운영체제(SW)
LAB #3

과 목 명	운영체제(SW)
교 수	최 종 무
학 번	32163006 32164420
이 름	이 건 욱 조 정 민

Contents

프로젝트 분석

Ⅰ. EXT File System

EXT(second EXTended filesystem, 이차 확장 파일 시스템)는 리눅스 파일 시스템 중 하나이고 미닉스(Minix) 파일시스템을 개선하여 만들어진 파일시스템이다.
리눅스에서 일반 파일 API를 제공하기 위해 가상 파일 시스템 레이어(VFS)가 리눅스 커널에 추가되면서 만들어졌다.
EXT 파일 시스템은 EXT를 최초로 하고 EXT에 있었던 여러 단점(분리 접근, 아이노트 수정 등 지원 안 함)을 개선한 EXT2, EXT2에 journaling 개념을 추가한 EXT3, EXT3에 allocation delay와 block mapping을 대신한 extends 방식 등을 추가한 EXT4 순으로 등장하였다.

EXT

EXT2 : EXT + Block Group, Pre-Allocation

EXT2 : EXT2 + Journaling

EXT2 : EXT3 + Larger file system capacity, Extent-based mapping

이번 과제에서 우리는 EXT2에 대해서 집중적으로 분석해볼 것이다.

Ⅱ. EXT2 File System

1. EXT2의 데이터 구조

ext2 공간은 블록으로 나뉘어 있다. 이 블록은 블록 그룹으로 나뉘는데, 이는 유닉스 파일 시스템의 실린더 그룹과 비슷하다. 일반적으로 거대한 파일 시스템에 수천 개의 블록이 있다.

주어진 파일의 데이터는 가능한 한 하나의 블록 그룹 내에 포함되어 있다. 이것은 외부 단편화를 줄이고 연속된 대량의 데이터를 읽을 때 디스크 탐색을 최소화시킨다.

각각의 블록 그룹은 블록 그룹 서술자 테이블과 슈퍼블록의 복사본을 포함하고 있으며, 모든 블록 그룹은 블록 비트맵, 아이노드 비트맵, 아이노드 테이블을 포함하고 있고, 마지막으로 실제 데이터 블록을 포함한다. 슈퍼블록은 매우 중요한 운영 체제 부팅, 즉 파일 시스템 내 여러 블록 그룹에서 만들어진 백업한 복사본과 같은 중요한 정보를 포함한다.

그러나, 일반적으로 파일 시스템의 첫 번째 블록에서 찾을 수 있는 첫 번째 복사본만 부팅에 쓰인다.

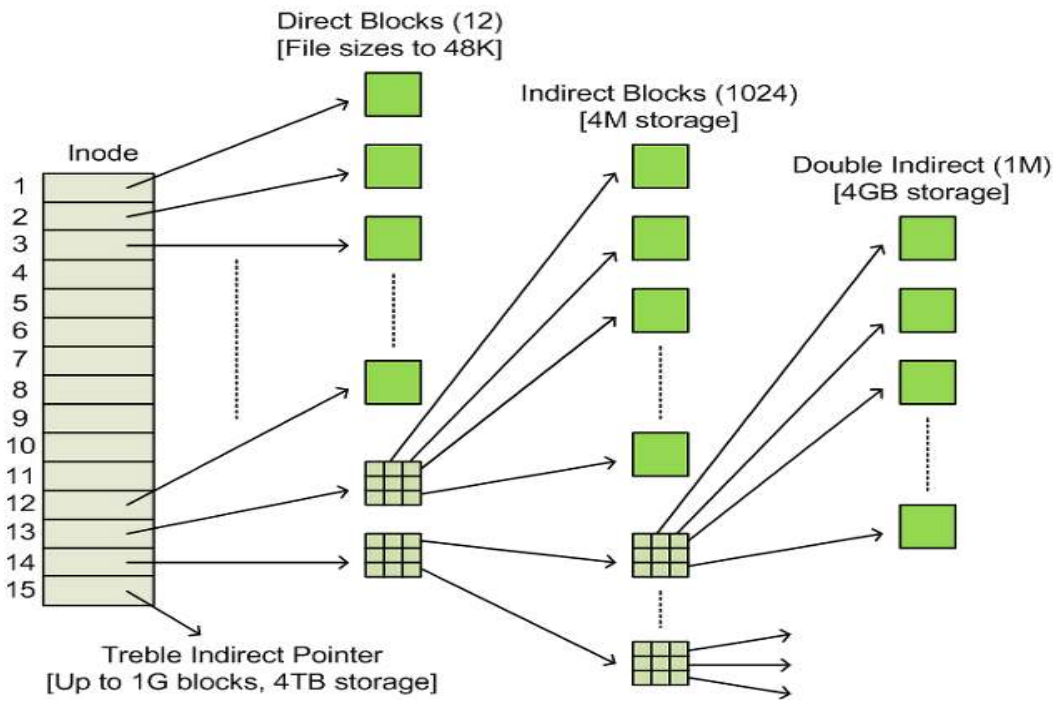
그룹 서술자는 블록 비트맵, 아이노드 비트맵과 모든 블록 그룹의 아이노드 테이블의 시작점의 위치에 저장되며 이들은 차례로 그룹 서술자 테이블에 저장된다.

2. Inode

모든 파일이나 디렉터리는 아이노드로 표현된다.

아이노드는 크기, 권한, 소유권, 그리고 파일이나 디렉터리의 디스크의 위치에 대한 데이터를 포함한다.

파일시스템의 모든 파일이나 디렉토리는 각기 하나의 Inode가 할당되어 있으며, 각 블록 그룹을 위한 Ext2 Inode는 어떤 Inode가 할당되었는지 아닌지를 확인하기 위한 Inode Bitmap과 함께 Inode Table에 저장되게 된다.



각각의 block들이 하나의 파일을 가르키게 되면 15개의 파일밖에 가르키지 못하기 때문에 inode에는 Direct, Single Indirect, Double Indirect, Triple Indirect block이 존재한다. (Direct Block의 Size는 10, 12, 15로 여러 종류가 있지만 많이 사용되는 크기인 12로 설명하겠다.)

- Direct Block Pointer: Direct라는 이름 그대로 직접 file을 가르키며 12개의 블록, 각 4KB로 구성되어 있어 총 12×4KB = 48KB를 가르킬 수 있다.

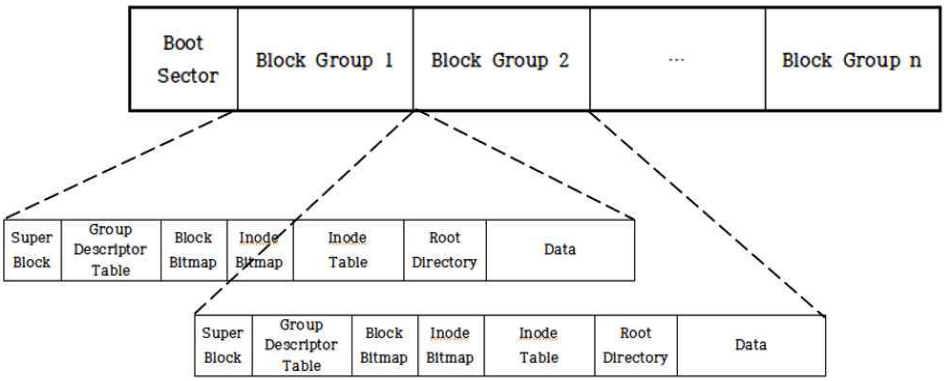
- Single Indirect Block Pointer : 파일이 Direct Block이 가르킬 수 있는 최대 크기인 48KB를 넘어간다면 추가적인 block pointer를 저장하기 위해 13번째 block에 48KB를 넘어간 크기의 data들을 가르키는 pointer를 저장한다.
  각각의 pointer의 크기는 4B이고 13번째 block의 크기는 4KB이므로 4KB/4B = 1024개의 pointer를 사용할 수 있다. 따라서 총 1×1024×4KB=4MB의 크기를 하나의 Single Indirect Block이 가르킬 수 있다.

- Double Indirect Block Pointer : 파일의 크기가 48KB+4MB보다 커지게 되면 filesystem은 추가적인 block pointer를 저장하기 위해 Double Indirect Block Pointer를 사용한다.
  방식은 Single과 비슷하지만 Double은 1024개의 data block을 가르키는 1024개의 indirect block을 갖게 된다. 따라서 1×1024×1024×4KB = 4GB의 크기를 가르킬 수 있다.

- Triple Indirect Block Pointer : 파일의 크기가 48KB+4MB+4GB보다 커지게 되면 위의 경우들과 마찬가지로 1024개의 data block을 가르키는 1024개의 indirect block을 가르키는 1024개의 indirect block을 갖게 된다.
  따라서 1×1024×1024×1024×4KB = 4TB의 크기를 가르킬 수 있다.

