System Programming & OS 실습
6. File Programming
(mycreat, mycat, mycp, myls)

최민국
Dankook University
mgchoi@dankook.ac.kr

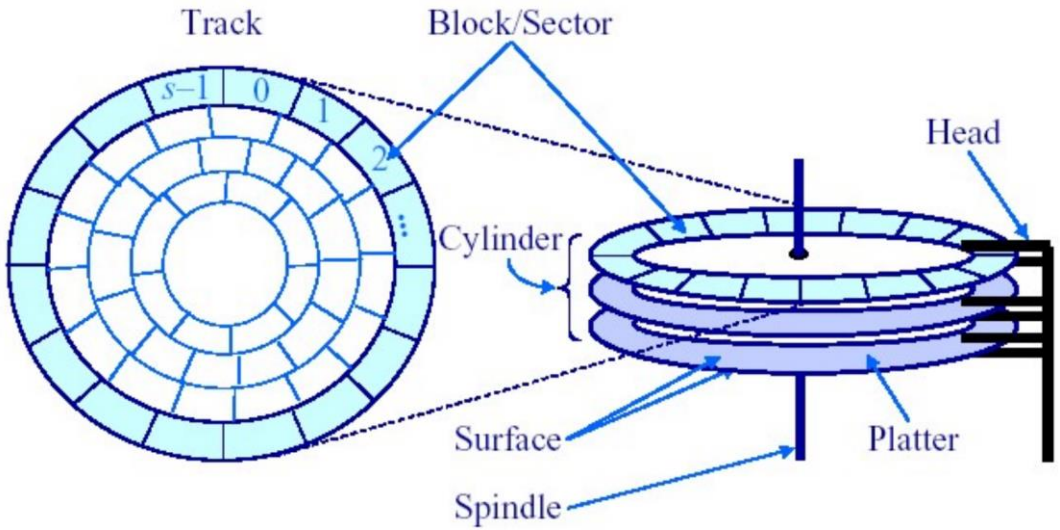
Introduction

- Issues on file
  - ✓File manipulation (create, access, remove, …)
  - ✓Manage file attributes/access control
  - ✓Associate a file name with actual data stored in disk (regular file)
  - ✓Support hierarchy structure (directory)
  - ✓Support a variety of file types (device file, pipe, socket, …)

- File related system calls
  - ✓open(), creat(): create a file, start accessing a file (authentication)
  - ✓read(), write(): read/write bytes from/to a file
  - ✓close(): finish accessing a file
  - ✓lseek(): jump to a particular offset (location) in a file
  - ✓unlink(), remove(): delete a file
  - ✓stat(), fstat(): return information about a file

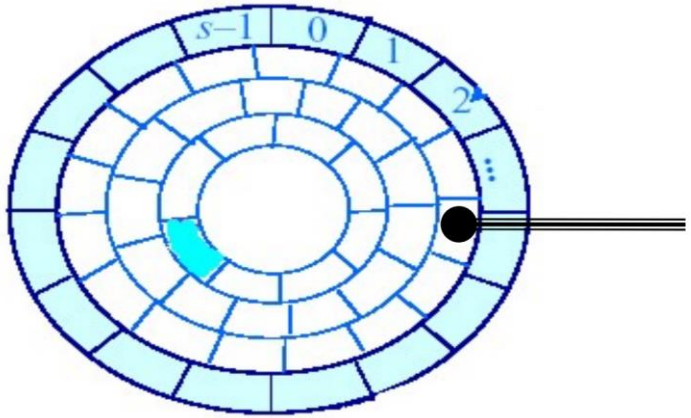
Disk structure

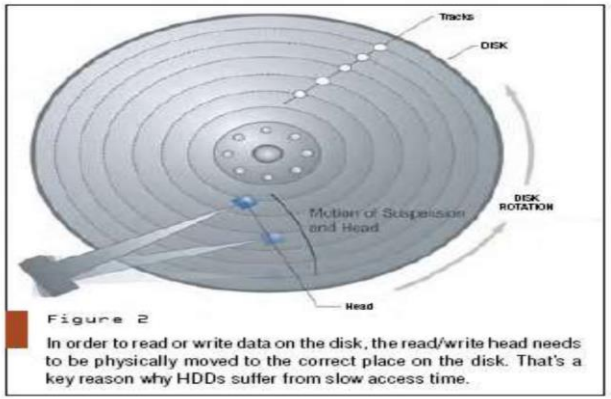
- Components
  - Physical Components
    - **Platter**: A circular disk where data is magnetically stored.
    - **Spindle**: Rotates the platters at a constant speed.
    - **Surface**: The top and bottom side of a platter where data is written.
  - Data Organization
    - **Track**: A circular path on the platter where data is recorded.
    - **Sector**: The smallest unit of storage on a disk, usually 512 bytes.
    - **Cylinder**: A set of tracks at the same position across multiple platters.
  - Read/Write Mechanism
    - **Head**: Reads and writes data on the platter.
    - **ARM (Actuator Arm):** Moves the head to the correct track position.



Disk structure

- Disk access
  - Sector addressing
    - LBA (Logical Block Address)
    - head(surface), track(cylinder), sector
  - Access time
    - **Seek time**: move head to appropriate track
    - **Rotational latency**: wait for the sector to appear under the head
    - **Transmission time**: read/write the request sector(s)
  - Try to reduce the Seek time and Rotational latency
    - Make use of various disk scheduling (eg. SCAN or elevator algorithm) and Parallel access techniques (RAID)
- Data Service Time?
  - HDD: 7200rpm (= 7200 rotation per minute)
    - Rotation per sec?
    - Average rotation time?

