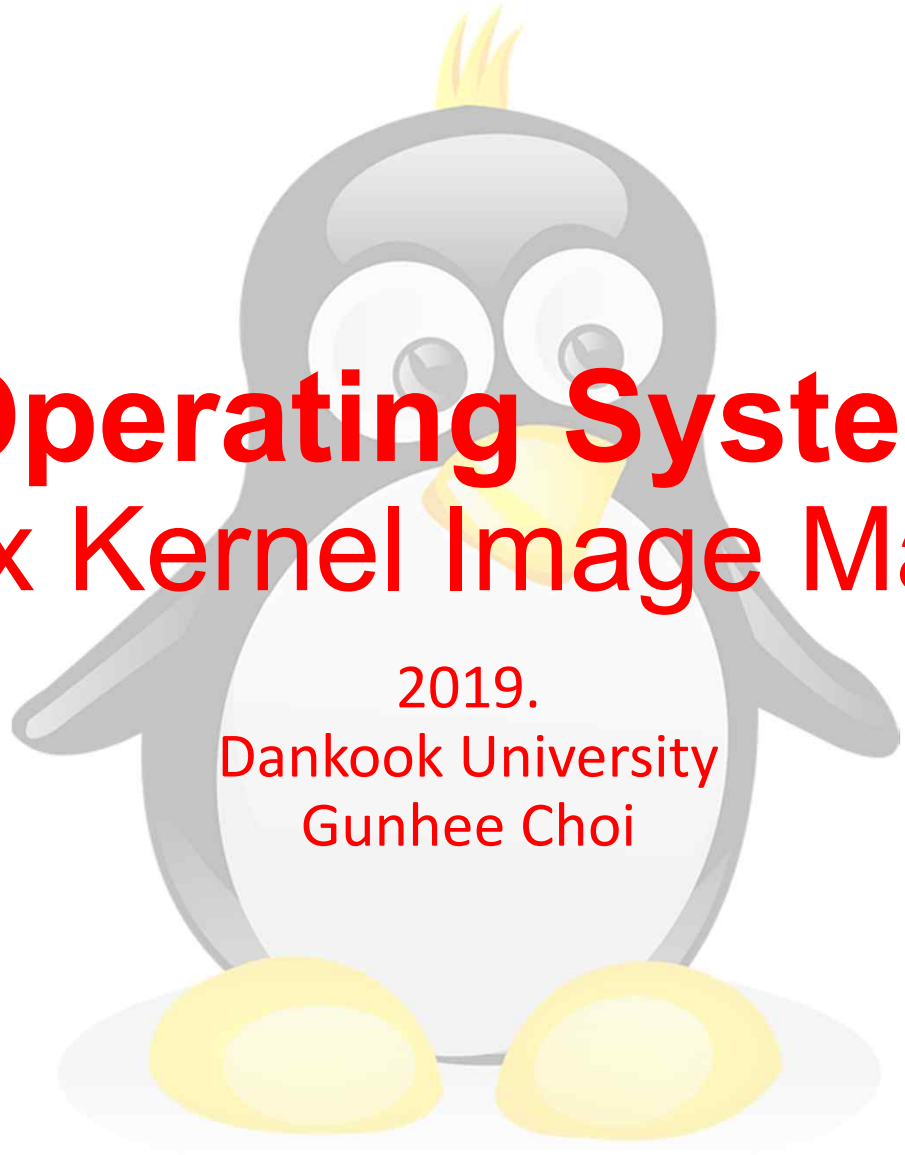


Operating System
- Linux Kernel Image Manual -

2019.

