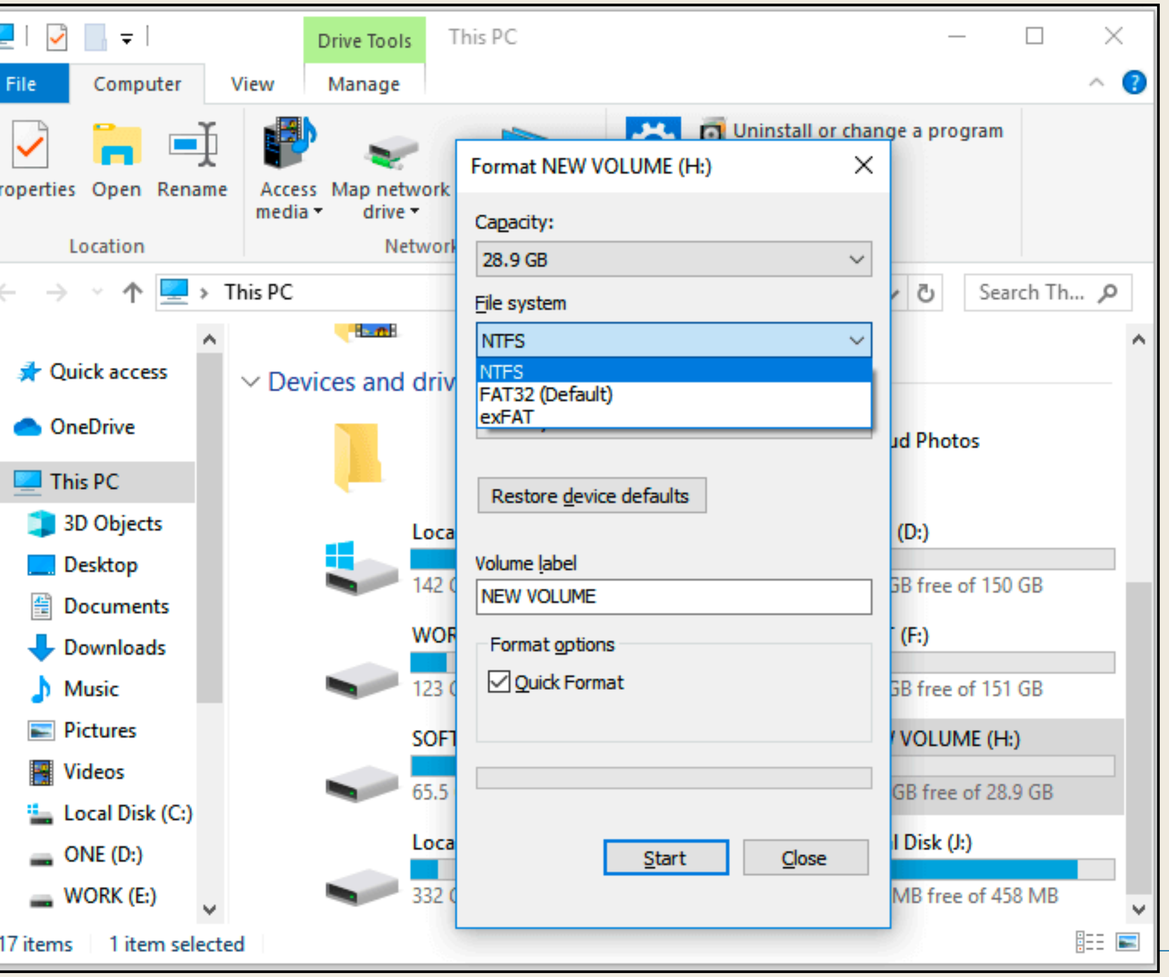
Virtual File System In Linux

2019102126 **전윤민**

Index

Virtual Again?

