과목명	운영체제
담당교수	최종무 교수님
학과	소프트웨어학과
학번	32151671 / 32153180
이름	박민혁 / 이상민
제출일자	2019.06.05

Ⅰ. 프로젝트 개요

1. 프로젝트 목표

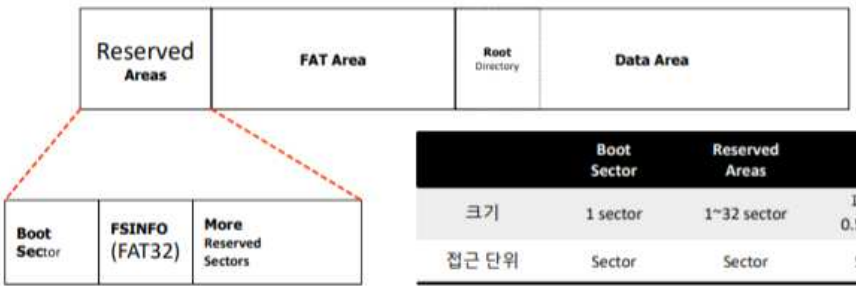
직접 작성한 프로그램을 통해 FAT32 구조 및 설명, 해당되는 파일 위치 및 디렉토리 엔트리 찾아가는 과정을 알아낸다.

2. 프로젝트 설계 과정

1. sudo su : 원활한 작업을 위해 root 권한을 획득한다.
2. insmod ramdisk.ko : kernel object file을 커널 내부에 올린다.
3. mkdir mnt : mount를 위한 임시 디렉토리를 생성한다.
4. mkfs.fat -F 32 /dev/ramdisk : FAT32 파일 시스템을 생성한다.
5. mount /dev/ramdisk mnt : 임시 생성한 디렉토리에 ramdisk를 마운트시킨다.
   이를 통해 ramdisk 장치를 이용할 수 있다.
6. ./create.sh : 프로젝트를 위한 임시 파일들을 설치한다.
7. xxd -a -g 1 -s+0x00 /dev/ramdisk > hex.txt : dump를 뜬다.
8. hex.txt 파일을 통해 몇 번 클러스터에 있는지 분석한다.

Ⅱ. 프로젝트 시행

1. 설계 과정



	Boot Sector	Reserved Areas	FAT Area	Data Area
크기	1 sector	1~32 sector	파티션 0.5% ~ 1%	파티션 99%이상
접근 단위	Sector	Sector	Sector	Cluster

<FAT 기본 구조>

```
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 61 06 27 15 4e 4f 20 4e 41 4d 45 20 20  ..)a.'.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  ................
```

FAT cluster는 실제로 little endian으로 저장된다.

00 02 -> 02 00 (hex) : sector 당 byte 수 = 512byte (dec)

08 (hex) : cluster 당 sector 수 = 8개 (dec)

