

운영체제 실습

[Assignment #1]

Class : 목3

Professor: 최상호 교수님

Student ID: 2021202045

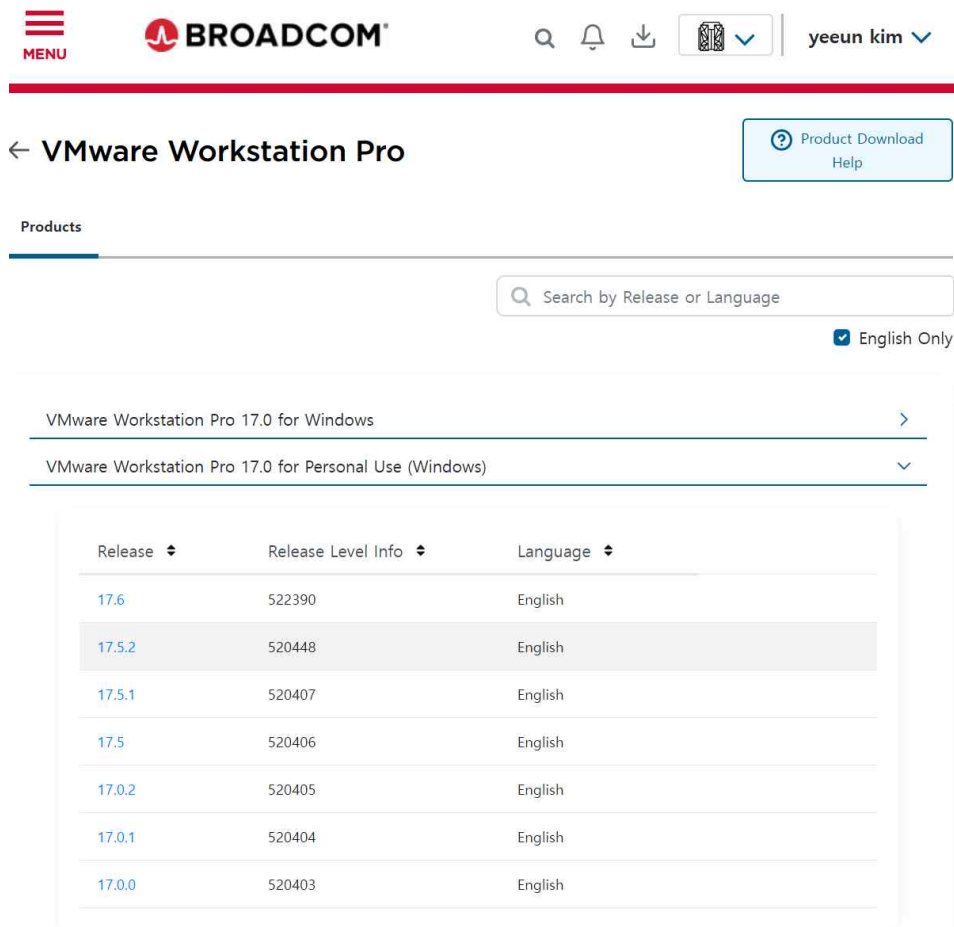
Name: 김예은

Introduction

운영체제 수업 첫 번째 과제로서, (1-1) 앞으로 사용할 VMware 및 Ubuntu 설치 후 수행 과정을 캡처해 설명 및 간단한 명령어를 terminal에 출력해본다. (1-2)는 kernel 5.4.282 버전을 환경설정을 하며 complie을 하는 건데, 4가지의 과정을 통해 compile한다. (1-3)은 ctags와 cscope을 이용하여 “Linux agpgart interface” 문자열을 찾아 과제의 의도에 맞게 문자열을 수정 및 추가한 후, dmesg를 통해 수행 결과를 확인한다. 여기서 우리가 설치하는 VMware는 하나의 컴퓨터로 여러 개의 운영체제를 사용하고 싶을 때 사용하는 것으로, 멀티 부팅과 달리 하나의 운영체제를 사용하기 위해 다른 운영체제를 꺼야 하는 시스템이 아니다.

결과화면

(1-1)