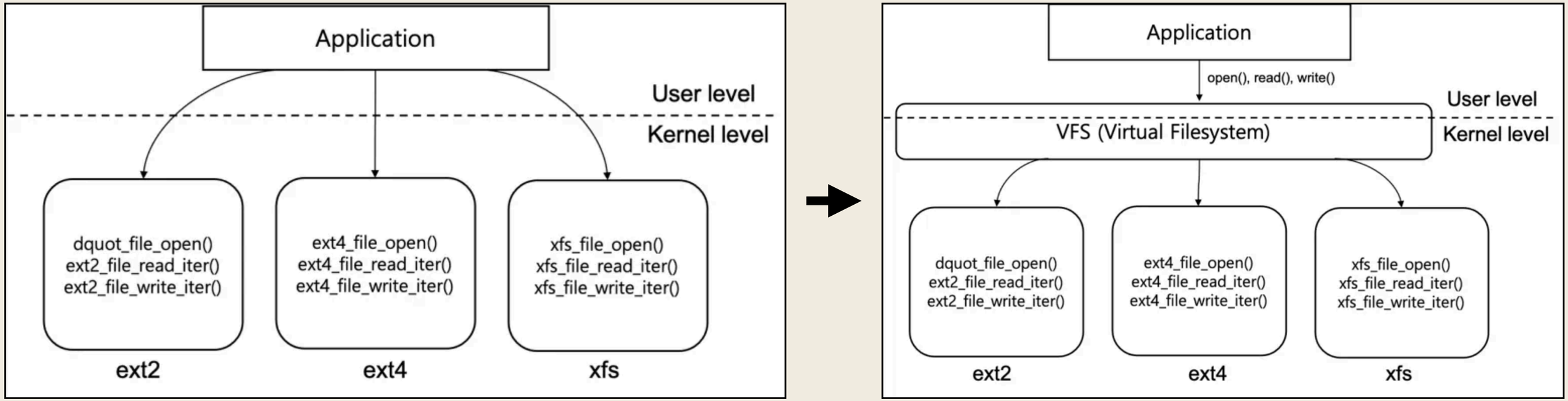
Virtual File System이 필요한 이유



- NTFS, FAT32, exFAT와 같은 **다양한 디스크 포맷** (File System)이 존재함.
- 각기 다른 File System으로 포맷된 디스크들은 **각기 다른 방식으로 조작됨.**

Virtual File System의 정의

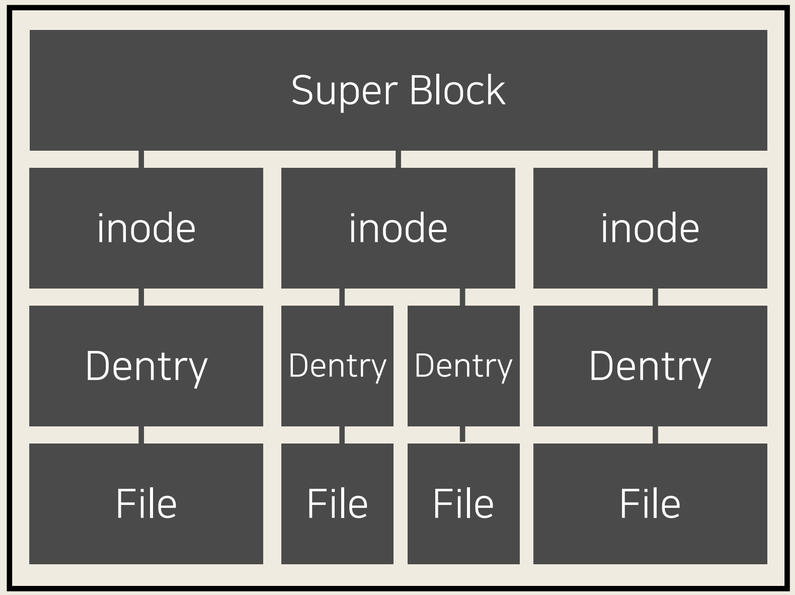
- File System 위에 존재하는 추상레이어.
- 유저 어플리케이션이 **각기 다른 File System을 통일된 방법으로 접근**할 수 있는 시스템 제공.