20 00 -> 00 20 (hex) : reserved area 크기 = 20(hex) * 512(dec) = 4000(hex)
따라서 4000부터 reserved area가 시작된다.
fc 07 00 00 -> 00 00 07 fc (hex) : FAT32 크기 = 07fc(hex) * 512(dec) * 2
= 1F F000(hex)
(FAT은 백업용까지 두 개를 만들기 때문에 곱하기 2를 해준다.)

```
00202fb0: ff ff ff 0f 00 00 00 00 ff ff ff 0f 00 00 00 00  ................
00202fc0: ff ff ff 0f ff ff ff 0f 00 00 00 00 ff ff ff 0f  ................
00202fd0: ff ff ff 0f ff ff ff 0f 00 00 00 00 00 00 00 00  ................
00202fe0: ff ff ff 0f ff ff ff 0f ff ff ff 0f ff ff ff 0f  ................
00202ff0: 00 00 00 00 00 00 00 00 ff ff ff 0f 00 00 00 00  ................
00203000: 30 20 20 20 20 20 20 20 20 20 20 10 00 00 cd 8a  0           .....
00203010: c3 4e c3 4e 00 00 23 8d c3 4e 03 00 00 00 00 00  .N.N..#..N......
00203020: 31 20 20 20 20 20 20 20 20 20 20 10 00 00 e4 8a  1           .....
00203030: c3 4e c3 4e 00 00 32 8d c3 4e 62 27 00 00 00 00  .N.N..2..Nb'....
00203040: 32 20 20 20 20 20 20 20 20 20 20 10 00 64 f8 8a  2           ..d..
00203050: c3 4e c3 4e 00 00 42 8d c3 4e c1 4e 00 00 00 00  .N.N..B..N.N....
00203060: 33 20 20 20 20 20 20 20 20 20 20 10 00 00 0f 8b  3           .....
00203070: c3 4e c3 4e 00 00 50 8d c3 4e 20 76 00 00 00 00  .N.N..P..N v....|
00203080: 34 20 20 20 20 20 20 20 20 20 20 10 00 00 26 8b  4           ...&.
00203090: c3 4e c3 4e 00 00 61 8d c3 4e 7f 9d 00 00 00 00  .N.N..a..N......
002030a0: 35 20 20 20 20 20 20 20 20 20 20 10 00 00 3b 8b  5           ...;.
002030b0: c3 4e c3 4e 00 00 70 8d c3 4e de c4 00 00 00 00  .N.N..p..N......
002030c0: 36 20 20 20 20 20 20 20 20 20 20 10 00 64 52 8b  6           ..dR.
002030d0: c3 4e c3 4e 00 00 81 8d c3 4e 3d ec 00 00 00 00  .N.N.....N=.....
002030e0: 37 20 20 20 20 20 20 20 20 20 20 10 00 00 69 8b  7           ...i.
002030f0: c3 4e c3 4e 01 00 90 8d c3 4e 9c 13 00 00 00 00  .N.N.....N......
```

data area : 1F F000(hex) + 4000(hex) = 0020 3000(hex)

2. 설계 결과

<이상민 - 0.txt>

```
00203300: 32 34 20 20 20 20 20 20 20 20 20 10 00 00 d4 3b  24          ....;
00203310: c4 4e c4 4e 03 00 cf 3d c4 4e eb b0 00 00 00 00  .N.N...=.N......
00203320: 32 35 20 20 20 20 20 20 20 20 20 10 00 64 fa 3b  25          ..d.;
00203330: c4 4e c4 4e 03 00 e1 3d c4 4e 4a d8 00 00 00 00  .N.N...=.NJ.....
00203340: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 8c 6f 00  Al.a.b._.f....o.
00203350: 6c 00 64 00 65 00 72 00 5f 00 00 00 30 00 00 00  l.d.e.r._...0...
00203360: 4c 41 42 5f 46 4f 7e 31 20 20 20 10 00 64 e9 3d  LAB_FO~1   ..d.=
00203370: c4 4e c4 4e 00 00 e9 3d c4 4e 04 00 00 00 00 00  .N.N...=.N......
```

(student number % 20).txt = 80 % 20.txt = 0.txt
04 00 -> 00 04 (hex) : cluster이므로 뒤에 000을 붙이면 4000(hex)가 된다.

```
00205000: 2e 20 20 20 20 20 20 20 20 20 20 10 00 00 e9 3d  .           ....=
00205010: c4 4e c4 4e 00 00 e9 3d c4 4e 04 00 00 00 00 00  .N.N...=.N......
00205020: 2e 2e 20 20 20 20 20 20 20 20 20 10 00 00 e9 3d  ..          ....=
00205030: c4 4e c4 4e 00 00 e9 3d c4 4e 00 00 00 00 00 00  .N.N...=.N......
00205040: 42 6c 00 64 00 65 00 72 00 5f 00 0f 00 88 30 00  Bl.d.e.r._....0.
00205050: 00 00 ff ff ff ff ff ff ff ff 00 00 ff ff ff ff  ................
00205060: 01 6c 00 61 00 62 00 5f 00 69 00 0f 00 88 6e 00  .l.a.b._.i....n.
00205070: 73 00 69 00 64 00 65 00 5f 00 00 00 66 00 6f 00  s.i.d.e._...f.o.
00205080: 4c 41 42 5f 49 4e 7e 31 20 20 20 10 00 64 ea 3d  LAB_IN~1   ..d.=
00205090: c4 4e c4 4e 00 00 ea 3d c4 4e 83 00 00 00 00 00  .N.N...=.N......
002050a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

data area 0020 3000 위에 2개의 cluster가 더 있으므로 2000(hex)를 빼준 0020 1000에 위에서 구한 4000(hex)를 더해 0020 5000으로 간다.
83 00 -> 00 83 (hex) : 위와 같은 방법으로 뒤에 000을 붙이면 83000(hex)가 된다.
똑같이 0020 1000에 83000(hex)를 더해 0028 4000으로 간다.

