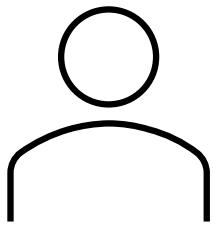


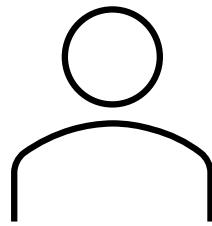
Introduction

Edge Cloud Lab

Introduction – 발표자 소개



김태현



윤성빈

- 학부생 3학년 진학
- Edge Cloud Lab Edge 팀

Introduction – 목표 소개



- Edge 컴퓨팅에서 Ransomware로부터 데이터 보호를 위한 스토리지 시스템 구현하기

Introduction – 목표 소개

What is Ransomware?

Introduction – 목표 소개

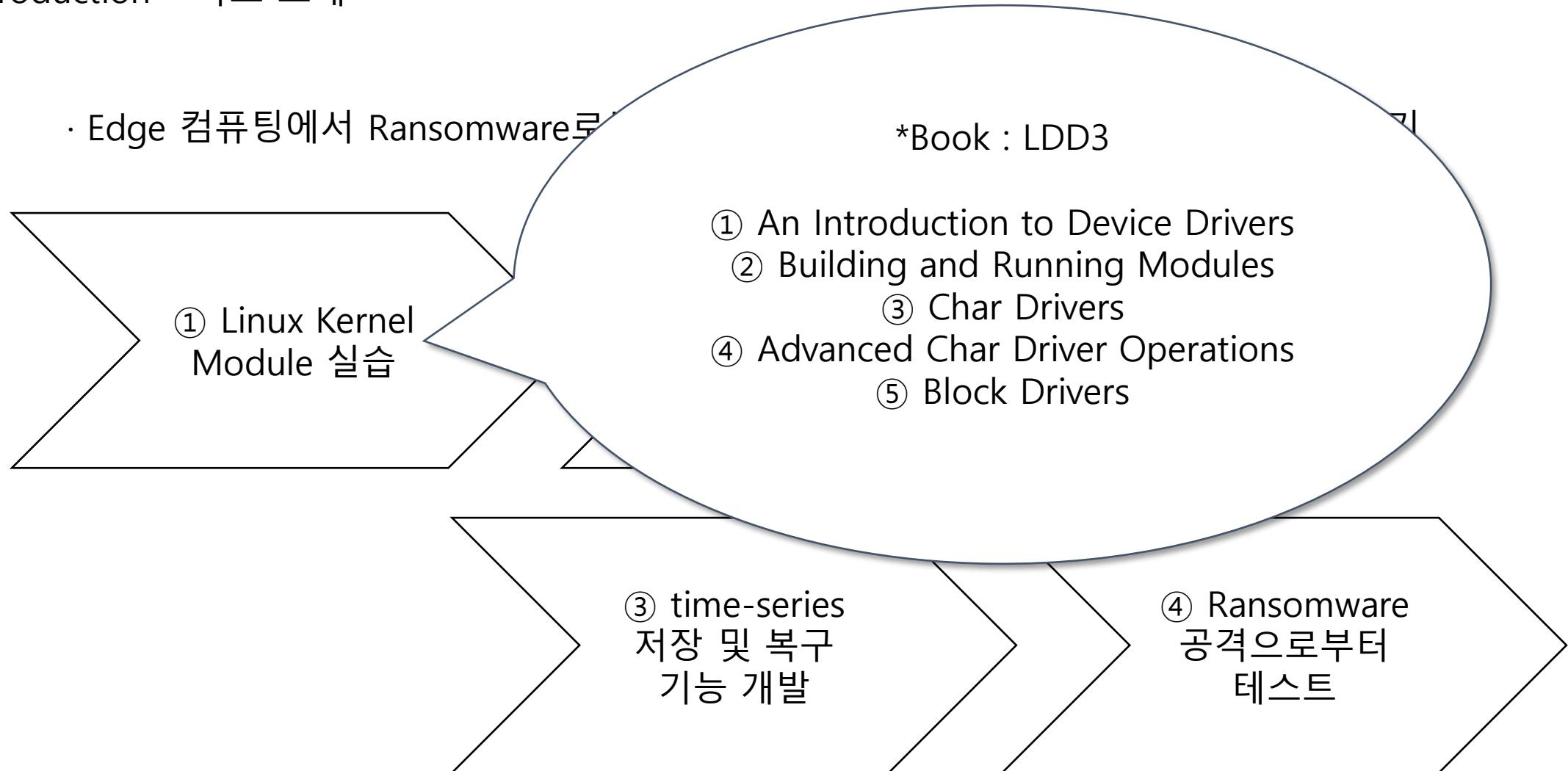
How Serious Is Ransomware Attack In Cloud System?

Introduction – 목표 소개

- Edge 컴퓨팅에서 Ransomware로부터 데이터 보호를 위한 스토리지 시스템 구현하기



Introduction – 목표 소개



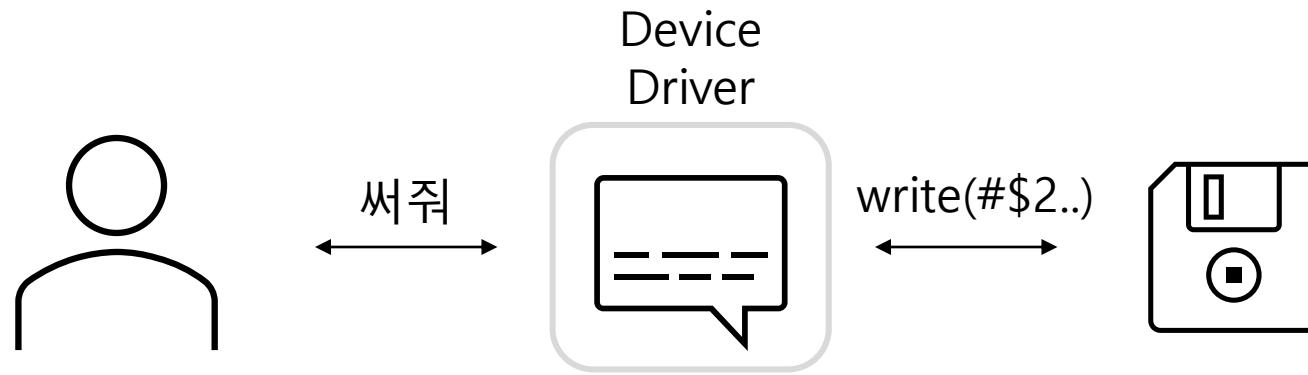
Introduction – 목표 소개

- Edge 컴퓨팅에서 Ransomware로부터 데이터 보호를 위한 스토리지 시스템 구현하기



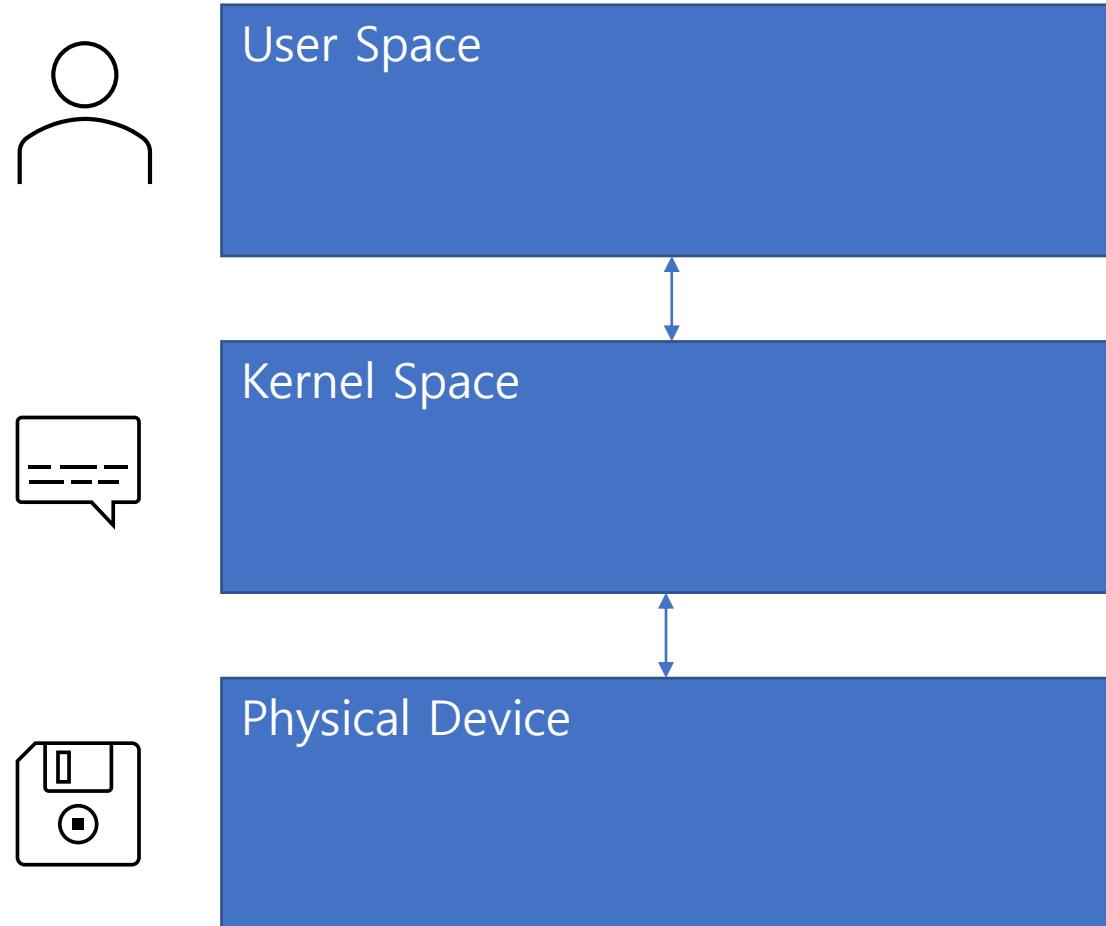
Device Driver

What is Device Driver?

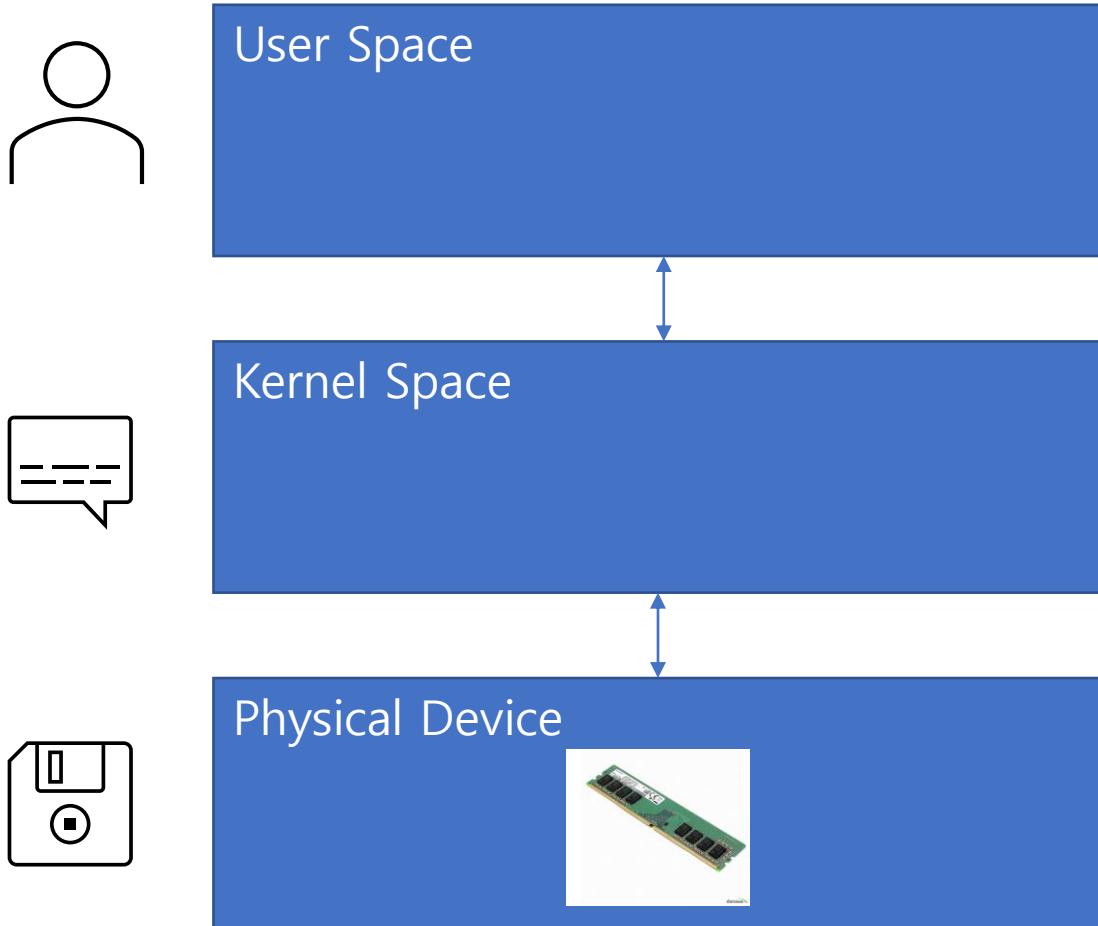


- 리눅스 OS에서 연결된 장치를 사용자의 입력에 따라 활용할 수 있도록 하는 드라이버

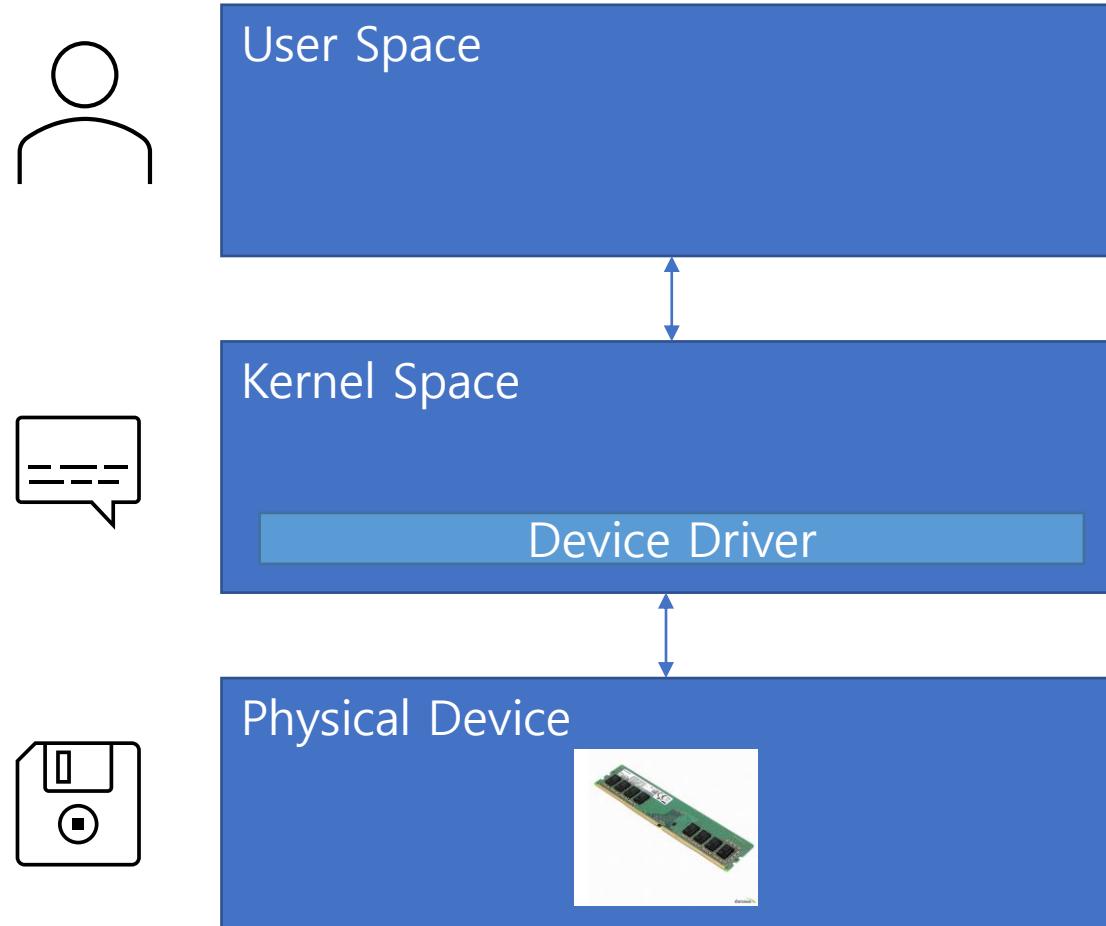
What is Device Driver for File-System?



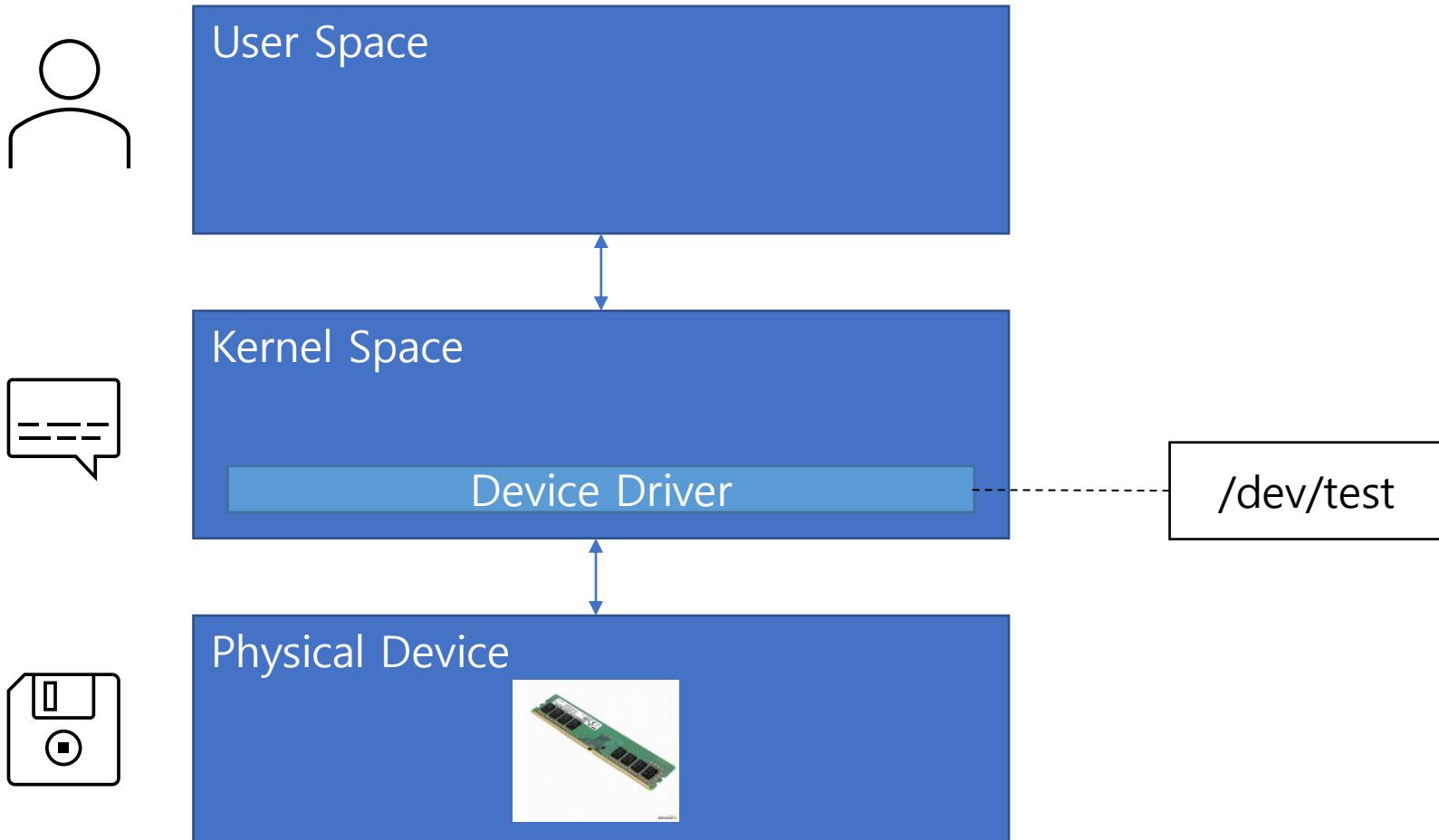
What is Device Driver for File-System?



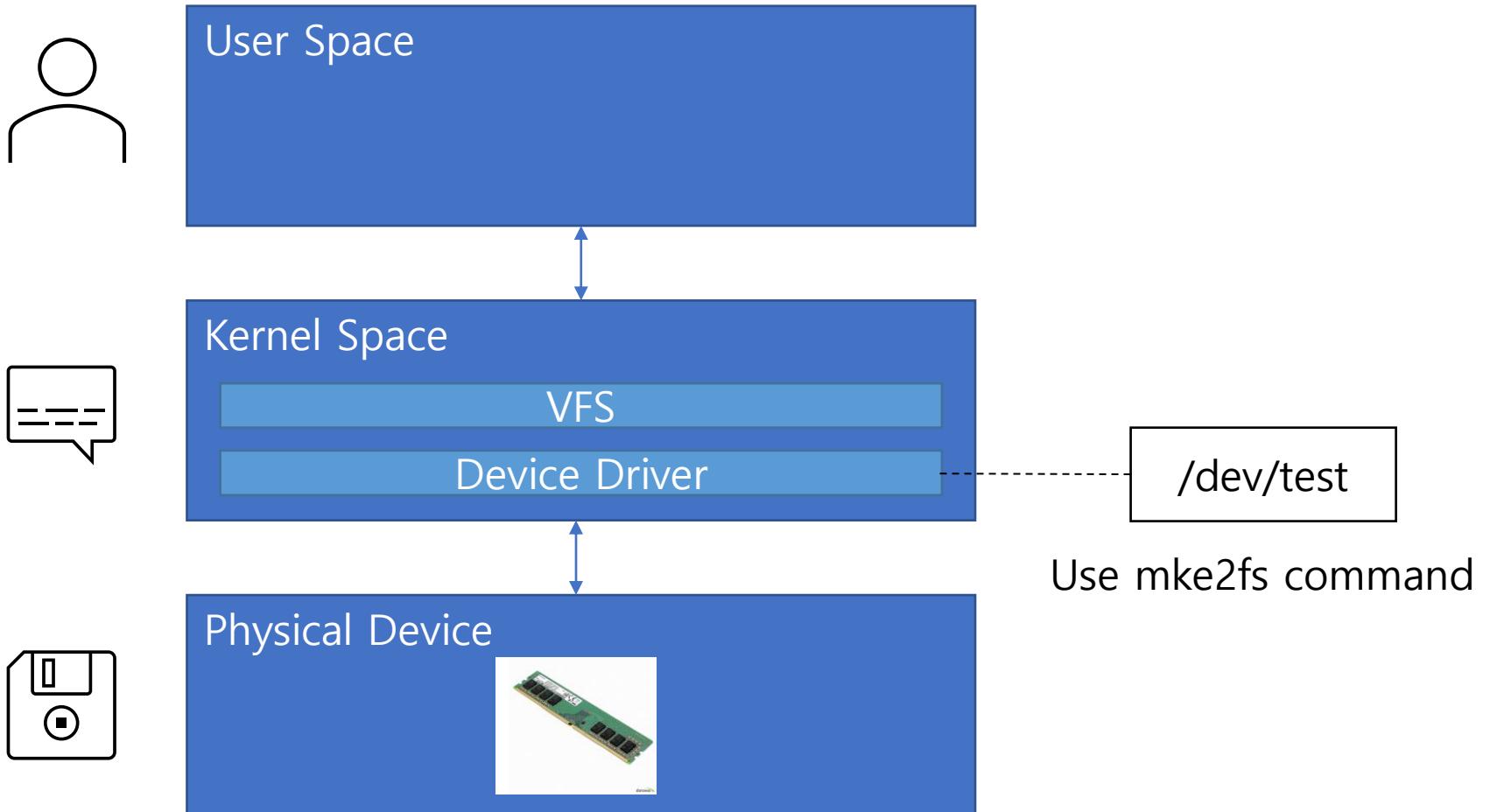
What is Device Driver for File-System?



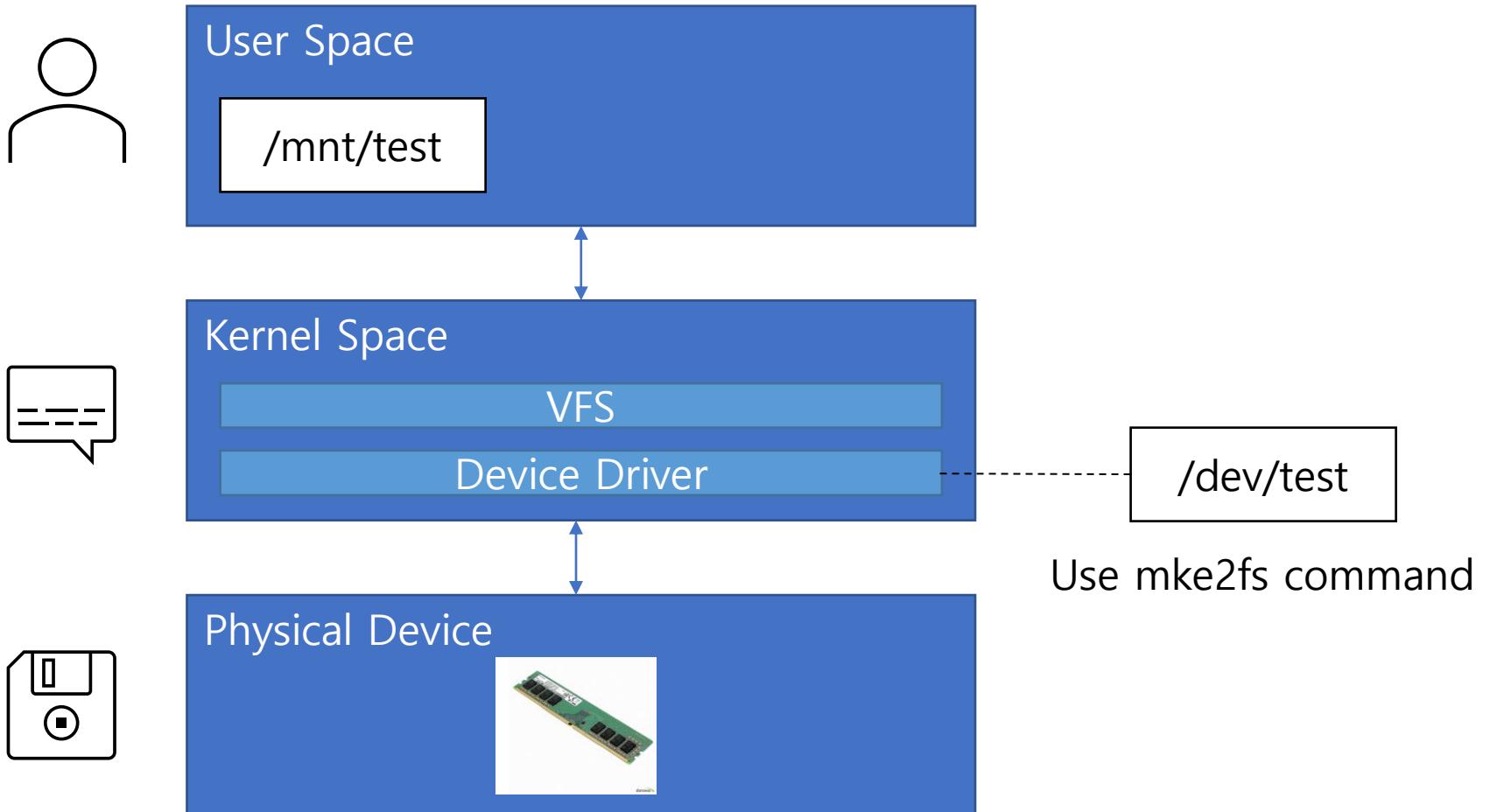
What is Device Driver for File-System?