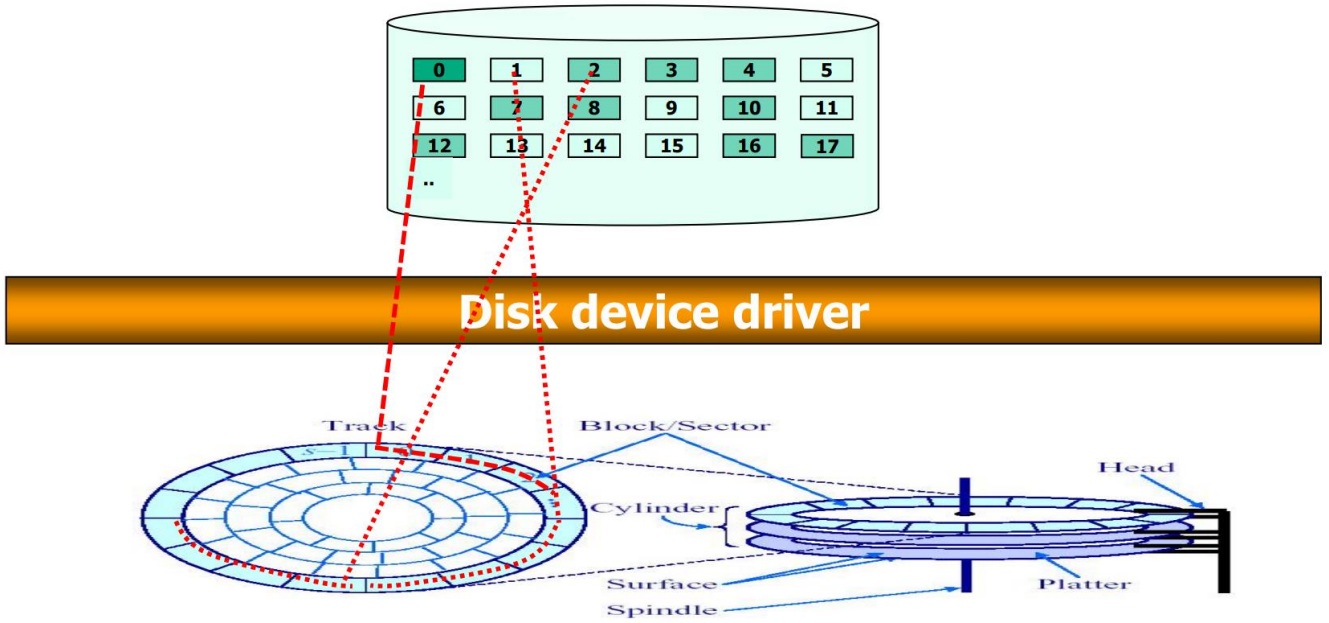




https://www.youtube.com/watch?v=ojGvHDjHPb4&ab_channel=Brian

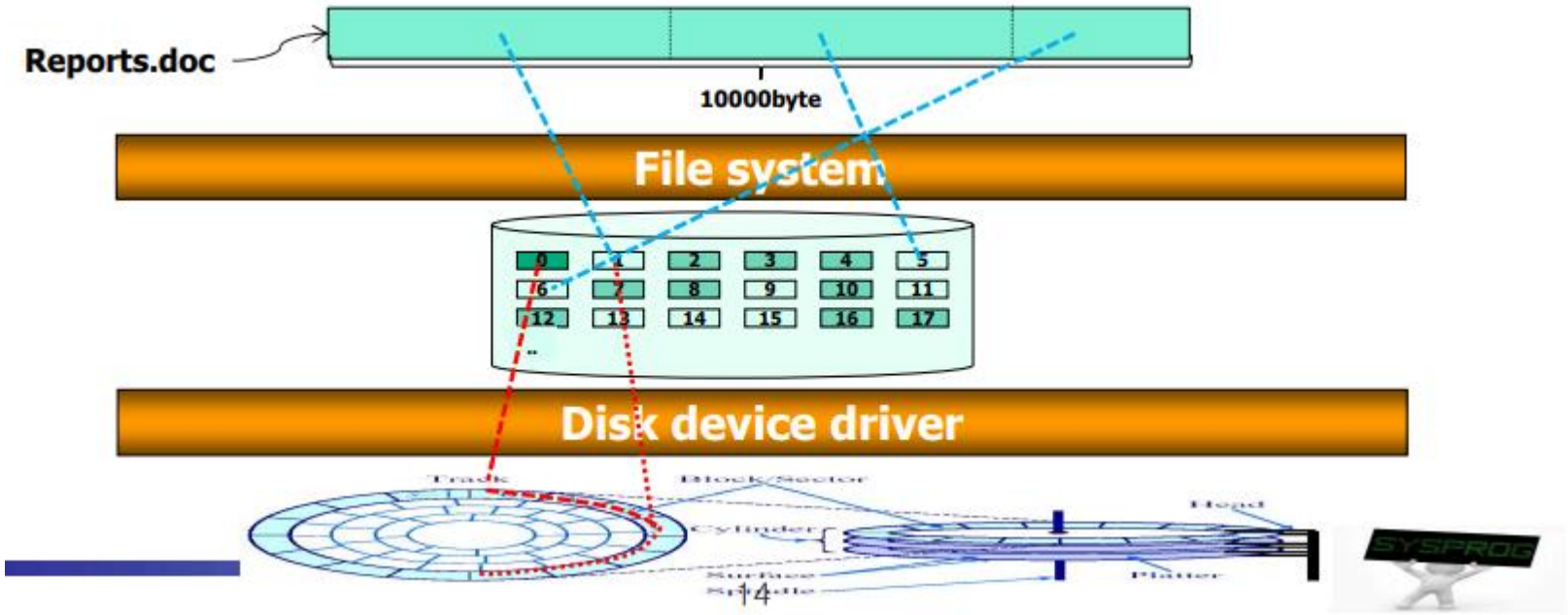
System programs for Disk

- Disk device driver
  - Abstract disk as a logical disk (a collection of disk blocks)
    - The size of a disk block is the same as that of page frame (4 or 8KB)
  - Disk command handling (ATA command: type, start, size, device, …)
  - Disk initialization, scheduling, error handling, …



