블록 디바이스 드라이버

목표

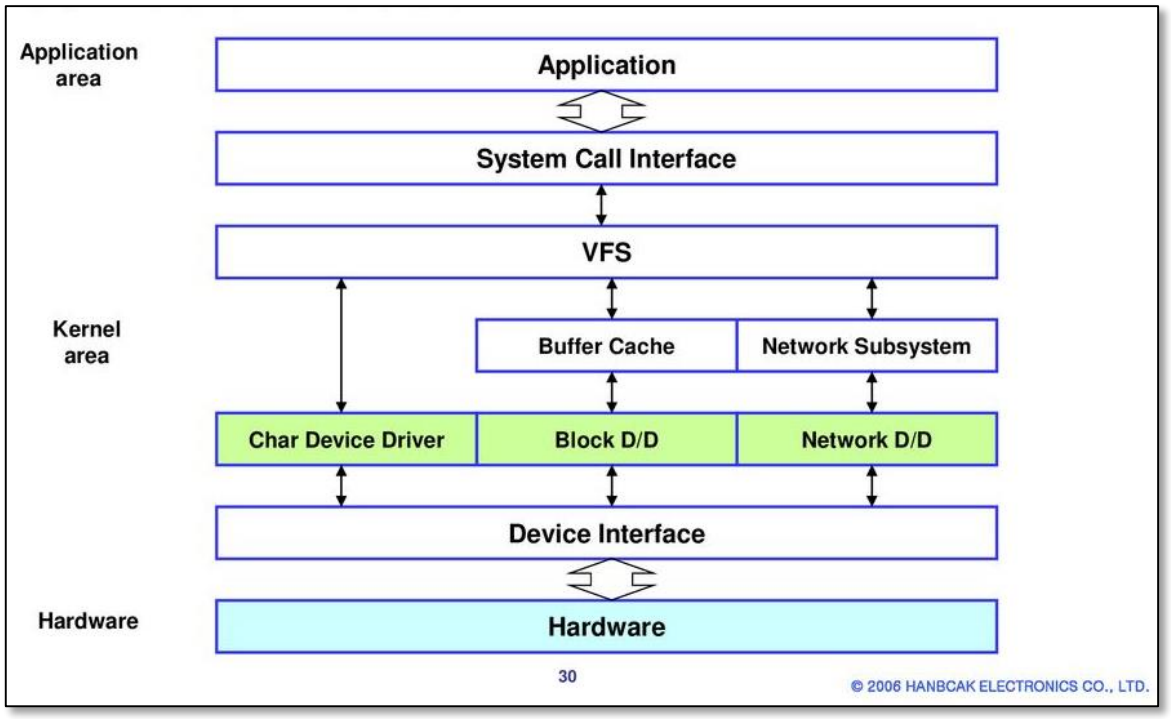
- Raspberry Pi에 자신이 제작한 블록 디바이스 드라이버를 커널에 빌드
- 자신이 제작한 블록 디바이스 드라이버를 직접 사용해보기

준비물

- 호스트 시스템에서 타겟 시스템의 kernel 소스코드**(이번 실습은 반드시 4.x.x의 커널 버전을 사용)**
  *호스트 시스템에 타겟 시스템의 Kernel 소스코드가 없다면 앞선 *"라즈베리파이 커널 컴파일"* 실습 내용 참조
- 호스트 시스템에서 타겟 시스템의 툴 체인 환경(Cross Compiler, Make 등)
  *호스트 시스템에 타겟 시스템의 툴 체인 환경이 없다면 앞선 *"라즈베리파이 커널 컴파일"* 실습 내용 참조

블록 디바이스 드라이버란?

- 리눅스는 디바이스(HDD, 키보드, 마우스 등)들을 파일로 취급하며 해당하는 디바이스 파일에 맞춘 드라이버를 제작하여 특정 디바이스를 제어한다.
- 본 실습에서 구현할 블록 디바이스 드라이버는 앞선 GPIO와 캐릭터 디바이스 드라이버와 다르게 버퍼나 캐시를 거친 뒤 디바이스의 내용을 입·출력한다.
- 블록 디바이스 드라이버 구조



타겟 시스템에 자신의 블록 디바이스 드라이버 모듈 등록

- *본 실습에서는 타겟 시스템 커널의 버전을 4.19.x로 사용하였다.
  이번 실습은 반드시 타겟 시스템의 커널 버전을 4.x.x대로 맞추어서 진행하여야 한다.

- 본 실습의 1.~3.1. 까지는 호스트 시스템에서 진행되는 것이며, 3.2. 부턴 타겟 시스템에서 진행된다.