Dankook University
Gunhee Choi

Contents

- FAT 구조
- FAT 분석 환경 세팅
- Bonus 과제

과제 링크

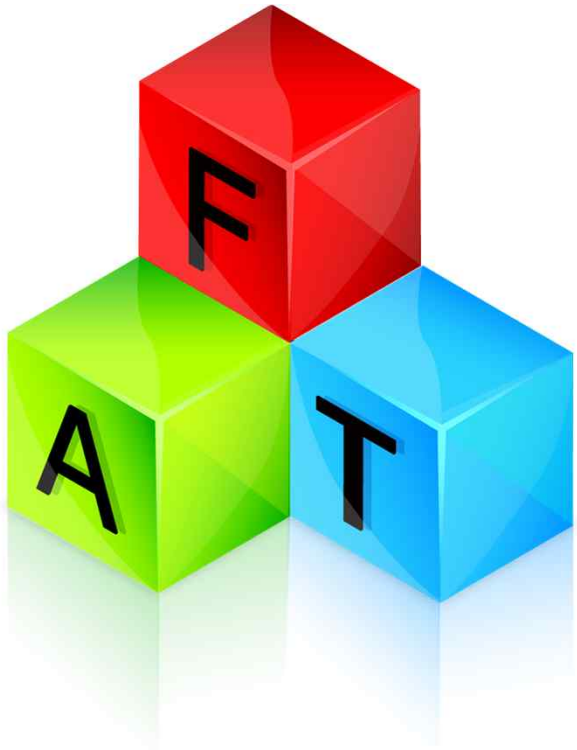
Git

https://github.com/ChoiGunHee/2019_DKU_OS

Kernel camp 2018 이미지 링크

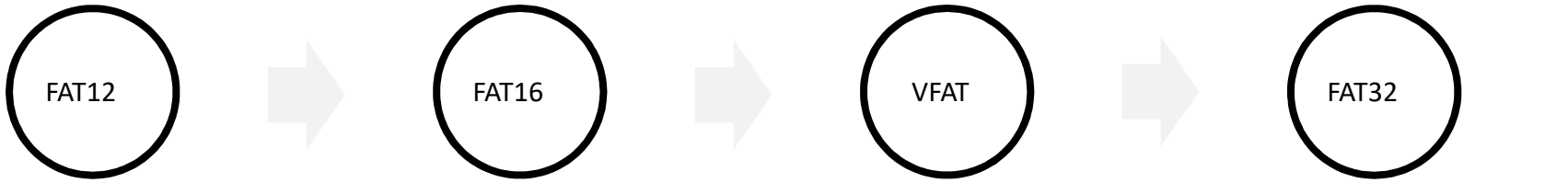
https://drive.google.com/open?id=1nDOef1QCtXNO49R87IVuYgpwCOdOsPK7

FAT 구조



FAT 구조

- FAT file system
  - FAT(File Allocation Table)은 이름 글대로 파일의 할당 정보를 테이블 형식으로 저장하여 관리하는 파일 시스템이다.
  - FAT file system은 가장 범용적인 파일시스템으로, 구조가 간단한 장점을 기반으로 다양한 OS에서 지원하며 USB, 메모리 카드, 디지털 카메라 등에 널리 사용된다.
  - FAT은 크게 FAT12, FAT16, FAT32로 나눌 수 있다.
    FAT 뒤의 숫자는 비트 수로 최대 표현 가능한 클러스터의 수를 의미한다.