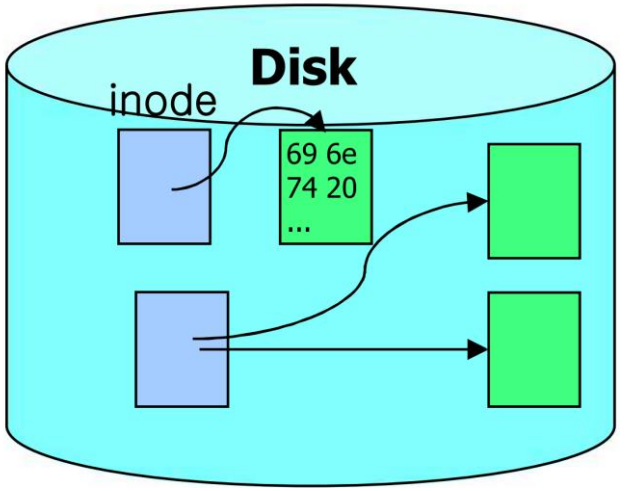
System programs for Disk

- File system
  - Support file abstraction: stream of bytes
  - Associate a file with disk blocks (inode, FAT)
  - Support file attribute/access control, directory, …



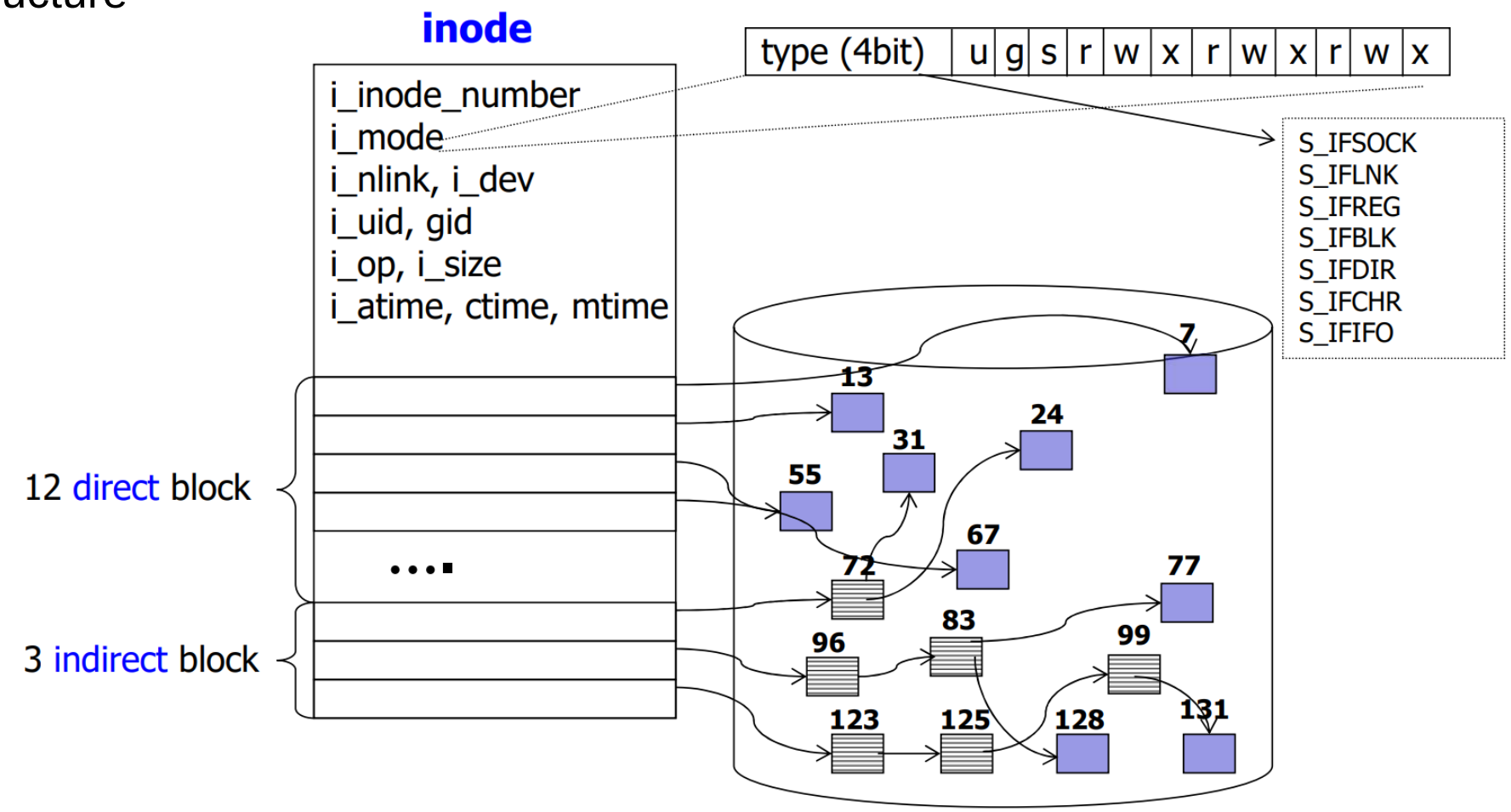
System programs for Disk

- File system
  - inode concept
  - An object for managing a file in a file system (metadata)
  - Used by various file systems such as UFS, FFS, Ext2/3/4, LFS, …
  - Maintain information for a file (e.g. “ls –l”)
    - file size
    - locations of disk blocks for a file
    - file owner, access permission
    - time information
    - file type: regular, directory, device, pipe, socket, …
  - Stored in disk
  - Constructed when a file is created