3. EXT2의 형태

EXT2 file system은 Boot Sector와 여러 개의 Block Group으로 구성되어 있다. 여기서 Boot Sector는 부팅을 위한 값들을 저장한다. Block Group은 FFS의 Cylinder Group과 굉장히 유사하다. Ext2 파일 시스템에서는 블록들을 여러 개의 그룹으로 나누어 파일시스템의 정보 및 데이터를 저장한다. 이때 OS 커널에서는 가능하면 파일에 속하는 데이터 블록은 같은 블록에 저장하려 하기 때문에 블록 그룹은 파일 단편화를 줄일 수 있다.



1) Super Block
- Ext2 파일시스템에서 사용되는 주요 설정 정보들이 기록되는 영역이며 첫 번재 블록에 위치한다.
- 파일시스템의 설정 파일들이 기록되어 있고, 부트 코드가 기록되어 있지는 않으며 슈퍼 블록의 사본은 모든 블록 그룹들의 첫 번째 블록에 저장된다.
- 슈퍼 블록에 저장되는 주요 데이터는 블록의 크기(1KB< 2KB, 4KB), 총 블록의 개수, 블록 그룹의 개수, Inode의 개수, 그룹 내의 블록/Inode의 개수이다.

2) Group Descriptor Table
- Super block의 다음 block브투 위치하며, 이 Table은 해당 파일시스템 내의 모든 블록 그룹에 대한 정보를 기록한다.
- 각각의 정보를 Group Descriptor라고 하며, Group Descriptor Table은 슈퍼 블록과 마찬가지로 모든 블록 그룹에 동일하게 중복 기록 되어 있다.
- Group Descriptor Table에 저장되는 주요 데이터는 다음과 가탇.
  v. Block Bitmap의 블록 번호
  v. Inode Bitmap의 블록 번호
  v. 첫 번째 Inode Table Block의 블록 번호
  v. 그룹 안에 있는 빈 블록 수
  v. 그룹 안에 있는 Inode 수
  v. 그룹 안에 있는 빈 Directory 수
- 블록 그룹의 개수와 크기를 계산하는 부분을 보고, 그룹 디스크립터 테이블 정보를 알게 되면 그룹의 크기와 그룹 내에 남은 용량 및 블록 그룹의 전체적인 레이아웃을 그릴 수 있다.

※ 슈퍼블록과 그룹 디스크립터 테이블은 매우 중요한 내용을 저장하고 있기 때문에 모든 블록 그룹들이 복사본을 가지고 있다. 그러나 항상 첫 번재 블록그룹의 정보만을 사용하며 여기에 오류가 발생할 경우, 다른 블록 그룹에서 그 정보를 가져온다.

3) Block Bitmap
- Group Descriptor Table 다음에 위치하며 테이블의 크기가 일정하지 않기 때문에 Block Bitmap의 위치도 상황에 따라 달라진다. 따라서 Group Descriptor의 정보에 Block Bitmap의 위치가 포함된다.

4) Inode
- Ext2 파일시스템에서 파일 객체가 저장되는 곳은 Inode 데이터 구조이며, 모든 Inode의 크기는 Super Block에 정의되어 있는 크기로 고정된다.
- 모든 파일과 디렉토리들은 각각 1개의 Inode를 할당하며, 모든 Inode들은 고유한 주소를 가진다.(Inode의 인덱스는 1부터 시작된다.)
- Inode는 Block Group 내의 Inode Table에 저장되고, 앞서 나온 것처럼 Inode Table의 위치는 Group Descriptor Table에 기록된다.

5) Inode Bitmap
- Inode Bitmap도 Block Bitmap과 마찬가지로 1개의 Inode는 1개의 Bit에 대응되어 해당 블록 그룹이 관리하는 모든 Inode의 현황을 나타낼 수 있다.
- Inode Bitmap 역시 하나의 블록안에 기록되어야 하며, 기능적으로 Block Bitmap과 거의 동일하지만, 블록에 대한 정보가 Inode에 대한 정보로 바뀔 뿐이다.
- Inode Bitmap은 Inode가 사용되고 있는 현황을 나타내며, 이는 Inode를 할당 또는 해제할 때보다 용이하게 쓰여진다.

6) Inode Table
- Inode Table은 인접한 연속된 블록으로 이루어져 있으며, 각 블록은 미리 정의된 Inode개수를 포함한다.
- Inode Table 내 첫 번째 블록의 번호를 Group Descriptor Table에 저장하며, 모든 Inode는 크기가 128Byte로 동일하다.

7) Root Directory
- 이름 그대로 Root와 관련된 정보가 담겨있고 Block을 찾아갈 때는 Root Directory 부터 시작되어 Block을 찾아간다. Root Directory의 Inode 번호는 2번이다.

8) Data
- 파일 혹은 디렉토리가 저장되는 공간이다.
- 모든 디렉토리의 경우 현재 디렉토리와, 상위 디렉토리를 의미하는 '.', '..' 엔트리로 시작하며, 그 아래로 하위 디렉토리나 여러 파일들이 위치한다.

프로젝트 수행 및 결과

Ⅰ. Lab3 사전준비

이번 과제의 목표는 EXT2 file system을 이용하여 우리의 학번 끝 세자리에 해당하는 파일을 찾아가는 것이다.
1) 파일은 총 세 개의 블록으로 구성되어진다.
2) 여기서 한 블록을 추가하여 총 네 개의 블록을 찾는다.
ex) 나의 학번이 32164420이라면 4번 디렉터리 안에 있는 20번 파일을 찾는다.

1. 'sudo su'를 사용하여 root 권한으로 변경 -> make -> ramdisk.ko 파일 확인
2. insmod ramdisk.ko -> lsmod | grep ramdisk (모듈 적재 후 확인)
3. mkdir mnt(mnt 디렉토리 생성)

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ls
append.c  create.sh  Makefile  os_ext2  ramdisk.c
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# sudo su
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# make
make -C /lib/modules/5.4.0-31-generic/build M=/home/kunuk/2020_DKU_OS/lab3_filesystem modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-31-generic'
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/ramdisk.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/ramdisk.mod.o
  LD [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/ramdisk.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-31-generic'
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ls
append.c   Makefile       Module.symvers  ramdisk.c   ramdisk.mod    ramdisk.mod.o
create.sh  modules.order  os_ext2        ramdisk.ko  ramdisk.mod.c  ramdisk.o
```

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# insmod ramdisk.ko
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# lsmod | grep ramdisk
ramdisk                16384  0
```

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# mkdir mnt
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ls
append.c   Makefile  modules.order   os_ext2    ramdisk.ko   ramdisk.mod.c  ramdisk.o
create.sh  mnt       Module.symvers  ramdisk.c  ramdisk.mod  ramdisk.mod.o
```

```
jeongmin@jeongmin-VirtualBox:~$ ls
2020_DKU_OS  examples.desktop  lab2_sync
jeongmin@jeongmin-VirtualBox:~$ cd 2020_DKU_OS
jeongmin@jeongmin-VirtualBox:~/2020_DKU_OS$ ls
README.md  lab1_sched  lab2_sync  lab3_filesystem
jeongmin@jeongmin-VirtualBox:~/2020_DKU_OS$ cd lab3_filesystem
jeongmin@jeongmin-VirtualBox:~/2020_DKU_OS/lab3_filesystem$ sudo su
[sudo] password for jeongmin:
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# make
make -C /lib/modules/5.3.0-28-generic/build M=/home/jeongmin/2020_DKU_OS/lab3_filesystem modules
make[1]: Entering directory '/usr/src/linux-headers-5.3.0-28-generic'
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/ramdisk.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/jeongmin/2020_DKU_OS/lab3_filesystem/ramdisk.mod.o
  LD [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/ramdisk.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.3.0-28-generic'
```

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# ls
Makefile        append.c   modules.order  ramdisk.c   ramdisk.mod    ramdisk.mod.o
Module.symvers  create.sh  os_ext2        ramdisk.ko  ramdisk.mod.c  ramdisk.o
```

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# insmod ramdisk.ko
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# lsmod | grep ramdisk
ramdisk                16384  0
```

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# mkdir mnt
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# ls
Makefile        append.c   mnt            os_ext2    ramdisk.ko   ramdisk.mod.c  ramdisk.o
Module.symvers  create.sh  modules.order  ramdisk.c  ramdisk.mod  ramdisk.mod.o
```

+) - lsmod : 이미 실행중인 모듈들의 리스트를 보여준다.
- insmod : 커널에 모듈을 삽입한다.
- (filename.ko) : Linux의 중심 구성요소 인 Kernel에 의해 사용되는 모듈 파일
- grep : grep은 사용자가 명령의 수행결과에서 불필요한 정보를 제거할 수 있도록 한다. grep을 필터로 사용하려면, 반드시 명령의 수행 결과를 grep을 통해 파이프(|) 해야 한다.