PRODUCT DOWNLOAD

← VMware Workstation Pro

Product Download Help

Products

Search by Release or Language

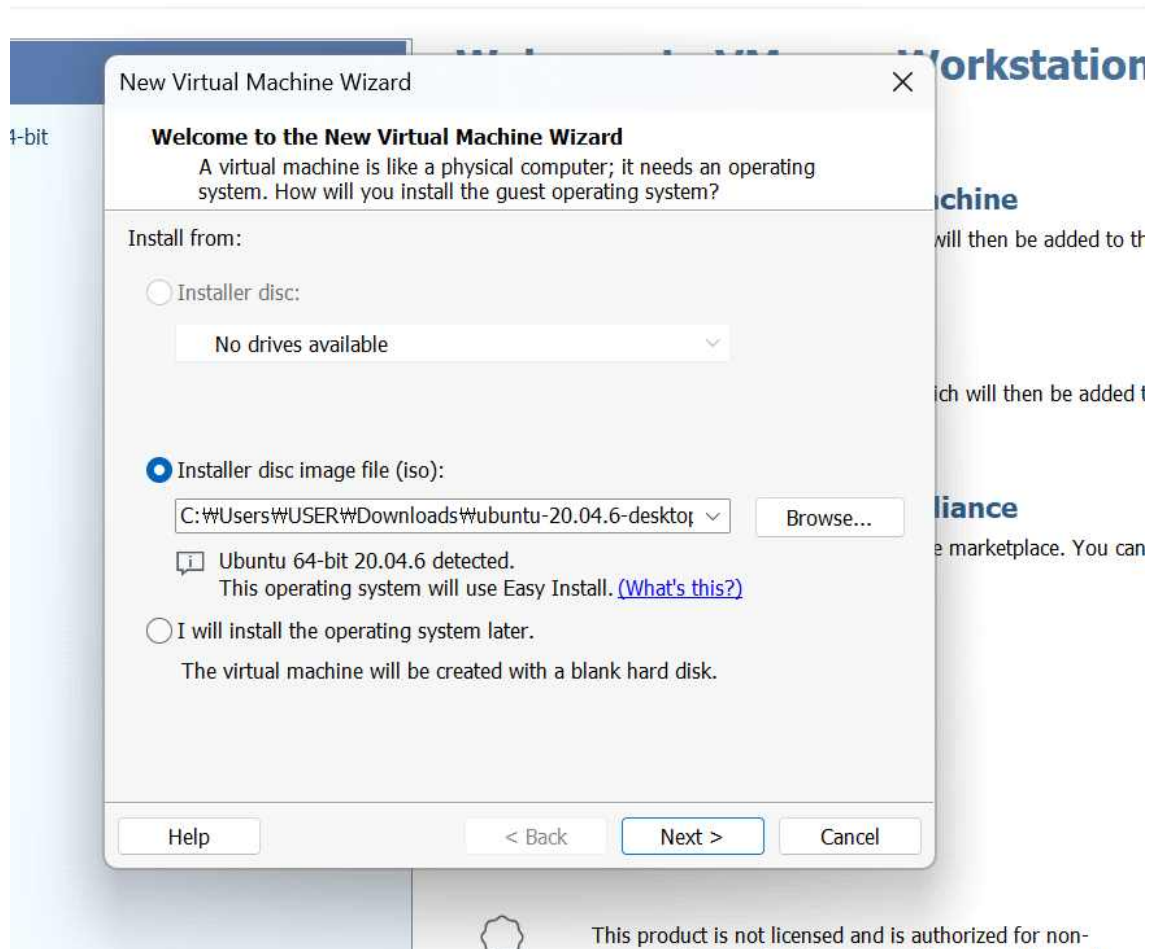
English Only

VMware Workstation Pro 17.0 for Windows

VMware Workstation Pro 17.0 for Personal Use (Windows)

Release	Release Level Info	Language
17.6	522390	English
17.5.2	520448	English
17.5.1	520407	English
17.5	520406	English
17.0.2	520405	English
17.0.1	520404	English
17.0.0	520403	English

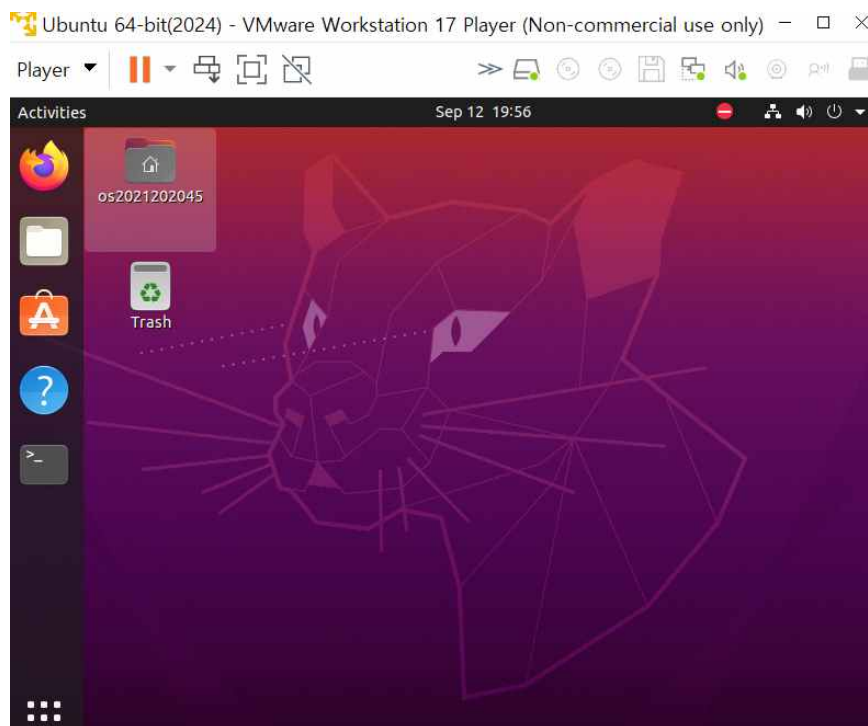
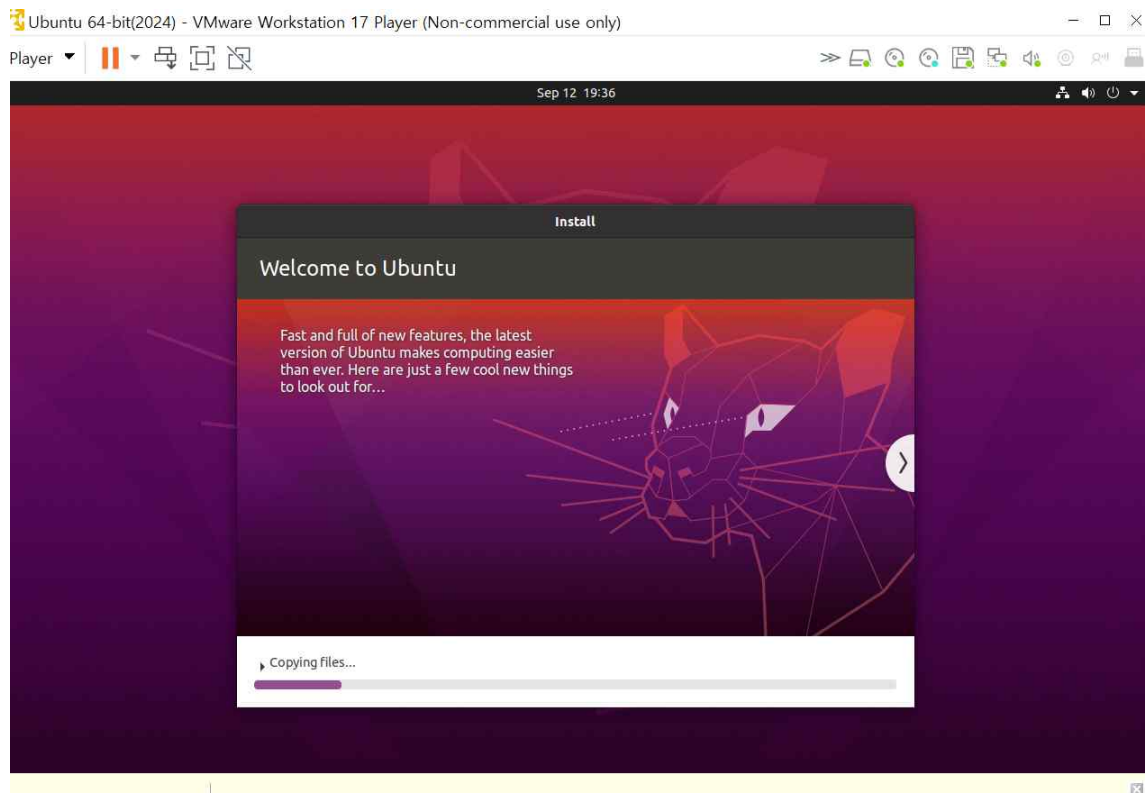
우선 BROADCOM에서 계정을 만든 후, VMware 17.5.2 버전을 다운받는다.