1. 블록 디바이스 드라이버 모듈 예제 코드 다운로드

1.1. 예제 코드를 받기 위해 Git 설치

명령어 : sudo apt-get install git

1.2. 저장소로부터 예제 코드 받기

명령어 : git clone https://github.com/INS-LAB/block-dev-driver.git

```
inslab-test-server@computer:~/git/rpi_kernel$ git clone https://github.com/INS-LAB/block-dev-driver.git
Cloning into 'block-dev-driver'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 1), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), 2.25 KiB | 1.12 MiB/s, done.
inslab-test-server@computer:~/git/rpi_kernel$ ls
block-dev-driver  kernel  linux  modules
```

>> git clone 명령어는 특정 저장소를 복사해오는 명령어이다.

>> 위 명령어를 실행하였다면 *block-dev-driver*라는 디렉터리가 하나 생성되었을 것이다.

2. 예제 코드 크로스 컴파일 진행

2.1. 다운로드 받은 예제 코드 디렉터리(block-dev-driver)로 이동

명령어 : cd block-dev-driver

```
inslab-test-server@computer:~/git/rpi_kernel/block-dev-driver$ ls
Makefile  vrd_dd.c
```

>> 해당 디렉터리 내에는 2개의 파일이 존재

>> Makefile : 블록 디바이스 드라이버 파일(vrd_dd.c)을 컴파일 하기 위한 파일

>> vrd_dd.c : 블록 디바이스 드라이버 예제 소스 코드 파일

2.2. 컴파일 전 Makefile 내용 수정

명령어 : vi Makefile

```
obj-m = vrd_dd.o

KDIR := # Please, insert the Rpi kernel directory path
PWD := $(shell pwd)

default:
	make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -C $(KDIR) M=$(PWD) modules

clean:
	make -C $(KDIR) M=$(PWD) clean
```

>> 파일의 내용 중 KDIR 변수에 타겟 시스템의 커널 경로를 작성

>> 앞선 *"라즈베리파이 커널 컴파일"* 실습 때 사용한 디렉터리의 경로를 작성

```
inslab-test-server@computer:~/git/rpi_kernel/linux$ ls
COPYING         LICENSES        System.map  crypto    init     modules.builtin  security  vmlinux
CREDITS         MAINTAINERS     arch        drivers   ipc      modules.order    sound     vmlinux.o
Documentation   Makefile        block       firmware  kernel   net              tools
Kbuild          Module.symvers  built-in.a  fs        lib      samples          usr
Kconfig         README          certs       include   mm       scripts          virt
inslab-test-server@computer:~/git/rpi_kernel/linux$ pwd
/home/inslab-test-server/git/rpi_kernel/linux
```

>> 타겟 시스템 커널 소스코드의 디렉터리로 이동 후 pwd 명령어를 통해 경로 출력

>> 해당 커널 소스코드는 크로스 컴파일이 완료되어 있어야 한다.

>> 앞선 *"라즈베리파이 커널 컴파일"* 실습을 진행하였다면 당연히 컴파일이 완료된 상태일 것이다.

>> 이제 타겟 시스템 커널 소스코드 디렉터리의 경로를 알아냈으니 Makefile의 KDIR 변수에 입력하면 된다.

```
obj-m = vrd_dd.o

KDIR := /home/inslab-test-server/git/rpi_kernel/linux
PWD := $(shell pwd)

default:
	make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -C $(KDIR) M=$(PWD) modules

clean:
	make -C $(KDIR) M=$(PWD) clean
```

2.3. 컴파일 진행

명령어 : make

```
inslab-test-server@computer:~/git/rpi_kernel/block-dev-driver$ ls
Makefile  vrd_dd.c
inslab-test-server@computer:~/git/rpi_kernel/block-dev-driver$ make
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -C /home/inslab-test-server/git/rpi_kernel/linux M
-server/git/rpi_kernel/block-dev-driver modules
make[1]: Entering directory '/home/inslab-test-server/git/rpi_kernel/linux'
  CC [M]  /home/inslab-test-server/git/rpi_kernel/block-dev-driver/vrd_dd.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/inslab-test-server/git/rpi_kernel/block-dev-driver/vrd_dd.mod.o
  LD [M]  /home/inslab-test-server/git/rpi_kernel/block-dev-driver/vrd_dd.ko
make[1]: Leaving directory '/home/inslab-test-server/git/rpi_kernel/linux'
inslab-test-server@computer:~/git/rpi_kernel/block-dev-driver$ ls
Makefile  Module.symvers  modules.order  vrd_dd.c  vrd_dd.ko  vrd_dd.mod.c  vrd_dd.mod.o  vrd_dd.o
```

>> block-dev-driver 디렉터리에서 Makefile의 내용을 따라 컴파일 하기 위해 make 명령어 실행

>> 해당 명령어를 실행하게 되면 vrd_dd.ko 파일이 생성된다.