4. 파일시스템 생성(mkfs) 후 mnt에 마운트

- mkfs.ext /dev/ramdisk -> mount /dev/ramdi나 ./mnt

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# mkfs.ext2 /dev/ramdisk
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 131072 4k blocks and 32768 inodes
Filesystem UUID: 176895b2-94c6-46af-b6b3-0b9c19b927b3
Superblock backups stored on blocks:
        32768, 98304

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

root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# mount /dev/ramdisk ./mnt
```

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# mkfs.ext2 /dev/ramdisk
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 131072 4k blocks and 32768 inodes
Filesystem UUID: 141f7313-db21-41cf-8c1e-6da21e370a8c
Superblock backups stored on blocks:
        32768, 98304

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

root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# mount /dev/ramdisk ./mnt
```

+) - mkfs.ext2 : ext2형식의 file system을 생성한다.
   - mount : 보조기억장치(HDD, FDD, CD-ROM 등)나 파일 시스템이 다른 디스크를 /의 하위 디렉토리로 연결하여 사용가능하게 해주는 명령어

5. df –h를 이용하여 확인

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            967M     0  967M   0% /dev
tmpfs           199M  1.4M  198M   1% /run
/dev/sda5        29G  7.6G   20G  28% /
tmpfs           994M     0  994M   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           994M     0  994M   0% /sys/fs/cgroup
/dev/loop0       94M   94M     0 100% /snap/core/9066
/dev/loop1       55M   55M     0 100% /snap/core18/1705
/dev/loop2       55M   55M     0 100% /snap/core18/1754
/dev/loop3      241M  241M     0 100% /snap/gnome-3-34-1804/24
/dev/loop4      256M  256M     0 100% /snap/gnome-3-34-1804/33
/dev/loop5       63M   63M     0 100% /snap/gtk-common-themes/1506
/dev/loop6       50M   50M     0 100% /snap/snap-store/433
/dev/loop7       50M   50M     0 100% /snap/snap-store/454
/dev/loop8       28M   28M     0 100% /snap/snapd/7264
/dev/loop9      4.3M  4.3M     0 100% /snap/tree/18
/dev/sda1       511M  4.0K  511M   1% /boot/efi
tmpfs           199M   28K  199M   1% /run/user/1000
/dev/loop10      31M   31M     0 100% /snap/snapd/7777
/dev/loop11      98M   98M     0 100% /snap/core/9289
/dev/sr0         58M   58M     0 100% /media/kunuk/VBox_GAs_6.1.8
/dev/ramdisk    504M  396K  478M   1% /home/kunuk/2020_DKU_OS/lab3_filesystem/mnt
```

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            467M     0  467M   0% /dev
tmpfs            99M  1.4M   97M   2% /run
/dev/sda1       9.8G  6.7G  2.7G  72% /
tmpfs           491M     0  491M   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           491M     0  491M   0% /sys/fs/cgroup
/dev/loop0       94M   94M     0 100% /snap/core/9066
/dev/loop2      2.3M  2.3M     0 100% /snap/gnome-system-monitor/145
/dev/loop1      1.0M  1.0M     0 100% /snap/gnome-logs/100
/dev/loop3      1.0M  1.0M     0 100% /snap/gnome-logs/93
/dev/loop4       63M   63M     0 100% /snap/gtk-common-themes/1506
/dev/loop5      161M  161M     0 100% /snap/gnome-3-28-1804/116
/dev/loop6      2.5M  2.5M     0 100% /snap/gnome-calculator/730
/dev/loop7       55M   55M     0 100% /snap/core18/1705
/dev/loop8       94M   94M     0 100% /snap/core/8935
/dev/loop9      2.5M  2.5M     0 100% /snap/gnome-calculator/748
/dev/loop10      55M   55M     0 100% /snap/gtk-common-themes/1502
/dev/loop11     384K  384K     0 100% /snap/gnome-characters/539
/dev/loop12     3.8M  3.8M     0 100% /snap/gnome-system-monitor/135
/dev/loop13      55M   55M     0 100% /snap/core18/1754
/dev/loop14      15M   15M     0 100% /snap/gnome-characters/495
/dev/loop15     256M  256M     0 100% /snap/gnome-3-34-1804/33
/dev/loop16     243M  243M     0 100% /snap/gnome-3-34-1804/27
tmpfs            99M   32K   99M   1% /run/user/1000
/dev/sr0         74M   74M     0 100% /media/jeongmin/VBox_GAs_6.0.16
tmpfs            99M     0   99M   0% /run/user/0
/dev/ramdisk    504M  396K  478M   1% /home/jeongmin/2020_DKU_OS/lab3_filesystem/mnt
```

+) - df : 시스템에 Mount된 하드디스크의 남은 용량을 확인할 때 사용하는 명령어다. 기본적으로 1,024Byte 블록 단위로 출력하며, 옵션을 통해 다른 단위로 출력이 가능하다.
-h : 사람이 읽을 수 있는 형태의 크기로 출력한다.
-a : 0 블록의 파일 시스템을 포함하여, 모든 파일시스템을 출력
-i : inode의 남은 공간, 사용 공간, 사용 퍼센트를 출력
-t : 보여주는 목록을 파일 시스템의 타입으로 제한
-T : 파일 시스템의 형태를 추가하여 각각의 파티션 정보를 출력

6. ./ctreate.sh 실행 후 mnt 디렉토리에 0~9번 디렉토리 생성되었는지 확인

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ./create.sh
create files ...
done
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ls mnt
0  1  2  3  4  5  6  7  8  9  lost+found
```

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# ./create.sh
create files ...
done
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# ls mnt
0  1  2  3  4  5  6  7  8  9  lost+found
```

+) - (filename).sh : shell script 확장자 파일. 실행하기 위해서는
(chmod +x (filename).sh)명령어를 실행하여 권한 변경 후 실행

7. 각 디렉터리 안에는 파일이 0~99번까지 생성되어 있다.

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ls mnt/0
0   13  18  22  27  31  36  40  45  5   54  59  63  68  72  77  81  86  90  95
1   14  19  23  28  32  37  41  46  50  55  6   64  69  73  78  82  87  91  96
10  15  2   24  29  33  38  42  47  51  56  60  65  7   74  79  83  88  92  97
11  16  20  25  3   34  39  43  48  52  57  61  66  70  75  8   84  89  93  98
12  17  21  26  30  35  4   44  49  53  58  62  67  71  76  80  85  9   94  99
```

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# ls mnt/0
0   13  18  22  27  31  36  40  45  5   54  59  63  68  72  77  81  86  90  95
1   14  19  23  28  32  37  41  46  50  55  6   64  69  73  78  82  87  91  96
10  15  2   24  29  33  38  42  47  51  56  60  65  7   74  79  83  88  92  97
11  16  20  25  3   34  39  43  48  52  57  61  66  70  75  8   84  89  93  98
12  17  21  26  30  35  4   44  49  53  58  62  67  71  76  80  85  9   94  99
```

8. 내가 찾아야할 파일에 한 블록 추가(3개 블록 -> 4개 블록)

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ls -l mnt/0/6
-rw-r--r-- 1 root root 8198  6월  4 17:29 mnt/0/6
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ./apd mnt/0/6 13 0/6-13
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ls -l mnt/0/6
-rw-r--r-- 1 root root 49159  6월  4 17:31 mnt/0/6
```

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# ls -l mnt/4/20
-rw-r--r-- 1 root root 8199  6월  4 16:12 mnt/4/20
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# ./apd mnt/4/20 13 4/20-13
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# ls -l mnt/4/20
-rw-r--r-- 1 root root 49160  6월  4 16:13 mnt/4/20
```

-> mnt/4/20 파일크기 49160 확인
(32163006 - 이건욱은 숫자가 한 개 적어(0/6) 1Byte가 작게 확인되었습니다.
-->8198 / 49159)

Ⅱ. Lab3 분석 및 결과

1. EXT2 Hex Data 읽기

1) EXT2 의 정보는 Little Endian으로 저장되어 있고, 일반 데이터는 Big Endian 으로 저장되어 있다.
2) 따라서, EXT2의 Metadata를 읽을 때는 보여지는 데이터를 Bite별로 반대로 읽어야 한다.

[출저] : 2020 OS LAB3

2. Super Block 영역 분석

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x400 /dev/ramdisk
00000400: 00800000 00000200 99190000 8ff70100  ................
00000410: f57f0000 00000000 02000000 02000000  ................
00000420: 00800000 00800000 00200000 7eb0d85e  ......... ..~..^
00000430: 7eb0d85e 0100ffff 53ef0000 01000000  ~..^....S.......
00000440: 72b0d85e 00000000 00000000 01000000  r..^............
00000450: 00000000 0b000000 00010000 38000000  ............8...
00000460: 02000000 03000000 176895b2 94c646af  .........h....F.
00000470: b6b30b9c 19b927b3 00000000 00000000  ......'.........
00000480: 00000000 00000000 2f686f6d 652f6b75  ......../home/ku
00000490: 6e756b2f 32303230 5f444b55 5f4f532f  nuk/2020_DKU_OS/
000004a0: 6c616233 5f66696c 65737973 74656d2f  lab3_filesystem/
000004b0: 6d6e7400 00000000 00000000 00000000  mnt.............
000004c0: 00000000 00000000 00000000 00001f00  ................
000004d0: 00000000 00000000 00000000 00000000  ................
000004e0: 00000000 00000000 00000000 e9800c64  ...............d
000004f0: a53c4164 b3c95f16 1d704e8f 01000000  .<Ad.._..pN.....

