Operating System
Lab 3

단국대학교
Dankook University

이름 : 박동학, 홍승기
학번 : 32151648, 32155068
단국대학교 소프트웨어학과

목차

I. 분석

1. 프로젝트 분석

본 프로젝트는 FAT 파일 시스템에 대한 분석을 실시하는데 목적이 있다.

프로젝트 요구사항

1. Create Ramdisk

2. Make FAT file system on Ramdisk

3. mount the FAT file system

4. run the script on the mount directory

5. find clusters that are used by two files assigned for each team

6. After step 3. create a file and explore FAT before and after the creation (with a long file name if possible)

본 프로젝트를 수행하기에 앞서 아래와 같은 내용을 알고 있어야한다.

1. File System

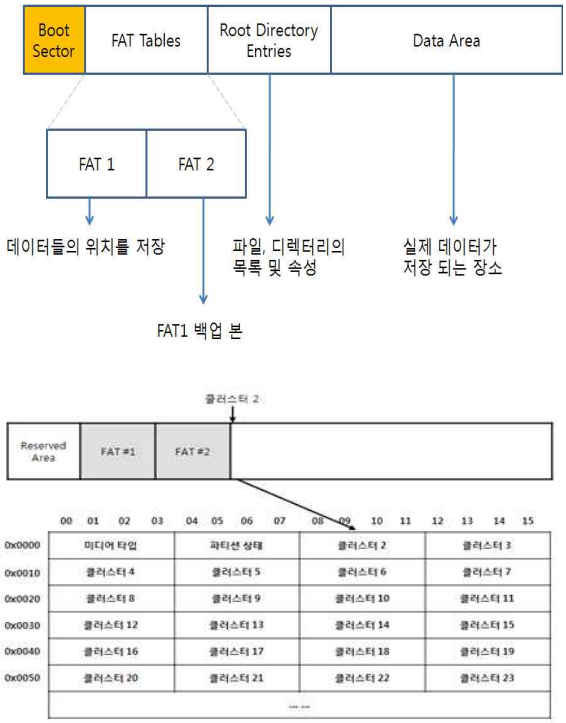
=> 파일시스템은 파일을 관리해 주는 시스템으로서 기본적으로는 Data Block, inode, Bitmap, Superblock 등의 레이아웃을 가지며 여러 인터페이스를 통해서 이를 생성, 삭제, 수정 할 수 있다.

2. FAT file System

=> 작은 용량을 가진 저장장치에 사용하기에는 inode 같은 Metadata 가 자치하는 비율이 너무 높아 비효율적이기 때문에 등장한 File system 이다. 기본적인 아이디어는 기본적인 파일시스템과는 다르게 Boot block(Boot

Sector + FSinfo), FAT(File Allocation Table), Directory 로 구성되어 파일을 관리한 것이다.

3. FAT File System 구조



FAT은 위의 그림과 같이 아래 4가지 영역으로 나누어진다.

1) Reserved Area : 이는 Boot Sector와 FSINFO와 여러 정보를 담고있는 예약된 영역이다. Boot Sector에는 부팅을 위한 여러 가지 값을 저장하고 있으며 크기는 1 Sector이다.

2) FAT : FAT Area는 Cluster를 관리하는 테이블이 모여 있는 영역이다. 파일들이 어떤 식으로 연결부분과 끝을 알 수 있는 부분이기 때문에 중요성 때문에 복사본으로 1개를 더 가지고 있는다.

3)Root Directory : Root Directory에 대한 정보를 가지고 있는다

4) Data : Data가 쓰여지는 영역이다.