1980년대 초
플로피디스크용으로 개발
최대 4,084 cluster

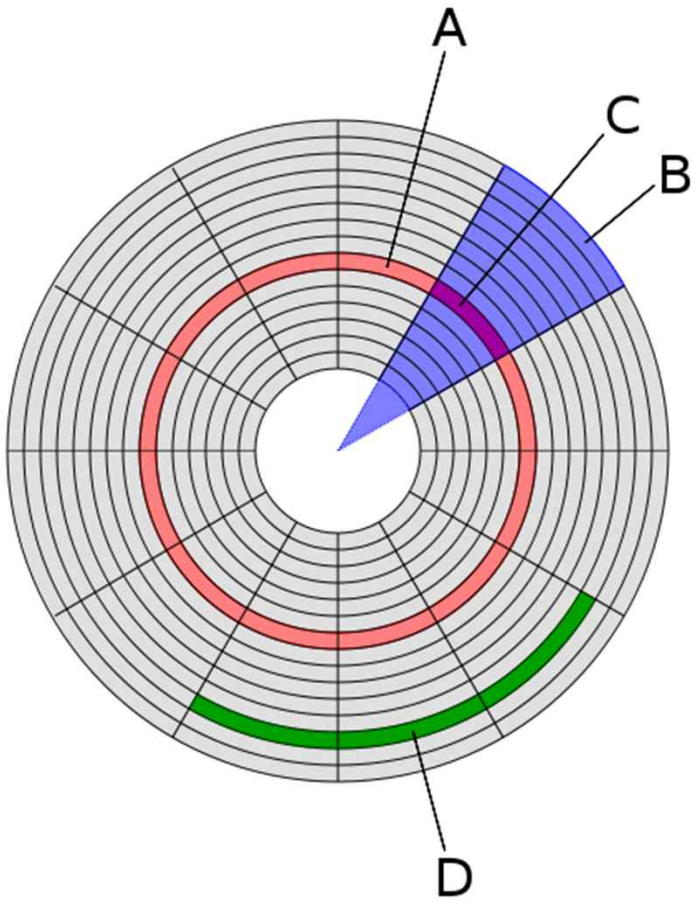
1980년대 말
하드디스크 지원
최대 65,524 cluster

1995년
FAT 성능 향상
긴 파일 이름 지원
최대 65,524 cluster

1996년
VFAT 확장 개발
고용량 하드디스크 지원
최대 268,435,444 cluster

FAT 구조

- Sector, Cluster, Track



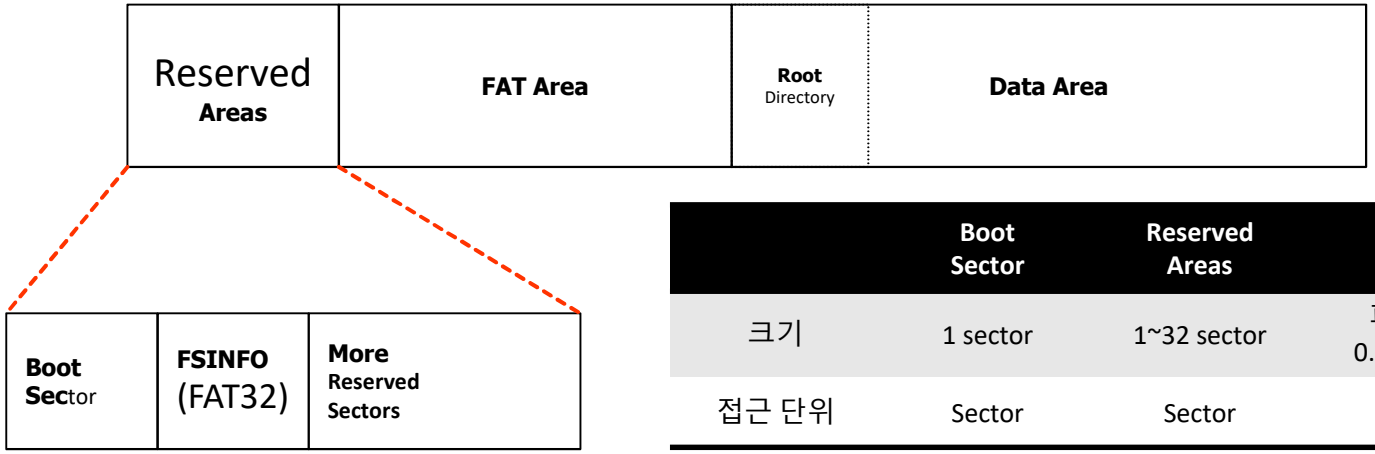
출처 : wikipedia

A : Track
B : Geometrical sector
C : Track sector
D : Cluster

일반적으로 FAT32의 경우,
Sector : 512bytes
Cluster : 8개의 Sector = 4KB

FAT 구조

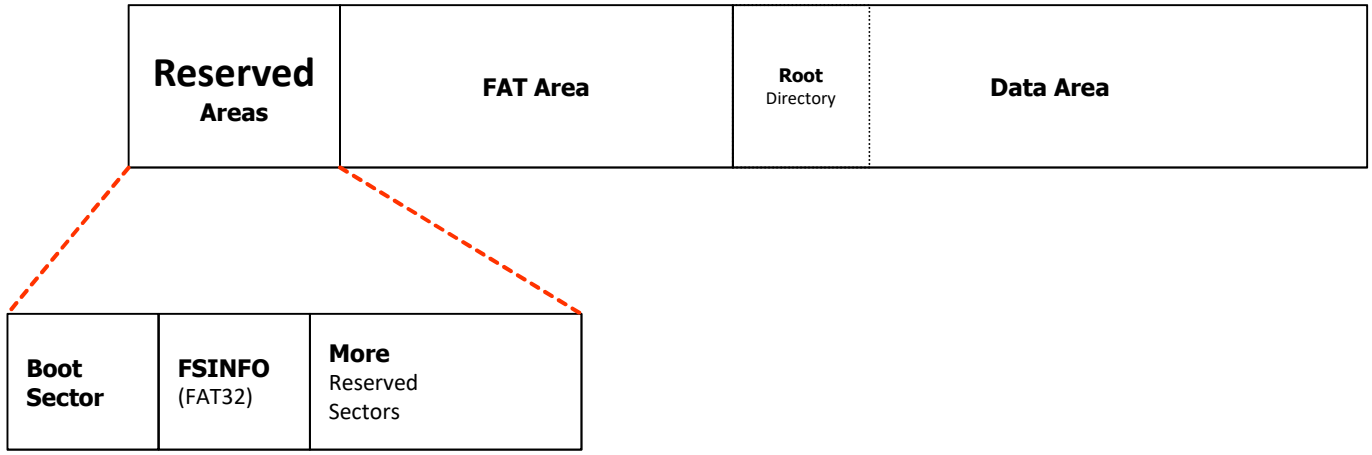
- FAT file system Layout
  - Reserved Areas
    - 예약된 영역으로 Boot Record, FSINFO, 추가 Reserved Area가 있다.
    - Boot Record는 부팅을 위한 여러가지 값을 저장하고 있고, 크기는 1 sector이다.
    - FAT32의 경우에는 FSINFO를 위한 영역이 존재한다.
  - FAT Area
    - FAT Area는 Cluster를 관리하는 테이블이 모여 있는 영역이다.
    - FAT Area를 통해서 어떤 파일에 어떤 클러스터가 연결되었는지 알 수 있다.
    - 매우 중요한 영역이기 때문에 기본적으로 FAT #1, FAT #2로 중복해서 관리한다.
  - Data Area
    - 파일 혹은 디렉토리가 저장되는 공간이다.
    - FAT16에서는 반드시 FAT #2 뒤 영역에 Root Directory가 존재한다.



