로 linux-image를 리스트로 볼 때 없었다. linux-image가 6.1.2 커널에 없는 것 같다.

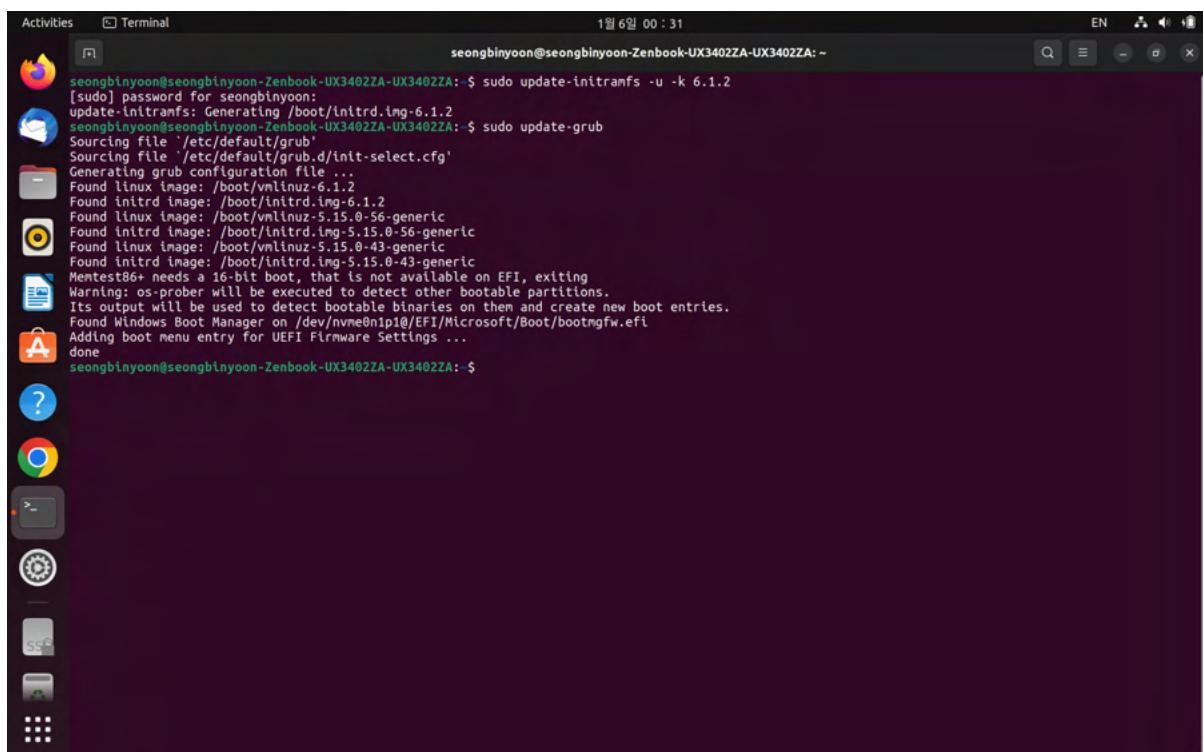
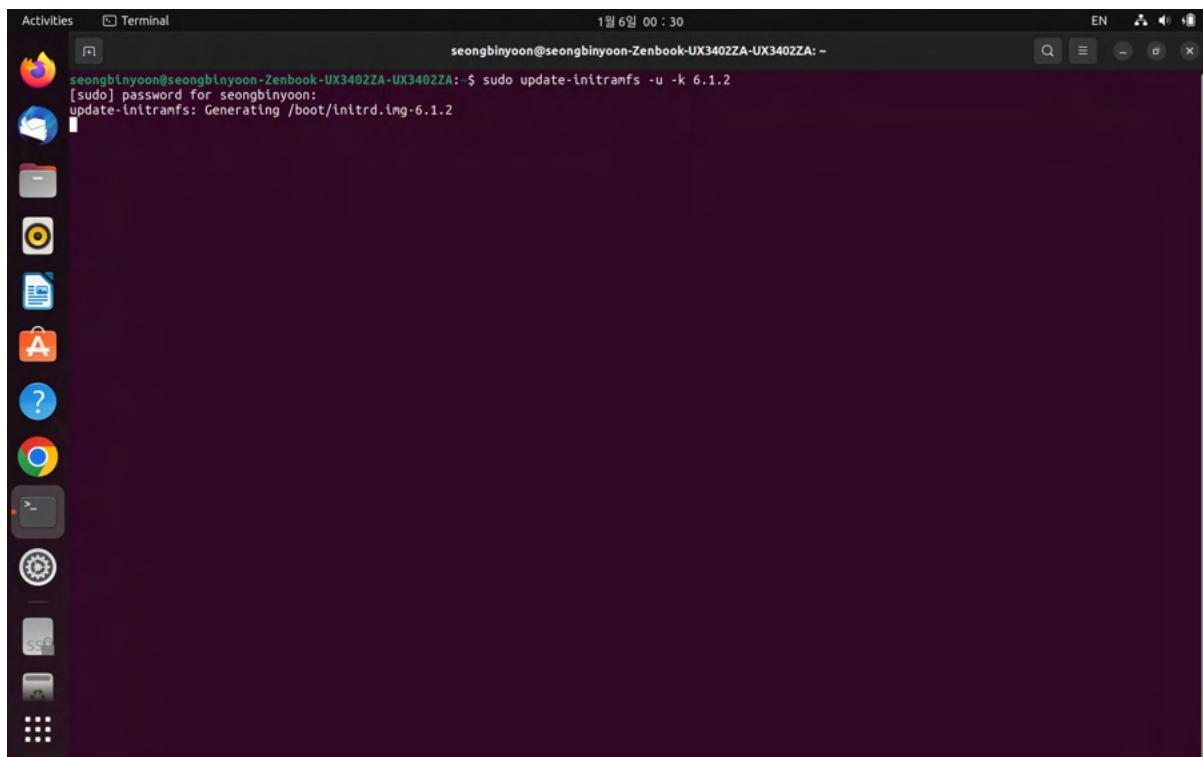
내가 생각한 원인은 커널 컴파일 시 RAM의 메모리 용량이 부족한 것이 원인이 아닐까 싶었다.

그래서 swap 메모리를 따로 할당해줄까 하여 찾아보기도 했다.

아래의 명령어를 입력해 6.1.2 커널에 initramfs를 업데이트 및 GRUB 업데이트도 시도해 보았다.

```
~$ sudo update-initramfs -u -k 6.1.2
```

```
~$ sudo update-grub
```

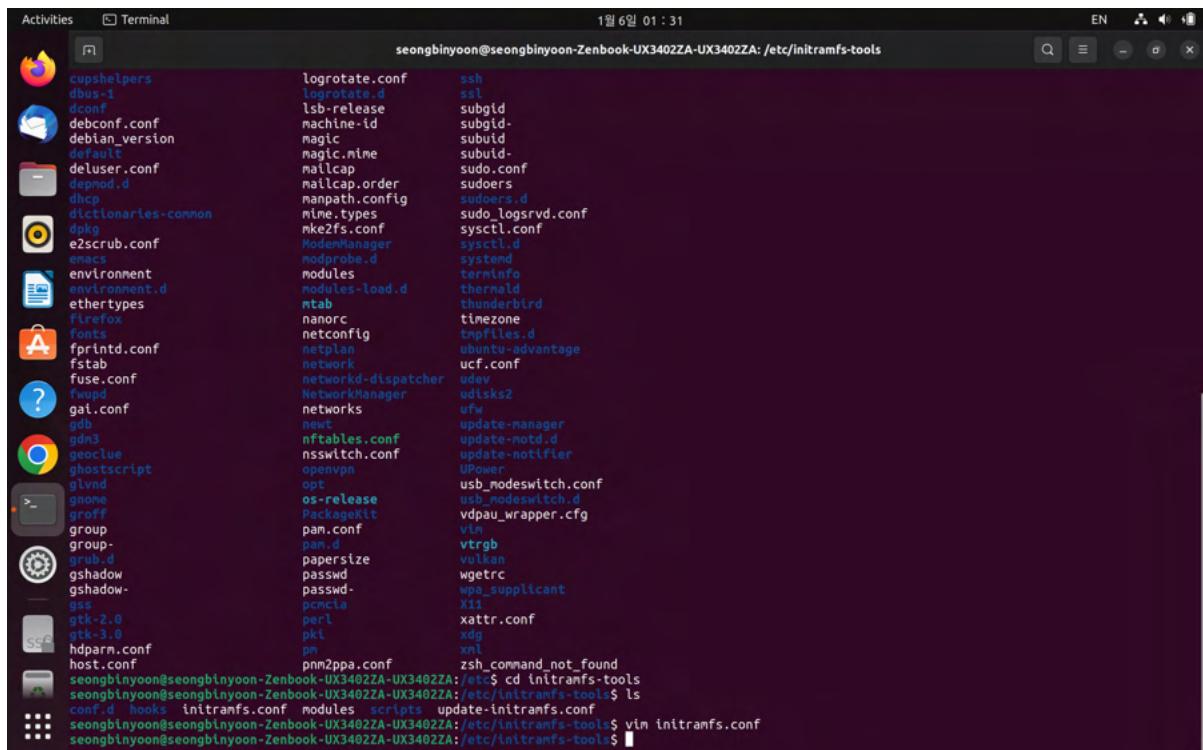


또 다른 글에서는 /etc/initramfs-tools에 있는 initramfs.conf 파일을 수정해야한다고 해서 아래와 같이 찾아보았다.
vim이 설치되어 있지 않다면 아래와 sudo 명령어로 설치한다.(Optional)

```
~$ sudo apt-get install vim
```

```
~$ cd /etc/initramfs-tools
```

```
~$ vim initramfs.conf
```



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and contains a list of files in the directory `/etc/initramfs-tools`. The files listed include logrotate.conf, ssh, logrotate.d, ssl, lsb-release, subgid, machine-id, subgld, magic, subuid, magic.mime, subuid-, mailcap, subutd, mailcap.order, sudo.conf, manpath.config, sudoers, mime.types, sudoers.d, nke2fs.conf, sudo_logsrvd.conf, ModemManager, sysctl.conf, depmod.d, modprobe.d, systcl.d, dhcpc, modules-load.d, systemd, dictionaries-common, nanorc, terminfo, e2scrub.conf, thunderbird, default, environment, netconfig, timezone, emacs, environment.d, netplan, tmpfiles.d, fonts, ucf.conf, ethertypes, network, udev, firefox, modules, NetworkManager, udisks2, fprintd.conf, networkd-dispatcher, ufw, fstab, networks, newt, update-manager, fuse.conf, nftables.conf, update-notified, gdb, nsswitch.conf, update-notifier, gdm3, openvpn, IPower, geoclue, opt, udisks, ghostscript, pm, vdpau_wrapper.cfg, glvnd, gnome, pam.conf, vtn, groff, pam.d, vtrgb, group, papersize, vulkan, group.d, passwd, wgetc, gshadow, passwd-, wpa_supplicant, gshadow-, pcmcia, X11, gss, perl, xattr.conf, gtk-2.0, pki, xdg, gtk-3.0, pm, xml, hdparm.conf, pm2ppa.conf, zsh_command_not_found, host.conf, seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/etc/initramfs-tools\$ ls, seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/etc/initramfs-tools\$ vim initramfs.conf, seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/etc/initramfs-tools\$

```

Activities Terminal 1월 6일 01:08
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /boot/grub
# DO NOT EDIT THIS FILE
#
# It is automatically generated by grub-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#
### BEGIN /etc/grub.d/00_header ####
if [ -s $prefix/grubenv ]; then
    set have_grubenv=true
load_env
fi
if [ "${initrdfail}" = 2 ]; then
    set initrdfail=
elif [ "${initrdfail}" = 1 ]; then
    set next_entry="${prev_entry}"
    set prev_entry=
    save_env prev_entry
    if [ "${next_entry}" ]; then
        set initrdfail=2
    fi
    if [ "${next_entry}" ]; then
        set default="${next_entry}"
        set next_entry=
        save_env next_entry
        set boot_once=true
    else
        set default="0"
    fi
if [ "${feature_menuentry_id}" = xy ]; then
    menuentry_id_option="--id"
else
    menuentry_id_option=""
fi
export menuentry_id_option
if [ "${prev_saved_entry}" ]; then
    set saved_entry="${prev_saved_entry}"
    save_env saved_entry
    set prev_saved_entry=
    save_env prev_saved_entry

```

```

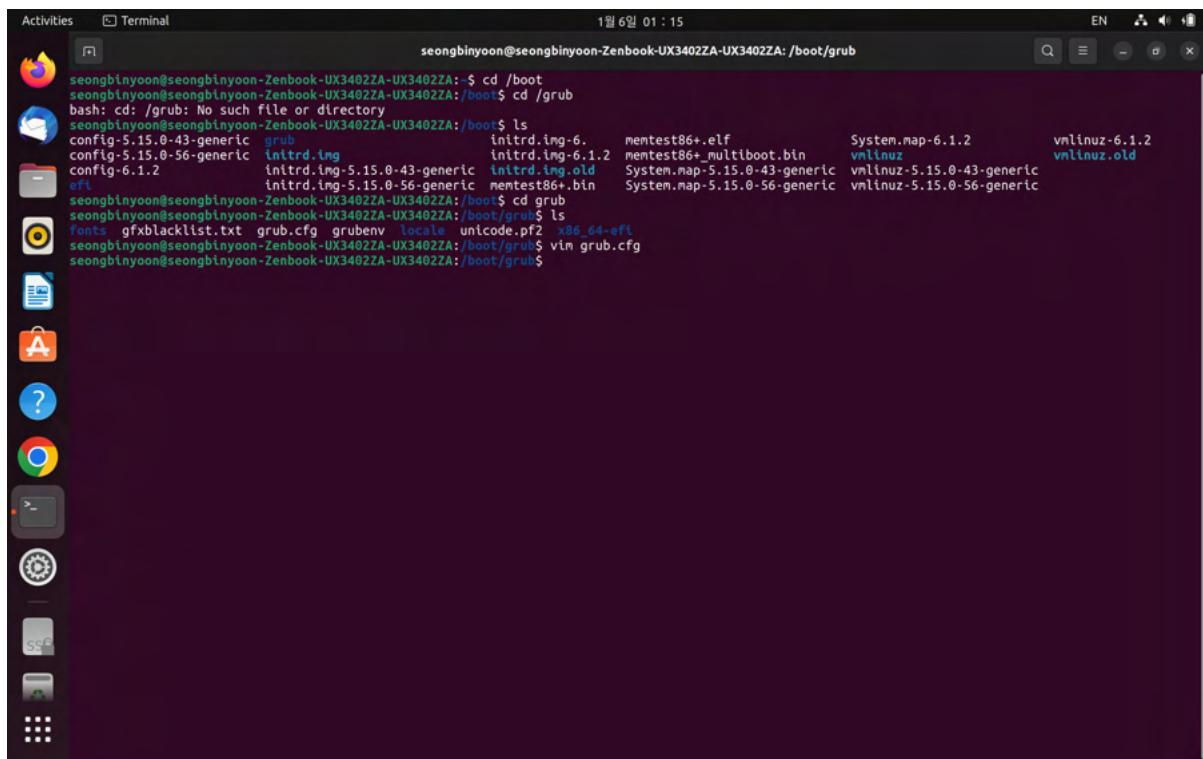
Activities Terminal 1월 6일 01:11
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /boot/grub
menuentry 'Ubuntu' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux-simple-2cf49fe4-251e-4ebf-aa30-21308df7e8aa' {
    recordfail
    load_video
    gfxmode $linux_gfx_mode
    insmod gzio
    if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
    insmod part_gpt
    insmod ext2
    search --no-floppy --fs-uuid --set=root 2cf49fe4-251e-4ebf-aa30-21308df7e8aa
    linux /boot/vmlinuz-6.1.2 root=UUID=2cf49fe4-251e-4ebf-aa30-21308df7e8aa ro quiet splash $vt_handoff
    initrd /boot/initrd.img-6.1.2
}
submenu 'Advanced options for Ubuntu' $menuentry_id_option 'gnulinux-advanced-2cf49fe4-251e-4ebf-aa30-21308df7e8aa' {
    menuentry 'Ubuntu, with Linux 6.1.2' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux-6.1.2-advanced-2cf49fe4-251e-4ebf-aa30-21308df7e8aa' {
        recordfail
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_gpt
        insmod ext2
        search --no-floppy --fs-uuid --set=root 2cf49fe4-251e-4ebf-aa30-21308df7e8aa
        echo 'Loading Linux 6.1.2 ...'
        linux /boot/vmlinuz-6.1.2 root=UUID=2cf49fe4-251e-4ebf-aa30-21308df7e8aa ro quiet splash $vt_handoff
        echo 'Loading initial ramdisk ...'
        initrd /boot/initrd.img-6.1.2
    }
    menuentry 'Ubuntu, with Linux 6.1.2 (recovery mode)' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux-6.1.2-recovery-2cf49fe4-251e-4ebf-aa30-21308df7e8aa' {
        recordfail
        load_video
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_gpt
        insmod ext2
        search --no-floppy --fs-uuid --set=root 2cf49fe4-251e-4ebf-aa30-21308df7e8aa
        echo 'Loading Linux 6.1.2 ...'
        linux /boot/vmlinuz-6.1.2 root=UUID=2cf49fe4-251e-4ebf-aa30-21308df7e8aa ro recovery nomodeset dis_ucode_ldr
        echo 'Loading initial ramdisk ...'
        initrd /boot/initrd.img-6.1.2
    }
}

```

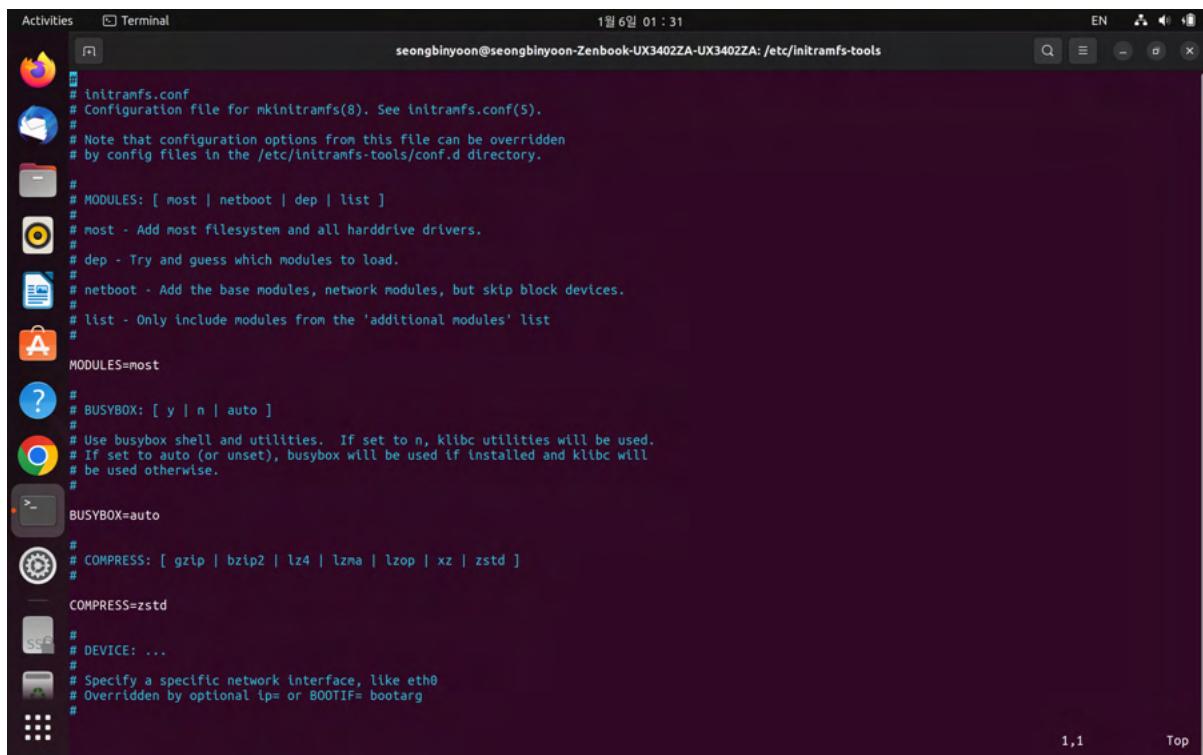
또 다른 글에서는 /boot/grub에 있는 grub.cfg를 수정해야 한다고 해서 다음과 같이 보았다.

```
~$ cd /boot/grub
```

```
~$ vim grub.cfg
```



```
Activities Terminal 1월 6일 01:15 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /boot/grub bash: cd: No such file or directory seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /boot$ cd /grub seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /boot$ ls config-5.15.0-43-generic grub initrd.img-6.1.2 memtest86+.elf System.map-6.1.2 vmlinuz-6.1.2 config-5.15.0-56-generic initrd.img initrd.img-6.1.2 memtest86+_multiboot.bin vmlinuz config-6.1.2 initrd.img-5.15.0-43-generic initrd.img.old System.map-5.15.0-43-generic vmlinuz-5.15.0-43-generic efi initrd.img-5.15.0-56-generic memtest86+.bin System.map-5.15.0-56-generic vmlinuz-5.15.0-56-generic seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /boot$ cd grub seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /boot/grub$ ls fonts gfxblacklist.txt grub.cfg grubenv locale unicode.pf2 x86_64-efi seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /boot/grub$ vim grub.cfg seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /boot/grub$
```



```
Activities Terminal 1월 6일 01:31 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: /etc/initramfs-tools bash: vim: command not found # initramfs.conf # Configuration file for mkinitramfs(8). See initramfs.conf(5). # # Note that configuration options from this file can be overridden # by config files in the /etc/initramfs-tools/conf.d directory. # # MODULES: [ most | netboot | dep | list ] # # most - Add most filesystem and all harddrive drivers. # # dep - Try and guess which modules to load. # # netboot - Add the base modules, network modules, but skip block devices. # # list - Only include modules from the 'additional modules' list # # MODULES=most # # BUSYBOX: [ y | n | auto ] # # Use busybox shell and utilities. If set to n, klibc utilities will be used. # If set to auto (or unset), busybox will be used if installed and klibc will # be used otherwise. # # BUSYBOX	auto # # COMPRESS: [ gzip | bztp2 | lz4 | lzma | lzop | xz | zstd ] # # COMPRESS=zstd # # DEVICE: ... # # Specify a specific network interface, like eth0 # Overridden by optional lp= or BOOTIF= bootarg #
```

이 부분의 MODULES=most를 dep로 바꾸라는 글이 있었지만 정확히 /etc/initramfs-tools/initramfs.conf 와, /boot/grub/grub.cfg 의 파일들이 정확히 무슨 일을 하는지, 수정하면 어떻게 변경되는지 몰라서 함부로 수정할 수가 없었다. 리눅스 커널을 공부하면서 알아보아야겠다.

Solution 1

커널 빌드(~\$ make modules_install)할 때 OS image(/boot/initrd-6.1.2) 용량이 굉장히 크게 잡히는 것이 out of memory 커널 패닉의 문제였다. 기본적으로 RAM에 할당되어 올라갈 수 있는 파일의 크기보다 크게 만들어진 것이다.

(아래의 내용으로 확인할 수 있음)

```
~$ ls -l /boot/initrd.img-6.1.2
-rw-r--r-- 1 root root 510582660 1월  5 22:06 /boot/initrd.img-6.1.2

~$ du -hs /boot/initrd.img-6.1.2
487M /boot/initrd.img-6.1.2

~$ du -hs /boot/initrd.img-5.15.0-56-generic
61M /boot/initrd.img-5.15.0-56-generic
```

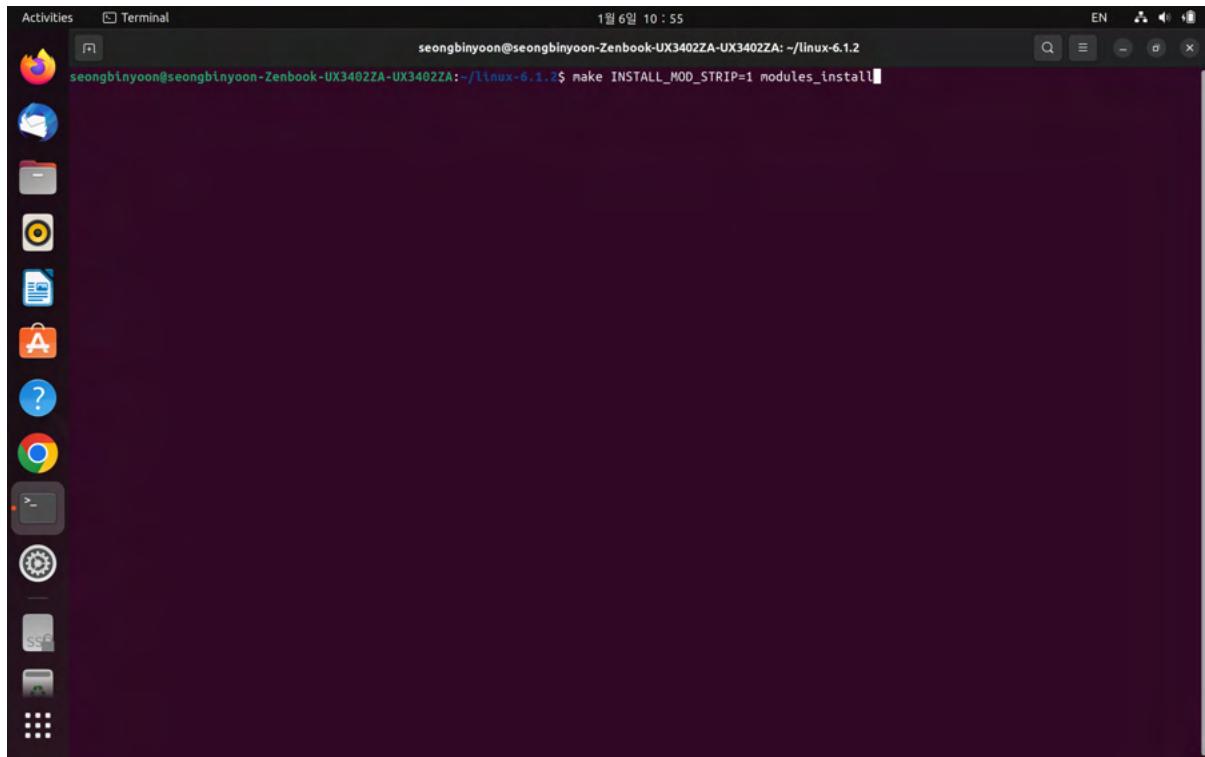
/boot/.config-6.1.2 파일을 참고해보면 CONFIG_BLK_DEV_RAM_SIZE=65536 으로 65MB임을 확인할 수 있다. 6.1.2 initrd 파일의 용량(487MB)이 메모리의 한도를 초과한 것이다. 하지만 부팅이 잘 되고 있는 5.15.0 initrd 이미지 파일의 용량은 61MB로 비교적 적다.

따라서, 아래의 명령어를 통해 STRIP 옵션이 필요없는 모듈은 이미지에 포함시키지 않게 해 사용하는 메모리의 크기를 줄일 수 있다.

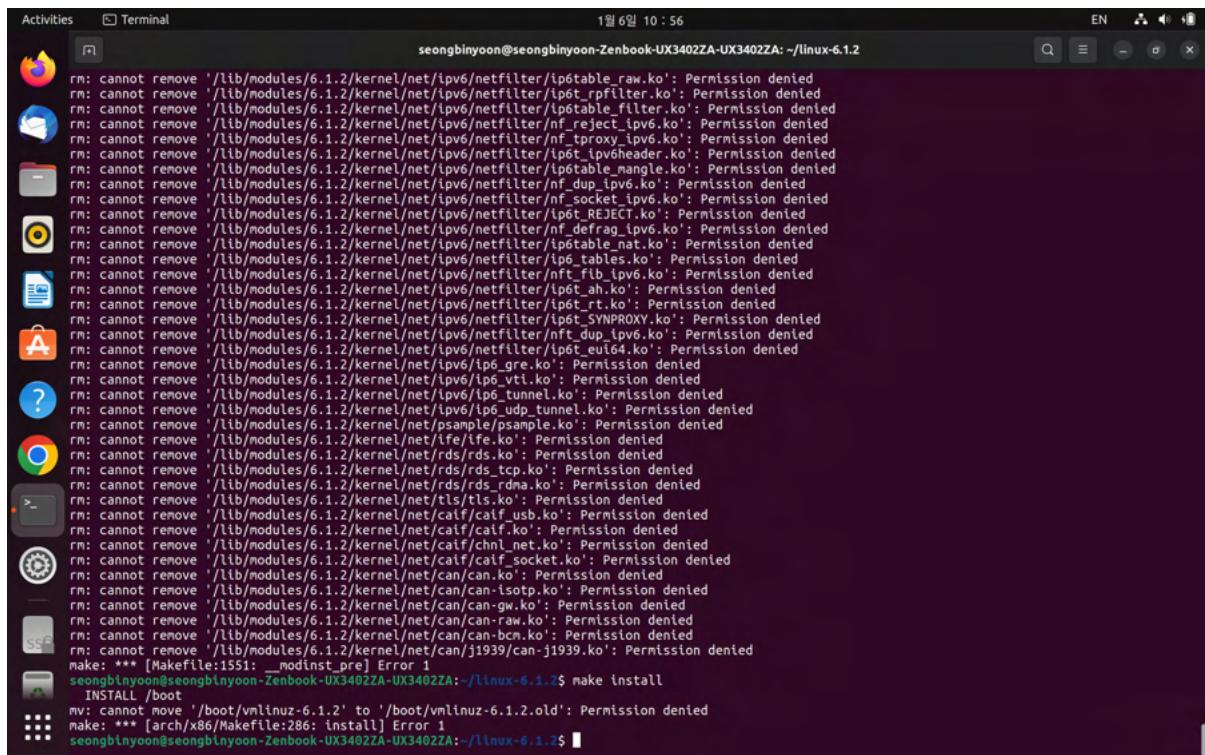
```
~$ sudo make INSTALL_MOD_STRIP=1 modules_install
```

후에 아래의 명령어를 통해 커널을 빌드한다.

```
~$ sudo make install
```



sudo 명령어를 붙이지 않고 하게 되어 Permission denied가 뜨니 우분투 설치 후 따로 root 권한을 설정해주지 않았다면 sudo를 꼭 붙이도록 하자.



```

Activities Terminal 1월 6일 11:07
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/linux-6.1.2
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-line6.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-line6.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-line6.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-pod.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-pod.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-pod.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-podhd.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-podhd.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-podhd.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-toneport.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-toneport.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-toneport.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-variax.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-variax.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/line6/snd-usb-variax.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/misc/snd-ua101.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/misc/snd-ua101.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/misc/snd-ua101.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/snd-usb-audio.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/snd-usb-audio.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/snd-usb-audio.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
STRIP /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
SIGN /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
INSTALL /lib/modules/6.1.2/kernel/sound/x86/snd-hdmi-lpe-audio.ko
STRIP /lib/modules/6.1.2/kernel/sound/x86/snd-hdmi-lpe-audio.ko
SIGN /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
INSTALL /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
STRIP /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
SIGN /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
DEPMOD /lib/modules/6.1.2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/linux-6.1.2$ sudo make install
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.1.2 /boot/vmlinuz-6.1.2
update-initramfs: Generating /boot/initrd.img-6.1.2

```

```

Activities Terminal 1월 6일 11:08
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/linux-6.1.2
INSTALL /lib/modules/6.1.2/kernel/sound/usb/snd-usbmidi-lib.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/snd-usbmidi-lib.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/snd-usbmidi-lib.ko
INSTALL /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
STRIP /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
SIGN /lib/modules/6.1.2/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
STRIP /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
SIGN /lib/modules/6.1.2/kernel/sound/virtio/virtio_snd.ko
INSTALL /lib/modules/6.1.2/kernel/sound/x86/snd-hdmi-lpe-audio.ko
STRIP /lib/modules/6.1.2/kernel/sound/x86/snd-hdmi-lpe-audio.ko
SIGN /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
INSTALL /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
STRIP /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
SIGN /lib/modules/6.1.2/kernel/sound/xen/snd_xen_front.ko
DEPMOD /lib/modules/6.1.2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/linux-6.1.2$ sudo make install
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.1.2 /boot/vmlinuz-6.1.2
update-initramfs: Generating /boot/initrd.img-6.1.2
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 6.1.2 /boot/vmlinuz-6.1.2
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.1.2 /boot/vmlinuz-6.1.2
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 6.1.2 /boot/vmlinuz-6.1.2
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.1.2 /boot/vmlinuz-6.1.2
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file...
Found linux image: /boot/vmlinuz-6.1.2
Found initrd image: /boot/initrd.img-6.1.2
Found linux image: /boot/vmlinuz-6.1.2.old
Found initrd image: /boot/initrd.img-6.1.2
Found linux image: /boot/vmlinuz-5.15.0-56-generic
Found initrd image: /boot/initrd.img-5.15.0-56-generic
Found linux image: /boot/vmlinuz-5.15.0-43-generic
Found initrd image: /boot/initrd.img-5.15.0-43-generic
Memtest86+ needs a 16-bit boot, that is not available on EFI, exiting
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot entries.
Found Windows Boot Manager on /dev/nvme0n1p0/EFI/Microsoft/Boot/bootmgfw.efi
Adding boot menu entry for UEFI Firmware Settings ...
done
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/linux-6.1.2$ 

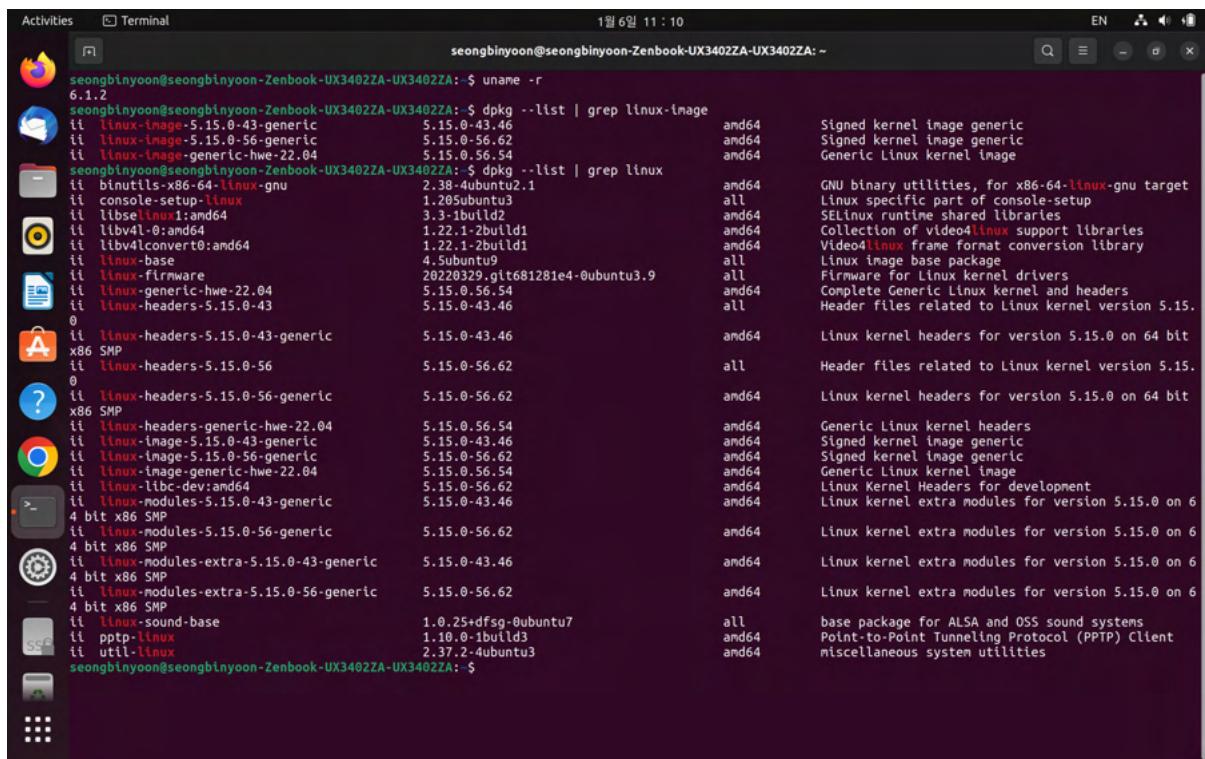
```

done을 끝으로 reboot을 해준 후 6.1.2를 부팅하니 정상적으로 돌아왔다!

```
~$ reboot
```

아래는 6.1.2 커널로 들어와 uname -r 명령어를 통해 현재의 커널 버전을 확인한 것이다.

```
~$ uname -r
```

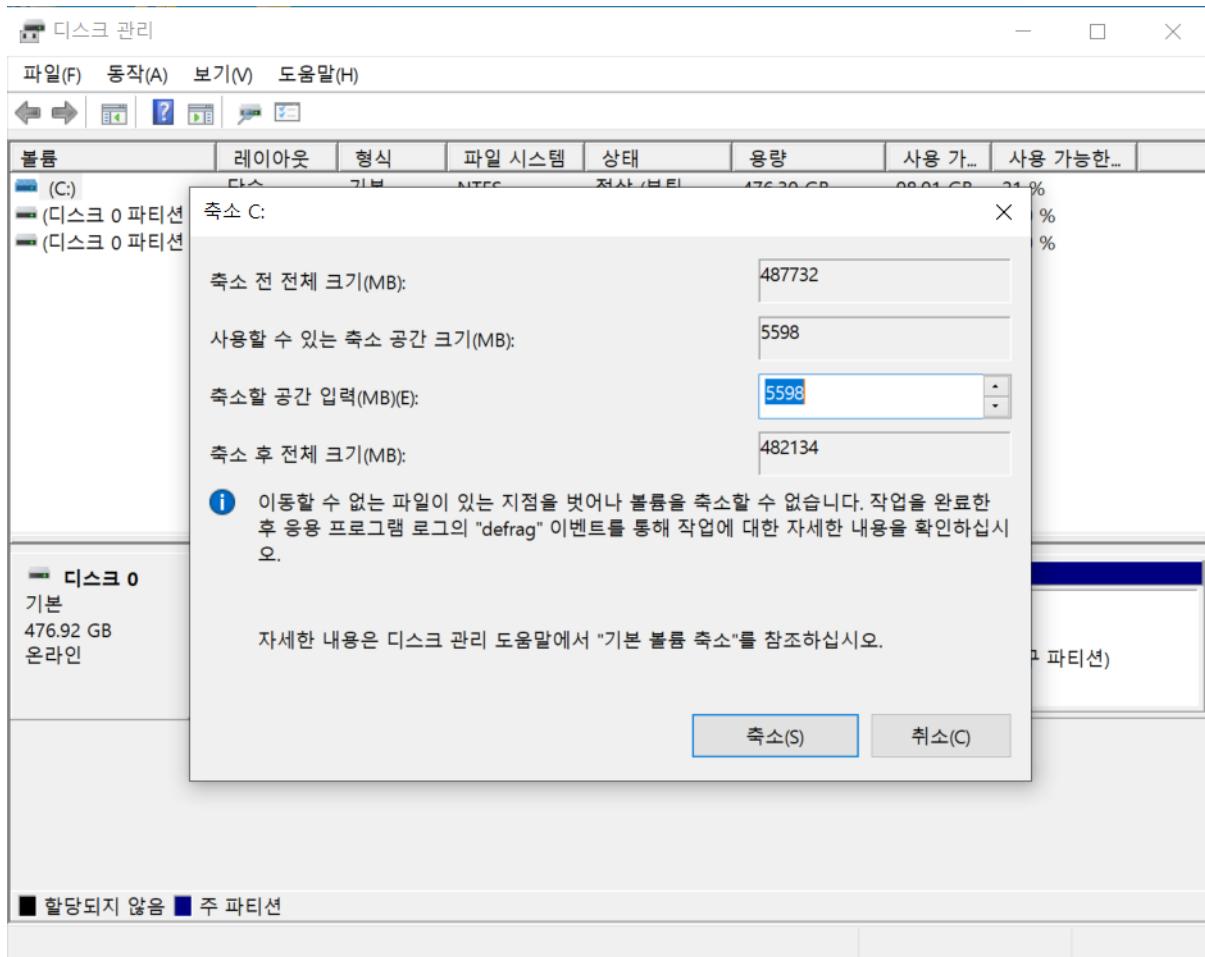


```
Activities Terminal 1월 6일 11:10 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ uname -r 6.1.2 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ dpkg --list | grep linux-image ii linux-image-5.15.0-43-generic 5.15.0-43.46 amd64 Signed kernel image generic ii linux-image-5.15.0-56-generic 5.15.0-56.62 amd64 Signed kernel image generic ii linux-image-generic-hwe-22.04 5.15.0-56.54 amd64 Generic Linux kernel image seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ dpkg --list | grep linux ii binutils-x86-64-linux-gnu 2.38-4ubuntu2.1 amd64 GNU binary utilities, for x86-64-linux-gnu target ii console-setup-linux 1.20Subuntu3 all Linux specific part of console-setup ii libselinux1:amd64 3.3-1build2 amd64 SELinux runtime shared libraries ii libv4l0:amd64 1.22.1-2build1 amd64 Collection of video4linux support libraries ii libv4lconvert0:amd64 1.22.1-2build1 amd64 Video4linux frame format conversion library ii linux-base 4.Subuntu9 all Linux image base package ii linux-firmware 20220329.git681281e4-0ubuntu3.9 all Firmware for Linux kernel drivers ii linux-generic-hwe-22.04 5.15.0-56.54 amd64 Complete Generic Linux kernel and headers ii linux-headers-5.15.0-43 5.15.0-43.46 all Header files related to Linux kernel version 5.15. 0 ii linux-headers-5.15.0-43-generic 5.15.0-43.46 amd64 Linux kernel headers for version 5.15.0 on 64 bit x86 SMP ii linux-headers-5.15.0-56 5.15.0-56.62 all Header files related to Linux kernel version 5.15. 0 ii linux-headers-5.15.0-56-generic 5.15.0-56.62 amd64 Linux kernel headers for version 5.15.0 on 64 bit x86 SMP ii linux-headers-generic-hwe-22.04 5.15.0-56.54 amd64 Generic Linux kernel headers ii linux-image-5.15.0-43-generic 5.15.0-43.46 amd64 Signed kernel image generic ii linux-image-5.15.0-56-generic 5.15.0-56.62 amd64 Signed kernel image generic ii linux-image-generic-hwe-22.04 5.15.0-56.54 amd64 Generic Linux kernel image ii linux-libc-dev:amd64 5.15.0-56.62 amd64 Linux Kernel Headers for development ii linux-modules-5.15.0-43-generic 5.15.0-43.46 amd64 Linux kernel extra modules for version 5.15.0 on 6 4 bit x86 SMP ii linux-modules-5.15.0-56-generic 5.15.0-56.62 amd64 Linux kernel extra modules for version 5.15.0 on 6 4 bit x86 SMP ii linux-modules-extra-5.15.0-43-generic 5.15.0-43.46 amd64 Linux kernel extra modules for version 5.15.0 on 6 4 bit x86 SMP ii linux-modules-extra-5.15.0-56-generic 5.15.0-56.62 amd64 Linux kernel extra modules for version 5.15.0 on 6 4 bit x86 SMP ii linux-sound-base 1.0.25+dfsg-0ubuntu7 all base package for ALSA and OSS sound systems ii pptp-linux 1.18.0-1build3 amd64 Point-to-Point Tunneling Protocol (PPTP) Client ii util-linux 2.37.2-4ubuntu3 amd64 miscellaneous system utilities seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$
```

이렇게 커널패닉이 해결되었다.