VFS 구현

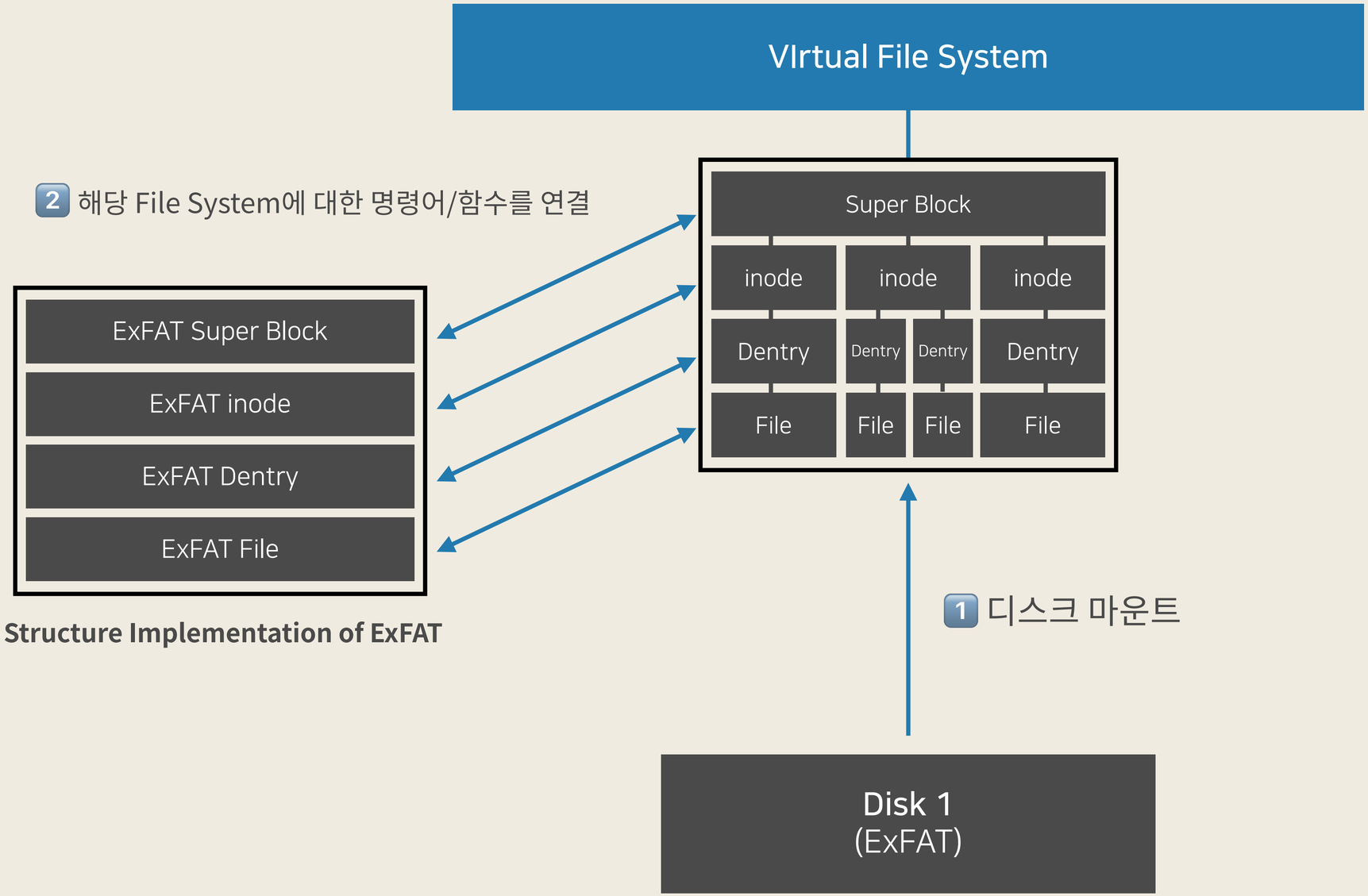
- VFS (Virtual File System)는 다양한 File System과 소통하기 위해 4가지 객체를 사용
  - **super block**
  - **inode**
  - **dentry**
  - **file**