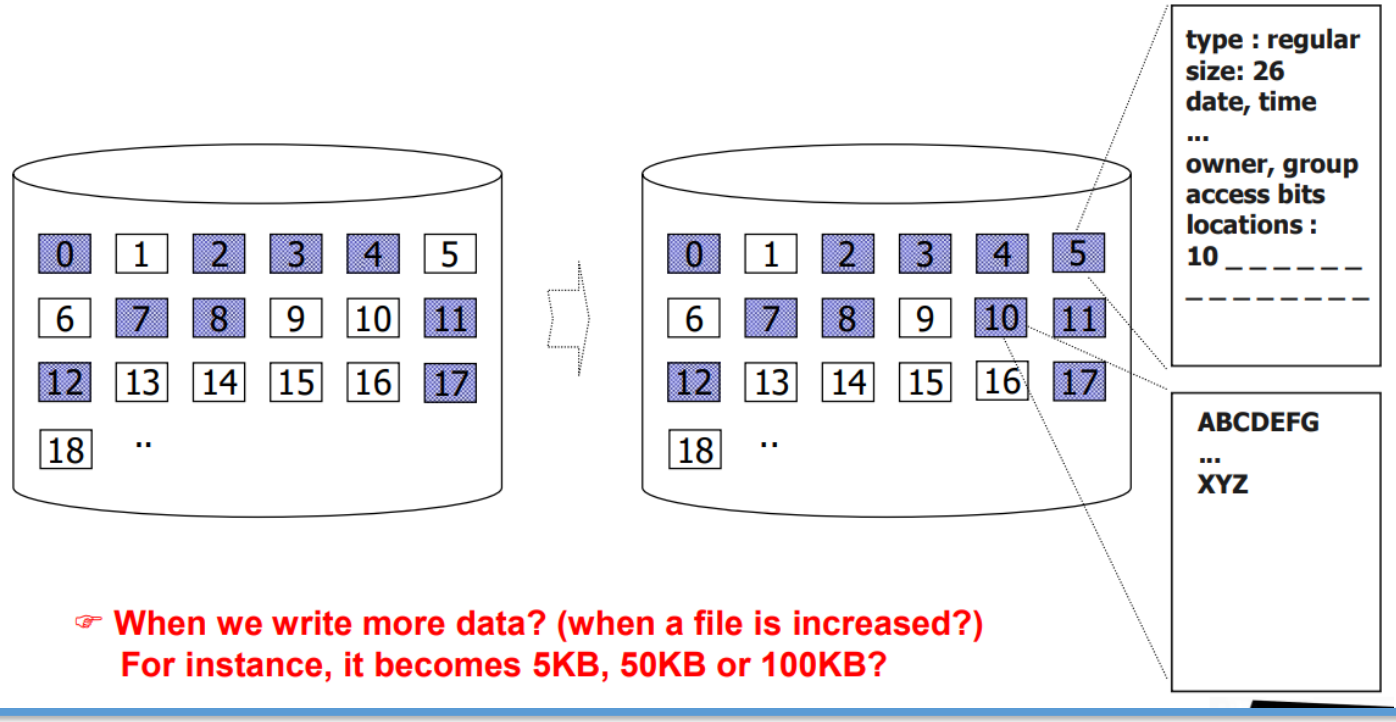
System programs for Disk

- File system
  - inode structure



System programs for Disk

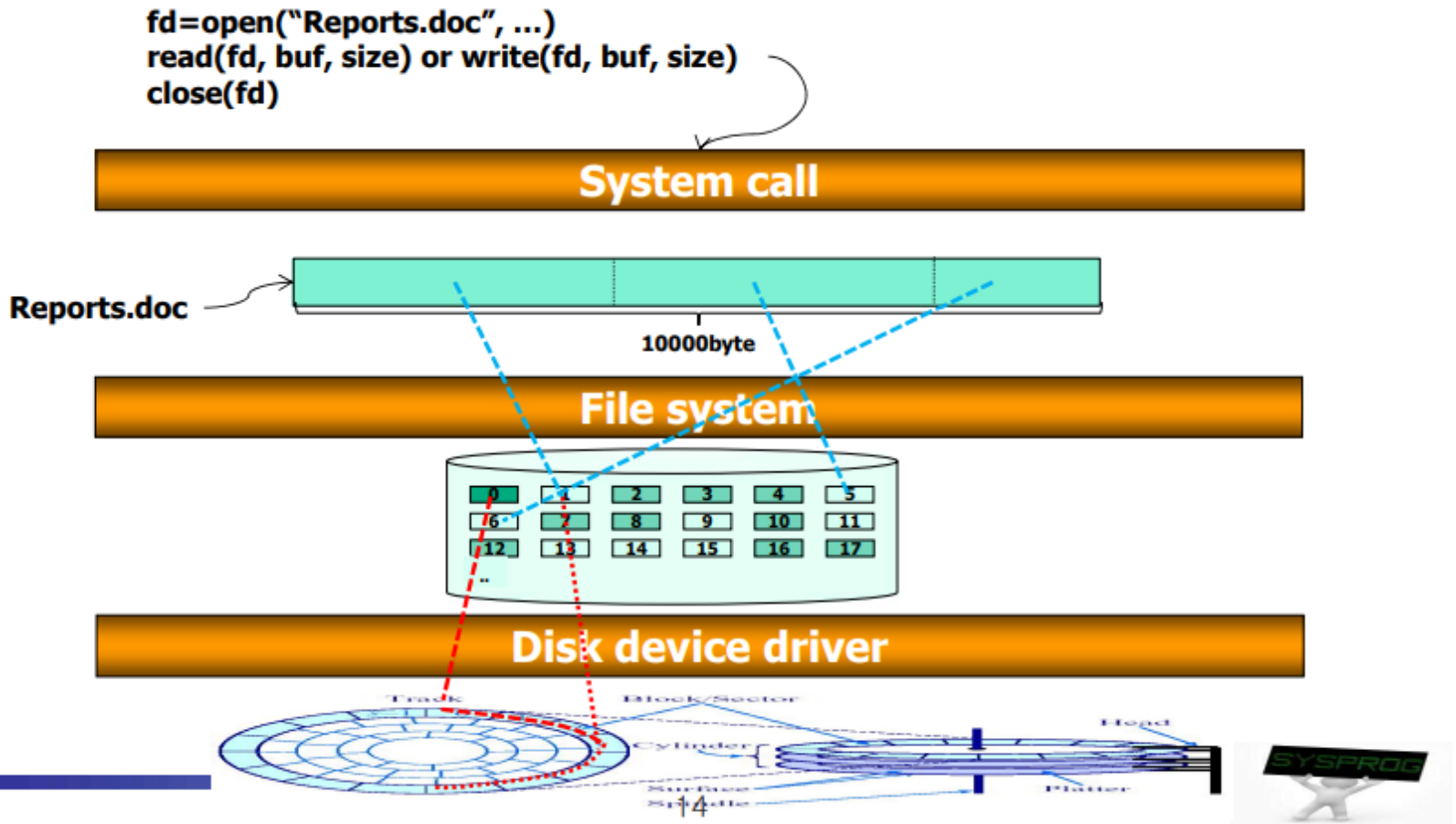
- File system
  - inode example
    - When we create a new file, named “alphabet.txt”, whose contents include “AB…Z”.
      - Note that, in actuality, the inode size is much smaller than the disk block size (128B or 256B)