II. 프로젝트

1. 프로젝트 결과 분석

1) FAT 분석

```
00000000: eb 58 90 6d 6b 66 73 2e 66 61 74 00 02 08 20 00  .X.mkfs.fat... .
00000010: 02 00 00 00 00 f8 00 00 10 00 04 00 00 00 00 00  ................
00000020: 00 00 20 00 fc 07 00 00 00 00 00 00 02 00 00 00  .. ............
00000030: 01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000040: 80 01 29 9a c2 9c 2c 4e 4f 20 4e 41 4d 45 20 20  ..)...,NO NAME
00000050: 20 20 46 41 54 33 32 20 20 20 0e 1f be 77 7c ac    FAT32   ...w|.
00000060: 22 c0 74 0b 56 b4 0e bb 07 00 cd 10 5e eb f0 32  ".t.V.......^..2
00000070: e4 cd 16 cd 19 eb fe 54 68 69 73 20 69 73 20 6e  .......This is n
00000080: 6f 74 20 61 20 62 6f 6f 74 61 62 6c 65 20 64 69  ot a bootable di
00000090: 73 6b 2e 20 20 50 6c 65 61 73 65 20 69 6e 73 65  sk.  Please inse
000000a0: 72 74 20 61 20 62 6f 6f 74 61 62 6c 65 20 66 6c  rt a bootable fl
000000b0: 6f 70 70 79 20 61 6e 64 0d 0a 70 72 65 73 73 20  oppy and..press
000000c0: 61 6e 79 20 6b 65 79 20 74 6f 20 74 72 79 20 61  any key to try a
000000d0: 67 61 69 6e 20 2e 2e 2e 20 0d 0a 00 00 00 00 00  gain ... .......
000000e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

위의 그림은 FAT 구조의 bootsector 영역을 보여준다. bootsector 영역에는 다른 파일시트템의 superblock 에서처럼 파일시스템에 대한 모든 정보가 포함되는데, 위에 처음 빨간 네모 칸부터 분석해보면

00 02 => 02 00 = 512byte 가 나오며 sector 당 512byte 가 저장된다는 의미이고

08 => 한 클러스터당 8 개의 sector 가 포함된다.

20 00 => 00 20 = 32setor 이며 예약된 영역의 크기를 의미함 즉,

32 x 512 = 16384 가 나오는데, 이 수를 16 진수로 바꿔주면 4000 이라는 숫자가 나오며 메모리 영역 4000 에서부터 예약된 영역이 시작됨을 알수 있다.

밑에 두 번째 네모 칸 fc 07 00 00 은 FAT 영역의 크기를 나타낸다

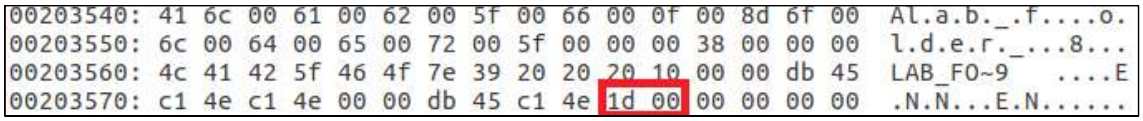
fc 07 00 00 => 00 00 07 fc = 2044 라는 값이 나오는데, FAT 영역이 두 개 만들어 지므로 곱하기 2 와 정확한 크기를 알기 위해 곱하기 512 를 해준다

==> 2044 x 2 x 512 = 2093056, HEX 로 나타내면 1F F000 이다.

이제 DATA 영역으로 내려갈수 있게 되었다. 데이터 영역으로 내려가기 위해서는 BOOT 영역 크기 + 예약된 영역 + FAT 영역 크기 이다.

==> 0 + 4000 + 1F F000 = 20 $3000_{(16)}$

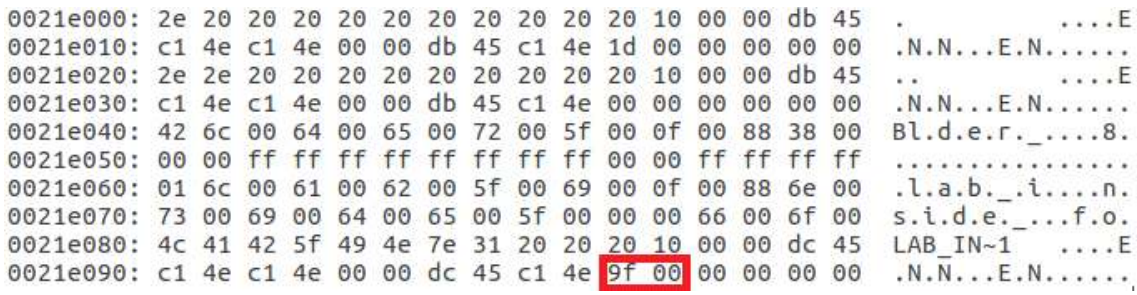
즉, 메모리 위치 20 3000 으로 내려가면 그 부분부터 데이터 영역이 시작 되게 된다.



위의 그림은 박동학의 학번을 나눈 나머지 8 을 이용해 8.txt 파일을 찾기 위해데이터 영역에서 lab_file 8 의 정보를 찾은 상황이다. lab_file 8 번으로 가기 위해서는 high byte 와 low byte 를 가지고 다음 클러스터를 찾을수 있다 high byte 는 대게 00 00 이기 때문에 생략하겠다.

low byte = 1d 00 = 00 1d 인데, 한 클러스터의 크기는 512 X 8 = 4096 이여서 HEX 로 1000 이 된다. 즉, 1d 000 만큼 이동 해야된다는 것을 알아냈고, 데이터 영역은 20 3000 에서 실제 데이터가 시작 되지만, 그 위에 root directory 를 가지고 있는 클러스터가 2 개 존재 하므로, 20 3000 - 2000 을 한 20 1000 이 완전한 데이터 영역 시작 부분이라고 볼수 있다.

==> 20 1000 + 1d 000 = 21 e000, 이 위치로 가면 lab_file 8 번을 확일 할수 있을 것이다.



위의 그림은 lab_file 8 번으로 이동한 상황이다. 정말 21 e000 의 위치로 가보니 .(자신)부터 폴더의 구성을 살펴 볼수 있었다. 이 폴더 안에는

또 하나의 폴더가 존재 했다. lab_inside 폴더가 존재했는데, 위에서 했던 방식대로 폴더를 이동해보면,

20 1000 + 9f 000 = 2a 0000, 2a 0000 위치로 가보면