	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

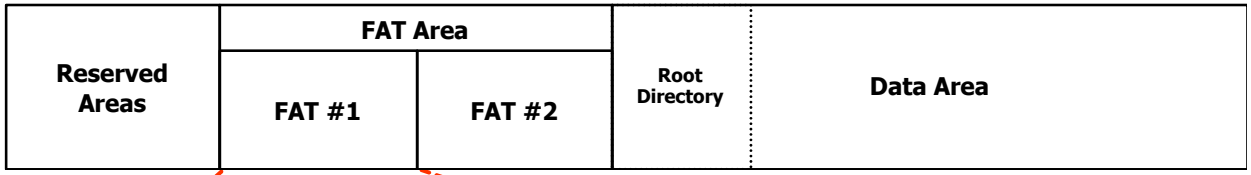
FAT 구조

- Reserved Area
  - 예약된 영역은 FAT File system에서 가장 앞 부분에 위치한 구조로서 여러 개의 sector로 구성된다.
  - 예약된 영역의 기본적으로 FAT16은 1개의 sector, FAT32는 32개의 sector로 구성된다.
  - 예약된 영역은 boot sector, FSINFO, 추가적인 sector로 구분된다.
    - Boot sector : 1개의 sector로 구성되며, 부팅에 필요한 정보들이 기록된다.
    - FSINFO : FAT32에만 사용되는 구조체가 정의된다.
    - More Reserved Sectors : 미래를 위해 남겨둔 예약된 영역이다.



FAT 구조

- FAT (File Allocation Table) 영역
  - FAT 영역은 파일 할당 테이블이다. 즉, 데이터 영역에 저장된 파일들의 할당 관계를 표시해주는 테이블이다.
  - 데이터 영역에 저장된 파일은 연속적인 클러스터에 저장될 수도 있지만, 조각나 저장될 수도 있다. FAT영역은 조각난 클러스터의 연결 상태까지 표현할 수 있다.
  - 4byte크기의 'FAT entry'가 해당 번호의 클러스터에 1대1로 대응한다.
  - 0번, 1번은 Media type, Partition status를 나타내므로 대응하는 entry는 없다.
    그러므로 실제 데이터는 Cluster 2부터 존재한다.



Media type	Partition status	Cluster2	Cluster3
Cluster4	Cluster5	Cluster6	Cluster7
Cluster8	Cluster9	Cluster10	Cluster11
Cluster12	Cluster13	Cluster14	Cluster15
.....			

FAT	설명
0x00	해당 클러스터가 사용이 가능하다는 의미로 비할당 상태를 나타낸다.
0xff8	파일의 끝(End-of-Marker)를 나타낸다. 즉, 파일의 마지막 클러스터를 나타낸다.
oxff7	Bad Cluster를 나타낸다. 따라서, 이곳에는 저장을 할 수 없음을 나타낸다.