VFS 구현

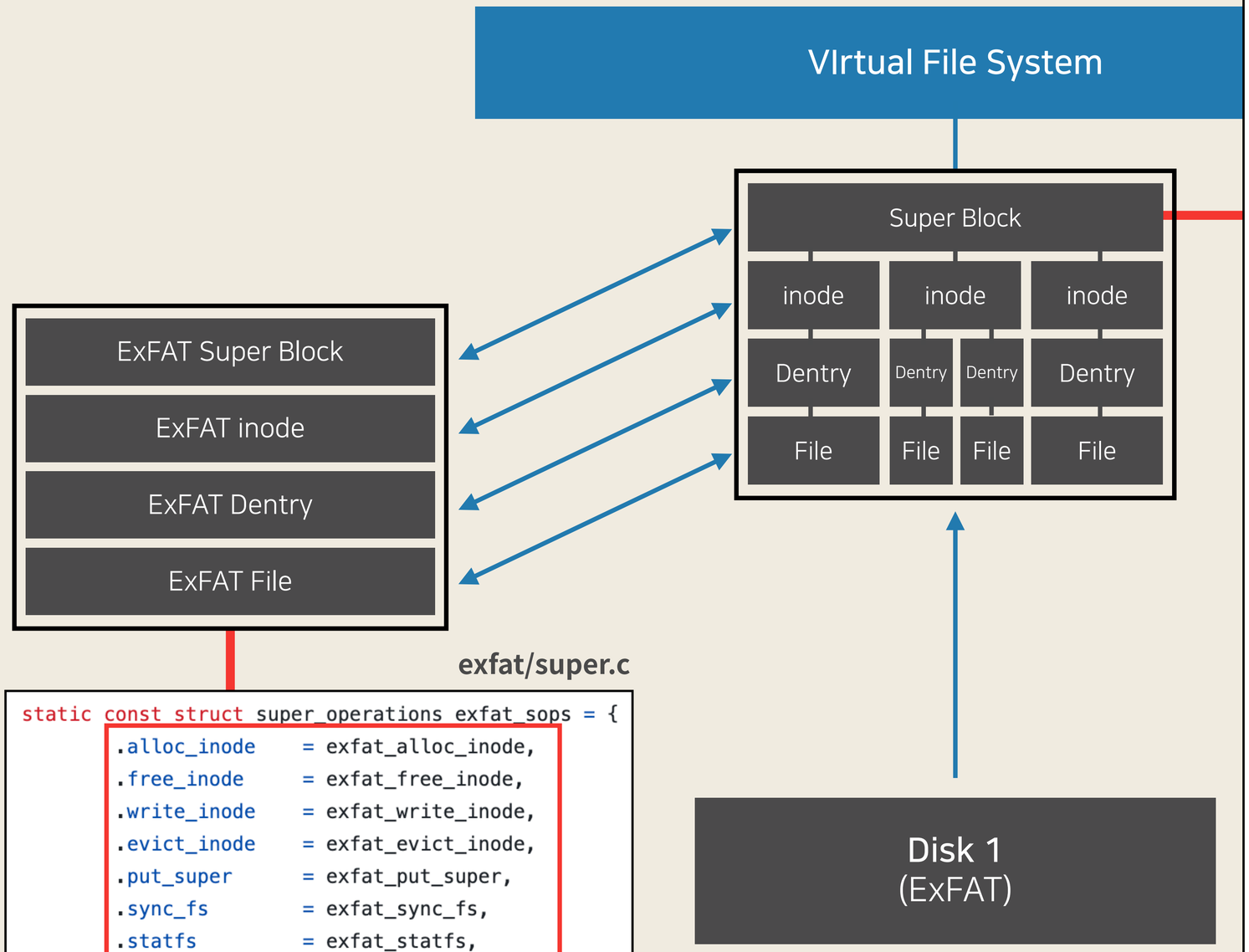


- **Super Block** : **마운트된 파일 시스템**에 대한 정보. 파일시스템당 한개.
- **Inode** : **특정 파일**에 대한 정보. 파일당 한개.
- **Dentry** : **Directory**에 대한 정보. 디렉토리 관련 작업을 처리하기 위해 부모 dentry를 가지고 있음.
- **File** : task가 알아야 하는 **Opened File**에 대한 정보. task 마다 하나씩 할당.

모든 구조체는 C언어로 작성되어 있습니다. 리눅스 소스코드에서 확인하실 수 있습니다.