```
00284000: 2e 20 20 20 20 20 20 20 20 20 20 10 00 64 e9 3d  .          ..d.=
00284010: c4 4e c4 4e 00 00 e9 3d c4 4e 83 00 00 00 00 00  .N.N...=.N......
00284020: 2e 2e 20 20 20 20 20 20 20 20 20 10 00 64 e9 3d  ..         ..d.=
00284030: c4 4e c4 4e 00 00 e9 3d c4 4e 04 00 00 00 00 00  .N.N...=.N......
00284040: 41 30 00 2e 00 74 00 78 00 74 00 0f 00 f8 00 00  A0...t.x.t......
00284050: ff ff ff ff ff ff ff ff ff ff 00 00 ff ff ff ff  ................
00284060: 30 20 20 20 20 20 20 20 54 58 54 20 00 64 ea 3d  0       TXT .d.=
00284070: c4 4e c4 4e 00 00 ea 3d c4 4e 60 02 54 1f 00 00  .N.N...=.N`.T...
00284080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

<Directory entry>

<0.txt file entry>

offset : 00284060

name : 3020202020202020 -> 2020202020202030, 0.txt

extension : 545854 -> 545854

attribute : 20, archive file

reserved : 0064 -> 6400

create time : ea3d -> 3dea(hex) = 0011 1101 1110 1010(bin)
= 7시 23분 42초

created date : c44e -> 4ec4(hex) = 0100 1110 1100 0100(bin)
= 2019년 6월 4일

last accessed date : c44e -> 4ec4 = 2019년 6월 4일

write time : ea3d -> 3dea = 7시 23분 42초

write date : c44e -> 4ec4 = 2019년 6월 4일

starting cluster high bytes : 0000 -> 0000

starting cluster low bytes : 6002 -> 0260

file size : 541f0000 -> 00001f54(hex) = 8020(dec) bytes

<박민혁 - 31.txt>

```
00203b00: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 a4 6f 00  Al.a.b._.f....o.
00203b10: 6c 00 64 00 65 00 72 00 5f 00 00 00 33 00 31 00  l.d.e.r._...3.1.
00203b20: 4c 41 38 37 33 35 7e 32 20 20 20 10 00 00 19 6a  LA8735~2   ....j
00203b30: c1 4e c1 4e 00 00 19 6a c1 4e 66 00 00 00 00 00  .N.N...j.Nf.....
```

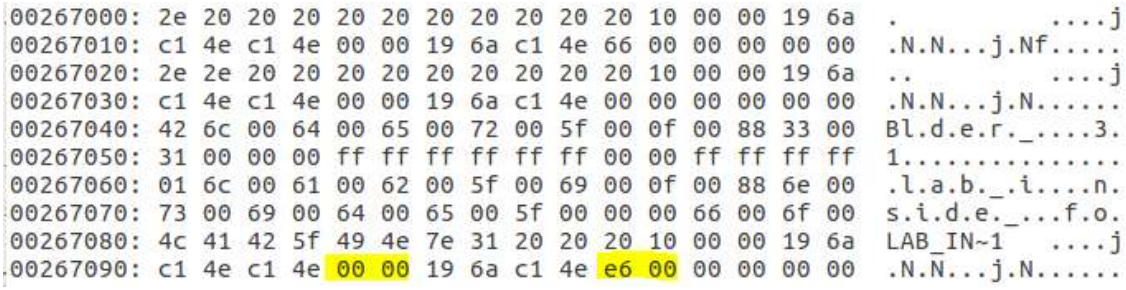
Starting Cluster High Byte + Starting Cluster Low Bytes = 00000066 = 66000

Data = 00203000 - > Cluster 4번째 영역 위에 두 개 영역이 있다.