다운로드가 완료된 VMware에 다운받은 우분투 20.04.6를 설치해준다.



This product is not licensed and is authorized for non-



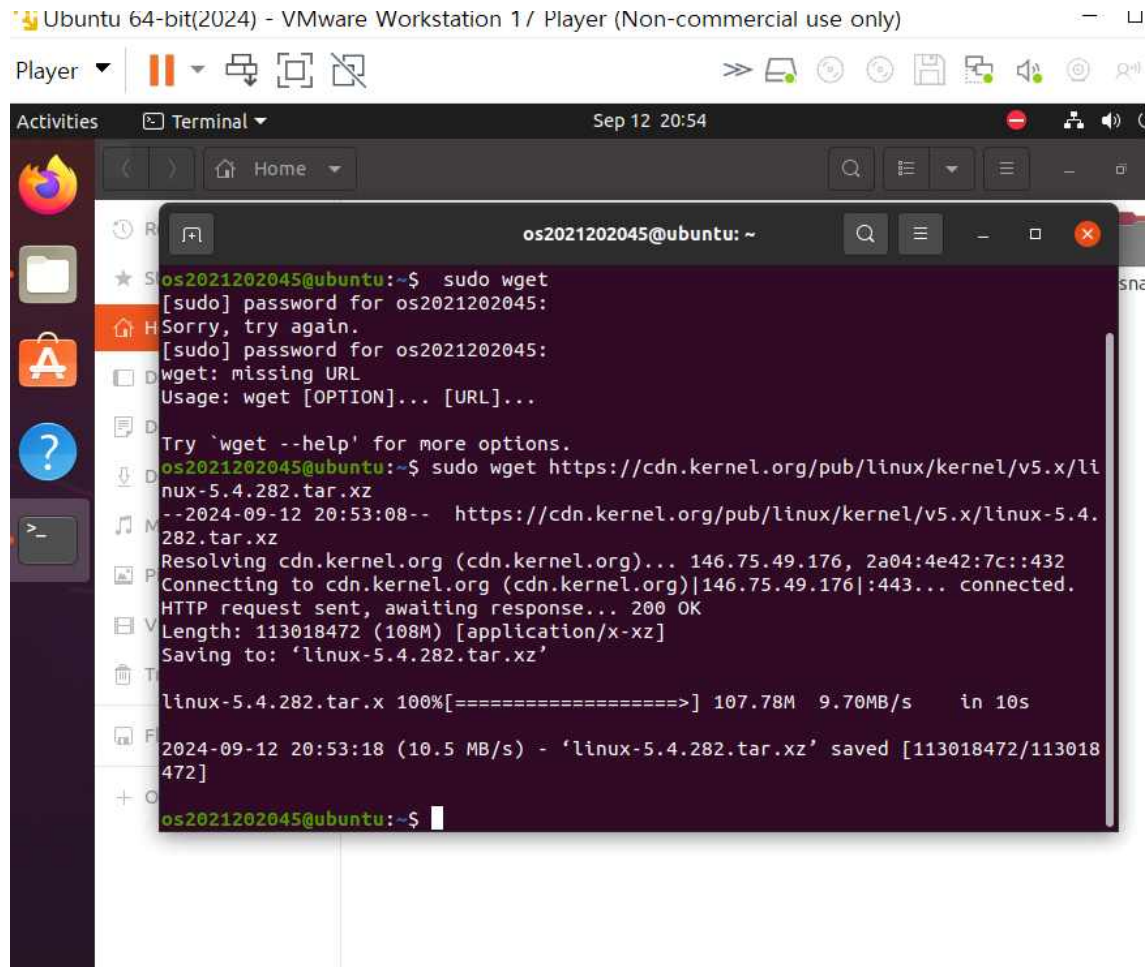
그럼 위와 같이 자동으로 설치가 완료되며, 자주 사용할 Terminal을 바로가기에 설정해준다. 또한 계정아이디는 os2021202045로 설정한다.

```
os2021202045@ubuntu: ~/work
Videos
work
os2021202045@ubuntu:~$ ls -l
total 44
drwxr-xr-x 2 os2021202045 os2021202045 4096 Sep 12 19:52 Desktop
drwxr-xr-x 2 os2021202045 os2021202045 4096 Sep 12 19:52 Documents
drwxr-xr-x 2 os2021202045 os2021202045 4096 Sep 12 19:52 Downloads
drwxr-xr-x 2 os2021202045 os2021202045 4096 Sep 12 19:52 Music
drwxr-xr-x 2 os2021202045 os2021202045 4096 Sep 12 19:52 Pictures
drwxr-xr-x 2 os2021202045 os2021202045 4096 Sep 12 19:52 Public
drwx----- 3 os2021202045 os2021202045 4096 Sep 12 19:53 snap
-rwxrwxrwx 1 os2021202045 os2021202045 2690 Sep 11 23:48 splab_commands
drwxr-xr-x 2 os2021202045 os2021202045 4096 Sep 12 19:52 Templates
drwxr-xr-x 2 os2021202045 os2021202045 4096 Sep 12 19:52 Videos
drwxrwxr-x 3 os2021202045 os2021202045 4096 Sep 12 19:59 work
os2021202045@ubuntu:~$ man ls
os2021202045@ubuntu:~$ pwd
/home/os2021202045
os2021202045@ubuntu:~$ cd ..
os2021202045@ubuntu:/home$ touch empty.txt
touch: cannot touch 'empty.txt': Permission denied
os2021202045@ubuntu:/home$ cd os2021202045
os2021202045@ubuntu:~$ cd work
os2021202045@ubuntu:~/work$ touch empty.txt
os2021202045@ubuntu:~/work$ ls
empty.txt file1.txt file2.txt file3.txt hello.txt SP_lab
os2021202045@ubuntu:~/work$ vi empty.txt
os2021202045@ubuntu:~/work$ cat empty.txt
Di1:1:
C

hello I am
os2021202045@ubuntu:~/work$ rm empty.txt
os2021202045@ubuntu:~/work$ ls
file1.txt file2.txt file3.txt hello.txt SP_lab
os2021202045@ubuntu:~/work$
```