```
002a0000: 2e 20 20 20 20 20 20 20 20 20 10 00 00 db 45  .          ....E
002a0010: c1 4e c1 4e 00 00 db 45 c1 4e 9f 00 00 00 00 00  .N.N...E.N......
002a0020: 2e 2e 20 20 20 20 20 20 20 20 10 00 00 db 45  ..         ....E
002a0030: c1 4e c1 4e 00 00 db 45 c1 4e 1d 00 00 00 00 00  .N.N...E.N......
002a0040: 41 38 00 2e 00 74 00 78 00 74 00 0f 00 4a 00 00  A8...t.x.t...J..
002a0050: ff ff ff ff ff ff ff ff ff ff 00 00 ff ff ff ff  ................
002a0060: 38 20 20 20 20 20 20 20 54 58 54 20 00 00 dc 45  8       TXT  ...E
002a0070: c1 4e c1 4e 00 00 dc 45 c1 4e 2a 02 54 1f 00 00  .N.N...E.N*.T...
```

inside 폴더를 가보니, 8.txt 파일을 찾을 수 있었다.

이제 홍승기의 학번을 나눈 나머지 28 번 텍스트 파일을 찾아 보겠다.

```
00203a40: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 76 6f 00  Al.a.b._.f...vo.
00203a50: 6c 00 64 00 65 00 72 00 5f 00 00 00 32 00 38 00  l.d.e.r._...2.8.
00203a60: 4c 41 43 37 32 45 7e 32 20 20 20 10 00 00 db 45  LAC72E~2   ....E
00203a70: c1 4e c1 4e 00 00 db 45 c1 4e 5e 00 00 00 00 00  .N.N...E.N ^.....
```

lab 28 번 파일의 위치를 담고있는 부분을 찾은 그림이다. 위에서 했던 것처럼 클러스터 위치를 찾아보면,

20 1000 + 5e 000 = 25 f000, 이 위치로 가보면,