Trouble 2

리눅스 멀티부팅을 위한 파티션 분할 시도 중 부딪힌 이슈이다.



남은 C드라이브 용량 100GB 중에서 50GB 만큼의 용량만큼 축소하여 리눅스 파티션에 할당하고 싶었지만 최대로 가능한 용량은 5598MB 으로 분할이 가능한 공간이 부족했음에 따라 특별 조치를 취할 필요성을 느꼈다.

Solution 2

윈도우 시스템 보호 기능 시스템에 의한 파티션의 일부 용량 확보 때문에 생긴 현상이다.

내 PC → 속성 → 시스템 보호 → 드라이브를 선택함 → 구성 → 시스템 보호 사용 안 함 → 확인

해당 솔루션을 바탕으로 축소할 수 있는 용량이 늘어났음을 확인할 수 있었고, 원하는 용량만큼의 파티션 분할에 성공하였다.

Trouble 3

우분투 멀티부팅 설치 후 Grub2 실행이 안 되는 현상 발생 (윈도우 기준 리눅스 멀티부팅 설치했을 시)

Solution 3

부트로더가 윈도우 EFI 파티션에 제대로 설치가 되지 않은 것으로 윈도우에서 BCD(Boot Configuration Editor)를 활용하여 부트로더를 직접 설정해주면 된다.

```
bcdeedit /set {bootmgr} path \EFI\ubuntu\grubx64.efi
```

관리자 권한으로 명령프롬프트에서 실행하면 시스템 종료 후 부팅했을 때 Grub2가 정상 실행하는 것을 발견할 수 있다.

Trouble 4

우분투OS 사용 중 프리징 (화면 멈춤) 현상이 종종 일어나면서 작업에 방해가 되었다.

프리징 현상이 일어나면 화면이 멈추고 마우스, 키보드와 같은 입력 장치도 먹통이 되며 전원 버튼을 통해 강제로 껐다가 켜야 하는 불상사가 발생한다.

이런 이유로 프리징에 대한 해결을 필요로 하였다.

Solution 4

NVIDIA GPU 드라이버와 OS 사이에서의 충돌되는 부분을 수정해주면서 문제를 해결할 수 있다.

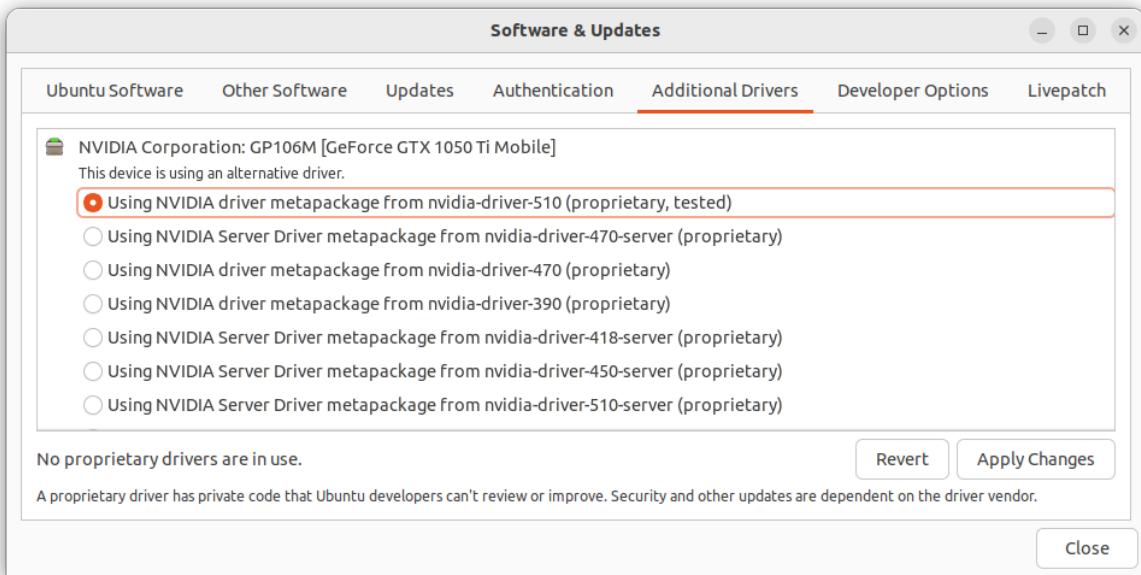
grub 설정에 들어간 뒤에,

```
$ sudo vim /etc/default/grub
```

아래의 옵션(nomodeset)을 추가해준다.

```
GRUB_CMDLINE_LINUX_DEFAULT = "quiet splash nomodeset"
```

Software & Update → Additional Drivers 에 들어가서 tested 로 등록된 드라이버로 교환해주면 된다.



Apply Changes 를 클릭하고 재부팅하면 정상적으로 전환됐음을 확인할 수 있다.

Directory Contents

/proc/devices

- device의 이름과 device number range를 담고 있다.
- cat /proc/devices 명령어로 현재 등록된 디바이스 드라이버 정보를 볼 수 있다.

/dev

- char, block 등 device에 대한 정보를 담고 있다.
- 해당 디렉토리에서 ls -l 명령어를 입력하면 device의 type등에 대한 정보를 볼 수 있다.

/usr/include/linux

- device number를 정보에 대한 파일인 types.h를 담고 있다.
- dev_t 타입에 대한 macro 함수에 대한 파일인 kdev_t.h를 담고 있다.
- device number의 등록, 할당, 해제(free)에 대한 함수에 대한 파일인 fs.h를 담고 있다.

```
/usr/src/linux/Documentation/devices.txt
```

- 현재 Linux 시스템에서 정의되어 있는 device들의 목록, major number, minor number 등에 대한 정보를 담고 있다.

```
/linux-6.1.2/include/linux/cdev.h
```

- char device driver를 관리하는 cdev구조체를 담고 있다.
- cdev 구조체 안에는 ops(디바이스에서 정의된 file operations), list(cdev list), dev(주번호와 부번호가 각각 저장되어 있는 dev_t 타입의 디바이스 번호), count 등이 정의되어 있다.

```
/linux-6.1.2/include/asm-generic/uaccess.h
```

- driver가 user-space buffer에 안전하게 접근하기 위해서 필요한 특별한 kernel-supplied 함수를 담고 있다.

```
/linux/errno.h
```

- 에러에 의해 반환되는 리턴값에 대한 정보를 담고 있다.
- read, write함수에 의한 negative values, -EINTR(interrupted system call), -EFAULT(bad address) 등을 포함한다.

```
/lib/modules/<current kernel version>/build/include/linux
```

- 현재 컴파일 되어있는 커널 버전의 include 파일을 담고 있다.
- 디바이스 드라이버에 적용되는 각 헤더파일 및 그 안의 함수를 담고 있다.

Linux Kernel Module Build

Linux kernel module build 를 예제 파일에 있는 내용을 바탕으로 진행해보려고 한다.

전체 과정을 간단하게 요약하자면, 다음과 같다.

Linux Kernel Module Exercise Process

1. 커널 모듈에 대한 C 코드 작성
2. Makefile 를 통한 빌드 후 .ko (kernel object) 생성
3. .ko (kernel object) 파일 커널 모듈로 등록 (insmod)
4. 등록된 모듈 확인 (lsmod)
5. 등록된 모듈 제거 (rmmod)

1. 커널 모듈에 대한 C 코드 작성

hello.c 파일에 다음과 같이 입력한다.

<hello.c>

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
MODULE_LICENSE("Dual BSD/GPL");

// Initialization Function
static int __init hello_init(void)
{
    printk(KERN_ALERT "Hello, world\n");
    return 0;
}

// Cleanup Function
static void __exit hello_exit(void)
{
    printk(KERN_ALERT "Goodbye, cruel world\n");
}

// Macros
module_init(hello_init);
module_exit(hello_exit);
```

간단하게 모듈을 입력하는 과정이다. 특별한 점으로는 cleanup function은 void 타입이고, 코드의 맨아래 매크로 함수로 모듈에 추가될 때와 제거될 때 호출할 함수를 미리 추가하는 것이 있다. (module_init, module_exit 프로시저 확인)

매크로 함수는 선언하지 않으면 initialization function 및 cleanup function이 호출되지 않아 커널로 하여금 모듈의 로드 및 언로드가 불가능하다.

(또한, __init, __exit 은 시작과 끝 함수임을 다른 함수들로부터 구분하기 쉽도록 사용하는 매크로이며, 해당 코드들이 오직 load와 unload를 위한 코드임을 명시한다. 따라서, 다른 용도로 쓰이는 __init 및 __exit은 오류가 발생할 수 있다.)

2. Makefile 를 통한 빌드 후 .ko (kernel object) 생성

커널 모듈을 컴파일하는데에는 Makefile과 make 명령이 필요하다.

같은 폴더에 Makefile 파일을 만들어 다음과 같이 입력한다.

<Makefile>

```
##### Makefile #####
ifeq ($(KERNELRELEASE),)
obj-m := hello.o

else
KERNELDIR ?= /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

default:
$(MAKE) -C $(KERNELDIR) M=$(PWD) modules

endif
```

간단하게 obj-m 변수에만 'hello.o'를 추가하여 빌드 대상을 'hello.c'에만 설정되도록 하였다.

Makefile 문법에 따르면,

'?=의 경우, 변수가 할당되지 않았으면 등록해주는 연산자이다.

':=의 경우, 우측 항에 대한 값을 매번 계산해서 할당하는 것이 아닌 첫 번째 선언됐을 때 할당되는 값이 고정되는 연산자이다.

'='의 경우, 반대로, 변수가 사용될 때마다 우측 항에 대한 식이 계산된다.

ifneq는 조건을 시작하고 지정하며, 괄호 안 콤마(,)로 구분된 두 매개변수가 일치할 경우 아래 코드를 수행한다.

else는 ifneq의 조건을 만족하지 않았다면 수행한다.

모든 조건문은 endif로 종료한다.

```
ifneq ($(KERNELRELEASE),) # 만약 환경변수 KERNELRELEASE가 정의되어 있지 않다면 다음을 추가
```

```
obj-m := hello.o # 컴파일할 모듈의 이름
```

```
# KERNELDIR: 커널 디렉토리
# shell uname -r: 커널의 최신버전 문자열로 치환
# build: 커널 버전에 맞는 빌드 디렉토리
KERNELDIR ?= /lib/modules/$(shell uname -r)/build
```

```
# 소스코드가 있는 디렉토리
# PWD: Print Working Directory, 현재 위치하고 있는 디렉토리를 출력(해당 명령어를 입력한 디렉토리를 출력)
PWD := $(shell pwd)
```

```
# default: 뒤에 아무 옵션을 주지 않았을 때 실행되는 명령어
# KERNELDIR: 커널 디렉토리
# M: 서브 디렉토리
# modules: 모듈 컴파일
default:
$(MAKE) -C $(KERNELDIR) M=$(PWD) modules
```

만약 여러 파일이 빌드 대상이라면 module-objs 변수에 확장명을 .o 로 변경하여 할당하면 정상적으로 빌드될 것이다.

예를 들어, module.o 가 file1.o, file2.o로 이루어진 파일이라면 다음과 같이 작성한다.

주의할 점은, module-objs에 나열되는 오브젝트 파일에는 obj-m에 있는 파일 이름이 존재해서는 안된다.

```
obj-m := module.o
module-objs := file1.o file2.o
```

이후 다음과 같이 같은 디렉토리에서 make를 통해 빌드하게 된다면 다음 메세지와 함께 .ko 파일이 생성되는 것을 확인할 수 있다.

(일반 어플리케이션으로 컴파일을 하면 실행가능한 바이너리 파일이 만들어지지만, 커널 모듈로 컴파일을 하면 오브젝트 파일이 생성된다.)

```
devtae@devtae:~/sources$ sudo make
make -C /lib/modules/6.1.2/build M=/home/devtae/sources modules
make[1]: Entering directory '/home/devtae/Downloads/linux-6.1.2'
  CC [M] /home/devtae/sources/hello.o
  MODPOST /home/devtae/sources/Module.symvers
  CC [M] /home/devtae/sources/hello.mod.o
  LD [M] /home/devtae/sources/hello.ko
make[1]: Leaving directory '/home/devtae/Downloads/linux-6.1.2'
devtae@devtae:~/sources$ ls
hello.c    hello.mod    hello.mod.o  Makefile      modules.order
hello.ko   hello.mod.c  hello.o       Makefile_bak  Module.symvers
```

3. .ko (kernel object) 파일 커널 모듈로 등록 (insmod)

이어서 그대로 다음 방법과 같이 커널 모듈을 추가할 수 있다.

```
~$ sudo insmod hello.ko
```

printk(KERN_ALERT "Hello, world\n"); 에 대한 출력 결과는 콘솔로 뜨지 않고, dmesg를 통해 확인할 수 있다.

```
devtae@devtae:~/sources$ sudo dmesg | grep 'Hello, world'  
[ 320.788529] Hello, world
```

또한, /var/log/kern.log 파일에서 출력 결과를 확인할 수 있다.

insmod로 로드하면 Hello, world가, rmmod로 언로드하면 Goodbye, cruel world가 출력되는 것을 볼 수 있다.

```
~$ cat /var/log/kern.log | grep '필터링 문자'
```

```
Jan 11 13:34:29 seongbinyoon-Zenbook-UX3402ZA-UX3402ZA kernel: [71115.519067] Hello, world  
Jan 11 13:35:31 seongbinyoon-Zenbook-UX3402ZA-UX3402ZA kernel: [71177.395576] Goodbye, cruel world  
Jan 11 13:37:40 seongbinyoon-Zenbook-UX3402ZA-UX3402ZA kernel: [71306.424407] Hello, world  
Jan 11 13:49:13 seongbinyoon-Zenbook-UX3402ZA-UX3402ZA kernel: [71998.863400] Goodbye, cruel world  
Jan 11 13:57:48 seongbinyoon-Zenbook-UX3402ZA-UX3402ZA kernel: [72514.450647] Hello, world  
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:/var/log$
```

4. 등록된 모듈 확인 (lsmod)

다음과 같이 코드를 입력하면 확인할 수 있다.

```
~$ lsmod | grep hello
```

```
devtae@devtae:~/sources$ lsmod | grep hello  
hello 16384 0
```

모듈이 커널에 적재가 잘 되었다면 /proc/modules 파일에서도 확인이 가능하다.

```
hello 16384 0 - Live 0x0000000000000000 (0E)
```

5. 등록된 모듈 제거 (rmmod)

다음과 같이 코드를 입력하여 등록된 모듈을 제거할 수 있다.

```
$ sudo rmmod hello.ko
```

그 결과, 다음과 같이 module_exit에 등록한 함수의 출력 결과가 나와 있었고, 등록된 모듈 리스트에서 사라진 것을 확인할 수 있다.

```
[ 632.107497] Goodbye, curel world
devtae@devtae:~/sources$ lsmod | grep hello
devtae@devtae:~/sources$
```

Character Device Driver

Device?

- 컴퓨터와 연결된 여러 주변 장치
- 예를 들어, 네트워크 어댑터, LCD 디스플레이, 오디오, 터미널, 키보드, 하드디스크, 플로피디스크, 프린터

Device driver?

- 위와 같은 디바이스를 컨트롤하기 위한 드라이버
- 실제 장치들을 추상화시켜 사용자 프로그램이 정형화된 인터페이스를 통해 디바이스를 접근할 수 있게 해주는 프로그램
- 모든 것을 파일로 관리하는 리눅스에서는 디바이스 드라이버 또한 파일로 관리
- /dev 아래에 있는 파일들이 디바이스 드라이버 인터페이스
- 하드웨어와는 독립적으로 응용프로그램이 file open, read 등과 같은 함수로 접근.

Char device driver?

- 디바이스를 파일처럼 접근하여 직접 read/write를 수행한다.
- 버퍼 캐шу를 사용하지 않는다.
- 자료의 순차성을 지닌다. (시간 순으로 들어오는 데이터를 처리한다.)
- 마우스, 키보드, 사운드 카드, 터미널 등
- 아래와 같이 /dev에서 ls -al 명령어를 입력하면 나타나는 목록이 device 목록인데, 맨 앞 문자가 c이면 char device, b이면 block device이다.
- 오른쪽 ,로 이루어진 두개의 숫자는 각각 Major device number(디바이스 유형)와 minor device number(디바이스 단위)를 의미한다.

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:/dev$ ls -al
total 4
drwxr-xr-x 20 root      root        4900  1월 13 13:09 .
drwxr-xr-x 20 root      root        4096  1월  4 19:27 ..
crw---- 1 root      root        10,   122  1월 11 22:34 acpi_thermal_rel
crw---- 1 root      root        10,   235  1월 11 22:34 autofs
drwxr-xr-x 2 root      root        540  1월 11 22:39 block
crw---- 1 root      disk       10,   234  1월 15 12:30 btrfs-control
drwxr-xr-x 3 root      root        60   1월 11 22:34 bus
drwxr-xr-x 2 root      root        5400  1월 15 12:30 char
crw---- 1 root      tty       5,     1  1월 11 22:34 console
lrwxrwxrwx 1 root      root        11   1월 11 22:34 core -> /proc/kcore
drwxr-xr-x 18 root     root        360  1월 11 22:34 cpu
crw---- 1 root      root        10,   123  1월 11 22:34 cpu_dma_latency
crw---- 1 root      root        10,   203  1월 11 22:34 cuse
drwxr-xr-x 8 root      root        160  1월 11 22:34 disk
drwxr-xr-x 2 root      root        60   1월 11 22:34 dma_heap
drwxr-xr-x 3 root      root        100  1월 11 22:34 dri
crw---- 1 root      root        510,    0  1월 11 22:34 drm_dp_aux0
crw---- 1 root      root        510,    1  1월 11 22:34 drm_dp_aux1
crw---- 1 root      root        510,    2  1월 11 22:34 drm_dp_aux2
crw---- 1 root      root        510,    3  1월 11 22:34 drm_dp_aux3
crw---- 1 root      root        510,    4  1월 11 22:34 drm_dp_aux4
crw---- 1 root      root        10,   125  1월 11 22:34 encryptfs
crw---- 1 root      video     29,    0  1월 11 22:34 fb0
lrwxrwxrwx 1 root      root        13   1월 11 22:34 fd -> /proc/self/fd
crw---- 1 root      root        1,     7  1월 11 22:34 full
crw---- 1 root      root        10,   229  1월 11 22:34 fuse
crw---- 1 root      root        254,    0  1월 11 22:34 gpiochip0
crw---- 1 root      root        239,    0  1월 11 22:34 hidraw0
crw---- 1 root      root        239,    1  1월 13 13:09 hidraw1
crw---- 1 root      root        239,    2  1월 13 13:09 hidraw2
crw---- 1 root      root        239,    3  1월 13 13:09 hidraw3
crw---- 1 root      root        10,   228  1월 11 22:34 hpet
drwxr-xr-x 2 root      root        0   1월 11 22:34 hugepages
crw---- 1 root      root        10,   183  1월 11 22:34 hwrng
crw---- 1 root      root        89,    0  1월 11 22:34 i2c-0
crw---- 1 root      root        89,    1  1월 11 22:34 i2c-1
crw---- 1 root      root        89,    10  1월 11 22:34 i2c-10
crw---- 1 root      root        89,    11  1월 11 22:34 i2c-11
crw---- 1 root      root        89,    12  1월 11 22:34 i2c-12
crw---- 1 root      root        89,    13  1월 11 22:34 i2c-13
crw---- 1 root      root        89,    14  1월 11 22:34 i2c-14
```

Char device driver programming(scull0~scull3)

1. 소스코드 작성

Makefile

```
#If KERNELRELEASE is defined, we've been invoked from the kernel build system and can use its language.  
ifeq ($(KERNELRELEASE),)  
    obj-m := scull.o  
  
#Otherwise we were called directly from the command line; invoke the kernel build system.  
else  
    KERNELDIR ?= /lib/modules/$(shell uname -r)/build  
    PWD := $(shell pwd)  
  
default:  
    $(MAKE) -C $(KERNELDIR) M=$(PWD) modules  
endif
```

```
ifeq ($(KERNELRELEASE), )
```

- 만약 환경변수 KERNELRELEASE가 정의되어 있지 않다면 다음을 추가

```
obj-m := scull.o
```

- 컴파일할 모듈의 이름(scull.o)

```
KERNELDIR ?= /lib/modules/$(shell uname -r)/build
```

- 커널 디렉토리(KERNELDIR) 변수에 커널의 최신버전 문자열로 치환한 커널 버전에 맞는 빌드 디렉토리를 등록

```
PWD := $(shell pwd)
```

- PWD(print working directory) 변수에 현재 소스코드가 위치하고 있는 디렉토리(해당 명령어를 입력하고 있는 디렉토리)를 등록
- 해당 연산자 := 는 한번 등록하면 고정

```
# default: 뒤에 아무 옵션을 주지 않았을 때 실행되는 명령어  
# KERNELDIR: 커널 디렉토리  
# M: 서브 디렉토리  
# modules: 모듈 컴파일  
default:  
    $(MAKE) -C $(KERNELDIR) M=$(PWD) modules
```

- 커널 디렉토리(KERNELDIR)에 현재 모듈이 위치하고 있는 디렉토리를 서브 디렉토리로 인식시켜 모듈(target)을 컴파일

scull_load

```

#!/bin/sh
module="scull"
device="scull"
mode="664"

# Invoke insmod with all arguments we got
# And use a pathname, as newer modutils don't look in . by default
/sbin/insmod ./${module}.ko $* || exit 1

# Remove stale nodes
rm -f /dev/${device}[0-3]

major=$(awk "\$2==\"$module\" {print \$1}" /proc/devices)

mknod /dev/${device}0 c $major 0
mknod /dev/${device}1 c $major 1
mknod /dev/${device}2 c $major 2
mknod /dev/${device}3 c $major 3

# Give appropriate group/permissions, and change the group.
# Not all distributions have staff, some have "wheel" instead.
group="staff"
grep -q '^staff:' /etc/group || group="wheel"

chgrp $group /dev/${device}[0-3]
chmod $mode /dev/${device}[0-3]

~
scull_load

```

21,62

All

```
#!/bin/sh
```

- #! 로 스크립트를 실행할 쉘을 지정
- 이 경우 스크립트를 실행하면 아래와 같이 /bin/sh에 연결된 dash 쉘의 도움을 받아 실행한다. (이전에는 bash 였다)

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ ls -al /bin/sh
lrwxrwxrwx 1 root root 4  1월  4 19:23 /bin/sh -> dash

```

```
/sbin/insmod ./${module}.ko $* || exit 1
```

- insmod로 scull.ko 드라이버를 커널에 적재

```
rm -f /dev/${device}[0-3]
```

- mknod로 드라이버 파일을 만들기 전, 남아있던 낡은 노드(드라이버 파일)를 제거(초기화)

```
major=$(awk "\$2==\"$module\" {print \$1}" /proc/devices)
```

- awk를 사용하여 /proc/devices에 할당된 주번호를 찾아 major 변수에 입력
- /proc/devices 파일에서 첫 번째 열에 할당된 major 번호가 나오고, 두 번째 열에서 dev 이름이 나온다.

```
mknod /dev/${device}0 c $major 0
mknod /dev/${device}1 c $major 1
mknod /dev/${device}2 c $major 2
mknod /dev/${device}3 c $major 3
```

- mknod 명령어로 파일 시스템에서 4개의 디바이스 노드(드라이버 파일)를 생성
- 예를 들어, 'mknod /dev/\$scull0 c 254 0' 과 같이 name0| scull0, 주번호가 254, 부번호가 0인 char device file을 생성
- 제거 시에는 rmnod를 사용

scull_unload

```
#!/bin/sh
module="scull"
device="scull"

# Invoke rmmod with all arguments we got
/sbin/rmmod $module $* || exit 1

# Remove stale nodes
rm -f /dev/${device} /dev/${device}[0-3]
```

```
/sbin/rmmod $module $* || exit 1
```

- rmmod로 커널에 적재했던 scull 드라이버를 해제

```
rm -f /dev/${device} /dev/${device}[0-3]
```

- /dev/scull 디렉토리에 있는 4개의 디바이스 노드(드라이버 파일) scull0, scull1, scull2, scull3을 삭제

scull.h

```
/* This is header file 'scull.h', and it contains definitions of the scull file */

/* File name */
#ifndef _SCULL_H
#define _SCULL_H

/* Dynamic major number by default */
#ifndef SCULL_MAJOR
#define SCULL_MAJOR 0
#endif

/* scull0 to scull3 */
#ifndef SCULL_NUM_DEV
#define SCULL_NUM_DEV 4
#endif

/* Memory area */
#ifndef SCULL_QUANTUM
#define SCULL_QUANTUM 4000
#endif

/* Array of memory area */
#ifndef SCULL_QSET
#define SCULL_QSET 1000
#endif

/* Define scull quantum set */
struct scull_qset {
    void **data;
    struct scull_qset *next;
};

/* Define scull devices */
struct scull_dev {
    struct scull_qset *data;          /* Pointer to first quantum set */
    int quantum;                    /* the current quantum size */
    int qset;                       /* the current array size */
    unsigned long size;             /* amount of data stored here */
}
```

```
/* Global variables (main.c) */
extern int scull_major;
extern int scull_num_dev;
extern int scull_quantum;
extern int scull_qset;

/* Prototypes for shared functions */
int scull_open(struct inode *inode, struct file *filp);
int scull_release(struct inode *inode, struct file *filp);
int scull_trim(struct scull_dev *dev);
ssize_t scull_read(struct file *filp, char __user *buf, size_t count, loff_t *f_pos);
ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos);
loff_t scull_llseek(struct file *filp, loff_t off, int whence);

#endif
"scull.h" 67L, 1569B
```

- char device driver인 scull의 구조체 등 정의를 담고있는 헤더파일이다.

scull.c

```
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/slab.h>
#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/types.h>
#include <linux/fcntl.h>
#include <linux/cdev.h>
#include <linux/proc_fs.h>
#include <linux/seq_file.h>
#include <asm/uaccess.h>
#include "scull.h"

/* Parameters */
int scull_major = SCULL_MAJOR; /* major number */
int scull_minor = 0; /* minor number */
int scull_num_dev = SCULL_NUM_DEV; /* number of bare scull devices */
int scull_quantum = SCULL_QUANTUM;
int scull_qset = SCULL_QSET;

module_param(scull_major, int, S_IRUGO);
module_param(scull_minor, int, S_IRUGO);
module_param(scull_num_dev, int, S_IRUGO);
module_param(scull_quantum, int, S_IRUGO);
module_param(scull_qset, int, S_IRUGO);

MODULE_AUTHOR("Alessandro Rubini, Jonathan Corbet");
MODULE_LICENSE("Dual BSD/GPL");

/* Define scull devices */
struct scull_dev *scull_devices; /* allocated in scull_init_module */
```

```

int scull_trim(struct scull_dev *dev) {
    struct scull_qset *next, *dptr;
    int qset = dev->qset; /* "dev" is not-null */
    int i;

    for (dptr = dev->data; dptr; dptr = next) { /* all the list items */
        if (dptr->data) {
            for (i = 0; i < qset; i++)
                kfree(dptr->data[i]);
            kfree(dptr->data);
            dptr->data = NULL;
        }
        next = dptr->next;
        kfree(dptr);
    }
    dev->size = 0;
    dev->quantum = scull_quantum;
    dev->qset = scull_qset;
    dev->data = NULL;
    return 0;
}

/* Open */
int scull_open(struct inode *inode, struct file *filp) {
    struct scull_dev *dev; /* device information */

    dev = container_of(inode->i_cdev, struct scull_dev, cdev);
    filp->private_data = dev; /* for other methods */

    /* now trim to 0 to the length of the device if open was write-only */
    if ((filp->f_flags & O_ACCMODE) == O_WRONLY) {
        scull_trim(dev); /* ignore errors */
    }
    return 0; /* success */
}

```

```

/* Release */
int scull_release(struct inode *inode, struct file *filp) {
    return 0;
}

/* Follow the list */
static struct scull_qset *scull_follow(struct scull_dev *dev, int n) {
    struct scull_qset *qs = dev->data;

    /* Allocate first qset explicitly if need be */
    if (!qs) {
        qs = dev->data = kmalloc(sizeof(struct scull_qset), GFP_KERNEL);
        if (qs == NULL)
            return NULL; /* Never mind */

        memset(qs, 0, sizeof(struct scull_qset));
    }

    /* Then follow the list */
    while (n--) {
        if (!qs->next) {
            qs->next = kmalloc(sizeof(struct scull_qset), GFP_KERNEL);
            if (qs->next == NULL)
                return NULL; /* Never mind */

            memset(qs->next, 0, sizeof(struct scull_qset));
        }
        qs = qs->next;
        continue;
    }
    return qs;
}

```

```

/* Read */
ssize_t scull_read(struct file *filp, char __user *buf, size_t count, loff_t *f_pos) {
    struct scull_dev *dev = filp->private_data;
    struct scull_qset *dptr;           /* the first listitem */
    int quantum = dev->quantum, qset = dev->qset;
    int itemsize = quantum * qset;   /* how many bytes in the listitem */
    int item, s_pos, q_pos, rest;
    ssize_t retval = 0;

    if (down_interruptible(&dev->sem))
        return -ERESTARTSYS;
    if (*f_pos >= dev->size)
        goto out;
    if (*f_pos + count > dev->size)
        count = dev->size - *f_pos;

    /* find listitem, qset index, and offset in the quantum */
    item = (long)*f_pos / itemsize;
    rest = (long)*f_pos % itemsize;
    s_pos = rest / quantum; q_pos = rest % quantum;

    /* follow the list up to the right position(defined elsewhere) */
    dptr = scull_follow(dev, item);

    if (dptr == NULL || !dptr->data || !dptr->data[s_pos])
        goto out;          /* don't fill holes */

    /* read only up to the end of this quantum */
    if (count > quantum - q_pos)
        count = quantum - q_pos;

    if (copy_to_user(buf, dptr->data[s_pos] + q_pos, count)) {
        retval = -EFAULT;
        goto out;
    }
    *f_pos += count;
    retval = count;
}

out:
    up(&dev->sem);
    return retval;
}

```

```

/* Write */
ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos) {
    struct scull_dev *dev = filp->private_data;
    struct scull_qset *dptr;
    int quantum = dev->quantum, qset = dev->qset;
    int itemsize = quantum * qset;
    int item, s_pos, q_pos, rest;
    ssize_t retval = -ENOMEM; /* value used in "goto out" statements */

    if (down_interruptible(&dev->sem))
        return -ERESTARTSYS;

    /* find listitem, qset index and offset in the quantum */
    item = (long)*f_pos / itemsize;
    rest = (long)*f_pos % itemsize;
    s_pos = rest / quantum; q_pos = rest % quantum;

    /* follow the list up to the right position */
    dptr = scull_follow(dev, item);
    if (dptr == NULL)
        goto out;
    if (!dptr->data) {
        dptr->data = kmalloc(qset * sizeof(char *), GFP_KERNEL);
        if (!dptr->data)
            goto out;
        memset(dptr->data, 0, qset * sizeof(char *));
    }
    if (!dptr->data[s_pos]) {
        dptr->data[s_pos] = kmalloc(quantum, GFP_KERNEL);
        if (!dptr->data[s_pos])
            goto out;
    }
    /* write only up to the end of this quantum */
    if (count > quantum - q_pos)
        count = quantum - q_pos;

    if (copy_from_user(dptr->data[s_pos]+q_pos, buf, count)) {
        retval = -EFAULT;
        goto out;
    }
    *f_pos += count;
    retval = count;
}

```

```

/* update the size */
if (dev->size < *f_pos)
    dev->size = *f_pos;

out:
    up(&dev->sem);
    return retval;
}

/* Change the current read/write position in a file */
/* New position is returned as a positive return value */
loff_t scull_llseek(struct file *filp, loff_t off, int whence) {
    struct scull_dev *dev = filp->private_data;
    loff_t newpos;

    switch(whence) {
        case 0: /* SEEK_SET */
            newpos = off;
            break;

        case 1: /* SEEK_CUR */
            newpos = filp->f_pos + off;
            break;

        case 2: /* SEEK_END */
            newpos = dev->size + off;
            break;

        default: /* can't happen */
            return -EINVAL;
    }
    if (newpos < 0) return -EINVAL;
    filp->f_pos = newpos;
    return newpos;
}

```

```

/* Set file operations */
struct file_operations scull_fops = {
    .owner = THIS_MODULE,
    .llseek = scull_llseek,
    .read = scull_read,
    .write = scull_write,
    .open = scull_open,
    .release = scull_release
};

/* Cleanup module */
void scull_cleanup_module(void) {
    int i;
    dev_t devno = MKDEV(scull_major, scull_minor);

    /* Get rid of our char dev entries */
    if (scull_devices) {
        for (i = 0; i < scull_num_dev; i++) {
            scull_trim(scull_devices + i);
            cdev_del(&scull_devices[i].cdev);
        }
        kfree(scull_devices);
    }

    /* Unregister */
    /* cleanup_module is never called if registering failed */
    unregister_chrdev_region(devno, scull_num_dev);
}

```

```

/* Set cdev */
static void scull_setup_cdev(struct scull_dev *dev, int index) {
    int err, devno;

    devno = MKDEV(scull_major, scull_minor + index);
    cdev_init(&dev->cdev, &scull_fops);
    dev->cdev.owner = THIS_MODULE;
    dev->cdev.ops = &scull_fops;
    err = cdev_add (&dev->cdev, devno, 1);

    /* Fail gracefully if need be */
    if (err)
        printk(KERN_NOTICE "Error %d adding scull%d", err, index);
}

```

```

/* Initialize module */
int scull_init_module(void) {
    int result, i;
    dev_t dev = 0;

    if (scull_major) {
        dev = MKDEV(scull_major, scull_minor);
        /* Register */
        result = register_chrdev_region(dev, scull_num_dev, "scull");
    }
    else {
        result = alloc_chrdev_region(&dev, scull_minor, scull_num_dev, "scull");
        scull_major = MAJOR(dev);
    }
    if (result < 0) {
        printk(KERN_WARNING "scull: can't get major %d\n", scull_major);
        return result;
    }

    /* allocate the devices, we can't have them static, as the number
     * can be specified at load time
     */
    scull_devices = kmalloc(scull_num_dev * sizeof(struct scull_dev), GFP_KERNEL);
    if (!scull_devices) {
        result = -ENOMEM;
        goto fail;      /* Make this more graceful */
    }
    memset(scull_devices, 0, scull_num_dev * sizeof(struct scull_dev));

    /* Initialize each device. */
    for (i = 0; i < scull_num_dev; i++) {
        scull_devices[i].quantum = scull_quantum;
        scull_devices[i].qset = scull_qset;
        sema_init(&scull_devices[i].sem, 1);
        scull_setup_cdev(&scull_devices[i], i);
    }

    return 0;          /* succeed */

fail:
    scull_cleanup_module();
    return result;
}

```

```

module_init(scull_init_module);
module_exit(scull_cleanup_module);

```

- scull driver에 대한 파일

2. 소스코드 컴파일

Make

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ sudo make
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/chardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/chardd/scull.o
  MODPOST /home/seongbinyoon/EdgeClab/chardd/Module.symvers
  CC [M]  /home/seongbinyoon/EdgeClab/chardd/scull.mod.o
  LD [M]  /home/seongbinyoon/EdgeClab/chardd/scull.ko
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ 

```

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ ls
Makefile modules.order Module.symvers scull.c scull.h scull.ko scull_load scull.mod scull.mod.c scull.mod.o scull.o scull_unload
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ 

```

- make 명령어를 통해 작성한 모듈 소스코드를 컴파일한다.

3. 모듈 적재

```
devtae@devtae:~/sources/scull$ sudo sh scull_load
```

```
$ sudo sh scull_load
```

- 관리자 권한을 통해 scull_load 스크립트를 실행
- scull_load 를 통해 이전 등록된 /dev 폴더에 있던 scull device 파일을 제거한 뒤, 새롭게 device driver 를 적재하고 등록한다.