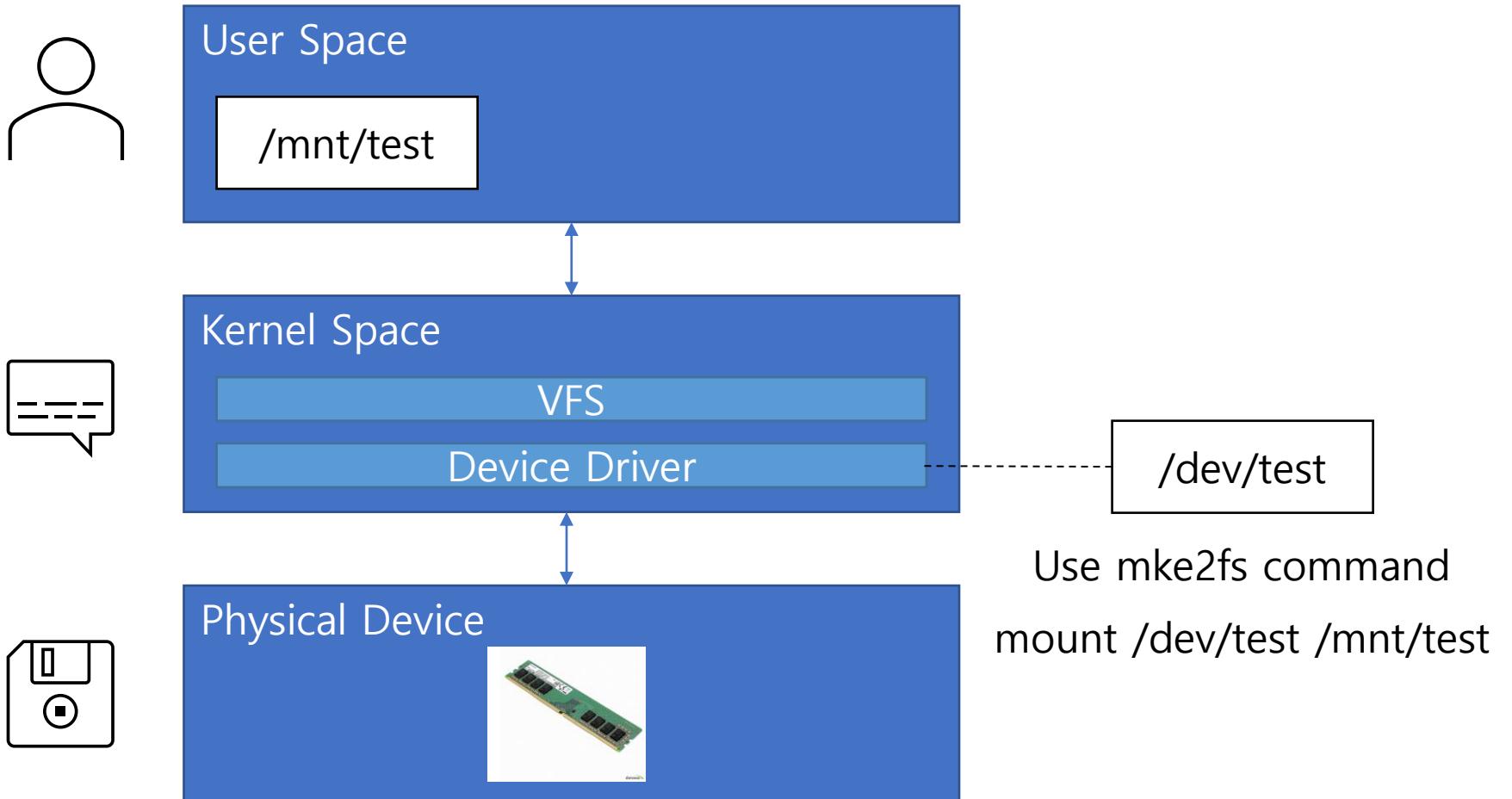
What is Device Driver for File-System?



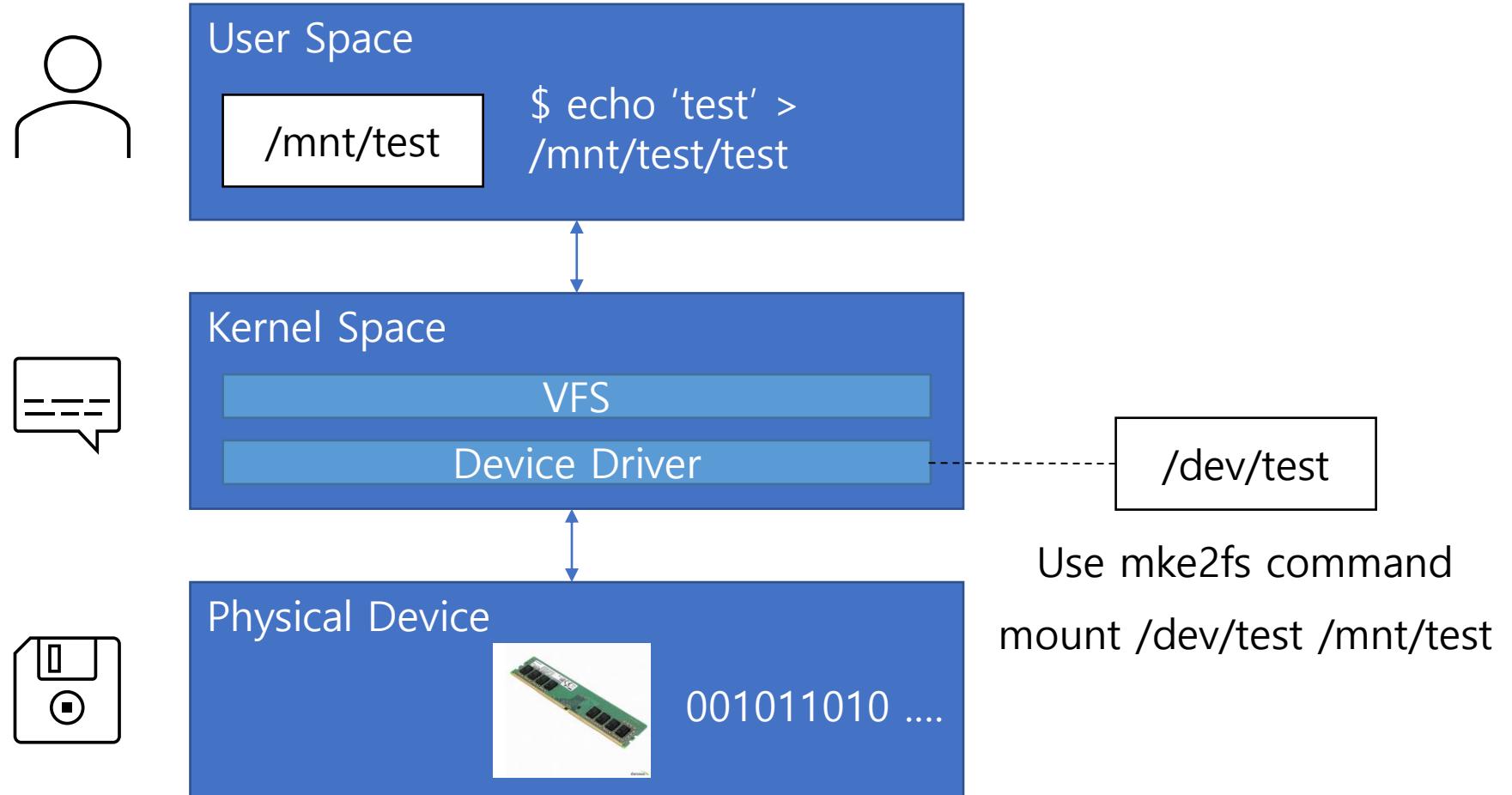
What is Device Driver for File-System?



What is Device Driver for File-System?



What is Device Driver for File-System?



환경 세팅

듀얼부팅

듀얼부팅(Dual boot)?

- PC에 2개 이상의 여러 운영 체제를 설치하는 것
- 컴퓨터 전원을 켜 때 시동할 운영 체제를 선택



부트로더(Boot loader)?

- 여러 개의 운영 체제로 시동할 수 있게 만드는 프로그램
- 예) NTLDR, LILO, GRUB
- 이 중 우리는 GRUB를 사용

/* Windows 10 및 11 기반의 PC에서 리눅스 배포판인 Ubuntu 22.04 LTS를 Dual boot할 예정 */

듀얼부팅

설치 전 준비

1. 최소 4GB 이상의 USB
2. 최소 40GB 이상의 드라이브 용량
3. Rufus 프로그램
4. Ubuntu 22.04 LTS.iso 파일
5. Wifi가 아닌 LAN(이더넷) 연결

설치 과정

1. BitLocker 해제 및 BIOS 모드 확인
2. 디스크 할당
3. Rufus로 USB에 ISO파일 쓰기
4. BIOS 옵션 변경 및 Ubuntu 설치
5. Ubuntu 설정 및 파티션 설정
6. 기타 설정

듀얼부팅

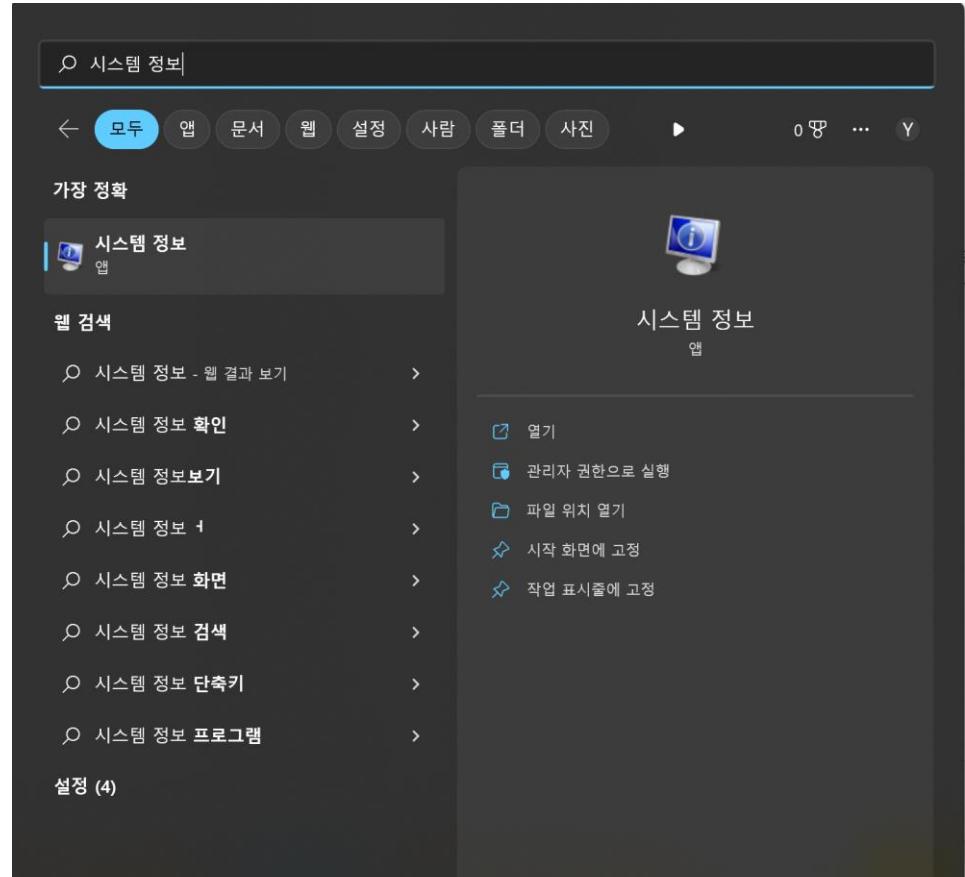
1. BitLocker 해제 및 BIOS 모드 확인

1. 설정에서 BitLocker 암호화가 설정되어 있다면 해제

⇒ 설정되어 있으면 설치 후 Windows로 부팅 시 복구 키 요구
하며 잠김

2. 시스템 정보에 들어가 BIOS 모드가 UEFI로 되어있는지 확인

⇒ UEFI인지 아닌 지에 따라 추후 방법이 다름



BIOS 모드

UEFI

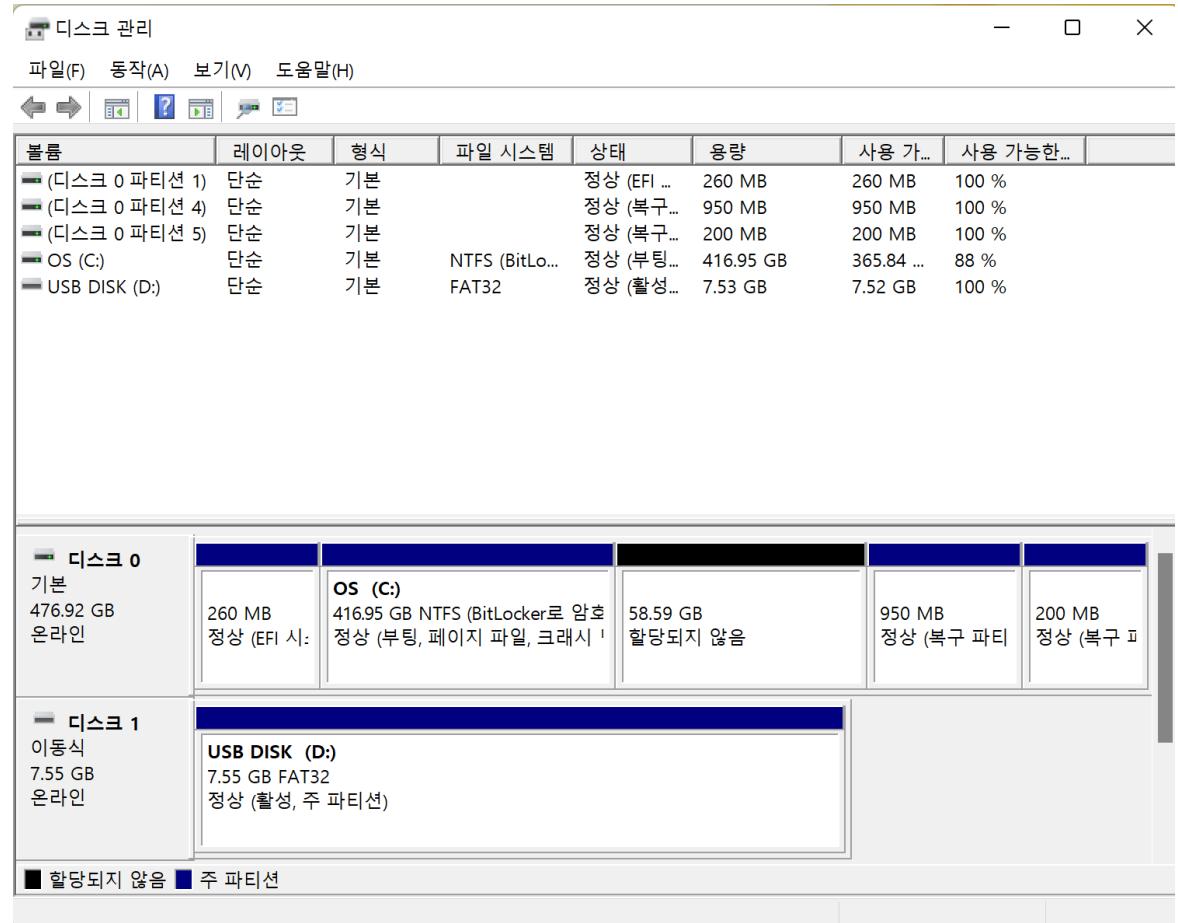
듀얼부팅

2. 디스크 할당

1. 디스크 관리에 들어가 C: 오른쪽 클릭 후 볼륨 축소

2. 할당할 용량 부여

⇒ 60GB의 경우 60000



듀얼부팅

3. Rufus로 USB에 ISO 파일 쓰기

1. USB를 PC에 꽂고 Rufus 프로그램을 실행

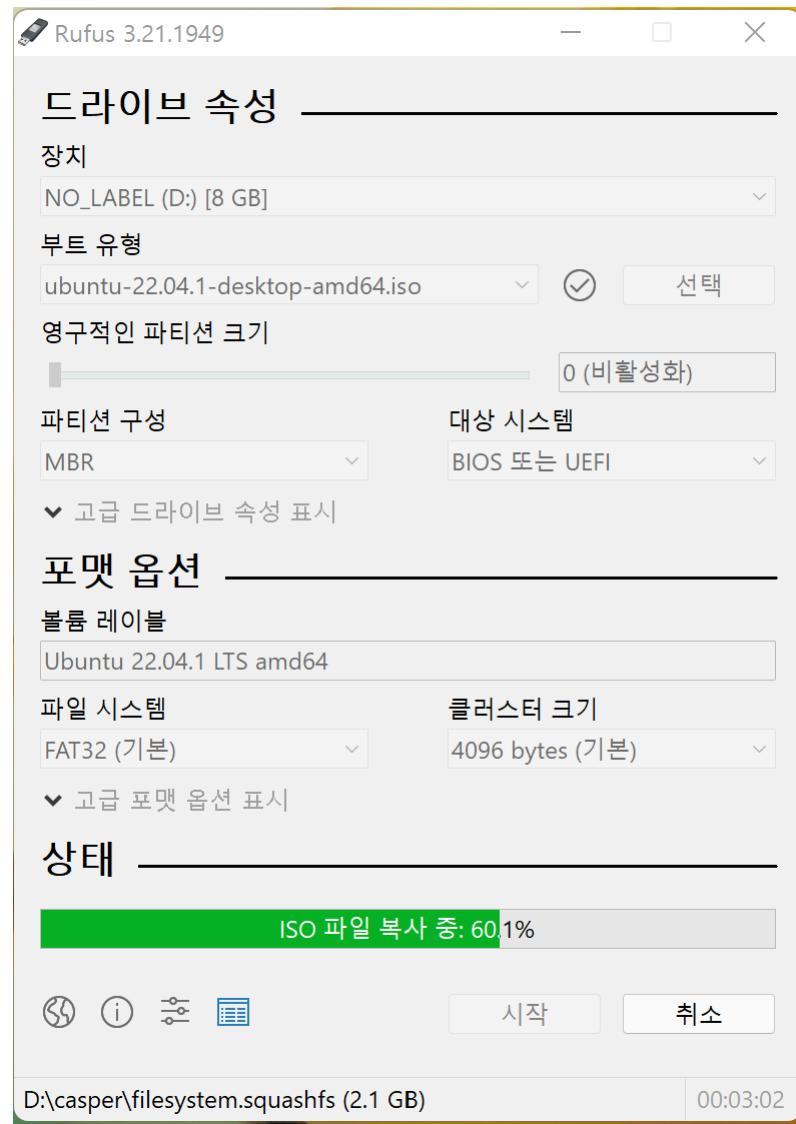
2. USB 장치와 부트 유형에 iso파일을 선택

3. 파티션 구성 및 대상 시스템 설정

⇒ BIOS 모드가 UEFI인 경우 GPT, UEFI(CSM 지원 안함) 선택

⇒ BIOS 모드가 Legacy인 경우 MBR, BIOS 또는 UEFI 선택

4. 파일 시스템은 FAT32, 클러스터 크기 기본 선택 -> 시작 클릭



듀얼부팅

4. BIOS 옵션 변경 및 Ubuntu 설치

1. 재부팅 후 BIOS 창으로 진입

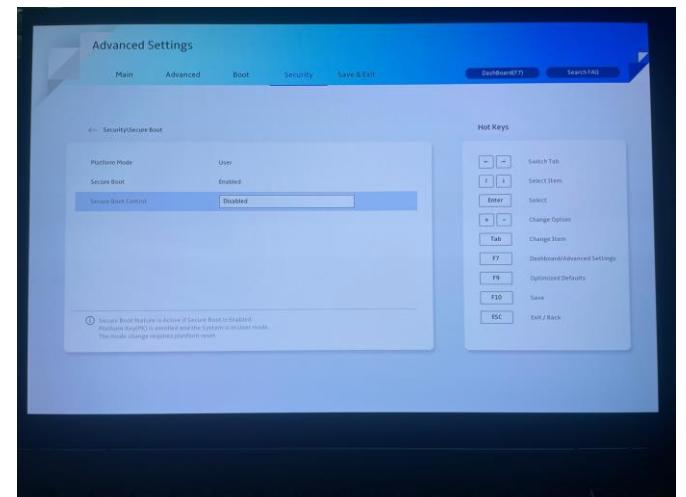
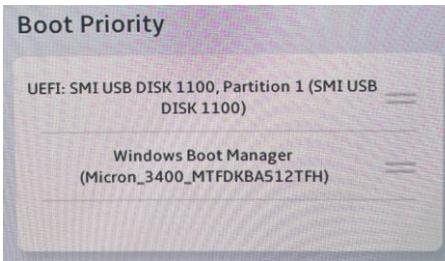
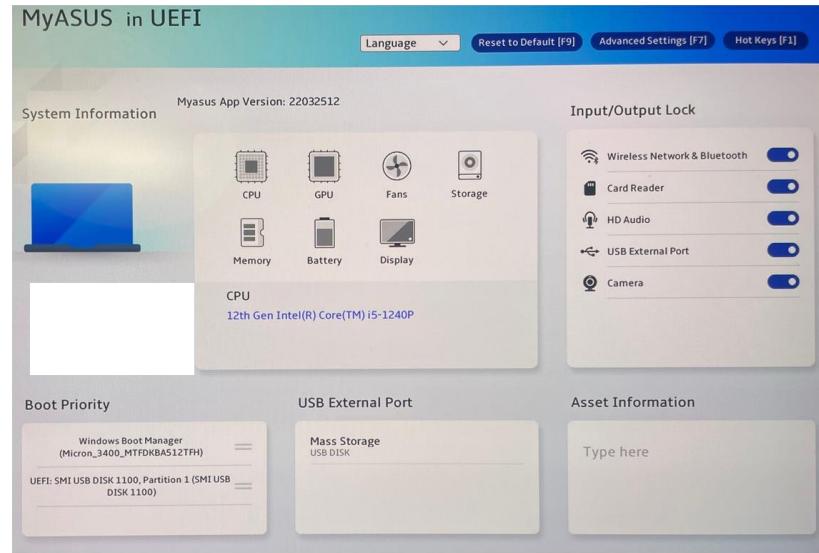
⇒ 제조사마다 BIOS 진입 키가 다를 수 있음

2. Boot Priority(부팅 순서)를 USB가 먼저 오도록 변경

3. Secure Boot이 설정되어 있다면 해제(Enabled -> Disabled)

⇒ 해제해야 외부에서 받은 인증되지 않은 OS 설치 가능

4. Save & Exit 후 자동 재부팅

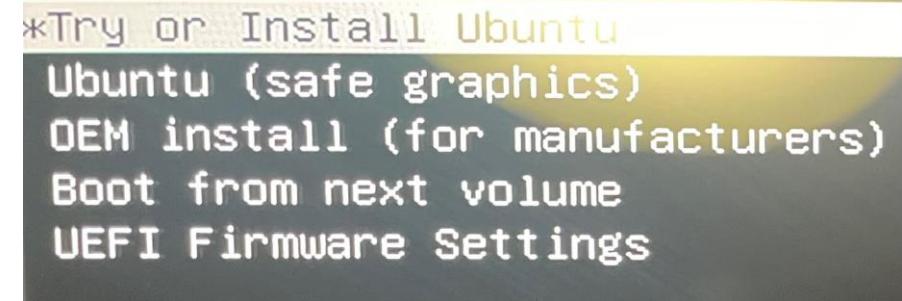


듀얼부팅

4. BIOS 옵션 변경 및 Ubuntu 설치

5. GRUB 창이 뜨면 Try or Install Ubuntu 선택

6. Ubuntu 로고가 뜨며 설치 로딩



듀얼부팅

5. Ubuntu 설정 및 파티션 설정

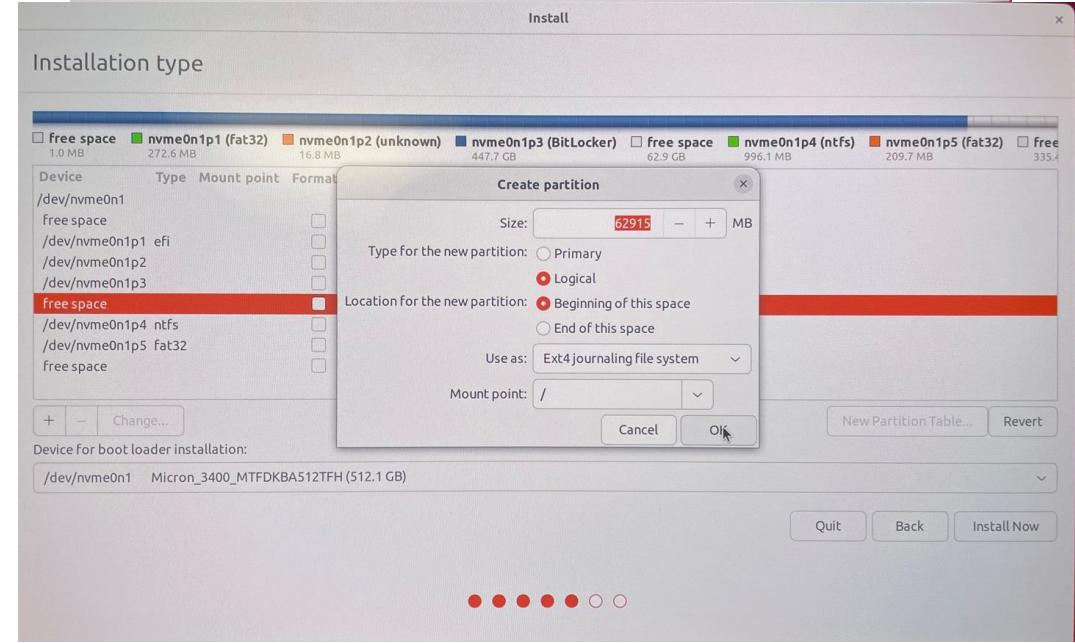
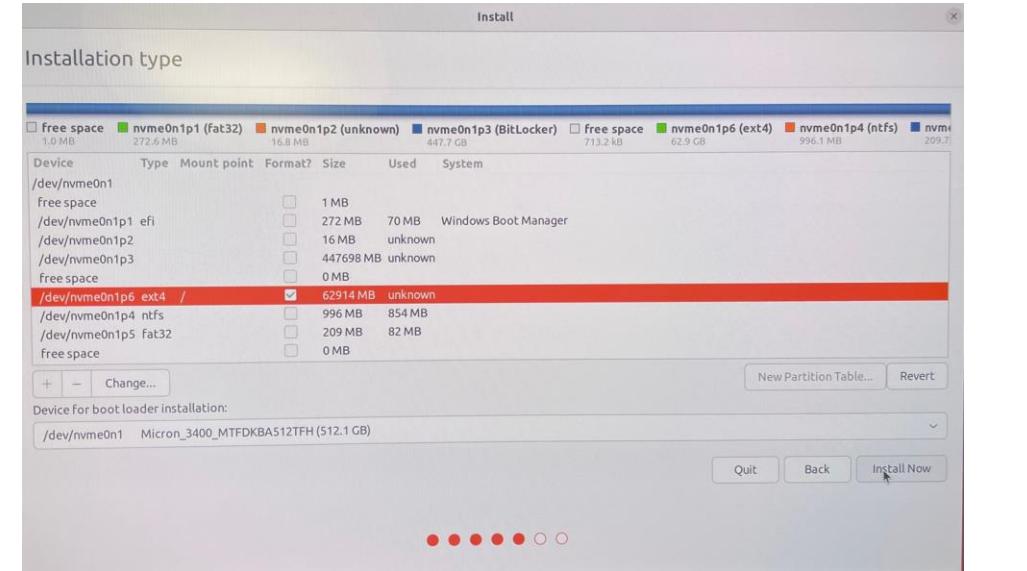
1. Try Ubuntu와 Install Ubuntu 중 Install Ubuntu 클릭
2. 언어 설정 English 선택 -> Normal Installation 선택 ->
Download updates while installing ubuntu 선택 -> Install third-party software...선택 -> Continue
3. 설치타입 설정 -> Something else 선택

듀얼부팅

5. Ubuntu 설정 및 파티션 설정

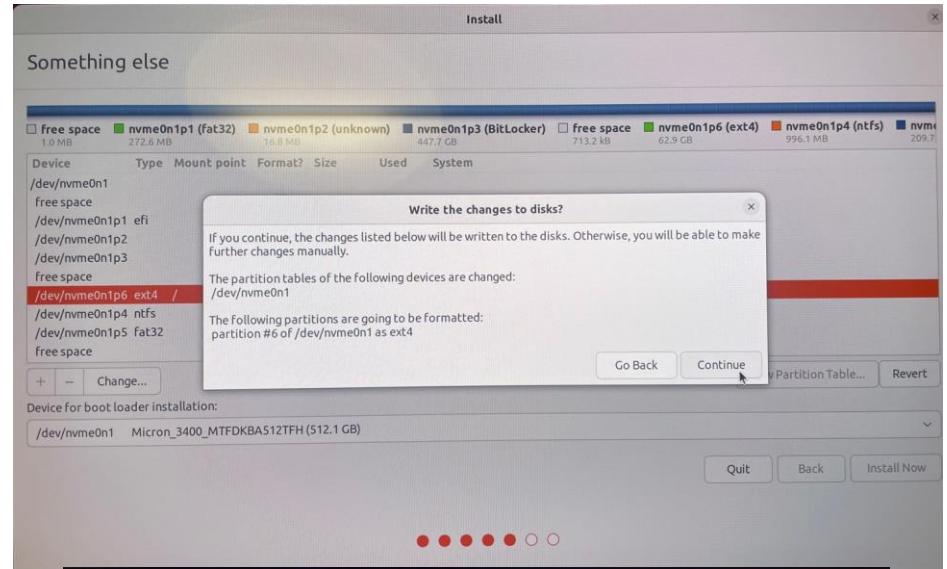
4. free space 선택 -> + 클릭 -> size는 RAM 크기 두배 권장 ->
Logical 선택 -> Beginning of this space 선택 -> Swap area 선택 -> Ok (Swap partition 생성)

5. free space 선택 -> + 클릭 -> size 그대로 -> 주파티션이 다 쓰이고 있다면 Logical, 남은 경우 Primary 선택 -> Beginning of this space 선택 -> ext4 journaling file system -> / 선택 -> Ok (Root partition 생성)