```
0025f000: 2e 20 20 20 20 20 20 20 20 20 10 00 00 db 45  .          ....E
0025f010: c1 4e c1 4e 00 00 db 45 c1 4e 5e 00 00 00 00 00  .N.N...E.N^.....
0025f020: 2e 2e 20 20 20 20 20 20 20 20 10 00 00 db 45  ..         ....E
0025f030: c1 4e c1 4e 00 00 db 45 c1 4e 00 00 00 00 00 00  .N.N...E.N......
0025f040: 42 6c 00 64 00 65 00 72 00 5f 00 0f 00 88 32 00  Bl.d.e.r._....2.
0025f050: 38 00 00 00 ff ff ff ff ff ff 00 00 ff ff ff ff  8...............
0025f060: 01 6c 00 61 00 62 00 5f 00 69 00 0f 00 88 6e 00  .l.a.b._.i....n.
0025f070: 73 00 69 00 64 00 65 00 5f 00 00 00 66 00 6f 00  s.i.d.e._...f.o.
0025f080: 4c 41 42 5f 49 4e 7e 31 20 20 20 10 00 64 db 45  LAB_IN~1   ..d.E
0025f090: c1 4e c1 4e 00 00 db 45 c1 4e de 00 00 00 00 00  .N.N...E.N......
```

lab 28 번의 정보를 가지고 있는 부분을 찾았다. 위의 동학이 것과 마찬가지로 inside 폴더가 하나더 있는걸 확인 할 수 있었고, 위치를 찾아보면.

20 1000 + de 000 = 2d f000, inside 폴더의 위치를 찾았다.

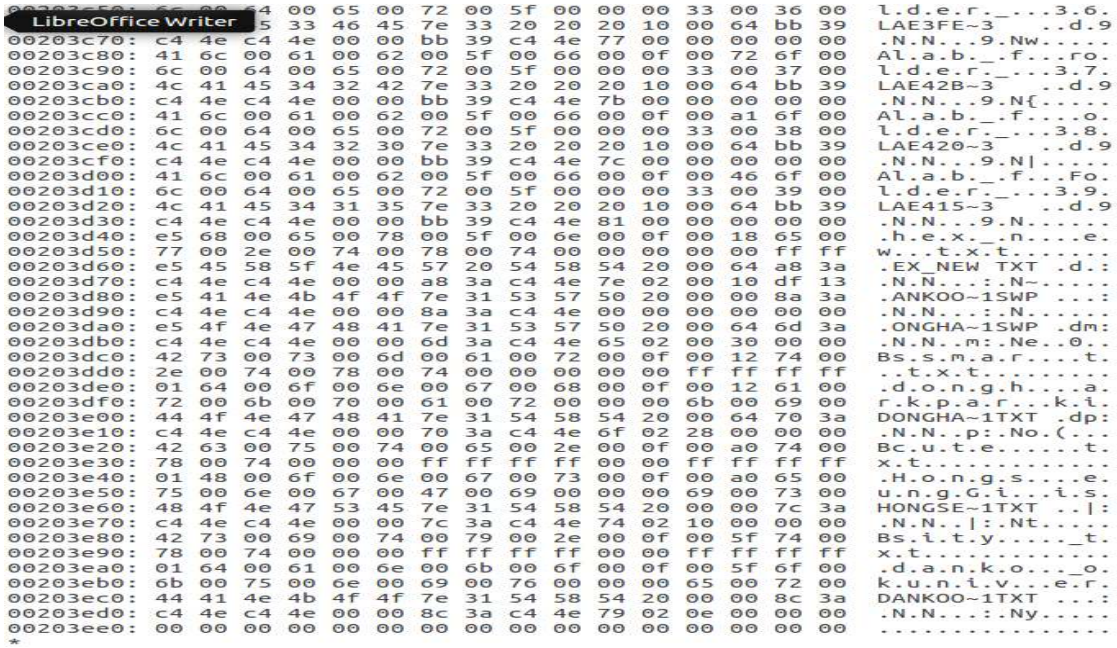
```
002df000: 2e 20 20 20 20 20 20 20 20 20 10 00 00 db 45  .          ....E
002df010: c1 4e c1 4e 00 00 db 45 c1 4e de 00 00 00 00 00  .N.N...E.N......
002df020: 2e 2e 20 20 20 20 20 20 20 20 10 00 00 db 45  ..         ....E
002df030: c1 4e c1 4e 00 00 db 45 c1 4e 5e 00 00 00 00 00  .N.N...E.N^.....
002df040: 41 32 00 38 00 2e 00 74 00 78 00 0f 00 25 74 00  A2.8...t.x...%t.
002df050: 00 00 ff ff ff ff ff ff ff ff 00 00 ff ff ff ff  ................
002df060: 32 38 20 20 20 20 20 20 54 58 54 20 00 64 db 45  28      TXT .d.E
002df070: c1 4e c1 4e 00 00 db 45 c1 4e 6e 01 e5 20 00 00  .N.N...E.Nn.. ..
```

inside 폴더를 찾았더니, 폴더 안에 있는 28.txt 파일을 찾을수 있었다.

다음으로 FAT 구조에 새로운 파일을 추가했을 때 구조의 변화를 확인 해 보겠다.