☞ When we write more data? (when a file is increased?)
For instance, it becomes 5KB, 50KB or 100KB?

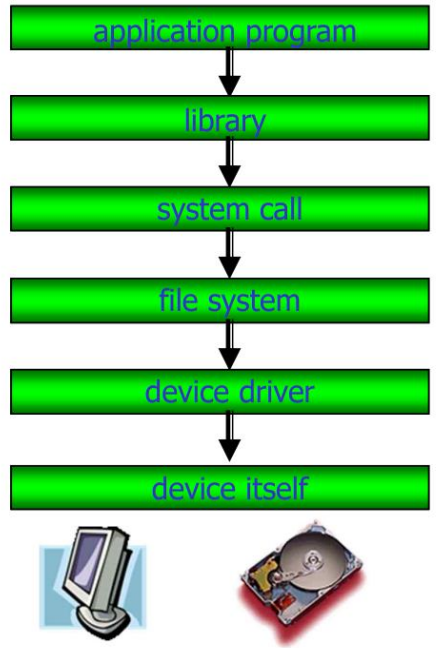
System programs for Disk

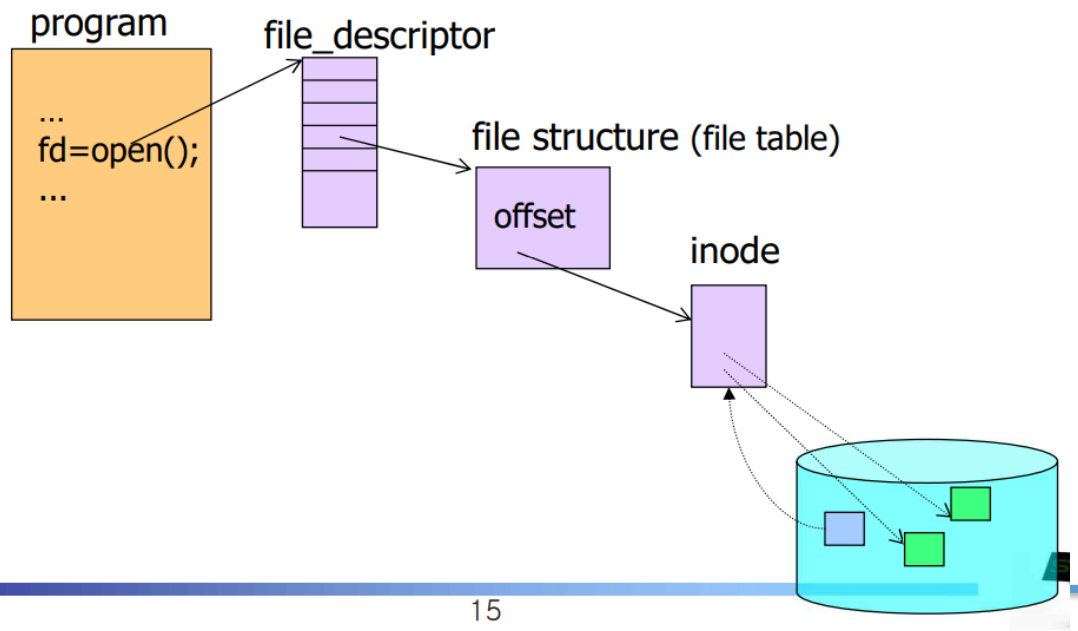
- System call
  - Support interfaces such as open(), read(), write(), close(), …