```
devtae@devtae:~/sources/scull$ lsmod | grep scull
scull           16384  0
```

```
$ lsmod | grep scull
```

- scull 모듈이 성공적으로 적재가 되었는지에 대해 확인할 수 있음.

```
devtae@devtae:~$ ls -al /dev | grep scull
crw-rw-r--  1 root  staff  510,      0  1월 27 17:09 scull0
crw-rw-r--  1 root  staff  510,      1  1월 27 17:09 scull1
crw-rw-r--  1 root  staff  510,      2  1월 27 17:09 scull2
crw-rw-r--  1 root  staff  510,      3  1월 27 17:09 scull3
```

```
$ ls -al /dev | grep scull
```

- scull 디바이스가 성공적으로 등록되어 있는지에 대해 확인할 수 있음.
- ls -al 명령어를 통해 확인해보았을 때 가장 먼저 'c' 가 뜨는 것을 바탕으로 char device 등록에 성공했다는 것을 알 수 있음.

4. 모듈 제거

```
devtae@devtae:~/sources/scull$ sudo sh scull_unload
```

```
$ sudo sh scull_unload
```

- 위 명령어를 통해 scull 디바이스에 대한 모듈을 unload 하고 등록된 device 파일을 제거한다.

```
devtae@devtae:~/sources/scull$ lsmod | grep scull
devtae@devtae:~/sources/scull$ ls -al /dev | grep scull
devtae@devtae:~/sources/scull$ 
```

```
$ lsmod | grep scull
$ ls -al /dev | grep scull
```

- 위 명령어를 통해 이전에 등록했던 모듈과 디바이스 파일이 제거되었음을 확인할 수 있다.

+) Char Device 테스트 실습

root 권한을 획득한 뒤, echo 출력 결과를 scull0 디바이스에 전송시키고나서 출력값을 확인하는 과정을 직접 해보겠다.

```
$ sudo -s
```

- 위 명령어를 통하여 root 권한을 얻어낸다.

```
# echo "hello, world!" > /dev/scull0
# cat /dev/scull0
hello, world!
```

- 다음과 같이 echo 명령문을 활용하여 출력 결과를 scull0 디바이스에 입력하도록 하고 입력된 데이터를 cat 을 통해 확인해볼 수 있다.

```
root@devtae:/home/devtae# echo "hello, world!" > /dev/scull0
root@devtae:/home/devtae# cat /dev/scull0
hello, world!
```

- 실제 실행 결과는 다음과 같다. 입력을 받고 출력하는 부분이 정상적으로 진행됨을 확인할 수 있다.

Trouble Shooting

Trouble 1

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ ls
Makefile scull.c scull.h scull_load scull_unload
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ sudo make
[sudo] password for seongbinyoon:
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/chardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/chardd/scull.o
/home/seongbinyoon/EdgeClab/chardd/scull.c:43:18: error: initialization of 'ssize_t (*)(struct file *, const char *, size_t, loff_t *)' {aka 'long int (*)(struct file *, const char *, long unsigned int, long long int *)'} from incompatible pointer type 'ssize_t (*)(struct file *, char *, size_t, loff_t *)' {aka 'long int (*)(struct file *, char *, long unsigned int, long long int *)'} [-Werror=incompatible-pointer-types]
  43 |         .write = scull_write,
     |             ^
/home/seongbinyoon/EdgeClab/chardd/scull.c:43:18: note: (near initialization for 'scull_fops.write')
/home/seongbinyoon/EdgeClab/chardd/scull.c:44:10: error: 'struct file_operations' has no member named 'ioctl'
  44 |         .ioctl = scull_ioctl,
     |             ^
/home/seongbinyoon/EdgeClab/chardd/scull.c:44:18: error: 'scull_ioctl' undeclared here (not in a function)
  44 |         .ioctl = scull_ioctl,
     |             ^
/home/seongbinyoon/EdgeClab/chardd/scull.c:44:18: error: positional initialization of field in 'struct' declared with 'designated_init' attribute [-Werror=designated-init]
/home/seongbinyoon/EdgeClab/chardd/scull.c:44:18: note: (near initialization for 'scull_fops')
/home/seongbinyoon/EdgeClab/chardd/scull.c: In function 'scull_read':
/home/seongbinyoon/EdgeClab/chardd/scull.c:255:12: error: expected expression before '%' token
  255 |     up->dev->sem);
     |             ^
/home/seongbinyoon/EdgeClab/chardd/scull.c: At top level:
/home/seongbinyoon/EdgeClab/chardd/scull.c:261:9: error: conflicting types for 'scull_write'; have 'ssize_t(struct file *, const char *, size_t, loff_t *)' {aka 'long int(struct file *, const char *, long unsigned int, long long int *)'}
  261 |     ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos) {
     |             ^
In file included from /home/seongbinyoon/EdgeClab/chardd/scull.c:14:
/home/seongbinyoon/EdgeClab/chardd/scull.h:64:9: note: previous declaration of 'scull_write' with type 'ssize_t(struct file *, char *, size_t, loff_t *)' {aka 'long int(struct file *, char *, long unsigned int, long long int *)'}
  64 |     ssize_t scull_write(struct file *filp, char __user *buf, size_t count, loff_t *f_pos);
     |             ^
/home/seongbinyoon/EdgeClab/chardd/scull.c: In function 'scull_write':
/home/seongbinyoon/EdgeClab/chardd/scull.c:310:12: error: expected expression before '%' token
  310 |     up->dev->sem);
     |             ^
/home/seongbinyoon/EdgeClab/chardd/scull.c: In function 'scull_init_module':
/home/seongbinyoon/EdgeClab/chardd/scull.c:85:1: error: control reaches end of non-void function [-Werror{return-type}]
  85 | }
     | ^
At top level:
```

```
At top level:
/home/seongbinyoon/EdgeClab/chardd/scull.c:51:13: warning: 'scull_setup_cdev' defined but not used [-Wunused-function]
  51 | static void scull_setup_cdev(struct scull_dev *dev, int index) {
     |             ^
cc1: some warnings being treated as errors
make[2]: *** [scripts/Makefile.build:250: /home/seongbinyoon/EdgeClab/chardd/scull.o] Error 1
make[1]: *** [Makefile:1992: /home/seongbinyoon/EdgeClab/chardd] Error 2
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
make: *** [Makefile:1: default] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$
```

- char device driver 모듈을 만들고 컴파일을 하기 위해서 sudo make 명령어를 입력했을 때 나타난 오류 목록이다.

```
implicit declaration of function 'scull_cleanup_module'; did you mean 'scull_init_module'?
```

- 해당 에러가 발생하는데, 찾아보니 scull_init_module 함수 내에 호출된 scull_cleanup_module() 함수의 반환, 이름, 파라미터 타입이 맞지 않는 것 같다. 아무리 확인해봐도 void 타입의 파라미터를 가지고 void 타입의 리턴값을 반환하는 함수라 맞는 것 같은데 해결되지 않았다.

```
initialization of 'ssize_t (*)(struct file *, const char *, size_t, loff_t *)' {aka 'long int ()(struct file *, const char *, long unsigned int, long long int *)'} from incompatible pointer type 'ssize_t ()(struct file *, char *, size_t, loff_t *)' {aka 'long int ()(struct file *, char *, long unsigned int, long long int *)'}
```

```
conflicting types for 'scull_write'; have 'ssize_t(struct file *, const char *, size_t, loff_t *)' {aka 'long int(struct file *, const char *, long unsigned int, long long int *)'}
```

```
previous declaration of 'scull_write' with type 'ssize_t(struct file *, char *, size_t, loff_t *)' {aka 'long int(struct file *, char *, long unsigned int, long long int *)'}
```

- 해당 에러에서 볼 수 있듯이, scull_write 함수에 관한 초기화 문제 또는 타입 오류인 것 같다.

Solution 1

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/chardd$ sudo make
[sudo] password for seongbinyoon:
make[1]: Entering directory '/home/seongbinyoon/EdgeClab/chardd/modules'
makefile:1: /home/seongbinyoon/EdgeClab/chardd/scull.o
makefile:1: /home/seongbinyoon/EdgeClab/chardd/scull.h:43:18: error: initialization of 'ssize_t (*)(struct file *, const char *, size_t, loff_t *)' {aka 'long int (*)(struct file *, const char *, long unsigned int, long long int *)'} from incompatible pointer type 'ssize_t (*)(struct file *, char *, size_t, loff_t *)' {aka 'long int (*)(struct file *, char *, long unsigned int, long long int *)'} [-Werror=incompatible-pointer-types]
  43 |         .write = scull_write,
     |         ^
/home/seongbinyoon/EdgeClab/chardd/scull.c:43:18: note: (near initialization for 'scull_fops.write')
/home/seongbinyoon/EdgeClab/chardd/scull.h:64:9: error: conflicting types for 'scull_write'; have 'ssize_t(struct file *, const char *, size_t, loff_t *)' {aka 'long int(struct file *, const char *, long unsigned int, long long int *)'}
  64 |     ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos) {
     |     ^
In file included from /home/seongbinyoon/EdgeClab/chardd/scull.c:14:
/home/seongbinyoon/EdgeClab/chardd/scull.h:64:9: note: previous declaration of 'scull_write' with type 'ssize_t(struct file *, char *, size_t, loff_t *)' {aka 'long int(struct file *, char *, long unsigned int, long long int *)'}
  64 |     ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos);
     |     ^
/home/seongbinyoon/EdgeClab/chardd/scull.c: In function 'scull_init_module':
/home/seongbinyoon/EdgeClab/chardd/scull.c:84:1: error: control reaches end of non-void function [-Werror=return-type]
  84 |
At top level:
/home/seongbinyoon/EdgeClab/chardd/scull.c:50:13: warning: 'scull_setup_cdev' defined but not used [-Wunused-function]
  50 | static void scull_setup_cdev(struct scull_dev *dev, int index) {
     |     ^
cc1: some warnings being treated as errors
makefile:2: *** [scripts/Makefile.build:250: /home/seongbinyoon/EdgeClab/chardd/scull.o] Error 1
makefile:1: *** [Makefile:1992: /home/seongbinyoon/EdgeClab/chardd] Error 2
makefile:1: Leaving directory '/home/seongbinyoon/linux-6.1.2'
makefile: *** [Makefile:11: default] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/chardd$
```

- %연산자를 &연산자로 수정하고, file_operation structure에 .ioctl을 삭제한 후 sudo make 명령어를 입력했을 때 줄어든 오류의 모습이다. (일단 ioctl은 나중에 짜보기로 했다.)

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ sudo make
[sudo] password for seongbinyoon:
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/chardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/chardd/scull.o
/home/seongbinyoon/EdgeClab/chardd/scull.c:43:18: error: initialization of 'ssize_t (*)(struct file *, const char *, size_t, loff_t *)' (aka 'long int (*)(struct file *, const char *, long unsigned int, long long int *)') from incompatible pointer type 'ssize_t (*)(struct file *, char *, size_t, loff_t *)' (aka 'long int (*)(struct file *, char *, long unsigned int, long long int *)') [-Werror=incompatible-pointer-types]
  43 |     .write = scull_write,
|               ^
/home/seongbinyoon/EdgeClab/chardd/scull.c:43:18: note: (near initialization for 'scull_fops.write')
/home/seongbinyoon/EdgeClab/chardd/scull.c: In function 'scull_init_module':
/home/seongbinyoon/EdgeClab/chardd/scull.c:107:9: error: implicit declaration of function 'scull_cleanup_module'; did you mean 'scull_init_module'? [-Werror=implicit-function-declaration]
  107 |     scull_cleanup_module();
|           ^
|           scull_init_module
/home/seongbinyoon/EdgeClab/chardd/scull.c: At top level:
/home/seongbinyoon/EdgeClab/chardd/scull.c:113:6: warning: conflicting types for 'scull_cleanup_module'; have 'void(void)'
  113 | void scull_cleanup_module(void) {
|           ^
|           ~~~~~
/home/seongbinyoon/EdgeClab/chardd/scull.c:107:9: note: previous implicit declaration of 'scull_cleanup_module' with type 'void(void)'
  107 |     scull_cleanup_module();
|           ^
|           ~~~~~
/home/seongbinyoon/EdgeClab/chardd/scull.c:285:9: error: conflicting types for 'scull_write'; have 'ssize_t(struct file *, const char *, size_t, loff_t *)' (aka 'long int(struct file *, const char *, long unsigned int, long long int *)')
  285 | ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos) {
|           ^
|           ~~~~~
In file included from /home/seongbinyoon/EdgeClab/chardd/scull.c:14:
/home/seongbinyoon/EdgeClab/chardd/scull.h:64:9: note: previous declaration of 'scull_write' with type 'ssize_t(struct file *, char *, size_t, loff_t *)' (aka 'long int(struct file *, char *, long unsigned int, long long int *)')
  64 | ssize_t scull_write(struct file *filp, char __user *buf, size_t count, loff_t *f_pos);
|           ^
ccl: some warnings being treated as errors
make[2]: *** [scripts/Makefile.build:250: /home/seongbinyoon/EdgeClab/chardd/scull.o] Error 1
make[1]: *** [Makefile:1992: /home/seongbinyoon/EdgeClab/chardd] Error 2
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
make: *** [Makefile:1: default] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$
```

- 이 후, `scull_init_module()` 함수를 수정해주었다. 뒤에 작성하지 않은 부분이 있어 깃허브를 참조해 작성하였다.

<https://github.com/freestyl3r/kernel-drivers/tree/master/scull>

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ sudo make
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/chardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/chardd/scull.o
/home/seongbinyoon/EdgeClab/chardd/scull.c:43:18: error: initialization of 'ssize_t (*)(struct file *, const char *, size_t, loff_t *)' (aka 'long int (*)(struct file *, const char *, long unsigned int, long long int *)') from incompatible pointer type 'ssize_t (*)(struct file *, char *, size_t, loff_t *)' (aka 'long int (*)(struct file *, char *, long unsigned int, long long int *)') [-Werror=incompatible-pointer-types]
  43 |     .write = scull_write,
|               ^
/home/seongbinyoon/EdgeClab/chardd/scull.c:43:18: note: (near initialization for 'scull_fops.write')
/home/seongbinyoon/EdgeClab/chardd/scull.c:285:9: error: conflicting types for 'scull_write'; have 'ssize_t(struct file *, const char *, size_t, loff_t *)' (aka 'long int(struct file *, const char *, long unsigned int, long long int *)')
  285 | ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos) {
|           ^
|           ~~~~~
In file included from /home/seongbinyoon/EdgeClab/chardd/scull.c:14:
/home/seongbinyoon/EdgeClab/chardd/scull.h:64:9: note: previous declaration of 'scull_write' with type 'ssize_t(struct file *, char *, size_t, loff_t *)' (aka 'long int(struct file *, char *, long unsigned int, long long int *)')
  64 | ssize_t scull_write(struct file *filp, char __user *buf, size_t count, loff_t *f_pos);
|           ^
ccl: some warnings being treated as errors
make[2]: *** [scripts/Makefile.build:250: /home/seongbinyoon/EdgeClab/chardd/scull.o] Error 1
make[1]: *** [Makefile:1992: /home/seongbinyoon/EdgeClab/chardd] Error 2
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
make: *** [Makefile:1: default] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$
```

- `scull_init_module`에서 모호한 선언에 대한 오류가 뜨는 이유가 뭘까 생각하다가 알아낸 것이 있다. 객체지향 언어에 익숙한 나는 C언어가 절차형 언어임을 잊고 아직 정의하지 않은 `scull_cleanup_module()` 함수를 `init` 보다 뒤에 작성했고, 이 때문에 컴파일러가 인식하지 못해 생긴 문제였다. `init` 함수 위에 `cleanup` 함수를 먼저 정의해주었고, 오류가 해결되었다.
- 나머지 오류도 이렇게 해결하면 될 것 같다.

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/chardd$ sudo make
[sudo] password for seongbinyoon:
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/chardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/chardd/scull.o
/home/seongbinyoon/EdgeClab/chardd/scull.c:190:9: error: conflicting types for 'scull_write'; have 'ssize_t(struct file *, const char *, size_t, loff_t *)' {
  aka 'long int(struct file *, const char *, long unsigned int, long long int *)'
   190 | ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos) {
      |
In file included from /home/seongbinyoon/EdgeClab/chardd/scull.c:14:
/home/seongbinyoon/EdgeClab/chardd/scull.h:64:9: note: previous declaration of 'scull_write' with type 'ssize_t(struct file *, char *, size_t, loff_t *)' {aka 'long int(struct file *, char *, long unsigned int, long long int *)'}
   64 | ssize_t scull_write(struct file *filp, char __user *buf, size_t count, loff_t *f_pos);
      |
make[2]: *** [scripts/Makefile.build:250: /home/seongbinyoon/EdgeClab/chardd/scull.o] Error 1
make[1]: *** [Makefile:1992: /home/seongbinyoon/EdgeClab/chardd] Error 2
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
make: *** [Makefile:11: default] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/chardd$
```

- 절차형 언어인 C언어임을 고려해서 모듈을 작성할 때 코드의 마지막에,

file operation

setup

cleanup

init

순으로 작성해야 함을 배웠다.

- 코드 구조를 변경 후 위와 같이 에러가 하나로 줄었다.
- 순서를

trim

open

release

follow

read

write

llseek

file operation

setup

cleanup

init

순으로 작성했는데, 전과 동일했다. 다른 메소드들의 순서 문제는 아닌 것 같다.

```

/* Prototypes for shared functions */
int scull_open(struct inode *inode, struct file *filp);
int scull_release(struct inode *inode, struct file *filp);
int scull_trim(struct scull_dev *dev);
ssize_t scull_read(struct file *filp, char __user *buf, size_t count, loff_t *f_pos);
ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos);
loff_t scull_llseek(struct file *filp, loff_t off, int whence);

#endif
"scull.h" 67L, 1569B
```

63,42 Bot

- 문제는 scull.h 헤더파일의 scull_write 함수 정의 부분이었다. __user *buf 파라미터 타입이 char로 되어 있었는데, 이를 const char로 바꿔주니 해결되었다.

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ sudo make
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/chardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/chardd/scull.o
  MODPOST /home/seongbinyoon/EdgeClab/chardd/Module.symvers
  CC [M]  /home/seongbinyoon/EdgeClab/chardd/scull.mod.o
  LD [M]  /home/seongbinyoon/EdgeClab/chardd/scull.ko
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$
```

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$ ls
Makefile modules.order Module.symvers scull.c scull.h scull.ko scull_load scull.mod scull.mod.c scull.mod.o scull.o scull_unload
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA:~/EdgeClab/chardd$
```

- 다음과 같이 컴파일에 성공하였다.

Trouble 2

```
#!/bin/sh
module="scull"
device="scull"
mode="664"

# Invoke insmod with all arguments we got
# And use a pathname, as newer modutils don't look in . by default
/sbin/insmod ./${module}.ko $* || exit 1

# Remove stale nodes
rm -f /dev/${device}[0-3]

major=$(awk "\$2==\"$module\" {print \\\\$1}" /proc/devices)

mknod/dev/${device}0 c $major 0
mknod/dev/${device}1 c $major 1
mknod/dev/${device}2 c $major 2
mknod/dev/${device}3 c $major 3

# Give appropriate group/permissions, and change the group.
# Not all distributions have staff, some have "wheel" instead.
group="staff"
grep -q '^staff:' /etc/group || group="wheel"

chgrp $group /dev/${device}[0-3]
chmod $mode /dev/${device}[0-3]
```

- scull_load 파일의 소스코드인데, major 변수에 awk로 가져와 할당하는 부분에서 \proc\devices로 역슬래시가 되어있었다.
- mknod 부분에서 띄어쓰기가 되어있지 않았다.

Solution 2

```

#!/bin/sh
module="scull"
device="scull"
mode="664"

# Invoke insmod with all arguments we got
# And use a pathname, as newer modutils don't look in . by default
/sbin/insmod ./${module}.ko $* || exit 1

# Remove stale nodes
rm -f /dev/${device}[0-3]

major=$(awk "\$2==\"$module\" {print \$1}" /proc/devices)

mknod /dev/${device}0 c $major 0
mknod /dev/${device}1 c $major 1
mknod /dev/${device}2 c $major 2
mknod /dev/${device}3 c $major 3

# Give appropriate group/permissions, and change the group.
# Not all distributions have staff, some have "wheel" instead.
group="staff"
grep -q '^staff:' /etc/group || group="wheel"

chgrp $group /dev/${device}[0-3]
chmod $mode /dev/${device}[0-3]

```

- \를 /로 수정하였고, mknod 부분의 띄어쓰기 또한 수정 완료하였다.

Trouble 3

- 작성하고 컴파일한 모듈을 커널에 적재하는 과정에서, 갈피를 잡지 못했다.

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeCloud/chardev$ sudo insmod scull.ko
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeCloud/chardev$ lsmod
Module           Size  Used by
scull            16384  0
hello            16384  0
nf_tables        270336  0
nfnetlink         20480  1 nf_tables
btrfs            1613824  0
blake2b_generic   20480  0
xor              24576  1 btrfs
raid6_pq          122880  1 btrfs
zstd_compress     348160  1 btrfs
ufs              102400  0
qnx4             16384  0
hfsplus           118784  0
hfs              65536  0
minix            49152  0
ntfs             122880  0
msdos            20480  0
jfs              229376  0
xfs              1810432  0
libcrc32c         16384  3 btrfs,nf_tables,xfs
cpuid             16384  0
ccm              20480  6
rfcomm            81920  4
snd_ctl_led       24576  0
snd_soc_skl_hda_dsp    28672  6
snd_soc_intel_hda_dsp_common 20480  1 snd_soc_skl_hda_dsp
snd_soc_probes     20480  0
snd_soc_hdac_hdmi 36864  1 snd_soc_skl_hda_dsp
snd_hda_codec_hdmi 77824  1
snd_hda_codec_realtek 159744  1
snd_hda_codec_generic 102400  1 snd_hda_codec_realtek
cmac              16384  3
algif_hash         16384  1
algif_skcipher    16384  1
af_alg             32768  6 algif_hash,algif_skcipher
bnef              28672  2
snd_soc_dmic      16384  1
snd_soc_pci_intel_tgl 16384  0
snd_sof_intel_hda_common 172032  1 snd_soc_pci_intel_tgl
soundwire_intel    45056  1 snd_sof_intel_hda_common
soundwire_generic_allocation 16384  1 soundwire_intel
soundwire_cadence   40960  1 soundwire_intel
snd_sof_intel_hda    20480  1 snd_sof_intel_hda_common

```

- lsmod 명령어를 통해 확인하면, scull 모듈이 커널에 적재되기는 하지만, /dev 디렉토리에서 ls-al 명령어를 입력했을 때 char device driver로 보이지 않았다. (문자형으로 정상적으로 적재되지 않았다.)
- advanced char device driver에서는 scull.init 파일이 따로 있고, insmod 명령어가 아닌 스크립트 실행 명령어 sudo ./scull.init으로 커널에 적재한다.

[https://m.blog.naver.com/PostView.naver?
isHttpsRedirect=true&blogId=pajaebeo&logNo=102854820](https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=pajaebeo&logNo=102854820) (참조)

Solution 3

Make 작업을 한 뒤에 scull.ko 파일을 직접 insmod 를 해주는 것이 아닌, scull_load 스크립트를 sh 프로그램을 통해 실행해야 했다.

insmod 를 통해서는 개발한 device 에 대한 driver 모듈만을 적재하는 것에 불과하였다. 즉, scull_init_module 함수에서 MKDEV 를 통해 major number 까지 할당 받았지만 실질적으로 device 파일을 생성하지는 못하였다.

따라서, insmod 부터 mknod 명령어를 통해 device 파일까지 생성하는 scull_load 스크립트를 실행해주는 방식으로 문제를 해결할 수 있었다.

```
devtae@devtae:~/sources/scull$ sudo sh scull_load
```

```
$ sudo sh scull_load
```

이 방식을 통해 문제를 해결할 수 있었다.

```
devtae@devtae:~/sources/scull$ ls -al /dev | grep scull
crw-rw-r-- 1 root staff 510,      0  1월 27 17:55 scull0
crw-rw-r-- 1 root staff 510,      1  1월 27 17:55 scull1
crw-rw-r-- 1 root staff 510,      2  1월 27 17:55 scull2
crw-rw-r-- 1 root staff 510,      3  1월 27 17:55 scull3
```

그 결과, scull[0-3] 디바이스 파일이 'c'har device 로 정상 등록됐음을 확인할 수 있었다.

+) 추가적으로 awk 문에서 오류가 발생하여 scull_load 실행에 문제가 생겼었다.

그에 따라, awk 문에서 제대로 반환값을 찾을 수 없었고 mknod 를 호출할 때 major number 를 제대로 된 인수로 전달하지 못하여 디바이스 파일이 정상적으로 할당이 안 되는 상황이 발생하였다.

따라서 major 변수에 할당하는 코드를 다음과 같이 수정한 뒤에 문제를 해결할 수 있었다.

```
major=$(awk "\$2==\"$module\" {print \$1}" /proc/devices)
```

- scull 디바이스의 major number 를 구해오기 위한 구문이다.
- awk “ .. 조건문 .. { .. 실행문 .. }” ..대상 파일.. 의 문법으로 awk 를 호출함을 알 수 있다.
- 실제로 /proc/devices 파일은 첫 번째 열에 major number, 두 번째 열에 디바이스 이름이 들어간다. 따라서, 해당 구문은 2 번째 열(\$2)의 값이 \$module(=scull) 인 경우 1 번째 열인 major number를 반환한다.

- 스크립트 코드에서 \$(command) 는 command 자체를 터미널에서 실행하는 결과를 가져온다.
- 변수를 할당할 때에는 \$ 표시를 넣지 않는다.
- 변수를 사용할 때에는 \$ 표시를 넣는다.
- 쌍따옴표와 \$와 같은 특수기호를 사용할 때 역슬래시(\)를 앞에 붙여줌으로써 정상적으로 인식할 수 있도록 할 수 있다.

Linux KVM

Linux KVM?

- KVM(Kernel-based Virtual Machine, 커널 기반 가상 머신)은 Linux에 구축되는 오픈소스 가상화 기술
- 구체적으로, KVM을 사용하면 Linux를 하이퍼바이저로 전환하여 호스트 머신에서 게스트 또는 VM(가상 머신) 등 격리된 가상 환경 여러 개가 실행되도록 할 수 있다.
- Type 1(베어메탈 하이퍼바이저)이다.

Hypervisor?

- 하이퍼바이저는 가상 머신(Virtual Machine, VM)을 생성하고 구동하는 소프트웨어
- 가상 머신 모니터(Virtual Machine Monitor, VMM)라고도 불리는 하이퍼바이저는 하이퍼바이저 운영 체제와 가상 머신의 리소스를 분리해 VM의 생성과 관리를 지원
- 하이퍼바이저로 사용되는 물리 하드웨어를 호스트(host)라고 하며, 리소스를 사용하는 여러 VM을 게스트 (guest)라고 한다.
- 모든 하이퍼바이저에서 VM을 실행하려면 메모리 관리 프로그램, 프로세스 스케줄러, I/O(입력/출력) 스택, 기기 드라이버, 보안 관리 프로그램, 네트워크 스택과 같은 운영 체제 수준의 구성 요소가 필요

- 하이퍼바이저는 할당되었던 리소스를 각 가상 머신에 제공하고, 물리 리소스에 대해 VM 리소스의 일정을 관리
- 물리적 하드웨어는 계속해서 실행 작업을 수행하므로 하이퍼바이저가 일정을 관리하는 동안 CPU가 VM에서 요청한 대로 CPU 명령을 계속 실행
- 서로 다른 여러 개의 운영 체제를 나란히 구동할 수 있으며, 하이퍼바이저를 사용해 동일한 가상화 하드웨어 리소스를 공유(가상화가 없다면 하드웨어에서 운영 체제를 1개만 구동할 수 있는 것)
- 2가지 유형이 있다.
 - Type 1: Native/Bare metal hypervisor
 - 호스트의 하드웨어에서 직접 구동, 게스트 운영 체제를 관리
 - ex) KVM
 - Type 2: Host hypervisor
 - 기존 운영 체제에서 소프트웨어 레이어 또는 애플리케이션으로서 구동, 호스트 운영 체제에서 게스트 운영 체제를 추상화하는 방식
 - ex) Oracle VirtualBox

Linux KVM Installation

Linux(Ubuntu 22.04 LTS) 환경에서 KVM 하이퍼바이저를 설치하는 방법을 담고 있다.

개인 PC에서 연구실의 서버용 PC로 ssh접속을 사용하여 설치를 진행하였고, ssh접속에 아울러 학교의 허가를 받아 anyconnect vpn을 통해 연구실 PC의 IP주소로 연결하였다.

1. ssh 접속

```
~$ ssh name@IPaddress
```

- 서버 PC로 ssh 접속한다.

2. 가상화 지원 확인

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ egrep -c '(vmx|svm)' /proc/cpuinfo  
24  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ █
```

```
~$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

- KVM 설치 전, 하드웨어가 KVM을 지원하는지 확인할 필요가 있다.
- CPU에서 하드웨어 가상화를 지원하는지 확인한다. (인텔 CPU: vmx 플래그, AMD CPU: svm플래그)
- CPU 코어 수인 0보다 큰 수가 출력되면 가상화를 지원하는 것이므로 다음 단계로 넘어간다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo kvm-ok  
sudo: kvm-ok: command not found  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo apt install cpu-checker  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  libflashrom1 libftdi1-2  
Use 'sudo apt autoremove' to remove them.  
The following additional packages will be installed:  
  msr-tools  
The following NEW packages will be installed:  
  cpu-checker msr-tools  
0 upgraded, 2 newly installed, 0 to remove and 5 not upgraded.  
Need to get 17.1 kB of archives.  
After this operation, 67.6 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://kr.archive.ubuntu.com/ubuntu jammy/main amd64 msr-tools amd64 1.3-4 [10.3 kB]  
Get:2 http://kr.archive.ubuntu.com/ubuntu jammy/main amd64 cpu-checker amd64 0.7-1.3build1 [6,800 B]  
Fetched 17.1 kB in 1s (23.7 kB/s)  
Selecting previously unselected package msr-tools.  
(Reading database ... 205881 files and directories currently installed.)  
Preparing to unpack .../msr-tools_1.3-4_amd64.deb ...  
Unpacking msr-tools (1.3-4) ...  
Selecting previously unselected package cpu-checker.  
Preparing to unpack .../cpu-checker_0.7-1.3build1_amd64.deb ...  
Unpacking cpu-checker (0.7-1.3build1) ...  
Setting up msr-tools (1.3-4) ...  
Setting up cpu-checker (0.7-1.3build1) ...  
Processing triggers for man-db (2.10.2-1) ...  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

```
~$ sudo kvm-ok
```

- 일부 시스템에서는 제조업체에 의해 BIOS에서 가상 기술 확장이 사용되지 않도록 설정될 수 있어, BIOS에서 VT가 설정되어 있는지 확인하려면 CPU 체커 패키지에 포함된 kvm-ok 도구를 사용한다.
- 위 명령어를 입력해 kvm-ok 유틸리티가 존재하는지 확인한다.

```
~$ sudo apt install cpu-checker
```

- kvm-ok 유틸리티가 설치되어 있지 않으면 위 명령어를 입력해 CPU 체커 패키지를 설치한다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

```
~$ sudo kvm-ok
```

- 설치 후 다시 명령어를 입력해보면 위와 같이 존재함을 확인할 수 있고, 다음 단계로 넘어간다.

3. KVM 설치

- KVM을 설치하는데 필요한 각 패키지는 다음과 같다. (종속적 패키지 포함)
 - qemu: 하드웨어 가상화를 수행할 수 있도록 해주는 애플리케이션
 - qemu-kvm: KVM 하이퍼바이저에 대한 하드웨어 에뮬레이션을 제공
 - qemu-system: KVM 에뮬레이터를 제공하는 필수 패키지
 - libvirt-daemon-system: libvirt 데몬을 시스템 서비스로 실행하는 구성 파일
 - libvirt: 하이퍼바이저를 관리하는 가상화 장치 필수 패키지
 - libvirt-clients: 가상화 플랫폼을 관리하기 위한 소프트웨어
 - libvirt-bin: 가상화 플랫폼을 관리하기 위한 소프트웨어
 - bridge-utils: 이더넷 브릿지를 구성하기 위한 명령 줄 도구 세트
 - virtinst: 가상 머신을 만들기 위한 명령 줄 도구 집합
 - virt-manager: 사용하기 쉬운 GUI 인터페이스와 libvirt를 통해 가상 머신을 관리하기 위한 명령 줄 유틸리티
 - ubuntu-vm-builder

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo apt-get update
Hit:1 https://dl.google.com/linux/chrome/deb stable InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:3 http://kr.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://kr.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:5 http://kr.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:6 http://kr.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [834 kB]
Get:7 http://kr.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [413 kB]
Get:8 http://kr.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [790 kB]
Get:9 http://kr.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [564 kB]
Fetched 2,925 kB in 5s (608 kB/s)
Reading package lists... Done
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo apt install qemu qemu-kvm qemu-system libvirt-daemon libvirt-daemon-system libvirt libvirt-clients libvirt-bin bridge-utils virtinst virt-manager ubuntu-vm-builder
```

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo apt install qemu qemu-kvm qemu-system libvirt-daemon libvirt-daemon-system libvirt libvirt-clients libvirt-bin bridge-utils virtinst virt-manager ubuntu-vm-builder
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'qemu-system-x86' instead of 'qemu-kvm'
Package libvirt-bin is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Unable to locate package libvirt
E: Package 'libvirt-bin' has no installation candidate
E: Unable to locate package ubuntu-vm-builder
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

- 위 패키지를 모두 설치하려고 했으나 호환성의 문제로 오류가 발생했다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo apt install -y qemu qemu-kvm libvirt-daemon libvirt-clients bridge-utils virt-manager
```

```
Setting up libspice-client-gtk-3.0-5:amd64 (0.39-3ubuntu1) ...
Setting up libgbfafp0:amd64 (10.1-1) ...
Setting up gir1.2-spiceclientgtk-3.0:amd64 (0.39-3ubuntu1) ...
Setting up virt-viewer (7.0-2build2) ...
Setting up libvirt-daemon-system (8.0.0-1ubuntu7.4) ...
Adding user libvirt-qemu to group libvirt-qemu
Enabling libvirt default network
Created symlink /etc/systemd/system/multi-user.target.wants/libvirtd.service → /lib/systemd/system/libvirtd.service.
Created symlink /etc/systemd/system/sockets.target.wants/virtlockd.socket → /lib/systemd/system/virtlockd.socket.
Created symlink /etc/systemd/system/sockets.target.wants/virtlogd.socket → /lib/systemd/system/virtlogd.socket.
Created symlink /etc/systemd/system/sockets.target.wants/libvirtd-ro.socket → /lib/systemd/system/libvirtd-ro.socket.
Created symlink /etc/systemd/system/multi-user.target.wants/libvirt-guests.service → /lib/systemd/system/libvirt-guests.service.
virtlogd.service is a disabled or a static unit, not starting it.
virtlogd.service is a disabled or a static unit, not starting it.
Created symlink /etc/systemd/system/sockets.target.wants/libvirtd-admin.socket → /lib/systemd/system/libvirtd-admin.socket.
Created symlink /etc/systemd/system/sockets.target.wants/virtlockd-admin.socket → /lib/systemd/system/virtlockd-admin.socket.
Created symlink /etc/systemd/system/sockets.target.wants/virtlogd-admin.socket → /lib/systemd/system/virtlogd-admin.socket.
Setting up libvirt-daemon dnsmasq configuration.
Setting up qemu-block-extra (1:6.2+dfsg-2ubuntu6.6) ...
Created symlink /etc/systemd/system/multi-user.target.wants/run-qemu.mount → /lib/systemd/system/run-qemu.mount.
Setting up liblvm2cmd2.03:amd64 (2.03.11-2.1ubuntu4) ...
Setting up dmeventd (2:1.02.175-2.1ubuntu4) ...
Created symlink /etc/systemd/system/sockets.target.wants/dm-event.socket → /lib/systemd/system/dm-event.socket.
dm-event.service is a disabled or a static unit, not starting it.
Setting up lvm2 (2.03.11-2.1ubuntu4) ...
update-intramfs: deferring update (trigger activated)
Created symlink /etc/systemd/system/sysinit.target.wants/blk-availability.service → /lib/systemd/system/blk-availability.service.
Created symlink /etc/systemd/system/sysinit.target.wants/lvm2-monitor.service → /lib/systemd/system/lvm2-monitor.service.
Created symlink /etc/systemd/system/sysinit.target.wants/lvm2-lvmpoold.socket → /lib/systemd/system/lvm2-lvmpoold.socket.
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Processing triggers for shared-mime-info (2.1-2) ...
Processing triggers for install-info (6.8-4build1) ...
Processing triggers for mailcap (3.70+nmu1ubuntu1) ...
Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...
Processing triggers for initramfs-tools (0.140ubuntu13.1) ...
update-initramfs: Generating /boot/initrd.img-5.15.0-58-generic
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu3) ...
Processing triggers for libglib2.0-0:amd64 (2.72.4-0ubuntu1) ...
Processing triggers for libbc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
```

```
~$ sudo apt-get update
```

```
~$ sudo apt install -y qemu qemu-kvm libvirt-daemon libvirt-clients bridge-utils virt-manager
```

- 따라서, 이 중 qemu, qemu-kvm, libvirt-daemon, libvirt-clients, bridge-utils, virt-manager 만 설치하고 진행하였다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ kvm --version
QEMU emulator version 6.2.0 (Debian 1:6.2+dfsg-2ubuntu6.6)
Copyright (c) 2003-2021 Fabrice Bellard and the QEMU Project developers
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

```
~$ kvm --version
```

- KVM이 잘 설치되었는지 버전을 통해 확인한다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo systemctl status libvirdt
[sudo] password for ecl3:
● libvirdt.service - Virtualization daemon
  Loaded: loaded (/lib/systemd/system/libvirdt.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2023-01-25 18:13:07 KST; 19min ago
TriggeredBy: ● libvirdt-ro.socket
              ● libvirdt.socket
              ● libvirdt-admin.socket
    Docs: man:libvirdt(8)
          https://libvirt.org
  Main PID: 18214 (libvirdt)
    Tasks: 21 (limit: 32768)
   Memory: 10.0M
      CPU: 175ms
     CGroup: /system.slice/libvirdt.service
             ├─18214 /usr/sbin/libvirdt
             ├─18364 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr/lib/libvirt/libvirt_leaseshelper
             └─18367 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr/lib/libvirt/libvirt_leaseshelper

1월 25 18:13:07 ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC systemd[1]: Started Virtualization daemon.
1월 25 18:13:08 ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC dnsmasq[18364]: started, version 2.86 cachesize 150
1월 25 18:13:08 ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC dnsmasq[18364]: compile time options: IPv6 GNU-getopt DBus no-UBus l18n IDN2 DHCP DHCPv6 no-Lua TFTP
1월 25 18:13:08 ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC dnsmasq-dhcp[18364]: DHCP, IP range 192.168.122.2 -- 192.168.122.254, lease time 1h
1월 25 18:13:08 ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC dnsmasq-dhcp[18364]: DHCP, sockets bound exclusively to interface virbr0
1월 25 18:13:08 ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC dnsmasq[18364]: reading /etc/resolv.conf
1월 25 18:13:08 ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC dnsmasq[18364]: using nameserver 127.0.0.53#53
1월 25 18:13:08 ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC dnsmasq[18364]: read /etc/hosts - 7 addresses
1월 25 18:13:08 ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC dnsmasq[18364]: read /var/lib/libvirt/dnsmasq/default.addnhosts - 0 addresses
```

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo systemctl is-active libvirdt
[sudo] password for ecl3:
active
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ virsh
Welcome to virsh, the virtualization interactive terminal.