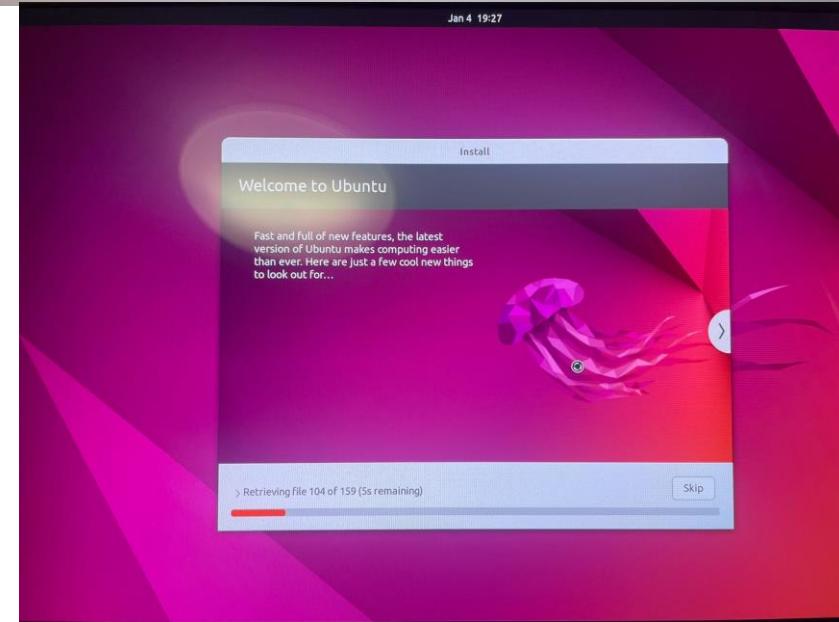
듀얼부팅

5. Ubuntu 설정 및 파티션 설정



6. space가 할당되면 클릭 -> Install Now 클릭 -> Continue

7. Install 창이 뜨면 설치 진행



듀얼부팅

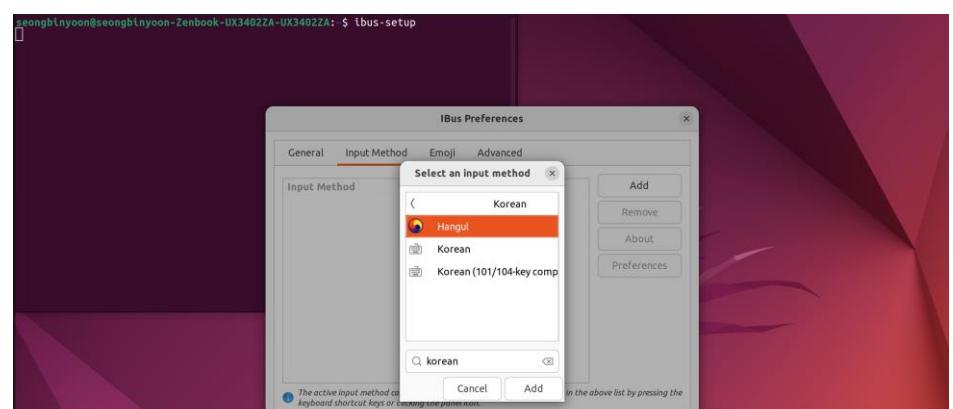
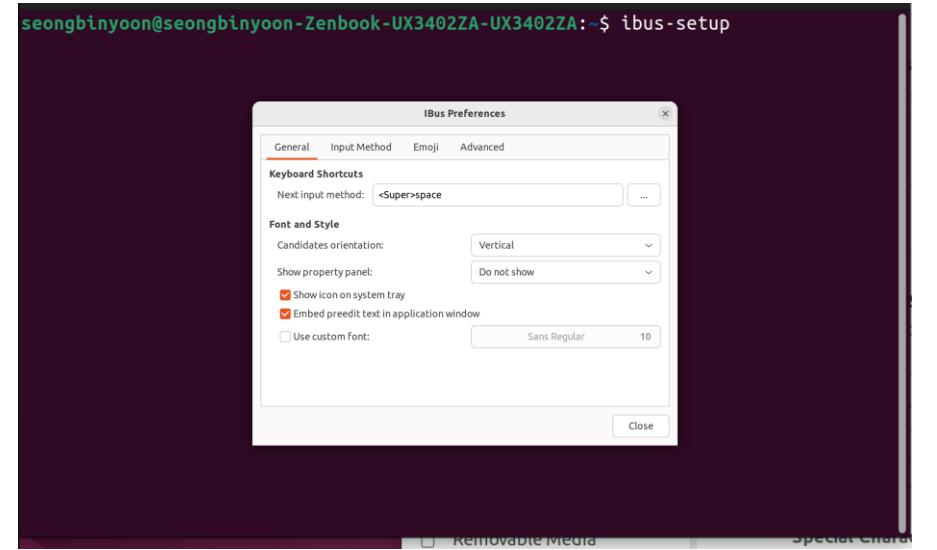
6. 기타 설정

1. 설치 완료 후 업데이트 및 재부팅

⇒ ~\$ sudo apt-get update
⇒ ~\$ reboot

2. 설치 완료 후 한글 설정

⇒ 터미널(Ctrl + Alt + T) -> ~\$ ibus-setup 입력 -> Input Method
-> Add -> Korean -> Hangul -> Add

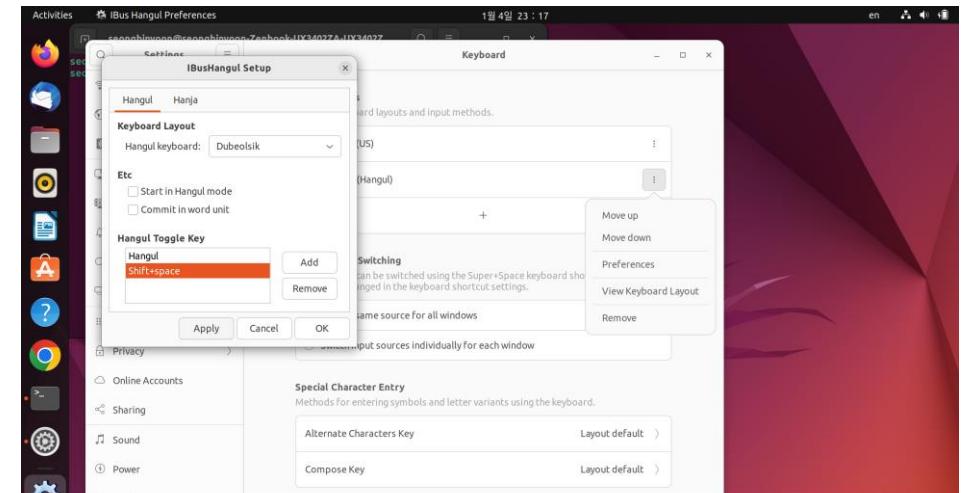


듀얼부팅

6. 기타 설정

3. Setting -> Keyboard -> English(US) 삭제 -> Korean(Hangul)
오른쪽 클릭 -> Preferences -> 원하는 키 설정 -> Ok

4. 추가적으로 Google Chrome, Notion, Slack 등을 설치 가능
⇒ Chrome에서 deb파일 다운로드 -> dpkg 명령어로 설치
⇒ snapd 명령어로 설치



리눅스 커널 컴파일

운영 체제(Operating System)?

- Hardware와 Software resource를 관리하는 프로그램
- Kernel, Shell, System Program으로 이루어짐
 - ⇒ Kernel: OS의 핵심, 하드웨어를 통제하는 프로그램
 - ⇒ Shell: User interface
 - ⇒ System Program: Kernel과 Shell을 작동하게 해주는 프로그램

리눅스 커널(Linux Kernel)?

- 리눅스 운영체제의 주요 구성 요소
- 컴퓨터 하드웨어와 프로세스를 잇는 핵심 인터페이스
 - ⇒ 이 두 가지가 최대한 효과적으로 통신하게 함

리눅스 커널 컴파일

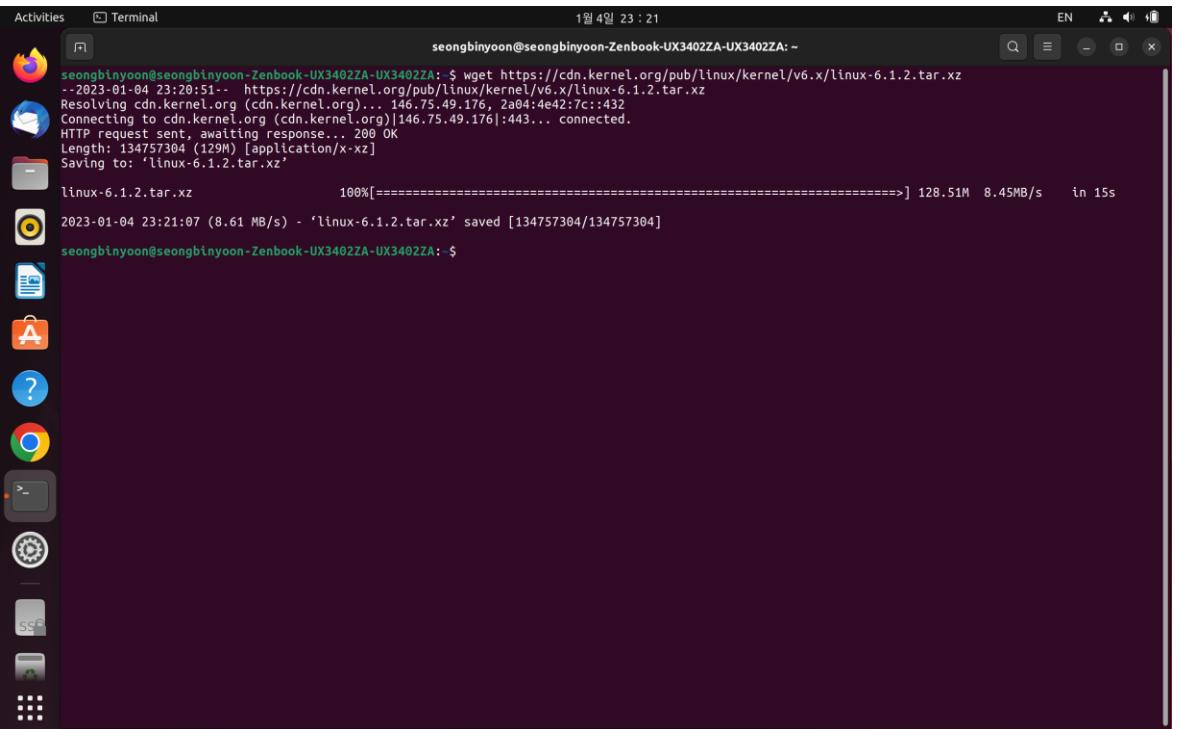
커널 컴파일 과정

1. 커널 파일 다운로드
2. 소스코드 추출
3. 필요한 패키지 설치
4. Configuration 파일을 .config 파일로 복사 및 적용
5. 커널 빌드
6. 모듈 및 커널 설치
7. Initramfs 및 부트로더 GRUB 업데이트
8. 재부팅 후 커널 버전 확인

리눅스 커널 컴파일

1. 커널 파일 다운로드

- wget 명령어를 사용하여 리눅스 커널 다운로드
⇒ ~\$ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1.2.tar.xz
- 받고자 하는 버전 숫자를 정확히 입력



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and contains the following text:

```
Activities Terminal 1월 4일 23:21 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: $ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1.2.tar.xz
- 2023-01-04 23:20:51-- https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1.2.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 146.75.49.176, 2a04:4e42:7c::432
Connecting to cdn.kernel.org (cdn.kernel.org)|146.75.49.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 134757304 (129M) [application/x-xz]
Saving to: 'linux-6.1.2.tar.xz'

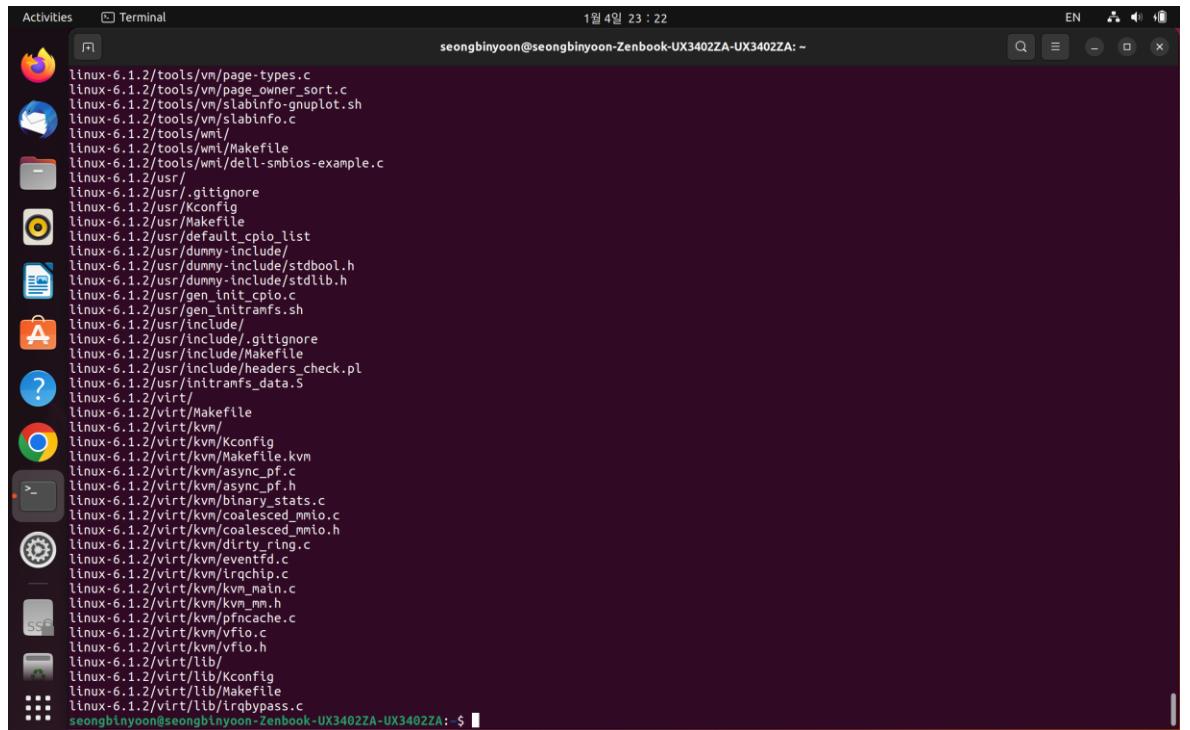
linux-6.1.2.tar.xz      100%[=====] 128.51M  8.45MB/s   in 15s
2023-01-04 23:21:07 (8.61 MB/s) - 'linux-6.1.2.tar.xz' saved [134757304/134757304]
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: $
```

The terminal window is titled "Terminal" and shows the command "wget" being run to download the Linux kernel source code. The progress bar indicates the download is complete at 100%. The desktop interface includes a dock with icons for various applications like a browser, file manager, and terminal.

리눅스 커널 컴파일

2. 소스코드 추출

- tar 명령어를 사용해 소스코드를 추출
⇒ ~\$ tar xvf linux-6.1.2.tar.xz

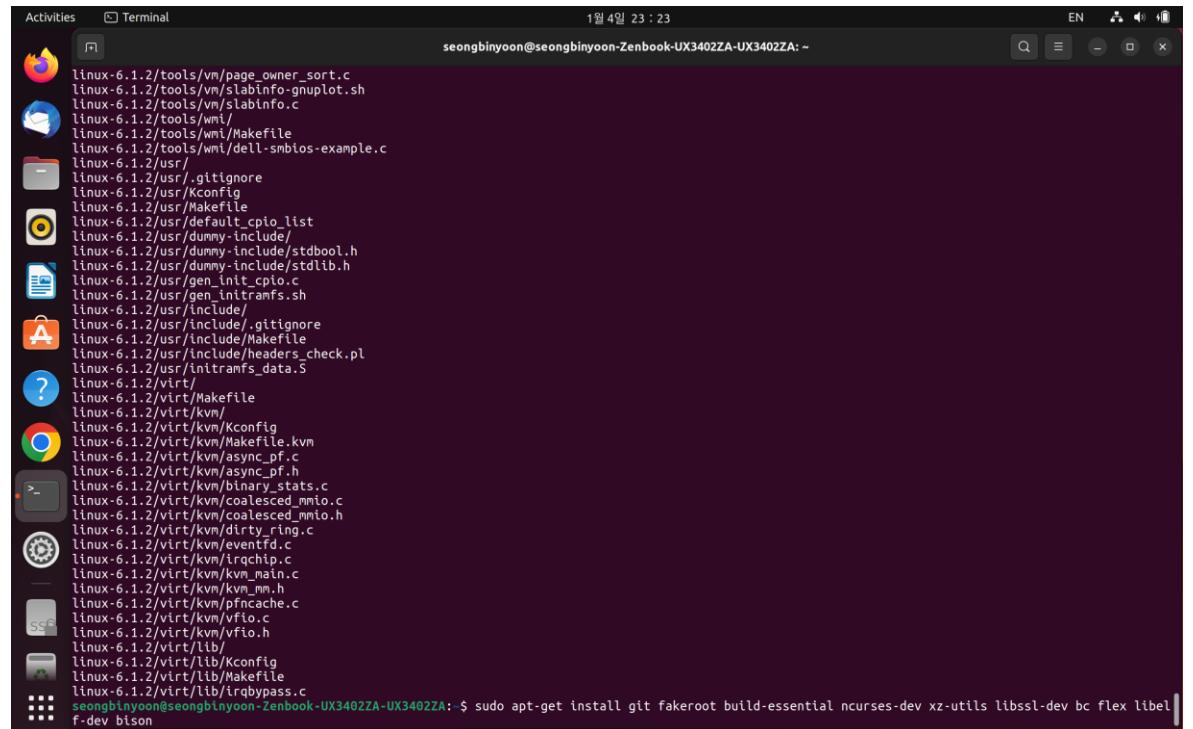


```
Activities Terminal 1월 4일 23:22 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~
linux-6.1.2/tools/vm/page-types.c
linux-6.1.2/tools/vm/page_owner_sort.c
linux-6.1.2/tools/vm/slabInfo-gnuplot.sh
linux-6.1.2/tools/vm/slabInfo.c
linux-6.1.2/tools/wml/
linux-6.1.2/tools/wml/Makefile
linux-6.1.2/tools/wml/dell-smbios-example.c
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/stdlib.h
linux-6.1.2/usr/gen_init_cpio.c
linux-6.1.2/usr/gen_intransfs.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/intransfs_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_mm.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: ~
```

리눅스 커널 컴파일

3. 필요한 패키지 설치

- 커널을 설치하는데 필요한 패키지를 설치
⇒ ~\$ 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 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/wml/
linux-6.1.2/tools/wml/Makefile
linux-6.1.2/tools/wml/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/stdcblib.h
linux-6.1.2/usr/gen_init_cpio.c
linux-6.1.2/usr/gen_intrfs.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/include/nitrofs_data.S
linux-6.1.2/virt/
linux-6.1.2/virt/Kconfig
linux-6.1.2/virt/kvm/
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_mio.c
linux-6.1.2/virt/kvm/coalesced_mto.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/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
```

리눅스 커널 컴파일

4. Configuration 파일을 .config 파일로 복사 및 적용

- 커널 디렉토리로 이동

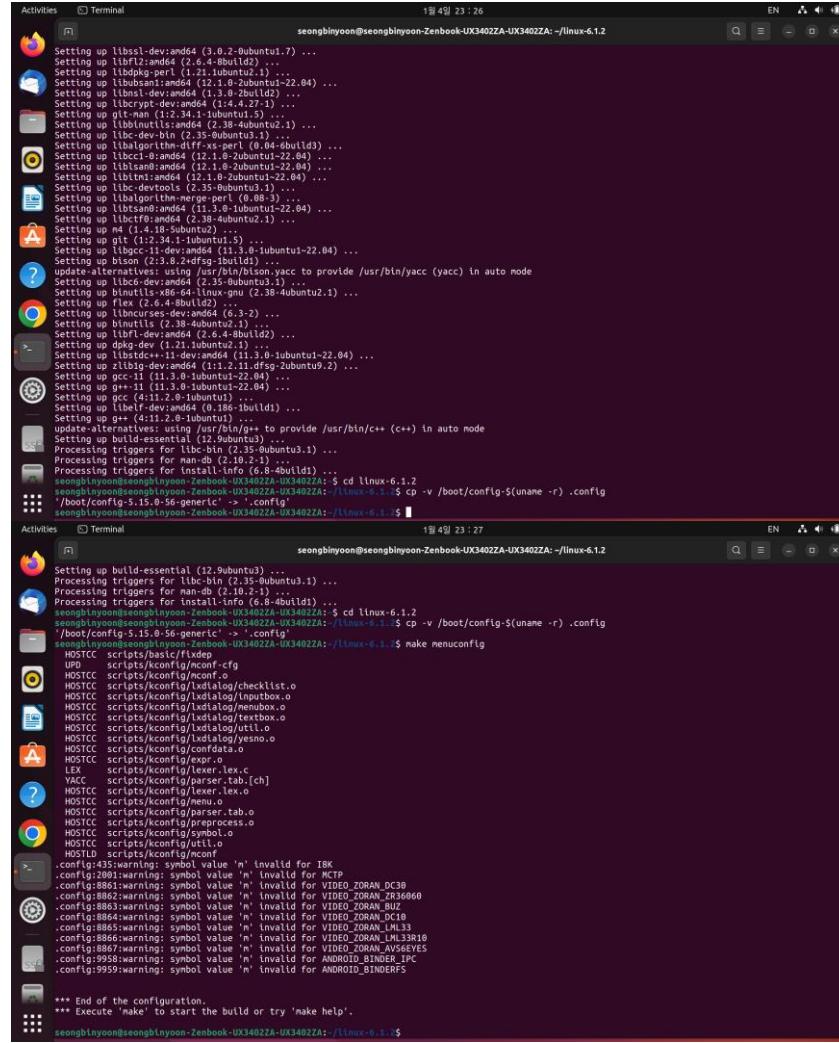
⇒ ~\$ cd linux-6.1.2

- 디렉토리의 configuration 파일을 .config 파일로 복사

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

- configuration 파일에 변경사항을 적용

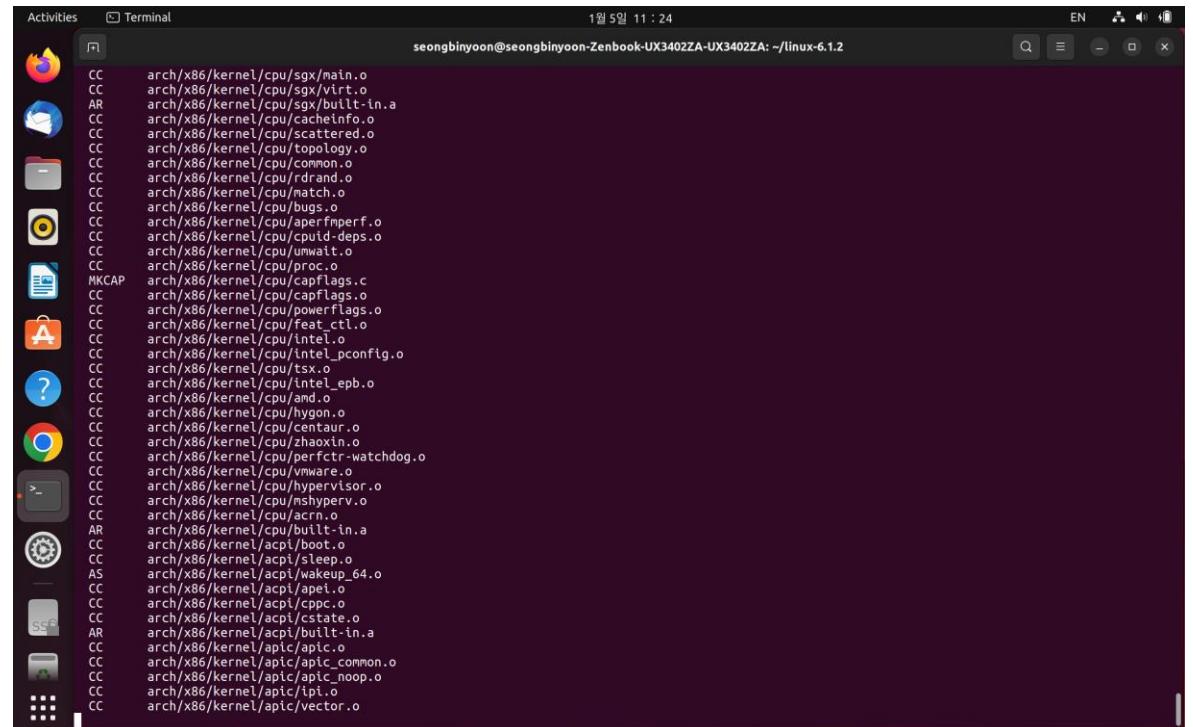
⇒ ~\$ make menuconfig



리눅스 커널 컴파일

5. 커널 빌드

- 커널을 빌드
⇒ ~\$ make



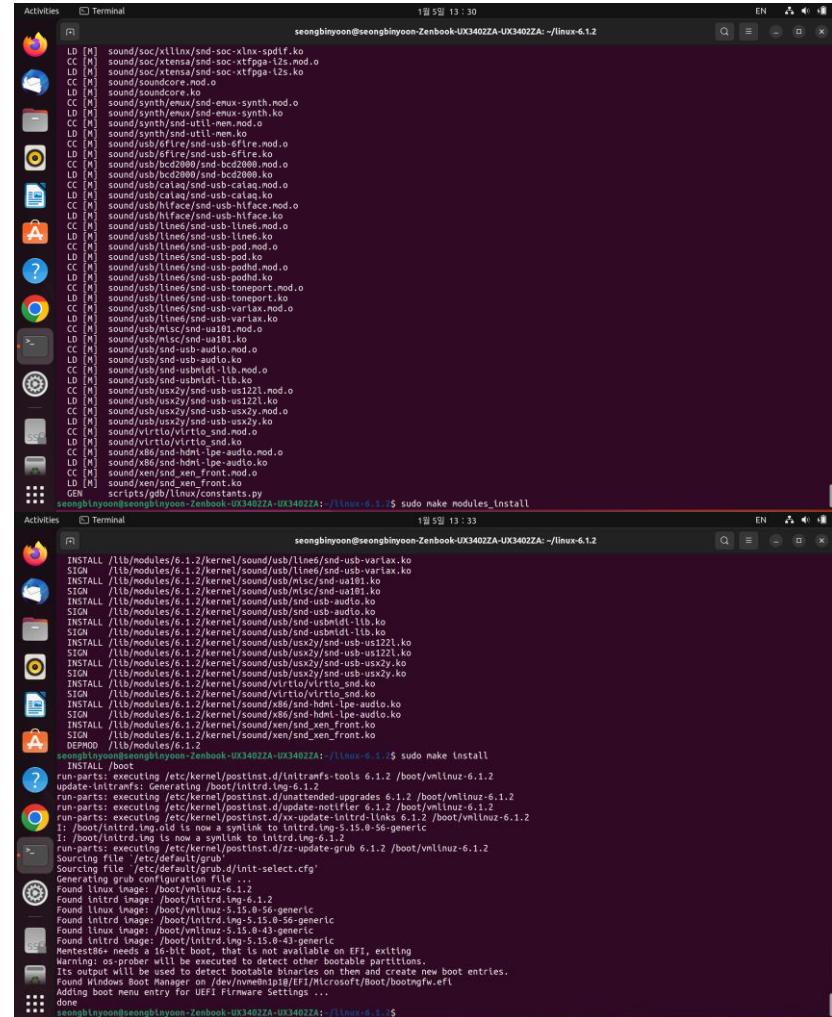
The screenshot shows a Linux desktop environment with a dark theme. A terminal window is open in the top right corner, displaying the command "make" being run. The output of the compilation process is shown, listing numerous object files (".o") and assembly files (".S") being processed by the compiler (CC) and linker (AR). The terminal window has a title bar with "Activities" and "Terminal", and a status bar at the bottom indicating the date and time: "1월 5일 11:24" and the user's name: "seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2". The desktop background shows various application icons.