FAT 구조

- FAT Cluster 연결 방식
  - FAT 영역은 파일이나 디렉토리가 데이터 영역 내에 저장된 위치를 단일 연결 리스트로 표현한다. 각각의 FAT Entry들은 자신의 다음 클러스터 값을 담게 된다.
  - 0FFFFFFF는 해당 파일이나 디렉토리의 마지막 클러스터를 의미한다.
  - 실제로는 리틀 엔디언으로 저장되고 아래는 편의를 위해 빅 엔디언으로 표현했다.

XXXXXXXX	XXXXXXXX	00000009	00000004
00000005	00000007	00000000	00000008
0FFFFFFF	0000000A	0000000B	00000011
0000000D	0000000E	0FFFFFFF	00000010
00000012	0FFFFFFF	00000013	00000014
00000015	00000016	0FFFFFFF	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Root Directory :
2 → 9 → A → B → 11

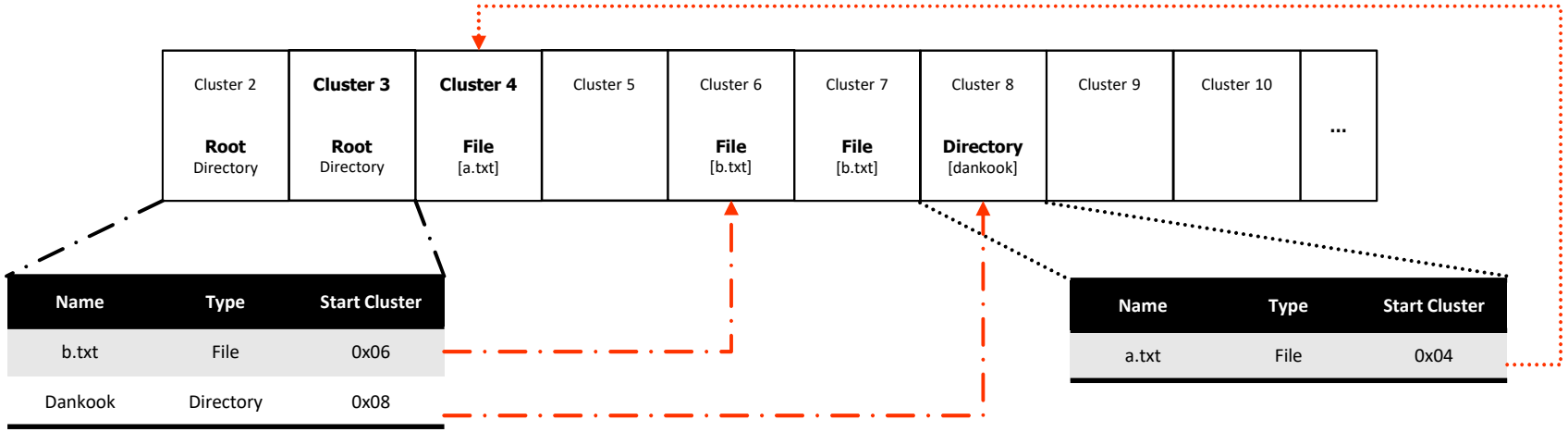
File #1 :
3 → 4 → 5 → 7 → 8

File #2 :
C → D → F

File #3 :
F → 10 → 12 → 13 → 14 → 15 → 16

FAT 구조

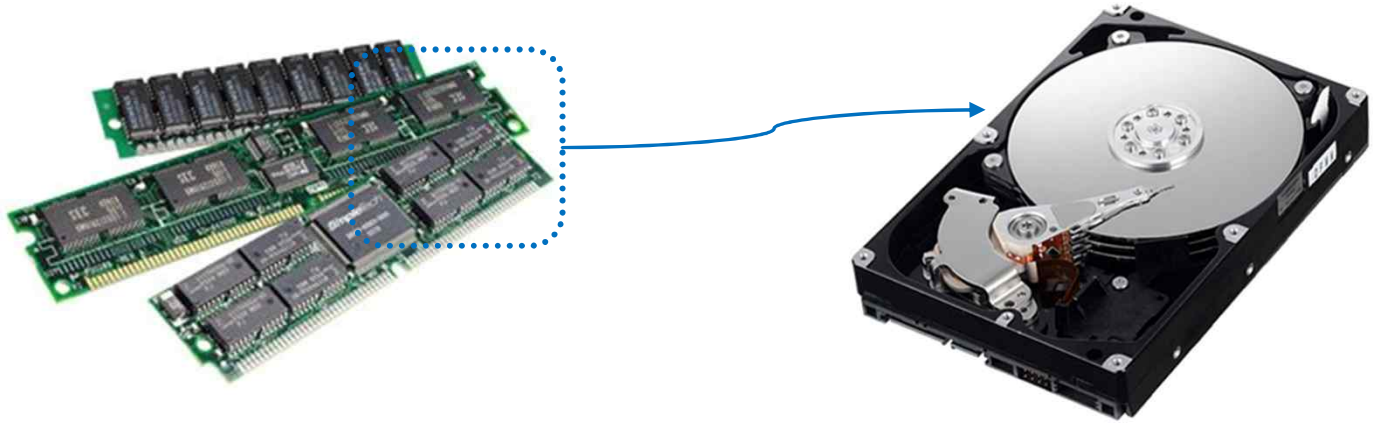
- Data 영역
  - 데이터들이 저장되고 관리되는 영역이다.
  - 데이터 영역은 논리적인 분할 단위인 클러스터로 접근된다. 파일 크기가 클러스터 크기보다 크다면 여러 클러스터에 나누어서 담을 수 있지만, 하나의 클러스터에 여러 개의 파일을 담을 수는 없다.
  - 데이터 영역에는 파일과 디렉토리 형식으로 2가지 형태의 데이터가 존재한다.
  - 디렉토리는 Directory Entry 구조체들의 집합이다. 이는 자신에게 속한 파일이나 하위 디렉토리 정보를 담고있다.
  - 파일은 파일의 내용을 담고있다.



FAT 분석 환경 세팅

FAT 분석 환경 세팅

- RAM Disk
  - RAM Disk는 주 기억 장치 활용 저장법이 아닌 RAM(DRAM)을 이용하여 디스크 드라이브를 구현하는 방식
  - 본 강의에서는 실제 Storage를 사용하지 않고 Ram Disk를 생성하여 이를 Storage로 활용한다.
  - 실습 자료로 공유된 이미지에는 쉽게 설치할 수 있도록 Ram Disk 설치 파일이 준비되어 있다.