Type: 'help' for help with commands
      'quit' to quit

virsh #
virsh #
virsh # quit
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

```
~$ sudo systemctl status libvirdt
```

또는

```
~$ sudo systemctl is-active libvirdt
```

- 위 명령어를 입력해 가상화 데몬인 libvirdt - daemon이 실행 중인지 확인한다.

```
~$ sudo systemctl start libvirdt
```

```
~$ sudo systemctl stop libvirtd
```

- libvirtd - daemon이 inactive라면 위 명령어를 통해 start/stop 할 수 있다.

```
~$ sudo systemctl enable --now libvirtd
```

- 위 명령어를 입력하면 부팅 시 시작할 수 있다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ lsmod | grep -i kvm
kvm_intel           368640  0
kvm                1028096  1 kvm_intel
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

```
~$ lsmod | grep -i kvm
```

- 위 명령어를 입력해 KVM 모듈이 로드되었는지 확인한다.
- 이로써 KVM이 성공적으로 설치되었음을 알 수 있다.

4. 사용자 계정 그룹에 추가

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo usermod -aG libvirt $USER
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo usermod -aG kvm $USER
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

```
~$ sudo usermod -aG libvirt $USER
```

```
~$ sudo usermod -aG kvm $USER
```

- 가상머신을 생성하고 관리하려면 사용자 계정을 “libvirt” 및 “kvm” 그룹에 추가해야 한다.
- \$USER는 현재 로그인한 사용자의 이름을 포함하는 환경 변수이다.
- 이 후, 로그 아웃하고 다시 로그인한다.

5. x11 포워딩

- 서버 측 PC에서 GUI 환경으로 실행된 프로그램의 경우 일반 옵션의 ssh로 접속할 때 GUI 화면을 불러올 수 없다.
- virt-manager는 서버 측 PC에 설치되어 있으므로 ssh로 원격접속 시, 위 명령어로는 열 수 없고, 원격의 x윈도우를 포워딩할 수 있도록(x11 forwarding) 설정 후에 실행해야 한다.
- 이 부분에서, 윈도우 기반 환경에서 서버 측 리눅스로 접근하는 프로그램은 'xming xserver'라는 오픈소스 프로그램을 이용하면 쉽지만, 리눅스 기반에서 서버 측 리눅스로 원격으로 GUI를 볼 수 있는 방법은 조금 다르다.

클라이언트 측 PC 설정

```
~$ sudo apt-get update
```

```
~$ sudo apt install openssh-server
```

- 클라이언트 측, 서버 측 모두 위 명령어로 openssh-server를 설치한다.

서버 측 PC 설정

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $ sudo vim /etc/ssh/sshd_config
```

```
~$ sudo vim /etc/ssh/sshd_config
```

- 위 명령어로 sshd_config 파일에 들어가 수정을 해야 한다.

```
# Example of overriding settings on a per-user basis
#Match User anoncvs
#    X11Forwarding no
#    AllowTcpForwarding no
#    PermitTTY no
#    ForceCommand cvs server
```

```
# Example of overriding settings on a per-user basis
#Match User anoncvs
#    X11Forwarding yes
#    AllowTcpForwarding no
#    PermitTTY no
#    ForceCommand cvs server
"/etc/ssh/sshd_config" 122L, 3255B written
```

- 위와 같이 X11Forwarding no 부분을 yes로 수정한다. 주석은 제거한다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $ sudo vim /etc/ssh/sshd_config
```

```
~$ sudo vim /etc/ssh/sshd_config
```

- 위 명령어로 ssh_config 파일에 들어가 수정을 해야 한다. 주석은 제거한다.

```
Host *
# ForwardAgent no
# ForwardX11 no
# ForwardX11Trusted yes
```

```
Host *
# ForwardAgent no
# ForwardX11 yes
# ForwardX11Trusted yes
# PasswordAuthentication yes
"/etc/ssh/sshd_config" 122L, 3255B written
```

- 위와 같이 ForwardX11과 ForwardX11Trusted 부분을 yes로 수정해준다. 주석은 제거한다.
- 이 후 exit로 로그아웃한다.

```
~$ ssh -X <username>@<ipaddress>
```

- 다시 ssh 접속을 할 때, -X를 붙여 명령어를 실행한다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $ sudo virt-manager  
[sudo] password for ecl3:  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $ X11 connection rejected because of wrong authentication.  
X11 connection rejected because of wrong authentication.  
X11 connection rejected because of wrong authentication.
```

```
~$ sudo virt-manager
```

- 환경마다 다른데, GUI를 실행시켰을 때 바로 실행이 되는 경우가 있고, 위처럼 authentication 경고가 뜨며 실행이 되지 않는 경우가 있다.
- 진행하며 권한 오류가 나타났다.
- 해당 권한 오류를 해결하기 위해 몇가지 단계를 진행한다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $ sudo apt install xauth  
[sudo] password for ecl3:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
xauth is already the newest version (1:1.1-1build2).  
xauth set to manually installed.  
The following packages were automatically installed and are no longer required:  
  libflashrom1 libltdl2  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 27 not upgraded.  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $
```

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $ touch ~/.Xauthority  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $ sudo virt-manager  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $ X11 connection rejected because of wrong authentication.  
X11 connection rejected because of wrong authentication.  
X11 connection rejected because of wrong authentication.
```

```
~$ sudo apt install xauth
```

- 이미 설치되어 있는 것으로 나타나지만, 혹시 모르니 xauth 패키지를 설치하자.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ xhost  
access control enabled, only authorized clients can connect  
SI:localuser:seongbinyoon  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo xhost  
X11 connection rejected because of wrong authentication.  
xhost: unable to open display "localhost:10.0"
```

```
~$ xhost
```

```
~$ sudo xhost
```

- xhost 명령어를 통해 연결 및 권한 확인을 한다.
- 사용자 계정은 enabled지만, root 권한은 rejected된 것을 볼 수 있다. 이 권한을 허용해주기 위해 다음 작업을 수행한다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ echo $DISPLAY  
localhost:10.0  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ xauth list  
ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC/unix:11  MIT-MAGIC-COOKIE-1  35ad68a8ddbe18b142d0653683e9288d  
ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC/unix:10  MIT-MAGIC-COOKIE-1  310887fed9277f9c9ccedc060599f0c2
```

```
~$ echo $DISPLAY
```

```
~$ xauth list
```

- \$DISPLAY의 값을 기억하고, 위 명령어로 \$DISPLAY 값에 맞는 magic cookie value를 복사한다. 이 경우 unix: 10에 해당하는 줄이다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo su -  
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# vim .Xauthority  
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# ls -al  
total 48  
drwx----- 5 root root 4096 1월 30 12:48 .  
drwxr-xr-x 20 root root 4096 1월 16 14:56 ..  
-rw----- 1 root root 87 1월 30 12:09 .bash_history  
-rw-r--r-- 1 root root 3106 10월 15 2021 .bashrc  
drwx----- 3 root root 4096 1월 25 21:41 .cache  
-rw----- 1 root root 28 1월 25 21:27 .lesshst  
drwxr-xr-x 3 root root 4096 1월 25 23:54 .local  
-rw-r--r-- 1 root root 161 7월 9 2019 .profile  
drwx----- 5 root root 4096 1월 16 14:59 snap  
-rw-r--r-- 1 root root 0 1월 30 11:51 .sudo_as_admin_successful  
-rw----- 1 root root 9288 1월 30 12:48 .viminfo
```

```
~$ sudo su
```

- root 계정으로 접속한다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# ls -al
total 52
drwx----- 5 root root 4096 1월 30 12:51 .
drwxr-xr-x 20 root root 4096 1월 16 14:56 ..
-rw----- 1 root root 87 1월 30 12:09 .bash_history
-rw-r--r-- 1 root root 3106 10월 15 2021 .bashrc
drwx----- 3 root root 4096 1월 25 21:41 .cache
-rw----- 1 root root 20 1월 25 21:27 .lessht
drwxr-xr-x 3 root root 4096 1월 26 23:54 .local
-rw-r--r-- 1 root root 161 7월 9 2019 .profile
drwx----- 5 root root 4096 1월 16 14:59 snap
-rw-r--r-- 1 root root 0 1월 30 11:51 .sudo_as_admin_successful
-rw----- 1 root root 9288 1월 30 12:48 .viminfo
-rw----- 1 root root 85 1월 30 12:52 .Xauthority
```

```
~# ls -al
```

- 위 명령어를 통해 .Xauthority 파일의 존재를 확인할 수 있다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:/home/ecl3# mv .Xauthority .Xauthority.old
```

```
~$ mv .Xauthority .Xauthority.old
```

- 혹시 모를 상황에 대비해 Xauthority 파일을 old로 백업한다.

```
~$ touch ~/.Xauthority
```

- Xauthority라는 파일을 생성한다.

x11 Forwarding의 한가지 단점은 매 세션마다 magic cookie value가 변경되어 권한 설정을 수동으로 해줘야 한다는 점이다. 따라서 아래 단계부터는 매 세션(로그아웃 후 재로그인 시)마다 진행해야 한다.

여기부터

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# xauth add ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC/unix:10 MIT-MAGIC-COOKIE-1 310887fed9277f9c9ccedc060599f0c  
2  
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# vim .Xauthority
```

```
~# xauth add <Magic cookie value>
```

- 위에서 복사한 Magic cookie value를 붙여넣고 xauth에 권한을 추가한다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# cat ~/.Xauthority  
|ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC10MIT-MAGIC-COOKIE-1***utg0***root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# export DISPLAY=localhost:10.0
```

```
~# cat ~/.Xauthority  
cp <Magic cookie value> ~# export DISPLAY=localhost:10.0
```

- 제대로 추가되었는지 확인한다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# xhost  
access control enabled, only authorized clients can connect  
SI:localuser:seongbinyoon
```

```
~# xhost
```

- 위 명령어로 접근 권한이 추가되었는지 확인한다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# virt-manager  
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# (virt-manager:4085): dbind-WARNING **: 15:29:45.166: Couldn't register with accessibility bus: Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.
```

```

Virtual Machine Manager (on ecl3-HP-Pro-Tower-280-G9-PCI-...)
File Edit View Help
Console Open CPU usage
Name QEMU/KVM - Connecting...
'ec1 acc SI: ecl X11 xhc rocl Co sna rocl exi ecl loc loc ecl ecl loc rocl 3
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# cat /.Xauthority
Xauthority
*****root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:/home/ecl3# quit
dd ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC/unix:10 MIT-MAGIC-COOKIE-1 310887fed9277f9c9ccedc
Xauthority
*****root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:/home/ecl3# export DISPLAY=localhost:10.0
E-1 19bcf955d4c6ad1667300fb01ede0b3
P-Pro-Tower-280-G9-PCI-Desktop-PC/unix:10 MIT-MAGIC-COOKIE-1 19bcf955d4c6ad1667300fb01ede0b
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# xhost
access control enabled, only authorized clients can connect
SI:localuser:seongbinyoon
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# virt-manager
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~#
(virt-manager:4085): dbind-WARNING **: 13:29:45.168: Couldn't register with accessibility bus: Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.
virt-manager
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~#
(virt-manager:4186): dbind-WARNING **: 13:31:57.024: Couldn't register with accessibility bus: Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.

```

```
~# virt-manager
```

- virt-manager를 실행시키면 위와 같이 클라이언트 PC에서 서버 PC의 GUI를 볼 수 있다.

```

root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# exit
logout
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ xhost
access control enabled, only authorized clients can connect
SI:localuser:seongbinyoon
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo xhost
access control enabled, only authorized clients can connect

```

The screenshot shows a Linux desktop environment. On the left, the 'Virtual Machine Manager' window is open, displaying a list of VMs with their names and CPU usage. On the right, a terminal window is running a command to export the DISPLAY variable to the local host. The terminal output shows several warning messages about failed accessibility bus registrations.

```
Virtual Machine Manager (on ecl3-HP-Pro-Tower-280-G9-PCI-... - X) | X11 File Edit View Help  
xhc  
ecl  
loc  
ecl Name CPU usage  
loc QEMU/KVM  
ecl  
ecl  
roc  
3  
roc  
'ec  
roc  
acc  
SI:  
roc  
roc  
(vi  
pli  
vtr  
roc  
(vi  
pli  
qui  
Cor  
sna  
roc  
acc  
SI:  
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# virt-manager  
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# (virt-manager:4223): dbind: WARNING **: 15:32:58.640: Couldn't register with accessibility bus: Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.  
quit  
Command 'quit' not found, but can be installed with:  
snap install quit  
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# quit  
Command 'quit' not found, but can be installed with:  
snap install quit  
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# exit  
logout  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo virt-manager  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

```
~# exit
```

- exit으로 사용자 계정으로 돌아간다.

```
~$ xhost
```

```
~$ sudo xhost
```

- 위 명령어로 사용자 및 root 계정이 모두 접근 허용인지 확인한다.

```
~$ sudo virt-manager
```

- virt-manager를 실행시키면 마찬가지로 클라이언트 PC에서 서버 PC의 GUI를 볼 수 있다.

여기까지

- 이 후 KVM 하이퍼바이저에서의 VM(가상머신) 설치에 관한 내용은  [VM Installation](#) 문서를 참조하자.

위 과정을 스크립트 하나로 진행

- 매번 X.11 인증 작업을 해줄 필요가 있는 상황에 위 과정을 단순화하는 스크립트가 필요했다.
- Bash Shell Script 를 통하여 제작해보았다.
- 아래 파일을 원하는 디렉토리에 작성하고 root 계정이 아닌 일반 계정으로 실행하면 된다.
- 한 번이라도 위 설정을 정상적으로 진행 완료한 이후에 사용 가능하다. (DISPLAY 환경변수까지 설정하진 않는다.)

<get-xwindow.sh>

```

#!/bin/bash
# Set the xauth cache config of root account to get a user access of the x-window
# Developed by DevTae
#
# <How to Use>
# 1. Login to local user not a root account
# 2. Open the terminal and type 'bash get-xwindow.sh'
# 3. Execute the gui program as like 'sudo virt-manager'

display_split=($(echo $DISPLAY | tr ":" "\n"))
localhost=${display_split[0]}    # localhost
dis_num=${display_split[1]}      # display number
cache=($(echo $(xauth list)))   # get user xauth cache
index=-1

echo "The display number of user is $dis_num"

for ((i=0; ; i++));
do
    cache_split=($(echo ${cache[i*3]} | tr ":" "\n"))
    if [ "${cache_split[1]}" = "${dis_num}" ]
    then
        index=$i
        break
    elif [ "${cache_split[0]}" = "" ]
    then
        echo "Failed to find a user's xauth cache. Log-out the super-user and retry."
        exit 1
    else
        continue
    fi
done

loaded_cache="${cache[index*3]} ${cache[index*3+1]} ${cache[index*3+2]}"
if [ $index -ne -1 ]
then
    echo "Success to find your user's cache :"
    echo "$loaded_cache"
    echo
else
    echo "Fail to find your user's cache"
    exit 1
fi

sudo -s xauth add $loaded_cache
echo "The process to add the user's cache to root account is completed."

```

```
echo "Below this message, there are contents of the xauth cache file of root."
sudo -s xauth list
echo

echo "This is the end of the prcess to add the user's xauth cache."
echo "Type command like this : sudo virt-manager"
```

- 그 이후 다음과 같이 터미널에서 실행해주면 된다.

```
$ bash get-xwindow.sh
```

<실행 이전>

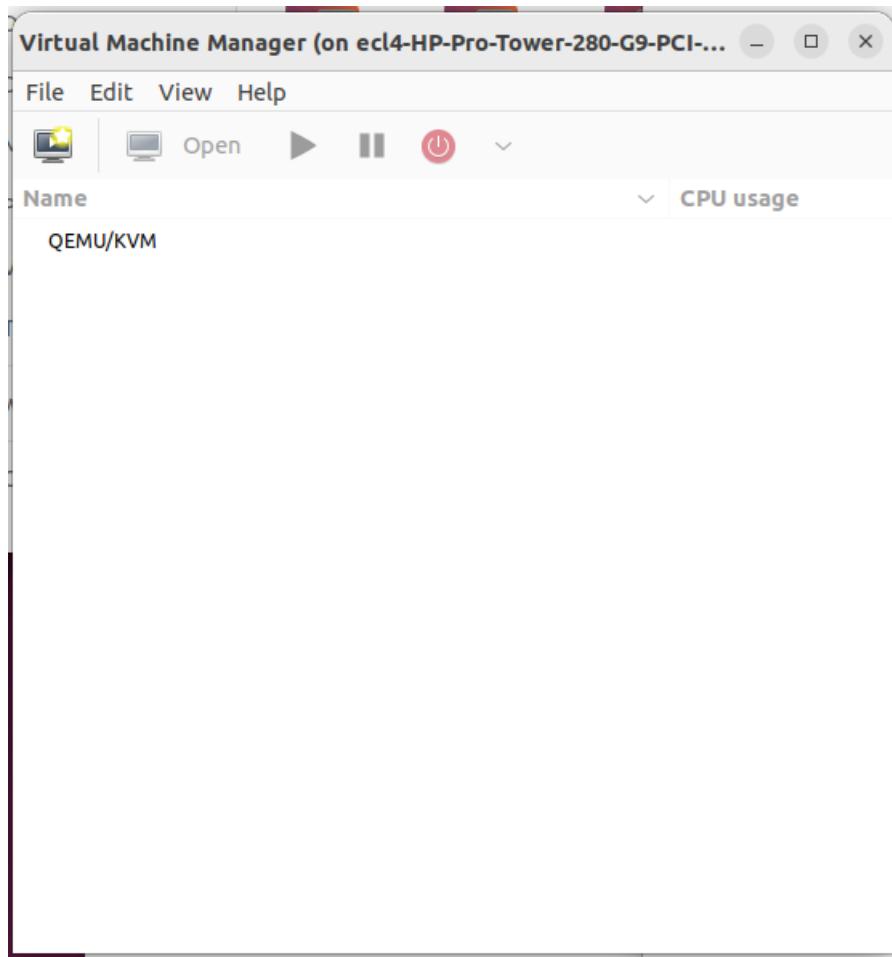
```
ecl4@ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo virt-manager
ecl4@ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ X11 connection rejected because of wrong authentication.
X11 connection rejected because of wrong authentication.
X11 connection rejected because of wrong authentication.
^C
ecl4@ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo -s
root@ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC:/home/ecl4# xauth list
root@ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC:/home/ecl4# exit
```

<실행 후>

```
ecl4@ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ bash get-xwindow.sh
The display number of user is 10
Success to find your user's cache :
ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC/unix:10  MIT-MAGIC-COOKIE-1  c8c5f8ed5
f9be50ee26d5e8b624c9361

The process to add the user's cache to root account is completed.
Below this message, there are contents of the xauth cache file of root.
ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC/unix:10  MIT-MAGIC-COOKIE-1  c8c5f8ed5
f9be50ee26d5e8b624c9361

This is the end of the prcess to add the user's xauth cache.
Type command like this : sudo virt-manager
ecl4@ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo virt-manager
ecl4@ecl4-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ █
```



Trouble Shooting

Trouble 1

- X11 Forwarding 중 권한 오류가 발생했다.
- 포워딩 후 GUI 프로그램을 실행시켰을 때, 아래와 같은 오류가 발생한다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo virt-manager
[sudo] password for ecl3:
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$ X11 connection rejected because of wrong authentication.
X11 connection rejected because of wrong authentication.
X11 connection rejected because of wrong authentication.
```

```
X11 connection rejected because of wrong authentication.
```

- 환경마다 다른데, GUI를 실행시켰을 때 바로 실행이 되는 경우가 있고, 위처럼 authentication 경고가 뜨며 실행이 되지 않는 경우가 있다.
- 해당 권한 오류를 해결하기 위해 몇가지 단계를 진행했다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$ sudo apt install xauth  
[sudo] password for ecl3:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
xauth is already the newest version (1:1.1-1build2).  
xauth set to manually installed.  
The following packages were automatically installed and are no longer required:  
  libflashrom1 libftdi1-2  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 27 not upgraded.  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$
```

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$ touch ~/.Xauthority  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$ sudo virt-manager  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$ X11 connection rejected because of wrong authentication.  
X11 connection rejected because of wrong authentication.  
X11 connection rejected because of wrong authentication.
```

```
~$ sudo apt install xauth
```

- 이미 설치되어 있는 것으로 나타나지만, 혹시 모르니 xauth 패키지를 설치하였다.

```
~$ touch ~/.Xauthority
```

- Xauthority라는 파일을 생성하였다.
- 이 후에도 오류가 계속되었다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$ xauth merge /home/ecl3/.Xauthority
```

- 위 방법도 해보았지만 여전히 오류가 발생하였다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$ chown ecl3:ecl3 ~/.Xauthority
```

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ chmod 0600 ~/.Xauthority
```

- 위 방법도 해보았지만 여전히 오류가 발생하였다.

Solution 1

- Linux KVM Installation 챕터를 참고한 내용을 정리하자면,

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ xhost  
access control enabled, only authorized clients can connect  
SI:localuser:seongbinyoon  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo xhost  
X11 connection rejected because of wrong authentication.  
xhost: unable to open display "localhost:10.0"
```

```
~$ xhost
```

```
~$ sudo xhost
```

- xhost 명령어를 통해 연결 및 권한 확인을 했을 때, 사용자 계정은 enabled지만, root 권한은 rejected된 것을 볼 수 있다. 이 권한을 허용해주기 위해 아래 단계를 수행한다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ echo $DISPLAY  
localhost:10.0  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ xauth list  
ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC/unix:11  MIT-MAGIC-COOKIE-1  35ad68a8ddbe18b142d0653683e9288d  
ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC/unix:10  MIT-MAGIC-COOKIE-1  310887fed9277f9c9ccedc060599f0c2
```

```
~$ echo $DISPLAY
```

```
~$ xauth list
```

- \$DISPLAY의 값을 기억하고, 위 명령어로 \$DISPLAY 값에 맞는 magic cookie value를 복사한다. 이 경우 unix: 100에 해당하는 줄이다.

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo su -
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# vlm .Xauthority
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# ls -al
total 48
drwx----- 5 root root 4096 1월 30 12:48 .
drwxr-xr-x 20 root root 4096 1월 16 14:56 ..
-rw------- 1 root root 87 1월 30 12:09 .bash_history
-rw-r--r-- 1 root root 3106 10월 15 2021 .bashrc
drwx----- 3 root root 4096 1월 25 21:41 .cache
-rw----- 1 root root 20 1월 25 21:27 .lesshtst
drwxr-xr-x 3 root root 4096 1월 26 23:54 .local
-rw-r--r-- 1 root root 161 7월 9 2019 .profile
drwx----- 5 root root 4096 1월 16 14:59 snap
-rw-r--r-- 1 root root 0 1월 30 11:51 .sudo_as_admin_successful
-rw----- 1 root root 9288 1월 30 12:48 .viminfo
```

```
~$ sudo su
```

- root 계정으로 접속한다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# ls -al
total 52
drwx----- 5 root root 4096 1월 30 12:51 .
drwxr-xr-x 20 root root 4096 1월 16 14:56 ..
-rw------- 1 root root 87 1월 30 12:09 .bash_history
-rw-r--r-- 1 root root 3106 10월 15 2021 .bashrc
drwx----- 3 root root 4096 1월 25 21:41 .cache
-rw----- 1 root root 20 1월 25 21:27 .lesshtst
drwxr-xr-x 3 root root 4096 1월 26 23:54 .local
-rw-r--r-- 1 root root 161 7월 9 2019 .profile
drwx----- 5 root root 4096 1월 16 14:59 snap
-rw-r--r-- 1 root root 0 1월 30 11:51 .sudo_as_admin_successful
-rw----- 1 root root 9288 1월 30 12:48 .viminfo
-rw----- 1 root root 85 1월 30 12:52 .Xauthority
```

```
~# ls -al
```

- 위 명령어를 통해 .Xauthority 파일의 존재를 확인할 수 있다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:/home/ecl3# mv .Xauthority .Xauthority.old
```

```
~$ mv .Xauthority .Xauthority.old
```

- 혹시 모를 상황에 대비해 Xauthority 파일을 old로 백업한다.

```
~$ touch ~/.Xauthority
```

- Xauthority라는 파일을 생성한다.

x11 Forwarding의 한가지 단점은 매 세션마다 magic cookie value가 변경되어 권한 설정을 수동으로 해줘야 한다는 점이다. 따라서 아래 단계부터는 매 세션(로그아웃 후 재로그인 시)마다 진행해야 한다.

여기부터

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# xauth add ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC/unix:10 MIT-MAGIC-COOKIE-1 310887fed9277f9c9ccedc060599f0c
2
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# vim .Xauthority
```

```
~# xauth add <Magic cookie value>
```

- 위에서 복사한 Magic cookie value를 붙여넣고 xauth에 권한을 추가한다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# cat ~/.Xauthority
'ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC10MIT-MAGIC-COOKIE-1*U*Eg0***root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# export DISPLAY=localhost:10.0
```

```
~# cat ~/.Xauthority
cp <Magic cookie value> ~# export DISPLAY=localhost:10.0
```

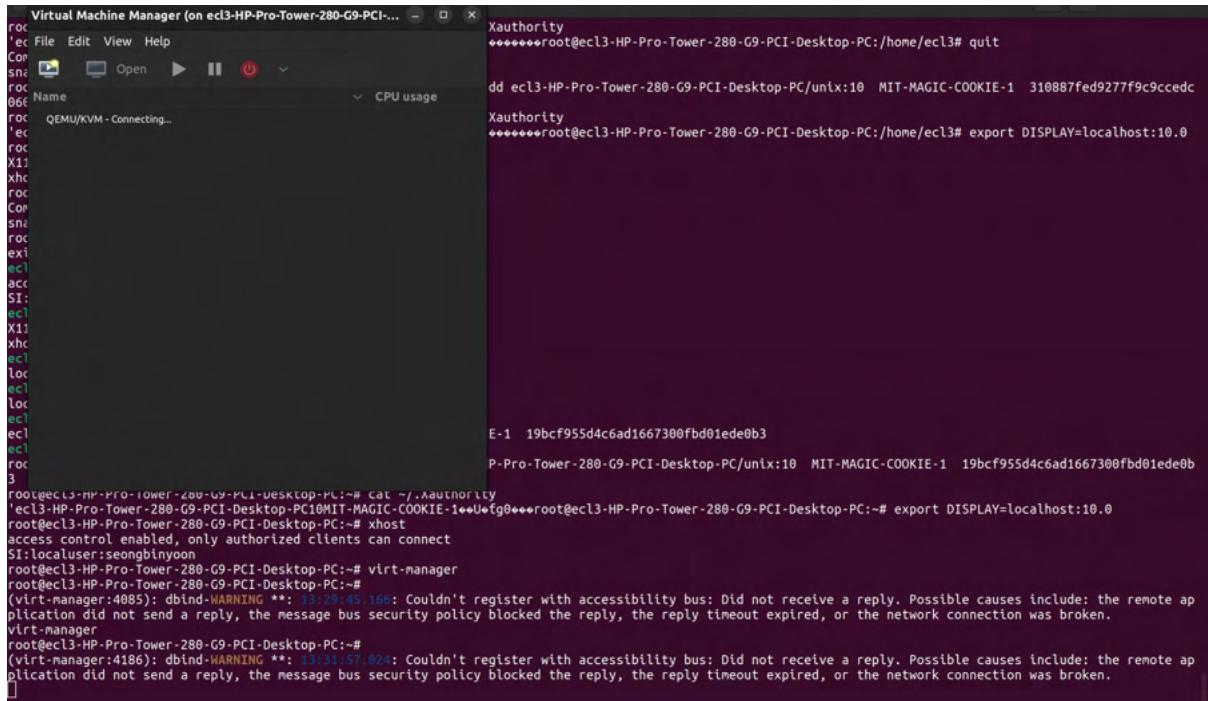
- 제대로 추가되었는지 확인한다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# xhost
access control enabled, only authorized clients can connect
SI:localuser:seongbinyoon
```

```
~# xhost
```

- 위 명령어로 접근 권한이 추가되었는지 확인한다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# virt-manager
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~#
(virt-manager:4085): dbind-WARNING **: 13:29:45.166: Couldn't register with accessibility bus: Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.
```



```
~# virt-manager
```

- virt-manager를 실행시키면 위와 같이 클라이언트 PC에서 서버 PC의 GUI를 볼 수 있다.

```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# exit
logout
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ xhost
access control enabled, only authorized clients can connect
SI:localuser:seongbinyoon
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo xhost
access control enabled, only authorized clients can connect
```

The screenshot shows a Linux desktop environment. On the left, the 'Virtual Machine Manager' window is open, displaying a list of virtual machines. On the right, a terminal window is running a command to export the DISPLAY variable to the local host.

```
E-1 19bcf955d4c6ad1667300fb01ede0b3
P-Pro-Tower-280-G9-PCI-Desktop-PC/unix:10 MIT-MAGIC-COOKIE-1 19bcf955d4c6ad1667300fb01ede0b
y
tg0***@root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# export DISPLAY=localhost:10.0

gister with accessibility bus: Did not receive a reply. Possible causes include: the remote ap
blocked the reply, the reply timeout expired, or the network connection was broken.

gister with accessibility bus: Did not receive a reply. Possible causes include: the remote ap
blocked the reply, the reply timeout expired, or the network connection was broken.

root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# virt-manager
(virt-manager:4223): dbind-WARNING **: 15:32:58.640: Couldn't register with accessibility bus: Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.
quit
Command 'quit' not found, but can be installed with:
snap install quit
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# quit
Command 'quit' not found, but can be installed with:
snap install quit
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~# exit
logout
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo virt-manager
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

```
~# exit
```

- exit으로 사용자 계정으로 돌아간다.

```
~$ xhost
```

```
~$ sudo xhost
```

- 위 명령어로 사용자 및 root 계정이 모두 접근 허용인지 확인한다.

```
~$ sudo virt-manager
```

- virt-manager를 실행시키면 마찬가지로 클라이언트 PC에서 서버 PC의 GUI를 볼 수 있다.

여기까지

Trouble 2

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$ xauth list  
xauth: /home/ecl3/.Xauthority not writable, changes will be ignored
```

```
~$ xauth list
```

- 접근 권한 추가를 해주다가 위와 같은 not writable 오류가 발생했다.

Solution 2

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$ sudo chmod 666 /home/ecl3/.Xauthority  
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: ~$
```

```
~$ sudo chmod 666 /home/<username>/.Xauthority
```

- .Xauthority 파일을 만지다가 충분히 발생할 수 있는 문제로, .Xauthority의 파일을 확인해 보면 권한이 600(root 계정만 읽고 쓰기 가능)으로 되어 있는 경우가 있다.
- 일단 VPN을 통해 교내 IP로 연구실 PC로만 접속하므로 VM을 설정하고 실행하기 위해 진행하는 것으로, 보안성을 생각한다면 다른 방법을 강구해야 한다.
- 위 명령어는 이 파일의 권한을 모든 사용자가 읽고 쓰게(666) 변경한다.

Trouble 3

연구실 컴퓨터를 세팅한 뒤에 할당된 아이피를 통ssh 를 통해 연결하려는데 연결이 되지 않고 오류가 발생하였다.