```
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/rdrand.o
CC arch/x86/kernel/cpu/natch.o
CC arch/x86/kernel/cpu/bugs.o
CC arch/x86/kernel/cpu/aperfmperf.o
CC arch/x86/kernel/cpu/cpuid-deps.o
CC arch/x86/kernel/cpu/umwait.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/ttsx.o
CC arch/x86/kernel/cpu/intel_epb.o
CC arch/x86/kernel/cpu/amd.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/perfctr-watchdog.o
CC arch/x86/kernel/cpu/vmware.o
CC arch/x86/kernel/cpu/hypervisor.o
CC arch/x86/kernel/cpu/nshyperv.o
CC arch/x86/kernel/cpu/acrn.o
AR arch/x86/kernel/cpu/built-in.a
CC arch/x86/kernel/acpi/boot.o
CC arch/x86/kernel/acpi/sleep.o
AS arch/x86/kernel/acpi/wakeup_64.o
CC arch/x86/kernel/acpi/apel.o
CC arch/x86/kernel/acpi/cppc.o
CC arch/x86/kernel/acpi/cstate.o
AR arch/x86/kernel/acpi/built-in.a
CC arch/x86/kernel/acpi/apic.o
CC arch/x86/kernel/acpi/apic_common.o
CC arch/x86/kernel/acpi/apic_noop.o
CC arch/x86/kernel/acpi/ipi.o
CC arch/x86/kernel/acpi/vector.o
```

리눅스 커널 컴파일

6. 모듈 및 커널 설치

- 모듈을 설치
⇒ ~\$ sudo make modules_install
 - 커널을 설치
⇒ ~\$ sudo make install



리눅스 커널 컴파일

7. Initramfs 및 부트로더 GRUB 업데이트

- Initramfs를 업데이트

⇒ ~\$ sudo update-initramfs –c –k 6.1.2

- 부트로더 GRUB를 업데이트

⇒ ~\$ sudo update-grub

리눅스 커널 컴파일

8. 재부팅 후 커널 버전 확인

- 재부팅
⇒ ~\$ reboot
- 커널 버전 확인
⇒ ~\$ uname -mrs

리눅스 커널 컴파일

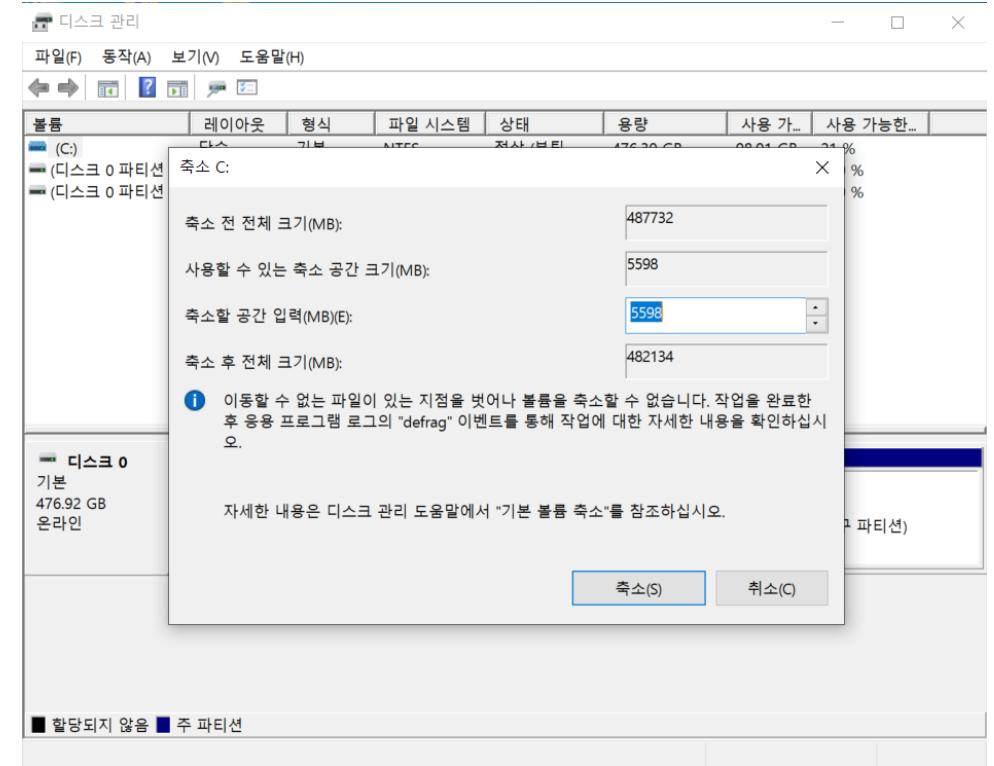
트러블 슈팅

- 문제점 1

: 파티션 볼륨 축소 불가

- 문제점 2

: 우분투 멀티부팅 설치 후 GRUB2 부트로더 실행 불가



리눅스 커널 컴파일

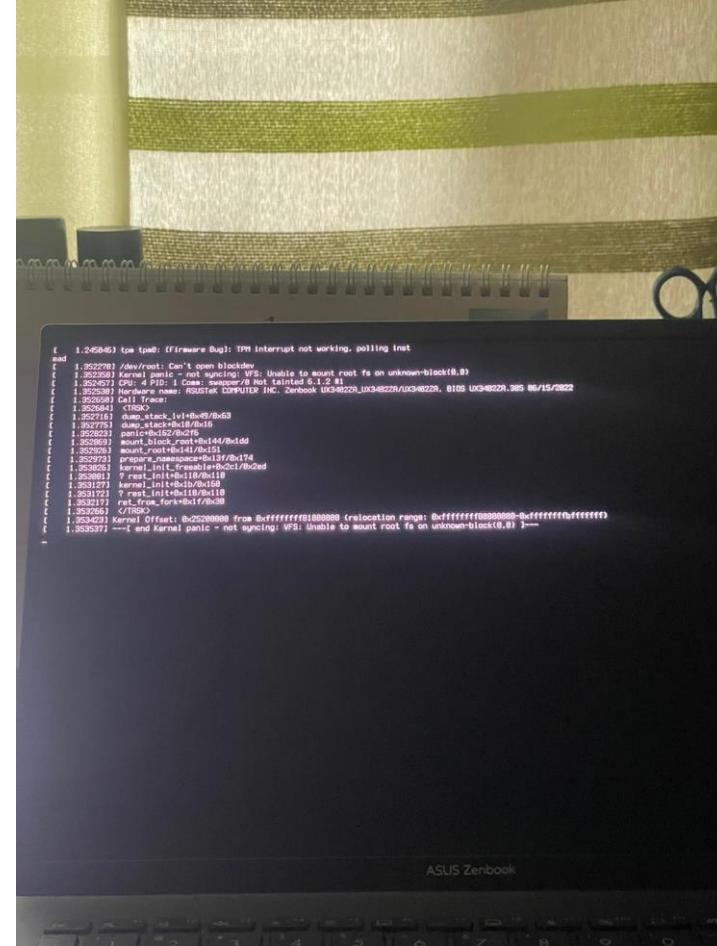
트러블 슈팅

- ### - 문제점 3

: 커널 패닉(Bad shim signature, not syncing: VFS: Unable to mount root fs on unknown-block(0,0))

- ## - 문제점 4

: 커널 패닉(out of memory)



리눅스 커널 컴파일

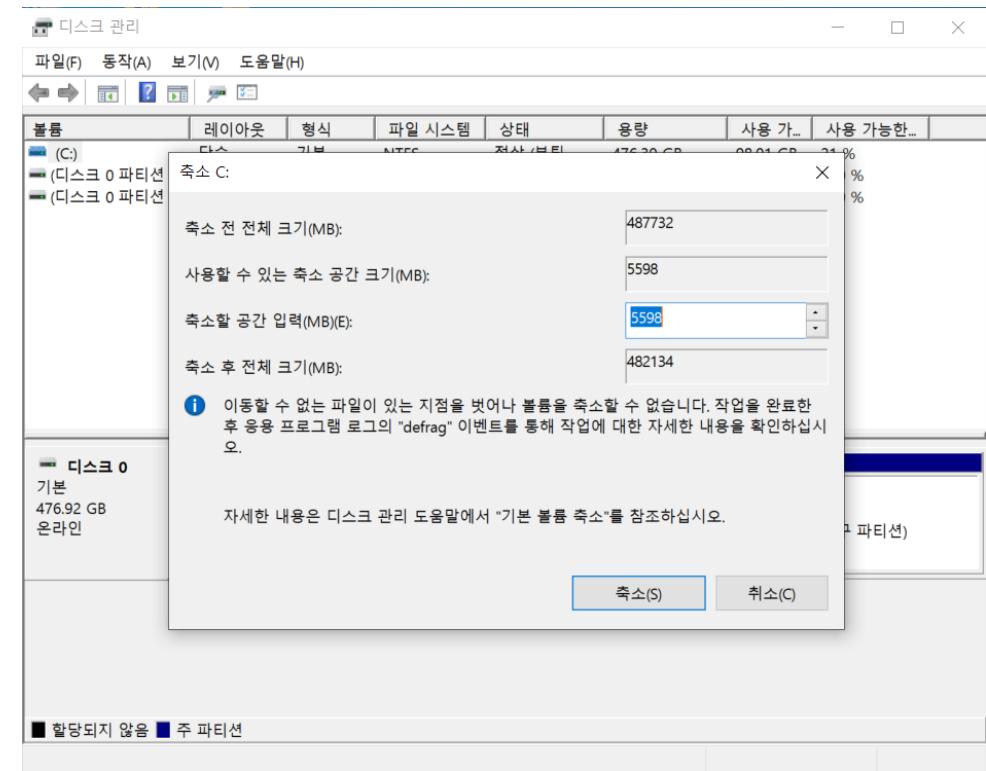
트러블 슈팅

- 문제점 5
: 우분투OS 사용 중 프리징(화면 멈춤) 현상

리눅스 커널 컴파일

트러블 슈팅(파티션 볼륨 축소 불가)

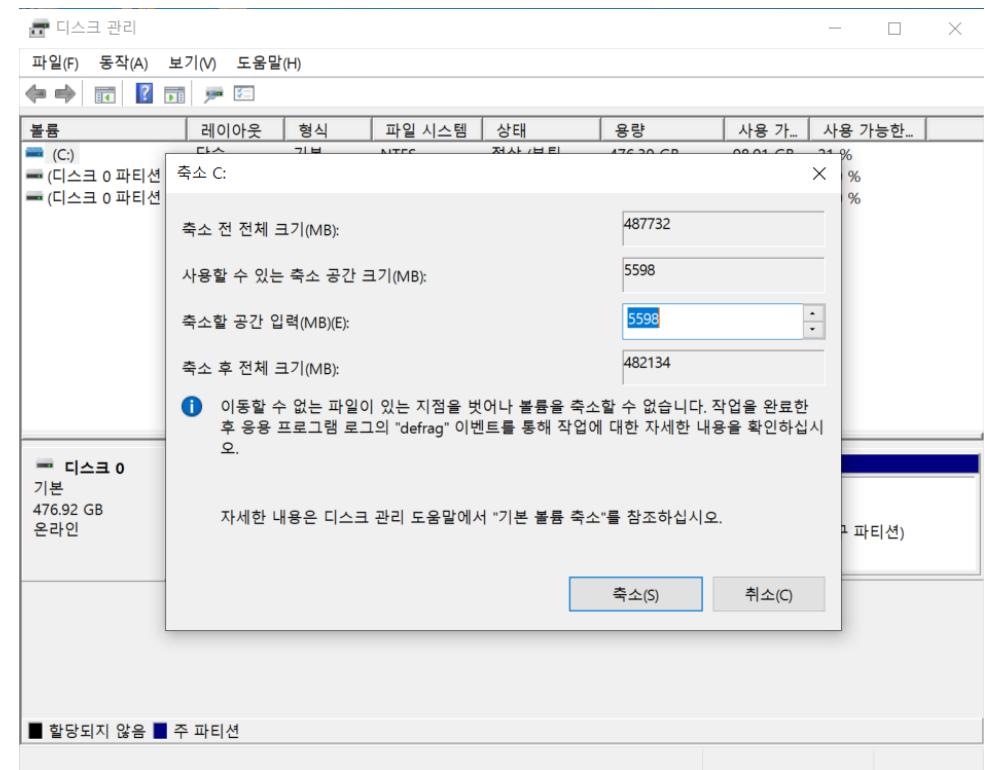
- 파티션 분할 시도 중 남은 C드라이브 용량 100GB 중 50GB만큼의 용량만큼 축소하여 리눅스 파티션에 할당 하려고 했으나, 최대로 가능한 용량은 5598MB로 볼륨 축소 불가



리눅스 커널 컴파일

트러블 슈팅(파티션 볼륨 축소 불가)

- 윈도우 시스템 보호 기능에 의한 파티션의 일부 용량 확보로 인해 생긴 현상
- 내 PC -> 속성 -> 시스템 보호 -> 드라이브를 선택함 -> 구성 -> 시스템 보호 사용 안 함 -> 확인



리눅스 커널 컴파일

트러블 슈팅(GRUB2 실행 불가)

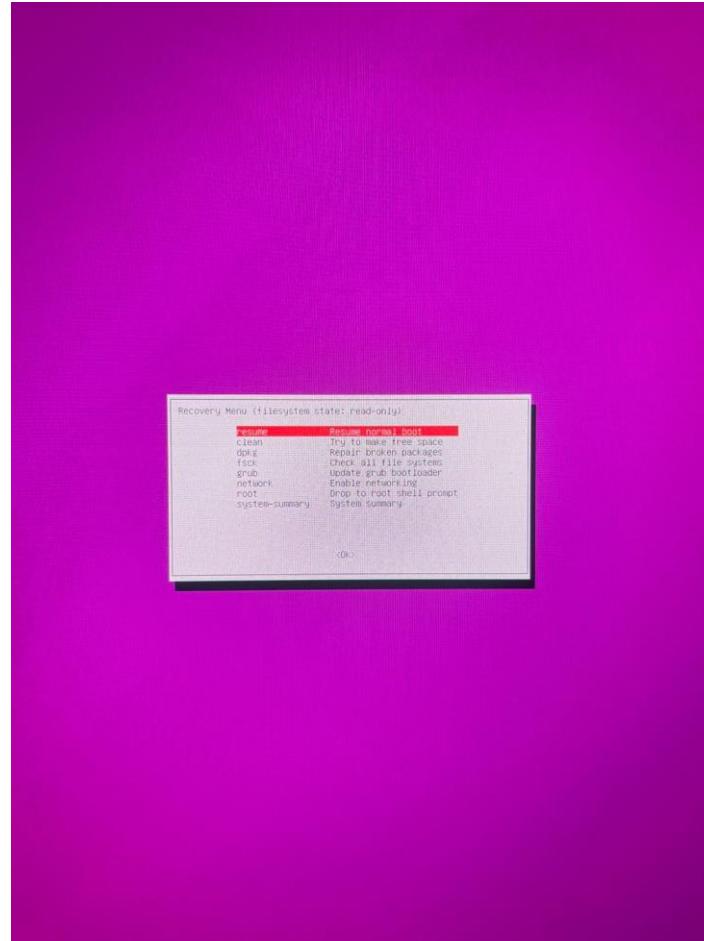
- 부트로더가 윈도우 EFI 파티션에 제대로 설치가 되지 않음
- 윈도우에서 BCD(Boot Configuration Editor)를 활용하여 부트로더 직접 설정
- 관리자 권한으로 명령프롬프트에서 다음 명령어 실행
⇒ `bcdedit /set {bootmgr} path \EFI\ubuntu\grubx64.efi`

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

리눅스 커널 컴파일

트러블 슈팅(Bad shim signature)

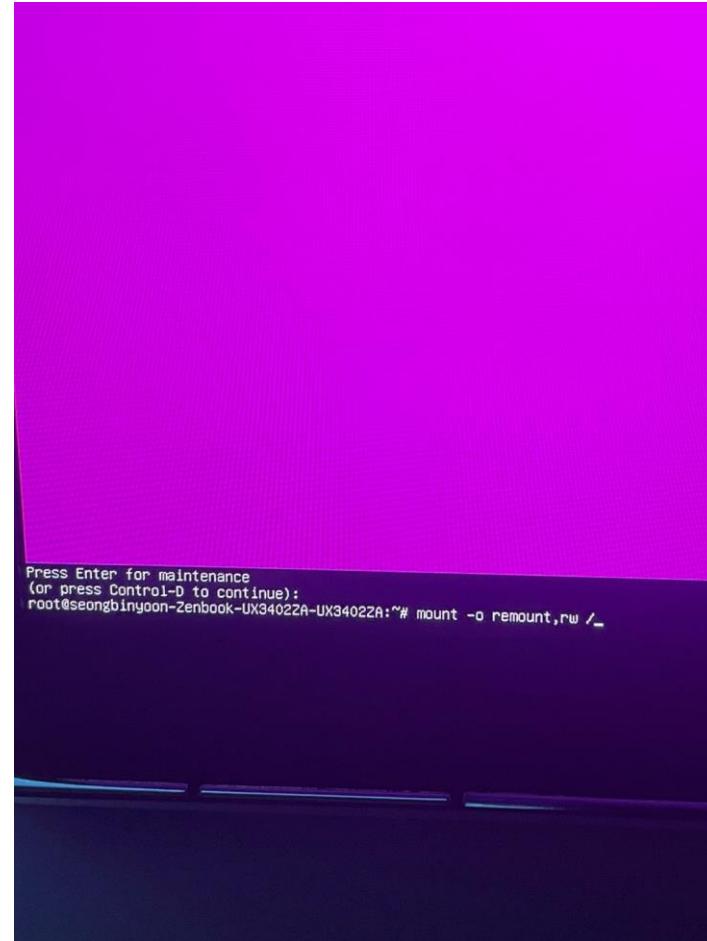
- GRUB에서 5.15.0, 6.1.2 및 6.1.2 recovery mode 모두 부팅 불가
- reboot 하여 GRUB에 들어가 ubuntu 5.15.0(이전 버전)의 recovery mode 진입
- 아래의 root의 Drop to root shell prompt를 선택



리눅스 커널 컴파일

트러블 슈팅(Bad shim signature)

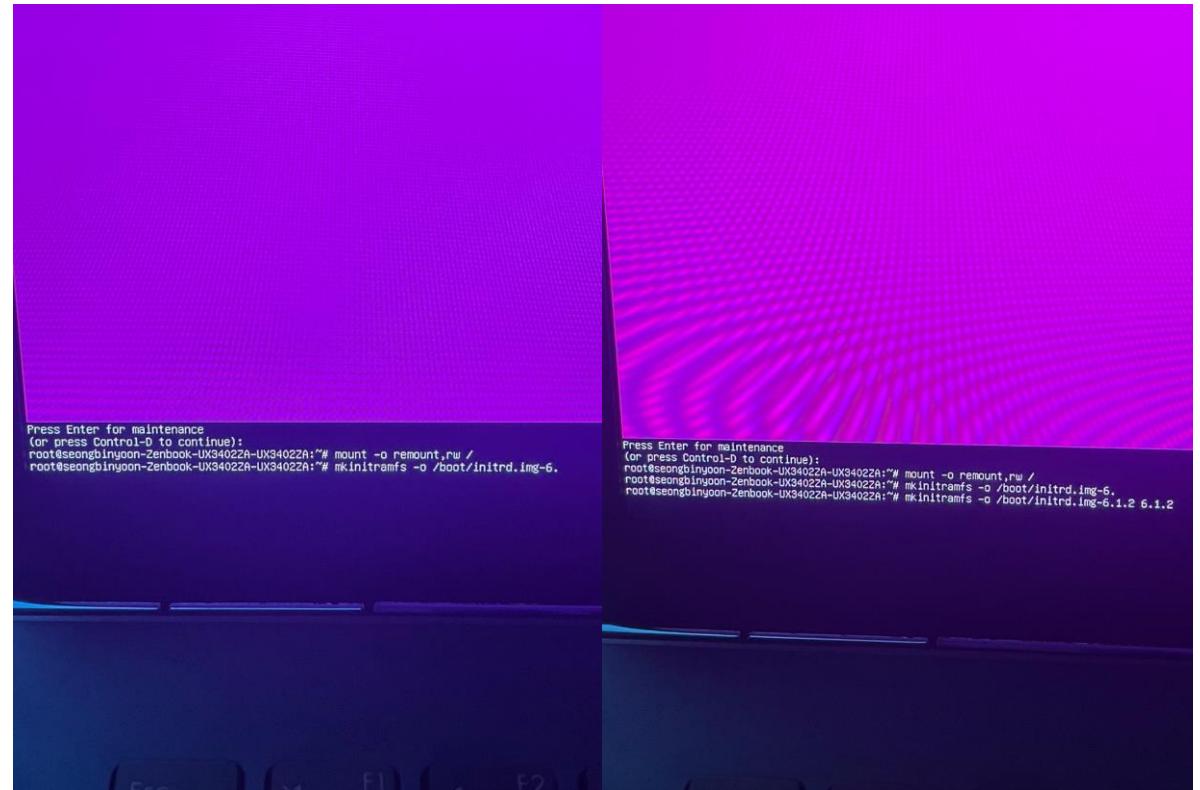
- 프롬프트에 다음 명령어를 입력하여 다시 마운트
⇒ Mount -o remount,rw /



리눅스 커널 컴파일

트러블 슈팅(Bad shim signature)

- mkinitramfs 명령어를 사용해 initrd.img를 boot에 생성
⇒ mkinitramfs -o /boot/initrd.img-6.
⇒ mkinitramfs -o /boot/initrd.img-6.1.2 6.1.2



리눅스 커널 컴파일

트러블 슈팅(Bad shim signature)

- GRUB를 업데이트
⇒ update-grub
⇒ update-grub2

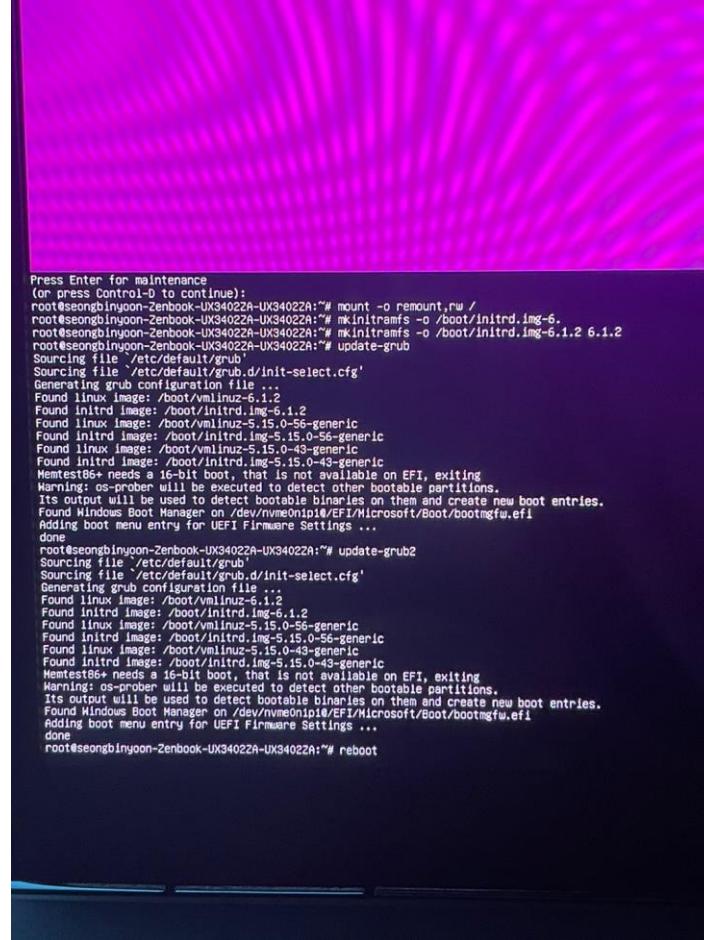
```
Press Enter for maintenance
(or press Control-D to continue):
root@seongbinyoon-Zenbook-UX34022A:~# mount -o remount,rw /
root@seongbinyoon-Zenbook-UX34022A:~# mkinitramfs -o /boot/initrd.img-6
root@seongbinyoon-Zenbook-UX34022A:~# mkinitramfs -o /boot/initrd.img-6.1.2
root@seongbinyoon-Zenbook-UX34022A:~# 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 initrd image: /boot/initrd.img-6.1.2
Found linux image: /boot/vmlinuz-5.15.0-43-generic
Found initrd image: /boot/initrd.img-5.15.0-43-generic
Found linux image: /boot/vmlinuz-5.15.0-43-generic
Found initrd image: /boot/initrd.img-5.15.0-43-generic
Warning: os-prober will be used to detect 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.efl
Adding boot menu entry for UEFI Firmware Settings ...
done
root@seongbinyoon-Zenbook-UX34022A:~# update-grub2
```

리눅스 커널 컴파일

트러블 슈팅(Bad shim signature)

- 재부팅

⇒ reboot



```
Press Enter for maintenance
(or press Control-D to continue):
root@seongbin-yoon-Zenbook-UX3402ZA-UX3402ZA:~# mount -o remount,rw /
root@seongbin-yoon-Zenbook-UX3402ZA-UX3402ZA:~# mkinitramfs -o /boot/initrd.img-6.
root@seongbin-yoon-Zenbook-UX3402ZA-UX3402ZA:~# mkinitramfs -o /boot/initrd.img-6.1.2 6.1.2
root@seongbin-yoon-Zenbook-UX3402ZA-UX3402ZA:~# update-grub
Sourcing file '/etc/default/grub.d/init-select.cfg'
Sourcing file '/etc/default/grub'
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
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/nvme0n1p1/EFI/Microsoft/Boot/bootmgfw.efi
Adding boot menu entry for UEFI Firmware Settings ...
done
root@seongbin-yoon-Zenbook-UX3402ZA-UX3402ZA:~# update-grub2
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
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/nvme0n1p1/EFI/Microsoft/Boot/bootmgfw.efi
Adding boot menu entry for UEFI Firmware Settings ...
done
root@seongbin-yoon-Zenbook-UX3402ZA-UX3402ZA:~# reboot
```

리눅스 커널 컴파일

트러블 슈팅(Bad shim signature)

- 위와 같이 진행한 결과 커널 5.15.0은 복구 완료
- 커널 6.1.2는 복구 되지 않고, 다음 오류가 발생하며 커널 패닉 지속
⇒ “Kernel panic: out of memory”

리눅스 커널 컴파일

트러블 슈팅(Out of memory)

- 커널 빌드(~\$ make modules_install 시)할 때 OS image(/boot/initrd-6.1.2) 용량이 굉장히 크게 잡힘
- 기본적으로 RAM에 할당되어 올라갈 수 있는 파일의 크기보다 크게 생성됨
- /boot/.config-6.1.2 파일을 참고해보면 CONFIG_BLK_DEV_RAM_SIZE=65536으로 65MB임을 확인할 수 있음

리눅스 커널 컴파일

트러블 슈팅(Out of memory)

- 6.1.2 initrd 파일 용량(487M)이 메모리 한도를 초과함

⇒ ~\$ du -hs /boot/initrd.img-6.1.2

- 부팅이 잘 되고 있는 5.15.0 initrd 이미지 파일의 용량은 61M로 비교적 적음

⇒ ~\$ du -hs /boot/initrd.img-5.15.0-56-generic

```
~$ 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
```

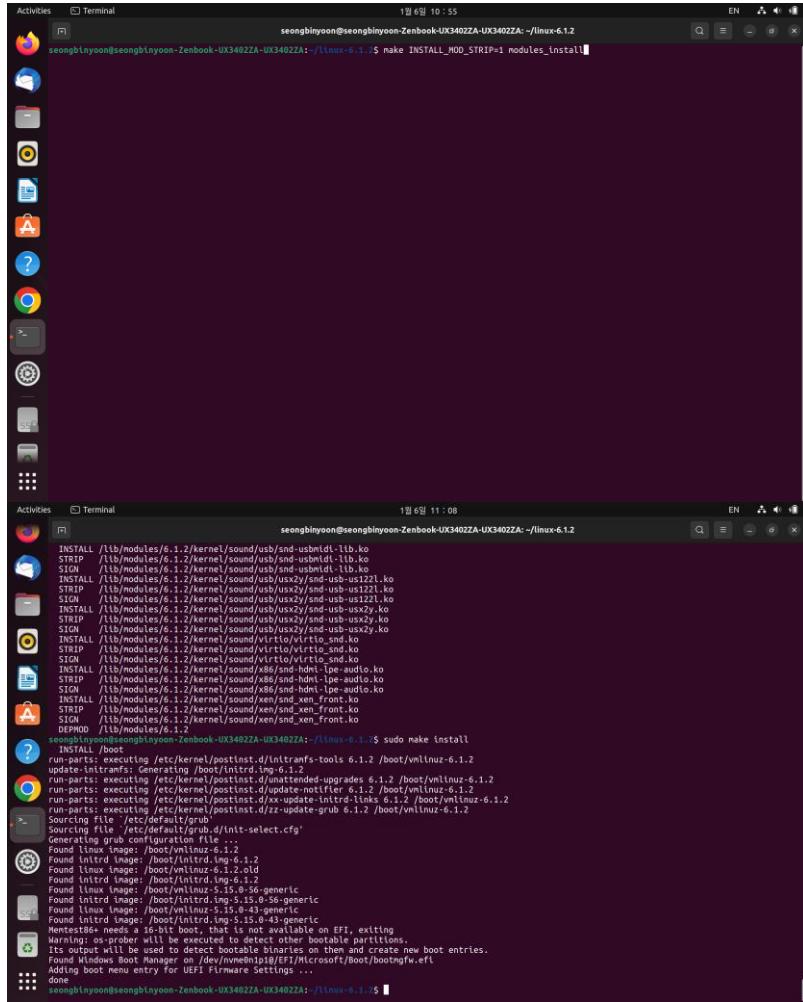
리눅스 커널 컴파일

트러블 슈팅(Out of memory)

- 아래의 명령어를 통해 STRIP 옵션이 필요없는 모듈은 이미지에 포함시키지 않게 해 사용하는 메모리의 크기 를 축소

⇒ ~\$ sudo make INSTALL_MOD_STRIP=1
modules_install

- 아래의 명령어를 통해 커널을 빌드
- ⇒ ~\$ sudo make install