ls -l을 통해 현재 디렉토리 내의 파일들의 디테일까지 출력한 후, man ls를 통해 ls 명령어의 매뉴얼을 볼 수 있다. pwd를 통해 현재위치의 path를 출력할 수 있으며 touch를 통해 empty.txt을 생성하여 ls를 통해 잘 생성되었는지 확인할 수 있었다. vi empty.txt 및 cat을 통해 file에 내용을 적고 출력한 뒤, rm을 통해 empty.txt를 삭제하는 명령을 했다.

(1-2)



```
os2021202045@ubuntu:~$ sudo wget
[sudo] password for os2021202045:
Sorry, try again.
[sudo] password for os2021202045:
wget: missing URL
Usage: wget [OPTION]... [URL]...

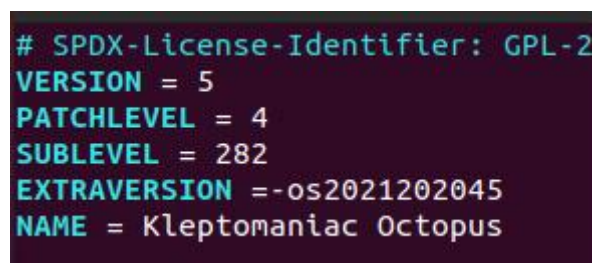
Try 'wget --help' for more options.
os2021202045@ubuntu:~$ sudo wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.4.282.tar.xz
--2024-09-12 20:53:08-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.4.282.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 146.75.49.176, 2a04:4e42:7c::432
Connecting to cdn.kernel.org (cdn.kernel.org)|146.75.49.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 113018472 (108M) [application/x-xz]
Saving to: 'linux-5.4.282.tar.xz'

linux-5.4.282.tar.x 100%[=====] 107.78M  9.70MB/s   in 10s

2024-09-12 20:53:18 (10.5 MB/s) - 'linux-5.4.282.tar.xz' saved [113018472/113018472]

os2021202045@ubuntu:~$
```

sudo를 이용해 root 권한으로 kernel 5.4.282버전을 설치 후, 코드 압축을 해제한 후 linux-5.4.282 directory로 cd를 이용하여 이동한다.



```
# SPDX-License-Identifier: GPL-2.0
VERSION = 5
PATCHLEVEL = 4
SUBLEVEL = 282
EXTRAVERSION = -os2021202045
NAME = Kleptomaniac Octopus
```

vi 명령어를 이용하여 kernel의 Makefile에서 extra version을 위와 같이 -os2021202045로 바꿔준다.

그 다음은 밑의 사진과 같이 명령어를 이용하여 kernel 환경설정을 해준다.

```
os2021202045@ubuntu:~/linux-5.4.282$ sudo make menuconfig
scripts/kconfig/mconf Kconfig

*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```

밑은 컴파일 시 문제가 될 수 있는 모듈을 제거하는 것이다.

```
DMABUF options --->
-* Auxiliary Display support --->
<M> Parallel port LCD/Keypad Panel support (OLD OPTION)
{M} Userspace I/O drivers --->
<*> VFIO Non-Privileged userspace driver framework --->
[*] Virtualization drivers --->
[*] Virtio drivers --->
    Microsoft Hyper-V guest support --->
    Xen driver support --->
<M> Greybus support --->
[*] Staging drivers ----
-* X86 Platform Specific Device Drivers --->
[ ] Platform support for Goldfish virtual devices ----
v(+)
```

<Select> <Exit> <Help> <Save> <Load>

밑의 사진은 컴파일 시 문제가 될 수 있는 옵션을 제거해주는 설정이다.

```
os2021202045@ubuntu: ~/linux-5.4.282
Config - Linux/x86 5.4.282-os2021202045 kernel Configuration
Binary Emulations

Binary Emulations
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

[*] IA32 Emulation
[ ] x32 ABI for 64-bit mode
```

밑은 커널 모듈 적재 시 발생할 수 있는 문제를 해결해주기 위해 Forced module loading을 체크해주는 것이다.

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < > module capable

```
-- Enable loadable module support
[*] Forced module loading
[*] Module unloading
[ ] Forced module unloading
[*] Module versioning support
[*] Source checksum for all modules
-* Module signature verification
[ ] Require modules to be validly signed
[*] Automatically sign all modules
Which hash algorithm should modules be signed with? (Sign modules with SH
[ ] Compress modules on installation
[ ] Allow loading of modules with missing namespace imports
[*] Enable unused/obsolete exported symbols
```

밑과 같이 CONFIG_SYSTEM_REVOACTION_KEYS와 CONFIG_SYSTEM_TRUSTED_KEYS를 disable 시킴으로써 시스템 폐기키 및 보안 키를 활성화해주었다.