System programs for Disk

- System call
  - Use fd (file descriptor) instead of file name (for efficiency)
    - fd: object to point out a file in kernel
    - return value of the open() system call
    - used by the following read(), write(), …, close() system calls
    - fd is connected into inode through various kernel objects (file table)





실습1: open(),read()

int open(const char *pathname, int flags, [mode_t mode])

- ✓ pathname : absolute path or relative path
- ✓ flags (see: /usr/include/asm/fcntl.h or Chapter 4.3 in the LPI)
  - O_RDONLY, O_WRONLY, O_RDWR
  - O_CREAT, O_EXCL
  - O_TRUNC, O_APPEND
  - O_NONBLOCK, O_SYNC
  - …
- ✓ mode
  - meaningful with the O_CREAT flag
  - file access mode (S_IRUSR, S_IWUSR, S_IXUSR, S_IRGRP, …, S_IROTH, …)
- ✓ return value
  - file descriptor if success
  - -1 if fail

int read(int fd, char *buf, int size) // same as the write(fd, buf, size)

- ✓ fd: file descriptor (return value of open())
- ✓ buf: memory space for keeping data
- ✓ size: request size
- ✓ return value
  - read size
  - -1 if fail

실습1: open(),read()


```
[centos@localhost ~]$ vim open.c
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 5
char fname[]="alphabet.txt";

int main(){
        int fd,size;
        char buf[MAX_BUF];

        fd = open(fname,O_RDONLY);
        if(fd<0){
                printf("Can't open %sfile with errno %d\n",fname,errno);
                exit(-1);
        }
        size = read(fd,buf,MAX_BUF);
        if(size < 0){
                printf("Can't read from file %s,size= %d\n",fname,size);
        }
        else
                printf("size of read data is %d\n",size);
        close(fd);
}
```

실습1: open(),read()


```
[centos@localhost ~]$ gcc -o open open.c
[centos@localhost ~]$ ls
Desktop    Downloads  open      Pictures  Templates
Documents  Music      open.c    Public    Videos
[centos@localhost ~]$ ./open
Can't open alphabet.txtfile with errno 2
```

오류코드: 파일 및 디렉토리 X

실습1: open(),read()


```
[centos@localhost ~]$ vim alphabet.txt
```

실습1: open(),read()


```
[centos@localhost ~]$ ./open
size of read data is 5
```

실습2: write()


```
[centos@localhost ~]$ vim write.c
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 5
char fname[]="alphabet.txt";

int main(){
        int fd,read_size,write_size;
        char buf[MAX_BUF];

        fd = open(fname,O_RDONLY);
        if(fd<0){
                printf("Can't open %sfile with errno %d\n",fname,errno);
                exit(-1);
        }
        read_size = read(fd,buf,MAX_BUF);
        if(read_size < 0){
                printf("Can't read from file %s,size= %d\n",fname,write_size);
        }
        write_size = write STDOUT_FILENO,buf,MAX_BUF);
        close(fd);
}
~
~
~
"write.c" 24L, 506B                                   22,33-40      All
```

```
#define STDIN_FILENO    0   // Standard input
#define STDOUT_FILENO   1   // Standard output
#define STDERR_FILENO   2   // Standard error
```

실습2: write()


```
[centos@localhost ~]$ gcc -o write write.c
[centos@localhost ~]$ ./write
abcde[centos@localhost ~]$
```

실습3: mycat


```
[centos@localhost ~]$ vim mycat.c
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64

int main(int argc, char *argv[]){
        int fd,read_size,write_size;
        char buf[MAX_BUF];

        if(argc != 2){
                printf("USAGE: %S file_name\n",argv[0]);
                exit(-1);
        }
        fd = open(argv[1], O_RDONLY);
        if(fd<0){
                //open error handling
        }
        while(1){
                read_size=read(fd,buf,MAX_BUF);
                if(read_size == 0)
                        break;
                write_size=write(STDOUT_FILENO,buf,read_size);
        }
        close(fd);
}
```

```
"mycat.c" 27L, 489B                                          18,4-18       All
```

실습3: mycat


```
[centos@localhost ~]$ gcc -o mycat mycat.c
[centos@localhost ~]$ ./mycat alphabet.txt
abcdefgh
```

실습4: create new file


```
[centos@localhost ~]$ vim creat.c
```

```c
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64
char fname[]="newfile.txt";
char dummy_data[]="abcdefg\n";


int main(){
        int fd,read_size,write_size;
        char buf[MAX_BUF];

        fd = open(fname,O_RDWR | O_CREAT | O_EXCL, 0664);
        if(fd<0){
                printf("Can't create %s file with errno %d\n",fname,errno);
                exit(1);
        }
        write_size=write(fd,dummy_data,sizeof(dummy_data));
        printf("write_size = %d\n",write_size);
        close(fd);

        fd=open(fname,O_RDONLY);
        read_size = read(fd,buf,MAX_BUF);
        printf("read_size = %d\n",read_size);
        write_size= write(STDOUT_FILENO,buf,read_size);

        close(fd);
}
```

```
                                                          30,1          Bot
```