```
Activities Terminal 1월 6일 10:55 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$ make INSTALL_MOD_STRIP=1 modules_install Activities Terminal 1월 6일 11:08 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2 seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA: ~/linux-6.1.2$ sudo make install
```

The terminal output for the bottom window shows the following steps:

- INSTALL /lib/modules/6.1.2/kernel/sound/usb/usbind-dt-lib.ko
- STRIP /lib/modules/6.1.2/kernel/sound/usb/usbind-dt-lib.ko
- SIGN /lib/modules/6.1.2/kernel/sound/usb/usbind-dt-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/usx2y/snd-usb-usx22l.ko
- STRIP /lib/modules/6.1.2/kernel/sound/usx2y/snd-usb-usx22l.ko
- SIGN /lib/modules/6.1.2/kernel/sound/usx2y/snd-usb-usx22l.ko
- INSTALL /lib/modules/6.1.2/kernel/sound/usx2y/snd-usb-usx2y.ko
- STRIP /lib/modules/6.1.2/kernel/sound/usx2y/snd-usb-usx2y.ko
- SIGN /lib/modules/6.1.2/kernel/sound/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/x86/snd-hdmi-lpe-audio.ko
- INSTALL /lib/modules/6.1.2/kernel/sound/xen/snd_xen.ko
- STRIP /lib/modules/6.1.2/kernel/sound/xen/snd_xen.ko
- SIGN /lib/modules/6.1.2/kernel/sound/xen/snd_xen.ko
- GRUBMD /boot/grub/grub.cfg

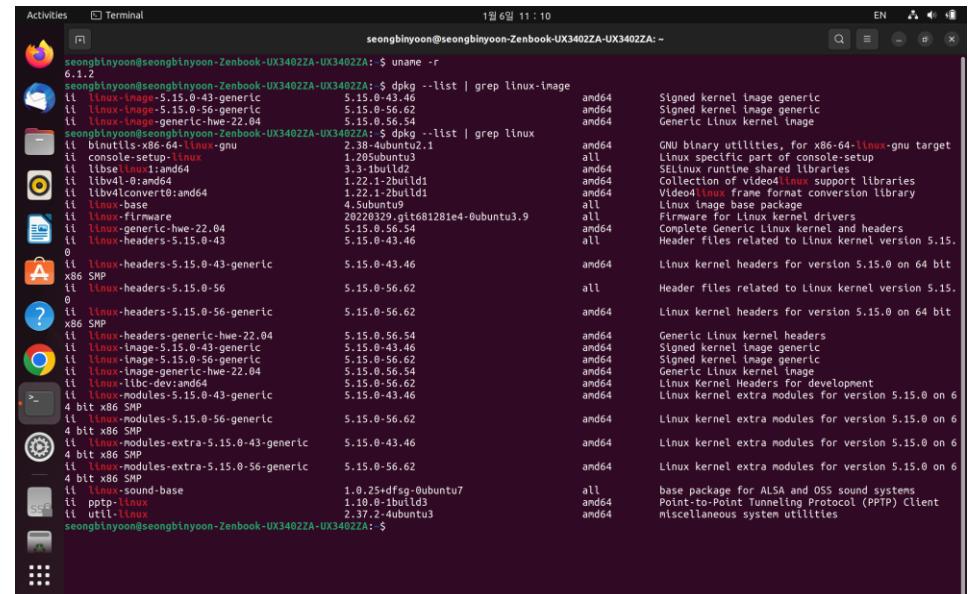
After the 'make install' command, the terminal shows the GRUB configuration process:

- Sourcing file '/etc/default/grub'
- Updating /boot/grub/grub.cfg
- Generating grub configuration file ...
- Found linux image: /boot/vmlinuz-6.1.2
- Found initrd image: /boot/initrd.img-6.1.2
- Found initrd image: /boot/initrd.img-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 initrd image: /boot/initrd.img-5.15.0-43-generic
- Found initrd image: /boot/initrd.img-5.15.0-43-generic
- 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/nvme0n1p8[EFI]/Microsoft/Boot/bootmgfw.efi
- Adding boot menu entry for UEFI Firmware Settings ...

리눅스 커널 컴파일

트러블 슈팅(Out of memory)

- 커널 6.1.2으로 재부팅
⇒ ~\$ reboot
 - 현재 커널 버전 확인
⇒ ~\$ uname -r



리눅스 커널 컴파일

트러블 슈팅(프리징 현상)

- 우분투 OS 사용 중 프리징(화면 멈춤) 현상이 일어나
마우스, 키보드 등 입력 장치 사용 불가
- NVIDIA GPU 드라이버와 OS 사이 충돌 부분 수정

리눅스 커널 컴파일

트러블 슈팅(프리징 현상)

- GRUB 설정 파일 열기

⇒ ~\$ sudo vim /etc/default/grub

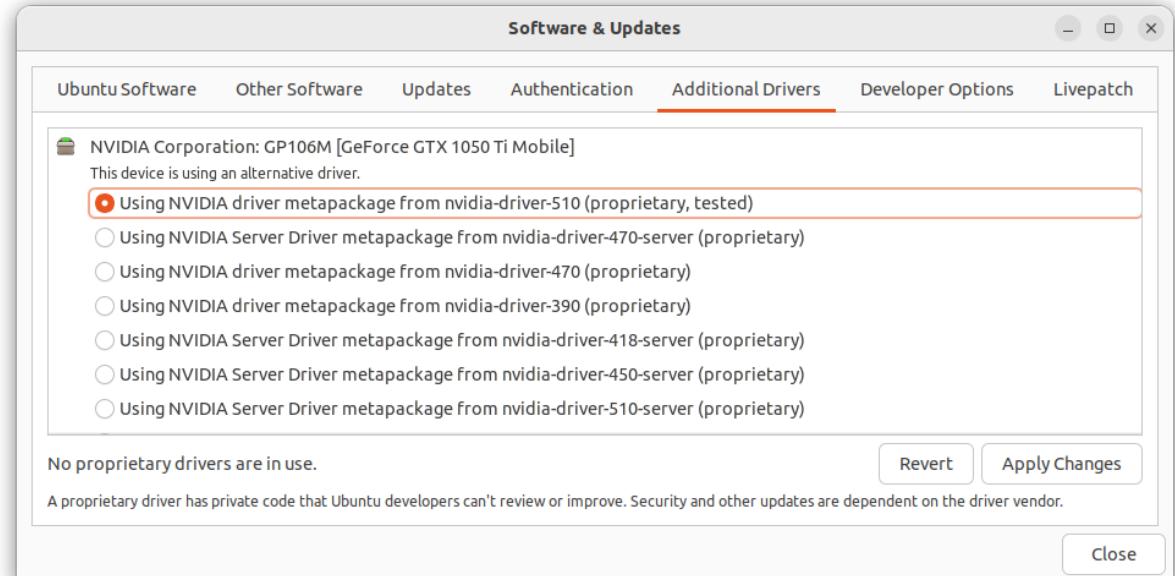
- 아래 옵션(nomodeset) 추가

⇒ GRUB_CMDLINE_LINUX_DEFAULT = "quiet splash
nomodeset"

- Software & Update -> Additional Drivers -> tested로 등록된 드라이버로 교환 -> Apply Changes -> reboot

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

JavaScript ▾
GRUB_CMDLINE_LINUX_DEFAULT = "quiet splash nomodeset"
```



KVM 구축

Linux KVM

- KVM 하이퍼바이저 설치
- 개인 PC에서 서버용 PC로 ssh 접속을 사용하여 설치
진행
- 학교의 허가를 받아 anyconnect vpn을 통해 연구실 PC
의 IP주소로 연결

KVM 환경 구축

1. ssh 접속
2. 가상화 지원 확인
3. KVM 설치 + 사용자 계정 그룹에 추가
4. X11 포워딩
5. VM 설치

KVM 구축

1. ssh 접속

- 서버 PC로 ssh 접속
⇒ ~\$ ssh name@ipaddress

KVM 구축

2. 가상화 지원 확인

- CPU에서 하드웨어 가상화를 지원하는지 확인
⇒ ~\$ egrep -c '(vmx|svm)' /proc/cpuinfo
- CPU 코어 수인 0보다 큰 수가 출력되면 가상화를 지원
- kvm-ok 유틸리티가 존재하는지 확인
⇒ ~\$ sudo kvm-ok
- 존재하지 않는다면 cpu-checker install

```
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:~$
```

KVM 구축

3. KVM 설치 + 사용자 계정 그룹에 추가

- KVM을 설치하는데 필요한 패키지 설치
⇒ ~\$ sudo apt-get update
⇒ ~\$ sudo apt install -y qemu qemu-kvm libvirt-daemon libvirt-clients bridge-utils virt-manager

```
el13@el13-HP-Pro-Tower-280-69-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 libspiceclient0:amd64 (0.39-3ubuntu1) ...
Setting up libvirt-spiceclientgtk-3.0.5:amd64 (0.39-3ubuntu1) ...
Setting up virt-viewer (7.0.2build2) ...
Setting up libvirt-daemon-system (0.8.0-1ubuntu7.4) ...
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/virtlogd-admin.socket → /lib/systemd/system/virtlogd-admin.socket.
Created symlink /etc/systemd/system/sockets.target.wants/libvirt-ro.socket → /lib/systemd/system/libvirt-ro.socket.
Created symlink /etc/systemd/system/multi-user.target.wants/libvirt-guests.service → /lib/systemd/system/libvirt-guests.service.
virtlockd.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.
Setting up libvirt-daemon dnsmasq configuration.
Processing triggers for libvirt-bin (0.8.0-1ubuntu7.6) ...
Created symlink /etc/systemd/system/multi-user.target.wants/run-qemu.mount → /lib/systemd/system/run-qemu.mount.
Setting up libvirt-cmd2 (0.3.11-2.1ubuntu4) ...
Setting up dm-event (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 (2:0.31.11-2.1ubuntu4) ...
update-intramfs: deferring update (trigger activated)
Created symlink /lib/systemd/system/systemd-target.wants/blk-availability.service → /lib/systemd/system/blk-availability.service.
Created symlink /etc/systemd/system/systemd-target.wants/lvm2-monitor.service → /lib/systemd/system/lvm2-monitor.service.
Created symlink /etc/systemd/system/syinit.target.wants/lvm2-lvmpolld.socket → /lib/systemd/system/lvm2-lvmpolld.socket.
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Processing triggers for shared-mime-info (6.8-4ubuntu1) ...
Processing triggers for install-info (6.8-4ubuntu1) ...
Processing triggers for mcmlac (3.70+mu1ubuntu1) ...
Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...
Processing triggers for intramfs-tools (0.14ubuntu13.1) ...
update-intramfs: deferring update (trigger activated)
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnomenu (3.36.0-1ubuntu3) ...
Processing triggers for libglib2.0-0:amd64 (2.72.4-0ubuntu1) ...
Processing triggers for liblbc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
el13@el13-HP-Pro-Tower-280-69-PCI-Desktop-PC: $
```

- KVM이 잘 설치되었는지 확인
⇒ ~\$ kvm --version

KVM 구축

3. KVM 설치 + 사용자 계정 그룹에 추가

- 가상화 데몬 libvirtd – daemon이 실행 중인지 확인

⇒ ~\$ sudo systemctl status libvirtd

⇒ ~\$ sudo systemctl is-active libvirtd

- 다음 명령어로 start/stop, kvm 모듈 로드 여부 확인

⇒ ~\$ sudo systemctl start libvirtd

⇒ ~\$ sudo systemctl stop libvirtd

⇒ ~\$ lsmod | grep -i kvm

```
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-gtk3.0.5:amd64 (0.39-3ubuntu1) ...
Setting up libspiceclient0:amd64 (0.39-3ubuntu1) ...
Setting up libspiceclientgdk3-0:amd64 (0.39-3ubuntu1) ...
Setting up virt-viewer (7.0.2build2) ...
Setting up libvirt-daemon-system (0.8.0-1ubuntu7.4) ...
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/virtlogd-admin.socket → /lib/systemd/system/virtlogd-admin.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.
virtlockd.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.
Setting up libvirt-daemon dnsmasq configuration.
Processing triggers for libvirt-bin (0.8.0-1ubuntu7.6) ...
Created symlink /etc/systemd/system/multi-user.target.wants/run-qemu.mount → /lib/systemd/system/run-qemu.mount.
Setting up libvirt-cmd2.03:amd64 (2.03.11-2.1ubuntu4) ...
Setting up dm-event (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 lvm (2.03.11-2.1ubuntu4) ...
update-intramfs: deferring update (trigger activated)
Created symlink /etc/systemd/system/multi-user.target.wants/blk-availability.service → /lib/systemd/system/blk-availability.service.
Created symlink /etc/systemd/system/multi-user.target.wants/lvm2-monitor.service → /lib/systemd/system/lvm2-monitor.service.
Created symlink /etc/systemd/system/syinit.target.wants/lvm2-lvmpolld.socket → /lib/systemd/system/lvm2-lvmpolld.socket.
Processing triggers for dbus (1.12.20-2ubuntu0.1) ...
Processing triggers for shared-mime-info (6.8-4ubuntu1) ...
Processing triggers for install-info (6.8-4ubuntu1) ...
Processing triggers for mdutil (3.70+mu1ubuntu1) ...
Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...
Processing triggers for intramfs-tools (0.14ubuntu13.1) ...
update-intramfs: deferring update (trigger activated)
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnomenu (3.36.0-1ubuntu3) ...
Processing triggers for libglib2.0-0:amd64 (2.72.4-0ubuntu1) ...
Processing triggers for libgb-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $
```

```
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $ lsmod | grep -i kvm
kvm_intel           368640   0
kvm                 102896   1 kvm_intel
ecl3@ecl3-HP-Pro-Tower-280-G9-PCI-Desktop-PC: $
```

KVM 구축

3. KVM 설치 + 사용자 계정 그룹에 추가

- 가상머신을 생성하고 관리하기 위해 사용자 계정을 “libvirtd” 및 “kvm” 그룹에 추가

⇒ ~\$ sudo usermod -aG libvirt \$USER

⇒ ~\$ sudo usermod -aG kvm \$USER

```
#cl3@cl3:HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo usermod -aG libvirt $USER  
#cl3@cl3:HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$ sudo usermod -aG kvm $USER  
#cl3@cl3:HP-Pro-Tower-280-G9-PCI-Desktop-PC:~$
```

- 이 후 재로그인

KVM 구축

4. X11 포워딩

- openssh-server 설치

⇒ ~\$ sudo apt install openssh-server

- sshd_config 파일 수정

⇒ ~\$ sudo vim /etc/ssh/sshd_config

- X11Forwarding no -> yes로 수정, 주석은 제거

```
# 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
```

KVM 구축

4. X11 포워딩

- ssh_config 파일 수정

⇒ ~\$ sudo vim /etc/ssh/ssh_config

- ForwardX11 및 ForwardX11Trusted 부분 no -> yes로 수정, 주석은 제거

- 다시 ssh 접속 시 -X 붙여 명령어 실행

⇒ ~\$ ssh -X <username>@<ipaddress>

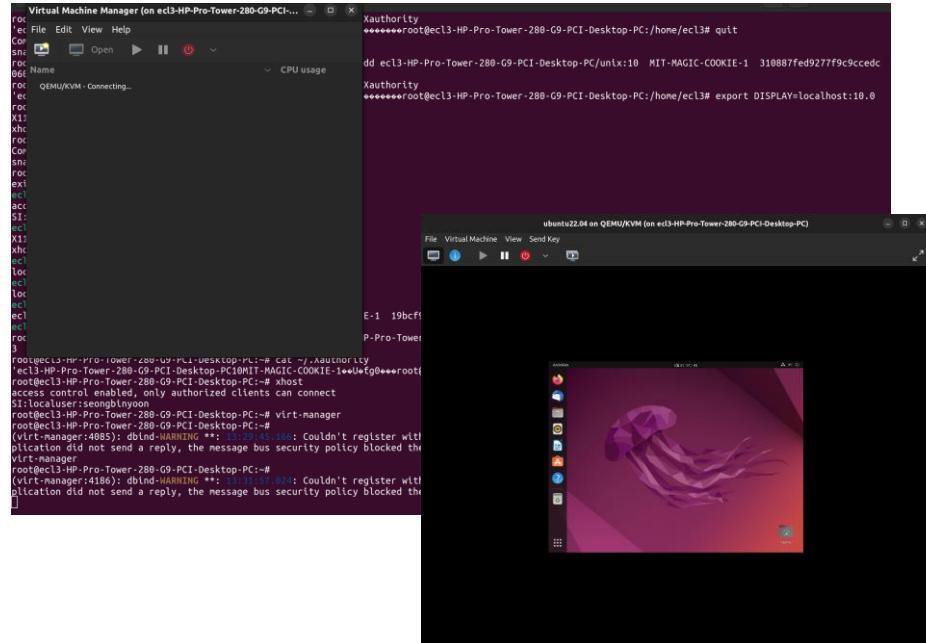
```
Host *
#   ForwardAgent no
#   ForwardX11 no
#   ForwardX11Trusted yes
```

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

KVM 구축

5. VM 설치

- virt-manager를 실행
⇒ ~\$ sudo virt-manager
- 듀얼 부팅과 마찬가지로 우분투 설치



모듈 컴파일

hello 모듈

커널 모듈(Kernel module)?

- 요청 시 커널에 로드 및 언로드할 수 있는 코드 조각
- 리눅스에서는 시스템을 재부팅할 필요 없이 런타임에 커널의 기능을 확장
- 커널 컴파일 시간 단축

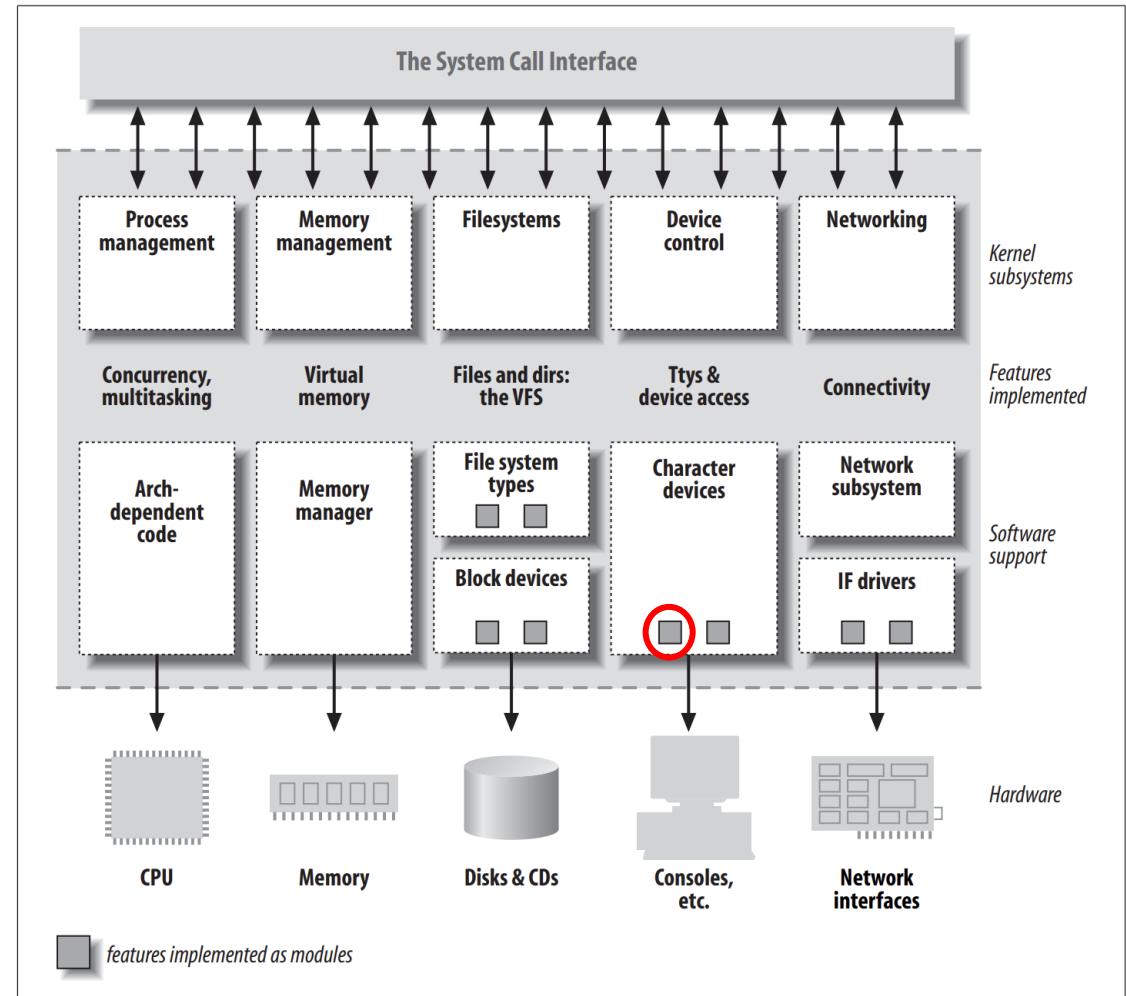


Figure 1-1. A split view of the kernel

Source: Linux Device Driver 3rd edition

hello 모듈

hello 모듈?

- 어떤 식으로 모듈을 작성하고, 컴파일하고, 커널에 적재하는지 연습하기 위한 hello module
- 필요한 파일
 - Makefile
 - hello.c

hello 모듈 작성 과정

1. 커널 모듈에 대한 소스코드 작성(hello.c)
2. Makefile을 통한 빌드 및 .ko(kernel object) 생성
3. .ko(kernel object) 파일을 커널에 등록(적재)
4. 등록된 모듈 확인
5. 등록된 모듈 제거

hello 모듈

1. 커널 모듈에 대한 소스코드 작성(hello.c)

- #include로 헤더파일 호출
- Initialization function은 int 타입, Cleanup function은 void 타입
- 커널 메시지를 출력하는 함수는 printk
 - 메시지 기록관리를 위한 로그레벨 지정
- 맨 아래 매크로 함수로 모듈에 추가될 때와 제거될 때 호출할 함수를 추가
 - 선언하지 않으면 커널에 load 및 unload 불가능
 - 다른 용도로 쓰이는 __init 및 __exit은 오류 발생

```
#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);
```

hello 모듈

2. Makefile을 통한 빌드 및 .ko 생성

- 커널 모듈을 컴파일 하는 데에는 Makefile 파일과 make 명령이 필요

```
#---- 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
```

hello 모듈

```
#----- Makefile -----#
ifeq ($(KERNELRELEASE),)
    obj-m := hello.o
else
    KERNELDIR ?= /lib/modules/$(shell uname -r)/build /* 커널 디렉토리 변수에 path(현재 커널 버전의 build 디렉토리) 할당 */
    PWD := $(shell pwd) /* 현재 위치하고 있는 디렉토리 변수에 소스코드가 있는 디렉토리 할당 */
default: /* default: 뒤에 아무 옵션을 주지 않았을 때 실행되는 명령어 */
    $(MAKE) -C $(KERNELDIR) M=$(PWD) modules /* 커널 디렉토리 변수에 서브 디렉토리(PWD 변수)를 적용하여 모듈을 컴파일 */
endif
```

hello 모듈

2. Makefile을 통한 빌드 및 .ko 생성

- 작성 후, make를 통해 빌드
⇒ .ko(커널 오브젝트) 파일 생성
- 커널 모듈로 컴파일을 하면 커널 오브젝트 파일 생성
(일반 애플리케이션으로 컴파일 시 실행 가능한 바이너리 파일 생성)

```
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
```

hello 모듈

3. .ko 파일을 커널에 등록(적재)

- 다음 명령어로 빌드된 모듈을 커널에 적재
⇒ ~\$ sudo insmod hello.ko
- printk(KERN_ALERT "Hello, world\n"); 에 대한 출력
결과는 콘솔로 뜨지 않고, dmesg 를 통해 확인
⇒ ~\$ sudo dmesg | grep 'Hello, world'