Structure Implementation of ExFAT

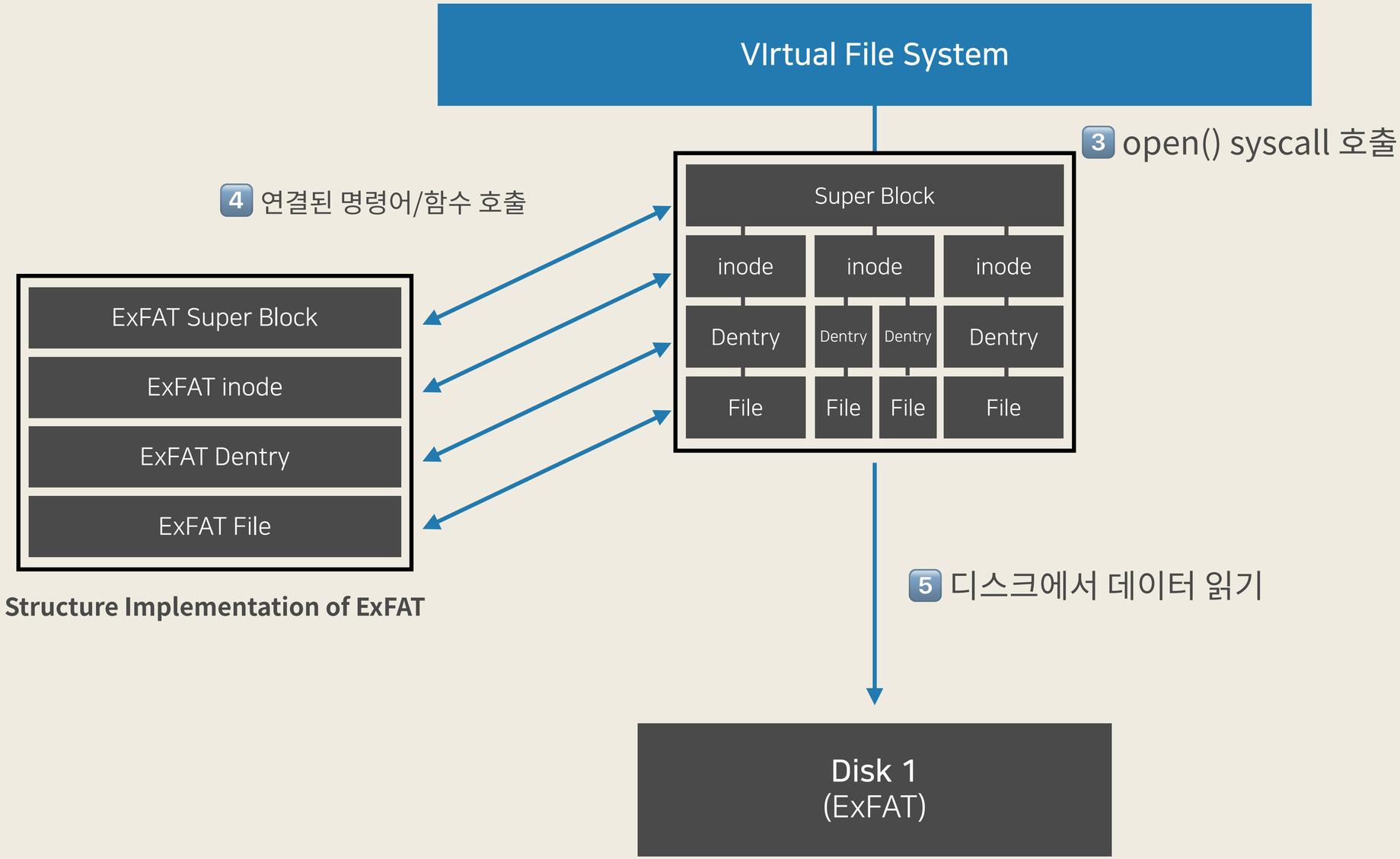


exfat/super.c

```c
static const struct super_operations exfat_sops = {
        .alloc_inode    = exfat_alloc_inode,
        .free_inode     = exfat_free_inode,
        .write_inode    = exfat_write_inode,
        .evict_inode    = exfat_evict_inode,
        .put_super      = exfat_put_super,
        .sync_fs        = exfat_sync_fs,
        .statfs         = exfat_statfs,
        .show_options   = exfat_show_options,
};
```