```
# end of Kernel hacking
# CONFIG_SYSTEM_REVOCATION_KEYS is not set
```

```
CONFIG_MODULE_SIG_KEY="certs/signing_k
CONFIG_SYSTEM_TRUSTED_KEYRING=y
CONFIG_SYSTEM_TRUSTED_KEYS=""
CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4
CONFIG_SECONDARY_TRUSTED_KEYRING=y
```

밑의 두 사진은 initramfs의 설정을 변경하기 위해 해당 파일에 write할 수 있는 권한을 준 후, vi를 통해 initramfs.conf의 MODULE부분을 most에서 dep로 변경한 것이다. 이는 initd is too big이라는 에러 메시지를 방지하기 위한 것이다.

```
CLEAN arch/x86/tools
os2021202045@ubuntu:~/linux-5.4.282$ cd /etc/initramfs-tools
os2021202045@ubuntu:/etc/initramfs-tools$ vi initramfs.conf
os2021202045@ubuntu:/etc/initramfs-tools$ ls -l initramfs.conf
-rw-r--r-- 1 root root 1807 Jun 18 2021 initramfs.conf
os2021202045@ubuntu:/etc/initramfs-tools$ chmod a+w initramfs.conf
chmod: changing permissions of 'initramfs.conf': Operation not permitted
os2021202045@ubuntu:/etc/initramfs-tools$ vi initramfs.conf
os2021202045@ubuntu:/etc/initramfs-tools$ ls -l initramfs.conf
-rw-r--r-- 1 root root 1807 Jun 18 2021 initramfs.conf
os2021202045@ubuntu:/etc/initramfs-tools$ sudo chmod a+w initramfs.conf
[sudo] password for os2021202045:
os2021202045@ubuntu:/etc/initramfs-tools$ ls -l initramfs.conf
-rw-rw-rw- 1 root root 1807 Jun 18 2021 initramfs.conf
os2021202045@ubuntu:/etc/initramfs-tools$ ls -l initramfs.conf
-rw-rw-rw- 1 root root 1807 Jun 18 2021 initramfs.conf
os2021202045@ubuntu:/etc/initramfs-tools$ vi initramfs.conf
os2021202045@ubuntu:/etc/initramfs-tools$
```



```
os2021202045@ubuntu: /etc/initramfs-tools
#
# 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=dep
```

본인의 경우 VMware에 적용된 코어 수가 2였기 때문에 make -j4 명령어를 통해 kernel을 compile해주었다.

```
os2021202045@ubuntu: /$ cd ~/linux-5.4.282
os2021202045@ubuntu: ~/linux-5.4.282$ make -j4
HOSTCC      scripts/basic/fixdep
DESCEND     objtool
HOSTCC      /home/os2021202045/linux-5.4.282/tools/objtool/fixdep.o
HOSTLD      /home/os2021202045/linux-5.4.282/tools/objtool/fixdep-in.o
LINK        /home/os2021202045/linux-5.4.282/tools/objtool/fixdep
CC          /home/os2021202045/linux-5.4.282/tools/objtool/exec_cmd.o
CC          /home/os2021202045/linux-5.4.282/tools/objtool/help.o
CC          /home/os2021202045/linux-5.4.282/tools/objtool/pager.o
CC          /home/os2021202045/linux-5.4.282/tools/objtool/page_sort.o
LD [M]      sound/soc/intel/atom/snd-soc-sst-atom-hifi2-platform.ko
LD [M]      sound/soc/intel/atom/snd-soc-sst-intel-sst-acpi.ko
LD [M]      sound/soc/intel/atom/snd-soc-sst-intel-sst-core.ko
LD [M]      sound/soc/intel/atom/snd-soc-sst-intel-sst-pcl.ko
LD [M]      sound/soc/intel/boards/snd-soc-skl_nau88125_max98357a.ko
LD [M]      sound/soc/intel/boards/snd-soc-skl_da7219_max98357a.ko
LD [M]      sound/soc/intel/boards/snd-soc-skl_da7219_max98927.ko
LD [M]      sound/soc/intel/boards/snd-soc-skl_rt5660.ko
LD [M]      sound/soc/intel/boards/snd-soc-skl_rt5663_max98927.ko
LD [M]      sound/soc/intel/boards/snd-soc-skl_rt5663_rt5514_max98927.ko
LD [M]      sound/soc/intel/boards/snd-soc-skl_hda_dsp.ko
LD [M]      sound/soc/intel/boards/snd-soc-skl_nau88125_ssm4567.ko
LD [M]      sound/soc/intel/boards/snd-soc-skl_rt286.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-bxt-da7219_max98357a.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-bxt-rt298.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-byt-cht-cx2072x.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-byt-cht-es8316.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-byt-cht-da7213.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-bytcr-rt5640.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-bytcr-rt5651.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-cht-bsw-nau98090_tl.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-cht-bsw-nau8824.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5645.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5672.ko
LD [M]      sound/soc/intel/boards/snd-soc-sst-glk-rt5682_max98357a.ko
LD [M]      sound/soc/intel/common/snd-soc-acpi-intel-match.ko
LD [M]      sound/soc/intel/common/snd-soc-sst-dsp.ko
LD [M]      sound/soc/intel/common/snd-soc-sst-ipc.ko
LD [M]      sound/soc/snd-soc-acpi.ko
LD [M]      sound/soc/snd-soc-core.ko
LD [M]      sound/soc/sof/snd-sof-acpi.ko
LD [M]      sound/soc/sof/snd-sof-pcl.ko
LD [M]      sound/soc/sof/snd-sof.ko
LD [M]      sound/soc/xilinx/snd-soc-xlnx-formatter-pcm.ko
LD [M]      sound/soc/xilinx/snd-soc-xlnx-i2s.ko
LD [M]      sound/soc/xilinx/snd-soc-xlnx-spdif.ko
LD [M]      sound/soc/xtensa/snd-soc-xtfpga-i2s.ko
LD [M]      sound/soundcore.ko
LD [M]      sound/synth/emux/snd-emux-synth.ko
LD [M]      sound/synth/snd-util-men.ko
LD [M]      sound/usb/ofire/snd-usb-ofire.ko
LD [M]      sound/usb/bcd2000/snd-bcd2000.ko
LD [M]      sound/usb/caiaq/snd-usb-caiaq.ko
LD [M]      sound/usb/hiface/snd-usb-hiface.ko
LD [M]      sound/usb/line6/snd-usb-line6.ko
LD [M]      sound/usb/line6/snd-usb-pod.ko
LD [M]      sound/usb/line6/snd-usb-podhd.ko
LD [M]      sound/usb/line6/snd-usb-toneport.ko
LD [M]      sound/usb/line6/snd-usb-variax.ko
LD [M]      sound/usb/misc/snd-ua101.ko
LD [M]      sound/usb/snd-usb-audio.ko
LD [M]      sound/usb/snd-usbmidi1.ko
LD [M]      sound/usb/usx2y/snd-usb-usx2l.ko
LD [M]      sound/usb/usx2y/snd-usb-usx2y.ko
LD [M]      sound/x86/snd-hdmi-lpe-audio.ko
LD [M]      sound/xen/snd_xen_front.ko
os2021202045@ubuntu: ~/linux-5.4.282$
```

위와 같이 kernel compile이 오류 없이 잘 compile 된 것을 확인할 수 있다.

그 다음 make modules_install 명령어를 통해 module을 install를 해주었다.