root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x400 /dev/ramdisk
00000400: 00800000 00000200 99190000 8ff70100  ................
00000410: f57f0000 00000000 02000000 02000000  ................
00000420: 00800000 00800000 00200000 c7a3dc5e  ......... .....^
00000430: c7a3dc5e 0100ffff 53ef0000 01000000  ...^....S.......
00000440: c1a3dc5e 00000000 00000000 01000000  ...^............
00000450: 00000000 0b000000 00010000 38000000  ............8...
00000460: 02000000 03000000 813f366c d0134f07  .........?6l..O.
00000470: 8715edb6 52e0c31d 00000000 00000000  ....R...........
00000480: 00000000 00000000 2f686f6d 652f6a65  ......../home/je
00000490: 6f6e676d 696e2f32 3032305f 444b555f  ongmin/2020_DKU_
000004a0: 4f532f6c 6162335f 66696c65 73797374  OS/lab3_filesyst
000004b0: 656d2f6d 6e740000 00000000 00000000  em/mnt..........
000004c0: 00000000 00000000 00000000 00001f00  ................
000004d0: 00000000 00000000 00000000 00000000  ................
000004e0: 00000000 00000000 00000000 b4c2464b  ..............FK
000004f0: 6cc9469a 8f3c119e 0e590614 01000000  l.F..<...Y......
```

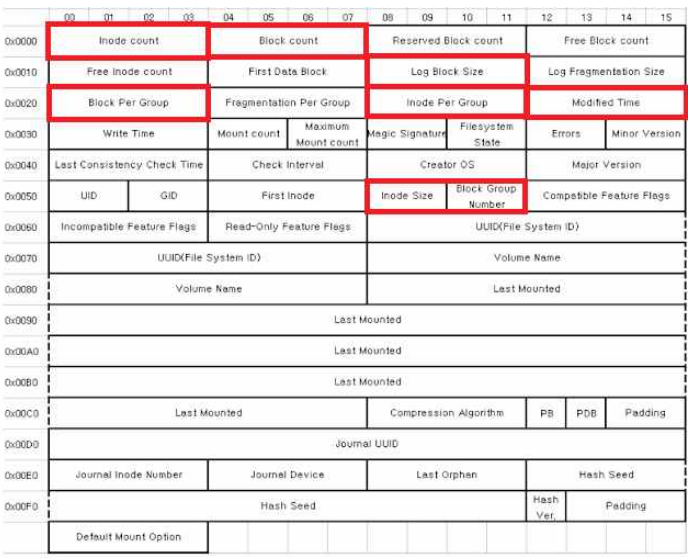
※ xxd

xxd는 주어진 파일 또는 표준입력을 16진수로 보여준다.

- g (byte) : byte단위로 출력하며 공백으로 끊어서 나타낸다. 그룹화를 하지 않기 위해선 –g 0 옵션을 주면 된다. default value는 일반모드에서 2, 비트모드에서 1로 설정된다.
- l (len) : len만큼 데이터를 읽은 뒤 중지한다.
- s (+/-)(offset) : offset은 말 그대로 현재 파일의 offset을 나타내며 +로 지정해 줄 시 앞에서부터 –로 지정해 줄 시 뒤에서부터 offset을 조정한다.

[출저] : https://linux.die.net/man/1/xxd

Name	Bytes	Desription
Inode Count	4	Total number of inodes in file system
Block Count	4	Total number of blocks in file system
Blocks Reserved	4	Reserved block count to prevent overfill
Free Block Count	4	Number of unallocated blocks
Free Inode Count	4	Number of unallocated inodes
Group 0	4	Block where first block group starts
Block Size	4	Left shifts of 1024 to obtain block size
Fragment Size	4	Left shifts of 1024 to obtain fragment size
Blocks per Block Grp.	4	Blocks in a typical block group
Fragments per Block Grp.	4	Fragment count in a typical block group
Inodes per Block Grp.	4	Inodes in a typical block group
Last Modified Time	4	Seconds rom epoch to last modified time
Last Written Time	4	Seconds from epoch to last write time
Mount Information	4	Total and max mounts of file system
Signature	2	0xEF53
File System State	2	Clean, error, recovering orphan inodes
Error Handling Method	2	Continue, remount as read only, or panic
Minor Version	2	Original or dynamic
Consistency Check	8	Last performed, interval
Creator OS	4	Linux, FreeBSD, etc.
Major Version	4	Original or dynamic
Reserved Block UID/GID	4	UID/GID tht can use reserved blocks
First Inode	4	First non-reserved inode in file system
Inode Size	2	Size of inode in bytes
Block Grp. Loc. of Copy	2	If backup copy, group of copy
Feature Flags	12	Features of the file system
File System ID	16	UUID of file system
Volume Name	16	OS's name for the volume
Other Misc. Information	72	Misc.
Journal Information	24	UUID, metadata inode, device
Orphan Inodes	4	Head of orphan inode list
Unused	788	Unused bytes



- Inode count : 0x8000
- Block count : 0x20000
- Log Block Size : 0x2
- Block Per Group : 0x8000
- Inode Per Group : 0x2000
- Modified Time : 0xe59d8ab6
- Inode Size : 0x100
- Block Group Number : 0X0

3. Group Descriptor Table 영역 분석

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x1000 /dev/ramdisk
00001000: 21000000 22000000 23000000 d47dc61e  !..."...#....}..
00001010: 05000400 00000000 00000000 00000000  ................
00001020: 21800000 22800000 23800000 2278361f  !..."...#..."x6.
00001030: 02000400 00000000 00000000 00000000  ................
00001040: 00000100 01000100 02000100 fb77361f  .............w6.
00001050: 02000400 00000000 00000000 00000000  ................
00001060: 21800100 22800100 23800100 da7dd11e  !..."...#....}..
00001070: 03000400 00000000 00000000 00000000  ................
00001080: 00000000 00000000 00000000 00000000  ................
00001090: 00000000 00000000 00000000 00000000  ................
000010a0: 00000000 00000000 00000000 00000000  ................
000010b0: 00000000 00000000 00000000 00000000  ................
000010c0: 00000000 00000000 00000000 00000000  ................
000010d0: 00000000 00000000 00000000 00000000  ................
000010e0: 00000000 00000000 00000000 00000000  ................
000010f0: 00000000 00000000 00000000 00000000  ................
```

- Group 0
  v. block bitmap : 0x21 블록부터 시작
  v. Inode bitmap : 0x22 블록부터 시작
  v. Inode table : 0x23 블록부터 시작
  v. 단위는 블록(4KB)임을 감안해야 함.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	block bitmap				Inode bitmap				Inode table				free blk cont		frree ino cnt	
10	used dir cnt		padding		reserved(padding)											

inode가 속한 그룹은 (inode number-1)/block per group 번 block group이므로 Ext2에서 Root inode number는 2번이며 super block영역을 분석했을 때 inodes per group은 **0x2000**이었다.

즉, 0번 Blcok Group의 Inode Table의 1번째에 위치한다.

4. Inode Table 영역 분석(0x23부터 시작)

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x1000 -s 0x23000 /dev/ramdisk
00023000: 00000000 00000000 a947db5e a947db5e  .........G.^.G.^
00023010: a947db5e 00000000 00000000 00000000  .G.^............
00023020: 00000000 00000000 00000000 00000000  ................
00023030: 00000000 00000000 00000000 00000000  ................
00023040: 00000000 00000000 00000000 00000000  ................
00023050: 00000000 00000000 00000000 00000000  ................
00023060: 00000000 00000000 00000000 00000000  ................
00023070: 00000000 00000000 00000000 00000000  ................
00023080: 00000000 00000000 00000000 00000000  ................
00023090: 00000000 00000000 00000000 00000000  ................
000230a0: 00000000 00000000 00000000 00000000  ................
000230b0: 00000000 00000000 00000000 00000000  ................
000230c0: 00000000 00000000 00000000 00000000  ................
000230d0: 00000000 00000000 00000000 00000000  ................
000230e0: 00000000 00000000 00000000 00000000  ................
000230f0: 00000000 00000000 00000000 00000000  ................
00023100: ed410000 00100000 b247db5e bc47db5e  .A.......G.^.G.^
00023110: bc47db5e 00000000 00000d00 08000000  .G.^............
00023120: 00000000 0a000000 23020000 00000000  ........#.......
00023130: 00000000 00000000 00000000 00000000  ................
00023140: 00000000 00000000 00000000 00000000  ................
00023150: 00000000 00000000 00000000 00000000  ................
00023160: 00000000 00000000 00000000 00000000  ................
00023170: 00000000 00000000 00000000 00000000  ................
00023180: 20000000 9049f378 9049f378 0c8ce177   ....I.x.I.x...w
00023190: a947db5e 00000000 00000000 00000000  .G.^............
000231a0: 00000000 00000000 00000000 00000000  ................
000231b0: 00000000 00000000 00000000 00000000  ................
000231c0: 00000000 00000000 00000000 00000000  ................
000231d0: 00000000 00000000 00000000 00000000  ................
000231e0: 00000000 00000000 00000000 00000000  ................
000231f0: 00000000 00000000 00000000 00000000  ................
```

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x1000 -s 0x23000 /dev/ramdisk
00023000: 00000000 00000000 c1a3dc5e c1a3dc5e  ...........^...^
00023010: c1a3dc5e 00000000 00000000 00000000  ...^............
00023020: 00000000 00000000 00000000 00000000  ................
00023030: 00000000 00000000 00000000 00000000  ................
00023040: 00000000 00000000 00000000 00000000  ................
00023050: 00000000 00000000 00000000 00000000  ................
00023060: 00000000 00000000 00000000 00000000  ................
00023070: 00000000 00000000 00000000 00000000  ................
00023080: 00000000 00000000 00000000 00000000  ................
00023090: 00000000 00000000 00000000 00000000  ................
000230a0: 00000000 00000000 00000000 00000000  ................
000230b0: 00000000 00000000 00000000 00000000  ................
000230c0: 00000000 00000000 00000000 00000000  ................
000230d0: 00000000 00000000 00000000 00000000  ................
000230e0: 00000000 00000000 00000000 00000000  ................
000230f0: 00000000 00000000 00000000 00000000  ................
00023100: ed410000 00100000 c7a3dc5e daa3dc5e  .A.........^...^
00023110: daa3dc5e 00000000 00000d00 08000000  ...^............
00023120: 00000000 0a000000 23020000 00000000  ........#.......
00023130: 00000000 00000000 00000000 00000000  ................
00023140: 00000000 00000000 00000000 00000000  ................
00023150: 00000000 00000000 00000000 00000000  ................
00023160: 00000000 00000000 00000000 00000000  ................
00023170: 00000000 00000000 00000000 00000000  ................
00023180: 20000000 404bbf21 404bbf21 f4d86674   ...@K.!@K.!..ft
00023190: c1a3dc5e 00000000 00000000 00000000  ...^............
000231a0: 00000000 00000000 00000000 00000000  ................
000231b0: 00000000 00000000 00000000 00000000  ................
000231c0: 00000000 00000000 00000000 00000000  ................
000231d0: 00000000 00000000 00000000 00000000  ................
000231e0: 00000000 00000000 00000000 00000000  ................
000231f0: 00000000 00000000 00000000 00000000  ................
```