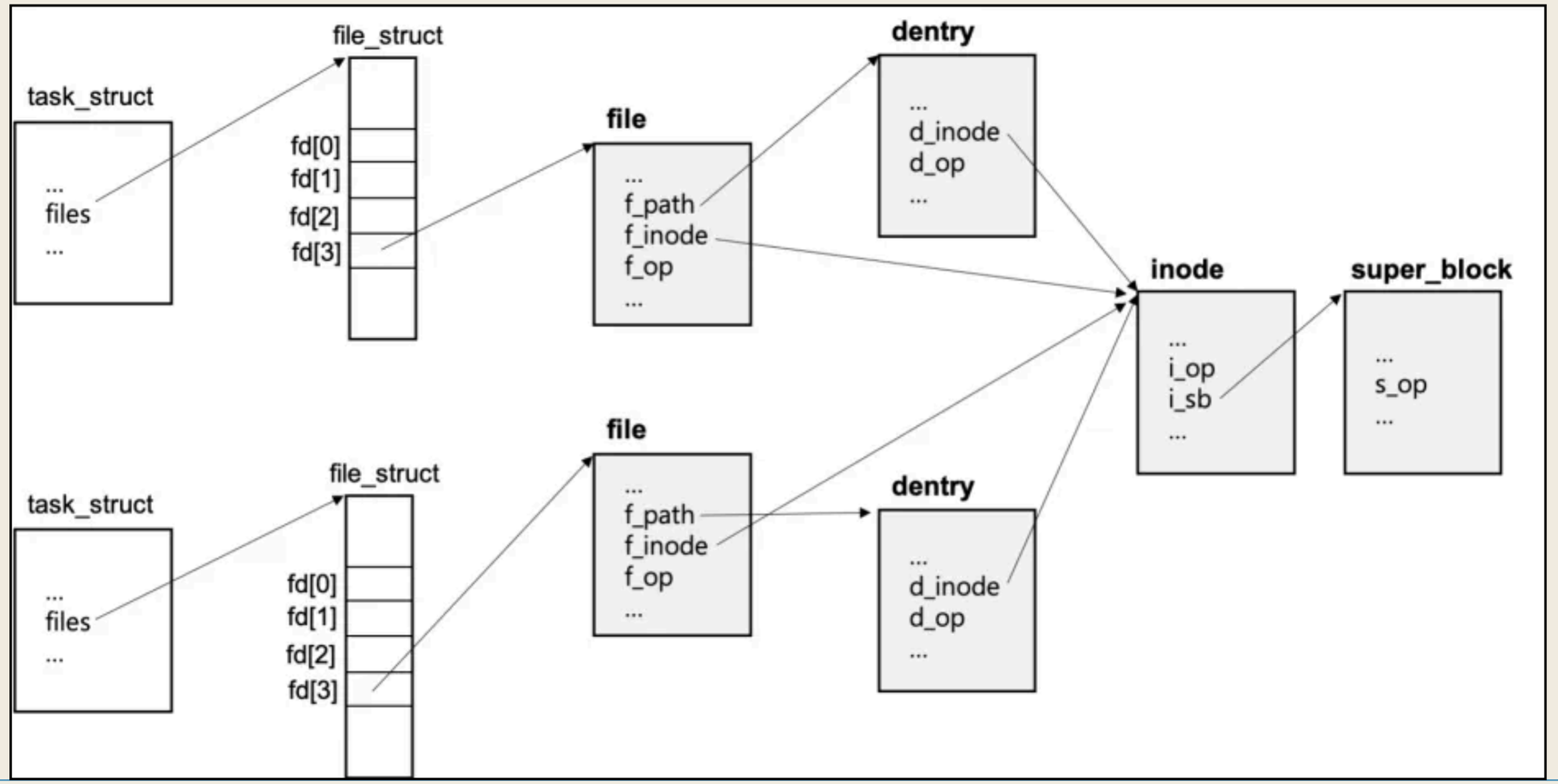
```c
struct super_operations {
        struct inode *(*alloc_inode)(struct super_block *sb);
        void (*destroy_inode)(struct inode *);
        void (*free_inode)(struct inode *);

        void (*dirty_inode) (struct inode *, int flags);
        int (*write_inode) (struct inode *, struct writeback_control *wbc
        int (*drop_inode) (struct inode *);
        void (*evict_inode) (struct inode *);
        void (*put_super) (struct super_block *);
        int (*sync_fs)(struct super_block *sb, int wait);
        int (*freeze_super) (struct super_block *, enum freeze_holder who
        int (*freeze_fs) (struct super_block *);
        int (*thaw_super) (struct super_block *, enum freeze_holder who);
        int (*unfreeze_fs) (struct super_block *);
        int (*statfs) (struct dentry *, struct kstatfs *);
        int (*remount_fs) (struct super_block *, int *, char *);
        void (*umount_begin) (struct super_block *);

        int (*show_options)(struct seq_file *, struct dentry *);
        int (*show_devname)(struct seq_file *, struct dentry *);
        int (*show_path)(struct seq_file *, struct dentry *);
        int (*show_stats)(struct seq_file *, struct dentry *);
#ifdef CONFIG_QUOTA
        ssize_t (*quota_read)(struct super_block *, int, char *, size_t,
        ssize_t (*quota_write)(struct super_block *, int, const char *, s
        struct dquot **(*get_dquots)(struct inode *);
#endif
        long (*nr_cached_objects)(struct super_block *,
                                  struct shrink_control *);
        long (*free_cached_objects)(struct super_block *,
                                    struct shrink_control *);
        void (*shutdown)(struct super_block *sb);
};
```