실습4: create new file


```
[centos@localhost ~]$ gcc -o creat creat.c
[centos@localhost ~]$ ./creat
Can't create newfile.txt file with errno 17
[centos@localhost ~]$ rm -rf newfile.txt
[centos@localhost ~]$ ./creat
write_size = 9
read_size = 9
abcdefg
```

실습5: lseek()

- ✓ Using lseek()

off_t lseek(int fd, off_t offset, int whence)

- ✓ fd : file descriptor
- ✓ offset : offset position
- ✓ whence (/usr/include/unistd.h)
  - SEEK_SET : New offset is set to offset bytes.
  - SEEK_CUR: New offset is set to its current location plus offset bytes.
  - SEEK_END: New offset is set to the size of the file plus offset bytes
- ✓ return value
  - new offset if success
  - -1 if fail

Negative value is allowed

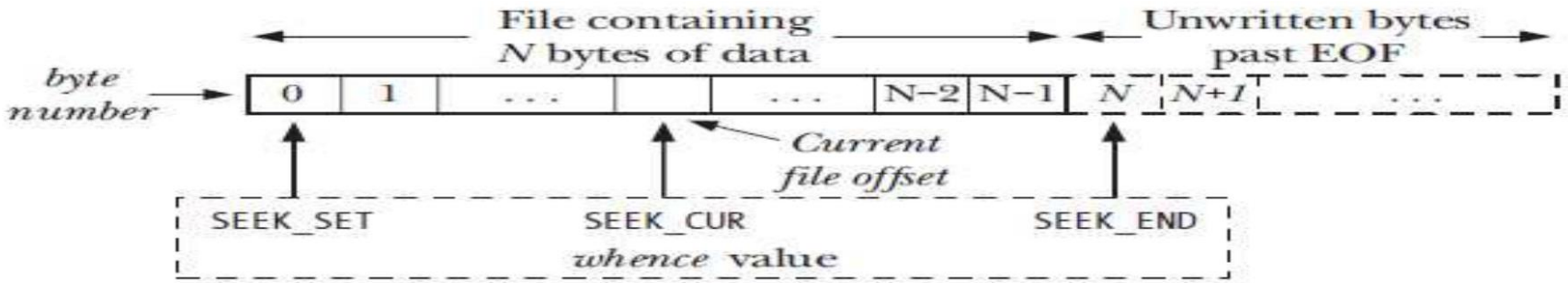


Figure 4-1: Interpreting the *whence* argument of *lseek()*

실습5: lseek()


```
[centos@localhost ~]$ vim lseek.c
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64
char fname[]="newfile.txt";
char dummy_data[]="abcdefg\n";

int main(){
        int fd,read_size,write_size,new_offset;
        char buf[MAX_BUF];

        fd = open(fname,O_RDWR | O_CREAT | O_EXCL, 0664);
        if(fd<0){
                printf("Can't create %s file with errno %d\n",fname,errno);
                exit(1);
        }
        write_size=write(fd,dummy_data,sizeof(dummy_data));
        close(fd);

        fd=open(fname,O_RDONLY);
        new_offset = lseek(fd,3,SEEK_SET);
        read_size = read(fd,buf,MAX_BUF);
        printf("read_size = %d\n",read_size);
        write_size = write(STDOUT_FILENO,buf,read_size);

        close(fd);
}
```

```
                                                              9,0-1         All
```

Read and write

O_CREAT 또는create()

파일이 존재하는지 확인

접근 권한

실습5: lseek()


```
[centos@localhost ~]$ gcc -o lseek lseek.c
[centos@localhost ~]$ ./lseek
Can't create newfile.txt file with errno 17
[centos@localhost ~]$ rm -rf newfile.txt
[centos@localhost ~]$ ./lseek
read_size = 6
defg
[centos@localhost ~]$
```

복습: system call

- Basic file operation in Linux
  - `int open(const char *pathname, int flags, [mode_t mode])`
    - Input params : file path
    - Return : (Success) ? file descriptor (int) : -1 (error)
  - `ssize_t read(int fd, void *buf, size_t count)`
    - Input params : file descriptor
    - Return : (Success) ? the number of bytes read : -1 (error)
  - `ssize_t write(int fd, void *buf, size_t count)`
    - Input params : file descriptor
    - Return : (Success) ? the number of bytes written : -1 (error)
  - `int close(int fd)`
    - Input params : file descriptor
    - Return : (Success) ? 0 : -1

실습 6: mycreat

- mycreat

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64
char fname[] ="newfile";
char dummy[] ="TABA OS 2023 [your name here]\n";
int main(int argc, char *argv[]){
      //변수 선언
      int fd, read_size, write_size =0;
      char buf[MAX_BUF];

      fd = /*    [1] fill out here using system call  */;
      write_size=write( /* [2] fill out here : write argument */ );
      //error handling
      printf("write to file %s write size %d\n", fname, write_size);
      /* [3] fill out here : */
}
```

[1] $man open

NAME top

open, openat, creat - open and possibly create a file

SYNOPSIS top

```c
#include <fcntl.h>

int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

[2] $man -s 2 write

NAME top

write - write to a file descriptor

SYNOPSIS top

```c
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);
```


[1] https://man7.org/linux/man-pages/man2/open.2.html
[2] https://man7.org/linux/man-pages/man2/write.2.html