한 Cluster는 8개의 Sector로 구성. 따라서 $8 \times 512 = 4096$ 16진수로 바꾸면 1000이 된다.

그래서 Data 영역의 본 시작점은 00201000이다.

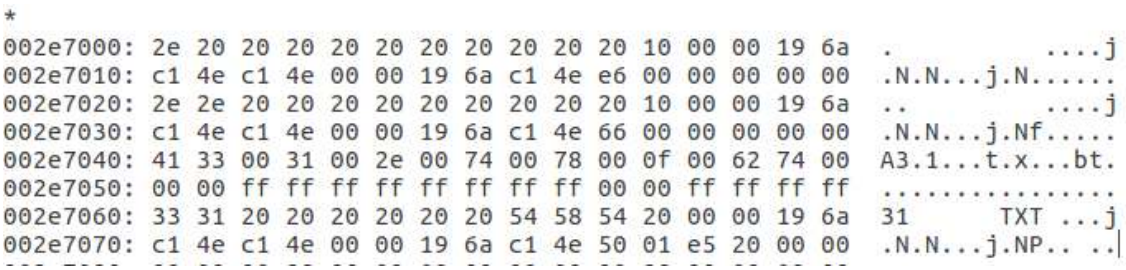
00201000+66000=00267000 으로 가면 두 번째 파일이 나온다.



```
00267000: 2e 20 20 20 20 20 20 20 20 20 20 10 00 00 19 6a  .          ....j
00267010: c1 4e c1 4e 00 00 19 6a c1 4e 66 00 00 00 00 00  .N.N...j.Nf.....
00267020: 2e 2e 20 20 20 20 20 20 20 20 20 10 00 00 19 6a  ..         ....j
00267030: c1 4e c1 4e 00 00 19 6a c1 4e 00 00 00 00 00 00  .N.N...j.N......
00267040: 42 6c 00 64 00 65 00 72 00 5f 00 0f 00 88 33 00  Bl.d.e.r._....3.
00267050: 31 00 00 00 ff ff ff ff ff ff 00 00 ff ff ff ff  1...............
00267060: 01 6c 00 61 00 62 00 5f 00 69 00 0f 00 88 6e 00  .l.a.b._.i....n.
00267070: 73 00 69 00 64 00 65 00 5f 00 00 00 66 00 6f 00  s.i.d.e._...f.o.
00267080: 4c 41 42 5f 49 4e 7e 31 20 20 20 10 00 00 19 6a  LAB_IN~1   ....j
00267090: c1 4e c1 4e 00 00 19 6a c1 4e e6 00 00 00 00 00  .N.N...j.N......
```

Starting Cluster High Byte + Starting Cluster Low Bytes = 000000e6 = e6000

00201000+e6000=2e7000 으로 가면 세 번째 파일이 나온다.



```
*
002e7000: 2e 20 20 20 20 20 20 20 20 20 20 10 00 00 19 6a  .          ....j
002e7010: c1 4e c1 4e 00 00 19 6a c1 4e e6 00 00 00 00 00  .N.N...j.N......
002e7020: 2e 2e 20 20 20 20 20 20 20 20 20 10 00 00 19 6a  ..         ....j
002e7030: c1 4e c1 4e 00 00 19 6a c1 4e 66 00 00 00 00 00  .N.N...j.Nf.....
002e7040: 41 33 00 31 00 2e 00 74 00 78 00 0f 00 62 74 00  A3.1...t.x...bt.
002e7050: 00 00 ff ff ff ff ff ff ff ff 00 00 ff ff ff ff  ................
002e7060: 33 31 20 20 20 20 20 20 54 58 54 20 00 00 19 6a  31      TXT ...j
002e7070: c1 4e c1 4e 00 00 19 6a c1 4e 50 01 e5 20 00 00  .N.N...j.NP.. ..
```