FAT 분석 환경 세팅

```
choigunhee@choigunhee-server-93:~/2019_DKU_OS/lab3_filesystem$ ls
create.sh.x  create.sh.x.c  Makefile  ramdisk.c
choigunhee@choigunhee-server-93:~/2019_DKU_OS/lab3_filesystem$ make
make -C /lib/modules/4.18.0-17-generic/build SUBDIRS=/home/choigunhee/2019_DKU_OS/lab3_filesystem V= modules
make[1]: Entering directory '/usr/src/linux-headers-4.18.0-17-generic'
  CC [M]  /home/choigunhee/2019_DKU_OS/lab3_filesystem/ramdisk.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/choigunhee/2019_DKU_OS/lab3_filesystem/ramdisk.mod.o
  LD [M]  /home/choigunhee/2019_DKU_OS/lab3_filesystem/ramdisk.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.18.0-17-generic'
choigunhee@choigunhee-server-93:~/2019_DKU_OS/lab3_filesystem$ ls
create.sh.x  create.sh.x.c  Makefile  modules.order  Module.symvers  ramdisk.c  ramdisk.ko  ramdisk.mod.c  ramdisk.mod.o  ramdisk.o
choigunhee@choigunhee-server-93:~/2019_DKU_OS/lab3_filesystem$ mkdir mnt
choigunhee@choigunhee-server-93:~/2019_DKU_OS/lab3_filesystem$ ls
create.sh.x  create.sh.x.c  Makefile  mnt  modules.order  Module.symvers  ramdisk.c  ramdisk.ko  ramdisk.mod.c  ramdisk.mod.o  ramdisk.o
```

위 순서대로 진행하면
ramdisk.ko 와 디렉토리 mnt가 생성되는 것을 확인 할 수 있다.