```
INSTALL sound/x86/snd-hdmi-lpe-audio.ko
INSTALL sound/xen/snd_xen_front.ko
DEPMOD 5.4.282-os2021202045
os2021202045@ubuntu:~/linux-5.4.282$
```

compile된 kenrenl을 부트 로더에 등록하기 위해 아래의 과정을 거친 뒤

```
os2021202045@ubuntu:~/linux-5.4.282$ sudo make install
sh ./arch/x86/boot/install.sh 5.4.282-os2021202045 arch/x86/boot/bzImage \
    System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.4.282-os2021202045 /b
oot/vmlinuz-5.4.282-os2021202045
update-initramfs: Generating /boot/initrd.img-5.4.282-os2021202045
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.4.282-os202120204
5 /boot/vmlinuz-5.4.282-os2021202045
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.4.282-os2021202045 /b
oot/vmlinuz-5.4.282-os2021202045
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.4.282-os202120
2045 /boot/vmlinuz-5.4.282-os2021202045
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-121-generic
I: /boot/initrd.img is now a symlink to initrd.img-5.4.282-os2021202045
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.4.282-os2021202045 /bo
ot/vmlinuz-5.4.282-os2021202045
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.15.0-121-generic
Found initrd image: /boot/initrd.img-5.15.0-121-generic
Found linux image: /boot/vmlinuz-5.15.0-67-generic
Found initrd image: /boot/initrd.img-5.15.0-67-generic
```

grub 설정 파일을 다음과 같이 수정해주었다.

GRUB_TIMEOUT_STYLE = menu

GRUB_TIMEOUT=10

GRUB_CMDLINE_LINUX_DEFAULT="quiet splash nokaslr"로 바꾸어주었다.

```
os2021202045@ubuntu: ~/linux-5.4.282
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash nokaslr"
GRUB_CMDLINE_LINUX="find_preseed=/preseed.cfg auto noprompt priority=critical local
e=en_US"

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command 'vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true
```

수정 후 grub 옵션을 업데이트 및 리부팅해 준 뒤, compile한 kernel 버전의 ubuntu를 선택 후 terminal에 가서 커널 버전을 확인하면

```
os2021202045@ubuntu:~$ uname -r
5.4.282-os2021202045
```

다음과 같이 커널 버전과 os학번이 잘 나온다.

(1-3)

```
os2021202045@ubuntu: ~/linux-5.4.282
C symbol: agp
File      Function      Line
0 backend.c <global>      101 static const struct { int mem,
    agp; } maxes_table[] = {
1 pci.h     <global>      26  } agp;
2 nouveau_drv.h <global>     148 } agp;
3 via_verifier.h <global>     53 int agp;
4 via_verifier.h <global>     58 struct drm_device *dev, int
    agp);

* Lines 1-6 of 351, 346 more - press the space bar to display more *

Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

위와 같이 cscope를 이용하여 agp 변수가 쓰인 곳을 찾아주는 것을 통해 cscope를 연습했다.

밑의 명령어를 통해 task_struct가 있는 곳부터 vi를 시작하게 했다.

```
os2021202045@ubuntu:~/linux-5.4.282$ vi -t task_struct
os2021202045@ubuntu:~/linux-5.4.282$
```



```

struct task_struct {
#ifdef CONFIG_THREAD_INFO_IN_TASK
    /*
     * For reasons of header soup (see current_thread_info()), this
     * must be the first element of task_struct.
     */
    struct thread_info          thread_info;
#endif
    /* -1 unrunnable, 0 runnable, >0 stopped: */
    volatile long               state;

    /*
     * This begins the randomizable portion of task_struct. Only