```
devtae@devtae:~/sources$ sudo ssh ecl4@  
ssh: connect to host 1[REDACTED] port 22: Connection timed out
```

Connection timed out error 발생

연구실 컴퓨터 측에서 /etc/ssh/sshd_config 파일 또한 수정하였고 ufw(ubuntu firewall) 설정까지 마친 상황이었다. 하지만, 무슨 이유에선지 Connection timed out 오류가 발생하였다.

네트워크 연결에서 문제인 듯 보여 ping 프로그램을 돌려보았지만 request에 대한 response 데이터가 오지 않았다.

Solution 3

학교 연구실의 네트워크는 학교 네트워크망 안에서 구축된 아이피로 할당되어 있다. 실제로, 학교에서 허용되지 않은 외부의 아이피를 차단하는 네트워크 시스템을 가동 중이라고 한다.

따라서, 모바일 핫스팟으로 연결되어 있는 노트북으로는 학교 네트워크망에 접속하지 못한 것이고 결과적으로 에러가 뜨는 것이었다. 그런 와중에도 옆 컴퓨터에서 ssh 접속을 시도했을 때는 성공적으로 작동하는 것을 볼 수 있다.

따라서, 다음 방법으로 학교에서 지원하는 VPN 프로그램을 활용하여 접속하면 성공적으로 작동할 것이다.

1. 한양대학교 포탈에 접속하여 VPN 신청
2. 승인된 후에 vpn.hanyang.ac.kr에 접속하여 메뉴얼 방식대로 VPN 프로그램 작동
3. 이후 이전과 동일한 방법으로 ssh 접속

```
devtae@devtae:~/sources$ sudo ssh ecl4@[REDACTED]
ecl4@[REDACTED]'s password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Last login: Mon Jan 30 12:11:02 2023 from [REDACTED]
```

성공적으로 로그인에 성공하였음

그 이후 성공적으로 작동하는 것을 확인할 수 있다.

+) 추가적으로 매번 아이피를 입력하는 것이 불편하다. 따라서, 스크립트를 짜서 그것만 실행한다면 로그인이 되도록 만들 수 있다.

connect (file)

```
ssh 'username'@'xxx.xxx.xxx.xxx'
```

이후 터미널에

```
sudo sh connect
```

를 하여 쉽게 접속할 수 있다.

VM Installation

KVM 설치 후 VM(Ubuntu 22.04.1 LTS 배포판)을 설치하는 방법에 대해 담고 있다.

이전 단계는 Linux KVM Installation을 참고하자.

1. Ubuntu 22.04.1 LTS 다운로드

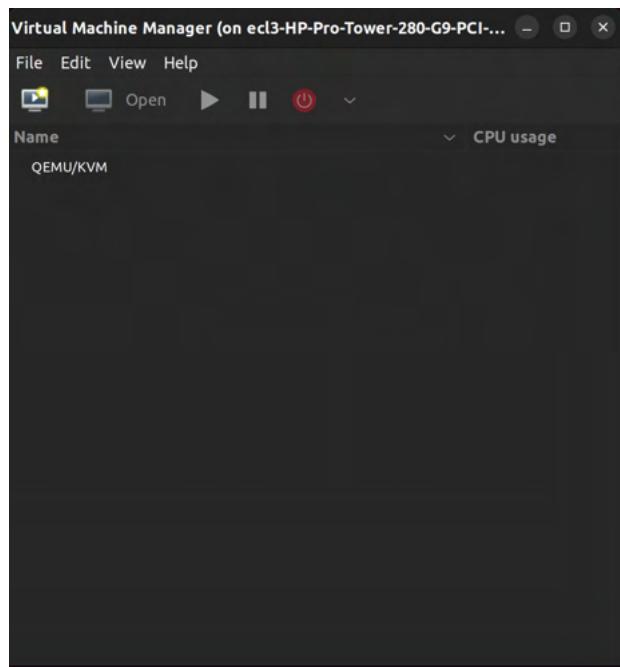
```
root@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC:/home/ecl3# wget https://releases.ubuntu.com/22.04/ubuntu-22.04.1-desktop-amd64.iso
--2023-01-31 15:31:48-- https://releases.ubuntu.com/22.04/ubuntu-22.04.1-desktop-amd64.iso
Resolving releases.ubuntu.com (releases.ubuntu.com)... 185.125.190.40, 91.189.91.124, 185.125.190.37, ...
Connecting to releases.ubuntu.com (releases.ubuntu.com)|185.125.190.40|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3826831360 (3.6G) [application/x-iso9660-image]
Saving to: ‘ubuntu-22.04.1-desktop-amd64.iso’

ubuntu-22.04.1-desktop-amd64.iso      5%[==>] 202.51M  11.5MB/s    eta 5m 32s
```

```
~# wget https://releases.ubuntu.com/22.04/ubuntu-22.04.1-desktop-amd64.iso
```

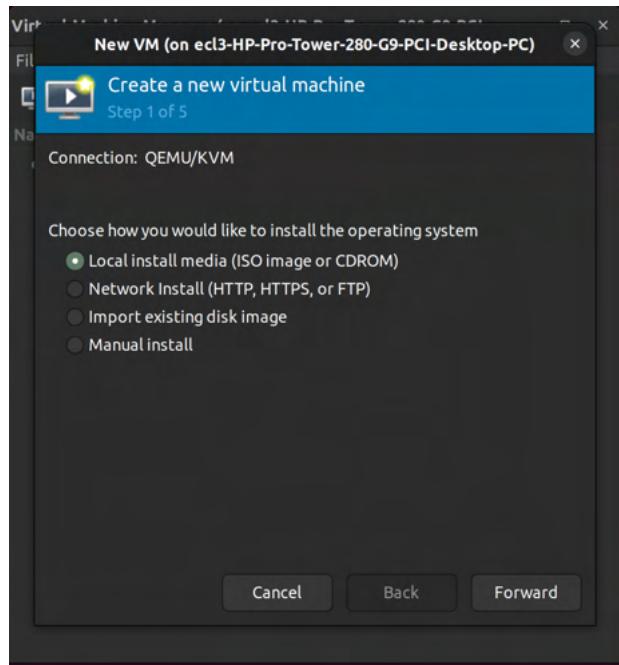
- wget 명령어로 우분투 iso 파일을 다운로드 받는다.
- 연구실의 경우 ssh 접속으로 데스크탑 로컬에 다운로드 받는다.

2. Configuration 설정(RAM, CPU, Network 등)

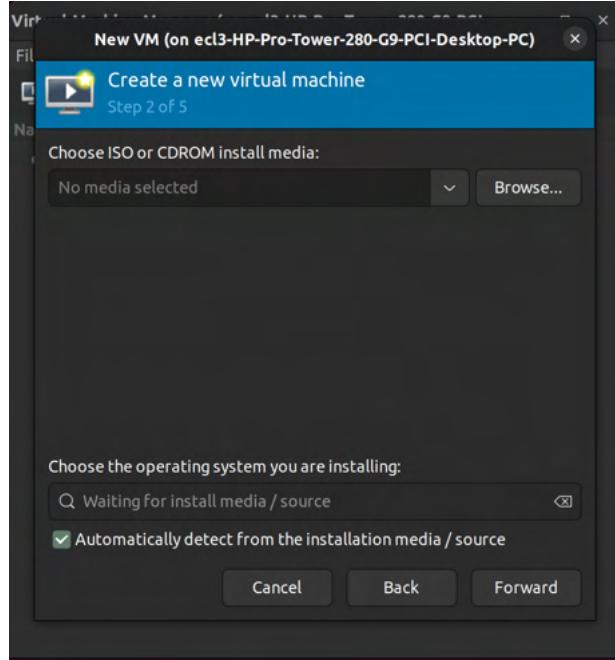


```
~# virt-manager
```

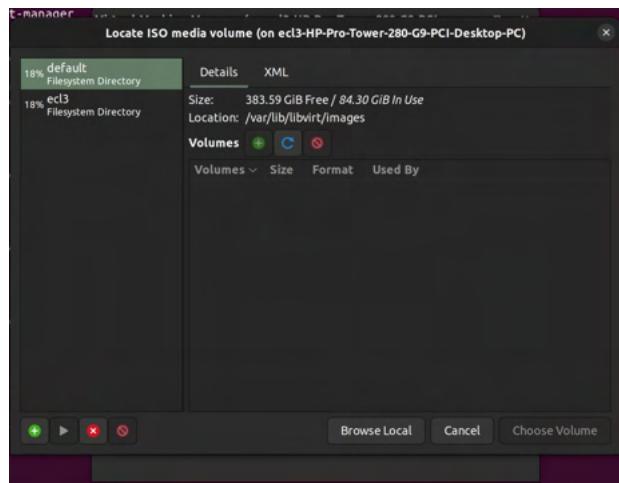
- virt-manager를 실행한다.
- File 아래 새 VM 아이콘을 클릭한다.



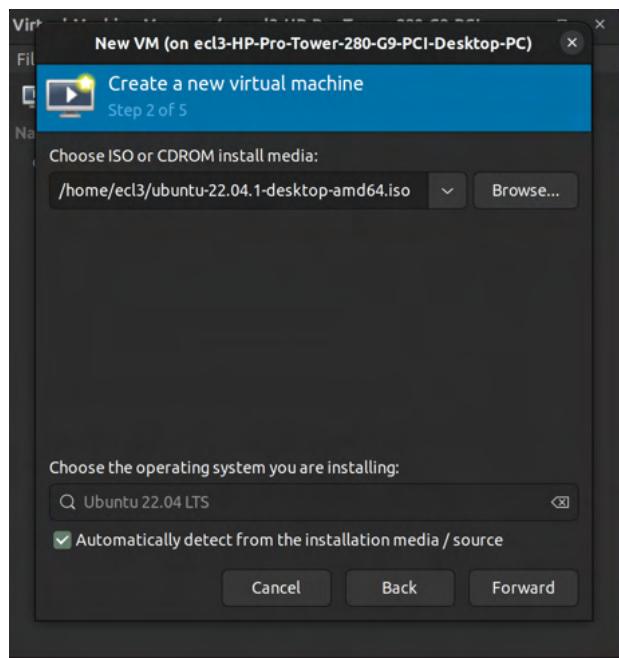
- Local install media를 선택 후 Forward를 클릭한다.



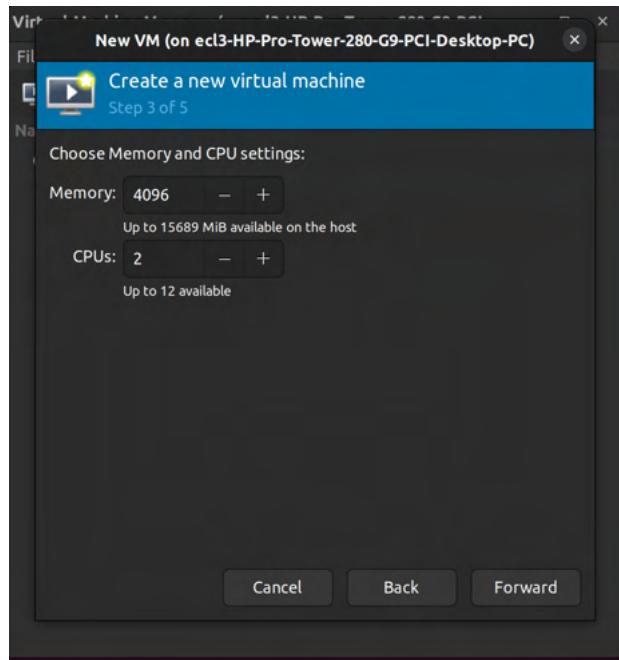
- 위와 같은 창이 뜨면 Browse..를 클릭한다.



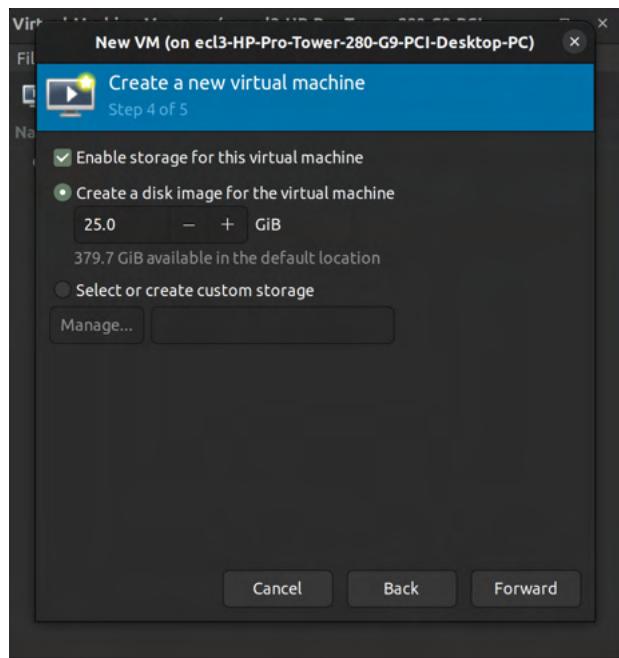
- default 디렉토리에 설치를 할 예정이므로 default를 클릭 후 Browse Local을 클릭한다.



- 아래 자동으로 적절한 우분투 버전이 감지되었는지 확인 후 Forward를 클릭한다.

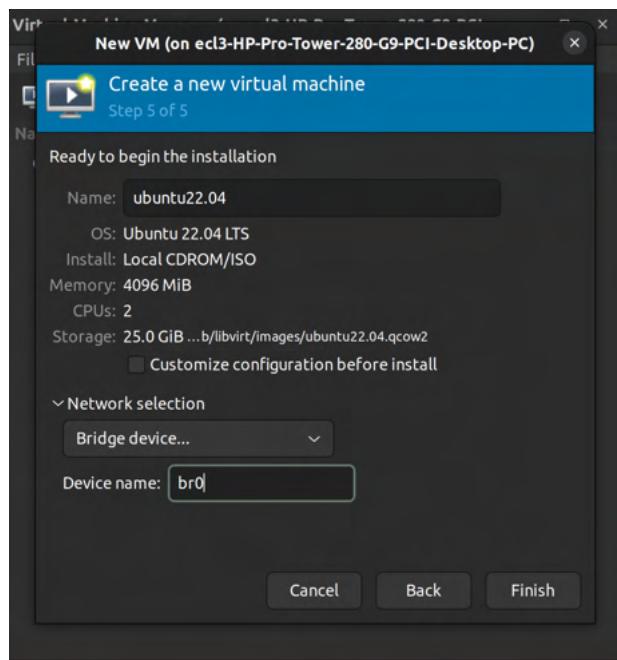
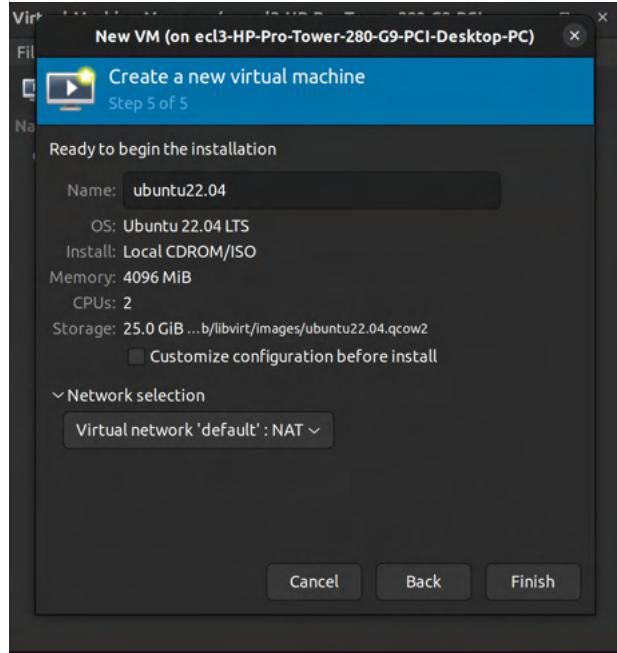


- RAM 용량 및 CPU 개수를 설정 후 Forward를 클릭한다.
- 이 경우 default 값인 4096MB, 2개로 설정했다.



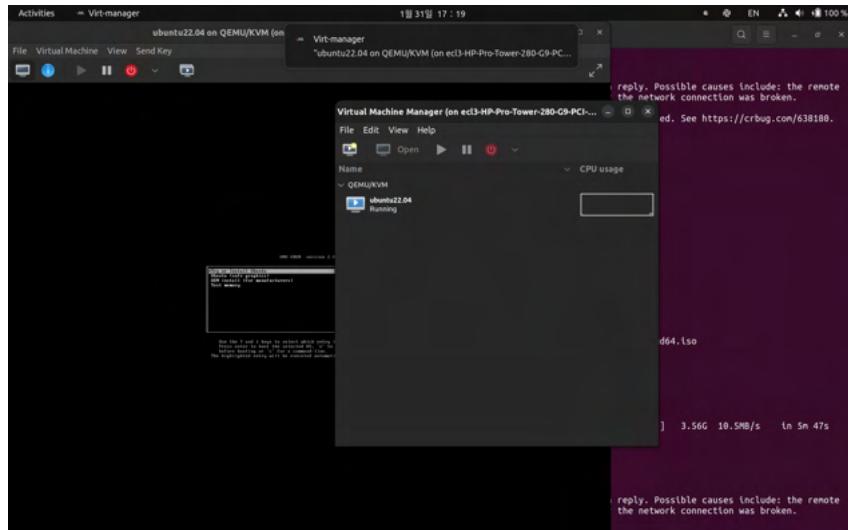
- disk image 크기를 설정 후 Forward를 클릭한다.
- 이 경우 default 값인 25.0 GiB로 하였다.

- 커스텀으로 디스크 이미지 공간을 설정하고 싶으면 아래 Select or create custom storage를 클릭하고 아래에 설정할 수 있다.

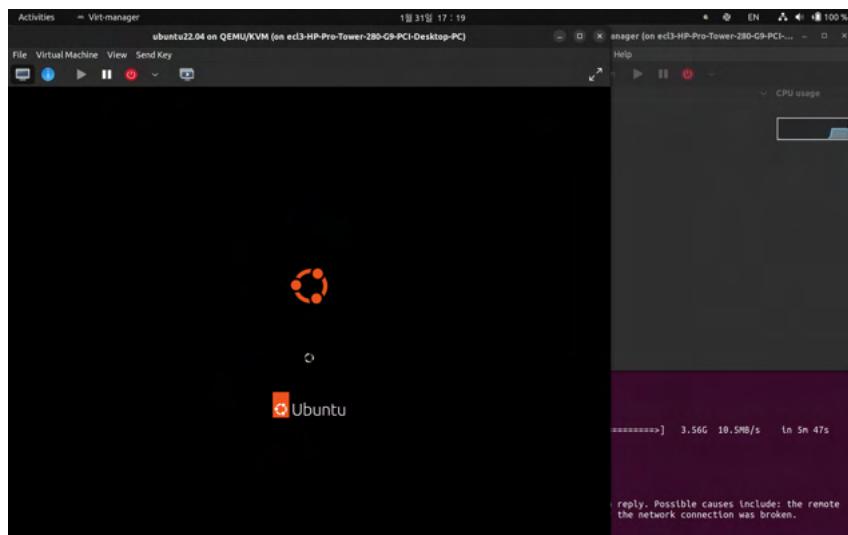


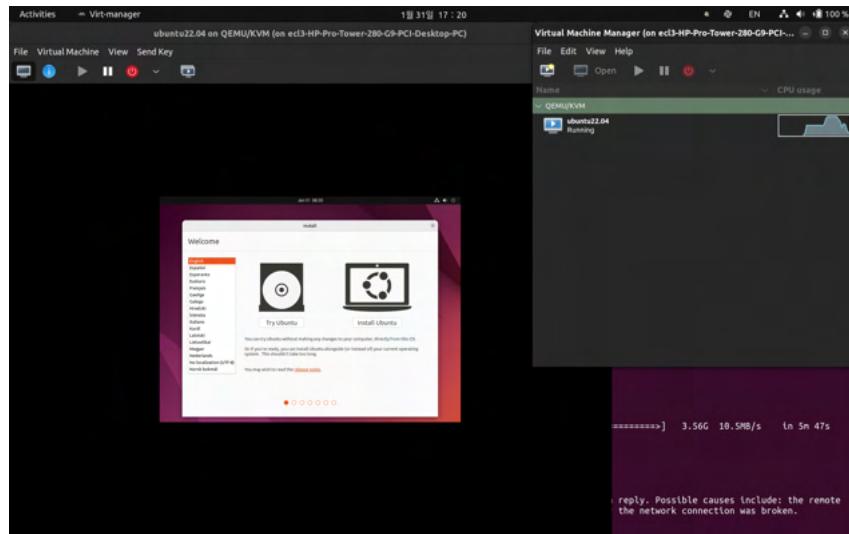
- 마지막으로 Network 방식을 설정 후 Finish를 클릭한다.
- default 값인 NAT와 Bridge 방식이 있는데, 쓰임에 따라 적절한 방식을 선택한다.
- 이 경우 NAT를 선택하였다.

3. Ubuntu Installation

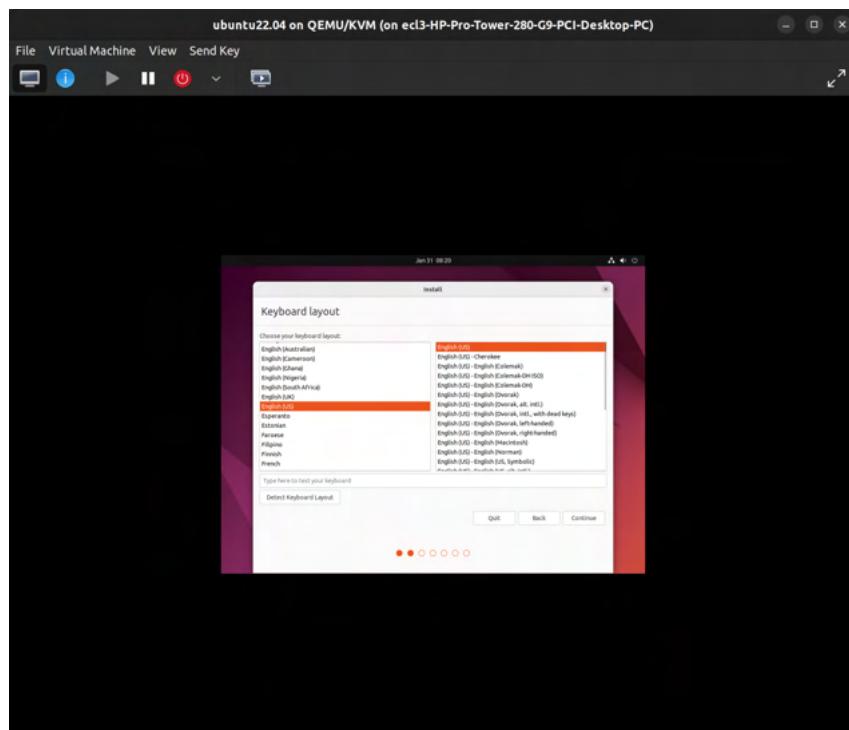


- 화면이 꺼지고 GRUB가 뜨면 Try or Install Ubuntu를 선택하고 엔터

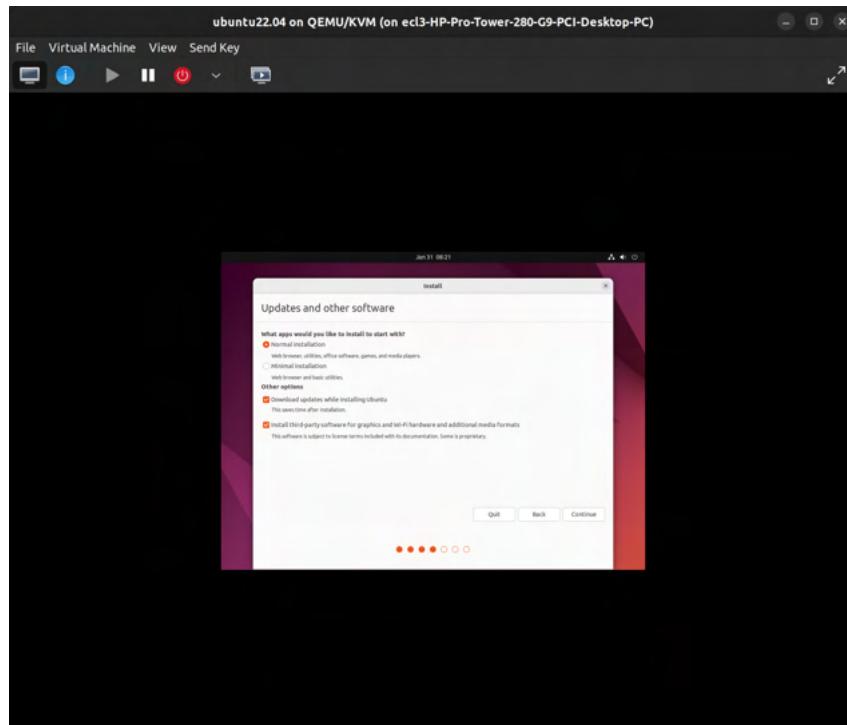




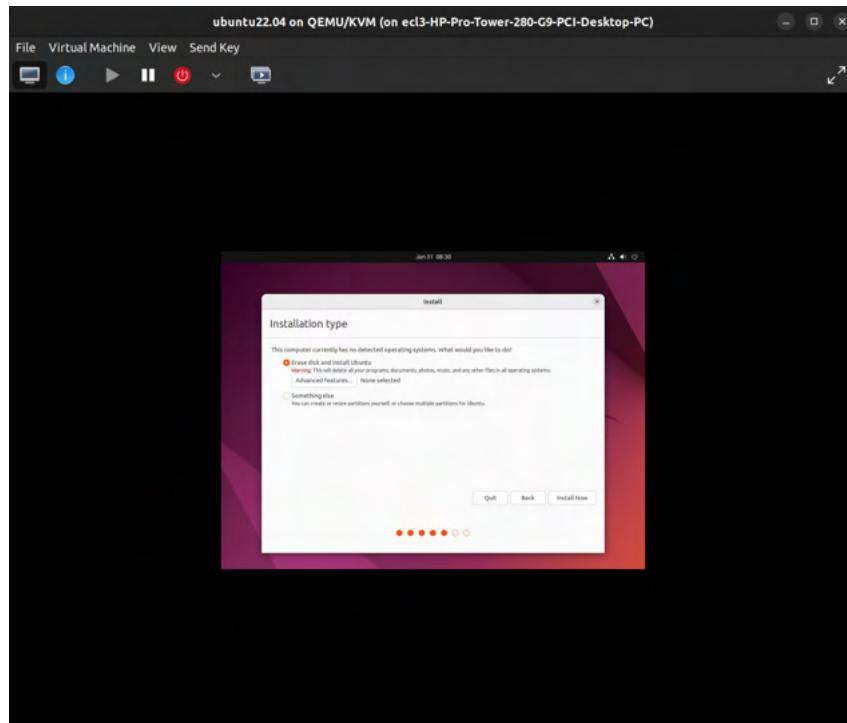
- 부팅이 완료되고 위와 같은 화면이 뜨면 Install Ubuntu를 클릭



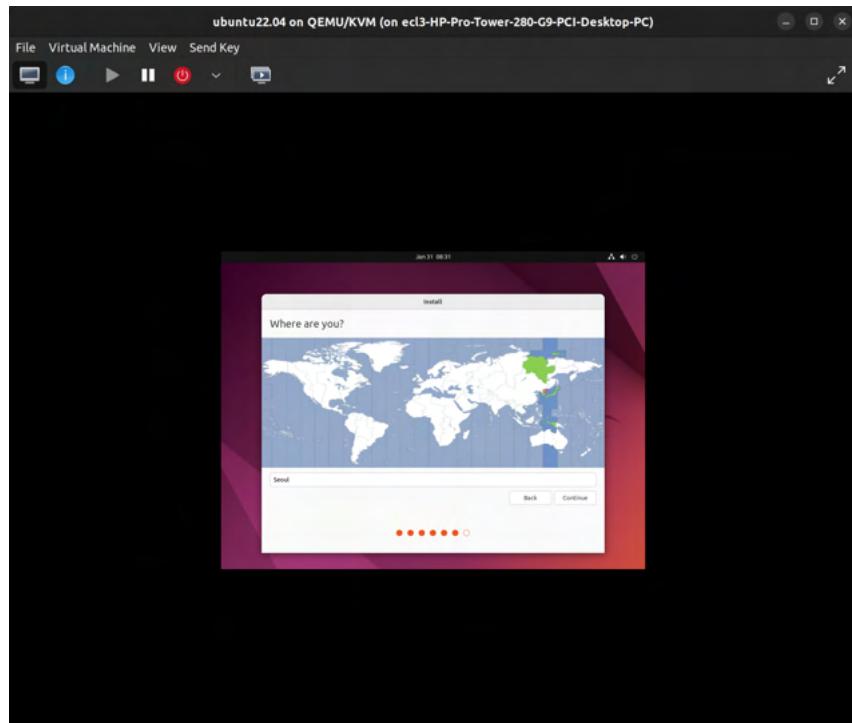
- English 선택 후 Continue를 클릭



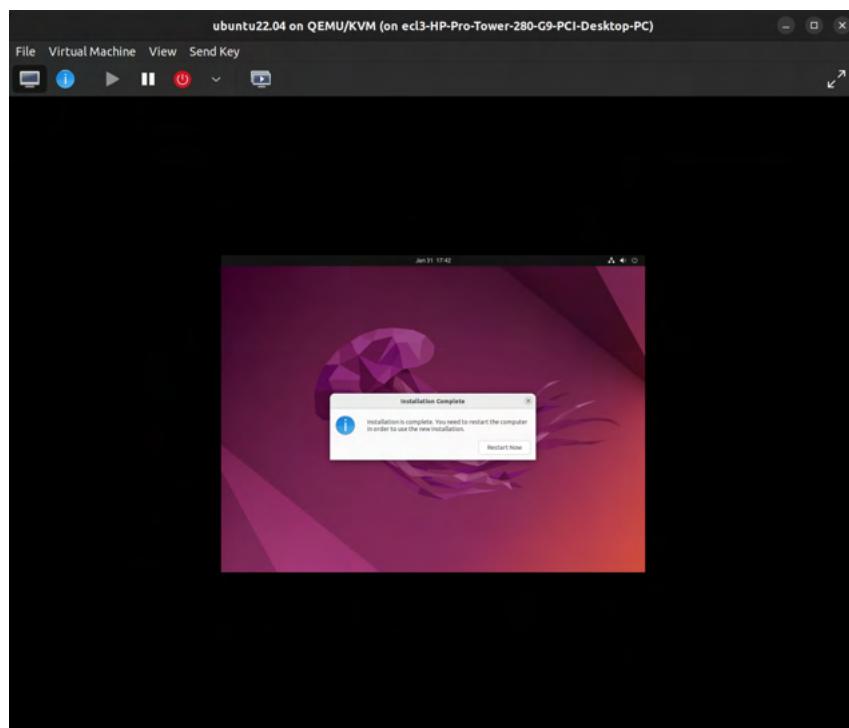
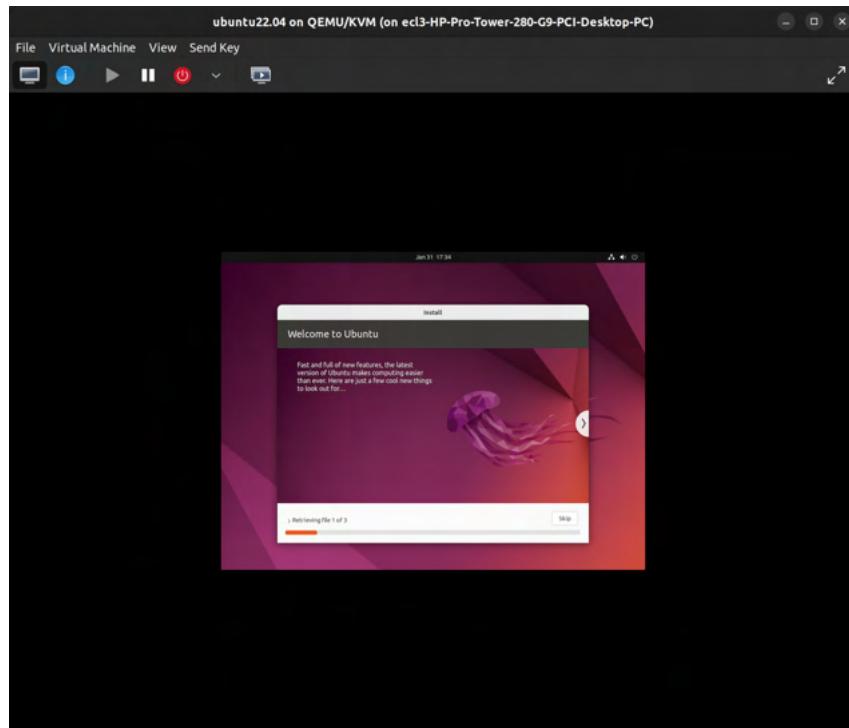
- Normal Installation 선택
- Download updates while installing ubuntu 선택
- Install third party software... 선택 후 Continue

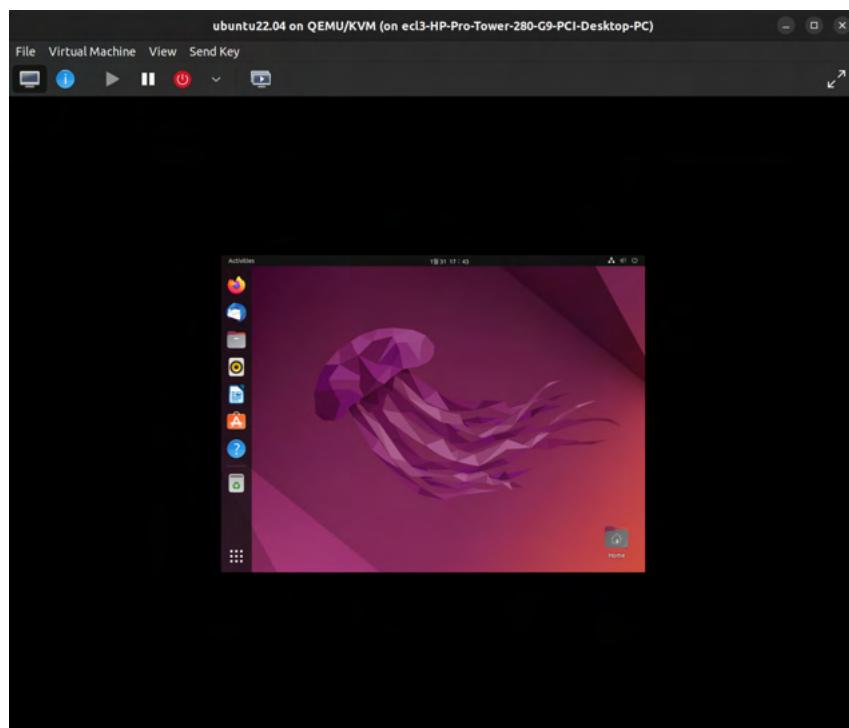
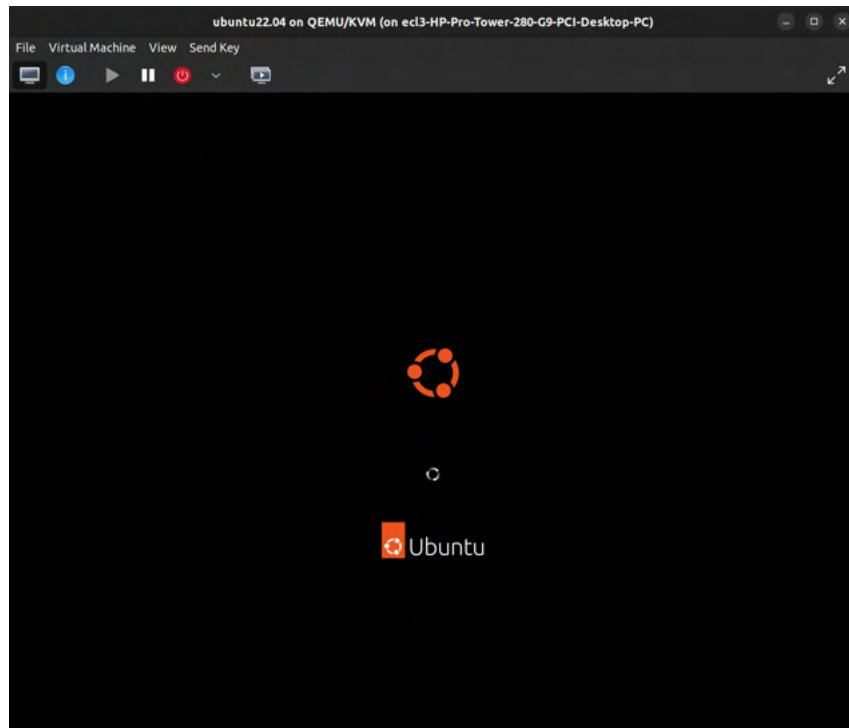


- Erase disk and install Ubuntu 선택 후 Install Now 클릭
- Dual boot 시에는 디스크 파티션 한 곳에 설치해야 하므로 Something else를 선택해 swap 및 install partition을 따로 지정해 줬지만, VM 설치는 그럴 필요가 없다.

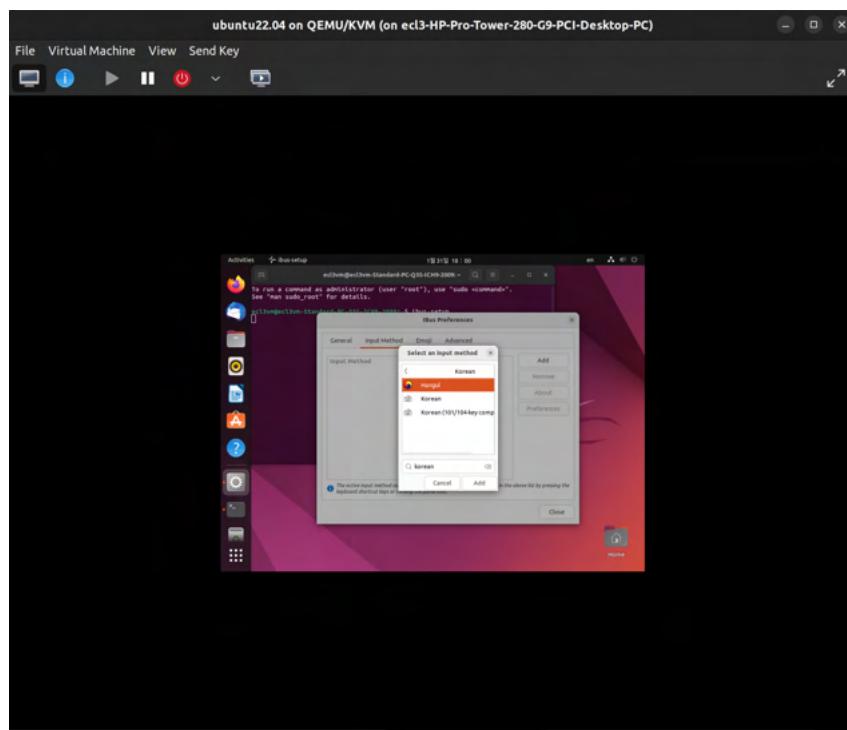
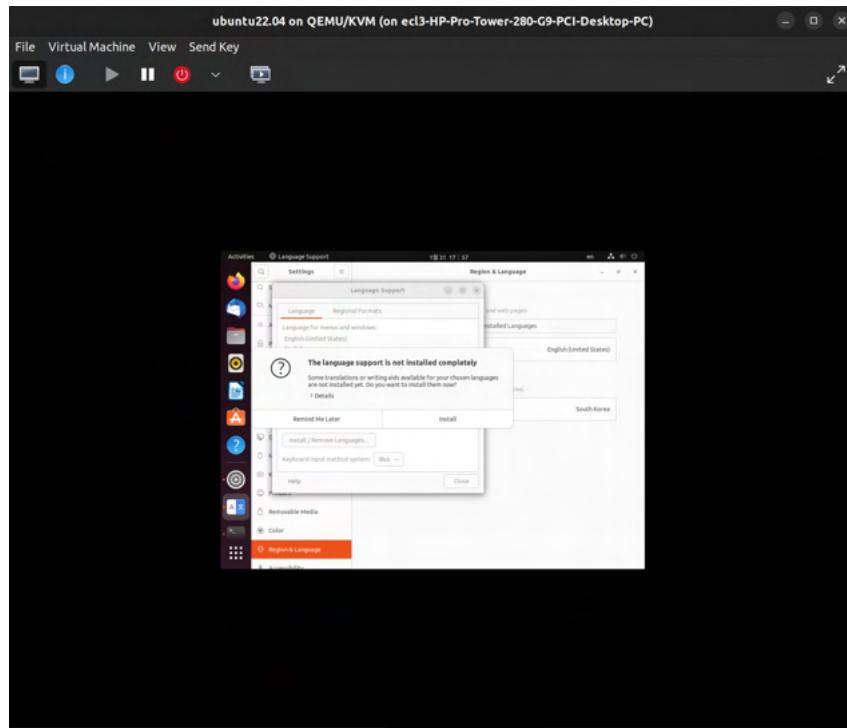


- 국가/지역 설정 → 사용자 계정(Username, Password 등) 설정 후 Continue 클릭





- 이 후 설치 및 재부팅이 진행되면 VM 설치는 완료된다.



- 그 외 한글 등 세부 설정은 Optional하다.

ioctl Function

ioctl 함수

- 하드웨어의 제어와 상태 정보를 얻기 위해 사용되는 함수
- 일반적으로 I/O Control에 관련된 작업을 수행하는 함수
- 대부분의 ioctl 메소드 구현은 cmd 인수 값에 따라 올바른 동작을 선택하는 switch문으로 구성한다.

User Space에서 ioctl 함수 system call

```
#include <sys/unistd.h>
int ioctl(int fd, int cmd, ...)
```

- 첫 번째 인자는 fd(file descriptor)를 넣고, 두 번째 인자 cmd는 다양한 Request Operation Code 중에서 적절한 값을 설정하면 된다. 마지막 인자들은 ioctl Operation의 처리 대상인 데이터를 입력한다.

Kernel Space에서 ioctl 함수 system call

```
int (*unlocked_ioctl) (struct file *filp, unsigned int cmd, unsigned long arg);
```

- inode와 filp 포인터는 응용프로그램의 fd(file descriptor)와 일치하는 인수이다.
- cmd 인수는 명령을 나타내는 응용프로그램의 인수를 전달한다.
- arg 인수는 명령 실행의 결과 데이터가 전달되는 unsigned long 타입의 정수 또는 포인터이다.

cmd 명령의 해석 매크로 함수

- _IOC_NR : 구분 번호 필드 값을 읽는 매크로

- `_IOC_TYPE` : 매직 넘버 필드 값을 읽는 매크로
- `_IOC_SIZE` : 데이터의 크기 필드 값을 읽는 매크로
- `_IOC_DIR` : 읽기와 쓰기 속성 필드 값을 읽는 매크로

example)