fs.h



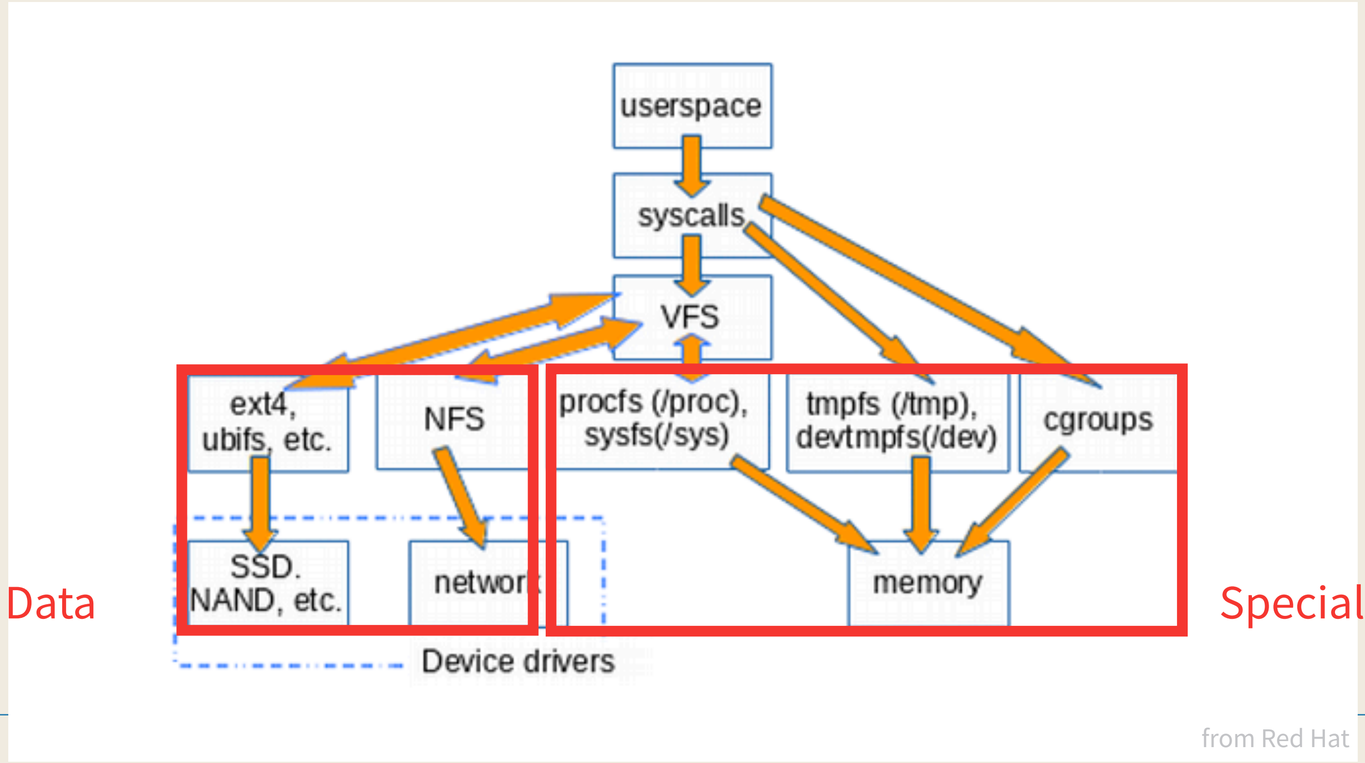
Structure Implementation of ExFAT

VFS 구현


task_struct
...
files
...
file_struct
fd[0]
fd[1]
fd[2]
fd[3]
file
...
f_path
f_inode
f_op
...
dentry
...
d_inode
d_op
...
inode
...
i_op
i_sb
...
super_block
...
s_op
...
task_struct
...
files
...
file_struct
fd[0]
fd[1]
fd[2]
fd[3]
file
...
f_path
f_inode
f_op
...
dentry
...
d_inode
d_op
...