<31.txt file entry>

offset : 002e7060

name : 3331202020202020 -> 2020202020203133, 0.txt

extension : 545854 -> 545854

attribute : 20, archive file

reserved : 0000

create time : 196a -> 6a19(hex) = 0110 1010 0001 1001(bin)

= 13시 17분 25초

created date : c14e -> 4ec1(hex) = 0100 1110 1100 0001(bin)

= 2019년 6월 1일

last accessed date : c44e -> 4ec4 = 2019년 6월 1일

write time : 196a -> 6a19 = 13시 17분 25초

write date : c14e -> 4ec1 = 2019년 6월 1일

starting cluster high bytes : 0000 -> 0000

starting cluster low bytes : 5001 -> 0150

file size : e5200000 -> 000020e5(hex) = 8421(dec) bytes

<.txt file 3개 추가>

```
00203c20: 4c 41 38 37 32 42 7e 32 20 20 20 10 00 00 19 6a  LA872B~2   ....j
00203c30: c1 4e c1 4e 00 00 19 6a c1 4e 74 00 00 00 00 00  .N.N...j.Nt.....
00203c40: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 08 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 38 37 32 30 7e 32 20 20 20 10 00 00 19 6a  LA8720~2   ....j
00203c70: c1 4e c1 4e 00 00 19 6a c1 4e 77 00 00 00 00 00  .N.N...j.Nw.....
00203c80: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 ac 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 38 37 31 35 7e 32 20 20 20 10 00 00 19 6a  LA8715~2   ....j
00203cb0: c1 4e c1 4e 00 00 19 6a c1 4e 7b 00 00 00 00 00  .N.N...j.N{.....
00203cc0: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 21 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 38 37 34 44 7e 32 20 20 20 10 00 00 19 6a  LA874D~2   ....j
00203cf0: c1 4e c1 4e 00 00 19 6a c1 4e 7c 00 00 00 00 00  .N.N...j.N|.....
00203d00: 41 6c 00 61 00 62 00 5f 00 66 00 0f 00 10 6f 00  Al.a.b._.f....o.
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 38 37 34 32 7e 32 20 20 20 10 00 00 19 6a  LA8742~2   ....j
00203d30: c1 4e c1 4e 00 00 19 6a c1 4e 81 00 00 00 00 00  .N.N...j.N......
00203d40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

<.txt file 추가 전 본래 data area 끝 부분>

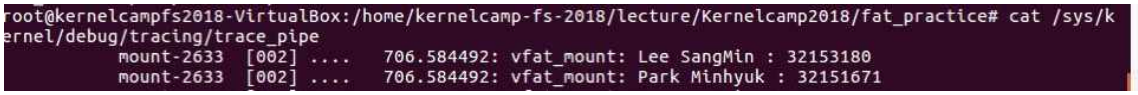
```
root@os-lecture:/home/os-lecture/lab3/2019_DKU_OS/lab3_filesystem/mnt# vi sang.t
xt
root@os-lecture:/home/os-lecture/lab3/2019_DKU_OS/lab3_filesystem/mnt# vi min.tx
t
root@os-lecture:/home/os-lecture/lab3/2019_DKU_OS/lab3_filesystem/mnt# vi hyuk.t
xt
root@os-lecture:/home/os-lecture/lab3/2019_DKU_OS/lab3_filesystem/mnt# ls
0   2   8              lab_folder_17  lab_folder_28  lab_folder_39
1   20  9              lab_folder_18  lab_folder_29  lab_folder_4
10  21  hyuk.txt       lab_folder_19  lab_folder_3   lab_folder_5
11  22  lab_folder_0   lab_folder_2   lab_folder_30  lab_folder_6
12  23  lab_folder_1   lab_folder_20  lab_folder_31  lab_folder_7
13  24  lab_folder_10  lab_folder_21  lab_folder_32  lab_folder_8
14  25  lab_folder_11  lab_folder_22  lab_folder_33  lab_folder_9
15  3   lab_folder_12  lab_folder_23  lab_folder_34  min.txt
16  4   lab_folder_13  lab_folder_24  lab_folder_35  sang.txt
17  5   lab_folder_14  lab_folder_25  lab_folder_36
18  6   lab_folder_15  lab_folder_26  lab_folder_37
19  7   lab_folder_16  lab_folder_27  lab_folder_38
```

<sang.txt / min.txt / hyuk.txt 3개의 .txt file을 추가>