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

hello 모듈

4. 등록된 모듈 확인

- 다음과 같이 입력하여 적재된 모듈을 확인
⇒ ~\$ lsmod | grep hello

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

hello 모듈

4. 등록된 모듈 확인

- /var/log/kern.log 파일에서도 출력 결과 확인 가능
⇒ ~\$ 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$
```

- 커널에 load(insmod)하면 printk로 작성했던 'Hello, world'가 출력
- 커널에 unload(rmmod)하면 printk로 작성했던 'Goodbye, cruel world'가 출력

hello 모듈

4. 등록된 모듈 확인

- proc/modules 파일에서도 확인 가능

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

hello 모듈

5. 등록된 모듈 제거

- 다음 명령어로 적재된 모듈을 커널에서 제거
⇒ ~\$ sudo rmmod hello.ko

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

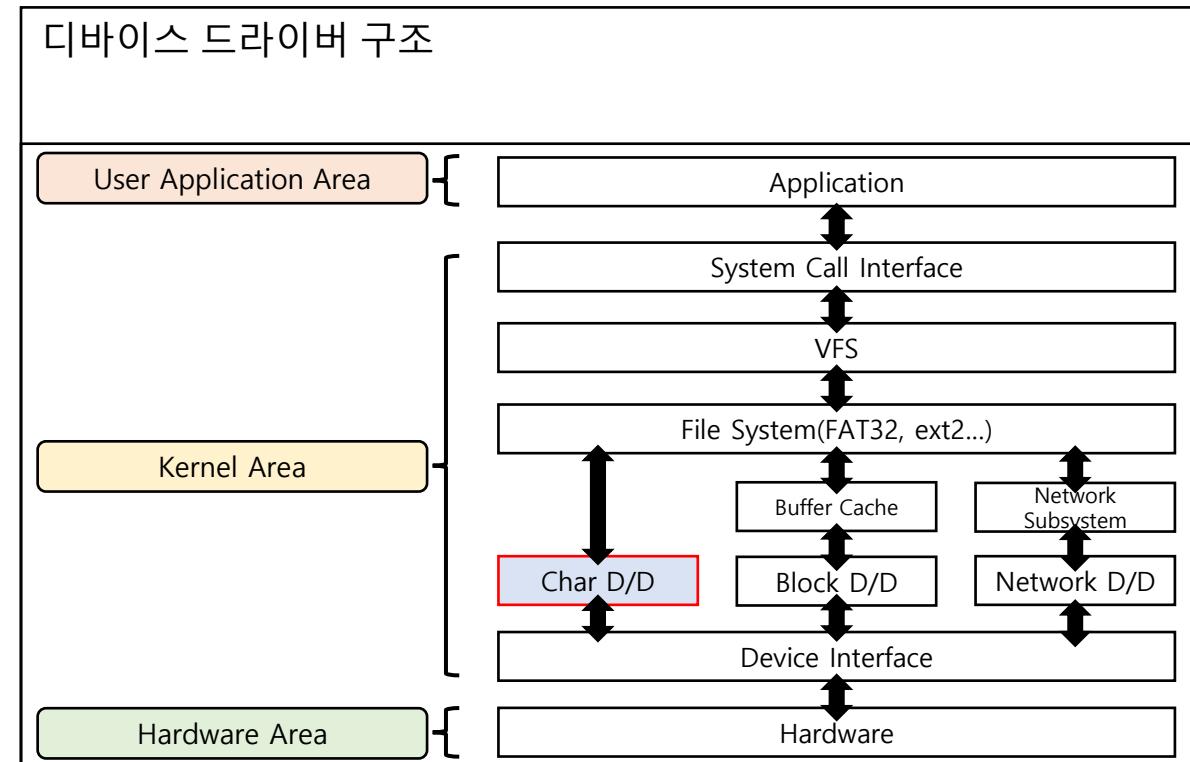
- module_exit에 등록한 함수의 printk 출력
⇒ ~\$ sudo dmesg | grep 'Hello, world'
- 등록된 모듈 리스트에서 사라진 것을 확인
⇒ ~\$ lsmod | grep hello

Character Device Driver

Character Device Driver

Character Device Driver?

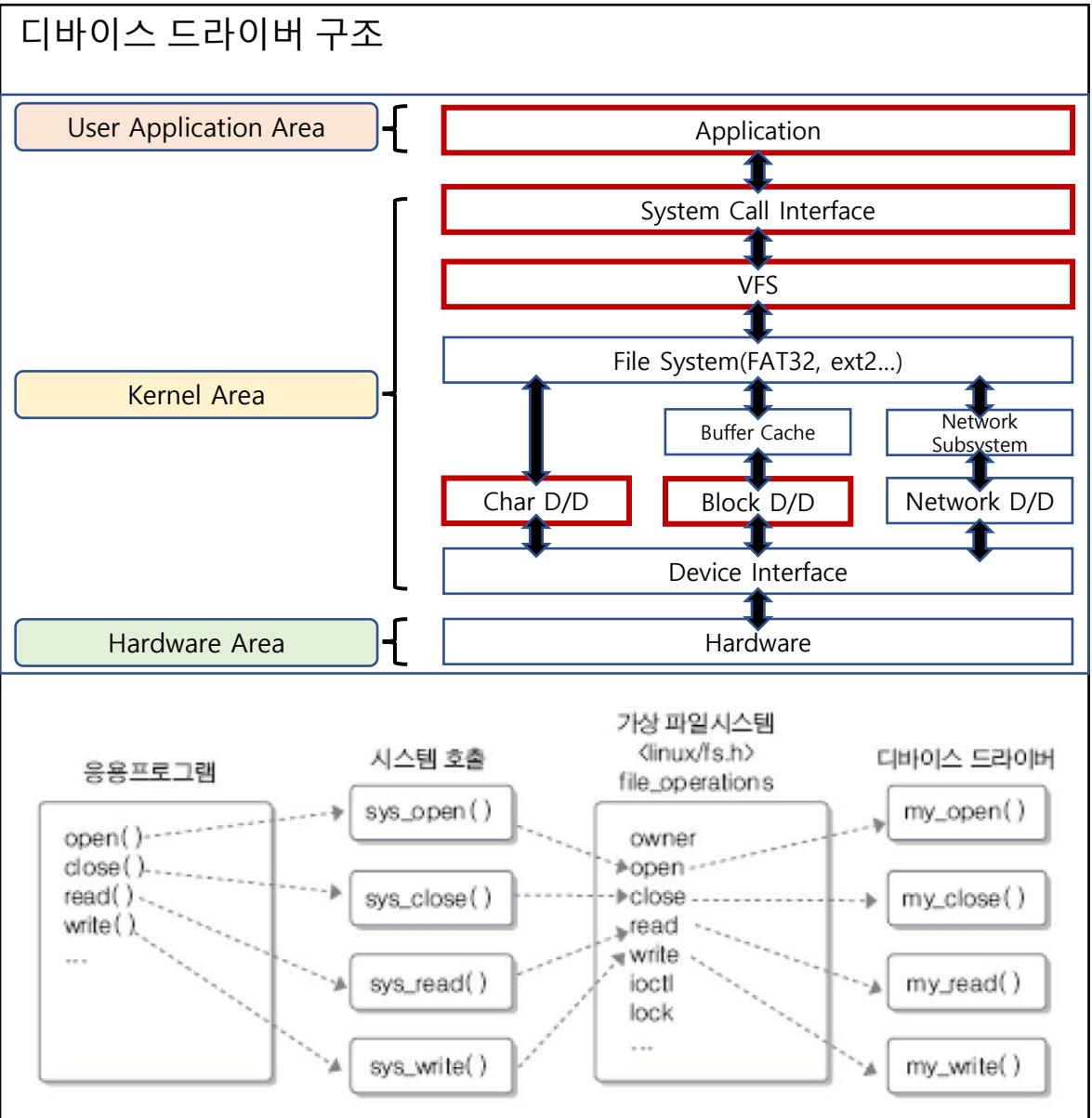
- Device를 파일처럼 접근하여 직접 read/write 수행
- Block Device Driver와는 달리 Buffer Cache가 없음
- 시간 순차적으로 처리
- 터미널, 키보드, 스캐너, 프린터, 마우스 등



Character Device Driver

Character Device Driver 작동 원리

1. 응용프로그램에서 open, read, write, close 연산 수행
2. 커널 영역에서 시스템 콜을 호출
3. file operations 구조체로 매팅된 디바이스 드라이버의 함수 호출



Character Device Driver

Character Device Driver 구조체 및 함수

- 핵심 자료구조 및 구조체는 다음과 같음

- **struct cdev** (include/linux/cdev.h)
⇒ 문자 디바이스 드라이버는 커널 내부에서 cdev로 관리됨
- **dev_t** (include/linux/kdev_t.h)
⇒ 디바이스를 구분하기 위한 번호
⇒ 상위 12비트를 주번호, 하위 20비트를 부번호로 사용
⇒ MAJOR(), MINOR() 매크로로 파싱하여 알아낼 수 있음
- **MAJOR(dev_t dev)** (include/linux/kdev_t.h)
⇒ 주번호를 추출하는 매크로
- **MINOR(dev_t dev)** (include/linux/kdev_t.h)
⇒ 부번호를 추출하는 매크로

Character Device Driver

Character Device Driver 구조체 및 함수

- 핵심 자료구조 및 구조체는 다음과 같음

- **MKDEV** (include/linux/kdev_t.h)
⇒ 반대로 주번호와 부번호를 dev_t 구조체로 변환
- **struct file_operations** (include/linux/fs.h)
⇒ VFS에서 각 file system에게 알려주는 구현할 파일의 기능들을 알려주는 인터페이스 역할
⇒ read, write, open, release, ioctl(unlocked_ioctl) 등을 구현

Character Device Driver

Character Device Driver 구조체 및 함수

- 핵심 API 및 함수는 다음과 같음

- **register_chrdev_region()**
 - int register_chrdev_region(dev_t first, unsigned int count, char* name)
 - ⇒ 디바이스 드라이버 등록 함수
 - ⇒ 원하는 디바이스의 번호를 미리 알고 있을 경우 사용
- **alloc_chrdev_region()**
 - int alloc_chrdev_region(dev_t *dev, unsigned int firstminor, unsigned int count, char *name)
 - ⇒ 디바이스 번호를 동적으로 할당하는 함수
 - ⇒ 디바이스 번호를 동적으로 할당 받아 파라미터 dev_t 구조체 포인터를 사용해 dev_t 구조체에 넣음
- **unregister_chrdev_region()**
 - void unregister_chrdev_region(dev_t first, unsigned int count)
 - ⇒ 사용 중인 디바이스 번호를 해제하는 함수

Character Device Driver

Character Device Driver 구조체 및 함수

- 핵심 API 및 함수는 다음과 같음

- **cdev_init()**
 - void cdev_init(struct *cdev, struct file_operations *fops)
⇒ cdev 구조체 초기화 함수
- **cdev_add()**
 - void cdev_add(struct cdev *cdev, dev_t num, unsigned int count)
⇒ 준비된 cdev 구조체를 커널에 등록(char device 등록)
- **cdev_del()**
 - int cdev_del(struct cdev *cdev)
⇒ 등록된 cdev 구조체 해제(등록된 char device 제거)
⇒ device 번호는 unregister_chrdev_region()으로 해제

Character Device Driver

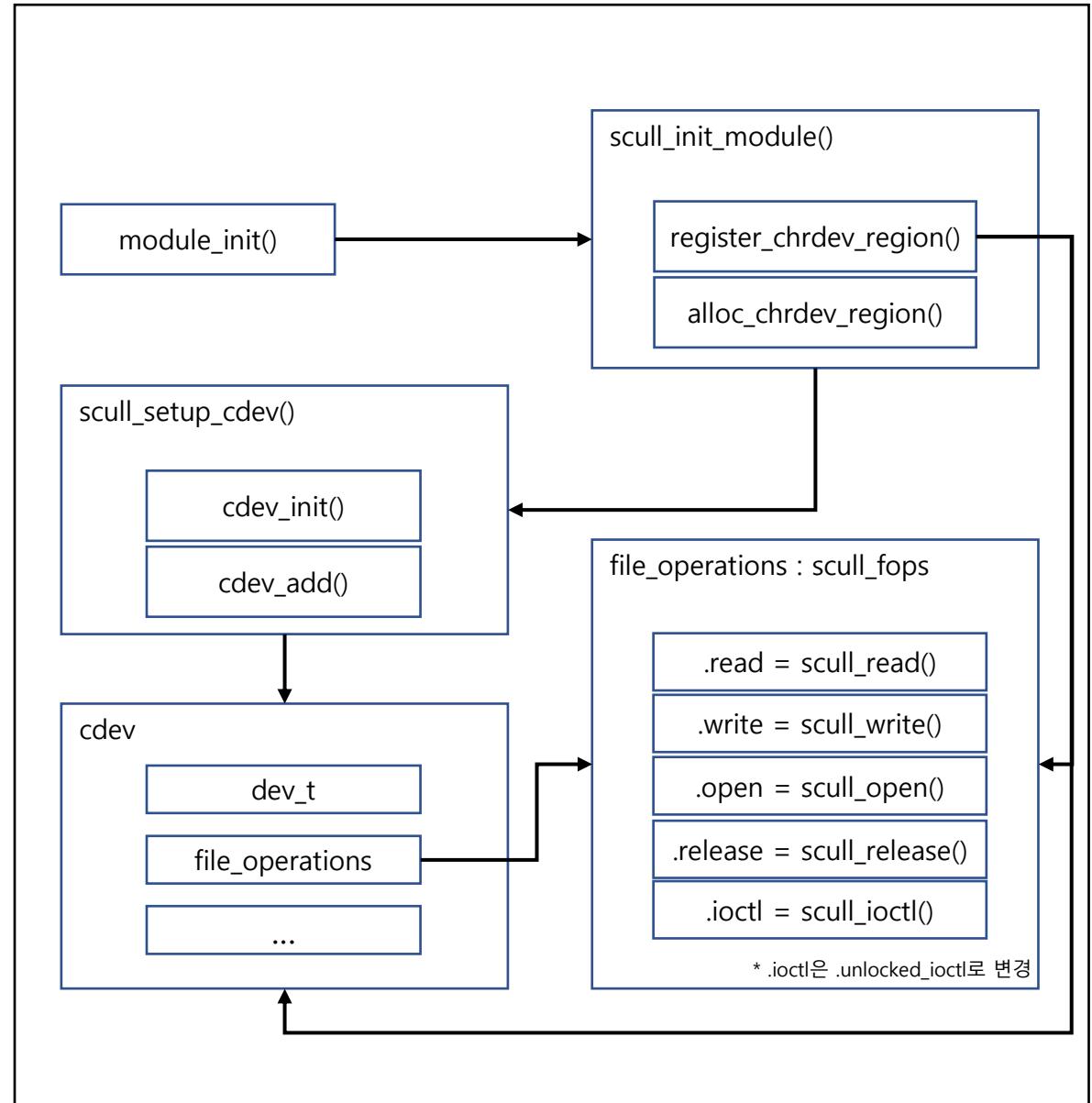
Character Device Driver 함수 호출 관계

insmod로 커널 load 시,

1. module_init() 매크로 호출

2. scull_init_module() 호출

⇒ register_chrdev_region() 호출(주번호와 함께 디바이스 구조체 cdev 등록 및 file_operations : fops 매핑 생성)



메모리 영역 내에서 수행되는 문자 디바이스 드라이버인 scull(Simple Character Utility for Loading Localities)

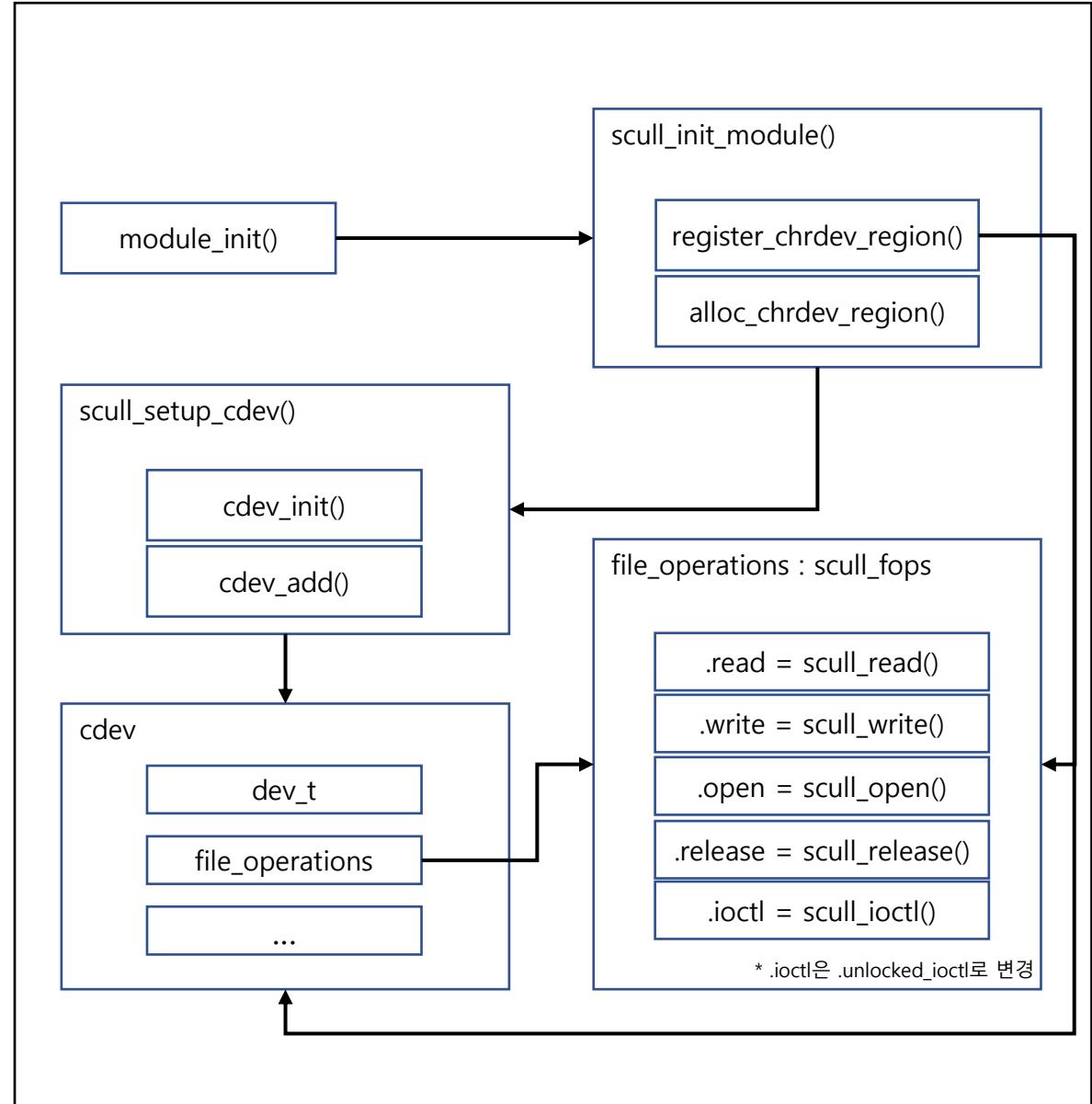
Character Device Driver

Character Device Driver 함수 호출 관계

3. scull_setup_cdev() 호출

⇒ cdev_init() 호출(cdev 구조체 초기화)

⇒ cdev_add() 호출(준비된 cdev 구조체 커널에 등록)



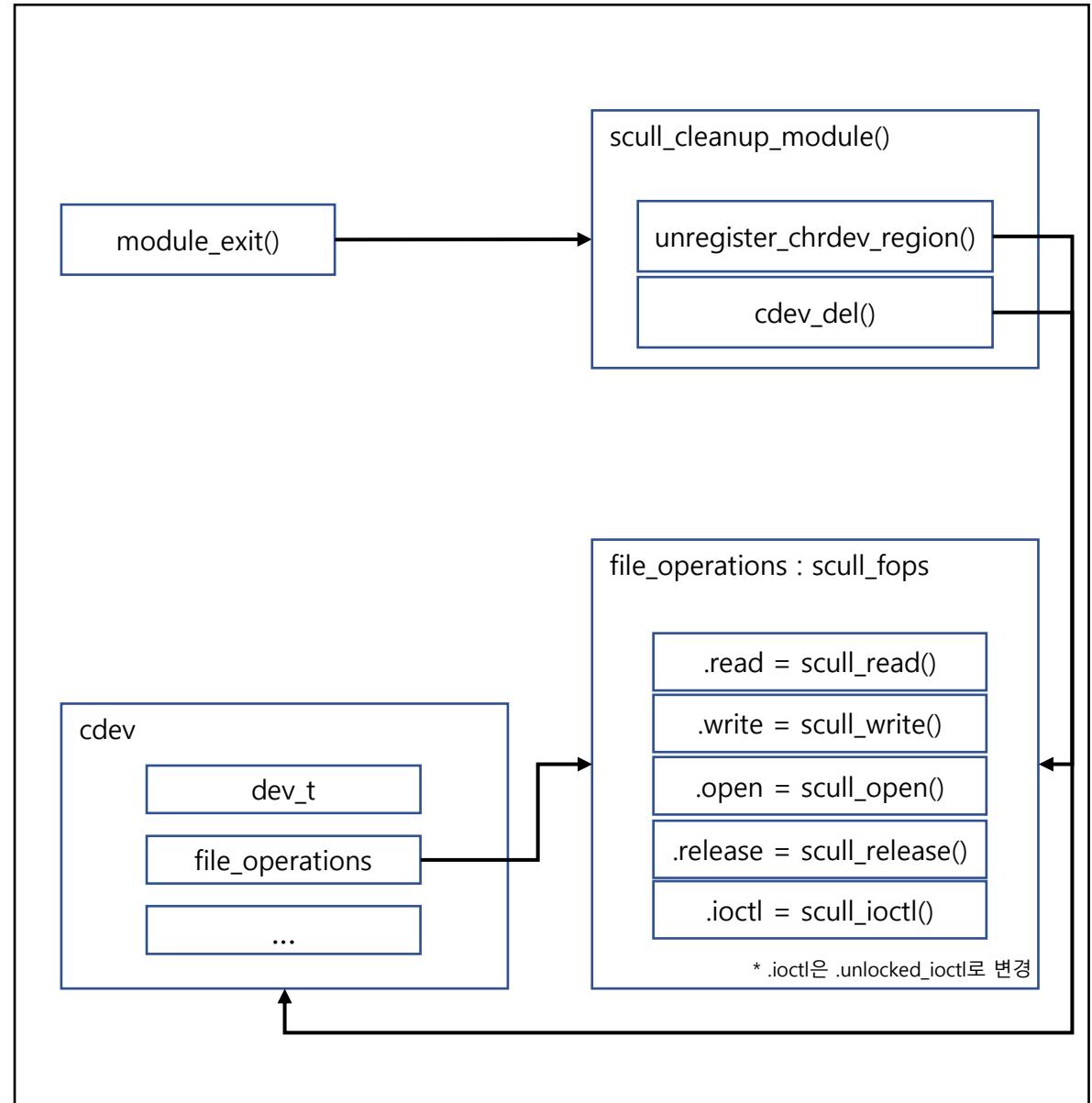
메모리 영역 내에서 수행되는 문자 디바이스 드라이버인 scull(Simple Character Utility for Loading Localities)

Character Device Driver

Character Device Driver 함수 호출 관계

rmmod로 커널 unload 시,

1. module_exit() 매크로 호출
2. scull_cleanup_module() 호출
 - ⇒ unregister_chrdev_region() 호출(device 번호 해제)
 - ⇒ cdev_del() 호출(cdev에 등록된 char device 제거)
 - ⇒ file_operations : fops 매팅 커널에서 제거



Character Device Driver

Character Device Driver 등록 및 확인

- make 명령어를 통해 작성한 소스코드 컴파일

⇒ ~\$ sudo make

- ls 명령어로 확인

⇒ ~\$ ls

```
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
```

Character Device Driver

Character Device Driver 등록 및 확인

- 관리자 권한을 통해 scull_load 스크립트를 실행
⇒ ~\$ sudo sh scull_load
- scull_load 스크립트의 경우, 이전 등록된 /dev 디렉토리에 있던 scull device 파일을 제거한 뒤, 새롭게 device driver를 커널에 적재하고 등록하는 파일
- insmod는 scull_load 스크립트 파일 안에 위치하므로 스크립트 실행으로 커널에 등록(적재)

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

Character Device Driver

Character Device Driver 등록 및 확인

- lsmod 명령어 및 ls -al 명령어로 scull 모듈이 성공적으로 적재되었는지 확인
⇒ ~\$ lsmod | grep scull
⇒ ~\$ ls -al /dev | grep scull
- 첫번째 문자에 'c'가 뜨고 scull0~3까지 나타나는 것으로 보아 성공적으로 등록

```
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$
```

Character Device Driver

Character Device Driver 제거 및 확인

- 관리자 권한을 통해 scull_unload 스크립트를 실행
⇒ ~\$ sudo sh scull_unload
- 반대로 scull_unload 스크립트에는 rmmod가 포함
- lsmod 명령어 및 ls -al 명령어로 scull 모듈이 성공적으로 제거되었는지 확인
⇒ ~\$ lsmod | grep scull
⇒ ~\$ ls -al /dev | grep 'scull'

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

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

Character Device Driver

Character Device Driver 테스트

- ioctl 함수
 - ⇒ 하드웨어 제어와 상태 정보를 얻기 위해 사용
 - ⇒ I/O Control에 관련된 작업을 수행
 - ⇒ cmd 인수 값에 따라 동작 선택하는 switch문으로 구성

<User Space에서 ioctl 함수 system call>

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

- 첫번째 인자(fd): file descriptor
- 두번째 인자(cmd): 다양한 Request Operation Code 중에서 적절한 값을 설정
- 마지막 인자: ioctl operation의 처리 대상인 데이터 입력

Character Device Driver

Character Device Driver 테스트

- ioctl 함수
 - ⇒ 하드웨어 제어와 상태 정보를 얻기 위해 사용
 - ⇒ I/O Control에 관련된 작업을 수행
 - ⇒ cmd 인수 값에 따라 동작 선택하는 switch문으로 구성

<Kernel Space에서 ioctl 함수 system call>

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

- 첫번째 인자(*filp): 응용프로그램의 fd(file descriptor)
- 두번째 인자(cmd): 명령을 나타내는 응용프로그램의 인수 전달
- 마지막 인자(arg): 명령 실행의 결과 데이터가 전달되는 unsigned long 타입의 정수 또는 포인터

Character Device Driver

Character Device Driver 테스트

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

cmd 명령의 작성 매크로 함수

- `_IO(type, nr)`: 부가적인 데이터가 없는 명령을 작성
- `_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 써넣기 위한 명령을 작성
(user -> kernel 메모리 영역으로 복사가 일어나는 `copy_from_user` 동작이 일어날 때 사용)
- `_IORW(type, nr, size)`: 데이터를 읽고 쓰기 위한 명령을 작성
(kernel <-> user 양쪽 메모리 영역으로 모두 사용될 때 사용)

* type: 매직넘버(8bit) nr: 구분번호(명령어 구분) size: 데이터 크기(user, kernel 영역 사이 오가는 정수형 데이터 크기)

Character Device Driver

Character Device Driver 테스트

- ioctl_info와 SCULL_IOC_SQUANTUM과 같은 오퍼레이션 정보들은 User Space의 응용 프로그램에서 또한 적용해야 하므로 따로 정리하여 ioctl.h 헤더파일 작성

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_IORESET _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 "sHift": switch T and Q atomically
 */
#define SCULL_IOC_SQUANTUM _IOW(SCULL_IOC_MAGIC, 1, int)
#define SCULL_IOCQSET     _IOW(SCULL_IOC_MAGIC, 2, int)
#define SCULL_IOCQGET     _IO(SCULL_IOC_MAGIC, 3)
#define SCULL_IOCQSET     _IO(SCULL_IOC_MAGIC, 4)
#define SCULL_IOCQGET     _IOR(SCULL_IOC_MAGIC, 5, int)
#define SCULL_IOCQSET     _IOR(SCULL_IOC_MAGIC, 6, int)
#define SCULL_IOCQGET     _IO(SCULL_IOC_MAGIC, 7)
#define SCULL_IOCQSET     _IO(SCULL_IOC_MAGIC, 8)
#define SCULL_IOCQGET     _IOWR(SCULL_IOC_MAGIC, 9, int)
#define SCULL_IOCQSET     _IOWR(SCULL_IOC_MAGIC, 10, int)
#define SCULL_IOCQGET     _IO(SCULL_IOC_MAGIC, 11)
#define SCULL_IOCQSET     _IO(SCULL_IOC_MAGIC, 12)

#define SCULL_IOC_MAXNR 14
```

Character Device Driver

Character Device Driver 테스트

- scull_ioctl() 함수가 제대로 수행되는지 확인하기 위한 test 파일
- scull0 디바이스 file descriptor를 바탕으로 ioctl 함수 호출
⇒ fd = open("/dev/scull0", O_RDWR)

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_IOCQUANTUM, &set_data) < 0) {
        printf("Error in SCULL_IOCQUANTUM statement.\n");
        return -EPERM;
    }

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

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

return 0;
```

Character Device Driver

Character Device Driver 테스트

- SCULL_IOCSQUANTUM (Set) 옵션을 통해 User Space에서 Kernel Space로 set_data 변수 데이터 2023을 전송하여 scull_quantum 변수에 저장
⇒ ioctl(fd, SCULL_IOCSQUANTUM, &set_data)
- SCULL_IOCQQUANTUM (Get) 옵션을 통해 Kernel Space에서 User Space로 저장된 scull_quantum 변수 데이터를 전송하여 get_data 변수에 저장 및 출력
⇒ ioctl(fd, SCULL_IOCQQUANTUM, &get_data)

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_IOCSQUANTUM, &set_data) < 0) {
        printf("Error in SCULL_IOCSQUANTUM 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;
    }
}
```

Character Device Driver

Character Device Driver 테스트

- test.c 파일을 컴파일 후 a.out 파일 생성 확인

⇒ ~\$ gcc test.c

⇒ ~\$ ls

- 실행 파일 실행

⇒ ~\$ sudo ./a.out

- Kernel Space의 scull_quantum 값 '2023' 정상 출력

⇒ get_data : 2023

```
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$ █
```

Character Device Driver

트러블 슈팅

- 문제점 1

: 단순 오타 및 함수 호출 순서 문제

- 문제점 2

: 스크립트 파일에 대한 이해 부족

- 문제점 3

: access_ok() 함수 선언 부재

Character Device Driver

트러블 슈팅

- 문제점 4
: .ioctl 문제
- 문제점 5
: 헤더파일 참조 문제

Character Device Driver

트러블 슈팅(단순 오타 및 함수 호출 문제)

- make 명령어로 컴파일 진행했을 때 나타난 오류

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:/EdgeClab/chardd$ ls
Makefile scull.c scull.h scull_load scull_unload
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: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: 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:
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 are 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-UX3402ZA:/EdgeClab/chardd$
```

Character Device Driver

트러블 슈팅(단순 오타 및 함수 호출 문제)

- %연산자를 &연산자로 수정
- C언어는 객체지향이 아닌 절차형 언어이므로 변수 및 함수 선언과 호출에 유의해야 함
⇒ scull_cleanup_module() 함수를 init 보다 뒤에 작성하는 등 컴파일러가 인식하지 못해 생긴 오류

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-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 (*)(st
ruct 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 *)' {ak
a '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);
        |         ~~~~~
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:11: default] Error 2
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-UX3402ZA:~/EdgeClab/chardd$
```

```
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 *)' {ak
a '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$
```

Character Device Driver

트러블 슈팅(단순 오타 및 함수 호출 문제)

- 함수 호출과 변수에 맞게 코드 구조 다음과 같이 작성
⇒ trim(), open(), release(), follow(), read(), write(), llseek(), file_operations fops, setup(), cleanup(), init(), module_init() 매크로, module_exit() 매크로 순서
- scull.h 헤더파일의 scull_write 함수 프로토타입 정의
파라미터 타입 오류
⇒ char를 const char로 변경

```
/* 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

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-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-UX3402ZA:~/EdgeClab/chardd$
```

```
seongbinyoon@seongbinyoon-Zenbook-UX3402ZA-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-UX3402ZA:~/EdgeClab/chardd$
```

Character Device Driver

트러블 슈팅(스크립트에 대한 이해 부족)

- scull_load 및 scull_unload 스크립트 파일의 역할을 더
 둘게 이해함

⇒ 스크립트 파일에 각각 insmod와 rmmod 명령어가 위치
 해 해당 스크립트를 실행하는 것으로 load/unload가 수
 행됨

```
#!/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]
```

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

Character Device Driver

트러블 슈팅(access_ok() 함수 선언 부재)

- 커널 2.4.20 이후에는 사용자 공간을 가리키는 포인터가 정상인지 확인하는 작업을 verify_area() 대신 access_ok() 함수를 통해 실행
- access_ok에 대한 정의가 되어 있지 않아 발생한 오류
⇒ access_ok_version.h 헤더파일을 생성 후 include
⇒ scull.c 파일에서 access_ok -> access_ok_wrapper로 수정

```
/*
 * 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
~

/* 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;
```

Character Device Driver

트러블 슈팅(.ioctl 문제)

- 커널 2.6.36 이후부터 기존 device driver를 제어하기 위한 방법 중 하나였던 ioctl이 unlocked_ioctl로 대체됨
- ioctl이 BKL(Big Kernel Lock)로 보호되었다면, unlocked_ioctl은 BKL이 없음

*BKL은 커널 전체에 걸리는 락으로, 동시에 여러 개의 프로세스가 커널모드에 진입하지 못하도록 막는 락(비효율적)

⇒ Preemptible하게 되어 ioctl 동시성에 대해 driver 스스로 보호해야 함(device driver 스스로 락을 관리)

```
// 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
};
```

Character Device Driver

트러블 슈팅(헤더파일 참조 문제)

- 테스트 코드를 통해 ioctl이 제대로 동작하는지 확인하기 위한 테스트 코드 컴파일 과정에서 문제 발생
- error: field has incomplete type
⇒ 전방 선언(forward declaration) 또는 헤더파일 참조가 잘못되어 나타난 오류

```
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);
      |                     ^
```

Character Device Driver

트러블 슈팅(헤더파일 참조 문제)