> inode의 크기는 0x100byte이며 Root가 속한 Block Group은 0번이고 Index는 1이다.
즉, 0번 Block Group의 Inode Table의 0x100부터 Root Inode이다.
그리고, block pointer 0 : **0x223**이다.

5. Data 영역 분석

[32164420 조정민]

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x1000 -s 0x223000 /dev/ramdisk
00223000: 02000000 0c000102 2e000000 02000000  ................
00223010: 0c000202 2e2e0000 0b000000 14000a02  ................
00223020: 6c6f7374 2b666f75 6e640000 01400000  lost+found...@..
00223030: 0c000102 30000000 01600000 0c000102  ....0....`......
00223040: 31000000 01200000 0c000102 32000000  1.... ......2...
00223050: 0c000000 0c000102 33000000 71000000  ........3...q...
00223060: 0c000102 34000000 66200000 0c000102  ....4...f ......
00223070: 35000000 66600000 0c000102 36000000  5...f`......6...
00223080: 66400000 0c000102 37000000 cb600000  f@......7....`..
00223090: 0c000102 38000000 d6000000 680f0102  ....8.......h...
002230a0: 39000000 00000000 00000000 00000000  9...............
```

Inode number : 0x71

- ▶ Block Group : (71-1) / 2000 = 0 Block Group
- ▶ Inode Table Index : (71-1) % 2000 = 70
- ▶ 4번 디렉터리의 Inode는 0번 Block Group의 Inode Table에서 70번째에 위치함을 확인
- ▶ 230 + 70 = 2a000

[32163006 이건욱]

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x223000 /dev/ramdisk
00223000: 02000000 0c000102 2e000000 02000000  ................
00223010: 0c000202 2e2e0000 0b000000 14000a02  ................
00223020: 6c6f7374 2b666f75 6e640000 01400000  lost+found...@..
00223030: 0c000102 30000000 01600000 0c000102  ....0....`......
00223040: 31000000 01200000 0c000102 32000000  1.... ......2...
00223050: 0c000000 0c000102 33000000 71000000  ........3...q...
00223060: 0c000102 34000000 66200000 0c000102  ....4...f ......
00223070: 35000000 66600000 0c000102 36000000  5...f`......6...
00223080: 66400000 0c000102 37000000 cb600000  f@......7....`..
00223090: 0c000102 38000000 d6000000 680f0102  ....8.......h...
002230a0: 39000000 00000000 00000000 00000000  9...............
```

Inode number : 0x4001

- ▶ Block Group : (4001-1) / 2000 = 2 Blcok Group
- ▶ Inode Table Index : (4001-1) % 2000 = 0
- ▶ 0번 디렉터리의 Inode는 2번 Block Group의 Inode Table에서 0번째에 위치함을 확인

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x1000 /dev/ramdisk
00001000: 21000000 22000000 23000000 d47dc61e  !..."...#....}..
00001010: 05000400 00000000 00000000 00000000  ................
00001020: 21800000 22800000 23800000 2278361f  !..."...#..."x6.
00001030: 02000400 00000000 00000000 00000000  ................
00001040: 00000100 01000100 02000000 fb77361f  .............w6.
00001050: 02000400 00000000 00000000 00000000  ................
00001060: 21800100 22800100 23800100 da7dd11e  !..."...#....}..
00001070: 03000400 00000000 00000000 00000000  ................
00001080: 00000000 00000000 00000000 00000000  ................
00001090: 00000000 00000000 00000000 00000000  ................
000010a0: 00000000 00000000 00000000 00000000  ................
000010b0: 00000000 00000000 00000000 00000000  ................
000010c0: 00000000 00000000 00000000 00000000  ................
000010d0: 00000000 00000000 00000000 00000000  ................
000010e0: 00000000 00000000 00000000 00000000  ................
000010f0: 00000000 00000000 00000000 00000000  ................
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x10002000 /dev/ramdisk
10002000: ed410000 00100000 69f8dd5e 69f8dd5e  .A......i..^i..^
10002010: 69f8dd5e 00000000 00000200 08000000  i..^............
10002020: 00000000 65000000 02020100 00000000  ....e...........
10002030: 00000000 00000000 00000000 00000000  ................
10002040: 00000000 00000000 00000000 00000000  ................
10002050: 00000000 00000000 00000000 00000000  ................
10002060: 00000000 a5c5f88b 00000000 00000000  ................
10002070: 00000000 00000000 00000000 00000000  ................
10002080: 20000000 b86f2377 b86f2377 500a4774   ....o#w.o#wP.Gt
10002090: 69f8dd5e 500a4774 00000000 00000000  i..^P.Gt........
```

- ▶ 2번째 Block Group의 Inode Table을 찾는다.

6. 나의 학번 찾아가기

[32164420 조정민]

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x200 -s 0x2a000 /dev/ramdisk
0002a000: ed410000 00100000 daa3dc5e daa3dc5e  .A.........^...^
0002a010: daa3dc5e 00000000 00000200 08000000  ...^............
0002a020: 00000000 65000000 2a020000 00000000  ....e...*.......
0002a030: 00000000 00000000 00000000 00000000  ................
0002a040: 00000000 00000000 00000000 00000000  ................
0002a050: 00000000 00000000 00000000 00000000  ................
0002a060: 00000000 3720faf3 00000000 00000000  ....7 ..........
0002a070: 00000000 00000000 00000000 00000000  ................
0002a080: 20000000 04ee2819 04ee2819 c0aa3418   .....(...(...4.
0002a090: daa3dc5e c0aa3418 00000000 00000000  ...^..4.........
```

▶ 3행 3열은 Block pointer이다.
▶ 해당 위치를 Little Endian으로 풀면 '0x22a'이다.

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x200 -s 0x22a000 /dev/ramdisk
0022a000: 71000000 0c000102 2e000000 02000000  q...............
0022a010: 0c000202 2e2e0000 72000000 0c000101  ........r.......
0022a020: 30000000 73000000 0c000101 31000000  0...s.......1...
0022a030: 74000000 0c000101 32000000 75000000  t.......2...u...
0022a040: 0c000101 33000000 76000000 0c000101  ....3...v.......
0022a050: 34000000 77000000 0c000101 35000000  4...w.......5...
0022a060: 78000000 0c000101 36000000 79000000  x.......6...y...
0022a070: 0c000101 37000000 7a000000 0c000101  ....7...z.......
0022a080: 38000000 7b000000 0c000101 39000000  8...{.......9...
0022a090: 7c000000 0c000201 31300000 7d000000  |.......10..}...
0022a0a0: 0c000201 31310000 7e000000 0c000201  ....11..~.......
0022a0b0: 31320000 7f000000 0c000201 31330000  12..........13..
0022a0c0: 80000000 0c000201 31340000 81000000  ........14......
0022a0d0: 0c000201 31350000 82000000 0c000201  ....15..........
0022a0e0: 31360000 83000000 0c000201 31370000  16..........17..
0022a0f0: 84000000 0c000201 31380000 85000000  ........18......
0022a100: 0c000201 31390000 86000000 0c000201  ....19..........
0022a110: 32300000 87000000 0c000201 32310000  20..........21..
```

▶ 내가 찾아야할 학번의 뒷자리는 32164420 -> 20이다.
▶ 해당 위치는 86000000이다.
▶ Inode Table Index : (86-1)%2000 = 85
▶ 230 + 85 = 2b5

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x200 -s 0x2b500 /dev/ramdisk
0002b500: a4810000 08c00000 daa3dc5e f8a3dc5e  ...........^...^
0002b510: f8a3dc5e 00000000 00000100 28000000  ...^........(...
0002b520: 00000000 01000000 a4050100 1c090100  ................
0002b530: 80090100 00000000 00000000 00000000  ................
0002b540: 00000000 00000000 00000000 00000000  ................
0002b550: 00000000 00000000 04020100 00000000  ................
0002b560: 00000000 a5a3c4fe 00000000 00000000  ................
0002b570: 00000000 00000000 00000000 00000000  ................
0002b580: 20000000 14039734 14039734 04ee2819   ......4...4..(.
0002b590: daa3dc5e 04ee2819 00000000 00000000  ...^..(.........
0002b5a0: 00000000 00000000 00000000 00000000  ................
0002b5b0: 00000000 00000000 00000000 00000000  ................
0002b5c0: 00000000 00000000 00000000 00000000  ................
```