```
00203d40: e5 2e 00 68 00 79 00 75 00 6b 00 0f 00 4b 2e 00  ...h.y.u.k...K..
00203d50: 74 00 78 00 74 00 2e 00 73 00 00 00 77 00 70 00  t.x.t...s...w.p.
00203d60: e5 59 55 4b 54 58 7e 31 53 57 50 20 00 00 40 5b  .YUKTX~1SWP ..@[
00203d70: c4 4e c4 4e 00 00 40 5b c4 4e 70 02 00 10 00 00  .N.N..@[.Np.....
00203d80: 41 73 00 61 00 6e 00 67 00 2e 00 0f 00 ec 74 00  As.a.n.g......t.
00203d90: 78 00 74 00 00 00 ff ff ff ff 00 00 ff ff ff ff  x.t.............
00203da0: 53 41 4e 47 20 20 20 20 54 58 54 20 00 00 34 5b  SANG    TXT ..4[
00203db0: c4 4e c4 4e 00 00 34 5b c4 4e 69 02 0e 00 00 00  .N.N..4[.Ni.....
00203dc0: 41 6d 00 69 00 6e 00 2e 00 74 00 0f 00 a2 78 00  Am.i.n...t....x.
00203dd0: 74 00 00 00 ff ff ff ff ff ff 00 00 ff ff ff ff  t...............
00203de0: 4d 49 4e 20 20 20 20 20 54 58 54 20 00 64 3b 5b  MIN     TXT .d;[
00203df0: c4 4e c4 4e 00 00 3b 5b c4 4e 6f 02 0d 00 00 00  .N.N..;[.No.....
00203e00: 41 68 00 79 00 75 00 6b 00 2e 00 0f 00 86 74 00  Ah.y.u.k......t.
00203e10: 78 00 74 00 00 00 ff ff ff ff 00 00 ff ff ff ff  x.t.............
00203e20: 48 59 55 4b 20 20 20 20 54 58 54 20 00 64 44 5b  HYUK    TXT .dD[
00203e30: c4 4e c4 4e 00 00 44 5b c4 4e 74 02 0e 00 00 00  .N.N..D[.Nt.....
00203e40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

<3개의 .txt file을 추가한 후 data area>

<학번과 이름 출력>



static struct dentry *vfat_mount func trace_printk를 이용하여 학번과 이름 입력
trace_printk : ftrace buffer에서의 printf format (system message print)

Ⅲ. 프로젝트 결과

1. *박민혁 고찰*

사실상 이번 과제를 처음 시작 할 때부터 과제의 요점이 무엇인지 잘 몰랐다. 아는 것은 FAT File System 분석이라는 것만 알았다. 그래서 주변 사람들에게 물어 봐서, 과제의 요점을 파악했다. FAT File System을 이용해 내 학번에서 뒷자리 두 개를 40으로 나눈 값의 파일을 찾아가는 것이 과제였다. 과제를 알고 나니 FAT File System 구조를 알아야만 과제를 풀어 나갈 수 있었다.

하지만 우리는 과제를 시작 했을 때 상민이와 나는 처음부터 많이 꼬였었다. 상민이는 뭐가 문젠지는 모르겠지만 ./create.sh가 되지 않았고, 나는 계속 Directory를 읽을 수 없다고 떠서 오류인줄 알아서 이것저것 검색 하다가 Virtual box에 cmd가 사라지기도 하였다. 그래서 처음부터 다시 Virtual box에 OS_LAB 파일을 올리는 것부터 다시 시작했다. 처음부터 많이 꼬이기 시작하여서 과제를 하는데 어려움이 있었다.