```
00203b80: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 a9 6f 00  Al.a.b._.f....o.
00203b90: 6c 00 64 00 65 00 72 00 5f 00 00 00 33 00 33 00  l.d.e.r._...3.3.
00203ba0: 4c 41 43 37 33 32 7e 32 20 20 20 10 00 00 db 45  LAC732~2   ....E
00203bb0: c1 4e c1 4e 00 00 db 45 c1 4e 6d 00 00 00 00 00  .N.N...E.Nm.....
00203bc0: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 a1 6f 00  Al.a.b._.f....o.
00203bd0: 6c 00 64 00 65 00 72 00 5f 00 00 00 33 00 34 00  l.d.e.r._...3.4.
00203be0: 4c 41 43 37 33 33 7e 32 20 20 20 10 00 00 db 45  LAC733~2   ....E
00203bf0: c1 4e c1 4e 00 00 db 45 c1 4e 72 00 00 00 00 00  .N.N...E.Nr.....
00203c00: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 19 6f 00  Al.a.b._.f....o.
00203c10: 6c 00 64 00 65 00 72 00 5f 00 00 00 33 00 35 00  l.d.e.r._...3.5.
00203c20: 4c 41 43 37 33 34 7e 32 20 20 20 10 00 00 db 45  LAC734~2   ....E
00203c30: c1 4e c1 4e 00 00 db 45 c1 4e 74 00 00 00 00 00  .N.N...E.Nt.....
00203c40: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 3e 6f 00  Al.a.b._.f...>o.
00203c50: 6c 00 64 00 65 00 72 00 5f 00 00 00 33 00 36 00  l.d.e.r._...3.6.
00203c60: 4c 41 43 37 32 39 7e 32 20 20 20 10 00 00 db 45  LAC729~2   ....E
00203c70: c1 4e c1 4e 00 00 db 45 c1 4e 77 00 00 00 00 00  .N.N...E.Nw.....
00203c80: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 f1 6f 00  Al.a.b._.f....o.
00203c90: 6c 00 64 00 65 00 72 00 5f 00 00 00 33 00 37 00  l.d.e.r._...3.7.
00203ca0: 4c 41 43 37 33 35 7e 32 20 20 20 10 00 00 db 45  LAC735~2   ....E
00203cb0: c1 4e c1 4e 00 00 db 45 c1 4e 7b 00 00 00 00 00  .N.N...E.N{.....
00203cc0: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 96 6f 00  Al.a.b._.f....o.
00203cd0: 6c 00 64 00 65 00 72 00 5f 00 00 00 33 00 38 00  l.d.e.r._...3.8.
00203ce0: 4c 41 43 37 32 41 7e 32 20 20 20 10 00 00 db 45  LAC72A~2   ....E
00203cf0: c1 4e c1 4e 00 00 db 45 c1 4e 7c 00 00 00 00 00  .N.N...E.N|.....
00203d00: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 4a 6f 00  Al.a.b._.f...Jo.
00203d10: 6c 00 64 00 65 00 72 00 5f 00 00 00 33 00 39 00  l.d.e.r._...3.9.
00203d20: 4c 41 43 37 33 36 7e 32 20 20 20 10 00 00 db 45  LAC736~2   ....E
00203d30: c1 4e c1 4e 00 00 db 45 c1 4e 81 00 00 00 00 00  .N.N...E.N......
00203d40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
*
```

위 그림은 초기 데이터 영역을 확인 한 것이다.



위 그림은 새로운 TEXT 파일을 생성했을 때의 변화된 모습이다. lab 39 파일 밑으로 새로운 파일들이 추가된 모습을 확인 할 수 있다.

분석

1) 002a0060 : Offset

2) 38 20 20 20 20 20 20 20 : 파일 이름

3) 54 58 54 : 파일 확장자

4) 20 : 파일 속성

5) 00 : 시스템 보존

6) 00 : 파일 생성 시간(in tenths of seconds)

7) dc 45 : 생성 시간 => 45dc => 01000 101110 11100

=> 8분 46초 28*2초

8) c1 4e : 파일 생성 날짜 => 4ec1

=> 0100111 0110 00001 => 2019년 6월 1일

9) c1 4e : 마지막 방문 날짜 : 2019/6/1

10) 00 00 : High-order 2 bytes of address of first cluster : 실제 내용을 계산해 낼 수 있는 파일 위치 값

11) dc 45 : 파일 수정 시간 : 7)와 동일

12) c1 4e : 파일 수정 날짜 : 8)와 동일

13) 2a 02 : Low-order 2 bytes of address of first cluster

14) 54 1f 00 00 : 파일 크기 : 00 00 1f 54 : 8020

- 28 번 파일에 같은 경우 형식은 같고 내용만 다르기 때문에 생략하겠습니다.

- 2) 보너스 과제