FAT 분석 환경 세팅

```
choigunhee@choigunhee-server-93:~/2019_DKU_OS/lab3_filesystem$ sudo su
root@choigunhee-server-93:/home/choigunhee/2019_DKU_OS/lab3_filesystem# insmod ramdisk.ko
root@choigunhee-server-93:/home/choigunhee/2019_DKU_OS/lab3_filesystem# lsmod | grep ramdisk
ramdisk                16384  0
root@choigunhee-server-93:/home/choigunhee/2019_DKU_OS/lab3_filesystem# mkfs.fat -F 32 /dev/ramdisk
mkfs.fat 4.1 (2017-01-24)
root@choigunhee-server-93:/home/choigunhee/2019_DKU_OS/lab3_filesystem# mount /dev/ramdisk ./mnt/
root@choigunhee-server-93:/home/choigunhee/2019_DKU_OS/lab3_filesystem# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.8G     0  7.8G   0% /dev
tmpfs           1.6G  1.8M  1.6G   1% /run
/dev/sda2       234G   54G  168G  25% /
tmpfs           7.8G     0  7.8G   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/loop0      4.2M  4.2M     0 100% /snap/gnome-calculator/406
/dev/loop2      1.0M  1.0M     0 100% /snap/gnome-logs/57
/dev/loop1       15M   15M     0 100% /snap/gnome-characters/258
/dev/loop3      4.2M  4.2M     0 100% /snap/gnome-calculator/352
/dev/loop5      3.8M  3.8M     0 100% /snap/gnome-system-monitor/77
/dev/loop6       90M   90M     0 100% /snap/core/6673
/dev/loop4      152M  152M     0 100% /snap/gnome-3-28-1804/40
/dev/loop7      141M  141M     0 100% /snap/gnome-3-26-1604/82
/dev/loop9       54M   54M     0 100% /snap/core18/782
/dev/loop8      152M  152M     0 100% /snap/gnome-3-28-1804/36
/dev/loop10      15M   15M     0 100% /snap/gnome-characters/206
/dev/loop11     152M  152M     0 100% /snap/gnome-3-28-1804/31
/dev/loop12     1.0M  1.0M     0 100% /snap/gnome-logs/61
/dev/loop13     2.3M  2.3M     0 100% /snap/gnome-calculator/260
/dev/loop16      15M   15M     0 100% /snap/gnome-logs/45
/dev/loop15      36M   36M     0 100% /snap/gtk-common-themes/1198
/dev/loop14      90M   90M     0 100% /snap/core/6818
/dev/loop17      92M   92M     0 100% /snap/core/6531
/dev/loop19      15M   15M     0 100% /snap/gnome-characters/254
/dev/sda1       511M  6.1M  505M   2% /boot/efi
/dev/loop18     3.8M  3.8M     0 100% /snap/gnome-system-monitor/81
/dev/loop20      35M   35M     0 100% /snap/gtk-common-themes/818
/dev/loop21     141M  141M     0 100% /snap/gnome-3-26-1604/74
/dev/loop22      54M   54M     0 100% /snap/core18/941
/dev/loop23     3.8M  3.8M     0 100% /snap/gnome-system-monitor/70
tmpfs           1.6G   36K  1.6G   1% /run/user/1000
/dev/ramdisk   1022M  4.0K 1022M   1% /home/choigunhee/2019_DKU_OS/lab3_filesystem/mnt
root@choigunhee-server-93:/home/choigunhee/2019_DKU_OS/lab3_filesystem#
```

관리자 모드로 변경 후
ramdisk.ko 모듈을 올리고
FAT으로 포맷, mount한다.

Mount는
반드시 ./mnt에 수행한다.

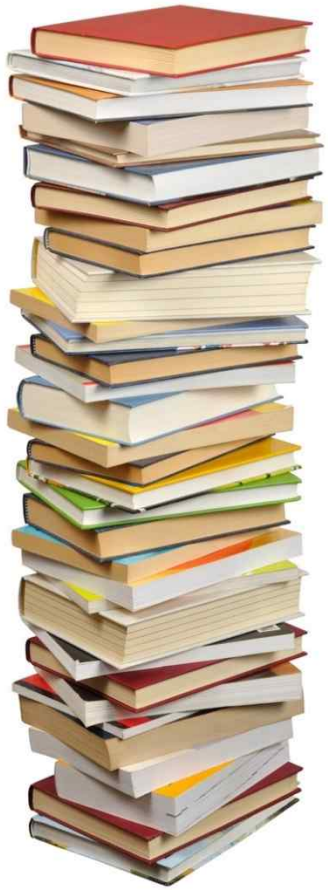
FAT 분석 환경 세팅

```
root@choigunhee-server-93:/home/choigunhee/2019_DKU_OS/lab3_filesystem# ./create.sh
-----------------------------------------
Hi
This workload has a very long execution time
It will run in 5 seconds
-----------------------------------------
```

제공된 스크립트를 실행하면
환경세팅이 완료된다.

스크립트는 매우 오랫동안 실행이 되는 것을 미리 알려드립니다.