다시 설치를 한 이후부터는 신중히 가했다. RAM을 이용하는 것이기 때문에 Virtual box를 끄게 되면 올려놨던 파일들이 다 날라 가기 시작해서 한 번에 끝을 내자는 생각으로 시작했다. 시작하기 전에 주변에서 0020 3000이라는 숫자만 알면 된다고 하여서 일단 무작정 LAB3 과제 설명과 PPT 10장을 보면서 시작을 했다. 환경 구성이 끝나고 hex.txt 파일에서 0020 3000을 찾아가 31번째 File을 찾아 가는 것은 쉬웠지만, 그 다음 파일을 찾아 가는 방법을 잘 몰랐다. 그래서 운영체제 강의자료 10장에서 다시 FAT 구조를 살펴봤다. FAT 구조를 찾아 보다 Starting Cluster High byte와 Starting Cluster Low Bytes를 이용해 다음 파일 구조를 찾아 갔다. 다음 파일에서도 똑같이 해서 마지막 파일까지 찾아갔다.

이렇게 하고 나서 끝인 줄 알았다. 하지만 왜 0020 3000이 Data 영역인지 의문을 가졌다. 그래서 FAT구조를 살펴보고 첫 번째 Cluster를 분석하기 시작했다. 처음에는 분석을 했을 때 hex dec 변환이 어려웠다. 그래서 주변에 도움을 받고 변환할 때를 잘 알고 하다 보니 Data 영역까지 구하는 식이 진행되었다. 하지만 처음에 왜 우리가 계산한 값이 0020 3000이 안 나오는지 몰랐다. FAT은 백업용까지 두 개를 만드는

것을 몰랐기 때문이다. 계속적으로 정보를 찾았고 그 결과 FAT은 두 개를 만드는 것을 알게 되어서 2를 곱하니 0020 3000 이 Data 영역인 것을 알았다.

Data 영역으로 가서 Cluster를 다시 분석 하였고, 한 Cluster는 8개의 Sector로 구성 돼 있고 따라서 위의 결과 부분에서 계산한 것처럼 1000이라는 숫자가 나온다. 그리고 다음 파일을 찾아갈 때 주의할 점이 Data 영역 위에 Root Directory가 Cluster를 두 개 차지한다. 그래서 0020 1000에서 위에 결과에서 계산한 값을 더해서 두 번째 File 세 번째 File을 찾아갔다.

위의 과제가 끝나고 mnt Directory에 sang.txt min.txt hyuk.txt file을 추가해서 Data 영역을 보는 것과 Bonus 과제를 하는데 어려움이 있지 않았다. 그냥 mnt Directory file에 가서 파일을 추가하고 Bonus 과제는 image file을 다운 받고 trace_printk를 왜 사용하는지에 대해서 찾는 것이 어려웠다.

이번 과제는 LAB1 LAB2 과제보다는 시간이 오래 걸리지 않았다. 하지만 환경을 만드는 것에만 시간이 많이 들어서 오래 걸린 기분이었다. 분석 하는 방법을 알지 못했을 때는 어려움이 있었지만 검색하고 자료들을 찾아봐서 알고 나니 어려운 과제는 아니었던 것 같다. 운영체제 마지막 과제를 잘 마무리해서 다행이라고 생각한다. 운영체제와 시스템 프로그래밍 과제를 하면서 시간이 오래 걸리긴 했지만 잘 해결해 나가서 조금은 내 지식이 쌓였지 않았을 까란 생각이 들었다.