```
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/ramdisk$ lsmod | grep ramdisk
ramdisk                16384  0
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/ramdisk$ ls
Makefile  modules.order  Module.symvers  ramdisk.c  ramdisk.ko  ramdisk.mod.c  ramdisk.mod.o  ramdisk.o
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/ramdisk$
```

```
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/ramdisk$ sudo make
[sudo] password for kernelcamp-fs-2018:
make -C /lib/modules/4.13.0/build SUBDIRS=/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk V= modu
les
make[1]: Entering directory '/usr/src/linux-4.13'
  CC [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk/ramdisk.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk/ramdisk.mod.o
  LD [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk/ramdisk.ko
make[1]: Leaving directory '/usr/src/linux-4.13'
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/ramdisk$ ls
Makefile  modules.order  Module.symvers  ramdisk.c  ramdisk.ko  ramdisk.mod.c  ramdisk.mod.o  ramdisk.o
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/ramdisk$ insmod ramdisk.ko
insmod: ERROR: could not insert module ramdisk.ko: Operation not permitted
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/ramdisk$ sudo insmod ramdisk.ko
```

```
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/fat_practice$ sudo make
make -C /lib/modules/4.13.0/build SUBDIRS=/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice mo
dules
make[1]: Entering directory '/usr/src/linux-4.13'
  CC [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/cache.o
  CC [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/dir.o
  CC [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/fatent.o
  CC [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/file.o
  CC [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/inode.o
  CC [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/misc.o
  CC [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/nfs.o
  LD [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/fat_kernel_camp.o
  CC [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/namei_vfat.o
  LD [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/vfat_lecture.o
  Building modules, stage 2.
  MODPOST 2 modules
  CC      /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/fat_kernel_camp.mod.o
  LD [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/fat_kernel_camp.ko
  CC      /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/vfat_lecture.mod.o
  LD [M]  /home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice/vfat_lecture.ko
make[1]: Leaving directory '/usr/src/linux-4.13'
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/fat_practice$ ls
cache.c   fatent.o              fat_kernel_camp.o  Kconfig        Module.symvers  nfs.o
cache.o   fat_kernel_camp.h     file.c             Makefile       namei_msdos.c   vfat_lecture.ko
dir.c     fat_kernel_camp.ko    file.o             misc.c         namei_vfat.c    vfat_lecture.mod.c
dir.o     fat_kernel_camp.mod.c inode.c            misc.o         namei_vfat.o    vfat_lecture.mod.o
fatent.c  fat_kernel_camp.mod.o inode.o            modules.order  nfs.c           vfat_lecture.o
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/fat_practice$ insmod fat_kernel_c
amp.ko
insmod: ERROR: could not insert module fat_kernel_camp.ko: Operation not permitted
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/fat_practice$ sudo insmod fat_ker
nel_camp.ko
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/fat_practice$ sudo insmod vfat_le
cture.ko
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/fat_practice$ lsmod | grep fat
vfat_lecture           20480  0
fat_kernel_camp        65536  0
kernelcamp-fs-2018@kernelcampfs2018-VirtualBox:~/lecture/Kernelcamp2018/fat_practice$
```

- 안내대로 설정을 하는 과정 -

namei_vfat.c 안에 있는 vfat_mount 안에 시스템 메시지를 출력하는 trace_printk 를 이용해서 학번과 이름을 노출 시킨다.