>> 이 파일은 블록 디바이스 드라이버 모듈이다.

2.4. 모듈 정보 확인

명령어 : modinfo vrd_dd.ko

```
inslab-test-server@computer:~/git/rpi_kernel/block-dev-driver$ modinfo vrd_dd.ko
filename:       /home/inslab-test-server/git/rpi_kernel/block-dev-driver/vrd_dd.ko
license:        GPL
srcversion:     CB0C26C1ABC84C9D8988849
depends:
name:           vrd_dd
vermagic:       4.19.127-v7l+ SMP mod_unload modversions ARMv7 p2v8
```

>> 모듈 정보를 확인해보면 ARM 코어의 4.19.127 커널에서 동작할 수 있는 모듈 임을 확인할 수 있다.

```
inslab-test-server@computer:~/git/rpi_kernel/block-dev-driver$ uname -a
Linux computer 4.19.126-jihun #57 SMP Sat Feb 5 19:06:35 KST 2022 x86_64 x86_64 x86_64 GNU/Linux
```

>> 현재 호스트 시스템의 환경은 x86 코어의 4.19.126 커널이다.

>> 즉, vrd_dd.ko 모듈은 호스트 시스템과 맞지 않음으로 해당 모듈은 시스템 모듈에 등록할 수 없다.

```
inslab-test-server@computer:~/git/rpi_kernel/block-dev-driver$ sudo insmod vrd_dd.ko
insmod: ERROR: could not insert module vrd_dd.ko: Invalid module format
```

>> vrd_dd.ko를 호스트 시스템에 등록하기 위해 insmod 명령어를 사용하게 되면 에러를 반환하고 시스템에 등록하지 못하는 것을 확인할 수 있다.

3. 블록 디바이스 드라이버 모듈을 타겟 시스템에 등록

3.1. 블록 디바이스 드라이버 모듈 이동

명령어 : scp vrd_dd.ko pi@[RPI IP 주소]:/home/pi

```
insla-test-server@computer:~/git/rpi_kernel/block-dev-driver$ ls
Makefile  Module.symvers  modules.order  vrd_dd.c  vrd_dd.ko  vrd_dd.mod.c  vrd_dd.mod.o  vrd_dd.o
insla-test-server@computer:~/git/rpi_kernel/block-dev-driver$ scp vrd_dd.ko pi@192.168.1.14:/home/pi
pi@192.168.1.14's password:
vrd_dd.ko                                                                   100% 7812    794.0KB
```

>> scp 명령어를 통해 타겟 시스템으로 모듈을 이동

>> scp 명령어 참조는 앞선 실습 내용인 *"툴체인"*을 참조하면 된다.

3.2. 타겟 시스템으로 원격 접속

명령어 : ssh pi@[RPI IP 주소]

```
insla-test-server@computer:~/git/rpi_kernel/block-dev-driver$ ssh pi@192.168.1.14
pi@192.168.1.14's password:
Linux raspberrypi 4.19.127-v7l+ #2 SMP Mon Mar 7 14:06:40 KST 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Mar  7 06:32:02 2022
pi@raspberrypi:~ $
```

>> ssh 명령어를 통해 원격 접속이 성공하였다면 명령 프롬프트가 변경되었을 것이다.

```
pi@raspberrypi:~ $ ls
vrd_dd.ko
```

>> ls 명령어를 실행해보면 vrd_dd.ko가 업로드 되어있는 것을 확인할 수 있다.

```
pi@raspberrypi:~ $ modinfo vrd_dd.ko
filename:       /home/pi/vrd_dd.ko
license:        GPL
srcversion:     CB0C26C1ABC84C9D8988849
depends:
name:           vrd_dd
vermagic:       4.19.127-v7l+ SMP mod_unload modversions ARMv7 p2v8
pi@raspberrypi:~ $ uname -a
Linux raspberrypi 4.19.127-v7l+ #2 SMP Mon Mar 7 14:06:40 KST 2022 armv7l GNU/Linux
```

>> modinfo 명령어와 uname -a 명령어를 통해 확인해보면 모듈의 커널 버전과 코어가 현재 타겟 시스템의 커널 버전과 코어와 동일할 것이다.