- 해결 방법은 다음과 같음

1. ioctl_info 구조체와 ioctl 정의를 scull.h 헤더파일이 아닌 따로 ioctl.h 헤더파일에 작성
2. scull.c 파일에 scull.h 및 ioctl.h 헤더파일 include
3. 테스트 파일 test.c 파일에 ioctl.h 헤더파일을 include

```
/* 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 */

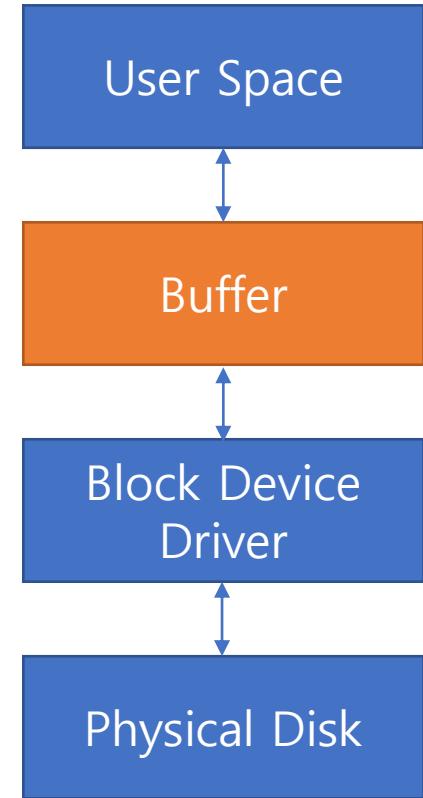
#define SET_DATA    _IOW(SCULL_IOC_MAGIC, 2, struct ioctl_info)
#define GET_DATA    _IOR(SCULL_IOC_MAGIC, 3, struct ioctl_info)
#define SCULL_IORESET _IO(SCULL_IOC_MAGIC, 0)

/*
 * S means "Get" thru read & write
 * T means "Tell" do #include <linux/module.h>
 * G means "Get": re #include <linux/moduleparam.h>
 * Q means "Query": #include <linux/init.h>
 * X means "Exchange": #include <linux/init.h>
 * H means "Huffman": #include <linux/kernel.h>
 */
#define SCULL_IOSQH "#include <linux/slab.h>"
#define SCULL_IOSQS "#include <linux/fs.h>"
#define SCULL_IOTCOS "#include <linux/errno.h>"
#define SCULL_IOTCQH "#include <linux/types.h>"
#define SCULL_IOTCQH "#include <linux/fcntl.h>"
#define SCULL_IOTCQH "#include <linux/cdev.h>"
#define SCULL_IOTCQH "#include <linux/proc_fs.h>"
#define SCULL_IOTCQH "#include <linux/seq_file.h>"
#define SCULL_IOC "#include <asm/uaccess.h>
#define SCULL_IOC "#include <scull.h>
#define SCULL_IOC "#include <ioctl.h>
#define SCULL_IOC "#include <access_ok_version.h>

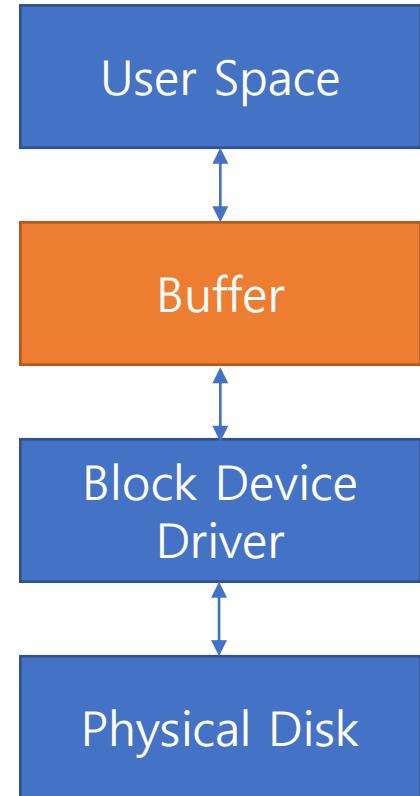
int main()
{
    int fd;
    int set_data = 2023; /* If success, it will be changed to 2023. */
    if ((fd = open("/dev/scull0", O_RDWR)) < 0) {
        printf("Cannot open /dev/scull0. Try again later.\n");
        return -ENXIO;
    }
    /* User Space -> Kernel Space */
    if (ioctl(fd, SCULL_IOCQuantum, &set_data) < 0) {
        printf("Error in SCULL_IOCQuantum statement.\n");
        return -EPERM;
    }
    /* Kernel Space -> User Space */
    if (ioctl(fd, SCULL_IOCQuantum, &get_data) < 0) {
        printf("Error in SCULL_IOCQuantum statement.\n");
        return -EPERM;
    }
    printf("get_data : %d\n", get_data);
    if (close(fd) != 0) {
        printf("Cannot close.\n");
        return -ENXIO;
    }
    return 0;
}
```

Block Device Driver

Block Device Driver



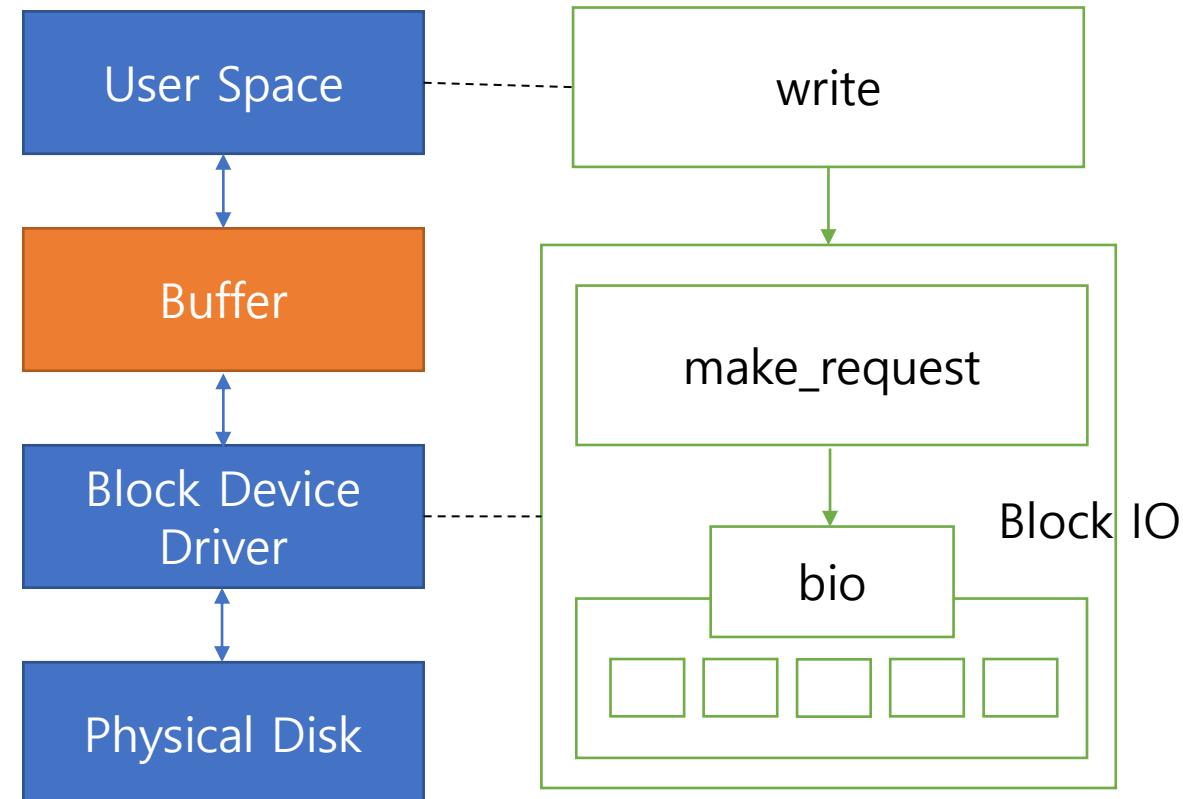
Block Device Driver



LDD3 supports Linux 2.6

But we use Linux 6.1.2

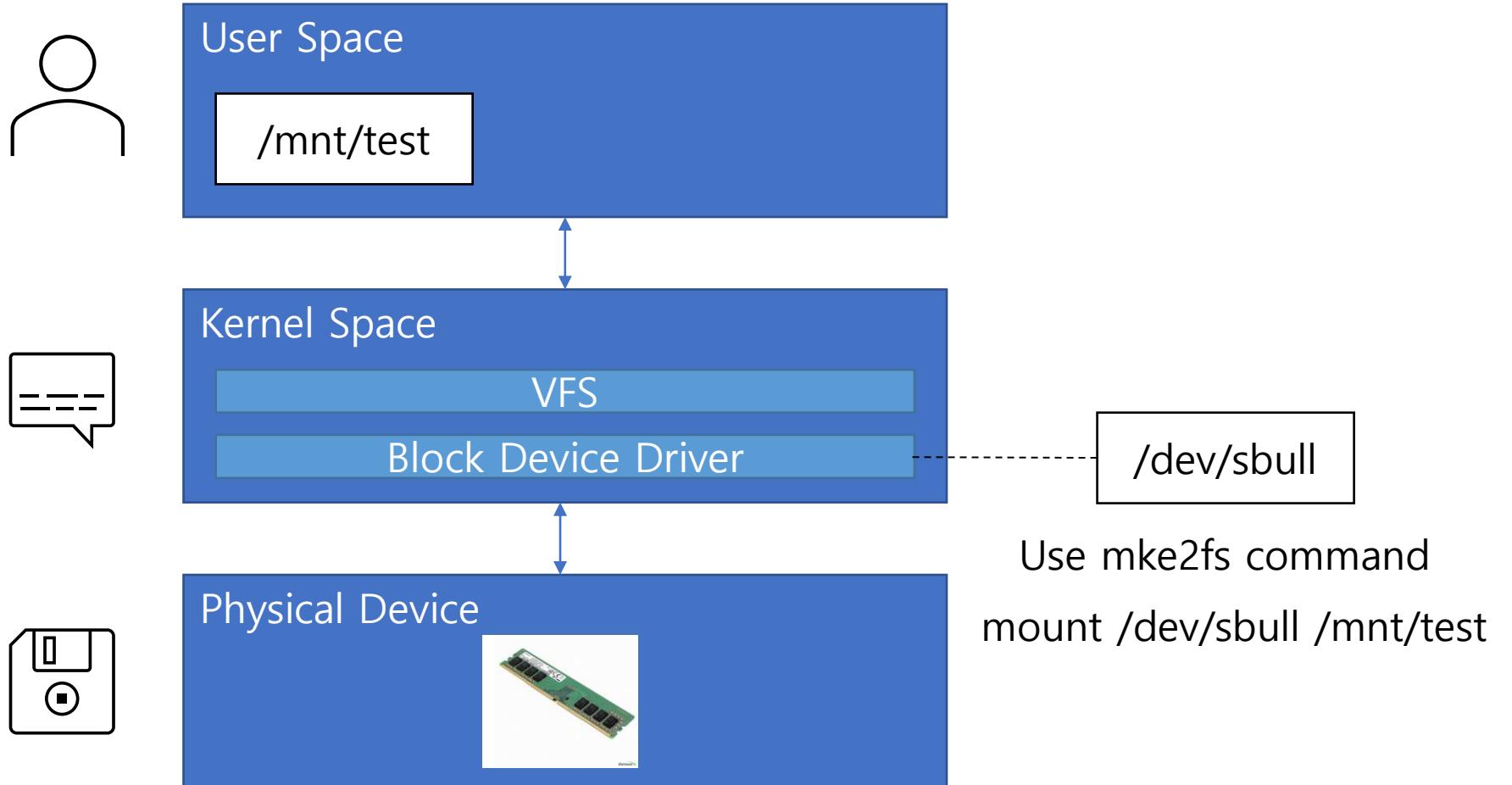
Block Device Driver



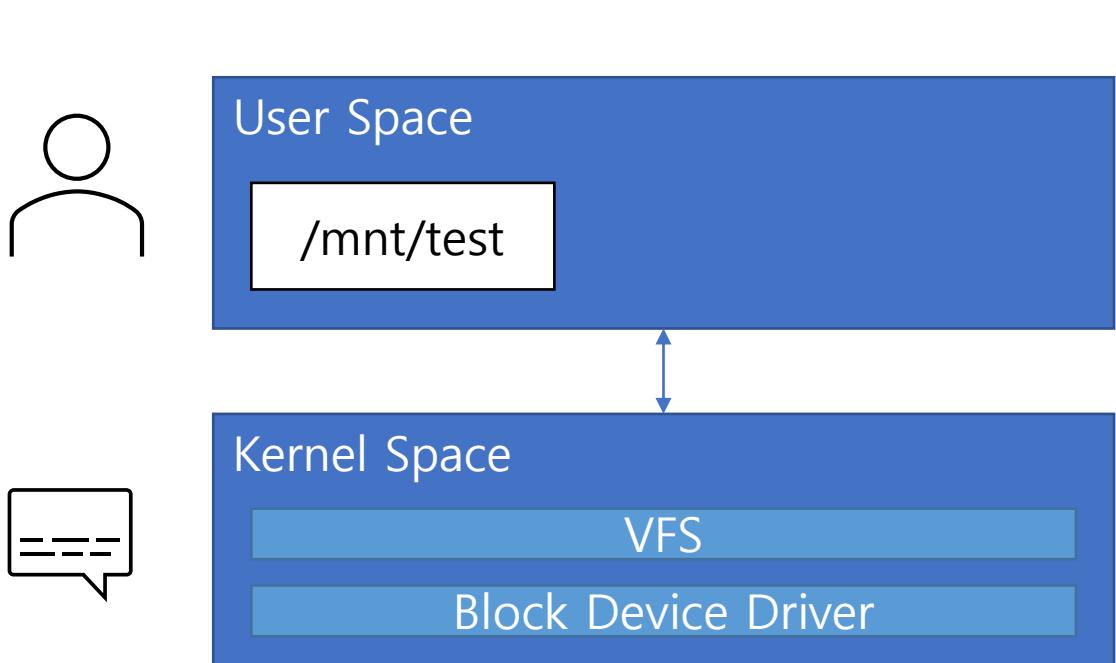
Block Device Driver for **VirtIO** File-System

Then, How operates cloud system?

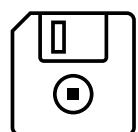
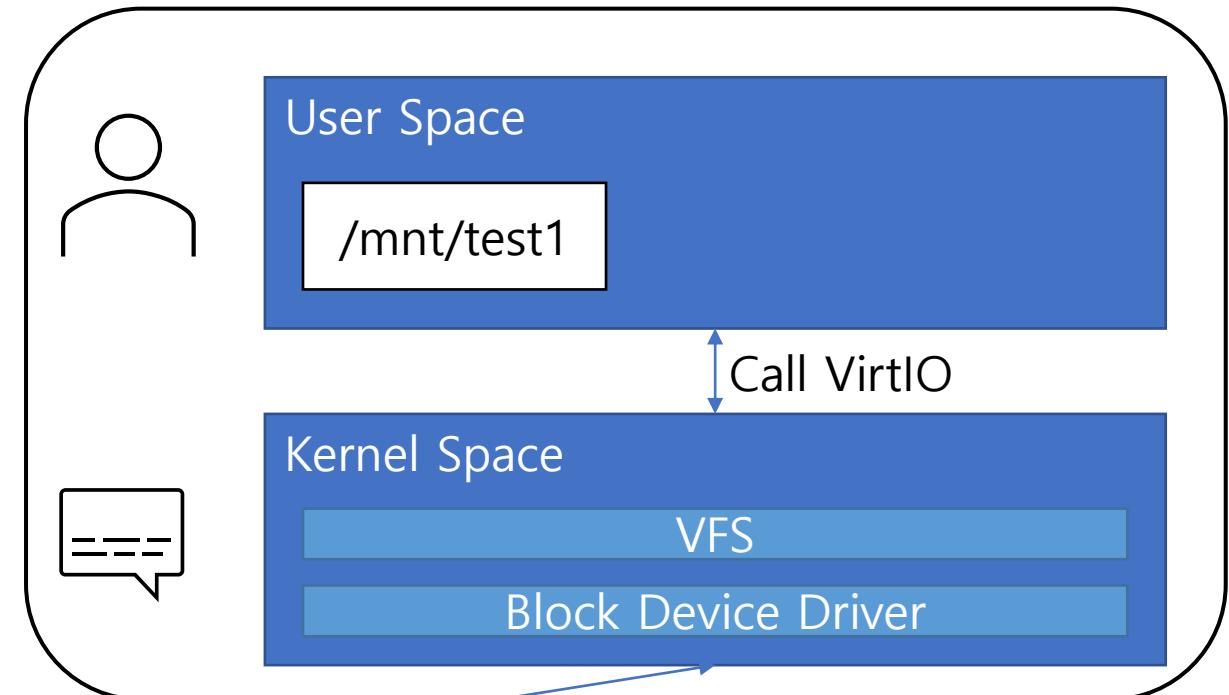
Block Device Driver for VirtIO File-System



Block Device Driver for VirtIO File-System



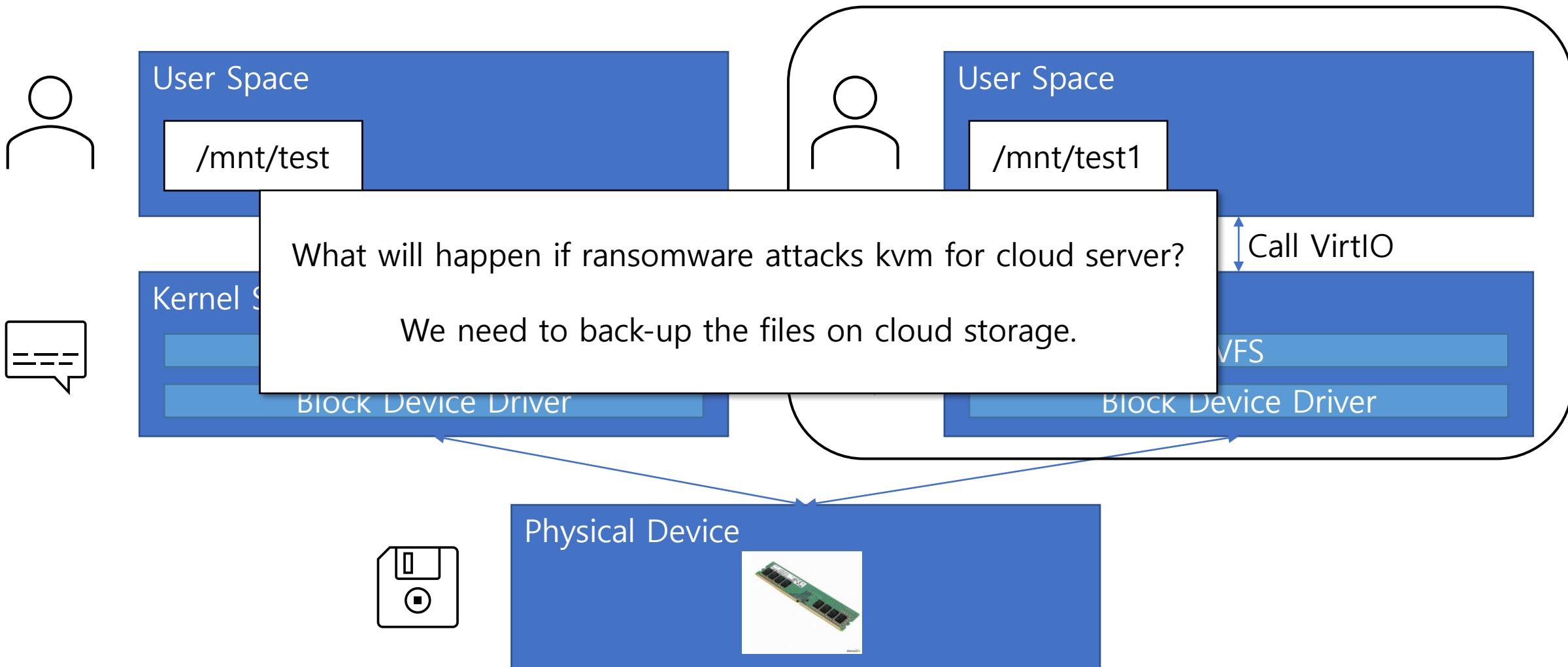
KVM for cloud server user



Physical Device

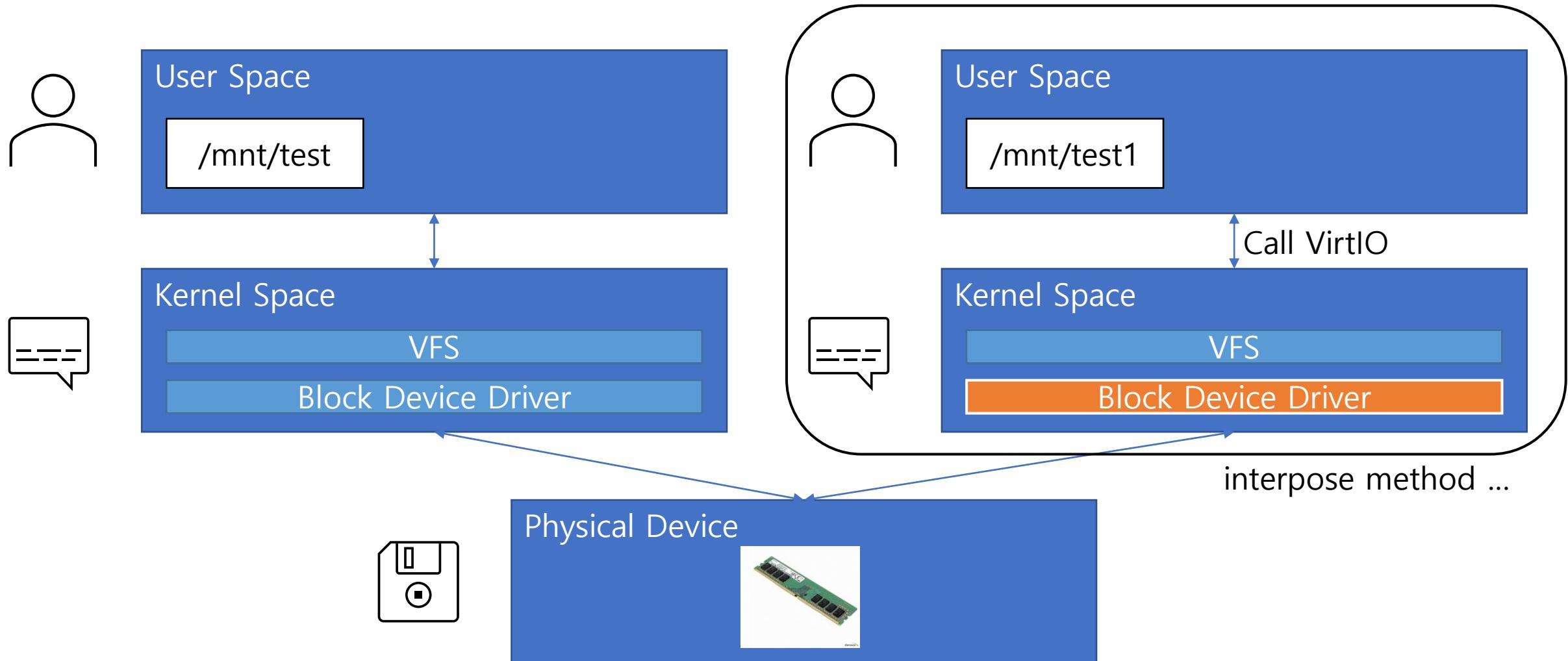
Block Device Driver for VirtIO File-System

KVM for cloud server user



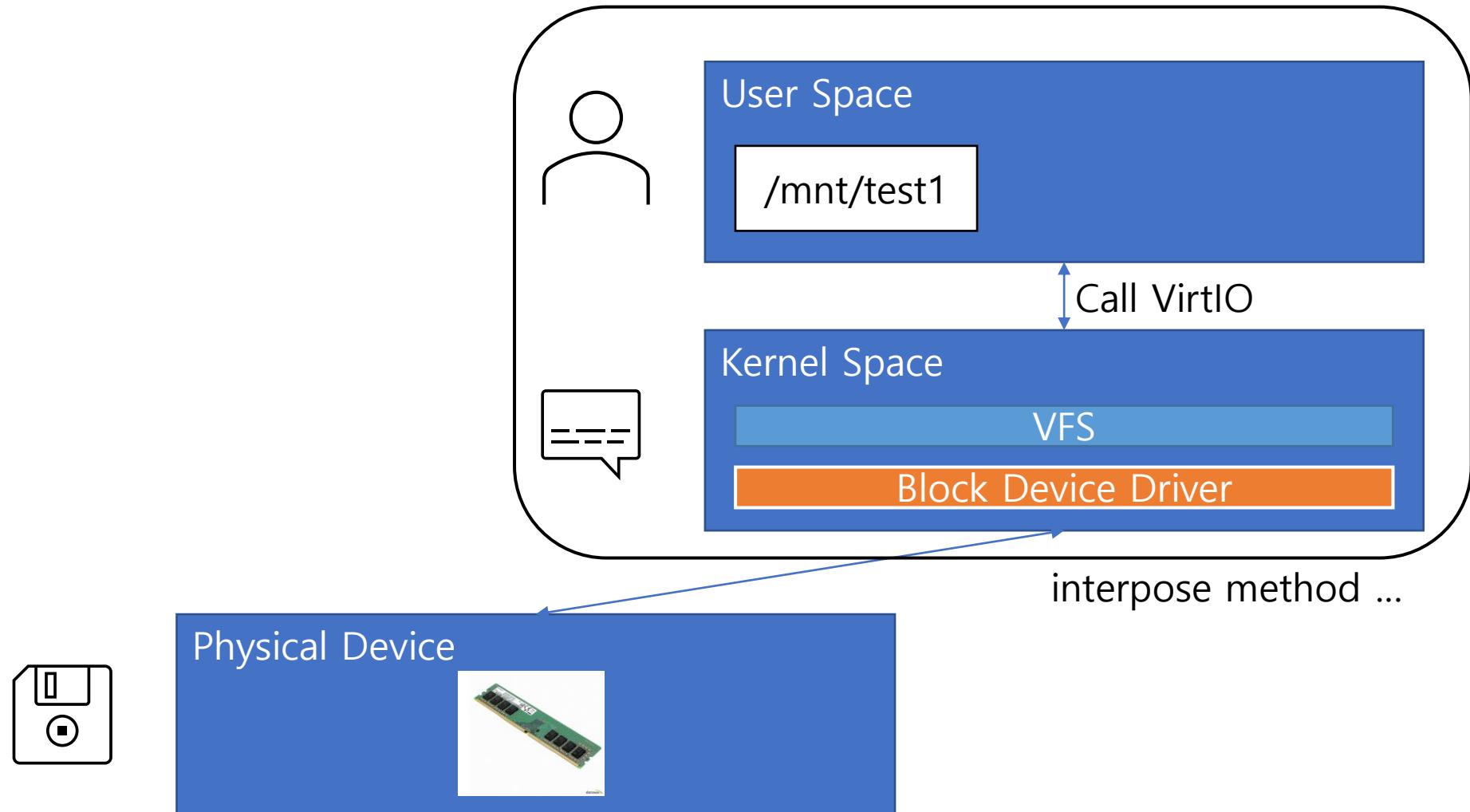
Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



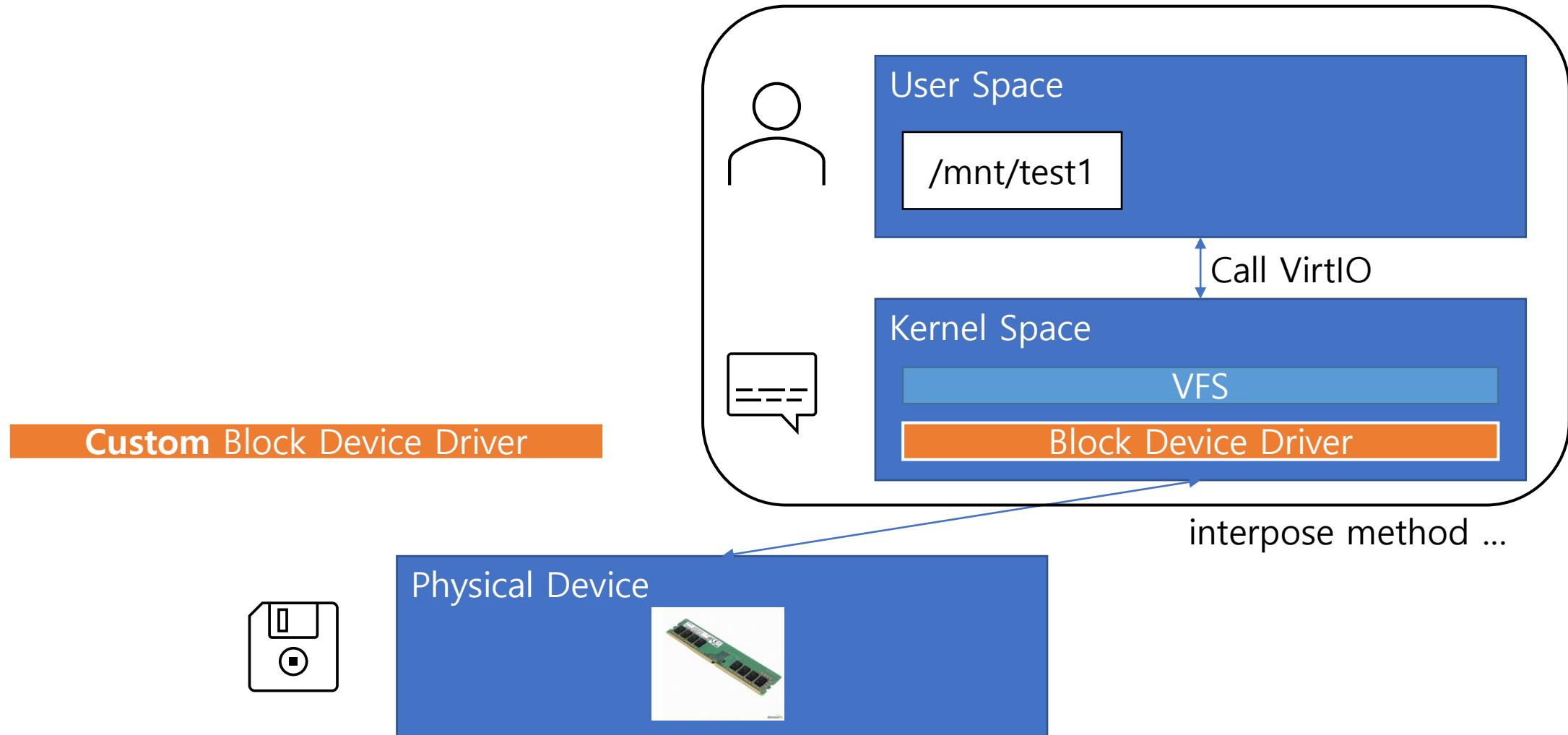
Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