```
mount - t vfat_lecture /dev/ramdisk /mnt
```

vfat_mount() 함수가 namei_vfat.c 파일에 구현되어 있고 이를 를 통해서

mount 시키고 시스템 로그를 통해서 출력한다.

cat /sys/kernel/debug/tracing/trace_pipe

```
static struct dentry *vfat_mount(struct file_system_type *fs_type,
                        int flags, const char *dev_name,
                        void *data)
{
        trace_printk("Donghak Park : 32151648");
        trace_printk("HongSeung Gi : 32155068");
        return mount_bdev(fs_type, flags, dev_name, data, vfat_fill_super);
}
```

– 결과 (이름 : 학번) –

```
root@kernelcampfs2018-VirtualBox:/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice# cat /sys/k
ernel/debug/tracing/trace_pipe
          mount-4110  [001] ....   654.859850: vfat_mount: Donghak Park : 32151648
          mount-4110  [001] ....   654.859855: vfat_mount: HongSeung Gi : 32155068
```

III. 논의

1. 고찰

– 프로젝트를 진행하면서 어려웠던 점 및 프로젝트 고찰–

32151648 박동학

– 처음 과제를 접했을 때는 정확하게 요구하는 것이 무엇인지 몰라서 팀원인 승기와 과제 설명하는 내용을 몇 번이고 읽어보면서 이해하고자 노력했습니다. 앞선 과제들 보다는 비교적 수월하게 수행했지만 그럼에도 많은 어려움이 있었습니다.

– 환경 설정에 있어서 어려움

설명서에 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 되었습니다. (ex/ dmesg, insmod, rmmod, lsmod 등) 이를 통해서 어떤 과정을 통해서 파일 시스템이 Mount 되는지를 이해하게 되었습니다.

– Hex.txt 파일을 파싱하여 읽는 것에 대한 어려움

작은 파일크기의 저장장치를 간단하게 관리해 주는 FAT 파일 시스템이라 하여 아무리 Hex 파일을 읽어 내는 것이라고 해도 간단하게 읽어 낼 수 있을 줄 알았지만 이론에서 배운 것 만큼 형식이 간단하지도 않았으며 Txt 파일의 용량이 1G 바이트가 넘을 정도로 양이 방대하여 처음에는 많이 당황했습니다. 하지만 인터넷에서 FAT32 의 형식에 대해서 자세히 조사하고 천천히 하나하나 분석하다 보니 처음 생각 했던 것 보다 복잡하지 않다는 것을 알 수 있었습니다. 분석을 통해서 우리가 다루고 있는 파일시스템의 기본적인 정보를 알아내고 그를 통해서 클러스터가 몇 개의 섹터로

이루어져 있는지 (1000 개) 섹터의 크기가 얼마인지 등등을 알아내고 인터넷에도 많이 검색하면서 파일 추적에 대해서 알아 갔습니다. 그렇게 많이 보다 보니 이론적으로 배웠던 부분이 이해되면서 찾아가는 것은 간단했습니다. 하지만 이런 hex 로 된 파일을 처음 분석하다 보니 많이 해멧던 것 같습니다.