Bonus 과제



Bonus 과제

Bonus과제는 반드시 제공되는 이미지를 사용해야합니다.→kernel camp 2018

- RAM Disk
  - cd /home/kerenlcamp-fs-2018/lecture/Kernelcamp2018/ramdisk

```
root@kernelcampfs2018-VirtualBox:/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk# ls -l
total 12
-rw-rw-r-- 1 kernelcamp-fs-2018 kernelcamp-fs-2018  325  8월  1 19:36 Makefile
-rw-rw-r-- 1 kernelcamp-fs-2018 kernelcamp-fs-2018 4756  8월  1 19:36 ramdisk.c
```

  - 준비된 ramdisk는 따로 수정하여 make 명령어를 통하여 설치가 가능하다.
    기본적으로 1GB로 설정되어 있다.

  - make

```
root@kernelcampfs2018-VirtualBox:/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk# make
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'
root@kernelcampfs2018-VirtualBox:/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk# ls
Makefile  modules.order  Module.symvers  ramdisk.c  ramdisk.ko  ramdisk.mod.c  ramdisk.mod.o  ramdisk.o
root@kernelcampfs2018-VirtualBox:/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk#
```

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

```
root@kernelcampfs2018-VirtualBox:/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk# lsmod | grep ra
mdisk
ramdisk                16384  0
root@kernelcampfs2018-VirtualBox:/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/ramdisk#
```

Bonus 과제

Bonus과제는 반드시 제공되는 이미지를 사용해야합니다.→kernel camp 2018

- Fat 모듈 컴파일 및 적재
  - fat_practice 폴더로 이동한다.
  - cd ..
  - cd fat_practice
  - make
  - 아래 명령어로 모듈 적재
  - insmod fat_kernel_camp.ko
  - insmod vfat_lecture.ko
  - 아래 명령어로 모듈 적재 확인
  - lsmod | grep fat

```
root@kernelcampfs2018-VirtualBox:/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice# lsmod | gr
ep fat
vfat_lecture           20480  0
fat_kernel_camp        65536  0
```

```
cache.c
dir.c
fatent.c
fat_kernel_camp.h
file.c
inode.c
Kconfig
Makefile
misc.c
modules.order
Module.symvers
namei_msdos.c
namei_vfat.c
nfs.c
```

-> make

```
dir.c
dir.o
.dir.o.cmd
fatent.c
fatent.o
.fatent.o.cmd
fat_kernel_camp.h
fat_kernel_camp.ko
.fat_kernel_camp.ko.cmd
fat_kernel_camp.mod.c
fat_kernel_camp.mod.o
.fat_kernel_camp.mod.o.cmd
fat_kernel_camp.o
.fat_kernel_camp.o.cmd
file.c
file.o
.file.o.cmd
inode.c
inode.o
.inode.o.cmd
Kconfig
Makefile
misc.c
misc.o
.misc.o.cmd
modules.order
Module.symvers
namei_msdos.c
namei_vfat.c
namei_vfat.o
.namei_vfat.o.cmd
nfs.c
nfs.o
.nfs.o.cmd
vfat_lecture.mod.o
.vfat_lecture.mod.o.cmd
vfat_lecture.o
.vfat_lecture.o.cmd
```

여기 소스코드를 수정하여 과제를 수행하시면 됩니다.

Bonus 과제

Bonus과제는 반드시 제공되는 이미지를 사용해야합니다.→kernel camp 2018

- FAT Format and Mount
  - mfks를 이용하여 Ramdisk Format
  - mkfs.vfat /dev/ramdisk
  - Mkfs.fat과 mkfs.vfat이 있는데 숨김 파일 및 긴 파일 이름이 지원되는 것을 제외하면 동일함

```
root@kernelcampfs2018-VirtualBox:/home/kernelcamp-fs-2018/lecture/Kernelcamp2018/fat_practice# mkfs.fat /
dev/ramdisk
mkfs.fat 3.0.28 (2015-05-16)
```

  - Ramdisk를 마운트한다. 목적지는 /mnt, 파일시스템 타입은 vfat_lecture
  - mount –t vfat_lecture /dev/ramdisk /mnt

  - 아래 명령어를 활용하여 마운트 된 ramdisk를 확인한다.
  - df –h

```
udev            2.3G     0  2.3G   0% /dev
tmpfs           496M  7.3M  489M   2% /run
/dev/sda1        25G   11G   13G  48% /
tmpfs           2.5G  212K  2.5G   1% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           2.5G     0  2.5G   0% /sys/fs/cgroup
tmpfs           496M   56K  496M   1% /run/user/1000
/dev/ramdisk   1022M  4.0K 1022M   1% /mnt
```