```

:ts schedule을 입력하여 schedule로 검색한 결과를 한 눈에 보게하는 연습을 했다.

```

# pri kind tag file
1 F m schedule drivers/gpu/drm/i915/gt/intel_engine_types.h
    struct:intel_engine_cs
    void (*schedule)(struct i915_request *request,
2 F m schedule drivers/net/wireless/ath/wcn36xx/hal.h
    struct:wcn36xx_hal_ts_info typeref:struct:wcn36xx_hal_ts_info::wcn36xx_hal_ts_info
    _sch
    struct wcn36xx_hal_ts_info_sch schedule;
3 F m schedule drivers/net/wireless/ath/wcn36xx/hal.h
    struct:wcn36xx_hal_ts_info_sch
    u8 schedule:1;
4 F m schedule drivers/net/wireless/intel/iwlwifi/fw/api/scan.h
    struct:iwl_scan_req_lmac typeref:struct:iwl_scan_req_lmac::iwl_scan_schedule_lmac
    struct iwl_scan_schedule_lmac schedule[IWL_MAX_SCHED_SCAN_PLANS];
5 F m schedule drivers/net/wireless/intel/iwlwifi/fw/api/scan.h
    struct:iwl_scan_req_umac_tail_v1 typeref:struct:iwl_scan_req_umac_tail_v1::iwl_sca
    n_umac_schedule
    struct iwl_scan_umac_schedule schedule[IWL_MAX_SCHED_SCAN_PLANS];
6 F m schedule drivers/net/wireless/intel/iwlwifi/fw/api/scan.h
    struct:iwl_scan_req_umac_tail_v2 typeref:struct:iwl_scan_req_umac_tail_v2::iwl_sca
    n_umac_schedule
    struct iwl_scan_umac_schedule schedule[IWL_MAX_SCHED_SCAN_PLANS];
7 F m schedule drivers/net/wireless/mediatek/mt76/mt76.h
    struct:mt76_queue_entry
    bool schedule:1;
8 F m schedule drivers/usb/host/isp116x.h
    struct:isp116x_ep typeref:struct:isp116x_ep::list_head
    struct list_head schedule;
9 F m schedule drivers/usb/host/isp1362.h
    struct:isp1362_ep typeref:struct:isp1362_ep::list_head
    struct list_head schedule; /* list of all EPs that need processing */
10 F m schedule drivers/usb/host/sl811.h
    struct:sl811h_ep typeref:struct:sl811h_ep::list_head
    struct list_head schedule;
11 F m schedule include/net/ip_vs.h
    struct:ip_vs_scheduler typeref:struct:ip_vs_scheduler::schedule
    struct ip_vs_dest* (*schedule)(struct ip_vs_service *svc,
12 F v schedule kernel/sched/core.c
    EXPORT_SYMBOL(schedule);
13 F f schedule kernel/sched/core.c
    asmlinkage __visible void __sched schedule(void)
14 FS m schedule drivers/net/usb/r8152.c
    struct:r8152 typeref:struct:r8152::delayed_work
    struct delayed_work schedule, hw_phy_work;
15 FS m schedule drivers/scsi/ncr53c8xx.c
    struct:launch typeref:struct:launch::link
    struct link schedule; /* Jump to scheduler point */
More --

```

위를 통해 Ctags와 Cscope tool을 사용하는 것을 연습한 뒤, 과제의 목적인 Linux agpgart interface가 실행되는 지점을 Cscope의 Find this text string을 활용하여 찾아냈다. Text string은 agpgart였고, 그 결과 다음의 사진과 같이 Linux agpgart interface 문자열이 있는 backend.c file이 k번째에 검색되는 것을 확인할 수 있다.

k를 엔터해서 들어가서 해당 문자열이 출력되는 곳에 과제의 의도에 맞게 수정해주면 다음과 같이 코드가 완성된다.

```

os2021202045@ubuntu: ~/linux-5.4.282
Text string: agpgart

File      Line
0 core_irongate.c 255 alpha_agpgart_size = 0;
1 core_irongate.c 303 #include <linux/agpgart.h>
2 core_irongate.c 322 if (!alpha_agpgart_size)
3 core_irongate.c 336 gart_bus_addr + alpha_agpgart_size)
4 core_marvel.c 912 if (!alpha_agpgart_size)
5 core_marvel.c 919 aper->pg_count = alpha_agpgart_size / PAGE_SIZE;
6 core_marvel.c 968 * The agpgart_be code has not programmed the card yet,
7 core_titan.c 595 if (!alpha_agpgart_size)
8 core_titan.c 603 aper->pg_count = alpha_agpgart_size / PAGE_SIZE;
9 pci_impl.h 170 extern unsigned long alpha_agpgart_size;
a setup.c 115 unsigned long alpha_agpgart_size = DEFAULT_AGP_APER_SIZE;
b setup.c 521 alpha_agpgart_size =
c sba_iommu.c 1636 printk(KERN_INFO PFX "reserving %dMb of IOVA space at 0x%lx for agpgart\n",
d agp.h 9 * Functions to keep the agpgart mappings coherent with the MMU. The
e agp.h 34 #define PFX "agpgart: "
f ali-agp.c 399 .name = "agpgart-ali",
g amd-k7-agp.c 543 .name = "agpgart-amdk7",
h amd64-agp.c 742 .name = "agpgart-amd64",
i ati-agp.c 559 .name = "agpgart-ati",
j backend.c 38 #include <linux/agpgart.h>
k backend.c 338 printk(KERN_INFO "Linux agpgart interface v%d.%d\n",

* Lines 1-22 of 50, 29 more - press the space bar to display more *
Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:

```

```

static int __init agp_init(void)
{
    if (!agp_off){
        printk(KERN_INFO "os2021202045_Linux agpgart interface v%d.%d\n",
            AGPGART_VERSION_MAJOR, AGPGART_VERSION_MINOR); //add os2021202045_
        printk(KERN_INFO "os2021202045_arg in agp_init(void)\n");} //print function and arg
    return 0;
}

```

backend.c파일에 “Linux agpgart interface” 문자열이 있고, 이 부분의 소스코드를 수정했으니 backend.c 파일의 경로를 찾기 위해, Cscope의 파일 위치를 찾아주는 역할을 이용해 backend.c 파일의 위치를 찾으니 다음과 같이 결과가 나왔다.

```

1+1
os2021202045@ubuntu: ~/linux-5.4.282
File: backend.c

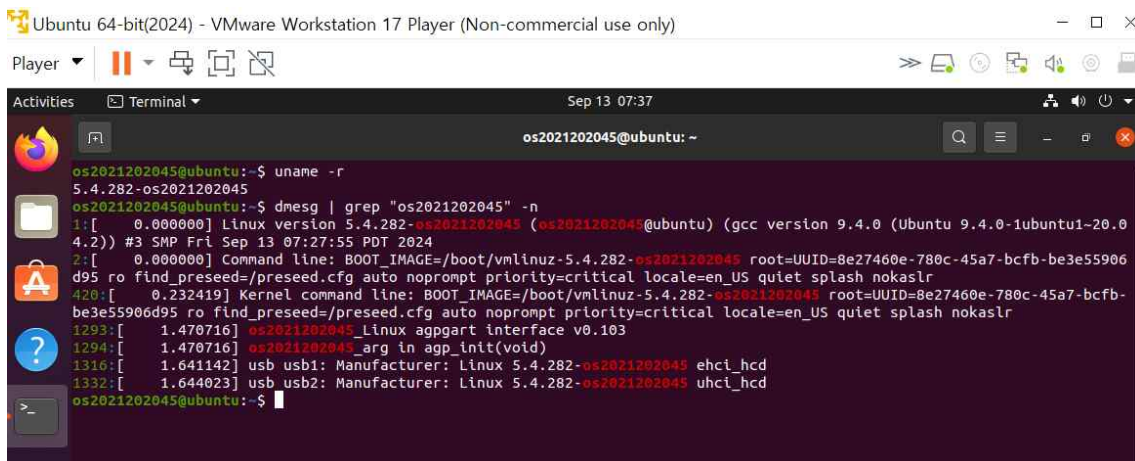
File
0 drivers/char/agp/backend.c
1 drivers/gpu/drm/sun4i/sun4i_backend.c
2 drivers/gpu/drm/ttm/ttm_agp_backend.c
3 drivers/xen/xenbus/xenbus_dev_backend.c
4 drivers/xen/xenbus/xenbus_probe_backend.c