-보너스 과제를 하는데 있어서 어려움

보너스 과제의 경우 새로운 이미지파일을 설치하여 하는 것이었는데 이 과제에도 생소한 명령어들도 많고 커널모드로 들어가서 해야 한다는 것을 늦게 알아 당황하면서 환경을 설치하려고 했었습니다. Ramdisk 를 insmod 하고 fat_practice 로 들어가서 모두 수행했을 때는 어디를 수정하여 이름이 출력되게 해야 하는지 많이 고민했고 피피티를 계속해서 보던 중 시스템 메시지라는 것을 보게 되었고 이에 대해서 검색하는 과정에서 printk 의 사용법에 대해서 알게 되었습니다. 이를 알더라도 어떤 함수가 마운트 되는지를 알아야 했는데 교수님께서 의도하셨는지 피피티에 경로가 적혀 있어 이를 토대로 천천히 따라가서 수행을 해보기도하고 파일 일부분이 지워져서 다시처음부터 설치하는 등 많은 고생을 하기도 했습니다. 결과적으로는 매우 간단하게 2 줄의 trace_printk 를 쓰면 되는 보너스 문제였지만 시간은 오히려 더 많이 든 것 같습니다. 함수를 읽어가는 와중에 mutex_lock 과 같은 것을 보니 배운 것이라서 좀 더 반갑운, 간단하면서도 머리 아픈 과제였던 것 같습니다.

32155068 홍승기

이번 프로젝트는 FAT32 파일시스템을 분석하여 자신 학번에 해당되는 text 파일을 찾는 것과, 새로운 파일을 추가했을 때의 파일시스템이 변화된 모습을 확인 해 보는 것이다.

프로젝트가 지금 까지 했던 프로젝트보다는 훨씬 쉬웠지만, 처음 과제를 시작하는 단계에서 FAT 구조를 분석하여 다음 클러스터의 위치를 찾는 방법이나, DATA 영역을 찾는 방법을 제대로 알지 못하여 시간이 좀 오래

걸렸다. 하지만 클러스터를 찾는 방법을 알고 나서 부터는 쉽게 쉽게 다음 과정으로 넘어갈 수 있었다.

boot 영역이 다른 파일시스템에서의 superblock 처럼 이 파일시스템에 관한 모든 정보를 포함 하고 있는 것을 이용하여, sector 의 크기와 클러스터가 몇 개의 sector 로 이루어져 있는지, 그리고 예약된 영역의 크기, FAT 영역의 크기 또한 모두 여기서 확인 할 수 있었다. 프로젝트에서 요구하는 것은 자신의 텍스트 파일이 들어있는 LAB 폴더를 찾고, 그 안에 INSIDE 폴더를 찾은 뒤, 그 안에 있는 TEXT 파일을 찾는 것 이였다, 한번 방법을 알고 나니, 좀 신기하고, 재밌었던 것 같다. FAT 이라는 것이 어떻게 구성이 되어있는지 직관적으로 HEX.TXT 파일로 확인 할 수있어서, 파일시스템이라는 것이 얼마나 복잡하게 이루어져 있는가를 한번 더 느낄 수 있었다. FAT 은 그나마 작은 파일시스템이라고 하셨는데, 다를 파일 시스템들은 어떻게 구성되어 있을 지도 궁금했고, FAT 도 이렇게 복잡한데, etx2, 3, 4 같은 파일 시스템은 얼마나 복잡할지 엄두도 나지 않았다.

보너스 과제를 하면서는 모듈 프로그래밍, 시스템 메시지 의 사용법을 대충이나마 알게 되었고 분석 하다 보니 뮤텍스나 세마포어 같은 상호베제를 위한 lock mechanism 기법들을 또한 확인 할수 있었다. 마운트 할 때 어떤 함수들이 사용되는지도 알 수 있었다.

이번 과제를 하면서 다른 프로젝트보다는 훨씬 쉽게 프로젝트를 마무리 했었지만, 제일 신기하고, 분석하는 과정에서 내가 한 방법이 들어맞을 때마다, 쾌감도 느꼈다. 마지막 프로젝트를 동학이와 무리 없이 끝낼 수 있어서 유종의 미를 거둔 것 같아 기분이 좋다. 한 학기동안 프로젝트를 통해 한번 더 성장 할 수 있었던 것 같다.