2. *이상민 고찰*

이번 프로젝트가 처음 나왔을 때 lab1이나 lab2처럼 따로 코딩을 하지 않아도 되기 때문에 쉬울 것이라고 생각했다. 하지만 프로젝트를 진행해본 결과 file system의 전반적인 개념과 FAT의 기본 구조를 잘 숙지하고 있어야 쉽게 분석할 수 있는 프로젝트였다. 처음에 가장 고생했던 것은 스크립트를 실행시키는 과정이었다. 마운트 후 ./create.sh 명령어를 실행할 때 계속 메모리가 부족하다는 메시지가 뜨면서 virtual machine 동작이 중지되었다. 처음에는 단지 노트북 메모리가 부족한 줄 알고 쓰지 않는 파일들을 정리했다. 그래도 같은 메시지가 계속 떴다. 불현듯이 내 노트북 RAM이 4GB밖에 되지 않는다는 사실과 처음 virtual machine을 설정할 때 4GB 모두 사용하도록 설정해둔 것이 생각났다. 그래서 RAM을 3GB 정도로만 잡아주고 스크립트를 실행시켰더니 정상적으로 잘 돌아갔다. 내 노트북으로 프로젝트를 하지 못하면 어떡하나 걱정을 많이 했는데 참으로 다행이었다.

sudo su 명령어를 처음 써봤는데 관리자 모드로 진입할 수 있는 것이 신기했다. 관리자 모드로 들어가 ramdisk.ko module을 올리고 FAT으로 포맷하고 mount 했다. FAT은 일반적으로 sector 당 512byte의 용량을 갖으며, 이러한 sector 8개를 모아 4KB의 cluster라고 한다. 처음에 xxd -a -g 1 -s+0x00 /dev/ramdisk > hex.txt 명령어를 통해 hex.txt로 내용을 옮기고 파일을 확인했을 때는 막막한 감이 없지 않아 있었다. 16진수로 된 엄청 많은 숫자들을 분석하는 것이 가능할까 싶었다. 우선 파일 첫 부분에 위에서 말한 기본적인 FAT 구조에 대한 내용들이 담겨있었다. 이러한 내용을 확인하고 우리는 reserved area(예약된 영역)를 찾아야 했다. 기본적으로 FAT32는 reserved area가 32개의 sector로 구성되어 있다. 따라서 16진수로 20이라고 표현되어 있었다. sector 당 512byte이므로 곱하면 16384가 나오고 16진수로 바꾸면 4000, 즉 offset 4000부터 reserved area가 시작된다. data area(데이터 영역)는 203000 offset부터 시작하는데, data area 위에 2개의 cluster가 있다. 한 cluster 당 8개의 sector가 있으므로 8*512 = 4096byte의 용량을 갖으며, 이것을 16진수로 바꾸면 1000이 된다. 따라서 내가 원하는 파일을 찾기 위해 처음 offset 203000에서 2개의 cluster의 양 2000만큼 빼준 뒤 시작해야 한다. 이러한 식으로 내 파일을 찾고나서 그 파일을 분석하는 것은 어렵지 않았다. 운영체제 수업 시간에 보는 ppt 맨 뒤 appendix에 자세히 나와있었기 때문이다. 그것에 기반하여 0.txt 파일을 분석해보았는데 생각보다 많은 정보들이 들어 있었다. 단순히 파일 이름이나 크기 정도만 있을 줄 알았는데 시간과 날짜 정보도 담겨 있었다. 그리고 3개의 파일을 추가하여 data area가 수정되는 것이 어쩌면 당연한 일인데도 불구하고 눈으로 직접 확인을 할 수 있어서 좋았다.

벌써 운영체제 수업의 마지막 프로젝트라니 실감이 안났다. 작년 시스템 프로그래밍 수업때와는 달리 모든 프로젝트가 팀으로 진행되었다. 솔직히 팀으로 했기 때문에 지금까지 잘 해왔다고 생각한다. 아마 혼자 했더라면 중간에 포기했을텐데 박민혁 학생과 서로 부족한 부분은 채워가며 할 수 있어서 좋았다. 이번 프로젝트를 포함한 모든 프로젝트에서 얻어가는 것이 참 많았고, 한 학기동안 열심히 해준 박민혁 학생과 내 스스로에게도 감사하다.