▶ 3행 3, 4열, 4행 1열, 6행 3열은 Block pointer이다.
▶ 해당 위치를 Little Endian으로 풀면 '0x0105a4', '0x01091c', '0x010980', '0x010204'이다.

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x200 -s 0x105a4
000 /dev/ramdisk
105a4000: 342f3230 2d310a00 00000000 00000000  4/20-1..........
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x200 -s 0x1091c000 /dev/ramdisk
1091c000: 342f3230 2d320a00 00000000 00000000  4/20-2..........
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x200 -s 0x10980000 /dev/ramdisk
10980000: 342f3230 2d330a00 00000000 00000000  4/20-3..........
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x200 -s 0x10204000 /dev/ramdisk
10204000: b8890000 00000000 00000000 00000000  ................
```

※ 4/20-13 블록은 Single Indirect Block(13)이므로 해당 block pointer ('b8890000' -> '0x89b8')를 이용해 찾아간다.

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x200 -s 0x89b8000 /dev/ramdisk
089b8000: 342f3230 2d31330a 00000000 00000000  4/20-13.........
```

▶ 4/20-1, 4/20-2, 4/20-3, 4/20-13

[32163006 이건욱]

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x10002000 /dev/ramdisk
10002000: ed410000 00100000 69f8dd5e 69f8dd5e  .A......i..^i..^
10002010: 69f8dd5e 00000000 00000200 08000000  i..^............
10002020: 00000000 65000000 02020100 00000000  ....e...........
10002030: 00000000 00000000 00000000 00000000  ................
10002040: 00000000 00000000 00000000 00000000  ................
10002050: 00000000 00000000 00000000 00000000  ................
10002060: 00000000 a5c5f88b 00000000 00000000  ................
10002070: 00000000 00000000 00000000 00000000  ................
10002080: 20000000 b86f2377 b86f2377 500a4774   ....o#w.o#wP.Gt
10002090: 69f8dd5e 500a4774 00000000 00000000  i..^P.Gt........
```

▶ 3행 3열은 Block pointer이다.
▶ 해당 위치를 Little Endian으로 풀면 '0x010202'이다.

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x010202000 /dev/ramdisk
10202000: 01400000 0c000102 2e000000 02000000  .@..............
10202010: 0c000202 2e2e0000 02400000 0c000101  .........@......
10202020: 30000000 03400000 0c000101 31000000  0....@......1...
10202030: 04400000 0c000101 32000000 05400000  .@......2....@..
10202040: 0c000101 33000000 06400000 0c000101  ....3....@......
10202050: 34000000 07400000 0c000101 35000000  4....@......5...
10202060: 08400000 0c000101 36000000 09400000  .@......6....@..
10202070: 0c000101 37000000 0a400000 0c000101  ....7....@......
10202080: 38000000 0b400000 0c000101 39000000  8....@......9...
10202090: 0c400000 0c000201 31300000 0d400000  .@......10...@..
102020a0: 0c000201 31310000 0e400000 0c000201  ....11...@......
102020b0: 31320000 0f400000 0c000201 31330000  12...@......13..
102020c0: 10400000 0c000201 31340000 11400000  .@......14...@..
102020d0: 0c000201 31350000 12400000 0c000201  ....15...@......
102020e0: 31360000 13400000 0c000201 31370000  16...@......17..
102020f0: 14400000 0c000201 31380000 15400000  .@......18...@..
```

▶ 내가 찾아야할 학번의 뒷자리는 32163006 -> 6이다.
▶ 해당 위치는 0x4008이다.
▶ Inode Table Index : (4008 - 1) %2000 = 2...7
▶ 100020 + 7 = 100027

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x010002700 /dev/ramdisk
10002700: a4810000 07c00000 69f8dd5e 8af8dd5e  ........i..^...^
10002710: 8af8dd5e 00000000 00000100 28000000  ...^........(...
10002720: 00000000 01000000 06040100 ee850000  ................
10002730: 52060100 00000000 00000000 00000000  R...............
10002740: 00000000 00000000 00000000 00000000  ................
10002750: 00000000 00000000 04020100 00000000  ................
10002760: 00000000 9e458b87 00000000 00000000  .....E..........
10002770: 00000000 00000000 00000000 00000000  ................
10002780: 20000000 58a1c313 58a1c313 202c3b75   ...X...X... ,;u
10002790: 69f8dd5e 202c3b75 00000000 00000000  i..^ ,;u........
```

▶ 3행 3, 4열, 4행 1열, 6행 3열은 Block pointer이다.
▶ 해당 위치를 Little Endian으로 풀면 '0x010406', '0x85ee', '0x010652', '0x010204' 이다.

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x010406000 /dev/ramdisk
10406000: 302f362d 310a0000 00000000 00000000  0/6-1...........
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x85ee000 /dev/ramdisk
085ee000: 302f362d 320a0000 00000000 00000000  0/6-2...........
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x010652000 /dev/ramdisk
10652000: 302f362d 330a0000 00000000 00000000  0/6-3...........
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x010204000 /dev/ramdisk
10204000: b8890000 00000000 00000000 00000000  ................
```

※ 4/20-13 블록은 Single Indirect Block(13)이므로 해당 block pointer ('b8890000' -> '0x89b8')를 이용해 찾아간다.

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# xxd -g 4 -l 0x100 -s 0x89b8000 /dev/ramdisk
089b8000: 302f362d 31330a00 00000000 00000000  0/6-13..........
```

▶ 0/6-1, 0/6-2, 0/6-3, 0/6-13

Ⅲ. Bonus 수행결과

1. 이전 실습에서 mnt를 정리

- umount /dev/radmi나
- rmmod ramdisk
- lsmod | grep ramdisk (확인)

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# umount /dev/ramdisk
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# rmmod ramdisk
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# lsmod | grep ramdisk
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem#
```

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# umount /dev/ramdisk
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# rmmod ramdisk
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# lsmod | grep ramdisk
```

2. 모듈 적재

- insmod ramdisk.ko
- lsmod | grep ramdisk(모듈 적재 확인)

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# insmod ramdisk.ko
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# lsmod | grep ramdisk
ramdisk                16384  0
```

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# insmod ramdisk.ko
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# lsmod | grep ramdisk
ramdisk                16384  0
```

3. os_ext2 디렉터리로 이동 후 파일 확인

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# cd os_ext2
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2# ls
Kconfig   acl.h     ext2.h    inode.c  super.c    xattr.c           xattr_trusted.c
Makefile  balloc.c  file.c    ioctl.c  symlink.c  xattr.h           xattr_user.c
acl.c     dir.c     ialloc.c  namei.c  tags       xattr_security.c
```

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# cd os_ext2
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2# ls
acl.c     dir.c   ialloc.c  Kconfig         namei.c    tags      xattr_security.c
acl.h     ext2.h  inode.c   Makefile        super.c    xattr.c   xattr_trusted.c
balloc.c  file.c  ioctl.c   Module.symvers  symlink.c  xattr.h   xattr_user.c
```

4. 소스 수정

- /os_ext2/super.c

```
static struct dentry *ext2_mount(struct file_system_type *fs_type,
        int flags, const char *dev_name, void *data)
{
        printk(KERN_ERR "os_ext2 : Cho Jeongmin OS Lab3"); //os_ext2 : Your Name OS Lab3
        return mount_bdev(fs_type, flags, dev_name, data, ext2_fill_super);
}
```

```
static struct dentry* ext2_mount(struct file_system_type* fs_type,
        int flags, const char* dev_name, void* data)
{
        printk(KERN_ERR "os_ext2 : Lee Kunuk OS Lab3"); //os_ext2 : Your Name OS Lab3
        return mount_bdev(fs_type, flags, dev_name, data, ext2_fill_super);
}
```

5. 소스 수정 후 make

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2# make
make -C /lib/modules/5.3.0-28-generic/build M=/home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2
modules
make[1]: Entering directory '/usr/src/linux-headers-5.3.0-28-generic'
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/balloc.o
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/dir.o
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/file.o
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/ialloc.o
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/inode.o
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/ioctl.o
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/namei.o
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/super.o
  CC [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/symlink.o
  LD [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/os_ext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/os_ext2.mod.o
  LD [M]  /home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2/os_ext2.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.3.0-28-generic'
```