```
if (_IOC_TYPE(cmd) != MY_MAGIC) return -EINVAL;
```

cmd 명령의 작성 매크로 함수

`cmd(request)`에는 명령어를 구성해야 하는데, 다음 4개의 매크로를 활용하여 사용 가능하다.

- `_IO(type, nr)` : 부가적인 데이터가 없는 명령을 만드는 매크로
- `_IOR(type, nr, size)` : 데이터를 읽어오기 위한 명령을 작성, kernel → user 메모리 영역으로 복사가 일어나는 `copy_to_user` 동작이 일어날 때 사용
- `_IOW(type, nr, size)` : 데이터를 써 넣기 위한 명령을 작성, user → kernel 메모리 영역으로 복사가 일어나는 `copy_from_user` 동작이 일어날 때 사용
- `_IORW(type, nr, size)` : 디바이스 드라이버에서 읽고 쓰기 위한 명령을 작성하는 매크로, 양 쪽 영역으로 모두 사용될 때 사용

example)

```
_IOW (매직넘버, 구분번호, 정수형)
```

`type`은 8 bit 의 Magic Number이고, `nr`은 명령어를 구분하기 위해 사용되는 명령어마다 유일한 정수값이다. 마지막으로, `size`는 커널 영역과 사용자 영역 사이에서 오고 갈 때의 데이터 크기를 바이트 단위로 나타낸 것이다.

이후 Advanced Char Device Driver 에서 위 함수에 대한 실습을 해볼 수 있다.

Advanced Character Device Driver

하드웨어의 제어와 상태 정보를 얻기 위해 ioctl을 이용한 문자형 디바이스 드라이버를 다루고 있다.
ioctl Function을 참조하자.

Advanced char device driver programming(scull0~scull3)

1. 소스코드 작성

Makefile

```
#If KERNELRELEASE is defined, we've been invoked from the kernel build system and can use its language.
ifeq ($(KERNELRELEASE),)
    obj-m := scull.o

#Otherwise we were called directly from the command line; invoke the kernel build system.
else
    KERNELDIR ?= /lib/modules/$(shell uname -r)/build
    PWD := $(shell pwd)

default:
    $(MAKE) -C $(KERNELDIR) M=$(PWD) modules
endif
```

```
ifeq ($(KERNELRELEASE),)
```

- 만약 환경변수 KERNELRELEASE가 정의되어 있지 않다면 다음을 추가

```
obj-m := scull.o
```

- 컴파일할 모듈의 이름(scull.o)

```
KERNELDIR ?= /lib/modules/$(shell uname -r)/build
```

- 커널 디렉토리(KERNELDIR) 변수에 커널의 최신버전 문자열로 치환한 커널 버전에 맞는 빌드 디렉토리를 등록

```
PWD := $(shell pwd)
```

- PWD(print working directory) 변수에 현재 소스코드가 위치하고 있는 디렉토리(해당 명령어를 입력하고 있는 디렉토리)를 등록
- 해당 연산자 := 는 한번 등록하면 고정

```
# default: 뒤에 아무 옵션을 주지 않았을 때 실행되는 명령어
# KERNELDIR: 커널 디렉토리
# M: 서브 디렉토리
# modules: 모듈 컴파일
default:
$(MAKE) -C $(KERNELDIR) M=$(PWD) modules
```

- 커널 디렉토리(KERNELDIR)에 현재 모듈이 위치하고 있는 디렉토리를 서브 디렉토리로 인식시켜 모듈(target)을 컴파일

scull_load

```
#!/bin/sh
module="scull"
device="scull"
mode="664"

# Invoke insmod with all arguments we got
# And use a pathname, as newer modutils don't look in . by default
/sbin/insmod ./${module}.ko $* || exit 1

# Remove stale nodes
rm -f /dev/${device}[0-3]

major=$(awk "\$2==\"$module\" {print \$1}" /proc/devices)

mknod /dev/${device}0 c $major 0
mknod /dev/${device}1 c $major 1
mknod /dev/${device}2 c $major 2
mknod /dev/${device}3 c $major 3

# Give appropriate group/permissions, and change the group.
# Not all distributions have staff, some have "wheel" instead.
group="staff"
grep -q '^staff:' /etc/group || group="wheel"

chgrp $group /dev/${device}[0-3]
chmod $mode /dev/${device}[0-3]

~
```

21,62

All

```
#!/bin/sh
```

- #! 로 스크립트를 실행할 쉘을 지정

- 이 경우 스크립트를 실행하면 아래와 같이 /bin/sh에 연결된 dash 쉘의 도움을 받아 실행한다. (이전에는 bash였다)

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~$ ls -al /bin/sh
lrwxrwxrwx 1 root root 4 1월  4 19:23 /bin/sh -> dash
```

```
/sbin/insmod ./module.ko $* || exit 1
```

- insmod로 scull.ko 드라이버를 커널에 적재

```
rm -f /dev/${device}[0-3]
```

- mknod로 드라이버 파일을 만들기 전, 남아있던 낡은 노드(드라이버 파일)를 제거(초기화)

```
major=$(awk "\$2==\"$module\" {print \$1}" /proc/devices)
```

- awk를 사용하여 /proc/devices에 할당된 주번호를 찾아 major 변수에 입력
- /proc/devices 파일에서 첫 번째 열에 할당된 major 번호가 나오고, 두 번째 열에서 dev 이름이 나온다.

```
mknod /dev/${device}0 c $major 0
mknod /dev/${device}1 c $major 1
mknod /dev/${device}2 c $major 2
mknod /dev/${device}3 c $major 3
```

- mknod 명령어로 파일 시스템에서 4개의 디바이스 노드(드라이버 파일)를 생성
- 예를 들어, 'mknod /dev/scull0 c 254 0' 과 같이 name이 scull0, 주번호가 254, 부번호가 0인 char device file을 생성
- 제거 시에는 rmnod를 사용

scull_unload

```
#!/bin/sh
module="scull"
device="scull"

# Invoke rmmod with all arguments we got
/sbin/rmmod $module $* || exit 1

# Remove stale nodes
rm -f /dev/${device} /dev/${device}[0-3]
```

```
/sbin/rmmod $module $* || exit 1
```

- rmmod로 커널에 적재했던 scull 드라이버를 해제

```
rm -f /dev/${device} /dev/${device}[0-3]
```

- /dev/scull 디렉토리에 있는 4개의 디바이스 노드(드라이버 파일) scull0, scull1, scull2, scull3을 삭제

access_ok_version.h

```
/*
 * Header file access_ok_version.h
 */
#include <linux/version.h>

#if LINUX_VERSION_CODE < KERNEL_VERSION(5,0,0)
#define access_ok_wrapper(type,arg,cmd) \
    access_ok(type, arg, cmd)

#else
#define access_ok_wrapper(type,arg,cmd) \
    access_ok(arg, cmd)
#endif
~
```

- access_ok_에 관한 헤더파일이다.

scull.h

```

/* This is header file 'scull.h', and it contains definitions of the scull file */

/* File name */
#ifndef _SCULL_H
#define _SCULL_H

/* Ioclt header file include*/
#include <linux/ioclt.h>

/* Dynamic major number by default */
#ifndef SCULL_MAJOR
#define SCULL_MAJOR 0
#endif

/* scull0 to scull3 */
#ifndef SCULL_NUM_DEV
#define SCULL_NUM_DEV 4
#endif

/* Memory area */
#ifndef SCULL_QUANTUM
#define SCULL_QUANTUM 4000
#endif

/* Array of memory area */
#ifndef SCULL_QSET
#define SCULL_QSET 1000
#endif

/* Define scull quantum set */
struct scull_qset {
    void **data;
    struct scull_qset *next;
};

/* Define scull devices */
struct scull_dev {
    struct scull_qset *data;          /* Pointer to first quantum set */
    int quantum;                    /* the current quantum size */
    int qset;                       /* the current array size */
    unsigned long size;             /* amount of data stored here */
    unsigned int access_key;         /* used by sculluid and sculpriv */
    struct mutex lock;              /* mutual exclusion semaphore */
    struct cdev cdev;               /* Char device structure */
};

/* Global variables (scull.c) */
extern int scull_major;
extern int scull_num_dev;
extern int scull_quantum;
extern int scull_qset;

/* Prototypes for shared functions */
int scull_trim(struct scull_dev *dev);
ssize_t scull_read(struct file *filp, char __user *buf, size_t count, loff_t *f_pos);
ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos);
loff_t scull_llseek(struct file *filp, loff_t off, int whence);
long scull_ioctl(struct file *filp, unsigned int cmd, unsigned long arg);

#endif /* _SCULL_H */

```

- scull_dev 구조체, scull_qset 구조체, 함수의 prototype, 각종 변수 등의 정의를 담고있는 헤더파일

scull.c

```

#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/slab.h>
#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/types.h>
#include <linux/fcntl.h>
#include <linux/cdev.h>
#include <linux/proc_fs.h>
#include <linux/seq_file.h>
#include <asm/uaccess.h>
#include "scull.h"
#include "ioctl.h"
#include "access_ok_version.h"

/* Parameters */
int scull_major = SCULL_MAJOR; /* major number */
int scull_minor = 0; /* minor number */
int scull_num_dev = SCULL_NUM_DEV; /* number of bare scull devices */
int scull_quantum = SCULL_QUANTUM;
int scull_qset = SCULL_QSET;

module_param(scull_major, int, S_IRUGO);
module_param(scull_minor, int, S_IRUGO);
module_param(scull_num_dev, int, S_IRUGO);
module_param(scull_quantum, int, S_IRUGO);
module_param(scull_qset, int, S_IRUGO);

MODULE_AUTHOR("Alessandro Rubini, Jonathan Corbet");
MODULE_LICENSE("Dual BSD/GPL");

```

- 헤더파일 include
- 필요한 변수 선언 (major num, minor num, device 개수, 디바이스 메모리인 quantum, quantum의 집합이자 array인 qset 등)

```

/* Define scull devices */
struct scull_dev *scull_devices;           /* allocated in scull_init_module */

int scull_trim(struct scull_dev *dev) {
    struct scull_qset *next, *dptr;
    int qset = dev->qset;      /* "dev" is not-null */
    int i;

    for (dptr = dev->data; dptr; dptr = next) {      /* all the list items */
        if (dptr->data) {
            for (i = 0; i < qset; i++)
                kfree(dptr->data[i]);
            kfree(dptr->data);
            dptr->data = NULL;
        }
        next = dptr->next;
        kfree(dptr);
    }
    dev->size = 0;
    dev->quantum = scull_quantum;
    dev->qset = scull_qset;
    dev->data = NULL;
    return 0;
}

/* Open */
int scull_open(struct inode *inode, struct file *filp) {
    struct scull_dev *dev; /* device information */

    dev = container_of(inode->i_cdev, struct scull_dev, cdev);
    filp->private_data = dev; /* for other methods */

    /* now trim to 0 to the length of the device if open was write-only */
    if ((filp->f_flags & O_ACCMODE) == O_WRONLY) {
        scull_trim(dev); /* ignore errors */
    }
    return 0; /* success */
}

```

- scull_dev 타입의 scull_devices 선언
- scull_trim() 함수 정의
 - 전체 data를 해제하고 scull_open() 시(writing하기 위해 파일을 열 때) 또는 module cleanup 시 (사용한 메모리를 해제하기 위해) 호출
- 디바이스 open 함수 정의
 - container_of 매크로로 inode → i_cdev가 가리키는 구조체 cdev를 포함하는 구조체 scull_dev의 주소를 반환

```

/* Release */
int scull_release(struct inode *inode, struct file *filp) {
    return 0;
}

/* Follow the list */
static struct scull_qset *scull_follow(struct scull_dev *dev, int n) {
    struct scull_qset *qs = dev->data;
    /* Allocate first qset explicitly if need be */
    if (!qs) {
        qs = dev->data = kmalloc(sizeof(struct scull_qset), GFP_KERNEL);
        if (qs == NULL)
            return NULL; /* Never mind */
        memset(qs, 0, sizeof(struct scull_qset));
    }
    /* Then follow the list */
    while (n--) {
        if (!qs->next) {
            qs->next = kmalloc(sizeof(struct scull_qset), GFP_KERNEL);
            if (qs->next == NULL)
                return NULL; /* Never mind */
            memset(qs->next, 0, sizeof(struct scull_qset));
        }
        qs = qs->next;
        continue;
    }
    return qs;
}

```

- 디바이스 release 함수 정의
- scull_qset 타입의 *scull_follow 함수 정의
 - 필요 시 명시적으로 메모리를 할당하기 위해 qs가 NULL인 경우, *qs에 scull_qset 크기만큼 동적 할당하고 memset으로 초기화
 - 리스트를 따라 개수만큼 포인터가 가리키는 qs(scull_qset)를 메모리 할당 후 초기화

```

/* Read */
ssize_t scull_read(struct file *filp, char __user *buf, size_t count, loff_t *f_pos) {
    struct scull_dev *dev = filp->private_data;
    struct scull_qset *dptr; /* the first listitem */
    int quantum = dev->quantum, qset = dev->qset;
    int itemsize = quantum * qset; /* how many bytes in the listitem */
    int item, s_pos, q_pos, rest;
    ssize_t retval = 0;

    if (down_interruptible(&dev->sem))
        return -ERESTARTSYS;
    if (*f_pos >= dev->size)
        goto out;
    if (*f_pos + count > dev->size)
        count = dev->size - *f_pos;

    /* find listitem, qset index, and offset in the quantum */
    item = (long)*f_pos / itemsize;
    rest = (long)*f_pos % itemsize;
    s_pos = rest / quantum; q_pos = rest % quantum;

    /* follow the list up to the right position(defined elsewhere) */
    dptr = scull_follow(dev, item);

    if (dptr == NULL || !dptr->data || !dptr->data[s_pos])
        goto out; /* don't fill holes */

    /* read only up to the end of this quantum */
    if (count > quantum - q_pos)
        count = quantum - q_pos;

    if (copy_to_user(buf, dptr->data[s_pos] + q_pos, count)) {
        retval = -EFAULT;
        goto out;
    }
    *f_pos += count;
    retval = count;

out:
    up(&dev->sem);
    return retval;
}

```

- scull_read() 함수 정의

```

/* Write */
ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos) {
    struct scull_dev *dev = filp->private_data;
    struct scull_qset *dptr;
    int quantum = dev->quantum, qset = dev->qset;
    int itemsize = quantum * qset;
    int item, s_pos, q_pos, rest;
    ssize_t retval = -ENOMEM; /* value used in "goto out" statements */

    if (down_interruptible(&dev->sem))
        return -ERESTARTSYS;

    /* find listitem, qset index and offset in the quantum */
    item = (long)*f_pos / itemsize;
    rest = (long)*f_pos % itemsize;
    s_pos = rest / quantum; q_pos = rest % quantum;

    /* follow the list up to the right position */
    dptr = scull_follow(dev, item);
    if (dptr == NULL)
        goto out;
    if (!dptr->data) {
        dptr->data = kmalloc(qset * sizeof(char *), GFP_KERNEL);
        if (!dptr->data)
            goto out;
        memset(dptr->data, 0, qset * sizeof(char *));
    }
    if (!dptr->data[s_pos]) {
        dptr->data[s_pos] = kmalloc(quantum, GFP_KERNEL);
        if (!dptr->data[s_pos])
            goto out;
    }
    /* write only up to the end of this quantum */
    if (count > quantum - q_pos)
        count = quantum - q_pos;

    if (copy_from_user(dptr->data[s_pos]+q_pos, buf, count)) {
        retval = -EFAULT;
        goto out;
    }
    *f_pos += count;
    retval = count;

    /* update the size */
    if (dev->size < *f_pos)
        dev->size = *f_pos;

out:
    up(&dev->sem);
    return retval;
}

```

- scull_write() 함수 정의

```

/* ioctl */
long scull_ioctl(struct file *filp, unsigned int cmd, unsigned long arg) {
    int err = 0, tmp;
    int retval = 0;

    /*
     * extract the type and number bitfields, and don't decode
     * wrong cmds: return ENOTTY (inappropriate ioctl) before access_ok()
     */
    if(_IOC_TYPE(cmd) != SCULL_IOC_MAGIC) return -ENOTTY;
    if(_IOC_NR(cmd) > SCULL_IOC_MAXNR) return -ENOTTY;

    /*
     * the direction is a bitmask, and VERIFY_WRITE catches R/W
     * transfers. 'Type' is user-oriented, while
     * access_ok is kernel-oriented, so the concept of "read" and
     * "write" is reversed
     */
    if(_IOC_DIR(cmd) & _IOC_READ)
        err = !access_ok_wrapper(VERIFY_WRITE, (void __user *)arg, _IOC_SIZE(cmd));
    else if(_IOC_DIR(cmd) & _IOC_WRITE)
        err = !access_ok_wrapper(VERIFY_READ, (void __user *)arg, _IOC_SIZE(cmd));
    if(err) return -EFAULT;

    switch(cmd) {
        case SCULL_IOCRESET:
            scull_quantum = SCULL_QUANTUM;
            scull_qset = SCULL_QSET;
            break;

        case SCULL_IOCSQUANTUM: /* Set: arg points to the value */
            if(!capable(CAP_SYS_ADMIN))
                return -EPERM;
            retval = __get_user(scull_quantum, (int __user *)arg);
            break;

        case SCULL_IOWTQUANTUM: /* Tell: arg is the value */
            if(!capable(CAP_SYS_ADMIN))
                return -EPERM;
            scull_quantum = arg;
            break;

        case SCULL_IOCGQUANTUM: /* Get: arg is pointer to result */

            case SCULL_IOCGQUANTUM: /* Get: arg is pointer to result */
                retval = __put_user(scull_quantum, (int __user *)arg);
                break;

            case SCULL_IOWCQUANTUM: /* Query: return it (it's positive) */
                return scull_quantum;

            case SCULL_IOWXQUANTUM: /* eXchange: use arg as pointer */
                if (! capable (CAP_SYS_ADMIN))
                    return -EPERM;
                tmp = scull_quantum;
                retval = __get_user(scull_quantum, (int __user *)arg);
                if (retval == 0)
                    retval = __put_user(tmp, (int __user *)arg);
                break;

            case SCULL_IOWHQUANTUM: /* sHift: like Tell + Query */
                if (! capable (CAP_SYS_ADMIN))
                    return -EPERM;
                tmp = scull_quantum;
                scull_quantum = arg;
                return tmp;

            default:           /* redundant, as cmd was checked against MAXNR */
                return -ENOTTY;
    }
    return retval;
}

```

- scull_ioctl() 함수 정의

```
// Change the current read/write position in a file
// New position is returned as a positive return value
loff_t scull_llseek(struct file *filp, loff_t off, int whence) {
    struct scull_dev *dev = filp->private_data;
    loff_t newpos;

    switch(whence) {
        case 0: // SEEK_SET
            newpos = off;
            break;
        case 1: // SEEK_CUR
            newpos = filp->f_pos + off;
            break;
        case 2: // SEEK_END
            newpos = dev->size + off;
            break;

        default: // can't happen
            return -EINVAL;
    }

    if(newpos < 0) return -EINVAL;
    filp->f_pos = newpos;
    return newpos;
}

// Set file operations
struct file_operations scull_fops = {
    .owner = THIS_MODULE,
    .llseek = scull_llseek,
    .read = scull_read,
    .write = scull_write,
    .open = scull_open,
    .release = scull_release,
    .unlocked_ioctl = scull_ioctl
};
```

- scull_llseek() 함수 정의
 - 디바이스 드라이버의 파일 포인터 위치를 강제로 이동시켜 현재의 read, write position(어디까지 읽었는지를 나타내는 f_pos)을 변경
 - 새로운 position을 양수 값으로 return
- file_operations 구조체 및 필드 작성
 - device driver와 응용 프로그램을 연결하는 고리
 - linux/fs.h 파일에서 정의하며, 함수 포인터의 집합
 - 지정하지 않으면 NULL로 남겨둠
- ioctl을 사용하기 위해 CPU에서 모든 프로세스를 lock하던 예전 방식과는 달리, unlocked_ioctl을 사용하여 함수가 개별적으로 lock을 걸 수 있게 되었다.

```

/* Cleanup module */
void scull_cleanup_module(void) {
    int i;
    dev_t devno = MKDEV(scull_major, scull_minor);

    /* Get rid of our char dev entries */
    if (scull_devices) {
        for (i = 0; i < scull_num_dev; i++) {
            scull_trim(scull_devices + i);
            cdev_del(&scull_devices[i].cdev);
        }
        kfree(scull_devices);
    }

    /* Unregister */
    /* cleanup_module is never called if registering failed */
    unregister_chrdev_region(devno, scull_num_dev);
}

/* Set cdev */
static void scull_setup_cdev(struct scull_dev *dev, int index) {
    int err, devno;

    devno = MKDEV(scull_major, scull_minor + index);
    cdev_init(&dev->cdev, &scull_fops);
    dev->cdev.owner = THIS_MODULE;
    dev->cdev.ops = &scull_fops;
    err = cdev_add (&dev->cdev, devno, 1);

    /* Fail gracefully if need be */
    if (err)
        printk(KERN_NOTICE "Error %d adding scull%d", err, index);
}

```

- module unload 시 cleanup 실행
 - 디바이스 개수만큼 돌며 scull_trim() 함수 실행 및 scull_devices의 cdev구조체 제거
 - kfree()로 사용하지 않는 scull_devices의 메모리 반환
- 아래의 init module 시 cdev 구조체 셋업 및 초기화
 - cdev 구조체의 owner, ops 셋팅

```

/* Initialize module */
int scull_init_module(void) {
    int result, i;
    dev_t dev = 0;

    if (scull_major) {
        dev = MKDEV(scull_major, scull_minor);
        /* Register */
        result = register_chrdev_region(dev, scull_num_dev, "scull");
    } else {
        result = alloc_chrdev_region(&dev, scull_minor, scull_num_dev, "scull");
        scull_major = MAJOR(dev);
    }
    if (result < 0) {
        printk(KERN_WARNING "scull: can't get major %d\n", scull_major);
        return result;
    }

    /* allocate the devices, we can't have them static, as the number
     * can be specified at load time
     */
    scull_devices = kmalloc(scull_num_dev * sizeof(struct scull_dev), GFP_KERNEL);
    if (!scull_devices) {
        result = -ENOMEM;
        goto fail;      /* Make this more graceful */
    }
    memset(scull_devices, 0, scull_num_dev * sizeof(struct scull_dev));

    /* Initialize each device. */
    for (i = 0; i < scull_num_dev; i++) {
        scull_devices[i].quantum = scull_quantum;
        scull_devices[i].qset = scull_qset;
        sema_init(&scull_devices[i].sem, 1);
        scull_setup_cdev(&scull_devices[i], i);
    }

    return 0;      /* succeed */

fail:
    scull_cleanup_module();
    return result;
}

```

- Major number 및 minor number를 할당받아 등록하고, 디바이스 개수(scull_num_dev) * scull_dev 만큼의 메모리 크기를 동적 할당(kmalloc)
- memset으로 가비지 값 제거
 - 메모리 크기를 변경할 포인터: scull_devices
 - 초기화 값: 0
 - 초기화 크기 반환 값: 디바이스 개수 scull_num_dev * scull_dev
- sema_init으로 세마포어 초기화

```

module_init(scull_init_module);
module_exit(scull_cleanup_module);

```

- init은 module이 load될 때, exit는 module이 unload될 때 실행한다.

2. 소스코드 컴파일

Make

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo make
[sudo] password for seongbinyoon:
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/advchardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/advchardd/scull.o
  MODPOST /home/seongbinyoon/EdgeClab/advchardd/Module.symvers
  CC [M]  /home/seongbinyoon/EdgeClab/advchardd/scull.mod.o
  LD [M]  /home/seongbinyoon/EdgeClab/advchardd/scull.ko
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$
```

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ ls
access_ok_version.h  modules.order    scull.c      scull.ko      scull.mod      scull.mod.o   scull_unload
Makefile              Module.symvers  scull.h      scull_load   scull.mod.c   scull.o      test.c
```

- make 명령어를 통해 작성한 모듈 소스코드를 컴파일한다. (아래에서 설명할 ioctl test 과정에서 ioctl.h 파일을 생성하므로 여기까지 진행했을 때 include 오류가 발생할 수 있다.)

3. 모듈 적재

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo sh scull_load
```

```
$ sudo sh scull_load
```

- 관리자 권한을 통해 scull_load 스크립트를 실행
- scull_load 를 통해 이전 등록된 /dev 폴더에 있던 scull device 파일을 제거한 뒤, 새롭게 device driver 를 적재하고 등록한다.

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ lsmod | grep scull
scull           16384  0
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ ls -al /dev | grep scull
crw-rw-r--  1 root      staff  509,     0  2월  3 20:32 scull0
crw-rw-r--  1 root      staff  509,     1  2월  3 20:32 scull1
crw-rw-r--  1 root      staff  509,     2  2월  3 20:32 scull2
crw-rw-r--  1 root      staff  509,     3  2월  3 20:32 scull3
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$
```

```
$ lsmod | grep scull
```

- scull 모듈이 성공적으로 적재가 되었는지에 대해 확인할 수 있음.

```
$ ls -al /dev | grep scull
```

- scull 디바이스가 성공적으로 등록되어 있는지에 대해 확인할 수 있음.
- ls -al 명령어를 통해 확인해보았을 때 가장 먼저 'c' 가 뜨는 것을 바탕으로 char device 등록에 성공했다는 것을 알 수 있음.

4. 모듈 제거

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo sh scull_unload  
[sudo] password for seongbinyoon:
```

```
$ sudo sh scull_unload
```

- 위 명령어를 통해 scull 디바이스에 대한 모듈을 unload 하고 등록된 device 파일을 제거한다.

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ lsmod | grep scull  
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ ls -al /dev | grep 'scull'
```

```
$ lsmod | grep scull  
$ ls -al /dev | grep 'scull'
```

- 위 명령어를 통해 이전에 등록했던 모듈과 디바이스 파일이 제거되었음을 확인할 수 있다.

+) Advanced Char Device 테스트 실습

ioctl.h

```

/* ioctl struct */
struct ioctl_info {
    unsigned long size;
    char buf[128];
};

/*
 * ioctl definitions
 */
/* Use 'k' as magic number */
#define SCULL_IOC_MAGIC 'k'
/* Please use a different 8-bit number in your code */

#define SET_DATA      _IOW(SCULL_IOCTL_MAGIC, 2, struct ioctl_info)
#define GET_DATA      _IOR(SCULL_IOCTL_MAGIC, 3, struct ioctl_info)
#define SCULL_IOCRESET _IO(SCULL_IOC_MAGIC, 0)

/*
 * S means "Set" through a ptr,
 * T means "Tell" directly with the argument value
 * G means "Get": reply by setting through a pointer
 * Q means "Query": response is on the return value
 * X means "Exchange": switch G and S atomically
 * H means "shlft": switch T and Q atomically
 */
#define SCULL_IOCSQUANTUM _IOW(SCULL_IOC_MAGIC, 1, int)
#define SCULL_IOCSET     _IOR(SCULL_IOC_MAGIC, 2, int)
#define SCULL_IOTQUANTUM _IO(SCULL_IOC_MAGIC, 3)
#define SCULL_IOTCSET    _IO(SCULL_IOC_MAGIC, 4)
#define SCULL_IOCQQUANTUM _IOR(SCULL_IOC_MAGIC, 5, int)
#define SCULL_IOCQSET    _IOR(SCULL_IOC_MAGIC, 6, int)
#define SCULL_IOCQQUANTUM _IO(SCULL_IOC_MAGIC, 7)
#define SCULL_IOCQSET    _IO(SCULL_IOC_MAGIC, 8)
#define SCULL_IOCQXQUANTUM _IOWR(SCULL_IOC_MAGIC, 9, int)
#define SCULL_IOCQXSET   _IOWR(SCULL_IOC_MAGIC, 10, int)
#define SCULL_IOCQHQUANTUM _IO(SCULL_IOC_MAGIC, 11)
#define SCULL_IOCQHSET   _IO(SCULL_IOC_MAGIC, 12)

#define SCULL_IOC_MAXNR 14

```

- ioctl_info 와 SCULL_IOCSQUANTUM 과 같은 오퍼레이션 정보들은 User Space 의 응용 프로그램에서 또한 적용해야 하기 때문에 따로 정리하여 ioctl.h 헤더파일을 만들 수 있었다.

test.c

```

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/ioctl.h>
#include "ioctl.h"

int main() {
    int fd;
    int set_data = 2023;
    int get_data = -1; /* If success, it will be changed to 2023. */

    if ((fd = open("/dev/scull0", O_RDWR)) < 0) {
        printf("Cannot open /dev/scull0. Try again later.\n");
        return -ENXIO;
    }

    /* User Space -> Kernel Space */
    if (ioctl(fd, SCULL_IOCSQUANTUM, &set_data) < 0) {
        printf("Error in SCULL_IOTQUANTUM statement.\n");
        return -EPERM;
    }

    /* Kernel Space -> User Space */
    if (ioctl(fd, SCULL_IOCQQUANTUM, &get_data) < 0) {
        printf("Error in SCULL_IOCQQUANTUM statement.\n");
        return -EPERM;
    }

    printf("get_data : %d\n", get_data);

    if (close(fd) != 0) {
        printf("Cannot close.\n");
        return -ENXIO;
    }

    return 0;
}

```

43,1-8 80%

- ioctl 함수가 제대로 수행되는지 확인하기 위한 테스트 파일 test.c이다.
- scull0 디바이스 파일 디스크립터를 바탕으로 ioctl 함수를 호출한다.

- SCULL_IOCSQUANTUM (Set) 옵션을 통해 User Space에서 Kernel Space로 set_data 변수 데이터 (2023)를 전송하여 scull_quantum 변수에 저장한다.
- 그 이후, SCULL_IOCGQUANTUM (Get) 옵션을 통해 Kernel Space에서 User Space로 저장된 scull_quantum 변수 데이터를 전송하여 get_data 변수에 저장하고 출력한다.
- integer 데이터 이외에 구조체와 같이 정해진 규격의 데이터를 송수신하고 싶다면 struct 구조체를 바탕으로 copy_from_user() (User Space → Kernel Space) 혹은 copy_to_user() (Kernel Space → User Space) 함수를 바탕으로 구현할 수 있다.

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ gcc test.c
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ ls
access_ok_version.h    ioctl.h      modules.order   scull.c      scull.ko      scull.mod      scull.mod.o      scull_unload
a.out                  Makefile     Module.symvers scull.h      scull_load   scull.mod.c   scull.o      test.c
```

```
~$ gcc test.c
```

- 위 명령어를 통해 test.c 파일을 컴파일한다.
- ls 명령어를 입력하면 a.out 파일이 생성된 것을 확인할 수 있다.

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo ./a.out
get_data : 2023
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ █
```

```
~$ sudo ./a.out
```

- 이후 sudo ./a.out 명령어를 바탕으로 테스트해볼 수 있다. (sudo 붙이지 않으면 정상적으로 실행이 되지 않는다.)
- 정상적으로 Kernel Space 에 있는 scull_quantum 변수의 데이터인 “2023” 를 가져와 출력하는 데에 성공하였다.

Trouble Shooting

Trouble 1

```
/home/seongbinyoon/EdgeClab/advchardd/scull.c: In function 'scull_ioctl':
/home/seongbinyoon/EdgeClab/advchardd/scull.c:234:82: error: macro "access_ok" passed 3 arguments, but takes just 2
  234 |     err = !access_ok(VERIFY WRITE, (void __user *)arg, _IOC_SIZE(cmd));
                  ^
In file included from ./include/linux/uaccess.h:11,
                 from ./include/linux/sched/task.h:11,
                 from ./include/linux/sched/signal.h:9,
                 from ./include/linux/rcuwait.h:6,
                 from ./include/linux/percpu-rwsem.h:7,
                 from ./include/linux/fs.h:33,
                 from /home/seongbinyoon/EdgeClab/advchardd/scull.c:6:
./arch/x86/include/asm/uaccess.h:41: note: macro "access_ok" defined here
  41 | #define access_ok(addr, size) \
                  \
/home/seongbinyoon/EdgeClab/advchardd/scull.c:234:24: error: 'access_ok' undeclared (first use in this function)
  234 |     err = !access_ok(VERIFY WRITE, (void __user *)arg, _IOC_SIZE(cmd));
                  ^
/home/seongbinyoon/EdgeClab/advchardd/scull.c:234:24: note: each undeclared identifier is reported only once for each function it appears in
/home/seongbinyoon/EdgeClab/advchardd/scull.c:236:81: error: macro "access_ok" passed 3 arguments, but takes just 2
  236 |     err = !access_ok(VERIFY_READ, (void __user *)arg, _IOC_SIZE(cmd));
                  ^
In file included from ./include/linux/uaccess.h:11,
                 from ./include/linux/sched/task.h:11,
                 from ./include/linux/sched/signal.h:9,
                 from ./include/linux/rcuwait.h:6,
                 from ./include/linux/percpu-rwsem.h:7,
                 from ./include/linux/fs.h:33,
                 from /home/seongbinyoon/EdgeClab/advchardd/scull.c:6:
./arch/x86/include/asm/uaccess.h:41: note: macro "access_ok" defined here
  41 | #define access_ok(addr, size) \
                  \
/home/seongbinyoon/EdgeClab/advchardd/scull.c: At top level:
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:10: error: 'struct file_operations' has no member named 'ioctl'
  323 |     .ioctl = scull_ioctl,
                  ^
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:18: error: positional initialization of field in 'struct' declared with 'designated_init' attribute [-Werror=designated-init]
  323 |         .ioctl = scull_ioctl,
                  ^
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:18: note: (near initialization for 'scull_fops')
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:18: error: initialization of 'ssize_t (*)(struct kiocb *, struct iov_iter *)' {aka 'long int (*)(struct kiocb *, struct iov_iter *)'} from incompatible pointer type 'long int (*)(struct file *, unsigned int, long unsigned int)' [-Werror=incompatible-pointer-types]
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:18: note: (near initialization for 'scull_fops.read_iter')
cc1: some warnings being treated as errors
```

```
~$ sudo make
```

- 컴파일 중에 위와 같은 오류들이 발생했다.

Solution 1

access_ok_version.h

```
/*
 * Header file access_ok_version.h
 */
#include <linux/version.h>

#if LINUX_VERSION_CODE < KERNEL_VERSION(5,0,0)
#define access_ok_wrapper(type,arg,cmd) \
    access_ok(type, arg, cmd)

#else
#define access_ok_wrapper(type,arg,cmd) \
    access_ok(arg, cmd)
#endif
~
```

scull.c

```
#include "access_ok_version.h"
```

```
/* ioctl */
long scull_ioctl(struct file *filp, unsigned int cmd, unsigned long arg) {
    int err = 0, tmp;
    int retval = 0;

    /*
     * extract the type and number bitfields, and don't decode
     * wrong cmds: return ENOTTY (inappropriate ioctl) before access_ok()
     */
    if (_IOC_TYPE(cmd) != SCULL_IOC_MAGIC) return -ENOTTY;
    if (_IOC_NR(cmd) > SCULL_IOC_MAXNR) return -ENOTTY;

    /*
     * the direction is a bitmask, and VERIFY_WRITE catches R/W
     * transfers. 'Type' is user-oriented, while
     * access_ok is kernel-oriented, so the concept of "read" and
     * "write" is reversed
     */
    if (_IOC_DIR(cmd) & _IOC_READ)
        err = !access_ok_wrapper(VERIFY_WRITE, (void __user *)arg, _IOC_SIZE(cmd));
    else if (_IOC_DIR(cmd) & _IOC_WRITE)
        err = !access_ok_wrapper(VERIFY_READ, (void __user *)arg, _IOC_SIZE(cmd));
    if (err) return -EFAULT;
```

```
#include "access_ok_version.h"
```

- access_ok에 관한 define이 되어있지 않아 발생한 오류를 access_ok_version.h 헤더파일을 생성함으로써 해결하였다.
- scull.c 파일에서 access_ok → access_ok_wrapper로 수정

참조:

l3d3/scull at master · martinezjavier/l3d3

You can't perform that action at this time. You signed in with another tab or window.
You signed out in another tab or window. Reload to refresh your session. Reload to
refresh your session.

<https://github.com/martinezjavier/l3d3/tree/master/scull>

martinezjavier/l3d3

Linux Device Drivers 3 examples updated to work in
recent kernels



At 22 Contributors

6 Issues

2k Stars

821 Forks

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo make
```

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo make
[sudo] password for seongbinyoon:
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/advchardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/advchardd/scull.o
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:10: error: 'struct file_operations' has no member named 'ioctl'
  323 |     .ioctl = scull_ioctl,
      |     ^
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:18: error: positional initialization of field in 'struct' declared with 'designated_init' attribute [-Werror=designated-init]
  323 |     .ioctl = scull_ioctl,
      |     ^
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:18: note: (near initialization for 'scull_fops')
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:18: error: initialization of 'ssize_t (*)(struct kiocb *, struct iov_iter *)' {aka 'long int (*)(struct kiocb *, struct iov_iter *)'} from incompatible pointer type 'long int (*)(struct file *, unsigned int, long unsigned int)' [-Werror=incompatible-pointer-types]
/home/seongbinyoon/EdgeClab/advchardd/scull.c:323:18: note: (near initialization for 'scull_fops.read_iter')
cc1: some warnings being treated as errors
make[2]: *** [scripts/Makefile.build:250: /home/seongbinyoon/EdgeClab/advchardd/scull.o] Error 1
make[1]: *** [Makefile:1992: /home/seongbinyoon/EdgeClab/advchardd] Error 2
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
make: *** [Makefile:11: default] Error 2

```

~\$ sudo make

- 이 후 다시 컴파일 진행하였고, error의 수가 2개로 줄어든 것을 확인할 수 있다.

```

/* Set file operations */
struct file_operations scull_fops = {
    .owner = THIS_MODULE,
    .llseek = scull_llseek,
    .read = scull_read,
    .write = scull_write,
    .open = scull_open,
    .release = scull_release
};

```

.ioctl = scull_ioctl

- 위 코드를 삭제하였다.