```


즉 본인이 수정한 파일의 path는 0번째인 drivers/char/agp/backend.c이다.

```
DEPMOD 5.4.282-os2021202045
os2021202045@ubuntu:~/linux-5.4.282$ sudo make install
sh ./arch/x86/boot/install.sh 5.4.282-os2021202045 arch/x86/boot/bzImage \
    System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.4.282-os2021202045 /boot/vmlinuz-5.4.282-os2021202045
update-initramfs: Generating /boot/initrd.img-5.4.282-os2021202045
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.4.282-os2021202045 /boot/vmlinuz-5.4.282-os2021202045
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.4.282-os2021202045 /boot/vmlinuz-5.4.282-os2021202045
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.4.282-os2021202045 /boot/vmlinuz-5.4.282-os2021202045
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.4.282-os2021202045 /boot/vmlinuz-5.4.282-os2021202045
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.15.0-121-generic
Found initrd image: /boot/initrd.img-5.15.0-121-generic
Found linux image: /boot/vmlinuz-5.15.0-67-generic
Found initrd image: /boot/initrd.img-5.15.0-67-generic
Found linux image: /boot/vmlinuz-5.4.282-os2021202045
Found initrd image: /boot/initrd.img-5.4.282-os2021202045
Found linux image: /boot/vmlinuz-5.4.282-os2021202045.old
Found initrd image: /boot/initrd.img-5.4.282-os2021202045
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
os2021202045@ubuntu:~/linux-5.4.282$ reboot
```

부팅을 한 다음, dmesg 및 grep 명령어로 “os2021202045” 문자열이 출력된 메시지를 찾았다.



```
os2021202045@ubuntu:~$ uname -r
5.4.282-os2021202045
os2021202045@ubuntu:~$ dmesg | grep "os2021202045" -n
1:[ 0.000000] Linux version 5.4.282-os2021202045 (os2021202045@ubuntu) (gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.2)) #3 SMP Fri Sep 13 07:27:55 PDT 2024
2:[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-5.4.282-os2021202045 root=UUID=8e27460e-780c-45a7-bcfb-be3e55906d95 ro find_preseed=/preseed.cfg auto noprompt priority=critical locale=en_US quiet splash nokaslr
420:[ 0.232419] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-5.4.282-os2021202045 root=UUID=8e27460e-780c-45a7-bcfb-be3e55906d95 ro find_preseed=/preseed.cfg auto noprompt priority=critical locale=en_US quiet splash nokaslr
1293:[ 1.470716] os2021202045_Linux agpgart interface v0.103
1294:[ 1.470716] os2021202045_arg in agp_init(void)
1316:[ 1.641142] usb usb1: Manufacturer: Linux 5.4.282-os2021202045 ehci_hcd
1332:[ 1.644023] usb usb2: Manufacturer: Linux 5.4.282-os2021202045 uhci_hcd
os2021202045@ubuntu:~$
```

위의 1293: os2021202045_Linux agpgart interface v0.103,
1294: os2021202045_arg in agp_init(void) 즉, 해당 interface를 실행시키는 함수의 함수명과 argument값이 중복 없이 출력되고 있음을 볼 수 있다.

고찰

1-1의 경우 문제 없이 설치하였지만 1-2에서 kernel을 compile할 때 VMware에 할당된 코어의 수가 2개인 이유와 저사양의 노트북이라는 환경 때문에 compile에서 약 4시간이 걸렸다. 다행히 에러없이 한번에 compile이 잘 되었다. 컴파일 환경설정을 할 때 vi로 기존 파일을 고치는 일이 많았는데 그때마다 write할 권한이 없어서 맨 처음에는 chmod를 이용하여 write권한을 파일에 준 후, 고쳤는데 그 다음부터는 sudo를 이용하여 root 권한을 부여하여 명령어를 사용하였다. 1-3의 경우 비교적 Ctags의 연습은 쉽게 했는데, Cscope는 다루기 힘들어 힘들었지만, 차근 차근 Cscope의 기본 명령어 (ex) Find this text string를 해석하니 나름 쉽게 사용할 수 있겠다 싶었다. 맨처음에 “Linux agpgart interface”를 찾기 위해 C symbol이나 find functions에 agpgart와 interface를 검색했는데 유효한 결과가 없어서 포기하고 싶었던 찰나, 그러고 보니 찾고자하는 것은 문자열이라는 사실을 깨닫고 find this text string에 agpgart를 검색했는데 바로 찾고자하는 문자열이 backend.c파일에 있다는 것이 검색되었다. 그 다음 바로 backend.c파일의 위치도 Cscope를 이용해 path를 찾아내니 Cscope가 정말 편한 tool임을 깨달을 수 있었다.

Reference

VMware 개념:

<https://growthjournal.tistory.com/entry/1-VMware%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80>

cscope:https://hoonkyu.blogspot.com/2012/05/cscope_5559.html

ctag: <https://bowbowbow.tistory.com/15>