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2# make
make -C /lib/modules/5.4.0-31-generic/build M=/home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2 modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-31-generic'
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/balloc.o
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/dir.o
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/file.o
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/ialloc.o
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/inode.o
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/ioctl.o
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/namei.o
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/super.o
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/symlink.o
  LD [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/os_ext2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/os_ext2.mod.o
  LD [M]  /home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2/os_ext2.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-31-generic'
```

6. 모듈 적재

- insmod os_ext.ko
- lsmod | grep ox_ext2(모듈 적재 확인)
- 이전 디렉터리로 이동

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2# insmod os_ext2.ko
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2# lsmod | grep os_ext2
os_ext2                73728  0
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem/os_ext2# cd ..
```

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2# insmod os_ext2.ko
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2# lsmod | grep os_ext2
os_ext2                73728  0
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem/os_ext2# cd ..
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# ls
apd       create.sh  mnt           Module.symvers  ramdisk.c   ramdisk.mod    ramdisk.mod.o
append.c  Makefile   modules.order  os_ext2         ramdisk.ko  ramdisk.mod.c  ramdisk.o
```

7. ext2로 포맷 후 os_ext2로 마운트

- mkfs.ext2 /dev/ramdisk
- mount –t os_ext2 /dev/ramdisk ./mnt

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# mkfs.ext2 /dev/ramdisk
mke2fs 1.44.1 (24-Mar-2018)
/dev/ramdisk contains a ext2 file system
        created on Thu Jun  4 17:11:16 2020
Proceed anyway? (y,N) y
/dev/ramdisk is mounted; will not make a filesystem here!
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# mount -t os_ext2 /dev/ramdisk ./mnt
```

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# mkfs.ext2 /dev/ramdisk
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 131072 4k blocks and 32768 inodes
Filesystem UUID: 481b6735-fc3c-46cf-9918-78229b1d6551
Superblock backups stored on blocks:
        32768, 98304

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

root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# mount -t os_ext2 /dev/ramdisk ./mnt
```

8. dmesg | grep os_ext2

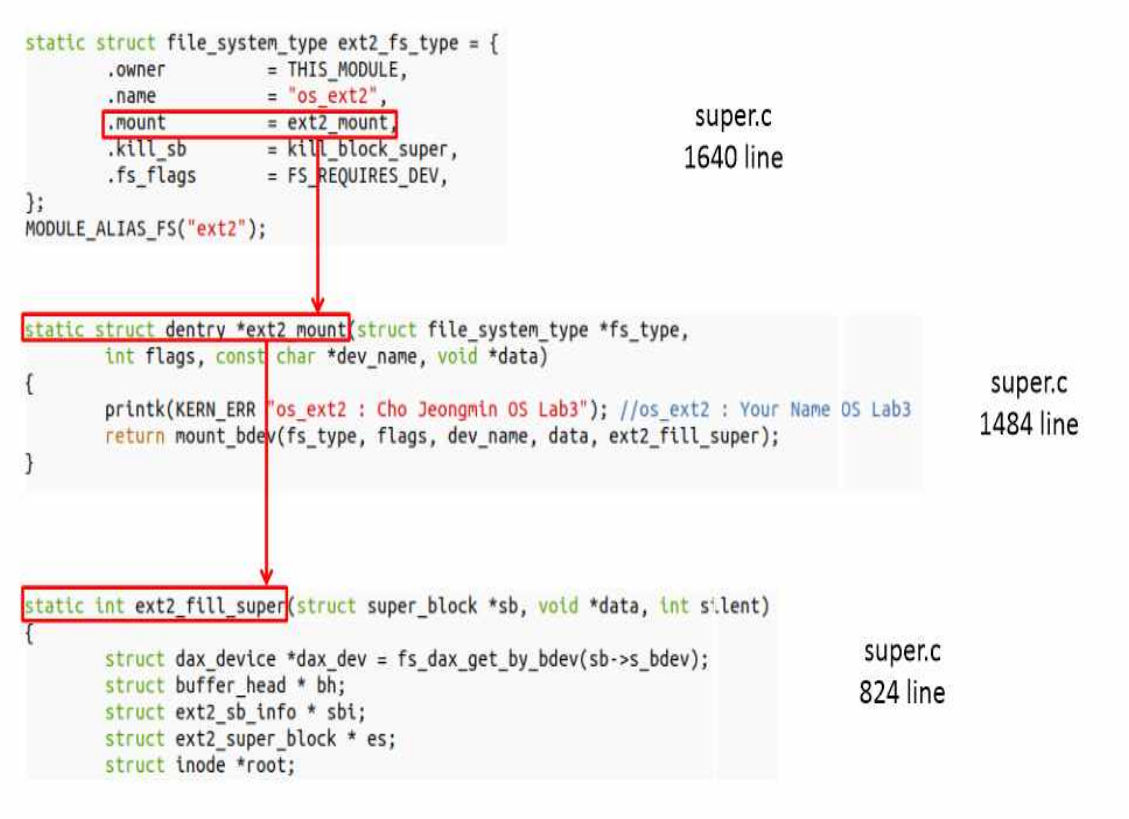
- os_ext2 : 이름 OS Lab3 문구 출력 확인

```
root@jeongmin-VirtualBox:/home/jeongmin/2020_DKU_OS/lab3_filesystem# dmesg | grep os_ext2
[  627.841784] os_ext2 : Cho Jeongmin OS Lab3
```

```
root@K-VirtualBox:/home/kunuk/2020_DKU_OS/lab3_filesystem# dmesg | grep os_ext2
[  874.696589] os_ext2 : Lee Kunuk OS Lab3
```

※ Super Block 초기화

-> filesystem이 mount 되는 시점에 수행



※ super.c의 ext2_fill_super 함수 분석(824~1226 line)

> ext2_fill_super 함수에서는 다음과 같은 네 가지의 역할을 수행한다.

1) super block 초기화
2) super block 읽기
3) ext2_sb_info 초기화
4) root 생성

```
set_opt(opts.s_mount_opt, RESERVATION);

if (!parse_options((char *) data, sb, &opts))
        goto failed_mount;

sbi->s_mount_opt = opts.s_mount_opt;
sbi->s_resuid = opts.s_resuid;
sbi->s_resgid = opts.s_resgid;

sb->s_flags = (sb->s_flags & ~SB_POSIXACL) |
        ((EXT2_SB(sb)->s_mount_opt & EXT2_MOUNT_POSIX_ACL) ?
         SB_POSIXACL : 0);
sb->s_iflags |= SB_I_CGROUPWB;
```

super.c
930 line
(super block 초기화)

```
if (blocksize != BLOCK_SIZE) {
        logic_sb_block = (sb_block*BLOCK_SIZE) / blocksize;
        offset = (sb_block*BLOCK_SIZE) % blocksize;
} else {
        logic_sb_block = sb_block;
}

if (!(bh = sb_bread(sb, logic_sb_block))) {
        ext2_msg(sb, KERN_ERR, "error: unable to read superblock");
        goto failed_sbi;
}
```

super.c
878 line
(super block 읽기

```
sbi->s_frag_size = EXT2_MIN_FRAG_SIZE <<
                   le32_to_cpu(es->s_log_frag_size);
if (sbi->s_frag_size == 0)
        goto cantfind_ext2;
sbi->s_frags_per_block = sb->s_blocksize / sbi->s_frag_size;

sbi->s_blocks_per_group = le32_to_cpu(es->s_blocks_per_group);
sbi->s_frags_per_group = le32_to_cpu(es->s_frags_per_group);
sbi->s_inodes_per_group = le32_to_cpu(es->s_inodes_per_group);

sbi->s_inodes_per_block = sb->s_blocksize / EXT2_INODE_SIZE(sb);
```

super.c
1023 line
(ext2_sb_info 초기화)