3.3. 블록 디바이스 드라이버 모듈 등록

명령어 : sudo insmod vrd_dd.ko

```
pi@raspberrypi:~ $ ls
vrd_dd.ko
pi@raspberrypi:~ $ sudo insmod vrd_dd.ko
```

>> 해당 명령어를 실행한 뒤 아무런 내용이 뜨지 않는다면 정상 등록이 된 것이다.

```
pi@raspberrypi:~ $ lsmod
Module                  Size  Used by
vrd_dd                 16384  0
bnep                   20480  2
hci_uart               40960  1
btbcm                  16384  1 hci_uart
serdev                 20480  1 hci_uart
bluetooth             397312  24 hci_uart,b
ecdh_generic           28672  1 bluetooth
aes_arm_bs             20480  0
```

>> lsmod를 통해 타겟 시스템에 등록된 모듈들을 찾아보면 vrd_dd가 등록되어 있는 것을 확인할 수 있다.

타겟 시스템에서 자신의 블록 디바이스 드라이버 모듈 사용해보기

- 3.3 까지 타겟 시스템 커널에 블록 디바이스 드라이버 모듈을 등록하는 과정을 진행하였으며 4.부터는 해당 모듈을 사용하는 방법을 설명한다.

4. 커널에 적재된 블록 디바이스 드라이버 모듈 정보 확인

4.1. 블록 디바이스 드라이버 Major 정보 확인

명령어 : cat /proc/devices

```
Block devices:
  1 ramdisk
  7 loop
  8 sd
 65 sd
 66 sd
 67 sd
 68 sd
 69 sd
 70 sd
 71 sd
128 sd
129 sd
130 sd
131 sd
132 sd
133 sd
134 sd
135 sd
179 mmc
259 blkext
300 vrd
```

>> 현재 커널에 적재된 디바이스 드라이버들을 확인가능

>> 왼쪽에 보이는 번호가 디바이스 드라이버의 Major 번호이다.

>> 예제의 vrd는 Major가 300번임을 확인할 수 있다.

4.2. 블록 디바이스 드라이버 디스크 및 파티션 정보 확인

명령어 : sudo fdisk -l(소문자L) /dev/vrd

```
pi@raspberrypi:~ $ sudo fdisk -l /dev/vrd
Disk /dev/vrd: 4 MiB, 4194304 bytes, 8192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

>> 명령어의 결과를 통해 본 실습의 블록 디바이스 정보를 확인할 수 있다.

5. 블록 디바이스 파일에 파일 시스템 올리기 및 마운트 시키기

5.1. 블록 디바이스 파일에 ext2 파일 시스템 올리기

명령어 : sudo mkfs –t ext2 /dev/vrd

```
pi@raspberrypi:~ $ sudo mkfs -t ext2 /dev/vrd
mke2fs 1.44.5 (15-Dec-2018)
Creating filesystem with 4096 1k blocks and 1024 inodes

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

>> 본 실습을 통해 생성된 블록 디바이스 파일에 Linux의 ext2 파일 시스템을 올리는 명령어이다.

5.2. 블록 디바이스 파일을 디렉터리에 마운트 시키기

5.2.1. 마운트 시킬 디렉터리 생성

명령어 : sudo mkdir /mnt/testdir

```
pi@raspberrypi:/mnt/testdir $ ls
pi@raspberrypi:/mnt/testdir $
pi@raspberrypi:/mnt/testdir $
```

>> 생성된 디렉터리로 이동한 뒤 ls 명령어로 확인해보면 아직 마운트 하지 않았으니 아무런 파일이 없을 것이다.

```
pi@raspberrypi:/mnt/testdir $ cd ..
pi@raspberrypi:/mnt $
pi@raspberrypi:/mnt $
```

>> 마운트를 시켜야 함으로 부모 디렉터리(cd ..)로 이동 해주자.

5.2.2. 생성시킨 디렉터리를 마운트 시키기

명령어 : sudo mount /dev/vrd /mnt/testdir

```
pi@raspberrypi:/mnt $ sudo mount /dev/vrd /mnt/testdir
pi@raspberrypi:/mnt $ cd testdir/
pi@raspberrypi:/mnt/testdir $ ls
lost+found
```

>> 마운트가 완료된 디렉터리로 이동 후 ls 명령어를 실행해보면 lost+found라는 디렉터리가 생성된 것을 확인할 수 있다.

>> 앞서 5.2.1.의 내용과는 다른 결과로 본 실습에서 생성한 블록 디바이스 드라이버가 잘 작동하는 것을 확인할 수 있다.