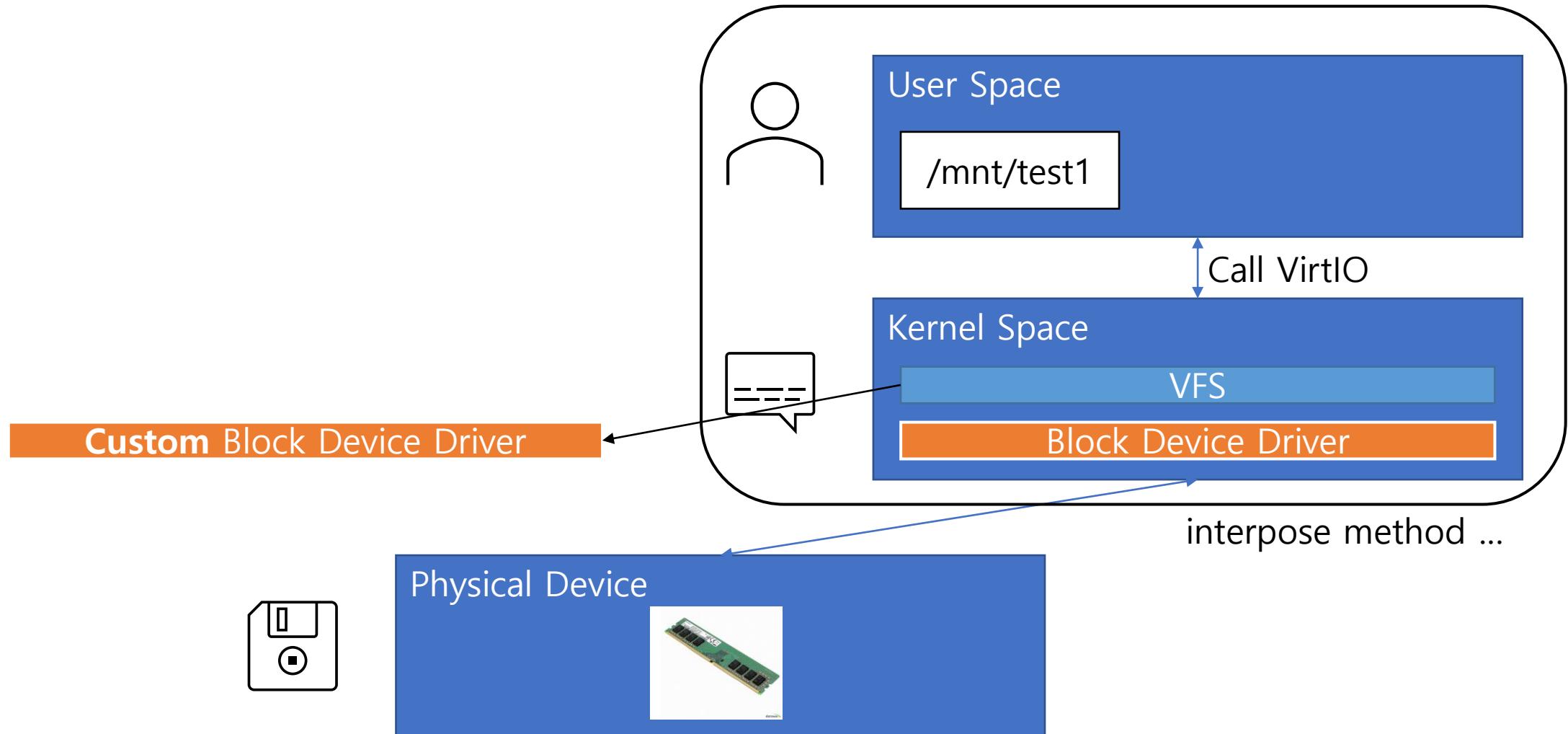
Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



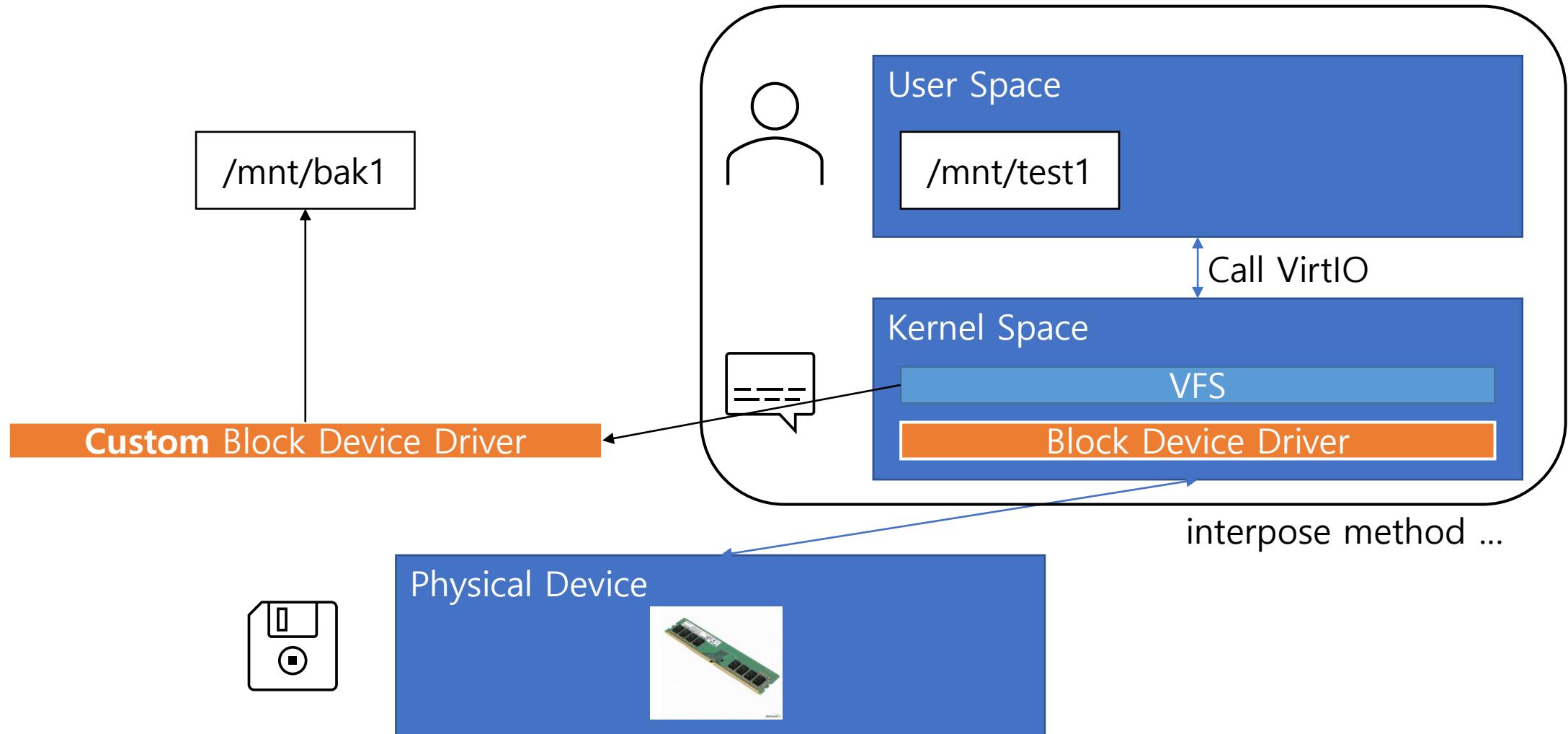
Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



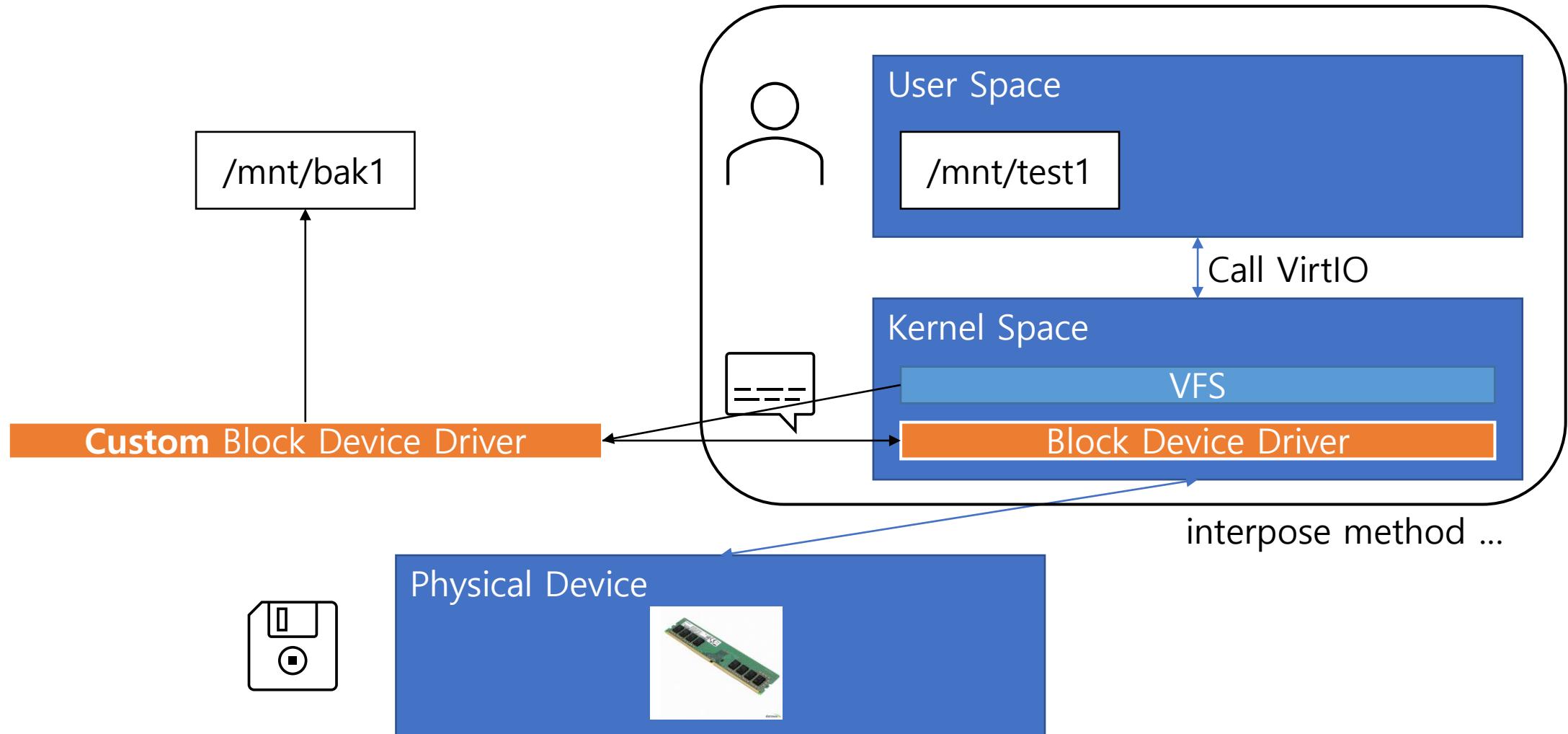
Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



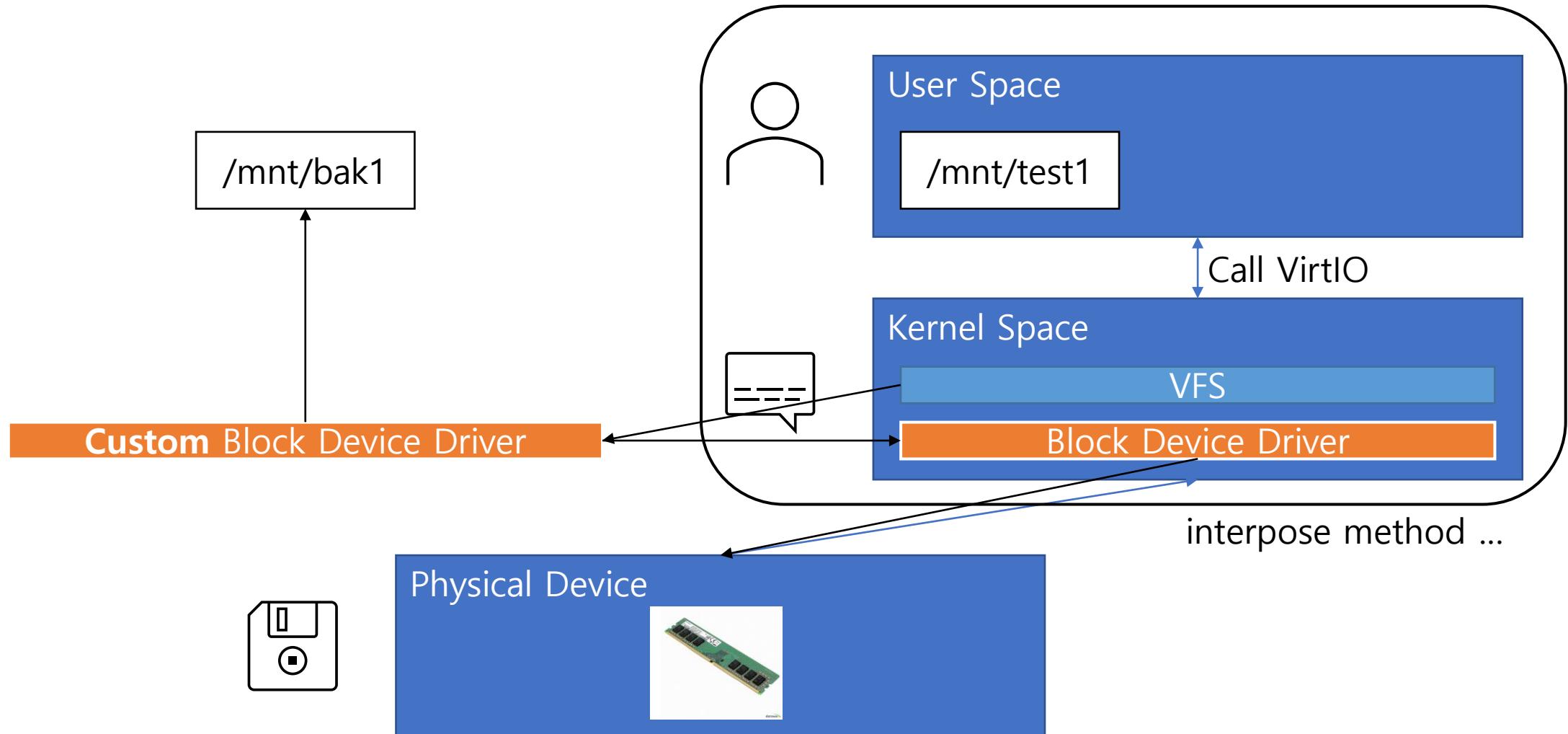
Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user

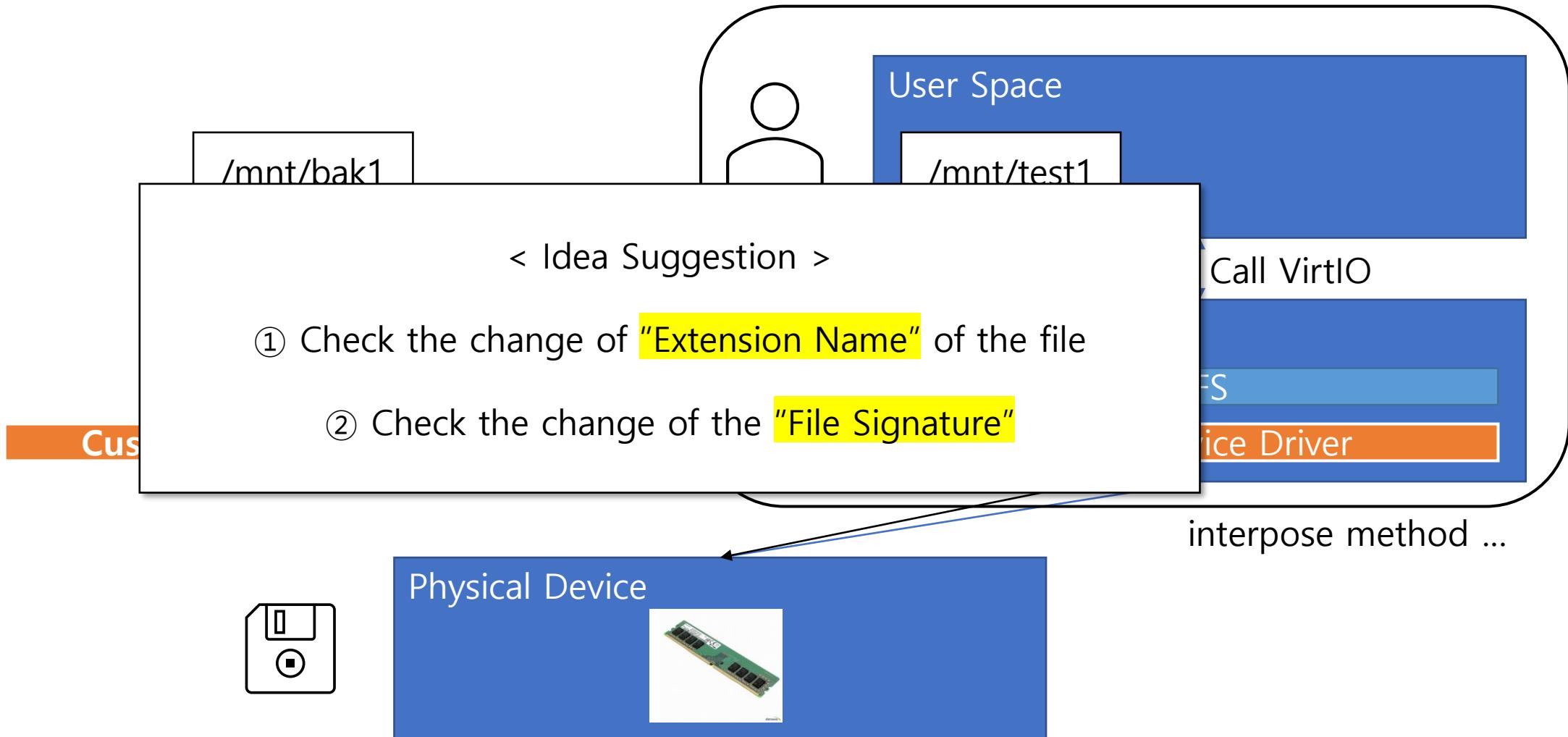


Block Device Driver

Then, how to find attacks of ransomware?

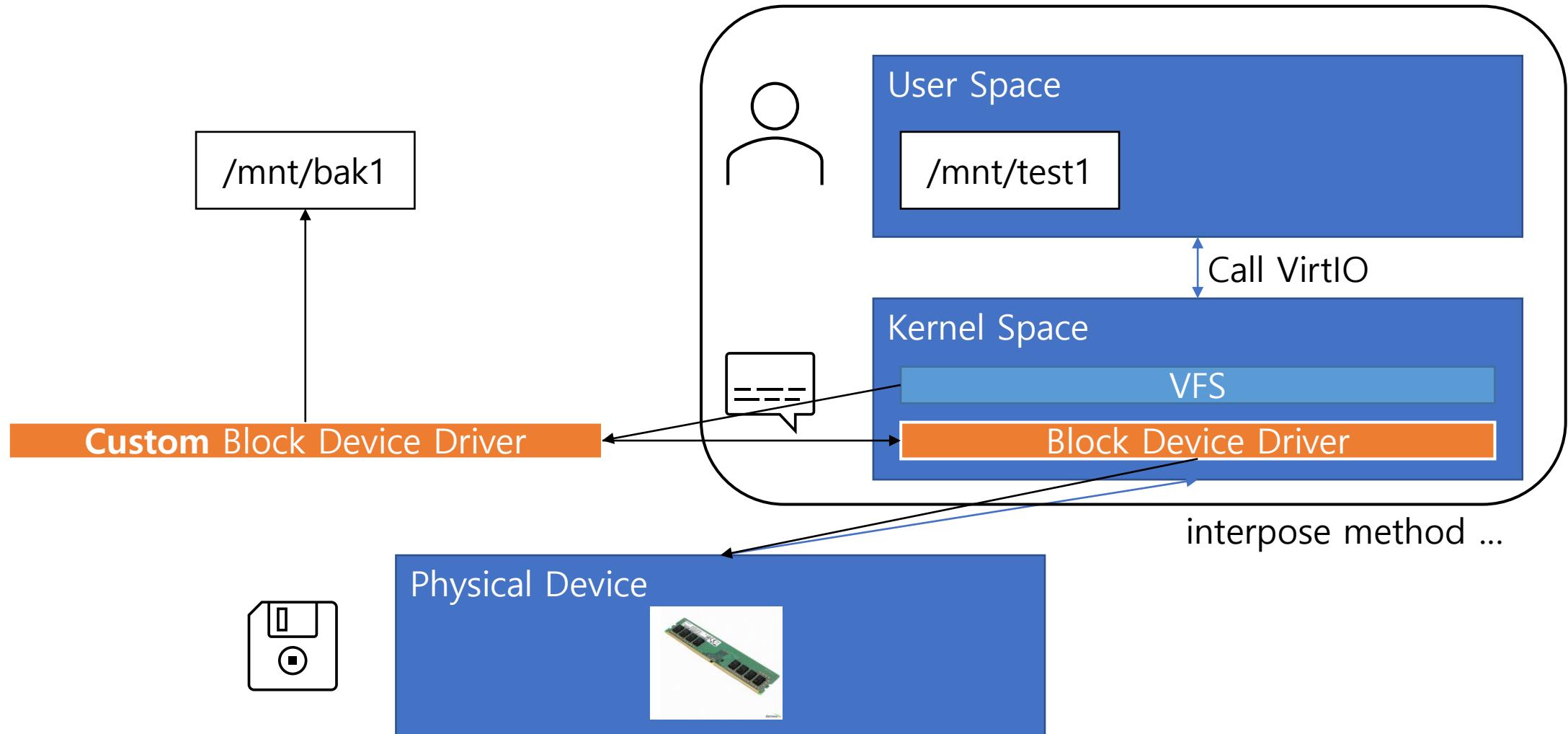
Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



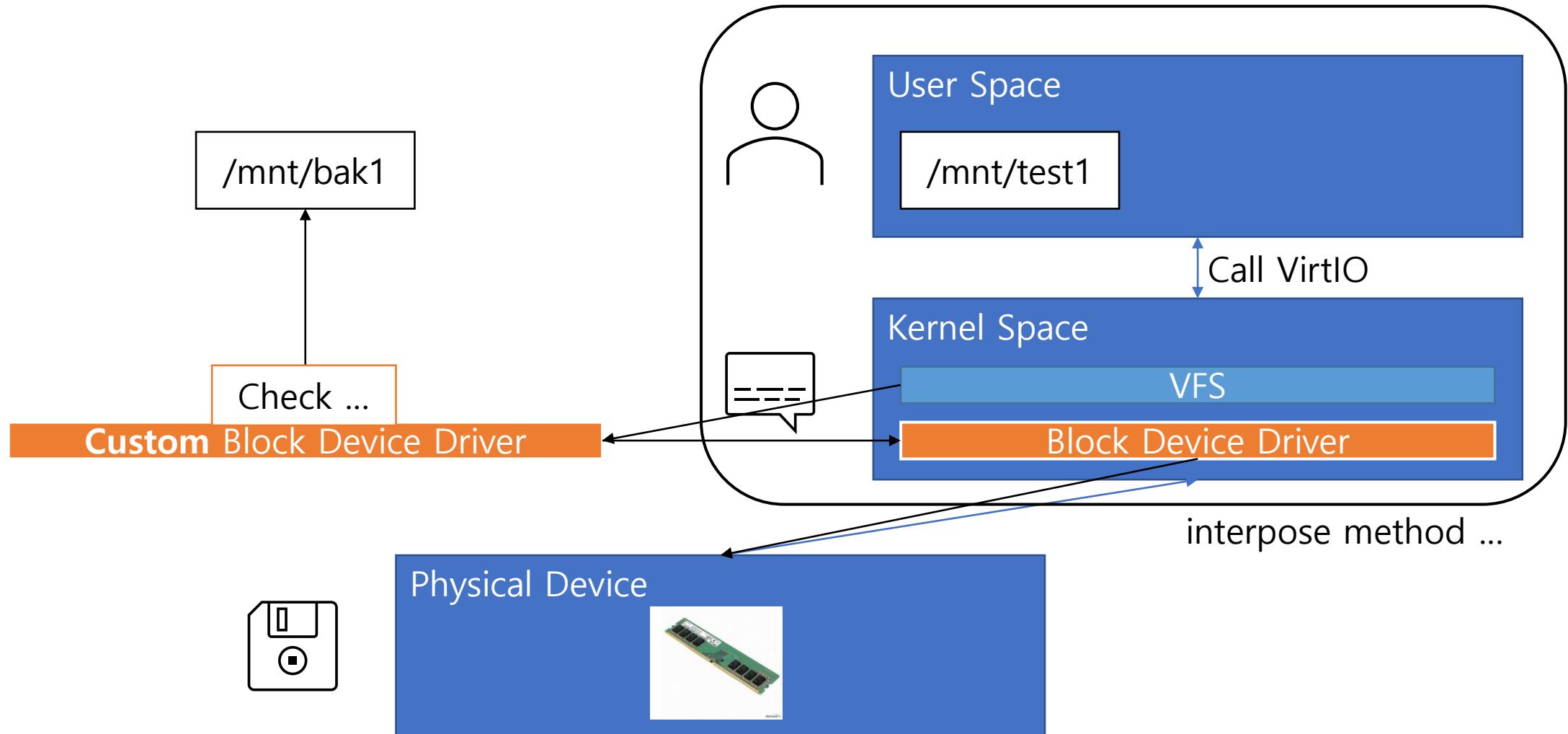
Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



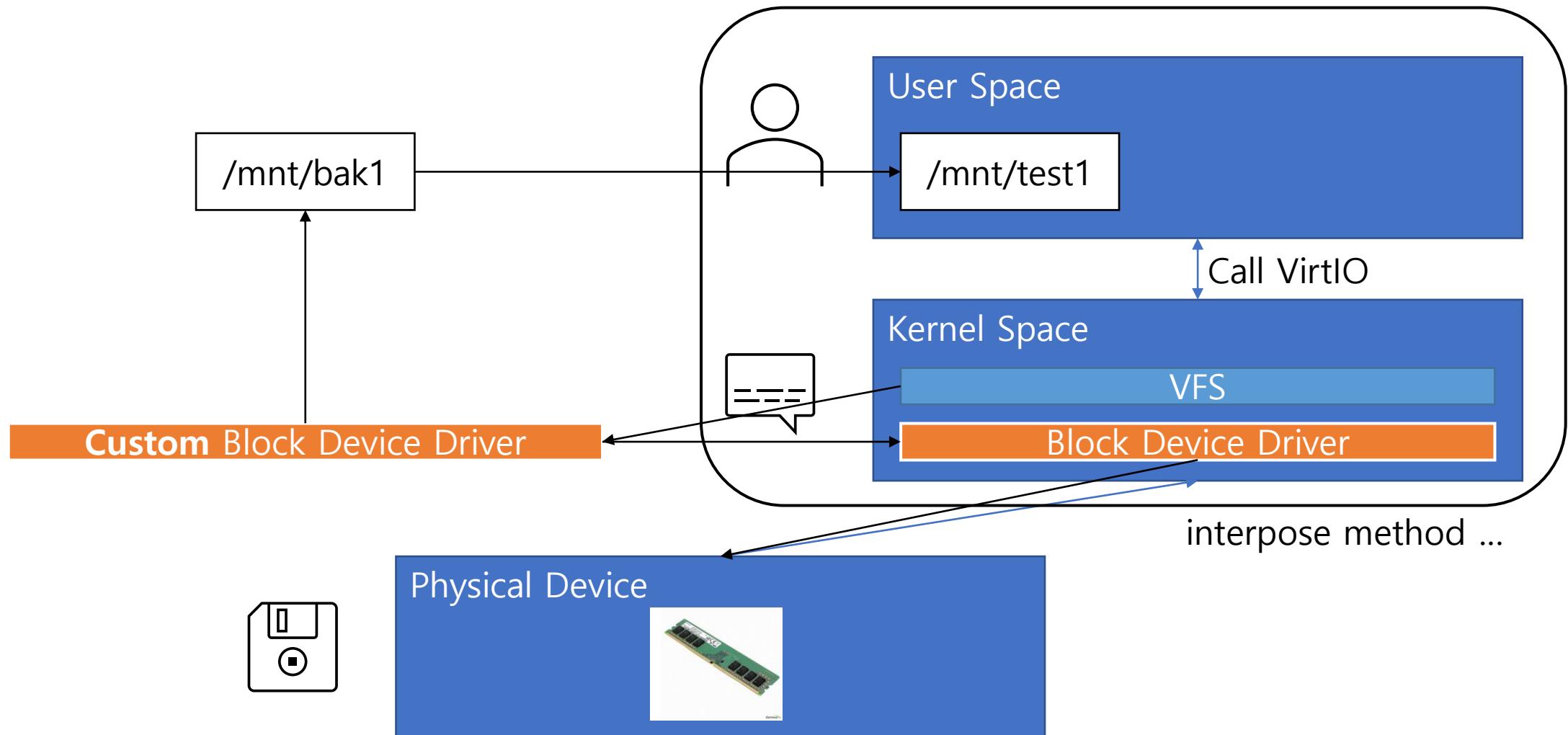
Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



Cloud Storage System Protecting From Ransomware Attack

KVM for cloud server user



Q&A

Thank you for listening

Edge Cloud Lab