```
root = ext2_iget(sb, EXT2_ROOT_INO);
if (IS_ERR(root)) {
        ret = PTR_ERR(root);
        goto failed_mount3;
}
if (!S_ISDIR(root->i_mode) || !root->i_blocks || !root->i_size) {
        iput(root);
        ext2_msg(sb, KERN_ERR, "error: corrupt root inode, run e2fsck");
        goto failed_mount3;
}

sb->s_root = d_make_root(root);
```

super.c
1175 line
(root 생성)

Discussion

[32163006 이건욱]

- 과제를 시작하면서

처음 과제를 접했을 때는 정확하게 요구하는 것이 무엇인지 몰라서 LAB3 ppt 를 몇 번이고 읽어보면서 이해하고자 노력했다. 솔직히 처음에 과제를 받았을 땐 다른 과제에 비해 수월할 것이라고 생각하고 ppt 를 보았지만 막상 보니 막막했다. 하지만 계속해서 볼수록 설명이 매우 자세하게 되어있어 전반적인 내용을 파악할 수 있었다. 앞선 과제들 보다는 비교적 수월하게 수행했지만 그럼에도 많은 어려움이 있었다.

- 환경 설정에 있어서 어려움

ppt 에 나와 있는 대로 Ramdisk 를 생성하고 Mount 를 하는데 있어서 많은 명령어들과 옵션들을 제대로 활용하지 못하고 우분투의 설정 값 때문에 용량이 부족하여 다시 설치하는 등 처음에 많은 시행착오를 겪으면서 구글링도 하고 사용법도 익히면서 리눅스 명령어에 대해서 많이 알게 되었다. (ex/ dmesg, insmod, rmmod, lsmod 등) 이를 통해서 어떤 과정을 통해서 파일 시스템이 Mount 되는지를 이해하게 되었다.

- 내 학번 찾기 과제를 수행하는데 있어서의 어려움

사실 난 처음부터 이 과제를 수행하면서 당황하였다. 처음에 ./apd mnt/0/6 13 0/6 –13 명령어로 한 블록을 추가해 주었지만 블록의 사이즈가 49160 이 아닌 49159 였기 때문이다. 하지만 여기까지는 내가 남들에 비해 한자리가 적어 한 Byte 가 적다고 생각하고 넘어갔다. 하지만 Super Block 을 분석하고 Group Descriptor Table 을 분석한 뒤 Inode Table 을 분석하고 Data 영역을 분석할 때 문제가 발생했다. 내가 속한 Block Group 을 찾기 위해 inode number : 0x4001 에서 1 개의 index 을 빼주고 그 값을 inodes per block : 0x2000 으로 나누어 주는데 몫이 2 나머지가 0 이 나왔다. 그래서 나의 학번이 속한 Block Group 이 2 번 Block Group 인 것까지는 파악했지만 다시 Group Descriptor Table 에서 Block Group 2 번의 영역을 찾는 것에서 한동안 막혔다. 하지만 난 계속 이번과제는 ppt 를 자세히 보고 또 보며 수행했기 때문에 다시 ppt 로 돌아가

보았다. 그러던 중 18 페이지에 있는 표를 발견하게 되었고 그 표로 인해 한 개의 Block Group 당 우리가 출력하는 xxd 에서 차지하는 범위를 알게 되었다. 그래서 2 번 Block Group 에 해당하는 영역을 찾을 수 있었다. 이번 과제는 전체적으로 쉬웠다고 생각한다. 하지만 막상 자세하게 들여다보면 매우 자세하고 중요하고 어려운 내용이라고 생각한다. 나의 번호를 찾아가는 게임을 하는 느낌이라 과제를 수행하는 거에 있어서는 재미를 많이 느꼈지만 나는 내 자신이 이 과제의 전반적인 내용에 대해서는 제대로 알고 있다고 생각하지 않기 때문에 지금까지는 이 과제를 수행하기에만 급급했다면 앞으로는 file system 에 대해 더 깊고 자세하게 들여다보고 싶어졌다.

- Bonus(이름 출력)과제를 수행하는데 있어서의 어려움

EXT2 에 대한 이론적인 내용들만 파악한 채로 그것들을 실제로 찾아보고 실행해보는 것은 상당히 어려웠다. 하지만 그런 와중에 또 그 많은 파일들 중에서 내 이름을 출력하는 코드를 작성할 파일을 찾는 것은 너무 막막했다. 당연히 그 많은 파일들을 모두 읽어보며 코드를 작성할 곳은 찾는 것은 무리라고 생각해서 lsmod | grep 의 기능을 파악해보기로 하였다. 찾아본 결과 lsmod | grep 은 Kernel 의 정보를 출력해준다는 것을 알게 되었고 바로 Super Block 을 떠올리게 되었다. 그래서 Super Block 의 파일로 파악되는 super.c 파일의 코드를 보던 와중 내 이름을 작성할 곳을 찾게 되었다. 솔직히 내가 이름을 작성하는 곳에 있던 주석이 없었다면 더 못 찾을 수도 있었을 것이라고 생각하지만 이번 과제는 충분한 설명이 있었기에 훨씬 수월하게 이번 과제를 수행할 수 있었던 것 같다.

[32164420 조정민]

Ⅰ. 본 실습 프로젝트를 진행하면서

이번 프로젝트는 Ext2 File System 을 이용하여 나의 학번 끝 세 자리에 해당하는 파일을 찾아가는 것이 목표였다. 확실히 이전의 프로젝트들과 비교했을 때, 난이도는 낮았지만 이해가 가지 않는 부분이 상당 수 있어서 조교님의 file system PDF 를 참조하여 과제를 진행하였다.

PDF 에는 Ext2 File System 에 관한 설명들이 나와있었고 프로젝트를 진행하기 앞서 관련 내용들을 보며 Ext2 File System 에 대한 전반적인 이해가 필요했다. Ext2 File System Layout(Super Block / Group Descriptor Table / Bitmap / Inode Table / Data)의 내용들을 숙지한 후 프로젝트를 진행하니 큰 어려움은 없이 프로젝트를 진행할 수 있었다.

프로젝트를 진행하기 위해서 우선 모듈 적재를 해야했다. PDF 에 나와있는 내용을 토대로 모듈을 적재하고 확인하기 위해 필요한 리눅스 명령어들 이용하여 차질 없이 모듈 적재를 완료하였다. 추가적으로 파일시스템을 생성하고 mnt Directory 에 마우트하는 과정까지 순조로웠다. 프로젝트를 한 번에 수행하지 못하고 우분투를 다시 실행하는 경우가 여러 번 있었는데, 이런 경우에 다시 모듈을 적재하고 마운트하는 과정을 거쳐야했다. 마운트가 잘 되었는지 'df –h'명령어(+옵션)을 통해 체크할 수 있었다.

Ext2 의 정보가 Little Endian 으로 저장되어 있으므로 Metadata 를 읽을 때 데이터를 바이트별로 반대로 읽어야 한다는 사실을 PDF 를 보고 다시 확인할 수 있었다. Super Block 영역을 분석하면서 나는 Inode 와 Block 에 관한 정보를 얻을 수 있었다. 내 학번에 해당하는 파일을 찾아가기 위한 정보들이었다. Inode Table 에서 Block pointer 를 찾았고 해당 Block pointer 를 찾아가면 Block Group 과 Inode Table Index 를 알 수 있었다. 그렇게 계속 Block pointer 를 찾아가보니 내 학번에 해당하는 파일들을 찾을 수 있었다. 하지만 새로 추가한 네 번째 블록을 찾아가는 데 다른 블록들과는 다르게 내용이 바로 확인되지 않아 당황했다. 알고보니 새로 추가한 네 번째 블록은 Single Indirect Block 이었고 다시 Block pointer 를 이용하니 네 번째까지 찾을 수 있었다.

이번 프로젝트를 진행하면서 보물찾기와 같다는 생각을 하였다. 내가 원하는 파일을 찾기 위해 주어진 정보들을 이용하여 하나 하나씩 찾아가는 과정이 흥미로웠다. Ext2 File system 뿐만 아니라 다른 File system 을 이용해 이런

프로젝트를 더 진행해보고 싶다는 생각도 들었다. 아직 완벽히 File system 을 이해하지는 못했지만 File system 에 대해서 더 많이 알아보고 싶은 계기가 되었다.

Ⅱ. Bonus 실습 프로젝트를 진행하면서

해당 프로젝트를 수행하기 위해서 os_ext2 Directory 의 어떤 c 파일의 코드를 수정해야하는 지 찾기 힘들었다. PDF 에서는 Super Block 에 대한 이야기가 있어 super.c 파일을 열어 확인하였는데 PDF 에서 나오는 여러 함수들이 super.c 의 함수들과 일치하였다. 수정해야할 C 파일을 찾기는 했으나 어떤 함수의 어떤 부분을 수정해야할지 막막했다. PDF 의 함수와 일치하는 super.c 의 함수들을 확인해보니 우연치않게 우리가 얻어야하는 결과 문구가 주석으로 달려있는 부분을 찾을 수 있었다. 해당 출력 문구에 나의 이름을 추가한 후 make 를 하였고, 모듈적재 후 마운트를 하고 실행해보니 원하는 결과 문구가 출력된 것을 확인할 수 있었다. 보너스 실습 프로젝트를 진행하면서 솔직히 내가 어떤 부분을 왜 수정하였는지 정확히 파악하지는 못했지만 그런 부분을 보완하기 위해서 더 공부를 해야한다는 것을 몸소 깨달았다.

세 번째 프로젝트를 모두 진행하면서 File system 에 관한 나의 부족한 지식을 좀 더 채울 수 있었고, 아직 이해하지 못한 부분은 앞으로 남은 시간을 활용해 더 학습해야겠다는 생각을 가졌다. 이론으로만 배웠던 File system 을 실습을 통해 접하니 더욱 흥미가 생겼다. 한 학기동안 세 번의 프로젝트를 진행하면서 좀 더 발전한 나를 볼 수 있었고, OS 분야에 새로운 관심이 생겼다. 앞으로 OS 분야에 대한 개인적인 공부를 더 할 수 있었으면 좋겠다는 생각이 든다.