Special File Systems

- 기기, 네트워크, 프로세스 등과 같은 **특수한 시스템 리소스를 표현**하는 데 사용되는 파일 시스템



Special File Systems

- **list command (ls)**로 리눅스의 root 조회

```
# ls /
bin   dev                          home        media  proc  sbin  tmp
boot  docker-entrypoint-initdb.d   js-yaml.js  mnt    root  srv   usr
data  etc                          lib         opt    run   sys   var
#
```

Special File Systems

- 일반적인 directory와 같은 형태로 동작.

```
# ls /proc
1             devices      kallsyms     modules       sysrq-trigger
527           diskstats    kcore        mounts        sysvipc
551           docker       key-users    mpt           thread-self
buddyinfo     driver       keys         net           timer_list
bus           execdomains  kmsg         pagetypeinfo  tty
cgroups       fb           kpagecgroup  partitions    uptime
cmdline       filesystems  kpagecount   self          version
config.gz     fs           kpageflags   slabinfo      vmallocinfo
consoles      interrupts   loadavg      softirqs      vmstat
cpuinfo       iomem        locks        stat          zoneinfo
crypto        ioports      meminfo      swaps
device-tree   irq          misc         sys
#
```

```
# ls /sys
block  class  devices   fs          kernel  power
bus    dev    firmware  hypervisor  module
#
```

```
# ls /dev
core  full    null  pts     shm     stdin   tty      zero
fd    mqueue  ptmx  random  stderr  stdout  urandom
#
```

```
# ls /run
lock  mount  systemd
#
```

Special File Systems

```
# cat /proc/cpuinfo
processor       : 0
BogoMIPS        : 48.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics
fphp asimdhp cpuid asimdrdm jscvt fcma lrcpc dcpop sha3 asimddp sha51
2 asimdfhm dit uscat ilrcpc flagm ssbs sb paca pacg dcpodp flagm2 fri
nt
CPU implementer : 0x00
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0x000
CPU revision    : 0

processor       : 1
BogoMIPS        : 48.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics
fphp asimdhp cpuid asimdrdm jscvt fcma lrcpc dcpop sha3 asimddp sha51
2 asimdfhm dit uscat ilrcpc flagm ssbs sb paca pacg dcpodp flagm2 fri
nt
CPU implementer : 0x00
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0x000
```

- 시스템 정보를 **하나의 파일처럼 접근**할 수 있음.
  (cat command로 조회)

장점

- **추상화 & 편의성 (Abstract & Convenience)**
  - 여러 파일시스템을 쉽고 통일된 방식으로 접근할 수 있다.
  - 시스템과 관련된 정보에 쉽게 접근할 수 있다. (Special File Systems)

- **유연성 & 이식성 (Flexibility & Portability)**
  - 하나의 인터페이스로 다양한 파일시스템을 지원함.
  - 개발자가 만든 프로그램이 특정 파일시스템에 종속되지 않음.

단점

- **제한된 표현력**
  - 모든 것을 파일로 표현하는 것을 때론 특정 타입의 리소스를 **과하게 간단하게(Oversimplify)** 표현해버릴 수도 있다.

- **보안 문제**
  - 민감한 정보나 시스템 제어와 관련된 메커니즘이 파일 시스템을 통해서 노출될 수도 있다.

- **항상 직관적이진 않다.**

왜 파일시스템으로 구현했을까?

“Everything Is File”

- UNIX 철학 중 하나 -

Windows와 MacOS에서의 VFS

- Windows와 MacOS 또한 가상파일시스템을 가지고 있다.
- 하지만 시스템과 관련된 정보를 리눅스처럼 파일시스템으로 관리하진 않는다. (Ex. /proc, /dev)