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo make
[sudo] password for seongbinyoon:
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/advchardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/advchardd/scull.o
  MODPOST /home/seongbinyoon/EdgeClab/advchardd/Module.symvers
  CC [M]  /home/seongbinyoon/EdgeClab/advchardd/scull.mod.o
  LD [M]  /home/seongbinyoon/EdgeClab/advchardd/scull.ko
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ 

```

```
~$ sudo make
```

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo sh scull_load
```

```
~$ sudo sh scull_load
```

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ lsmod | grep scull
scull           16384  0
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ ls -al /dev | grep scull
crw-rw-r--  1 root      staff  509,    0  2월  3 20:32 scull0
crw-rw-r--  1 root      staff  509,    1  2월  3 20:32 scull1
crw-rw-r--  1 root      staff  509,    2  2월  3 20:32 scull2
crw-rw-r--  1 root      staff  509,    3  2월  3 20:32 scull3
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$
```

```
~$ lsmod | grep scull
```

```
~$ ls -al /dev | grep scull
```

- 이 후, 다시 컴파일을 시도, 커널에 적재하였다.

Trouble 2

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeLab/advchardd$ gcc test.c
In file included from test.c:12:
scull.h:58:26: error: field 'sem' has incomplete type
  58 |     struct semaphore sem;           /* mutual exclusion semaphore */
      |             ^
scull.h:59:21: error: field 'cdev' has incomplete type
  59 |     struct cdev cdev;            /* Char device structure */
      |             ^
scull.h:72:44: warning: 'struct file' declared inside parameter list will not be visible outside of this definition or declaration
  72 | int scull_open(struct inode *inode, struct file *filp);
      |             ^
scull.h:72:23: warning: 'struct inode' declared inside parameter list will not be visible outside of this definition or declaration
  72 | int scull_open(struct inode *inode, struct file *filp);
      |             ^
scull.h:73:47: warning: 'struct file' declared inside parameter list will not be visible outside of this definition or declaration
  73 | int scull_release(struct inode *inode, struct file *filp);
      |             ^
scull.h:73:26: warning: 'struct inode' declared inside parameter list will not be visible outside of this definition or declaration
  73 | int scull_release(struct inode *inode, struct file *filp);
      |             ^
scull.h:75:51: error: expected ';' or ')' before '*' token
  75 | ssize_t scull_read(struct file *filp, char __user *buf, size_t count, loff_t *f_pos);
      |             ^
scull.h:76:58: error: expected ';' or ')' before '*' token
  76 | ssize_t scull_write(struct file *filp, const char __user *buf, size_t count, loff_t *f_pos);
      |             ^
scull.h:77:28: warning: 'struct file' declared inside parameter list will not be visible outside of this definition or declaration
  77 | loff_t scull_llseek(struct file *filp, loff_t off, int whence);
      |             ^
scull.h:78:25: warning: 'struct file' declared inside parameter list will not be visible outside of this definition or declaration
  78 | long scull_ioctl(struct file *filp, unsigned int cmd, unsigned long arg);
      |             ^

```

```
~$ gcc test.c
```

- advanced character device driver를 작성하고 테스트코드를 통해 ioctl이 제대로 동작하는지 확인하기 위한 테스트코드 컴파일 과정에서 문제가 발생하였다.
- 오류 메세지로 봐서는 scull.h 헤더파일에서 정의된 무언가가 문제인 것 같다.

scull.h

```

/* Define scull devices */
struct scull_dev {
    struct scull_qset *data;          /* Pointer to first quantum set */
    int quantum;                     /* the current quantum size */
    int qset;                        /* the current array size */
    unsigned long size;              /* amount of data stored here */
    unsigned int access_key;         /* used by sculluid and scullpriv */
    struct semaphore sem;            /* mutual exclusion semaphore */
    struct cdev cdev;                /* Char device structure */
};

```

```

/* Define scull devices */
struct scull_dev {
    struct scull_qset *data;          /* Pointer to first quantum set */
    int quantum;                     /* the current quantum size */
    int qset;                        /* the current array size */
    unsigned long size;              /* amount of data stored here */
    unsigned int access_key;          /* used by sculluid and scullpriv */
    struct mutex lock;               /* mutual exclusion semaphore */
    struct cdev cdev;                /* Char device structure */
};

```

scull.c(scull_open, scull_read, scull_write)

```
out:  
    up(&dev->sem);  
    return retval;  
}
```

```
out:  
    // up(&dev->sem);  
    mutex_unlock(&dev->lock);  
    return retval;  
}
```

```
if (down_interruptible(&dev->sem))  
    return -ERESTARTSYS;
```

```
if (mutex_lock_interruptible(&dev->lock))  
    return -ERESTARTSYS;  
if (*f_pos >= dev->size)  
    goto out;  
if (*f_pos + count > dev->size)  
    count = dev->size - *f_pos;
```

```
/* Initialize each device. */  
for (i = 0; i < scull_num_dev; i++) {  
    scull_devices[i].quantum = scull_quantum;  
    scull_devices[i].qset = scull_qset;  
    sema_init(&scull_devices[i].sem, 1);  
    scull_setup_cdev(&scull_devices[i], i);  
}
```

```
/* Initialize each device. */  
for (i = 0; i < scull_num_dev; i++) {  
    scull_devices[i].quantum = scull_quantum;  
    scull_devices[i].qset = scull_qset;  
    mutex_init(&scull_devices[i].lock);  
    scull_setup_cdev(&scull_devices[i], i);  
}
```

- semaphore sem에 대한 incomplete type 오류를 잡기 위해 각 파일을 mutex lock을 사용하였다. 하지만 결과는 동일했다.
- mutex란 mutual exclusion으로, 상호 배제이다. 즉, 쓰레드에서 제공하는 동기화 메커니즘으로, 공유 자원 공간에 대한 접근 시간 제어로 동기화를 달성한다.

- 세마포어 대신 사용할 수 있기 때문에 해당 오류와는 상관이 없다.

```
error: field has incomplete type
```

- 해당 오류를 찾아보니 전방선언 (forward declaration)이거나 헤더파일 참조가 잘못되었기 때문에 나타난 것으로 보인다.

Solution 2

- 해결 방법은 다음과 같다.
 - ioctl_info 구조체와 ioctl 정의를 scull.h 헤더파일이 아닌, 따로 ioctl.h 헤더파일에 담는다.
 - scull.c 파일에 scull.h와 ioctl.h 헤더파일을 include 한다.
 - 테스트 파일 test.c 파일에 ioctl.h 헤더파일을 include 한다.

ioctl.h

```
/* ioctl struct */
struct ioctl_info {
    unsigned long size;
    char buf[128];
};

/*
 * IOCTL definitions
 */

/* Use 'k' as magic number */
#define SCULL_IOC_MAGIC 'k'
/* Please use a different 8-bit number in your code */

#define SET_DATA      _IOW(SCULL_IOC_MAGIC, 2, struct ioctl_info)
#define GET_DATA      _IOR(SCULL_IOC_MAGIC, 3, struct ioctl_info)
#define SCULL_INCRESET _IO(SCULL_IOC_MAGIC, 0)

/*
 * S means "Set" through a ptr,
 * T means "Tell" directly with the argument value
 * G means "Get": reply by setting through a pointer
 * Q means "Query": response is on the return value
 * X means "exchange": switch G and S atomically
 * H means "shlft": switch T and Q atomically
 */
#define SCULL_IOCSQUANTUM _IOW(SCULL_IOC_MAGIC, 1, int)
#define SCULL_IOCSET      _IOW(SCULL_IOC_MAGIC, 2, int)
#define SCULL_IOTQUANTUM _IO(SCULL_IOC_MAGIC, 3)
#define SCULL_IOCQSET     _IO(SCULL_IOC_MAGIC, 4)
#define SCULL_IOCQUANTUM _IOR(SCULL_IOC_MAGIC, 5, int)
#define SCULL_IOCQSET     _IOR(SCULL_IOC_MAGIC, 6, int)
#define SCULL_IOCQQUANTUM _IO(SCULL_IOC_MAGIC, 7)
#define SCULL_IOCQSET     _IO(SCULL_IOC_MAGIC, 8)
#define SCULL_IOCXXQUANTUM _IOWR(SCULL_IOC_MAGIC, 9, int)
#define SCULL_IOCXXSET    _IOWR(SCULL_IOC_MAGIC, 10, int)
#define SCULL_IOCQHQUANTUM _IO(SCULL_IOC_MAGIC, 11)
#define SCULL_IOCQHSET    _IO(SCULL_IOC_MAGIC, 12)

#define SCULL_IOC_MAXNR 14
```

scull.c

```
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/slab.h>
#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/types.h>
#include <linux/fcntl.h>
#include <linux/cdev.h>
#include <linux/proc_fs.h>
#include <linux/seq_file.h>
#include <asm/uaccess.h>
#include "scull.h"
#include "ioctl.h"
#include "access_ok_version.h"
```

test.c

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/ioctl.h>
#include "ioctl.h"

int main() {
    int fd;
    int set_data = 2023; /* If success, it will be changed to 2023. */
    int get_data = -1;

    if ((fd = open("/dev/scull0", O_RDWR)) < 0) {
        printf("Cannot open /dev/scull0. Try again later.\n");
        return -ENXIO;
    }

    /* User Space -> Kernel Space */
    if (ioctl(fd, SCULL_IOCSETQUANTUM, &set_data) < 0) {
        printf("Error in SCULL_IOCSETQUANTUM statement.\n");
        return -EPERM;
    }

    /* Kernel Space -> User Space */
    if (ioctl(fd, SCULL_IOCGETQUANTUM, &get_data) < 0) {
        printf("Error in SCULL_IOCGETQUANTUM statement.\n");
        return -EPERM;
    }

    printf("get_data : %d\n", get_data);

    if (close(fd) != 0) {
        printf("Cannot close.\n");
        return -ENXIO;
    }

    return 0;
}
```

43,1-8

80%

재컴파일 후 테스트

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo make
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/advchardd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M] /home/seongbinyoon/EdgeClab/advchardd/scull.o
  MODPOST /home/seongbinyoon/EdgeClab/advchardd/Module.symvers
  CC [M] /home/seongbinyoon/EdgeClab/advchardd/scull.mod.o
  LD [M] /home/seongbinyoon/EdgeClab/advchardd/scull.ko
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo sh scull_load
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ ls -al /dev | grep 'scull'
crw-rw-r-- 1 root      staff 509,   0 2월 11 00:44 scull0
crw-rw-r-- 1 root      staff 509,   1 2월 11 00:44 scull1
crw-rw-r-- 1 root      staff 509,   2 2월 11 00:44 scull2
crw-rw-r-- 1 root      staff 509,   3 2월 11 00:44 scull3
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ gcc test.c
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ ls
access_ok_version.h ioctl.h modules.order scull.c scull.ko scull.mod scull.mod.o scull_unload
a.out           Makefile Module.symvers scull.h scull_load scull.mod.c scull.o test.c
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo ./a.out
get_data : 2023
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$
```

- 모듈을 다시 컴파일 후 커널 적재 및 test 파일 컴파일을 진행하였다.

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ gcc test.c
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ ls
access_ok_version.h ioctl.h modules.order scull.c scull.ko scull.mod scull.mod.o scull_unload
a.out           Makefile Module.symvers scull.h scull_load scull.mod.c scull.o test.c
```

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$ sudo ./a.out
get_data : 2023
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/advchardd$
```

- 위와 같이 a.out 파일이 생성되며 컴파일이 성공하였고, 정상적으로 Kernel Space에 있는 scull_quantum 변수의 데이터인 2023이 출력되었다.

Block Device Driver

Block Device Driver 이란?

- Block Device Driver는 Char Device Driver와 비슷하게 입출력 장치에 대해 함수를 제공해주는 프로그램을 말한다.
- 하지만, device를 파일처럼 접근하여 stream 방식으로 read/write하는 Char Device Driver과는 상반되게, Block Device Driver는 disk와 같은 file system을 기반으로 block 단위로 데이터를 읽고 쓴다.
- 지금부터 Block Device Driver의 구분 방법부터 구현 후 테스트까지 아래 과정을 통해 진행하려고 한다.

Block Device 이란?

- 블록 단위의 입출력을 바탕으로 한 장치
- 버퍼 캐쉬를 바탕으로 내부 장치를 표현
- 파일 시스템 (File System)에 의해 마운트 (Mount)되어 관리된다.
- Block Device 의 예시로는 하드 디스크, 플로피 디스크 등이 있음.

Block Device 구분 방법

- 이전에 Char Device Driver 를 인식하는 방법과 매우 유사하다.
- ls -al /dev/ 명령어를 바탕으로 나열된 파일 목록에서 맨 앞의 식별자를 확인하면 됨.
- 가장 앞에 'b' 로 되어 있다면 Block Device 라고 판단할 수 있음.
- 위 방식을 바탕으로 아래 코드를 바탕으로 등록된 Block Device 를 확인하려고 함.

Block Device Driver 예시 코드

Makefile

```

ifneq ($(KERNELRELEASE),)
obj-m := sbull.o
else
    KERNELDIR ?= /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

default:
    $(MAKE) -C $(KERNELDIR) M=$(PWD) modules

endif

```

- 작성된 C 코드를 빌드하여 .ko (kernel object) 파일을 생성한다.

sbull_load

```

#!/bin/bash

function make_minors {
    let part=1
    while (( $part < $minors )); do
        let minor=$part+$2
        mknod $1$part b $major $minor
        let part=$part+1
    done
}

# FIXME: This isn't handling minors (partitions) at all.
module="sbull"

```

```

device="sbull"
mode="664"
chardevice="sbullr"
minors=16

# Group: since distributions do it differently, look for wheel or use staff
if grep '^staff:' /etc/group > /dev/null; then
    group="staff"
else
    group="wheel"
fi

# invoke insmod with all arguments we got
# and use a pathname, as newer modutils don't look in . by default
insmod ./${module}.ko $* || exit 1

major=`cat /proc/devices | awk "\$2==\"$module\" {print \$1}"` 

# Remove stale nodes and replace them, then give gid and perms
rm -f /dev/${device}[a-d]* /dev/${device}

mknod /dev/${device}a b $major 0
make_minors /dev/${device}a 0
mknod /dev/${device}b b $major 16
make_minors /dev/${device}b 16
mknod /dev/${device}c b $major 32
make_minors /dev/${device}c 32
mknod /dev/${device}d b $major 48
make_minors /dev/${device}d 48
ln -sf ${device}a /dev/${device}
chgrp $group /dev/${device}[a-d]*
chmod $mode /dev/${device}[a-d]*

```

- insmod 를 통해 .ko (kernel object) 파일을 커널 모듈에 적재한다.
- sbull 에 대한 block device 의 디바이스 파일을 mknod 함수를 통해 생성한다.
- sbulla, sbullb, sbullc, sbulld 에 대해 각 16개의 minor number 를 할당한다.
- ln -sf 함수를 통해 /dev/sbulla 파일에 대한 심볼릭 링크를 /dev/sbll 파일로 설정한다.

sbull_unload

```

#!/bin/sh
module="sbull"
device="sbull"

# invoke rmmod with all arguments we got
rmmod $module $* || exit 1

# Remove stale nodes
rm -f /dev/${device}[a-d]* /dev/${device}

```

- 적재된 모듈을 rmmod 명령어를 통해 제거한다.
- 또한, 등록된 디바이스 파일을 rm -f (force) 명령어로 제거한다.

sbull.c

```

#include <linux/version.h>      /* LINUX_VERSION_CODE */
#include <linux/blk-mq.h>

#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/init.h>

#include <linux/sched.h>
#include <linux/kernel.h>        /* printk() */
#include <linux/slab.h>          /* kmalloc() */
#include <linux/fs.h>            /* everything... */
#include <linux/errno.h>          /* error codes */
#include <linux/timer.h>
#include <linux/types.h>          /* size_t */
#include <linux/fcntl.h>          /* O_ACCMODE */
#include <linux/hdreg.h>          /* HDIO_GETGEO */
#include <linux/kdev_t.h>
#include <linux/vmalloc.h>
// #include <linux/genhd.h>      We don't have genhd.h
#include <linux/blkdev.h>
#include <linux/buffer_head.h>    /* invalidate_bdev */
#include <linux/bio.h>

/* For linux kernel 6.1.2 */
#ifndef BLK_INTERNAL_H
#include "/lib/modules/6.1.2/build/block/blk.h"
#endif

MODULE_LICENSE("Dual BSD/GPL");

static int sbull_major = 0;
module_param(sbull_major, int, 0);
static int hardsect_size = 512;
module_param(hardsect_size, int, 0);
static int nsectors = 1024; /* How big the drive is */
module_param(nsectors, int, 0);
static int ndevices = 4;
module_param(ndevices, int, 0);

```

```

/*
 * The different "request modes" we can use.
 */
enum {
    RM_SIMPLE = 0, /* The extra-simple request function */
    RM_FULL = 1, /* The full-blown version */
    RM_NOQUEUE = 2, /* Use make_request */
};
static int request_mode = RM_SIMPLE;
module_param(request_mode, int, 0);

/*
 * Minor number and partition management.
 */
#define SBULL_MINORS 16
#define MINOR_SHIFT 4
#define DEVNUM(kdevnum) (MINOR(kdev_t_to_nr(kdevnum)) >> MINOR_SHIFT)

/*
 * We can tweak our hardware sector size, but the kernel talks to us
 * in terms of small sectors, always.
 */
#define KERNEL_SECTOR_SIZE 512

/*
 * After this much idle time, the driver will simulate a media change.
 */
#define INVALIDATE_DELAY 30*HZ

```

```

/*
 * The internal representation of our device.
 */
struct sbull_dev {
    int size;                      /* Device size in sectors */
    u8 *data;                      /* The data array */
    short users;                   /* How many users */
    short media_change;            /* Flag a media change? */
    spinlock_t lock;               /* For mutual exclusion */
    struct blk_mq_tag_set tag_set; /* tag_set added */
    struct request_queue *queue;   /* The device request queue */
    struct gendisk *gd;           /* The gendisk structure */
    struct timer_list timer;      /* For simulated media changes */
};

static struct sbull_dev *Devices = NULL;

/**
 * See https://github.com/openzfs/zfs/pull/10187/
 */
#ifndef LINUX_VERSION_CODE
static inline struct request_queue *
blk_generic_alloc_queue(make_request_fn make_request, int node_id)
#else
static inline struct request_queue *
blk_generic_alloc_queue(int node_id)
#endif
{
#ifndef LINUX_VERSION_CODE
    struct request_queue *q = blk_alloc_queue(GFP_KERNEL);
    if (q != NULL)
        blk_queue_make_request(q, make_request);

    return (q);
#endif
#if LINUX_VERSION_CODE < KERNEL_VERSION(5, 9, 0)
    return (blk_alloc_queue(make_request, node_id));
#elif LINUX_VERSION_CODE == KERNEL_VERSION(6, 1, 2)
    // return (blk_alloc_queue(node_id, false));
    return NULL;
#else
    return (blk_alloc_queue(node_id));
#endif
}

```

```

/*
 * Handle an I/O request.
 */
static void sbull_transfer(struct sbull_dev *dev, unsigned long sector,
                           unsigned long nsect, char *buffer, int write)
{
    unsigned long offset = sector*KERNEL_SECTOR_SIZE;
    unsigned long nbytes = nsect*KERNEL_SECTOR_SIZE;

    if ((offset + nbytes) > dev->size) {
        printk (KERN_NOTICE "Beyond-end write (%ld %ld)\n", offset, nbytes);
        return;
    }
    if (write)
        memcpy(dev->data + offset, buffer, nbytes);
    else
        memcpy(buffer, dev->data + offset, nbytes);
}

```

```

/*
 * The simple form of the request function.
 */
//static void sbull_request(struct request_queue *q)
static blk_status_t sbull_request(struct blk_mq_hw_ctx *hctx, const struct blk_mq_queue_data* bd) /* For blk-mq */
{
    struct request *req = bd->rq;
    struct sbull_dev *dev = req->part->bd_disk->private_data;
    struct bio_vec bvec;
    struct req_iterator iter;
    sector_t pos_sector = blk_rq_pos(req);
    void    *buffer;
    blk_status_t ret;

    blk_mq_start_request (req);

    if (blk_rq_is_passthrough(req)) {
        printk (KERN_NOTICE "Skip non-fs request\n");
        ret = BLK_STS_IOERR; // -EIO
        goto done;
    }
    rq_for_each_segment(bvec, req, iter)
    {
        size_t num_sector = blk_rq_cur_sectors(req);
        printk (KERN_NOTICE "Req dev %u dir %d sec %lld, nr %ld\n",
               (unsigned)(dev - Devices), rq_data_dir(req),
               pos_sector, num_sector);
        buffer = page_address(bvec.bv_page) + bvec.bv_offset;
        sbull_transfer(dev, pos_sector, num_sector,
                      buffer, rq_data_dir(req) == WRITE);
        pos_sector += num_sector;
    }
    ret = BLK_STS_OK;
done:
    blk_mq_end_request (req, ret);
    return ret;
}

```

```

/*
 * Transfer a single BIO.
 */
static int sbull_xfer_bio(struct sbull_dev *dev, struct bio *bio)
{
    struct bio_vec bvec;
    struct bvec_iter iter;
    // sector_t sector = bio->bi_iter.bi_sector;

    /* Do each segment independently. */
    bio_for_each_segment(bvec, bio, iter) {
        sector_t sector = iter.bi_sector;

        // char *buffer = __bio_kmap_atomic(bio, i, KM_USER0);
        // char *buffer = kmap_atomic(bvec.bv_page) + bvec.bv_offset;
        char *buffer = kmap_atomic(bvec.bv_page);
        unsigned long offset = bvec.bv_offset;
        size_t len = bvec.bv_len;

        // sbull_transfer(dev, sector, bio_cur_bytes(bio) >> 9,
        //                // sbull_transfer(dev, sector, (bio_cur_bytes(bio) / KERNEL_SECTOR_SIZE),
        //                //                buffer, bio_data_dir(bio) == WRITE);
        //                // sector += bio_cur_bytes(bio) >> 9;
        //                // sector += (bio_cur_bytes(bio) / KERNEL_SECTOR_SIZE);
        sbull_transfer(dev, sector, len, buffer + offset, bio_data_dir(bio) == WRITE);

        //__bio_kunmap_atomic(buffer, KM_USER0);
        kunmap_atomic(buffer);
    }
    return 0; /* Always "succeed" */
}

```

```

/*
 * Transfer a full request.
 */
static int sbull_xfer_request(struct sbull_dev *dev, struct request *req)
{
    struct bio *bio;
    int nsect = 0;

    __rq_for_each_bio(bio, req) {
        sbull_xfer_bio(dev, bio);
        //nsect += bio->bi_size/KERNEL_SECTOR_SIZE;
        nsect += bio->bi_iter.bi_size/KERNEL_SECTOR_SIZE;
    }
    return nsect;
}

```

```

/*
 * Smarter request function that "handles clustering".
 */
//static void sbull_full_request(struct request_queue *q)
static blk_status_t sbull_full_request(struct blk_mq_hw_ctx * hctx, const struct blk_mq_queue_data * bd)
{
    struct request *req = bd->rq;
    int sectors_xferred;
    //struct sbull_dev *dev = q->queuedata;
    struct sbull_dev *dev = req->q->queuedata;
    blk_status_t ret;

    blk_mq_start_request (req);
    //while ((req = blk_fetch_request(q)) != NULL) {
        //if (req->cmd_type != REQ_TYPE_FS) {
            if (blk_rq_is_passthrough(req)) {
                printk (KERN_NOTICE "Skip non-fs request\n");
                //__blk_end_request(req, -EIO, blk_rq_cur_bytes(req));
                ret = BLK_STS_IOERR; // -EIO;
                //continue;
                goto done;
            }
            sectors_xferred = sbull_xfer_request(dev, req);
            ret = BLK_STS_OK;
        done:
            //__blk_end_request(req, 0, sectors_xferred);
            blk_mq_end_request (req, ret);
        //}
    }
    return ret;
}

```

```
/*
 * The direct make request version.
 */
#endif
// static void sbull_make_request(struct request_queue *q, struct bio *bio)
static blk_qc_t sbull_make_request(struct request_queue *q, struct bio *bio)
#endif
#if LINUX_VERSION_CODE == KERNEL_VERSION(6, 1, 2)
static void sbull_make_request(struct bio *bio)
#else
static blk_qc_t sbull_make_request(struct bio *bio)
#endif
{
    //struct sbull_dev *dev = q->queuedata;
    struct sbull_dev *dev = bio->bi_private;
    int status;

    status = sbull_xfer_bio(dev, bio);
    bio->bi_status = status;
    bio_endio(bio);
#endif
#if LINUX_VERSION_CODE == KERNEL_VERSION(6, 1, 2)
#else
    return BLK_QC_T_NONE;
#endif
}

/*
 * Revalidate.
 * WE DO NOT TAKE THE LOCK HERE, for fear of deadlocking with open.
 * That needs to be reevaluated.
 */
int sbull_revalidate(struct gendisk *gd) {
    struct sbull_dev *dev = gd->private_data;

    if (dev->media_change) {
        dev->media_change = 0;
        memset(dev->data, 0, dev->size);
    }
    return 0;
}
```

```
/* Open */
static int sbull_open(struct block_device *bdev, fmode_t mode) {
    /* access to gendisk structure
     * and get driver's internal data structures. */
    struct sbull_dev *dev = bdev->bd_disk->private_data;

    del_timer_sync(&dev->timer);      /* remove the media removal timer if any is active. */
    // filp->private_data = dev;
    spin_lock(&dev->lock);
    if (!dev->users) {
#ifdef LINUX_VERSION_CODE < KERNEL_VERSION(5, 10, 0)
        check_disk_change(bdev);          /* check whether a media change has happened. */
#else
        /* For newer kernels (as of 5.10), bdev_check_media_change()
         * is used, in favor of check_disk_change(),
         * with the modification that invalidation
         * is no longer forced.
        */
        if (bdev_check_media_change(bdev)) {
            struct gendisk *gd = bdev->bd_disk;
            const struct block_device_operations *bdo = gd->fops;
            if (bdo)
                // bdo->revalidate_disk(gd);
                sbull_revalidate(gd);
        }
#endif
        dev->users++;
        spin_unlock(&dev->lock);
        return 0;
    }
}
```

```

/* Release */
static void sbull_release(struct gendisk *disk, fmode_t mode) {
    /* access to gendisk structure
     * and get driver's internal data structures. */
    struct sbull_dev *dev = disk->private_data;

    spin_lock(&dev->lock);
    dev->users--;
    /* decrement the user count. */

    if (!dev->users) {           /* start the media removal timer. */
        dev->timer.expires = jiffies + INVALIDATE_DELAY;
        add_timer(&dev->timer);
    }
    spin_unlock(&dev->lock);
}

/* To see whether the media has been changed */
int sbull_media_changed(struct gendisk *gd) {
    struct sbull_dev *dev = gd->private_data;
    return dev->media_change;
}

```

```

/*
 * The "invalidate" function runs out of the device timer; it sets
 * a flag to simulate the removal of the media.
 */
#ifndef LINUX_VERSION_CODE < KERNEL_VERSION(4, 15, 0)) && !defined(timer_setup)
void sbull_invalidate(unsigned long ldev)
{
    struct sbull_dev *dev = (struct sbull_dev *) ldev;
#else
void sbull_invalidate(struct timer_list * ldev)
{
    struct sbull_dev *dev = from_timer(dev, ldev, timer);
#endif

    spin_lock(&dev->lock);
    if (dev->users || !dev->data)
        printk (KERN_WARNING "sbull: timer sanity check failed\n");
    else
        dev->media_change = 1;
    spin_unlock(&dev->lock);
}

```

```

/* ioctl */
int sbull_ioctl(struct block_device *bdev, fmode_t mode, unsigned int cmd, unsigned long arg) {
    long size;
    struct hd_geometry geo;
    struct sbull_dev *dev = bdev->bd_disk->private_data;

    switch(cmd) {
        case HDIO_GETGEO:
            /*
             * Get geometry: since we are a virtual device, we have to make
             * up something plausible. So we claim 16 sectors, four heads,
             * and calculate the corresponding number of cylinders. We set the
             * start of data at sector four.
             */
            size = dev->size*(hardsect_size/KERNEL_SECTOR_SIZE);
            geo.cylinders = (size & ~0x3f) >> 6;
            geo.heads = 4;
            geo.sectors = 16;
            geo.start = 4;
            if (copy_to_user((void __user *) arg, &geo, sizeof(geo)))
                return -EFAULT;
            return 0;
    }

    return -ENOTTY; /* unknown command */
}

/*
 * The device operations structure.
 */
static struct block_device_operations sbull_ops = {
    .owner          = THIS_MODULE,
    .open           = sbull_open,
    .release        = sbull_release,
#ifndef LINUX_VERSION_CODE < KERNEL_VERSION(5, 9, 0)
    .media_changed  = sbull_media_changed, // DEPRECATED in v5.9
#else
    .submit_bio     = sbull_make_request,
#endif
    // .revalidate_disk = sbull_revalidate,
    .ioctl          = sbull_ioctl
};

```

```

static struct blk_mq_ops mq_ops_simple = {
    .queue_rq = sbull_request,
};

static struct blk_mq_ops mq_ops_full = {
    .queue_rq = sbull_full_request,
};

```

```

/* Setup device */
static void setup_device(struct sbull_dev *dev, int which) {
    int err;

    /* initialization */
    memset(dev, 0, sizeof(struct sbull_dev));
    dev->size = nsectors*hardsect_size;
    /* allocation */
    dev->data = vmalloc(dev->size);
    if (dev->data == NULL) {
        printk(KERN_NOTICE "vmalloc failure.\n");
        return;
    }
    spin_lock_init(&dev->lock);

    /*
     * The timer which "invalidates" the device.
     */
#ifndef LINUX_VERSION_CODE < KERNEL_VERSION(4, 15, 0)) && !defined(timer_setup)
    init_timer(&dev->timer);
    dev->timer.data = (unsigned long) dev;
    dev->timer.function = sbull_invalidate;
#else
    timer_setup(&dev->timer, sbull_invalidate, 0);
#endif
}

```

```

/*
 * The I/O queue, depending on whether we are using our own
 * make_request function or not.
 */
switch (request_mode) {
    case RM_NOQUEUE:
#ifndef LINUX_VERSION_CODE < KERNEL_VERSION(5, 9, 0)
        dev->queue = blk_generic_alloc_queue(sbull_make_request, NUMA_NO_NODE);
#else
        dev->queue = blk_generic_alloc_queue(NUMA_NO_NODE);
#endif
        if (dev->queue == NULL)
            goto out_vfree;
        break;

    case RM_FULL:
        // dev->queue = blk_init_queue(sbull_full_request, &dev->lock);
        // dev->queue = blk_mq_init_sq_queue(&dev->tag_set, &mq_ops_full, 128, BLK_MQ_F_SHOULD_MERGE);

        dev->tag_set.ops = &mq_ops_full;
        dev->tag_set.queue_depth = 128;
        dev->tag_set.flags = BLK_MQ_F_SHOULD_MERGE;
        err = blk_mq_alloc_tag_set(&dev->tag_set);

        if (err) {
            goto out_vfree;
        }

        dev->queue = blk_mq_init_queue(&dev->tag_set);

        if (dev->queue == NULL)
            // goto out_vfree;
            blk_mq_free_tag_set(&dev->tag_set);
        break;

    default:
        printk(KERN_NOTICE "Bad request mode %d, using simple\n", request_mode);
        /* fall into.. */
}

```

```

case RM_SIMPLE:
    // dev->queue = blk_init_queue(sbull_request, &dev->lock);
    // dev->queue = blk_mq_init_sq_queue(&dev->tag_set, &mq_ops_simple, 128, BLK_MQ_F_SHOULD_MERGE);

    dev->tag_set.ops = &mq_ops_simple;
    dev->tag_set.queue_depth = 128;
    dev->tag_set.flags = BLK_MQ_F_SHOULD_MERGE;
    err = blk_mq_alloc_tag_set(&dev->tag_set);

    if (err) {
        goto out_vfree;
    }

    dev->queue = blk_mq_init_queue(&dev->tag_set);

    if (dev->queue == NULL)
        // goto out_vfree;
        blk_mq_free_tag_set(&dev->tag_set);
    break;
}

blk_queue_logical_block_size(dev->queue, hardsect_size);
dev->queue->queuedata = dev;

/* allocate, initialize, install the gendisk structure */
dev->gd = blk_alloc_disk(SBULL_MINORS);
if(! dev->gd) {
    printk (KERN_NOTICE "alloc_disk failure\n");
    goto out_vfree;
}
dev->gd->major = sbull_major;
dev->gd->first_minor = which*SBULL_MINORS;
dev->gd->fops = &sbull_ops;
dev->gd->queue = dev->queue;
dev->gd->private_data = dev;
snprintf (dev->gd->disk_name, 32, "sbull%c", which + 'a');
set_capacity(dev->gd, nsectors*(hardsect_size/KERNEL_SECTOR_SIZE));
add_disk(dev->gd);
return;

out_vfree:
    if(dev->data)
        vfree(dev->data);
}

```

```

static int __init sbull_init(void)
{
    int i;
    /*
     * Get registered.
     */
    sbull_major = register_blkdev(sbull_major, "sbull");
    if (sbull_major <= 0) {
        printk(KERN_WARNING "sbull: unable to get major number\n");
        return -EBUSY;
    }
    /*
     * Allocate the device array, and initialize each one.
     */
    Devices = kmalloc(ndevices*sizeof (struct sbull_dev), GFP_KERNEL);
    if (Devices == NULL)
        goto out_unregister;
    for (i = 0; i < ndevices; i++)
        setup_device(Devices + i, i);

    return 0;
out_unregister:
    unregister_blkdev(sbull_major, "sbull");
    return -ENOMEM;
}

```

```

static void sbull_exit(void)
{
    int i;

    for (i = 0; i < ndevices; i++) {
        struct sbull_dev *dev = Devices + i;

        del_timer_sync(&dev->timer);
        if (!dev->gd) {
            del_gendisk(dev->gd);
            put_disk(dev->gd);
        }
        if (dev->queue) {
            if (request_mode == RM_NOQUEUE)
                //kobject_put (&dev->queue->kobj);
                blk_put_queue(dev->queue);
            else
                blk_mq_cleanup_rq;
        }
        if (dev->data)
            vfree(dev->data);
    }
    unregister_blkdev(sbull_major, "sbull");
    kfree(Devices);
}

module_init(sbull_init);
module_exit(sbull_exit);

```

Block Device Driver Kernel Object Compile

- 아래 명령어를 바탕으로 .ko (kernel object) 파일을 컴파일한다.

```
sudo make
```

```

devtae@devtae:~/sources/sbull$ ls
'\'          Module.symvers  sbull.ko      sbull.mod.c  sbull_unload
Makefile      sbull.c        sbull_load   sbull.mod.o
modules.order sbull.h        sbull.mod    sbull.o

```

- 성공적으로 커널 모듈이 컴파일됐음을 발견할 수 있다.

Block Device Driver 등록 과정

- bash 명령어를 바탕으로 실행하면 된다.
- 아래 명령어를 바탕으로 insmod 부터 mknod 및 chgrp, chmod 작업까지 모두 자동으로 진행한다.

```
sudo bash sbull_load
```

- 그 결과, 성공적으로 모듈이 적재되고 블록 디바이스 파일들이 적용됐음을 알 수 있다.

```
devtae@devtae:~/sources/sbull$ lsmod | grep sbull
sbull           16384  0
```

성공적으로 모듈이 적재되었음.

```
devtae@devtae:~/sources/sbull$ ls -al /dev/ | grep sbull
lrwxrwxrwx  1 root  root          6  2월 12 21:31 sbull  -> sbulla
brw-rw-r--  1 root  staff   252,    0  2월 12 21:31 sbulla
brw-rw-r--  1 root  staff   252,    1  2월 12 21:31 sbulla1
brw-rw-r--  1 root  staff   252,   10  2월 12 21:31 sbulla10
brw-rw-r--  1 root  staff   252,   11  2월 12 21:31 sbulla11
brw-rw-r--  1 root  staff   252,   12  2월 12 21:31 sbulla12
brw-rw-r--  1 root  staff   252,   13  2월 12 21:31 sbulla13
brw-rw-r--  1 root  staff   252,   14  2월 12 21:31 sbulla14
brw-rw-r--  1 root  staff   252,   15  2월 12 21:31 sbulla15
brw-rw-r--  1 root  staff   252,    2  2월 12 21:31 sbulla2
brw-rw-r--  1 root  staff   252,    3  2월 12 21:31 sbulla3
brw-rw-r--  1 root  staff   252,    4  2월 12 21:31 sbulla4
brw-rw-r--  1 root  staff   252,    5  2월 12 21:31 sbulla5
brw-rw-r--  1 root  staff   252,    6  2월 12 21:31 sbulla6
brw-rw-r--  1 root  staff   252,    7  2월 12 21:31 sbulla7
brw-rw-r--  1 root  staff   252,    8  2월 12 21:31 sbulla8
brw-rw-r--  1 root  staff   252,    9  2월 12 21:31 sbulla9
```

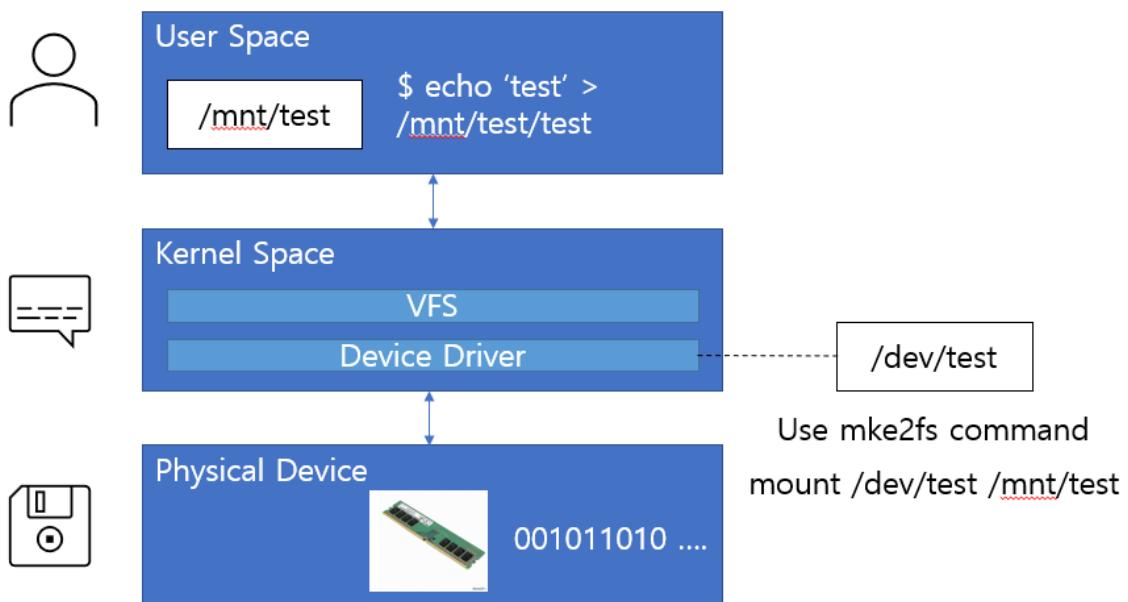
sbulla 까지 캡쳐. 이하 생략. b 로 시작되는 것을 바탕으로 block device 임을 알 수 있음.