5.2.3. 마운트가 정상적으로 되었는지 확인

명령어 : df –h

```
pi@raspberrypi:/mnt $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        59G  1.4G   55G   3% /
devtmpfs        1.8G     0  1.8G   0% /dev
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           2.0G   25M  1.9G   2% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           2.0G     0  2.0G   0% /sys/fs/cgroup
/dev/mmcblk0p1  253M   53M  200M  21% /boot
tmpfs           391M     0  391M   0% /run/user/1000
/dev/vrd        3.9M   30K  3.7M   1% /mnt/testdir
```

>> 정상적으로 /mnt/testdir에 /dev/vrd로 마운트 된 것을 확인할 수 있다.

>> 이제 /mnt/testdir은 /dev/vrd로 제어되며 이 /dev/vrd는 앞서 2.3.에서 생성한 블록 디바이스 드라이버 모듈(vrd_dd.ko)로 제어가 된다.

5.2.4. 마운트된 디렉터리에 새로운 파일 생성

명령어 : sudo vi hello

```
hello, world
~
```

>> 마운트된 디렉터리(/mnt/testdir)에 hello라는 파일을 생성한 뒤 파일의 내용으로 hello, world를 입력한 뒤 :wq로 저장해보자.

```
pi@raspberrypi:/mnt/testdir $ ls
hello  lost+found
```

>> 정상적으로 hello라는 파일이 생성된 것을 확인할 수 있다.

```
pi@raspberrypi:/mnt/testdir $ cat hello
hello, world
```

>> hello 파일의 내용도 정상적으로 저장된 것을 확인할 수 있다.

5.2.5. 마운트 및 블록 디바이스 드라이버 모듈 해제

명령어 : sudo umount /mnt/testdir

명령어 : sudo rmmod /dev/vrd

```
pi@raspberrypi:/mnt $ sudo umount /mnt/testdir
pi@raspberrypi:/mnt $ sudo rmmod vrd_dd
pi@raspberrypi:/mnt $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        59G  1.4G   55G   3% /
devtmpfs        1.8G     0  1.8G   0% /dev
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           2.0G   25M  1.9G   2% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           2.0G     0  2.0G   0% /sys/fs/cgroup
/dev/mmcblk0p1  253M   53M  200M  21% /boot
tmpfs           391M     0  391M   0% /run/user/1000
```

>> 위 명령어 2개를 차례대로 입력한 뒤 df –h 명령어를 실행해보면 앞선 5.2.3.과 다르게 /dev/vrd에 대한 내용이 빠진 것을 확인할 수 있으며 이는 정상적으로 마운트가 해제된 것을 의미한다.

5.2.6. 시스템 로그 확인

명령어 : sudo dmesg

```
[11530.325841] blk_device_driver init
[11530.332321] device open
[11530.334931] device release
[12341.826061] device open
[12341.830041] device release
[12487.995229] device open
[12487.995260] device release
[12487.995779] device open
[12487.997151] device release
[12487.997462] device open
[12487.997765] device release
[12487.998999] device open
[12487.999016] device release
[12487.999037] device open
[12487.999061] device release
[12487.999126] device open
[12487.999142] device release
[12487.999161] device open
[12487.999176] device release
[12487.999249] device open
[12487.999423] device release
[12487.999503] device open
[12488.003880] device release
[15097.418891] device open
[15097.418912] device release
[15097.418996] device open
[15097.419682] device release
[15122.101156] device open
[15122.102792] device release
[15122.102995] device open
[15122.103146] EXT4-fs (vrd): mounting ext2 file system using the ext4 subsystem
[15122.103956] EXT4-fs (vrd): mounted filesystem without journal. Opts: (null)
[15130.686509] device release
[15136.275931] device open
[15136.277939] device release
[15136.278191] device open
[15136.278355] EXT4-fs (vrd): mounting ext2 file system using the ext4 subsystem
[15136.279276] EXT4-fs (vrd): mounted filesystem without journal. Opts: (null)
[16095.555263] device release
[16121.563157] device open
[16121.564740] device release
[16121.564965] device open
[16121.565126] EXT4-fs (vrd): mounting ext2 file system using the ext4 subsystem
[16121.565879] EXT4-fs (vrd): mounted filesystem without journal. Opts: (null)
[16127.195694] device release
[16130.853251] blk_device_driver exit
```

>> 시스템 로그에서 블록 디바이스 드라이버를 통해 디바이스 파일(/dev/vrd)이 생성, 마운트, 파일 입출력, 마운트 해제, 모듈 해제 등등이 이루어진 것을 확인할 수 있다.

```
pi@raspberrypi:/mnt/testdir $ ls
pi@raspberrypi:/mnt/testdir $
pi@raspberrypi:/mnt/testdir $
pi@raspberrypi:/mnt/testdir $
```

>> 마운트가 해제 되면서 /mnt/testdir 디렉터리 내부 내용이 모두 지워진 것 또한 확인할 수 있다.