Block Device Driver File System 적용

- 성공적으로 모듈을 적재한 직후 테스트해볼 수 있는 영역이다.

이론

- Kernel 영역에 ext2 파일시스템 파티션 /dev/sbull을 생성한다.
- User 영역의 파티션 /mnt/test를 생성한 ext2 파티션 /dev/sbull에 마운트한다.
- VFS가 정상적으로 해당 파일 시스템을 찾아 적용하게끔 한다.



추상화하면 다음과 같다.

```
~$ sudo mke2fs /dev/sbull
```

```
~$ mount /dev/test /mnt/test
```

Trouble Shooting

전체적으로, LDD3에 있는 예제 코드는 리눅스 2.6을 기준으로 한 것이라 현재 6.1.2 버전과는 많은 차이가 존재한다.

그에 따른, 업데이트에 대한 함수 반영을 한 과정을 담았다.

Trouble 1

```
:devtae@devtae:~/sources/sbull$ make
make -C /lib/modules/6.1.2/build M=/home/devtae/sources/sbull modules
make[1]: Entering directory '/home/devtae/Downloads/linux-6.1.2'
  CC [M]  /home/devtae/sources/sbull/sbull.o
/home/devtae/sources/sbull/sbull.c:21:10: fatal error: linux/genhd.h: No such file or directory
   21 | #include <linux/genhd.h>
      |
compilation terminated.
make[2]: *** [scripts/Makefile.build:250: /home/devtae/sources/sbull/sbull.o] Error 1
make[1]: *** [Makefile:1992: /home/devtae/sources/sbull] Error 2
make[1]: Leaving directory '/home/devtae/Downloads/linux-6.1.2'
make: *** [Makefile:8: default] Error 2
```

- Block Device Driver 예제 코드를 작성한 후 컴파일하는 과정에서 genhd.h 파일의 부재로 인한 에러가 발생하였다.
- 따라서, 리눅스 라이브러리 저장이 되어 있는 /usr/include/linux 디렉토리로 가서 확인해보았더니, genhd.h 파일이 없다는 것을 알게 되었음.

Solution 1

- 아래 사이트를 참고하면 책의 예제코드로 사용한 genhd.h 헤더파일이 삭제되었음을 확인할 수 있었다.

remove

```
remove @ 2022-01-24 9:39 Christoph Hellwig 2022-01-24 9:39 ` [PATCH 1/3] block: move disk_{block,unblock,flush}_events to blk.h
Christoph Hellwig `(3 more replies) 0 siblings, 4 replies; 9+ messages in thread
From: Christoph Hellwig @ 2022-01-24 9:39 UTC (permalink / raw)
To: axboe;+Cc: linux-block
Hi Jens, this patchset removes the header, which has no clearly split responsibilities from
https://lore.kernel.org/all/9bb3f4c6-4e21-e1d7-b4e1-f65d273cd3f8@nvidia.com/t/#m22da6ad1bdc2da48953f945b1a022e467bf95dbf
```

- blkdev.h에 대한 커밋 내역을 리서치하는 도중 genhd.h 파일이 삭제되면서 blkdev.h로 파일이 변경되었음을 발견할 수 있었음.

block: remove genhd.h · torvalds/linux@322cbb5

There is no good reason to keep genhd.h separate from the main blkdev.h header that includes it. So fold the contents of genhd.h into blkdev.h and remove genhd.h entirely.
Signed-off-by: Christop...

<https://github.com/torvalds/linux/commit/322cbb50de711814c42fb088f6d31901502c711a>

```
#include <linux/version.h>      /* LINUX_VERSION_CODE */
#include <linux/blk-mq.h>

#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/init.h>

#include <linux/sched.h>
#include <linux/kernel.h>
#include <linux/slab.h>
#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/timer.h>
#include <linux/types.h>
#include <linux/fcntl.h>
#include <linux/hdreg.h>
#include <linux/kdev_t.h>
#include <linux/vmalloc.h>
// #include <linux/genhd.h>    We don't have genhd.h
#include <linux/blkdev.h>
#include <linux/buffer_head.h> /* invalidate_bdev */
#include <linux/bio.h>
```

- `#include <linux/genhd.h>` 줄을 주석처리해줌으로써 해당 문제를 해결할 수 있었지만, 더 큰 오류더미의 해결이 필요하였다. 다음 Trouble 2에 나와 있다.

Trouble 2

- genhd.h 헤더파일의 문제를 해결한 뒤에 컴파일한 결과, 다음과 같은 오류들을 만나볼 수 있었다. 리눅스가 업데이트되면서 이전의 블록 디바이스 드라이버 코드에서의 함수와 구조체를 지원하지 않는다는 점이었다.

```

seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/blockdd$ sudo make
[sudo] password for seongbinyoon:
make -C /lib/modules/6.1.2/build M=/home/seongbinyoon/EdgeClab/blockdd modules
make[1]: Entering directory '/home/seongbinyoon/linux-6.1.2'
  CC [M]  /home/seongbinyoon/EdgeClab/blockdd/sbull.o
/home/seongbinyoon/EdgeClab/blockdd/sbull.c: In function 'blk_generic_alloc_queue':
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:107:17: error: implicit declaration of function 'blk_alloc_queue'; did you mean 'blk_sync_queue'? [-Werror=implicit-function-declaration]
  107 |         return (blk_alloc_queue(node_id));
|           ^~~~~~                         blk_sync_queue
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:107:17: warning: returning 'int' from a function with return type 'struct request_queue *' makes pointer from integer without a cast [-Wint-conversion]
  107 |         return (blk_alloc_queue(node_id));
|           ^~~~~~                         blk_sync_queue
/home/seongbinyoon/EdgeClab/blockdd/sbull.c: In function 'sbull_request':
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:139:36: error: 'struct request' has no member named 'rq_disk'
  139 |         struct sbull_dev *dev = req->rq_disk->private_data;
|           ^~~~~~
/home/seongbinyoon/EdgeClab/blockdd/sbull.c: In function 'sbull_xfer_bio':
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:185:46: error: implicit declaration of function 'bio_cur_bytes'; did you mean 'blk_rq_cur_bytes'? [-Werror=implicit-function-declaration]
  185 |             sbull_transfer(dev, sector, (bio_cur_bytes(bio) / KERNEL_SECTOR_SIZE),
|               ^~~~~~                         blk_rq_cur_bytes
/home/seongbinyoon/EdgeClab/blockdd/sbull.c: In function 'sbull_make_request':
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:256:36: error: 'struct bio' has no member named 'bi_disk'
  256 |         struct sbull_dev *dev = bio->bi_disk->private_data;
|           ^~~~~~
/home/seongbinyoon/EdgeClab/blockdd/sbull.c: In function 'sbull_open':
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:289:39: error: 'const struct block_device_operations' has no member named 'revalidate_disk'
  289 |             if (bdo && bdo->revalidate_disk)
|               ^~~~~~
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:290:36: error: 'const struct block_device_operations' has no member named 'revalidate_disk'
  290 |             bdo->revalidate_disk(gd);
|               ^~~~~~
/home/seongbinyoon/EdgeClab/blockdd/sbull.c: At top level:
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:405:28: error: initialization of 'void (*)(struct bio *)' from incompatible pointer type 'blk_qc_t (*)(struct bio *)' [aka 'unsigned int (*)(struct bio *)'] [-Werror=incompatible-pointer-types]
  405 |             .submit_bio = sbull_make_request,
|               ^~~~~~
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:407:28: note: (near initialization for 'sbull_ops.submit_bio')
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:407:10: error: 'struct block_device_operations' has no member named 'revalidate_disk'
  407 |             .revalidate_disk = sbull_revalidate,
|               ^~~~~~

```

```

/home/seongbinyoon/EdgeClab/blockdd/sbull.c:407:28: error: initialization of 'int (*)(struct bio *, struct io_comp_batch *, unsigned int)' from incompatible pointer type 'int (*)(struct gendisk *)' [-Werror=incompatible-pointer-types]
  407 |             .revalidate_disk = sbull_revalidate,
|               ^~~~~~
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:407:28: note: (near initialization for 'sbull_ops.poll_bio')
/home/seongbinyoon/EdgeClab/blockdd/sbull.c: In function 'setup_device':
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:464:30: error: implicit declaration of function 'blk_mq_init_sq_queue'; did you mean 'blk_mq_init_queue'? [-Werror=implicit-function-declaration]
  464 |             dev->queue = blk_mq_init_sq_queue(&dev->tag_set, &mq_ops_full, 128, BLK_MQ_F_SHOULD_MERGE);
|               ^~~~~~                         blk_mq_init_queue
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:464:28: warning: assignment to 'struct request_queue *' from 'int' makes pointer from integer without a cast [-Wint-conversion]
  464 |             dev->queue = blk_mq_init_sq_queue(&dev->tag_set, &mq_ops_full, 128, BLK_MQ_F_SHOULD_MERGE);
|               ^~~~~~
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:475:28: warning: assignment to 'struct request_queue *' from 'int' makes pointer from integer without a cast [-Wint-conversion]
  475 |             dev->queue = blk_mq_init_sq_queue(&dev->tag_set, &mq_ops_simple, 128, BLK_MQ_F_SHOULD_MERGE);
|               ^~~~~~
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:485:19: error: implicit declaration of function 'alloc_disk'; did you mean 'alloc_uid'? [-Werror=implicit-function-declaration]
  485 |             dev->gd = alloc_disk(SBULL_MINORS);
|               ^~~~~~                         alloc_uid
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:485:17: warning: assignment to 'struct gendisk *' from 'int' makes pointer from integer without a cast [-Wint-conversion]
  485 |             dev->gd = alloc_disk(SBULL_MINORS);
|               ^~~~~~
/home/seongbinyoon/EdgeClab/blockdd/sbull.c: In function 'sbull_exit':
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:550:33: error: implicit declaration of function 'blk_cleanup_queue' [-Werror=implicit-function-declaration]
  550 |             blk_cleanup_queue(dev->queue);
|               ^~~~~~

```

```

In file included from ./include/linux/kernel.h:29,
                 from ./arch/x86/include/asm/percpu.h:27,
                 from ./arch/x86/include/asm/preempt.h:6,
                 from ./include/linux/preempt.h:78,
                 from ./include/linux/spinlock.h:56,
                 from ./include/linux/wait.h:9,
                 from ./include/linux/wait_bit.h:8,
                 from ./include/linux/fs.h:6,
                 from ./include/linux/highmem.h:5,
                 from ./include/linux/bvec.h:10,
                 from ./include/linux/blktypes.h:10,
                 from ./include/linux/blkdev.h:9,
                 from ./include/linux/blk-mq.h:5,
                 from /home/seongbinyoon/EdgeClab/blockdd/sbull.c:2:
/home/seongbinyoon/EdgeClab/blockdd/sbull.c: In function 'setup_device':
./include/linux/printk.h:457:44: warning: this statement may fall through [-Wimplicit-fallthrough=]
  457 | #define printk(fmt, ...) printk_index_wrap(fmt, ##_VA_ARGS__)
|               ^~~~~
./include/linux/printk.h:429:17: note: in definition of macro 'printk_index_wrap'
  429 |     _P_FUNC(_fmt, ##_VA_ARGS__);
|               ^~~~~_
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:470:17: note: in expansion of macro 'printk'
  470 |         printk(KERN_NOTICE "Bad request mode %d, using simple\n", request_mode);
|               ^~~~~_
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:473:13: note: here
  473 |         case REQ_SIMPLE:
|               ^~~~~_
/home/seongbinyoon/EdgeClab/blockdd/sbull.c:497:9: warning: ignoring return value of 'add_disk' declared with attribute 'warn_unused_result' [-Wunused-result]
  497 |         add_disk(dev->gd);
|               ^~~~~_
cc1: some warnings being treated as errors
make[2]: *** [scripts/Makefile.build:250: /home/seongbinyoon/EdgeClab/blockdd/sbull.o] Error 1
make[1]: *** [Makefile:1992: /home/seongbinyoon/EdgeClab/blockdd] Error 2
make[1]: Leaving directory '/home/seongbinyoon/linux-6.1.2'
make: *** [Makefile:9: default] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/blockdd$ 

```

- 간단하게 정리했을 때, 다음과 같다.

1. 함수가 존재하지 않음.

- blk_generic_alloc_queue
- blk_alloc_queue
- bio_cur_bytes 함수가 존재하지 않음.
- blk_cleanup_queue(dev->queue) 없는 함수.

2. 구조체 변수 변경.

- struct request 부분 멤버 변수가 어떻게 구성되어 있는지? rq_disk라는 멤버 변수가 존재하지 않음.
- struct bio 부분 또한 확인할 것! bi_disk 멤버 변수가 없다고 함.
- bdo->revalidate_disk 변수 없음.
- block_device_operations 구조체에 대해 파악
- blk_mq_queue_data* bd -> rq 존재하지 않음

3. 이외의 컴파일 에러

- blk_mq_start_request 인자가 잘못되었음. (206줄)
- blk_rq_is_passthrough 인자가 잘못되었음. (210줄)
- del_timer_sync 인자가 timer_list 가 와야 함.
- blk_mq_init_sq_queue
- alloc_disk(SBULL_MINORS)

Solution 2

- 현재, blkdev.h 등의 헤더파일과 깃허브 커밋 내역을 바탕으로 이슈 트래킹하여 해결하였다.

<blk_alloc_queue 함수에 대한 오류 해결>

- 일단 blk_alloc_queue 함수에 대한 오류는 선언이 되지 않아 문제가 생긴 것 같다.
- 함수에 대한 커밋 내역 검색을 진행하여 2021년 6월 1일 기록에 따르면 include/linux/blkdev.h에서 빠지고 block/blk.h에 포함됐음을 확인할 수 있다. (개인적으로 해당 날짜에 어떤 버전인지에 대한 궁금증이 있다)

```

v ⇧ 1 ████ block/blk-core.c □ ...
↑ 00 -599,7 +599,6 @@ struct request_queue *blk_alloc_queue(int node_id)
599 599         kmem_cache_free(blk_requestq_cachep, q);
600 600         return NULL;
601 601     }
602 - EXPORT_SYMBOL(blk_alloc_queue);
603 602
604 603     /**
605 604     * blk_get_queue - increment the request_queue refcount
...
v ⇧ 2 ████ block/blk.h □ ...
↑ 00 -359,4 +359,6 @@ int bio_add_hw_page(struct request_queue *q, struct bio *bio,
359 359             struct page *page, unsigned int len, unsigned int offset,
360 360             unsigned int max_sectors, bool *same_page);
361 361
362 + struct request_queue *blk_alloc_queue(int node_id);
363 +
362 364 #endif /* BLK_INTERNAL_H */
...
v ⇧ 1 ████ include/linux/blkdev.h □ ...
↑ 00 -1213,7 +1213,6 @@ static inline int blk_rq_map_sg(struct request_queue *q, struct request *rq,
1213 1213     extern void blk_dump_rq_flags(struct request *, char *);
1214 1214
1215 1215     bool __must_check blk_get_queue(struct request_queue *);
1216 - struct request_queue *blk_alloc_queue(int node_id);
1217 1216     extern void blk_put_queue(struct request_queue *);
1218 1217     extern void blk_set_queue_dying(struct request_queue *);
1219 1218
...

```

blk.h

```

#define ADDPART_FLAG_WHOLEDISK 2
int bdev_add_partition(struct gendisk *disk, int partno, sector_t start,
                      sector_t length);
int bdev_del_partition(struct gendisk *disk, int partno);
int bdev_resize_partition(struct gendisk *disk, int partno, sector_t start,
                        sector_t length);
void blk_drop_partitions(struct gendisk *disk);

struct gendisk *_alloc_disk_node(struct request_queue *q, int node_id,
                                 struct lock_class_key *lkclass);

int bio_add_hw_page(struct request_queue *q, struct bio *bio,
                    struct page *page, unsigned int len, unsigned int offset,
                    unsigned int max_sectors, bool *same_page);

static inline struct kmem_cache *blk_get_queue_kmem_cache(bool srcu)
{
    if (srcu)
        return blk_requestq_srcu_cachep;
    return blk_requestq_cachep;
}
struct request_queue *blk_alloc_queue(int node_id, bool alloc_srcu);

int disk_scan_partitions(struct gendisk *disk, fmode_t mode);

int disk_alloc_events(struct gendisk *disk);
void disk_add_events(struct gendisk *disk);
void disk_del_events(struct gendisk *disk);
void disk_release_events(struct gendisk *disk);
void disk_block_events(struct gendisk *disk);
void disk_unblock_events(struct gendisk *disk);
void disk_flush_events(struct gendisk *disk, unsigned int mask);
extern struct device_attribute dev_attr_events;
extern struct device_attribute dev_attr_events_async;
extern struct device_attribute dev_attr_events_poll_msecs;

extern struct attribute_group blk_trace_attr_group;

long blkdev_ioctl(struct file *file, unsigned cmd, unsigned long arg);
long compat_blkdev_ioctl(struct file *file, unsigned cmd, unsigned long arg);

extern const struct address_space_operations def_blk_aops;

int disk_register_independent_access_ranges(struct gendisk *disk);
/bulk_alloc_queue

```

sbulk.c

```
// for linux kernel 6.1.2
#ifndef BLK_INTERNAL_H
#include "/lib/modules/6.1.2/build/block/blk.h"
#endif
```

- 따라서, 직접 Internal 로 전환된 헤더파일을 include 해줌으로써 해당 문제는 해결할 수 있었다.
- 하지만, 예제 코드의 경우 argument 가 (int node_id) 인데에 반해, 실제 blk.h 의 경우 (int node_id, bool alloc_srcu) 인 것을 확인할 수 있었다.
- SRCU 의 경우 Sleepable Read-Copy-Update 임을 뜻한다. Spinlock, Mutex, Semaphore 와 같은 일반적인 동기화 메커니즘과 다르게 동시 접근 가능한 동기화 메커니즘을 말한다. 해당 예제는 spinlock 을 사용하기에 false 옵션을 주겠다.

```
#if (LINUX_VERSION_CODE < KERNEL_VERSION(5, 7, 0))
    struct request_queue *q = blk_alloc_queue(GFP_KERNEL);
    if(q != NULL)
        blk_queue_make_request(q, make_request);

    return (q);
#elif (LINUX_VERSION_CODE < KERNEL_VERSION(5, 9, 0))
    return (blk_alloc_queue(make_request, node_id));
#elif (LINUX_VERSION_CODE == KERNEL_VERSION(6, 1, 2))
    return (blk_alloc_queue(node_id, false));
#else
    return (blk_alloc_queue(node_id));
#endif
```

- 일단 6.1.2 버전에 대한 처리만 진행한 상태이다. 추후에 버전에 대한 정보를 얻을 수 있는 상황에 따로 업데이트를 할 수 있을 듯 보인다.
- 아래 부분을 보면 알 수 있는데, 현재 리눅스 6.1.2 버전에서는 blk_alloc_queue 함수가 무의미하다. 따라서, NULL 처리하였는데 자세한 내용은 아래 항목을 참고하길 바란다.

<struct request 멤버 변수 rq_disk 존재하지 않음 문제 해결>

- 구조체의 변수 선언을 바탕으로 이해한 뒤 문제를 해결할 수 있다.
- struct request 구조체 선언의 경우 다음과 같이 알아낼 수 있다.

```
devtae@devtae:/lib/modules/6.1.2/build/include/linux$ find ./ -name "*.h" | xargs grep "struct request {"
./blk-mq.h:struct request {
```

```
~$ cd /lib/modules/6.1.2/build/include/linux/
~$ find ./ -name "*.h" | xargs grep "struct request {"
```

```

*/
struct request {
    struct request_queue *q;
    struct blk_mq_ctx *mq_ctx;
    struct blk_mq_hw_ctx *mq_hctx;

    blk_opf_t cmd_flags;           /* op and common flags */
    req_flags_t rq_flags;

    int tag;
    int internal_tag;

    unsigned int timeout;

    /* the following two fields are internal, NEVER access directly */
    unsigned int __data_len;      /* total data len */
    sector_t __sector;           /* sector cursor */

    struct bio *bio;
    struct bio *biotail;

    union {
        struct list_head queuelist;
        struct request *rq_next;
    };

    struct block_device *part;
#endif CONFIG_BLK_RQ_ALLOC_TIME
    /* Time that the first bio started allocating this request. */
    u64 alloc_time_ns;
#endif

```

107,0-1

6%

- 그 결과, blk-mq.h 에 있다는 것을 파악할 수 있다.

```

struct block_device {
    sector_t             bd_start_sect;
    sector_t             bd_nr_sectors;
    struct disk_stats __percpu *bd_stats;
    unsigned long        bd_stamp;
    bool                bd_read_only; /* read-only policy */
    dev_t                bd_dev;
    atomic_t              bd_openers;
    struct inode *       bd_inode;     /* will die */
    struct super_block * bd_super;
    void *               bd_claiming;
    struct device        bd_device;
    void *               bd_holder;
    int                 bd HOLDERS;
    bool                bd_write_holder;
    struct kobject       *bd_holder_dir;
    u8                  bd_partno;
    spinlock_t           bd_size_lock; /* for bd_inode->i_size updates */
    struct gendisk *     bd_disk;
    struct request_queue *bd_queue;

    /* The counter of freeze processes */
    int                 bd_fsfreeze_count;

```

57,1-8

7%

- 위와 같은 방식으로 struct request 구조체 안에 struct block_device *part 가 있는 것을 확인하고 struct block_device 구조체 안에 gendisk *bd_disk 가 있는 것을 확인하고, 최종적으로 그 안의 private_data 로 연결할 수 있게 된다.

```
//struct sbull_dev *dev = req->rq_disk->private_data;
struct sbull_dev *dev = req->part->bd_disk->private_data;
```

<bio_cur_bytes 함수가 존재하지 않음 문제 해결>

- bio_cur_bytes 함수는 원래 bio.h 에 포함되어있었다. 하지만, linux kernel 6.1.2 버전을 기준으로 찾아보았을 때 발견할 수 없었다. 업데이트 등의 방법을 통해 다른 함수로 대체된 것 같다.
- bio.h 커밋 내역을 확인해보았을 때, bio.h 의 bio_cur_bytes 함수가 blk-mq.h 의 blk_rq_cur_bytes 로 바뀐 것을 확인할 수 있다.

```
diff --git a/include/linux/bio.h b/include/linux/bio.h
--- a/include/linux/bio.h
+++ b/include/linux/bio.h
@@ -69,14 +69,6 @@ static inline bool bio_no_advance_iter(const struct bio *bio)
         bio_op(bio) == REQ_OP_WRITE_ZEROES;
     }
 
-    static inline unsigned int bio_cur_bytes(struct bio *bio)
-    {
-        if (bio_has_data(bio))
-            return bio iovector(bio).bv_len;
-        else /* dataless requests such as discard */
-            return bio->bi_iter.bi_size;
-    }
-
 static inline void *bio_data(struct bio *bio)
{
    if (bio_has_data(bio))
}
 
diff --git a/include/linux/blk-mq.h b/include/linux/blk-mq.h
--- a/include/linux/blk-mq.h
+++ b/include/linux/blk-mq.h
@@ -927,7 +927,11 @@ static inline unsigned int blk_rq_bytes(const struct request *rq)
927 927
928 928     static inline int blk_rq_cur_bytes(const struct request *rq)
929 929     {
930 -        return rq->bio ? bio_cur_bytes(rq->bio) : 0;
930 +        if (!rq->bio)
931 +            return 0;
932 +        if (!bio_has_data(rq->bio)) /* dataless requests such as discard */
933 +            return rq->bio->bi_iter.bi_size;
934 +        return bio iovector(rq->bio).bv_len;
931 935     }
932 936
933 937     unsigned int blk_rq_err_bytes(const struct request *rq);
```

blk-mq.h

```
static inline int blk_rq_cur_bytes(const struct request *rq)
{
    if (!rq->bio)
        return 0;
    if (!bio_has_data(rq->bio)) /* dataless requests such as discard */
        return rq->bio->bi_iter.bi_size;
    return bio iovector(rq->bio).bv_len;
}
```

- 하지만 인자의 타입이 바뀌었기 때문에 적절하게 변환해주어야 한다.
- 따라서, request * 형식으로 인자를 입력해주기 위해서 bio 구조체를 바탕으로 request로 변환하여 반환하는 bio_to_request 매크로문에 대해 조사해보았다.
- 하지만, 이 또한 이전 버전에서 지원하는 매크로이며 현 6.1.2 버전에서는 지원하지 않는다는 점에서 새롭게 코드를 수정하게 될 수 밖에 없었다.

```
// Transfer a single BIO.
static int sbull_xfer_bio(struct sbull_dev *dev, struct bio *bio)
{
    struct bio_vec bvec;
    struct bvec_iter iter;
    //sector_t sector = bio->bi_iter.bi_sector;

    // Do each segment independently.
    bio_for_each_segment(bvec, bio, iter) {
        sector_t sector = iter.bi_sector;

        //char *buffer = __bio_kmap_atomic(bio, i, KM_USER0);
        //char *buffer = kmap_atomic(bvec.bv_page) + bvec.bv_offset;
        char *buffer = kmap_atomic(bvec.bv_page);
        unsigned long offset = bvec.bv_offset;
        size_t len = bvec.bv_len;

        //sbull_transfer(dev, sector, bio_cur_bytes(bio) >> 9),
        //sbull_transfer(dev, sector, (bio_cur_bytes(bio) / KERNEL_SECTOR_SIZE),
        //               buffer, bio_data_dir(bio) == WRITE);
        //sector += bio_cur_bytes(bio) == WRITE;
        //sector += (bio_cur_bytes(bio) / KERNEL_SECTOR_SIZE);
        sbull_transfer(dev, sector, len, buffer + offset, bio_data_dir(bio) == WRITE);

        //__bio_kunmap_atomic(buffer, KM_USER0);
        kunmap_atomic(buffer);
    }
    return 0; // Always succeed
}
```

- bio_cur_bytes 함수를 사용하는 이전과 다르게 bio_vec 구조체의 bv_len 멤버 변수를 활용하여 적용할 수 있었다.
- 그 결과, 컴파일 에러를 해결할 수 있었다.

<struct bio 의 bi_disk 멤버 변수 없음 문제 해결>

- 이전 Linux 버전에서는 struct bio에 대한 멤버 변수에 bi_disk가 존재하였지만, 현 버전에선 존재하지 않았다.
- 따라서, 현재 struct bio 구조체 구성에 대해 알아보았다.

```
struct bio {
    struct bio      *bi_next;    /* request queue link */
    struct block_device *bi_bdev;
    blk_opf_t        bi_opf;     /* bottom bits REQ_OP, top bits
                                  * req_flags.
                                  */
    unsigned short   bi_flags;   /* BIO_* below */
    unsigned short   bi_ioprio;
    blk_status_t     bi_status;
    atomic_t         __bi_remaining;

    struct bvec_iter bi_iter;

    blk_qc_t         bi_cookie;
    bio_end_io_t     *bi_end_io;
    void             *bi_private;
#endif CONFIG_BLK_CGROUP
/*
 * Represents the association of the css and request_queue for the bio.
linux/bk_types.h [RO]                                     270,1-4          47%
```

```
//struct sbull_dev *dev = q->queuedata;
struct sbull_dev *dev = bio->bi_private;
```

- bi_disk→private_data 대신 bi_private 변수로 수정하여 문제를 해결하였다.

<bdo의 revalidate_disk 속성 존재 안함 문제 해결>

- Block Device Operations 구조체에서 revalidate_disk 함수에 대한 할당을 진행하려는데 본 커널의 구조체에서 등록되지 않은 변수라는 에러가 발생하였다.

[9/9] block: remove revalidate_disk()

```
@@@ -397,7 +397,7 @@ struct mddev { * These locks are separate due to conflicting interactions * with bdev->bd_mutex. * Lock ordering is: - * reconfig_mutex -> bd_mutex : e.g. do_md_run -> revalidate_disk + * reconfig_mutex -> bd_mutex * bd_mutex -> open_mutex: e.g.
```

<https://patchwork.kernel.org/project/linux-nvdimm/patch/20200901155748.2884-10-hch@lst.de/>

- 위 링크를 바탕으로 확인해보면 revalidate_disk 함수에 대해 삭제를 하였다는 커밋 로그를 발견할 수 있었다.
- 따라서, revalidate_disk 함수 할당을 주석 처리하고 sbull_open에서 호출되는 방식을 bdo를 통해 접근하는 것이 아닌 곧바로 sbull_revalidate 함수로 접근할 수 있도록 수정하였음.

```

/*
 * The device operations structure.
 */
static struct block_device_operations sbull_ops = {
    .owner          = THIS_MODULE,
    .open           = sbull_open,
    .release        = sbull_release,
#if (LINUX_VERSION_CODE < KERNEL_VERSION(5, 9, 0))
    .media_changed  = sbull_media_changed, // DEPRECATED in v5.9
#else
    .submit_bio     = sbull_make_request,
#endif
    // .revalidate_disk = sbull_revalidate,
    .ioctl          = sbull_ioctl
};

```

```

    if (bdev_check_media_change(bdev)) {
        struct gendisk *gd = bdev->bd_disk;
        const struct block_device_operations *bdo = gd->fops;
        if (bdo)
            // bdo->revalidate_disk(gd);
            sbull_revalidate(gd);
    }
}

```

<.submit_bio = sbull_make_request 문제 해결>

- linux/blkdev.h 파일을 바탕으로 확인해보면, block_device_operations 구조체에 submit_bio 함수 포인터를 멤버로 가지고 있다.
- 함수 원형은 void (*submit_bio)(struct bio *bio); 와 같다.
- 하지만, sbull.c 파일을 바탕으로 sbull_make_request 함수를 확인해보았을 때 blk_qc_t 타입의 값을 반환하는 것으로 확인된다.
- 따라서, 함수 반환 타입을 void 로 수정하였다.

```

// The direct make request version.
#if (LINUX_VERSION_CODE < KERNEL_VERSION(5, 9, 0))
// static void sbull_make_request(struct request_queue *q, struct bio *bio)
static blk_qc_t sbull_make_request(struct request_queue *q, struct bio *bio)
#elif (LINUX_VERSION_CODE == KERNEL_VERSION(6, 1, 2))
static void sbull_make_request(struct bio *bio)
#else
static blk_qc_t sbull_make_request(struct bio *bio)
#endif
{
    //struct sbull_dev *dev = q->queuedata;
    struct sbull_dev *dev = bio->bi_private;
    int status;

    status = sbull_xfer_bio(dev, bio);
    bio->bi_status = status;
    bio_endio(bio);
#endif
#else
    return BLK_QC_T_NONE;
#endif
}

```

<blk_mq_init_sq_queue 함수 문제 해결>

- 이 함수 또한, 삭제된 것으로 확인되어진다.

[15/30] blk-mq: remove blk_mq_init_sq_queue - Patchwork
<https://patchwork.kernel.org/project/linux-mm/patch/20210602065345.355274-16-hch@lst.de/>

- 이전 blk_mq_init_sq_queue 함수의 생성자를 확인해보았을 때, 다음과 같았다.

```

struct request_queue *blk_mq_init_sq_queue(
    struct blk_mq_tag_set *set,
    const struct blk_mq_ops *ops,
    unsigned int queue_depth,
    unsigned int set_flags)

```

- 따라서, dev 속성 중에서 tag_set 에 이전 blk_mq_init_sq_queue 함수에서 호출하던 속성에 대해 다음과 같이 설정하고 blk_mq_init_sq_queue 함수를 호출하면 된다.

```

case RM_SIMPLE:
    //dev->queue = blk_init_queue(sbull_request, &dev->lock);
    //dev->queue = blk_mq_init_sq_queue(&dev->tag_set, &mq_ops_simple, 128, BLK_MQ_F_SHOULD_MERGE)

    dev->tag_set.ops = &mq_ops_simple;
    dev->tag_set.queue_depth = 128;
    dev->tag_set.flags = BLK_MQ_F_SHOULD_MERGE;
    err = blk_mq_alloc_tag_set(&dev->tag_set);

    if (err) {
        goto out_vfree;
    }

    dev->queue = blk_mq_init_queue(&dev->tag_set);

    if(dev->queue == NULL)
        goto out_vfree;
    break;

```

- RM_SIMPLE case 뿐만 아니라, 다른 케이스(RM_FULL)에서도 같은 방식으로 처리한다. (setup_device() 함수에서 처리하고, err의 경우 앞에 선언해야 함)
- 추가적으로, err if 문에서 걸린 경우 out_vfree 로 가면 되지만, dev->queue == NULL 인 경우에는 blk_mq_free_tag_set 함수 (tag_set 메모리 할당 해제)를 호출해주는 것이 좋겠다.

```

if (dev->queue == NULL)
    // goto out_vfree;
    blk_mq_free_tag_set(&dev->tag_set);

```

<alloc_disk 함수에 대한 부재>

- alloc_disk(int MINOR_NUMBER) 를 통해, gendisk * 형태의 반환값을 dev->gd 변수에 저장한다.
- 자료 서칭했을 때 alloc_disk 와 alloc_disk_node 에 대한 함수가 제거되었음을 확인할 수 있었다.
- 이에 따라, 커널 소스에서 alloc_disk 에 대한 내용을 검색해보았다.

```

devtae@devtae:/lib/modules/6.1.2/build/include$ find ./ -name "*.h" | xargs grep
"alloc_disk"
./linux/blkdev.h:struct gendisk *__blk_alloc_disk(int node, struct lock_class_key
*lkclass);
./linux/blkdev.h: * blk_alloc_disk - allocate a gendisk structure
./linux/blkdev.h:#define blk_alloc_disk(node_id) \
./linux/blkdev.h:      __blk_alloc_disk(node_id, &__key); \
./linux/blk-mq.h:struct gendisk *__blk_mq_alloc_disk(struct blk_mq_tag_set *set,
void *queuedata,
./linux/blk-mq.h:#define blk_mq_alloc_disk(set, queuedata) \
./linux/blk-mq.h:      __blk_mq_alloc_disk(set, queuedata, &__key); \
./linux/blk-mq.h:struct gendisk *blk_mq_alloc_disk_for_queue(struct request_queue
*q,

```

- 추가적으로 파일들을 조사해본 결과, blk_alloc_disk(int node_id) 를 사용하면 해결할 수 있을 듯 보였다.

```

struct gendisk * __blk_alloc_disk(int node, struct lock_class_key *lkclass);

/**
 * blk_alloc_disk - allocate a gendisk structure
 * @node_id: numa node to allocate on
 *
 * Allocate and pre-initialize a gendisk structure for use with BIO based
 * drivers.
 *
 * Context: can sleep
 */
#define blk_alloc_disk(node_id) \
({ \
    static struct lock_class_key __key; \
    \ \
    __blk_alloc_disk(node_id, &__key); \
})

```

- alloc_disk에서 blk_alloc_disk로 수정할 수 있다.
- 하지만, Linux 6.1.2에서 blk_alloc_disk에 대한 함수 선언은 있었지만 함수 내용은 어디에서도 찾을 수 없었다.
- 따라서, 해당 함수를 NULL 처리하였다.

```

#if (LINUX_VERSION_CODE < KERNEL_VERSION(5, 9, 0))
static inline struct request_queue *
blk_generic_alloc_queue(make_request_fn make_request, int node_id)
#else
static inline struct request_queue *
blk_generic_alloc_queue(int node_id)
#endif
{
#if (LINUX_VERSION_CODE < KERNEL_VERSION(5, 7, 0))
    struct request_queue *q = blk_alloc_queue(GFP_KERNEL);
    if(q != NULL)
        blk_queue_make_request(q, make_request);

    return (q);
#elif (LINUX_VERSION_CODE < KERNEL_VERSION(5, 9, 0))
    return (blk_alloc_queue(make_request, node_id));
#elif (LINUX_VERSION_CODE == KERNEL_VERSION(6, 1, 2))
    //return (blk_alloc_queue(node_id, false)); // we will not use Sleepable RCU
    return NULL; // blk_alloc_queue is deprecated in linux 6.1.2
#else
    return (blk_alloc_queue(node_id));
#endif
}

```

- 구글링, 깃허브 커밋 내역 리뷰, 헤더 파일 탐색 등의 방식들을 바탕으로 컴파일 문제를 해결할 수 있었다.
- 버전이 계속해서 업그레이드되면서 구조체와 함수 또한 계속해서 변형되어 생긴 오류로 보인다.
- 앞으로도 버전마다 Block Device Driver 코드가 모두 달라질 것이라 생각하여 천천히 오류 해결을 위해 접근을 해야 겠다고 느끼게 되었다.

Source

<https://phoenixnap.com/kb/build-linux-kernel>
<https://www.redhat.com/ko/topics/virtualization/what-is-KVM>
<https://www.redhat.com/ko/topics/virtualization/what-is-a-hypervisor>
<https://ko.linux-console.net/?p=705#gsc.tab=0>
<https://young-cow.tistory.com/86>
<https://jjeongil.tistory.com/1940>
http://www.chlux.co.kr/bbs/board.php?bo_table=board02&wr_id=172
http://www.linuxdata.org/bbs/board.php?bo_table=OpenShift&wr_id=44
<https://louky0714.tistory.com/35>
<https://blog.iolate.kr/265>
<https://blog.dalso.org/linux/ubuntu/16012>
<https://ko.savtec.org/articles/howto/how-to-install-kvm-and-create-virtual-machines-on-ubuntu.html>
<https://rainbound.tistory.com/entry/ubuntu-kvm-설치>
<https://osg.kr/archives/1458>
<https://unix.stackexchange.com/questions/412065/ssh-connection-x11-connection-rejected-because-of-wrong-authentication>
<http://bahndal.egloos.com/534415>
<https://harryp.tistory.com/684>
<https://superuser.com/questions/805725/how-do-i-debug-x11-connection-rejected-because-of-wrong-authentication>
<https://github.com/freestyle3r/kernel-drivers/tree/master/scull>
<https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=pajaebeo&logNo=102854820>
<https://github.com/martinezjavier/ldd3/tree/master/scull>
<https://lore.kernel.org/all/9bb3f4c6-4e21-e1d7-b4e1-f65d273cd3f8@nvidia.com/t/#m22da6ad1bdc2da48953f945b1a022e467bf95dbf>
<https://github.com/torvalds/linux/commit/322cbb50de711814c42fb088f6d31901502c711a>
<https://patchwork.kernel.org/project/linux-nvdimm/patch/20200901155748.2884-10-hch@lst.de/>
<https://patchwork.kernel.org/project/linux-mm/patch/20210602065345.355274-16-hch@lst.de/>