실습 6: mycreat

- mycreat 수행 결과 화면

```
$ make clean; make
inhoinno@inhoinno:~/TABA_OS_2023/mycreat$ make clean; make
rm -f *.o
rm -f *.gch
rm -f mycreat
gcc -g    -c -o mycreat-answer.o mycreat-answer.c
gcc -o mycreat mycreat-answer.o
```

```
$ ./mycreat; cat newfile
inhoinno@inhoinno:~/TABA_OS_2023/mycreat$ ./mycreat
write to file newfile write size 23
inhoinno@inhoinno:~/TABA_OS_2023/mycreat$ cat
answer/           Makefile          mycreat           mycreat-answer.c  mycreat-answer.o  mycreat.c         newfile
inhoinno@inhoinno:~/TABA_OS_2023/mycreat$ cat newfile
TABA OS 2023 inhoinno
inhoinno@inhoinno:~/TABA_OS_2023/mycreat$
```

실습7: mycat

- mycat

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64

int main(int argc, char *argv[]){
	//변수 선언
	int fd, read_size, write_size =0;
	char buf[MAX_BUF];
	//예외 처리
	if (argc!=2){
			printf("USAGE: %s file_name\n",argv[0]);
			exit(-1);
	}
	fd = /*	[1] fill out here using system call  */;
	if (fd < 0){
			//open error handling
			perror("fd open error\n");
	}
	while((read_size = /* [2] fall through. fill out here using syscall. */) != 0){
			//printf("%s",buf);
			write_size = /* [3] fall through. fill out here using syscall. */;
	}
	/* [4] fall through. fd must be closed. */;
}
```

hint

```
[ ] $man open
[ ] $man read
[ ] $man -s 2 write
[ ] $man close
```

standard file descriptors

```c
/* Standard file descriptors.  */
#define STDIN_FILENO    0       /* Standard input.  */
#define STDOUT_FILENO   1       /* Standard output.  */
#define STDERR_FILENO   2       /* Standard error output.  */
```

https://elixir.bootlin.com/uclibc-ng/latest/source/include/unistd.h#L214

실습7: mycat

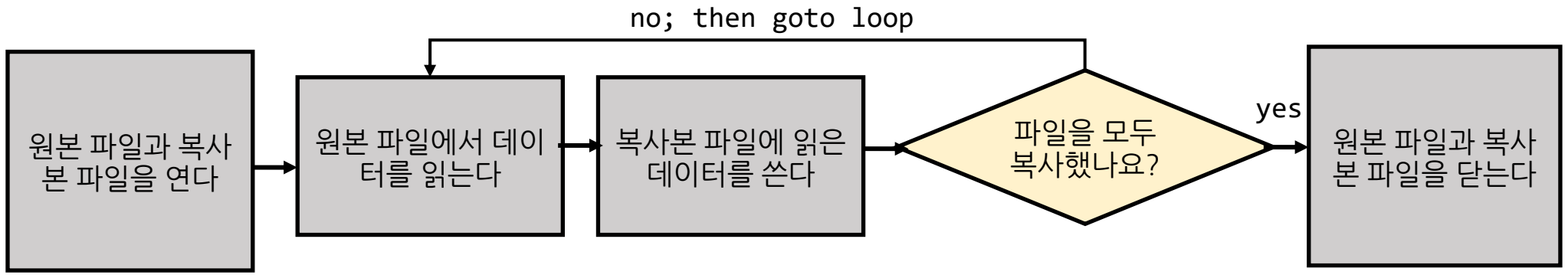
- mycat 수행 결과 화면

```
$ cp ../mycreat/newfile .
$ make
$ ./mycat newfile
```

```
inhoinno@inhoinno:~/TABA_OS_2023/mycat$ make
gcc -g    -c -o mycat.o mycat.c
gcc -o mycat mycat.o
inhoinno@inhoinno:~/TABA_OS_2023/mycat$ ./mycat newfile
TABA OS 2023 inhoinno
inhoinno@inhoinno:~/TABA_OS_2023/mycat$
```

실습8: mycp

- 프로그램 요구사항 기술 - mycp
  - Input
    - USAGE : `./mycp origin_file_here dest_file_here`
    - 내용이 적혀있는 원본파일origin file
  - Output
    - 원본 파일origin file 의 user data가 적혀있는 복사본 파일destination file
  - mycp algorithm



실습8: mycp

- mycp

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fctnl.h>
#include <errno.h>
#define MAX_BUF 64

int main(int argc, char *argv[]){
      //변수 선언
      int fd_origin, fd_dest, read_size, write_size =0;
      char buf[MAX_BUF];
      //예외 처리
      if (argc!=3){
                printf("USAGE: %s origin dest\n",argv[0]);
                exit(-1);
      }
      fd_origin = /*        [1] fill out here using system call  */;
      fd_dest = /*          [1-1] fill out here using system call*/;
      if (fd_origin < 0 || fd_dest <0){
                //open error handling
                perror("fd open error\n");
      }
      //read from the origin file
      while((read_size = /* [2] fall through. fill out here using syscall. */) != 0){
                 //write to the dest file
                write_size = /* [3] fall through. fill out here using syscall. */;
      }
      /* [4] fall through. fd must be closed. */;
}
```

hint

```
[ ] $man open
[ ] $man read
[ ] $man -s 2 write
[ ] $man close
```

실습8: mycp

- mycp 실행 화면

```
$ make
$ ls -l
```

```
inhoinno@inhoinno:~/TABA_OS_2023/mycp$ ls -l
total 20
drwxrwxr-x 2 inhoinno inhoinno 4096  3월   8 01:04 answer
-rw-rw-r-- 1 inhoinno inhoinno  806  3월   8 01:07 bak_my
-rw-rw-r-- 1 inhoinno inhoinno  219  3월   8 01:04 Makefile
-rw-rw-r-- 1 inhoinno inhoinno 1074  3월   8 01:07 mycp.c
-rw-r--r-- 1 inhoinno inhoinno   29  3월   8 00:55 origin
```

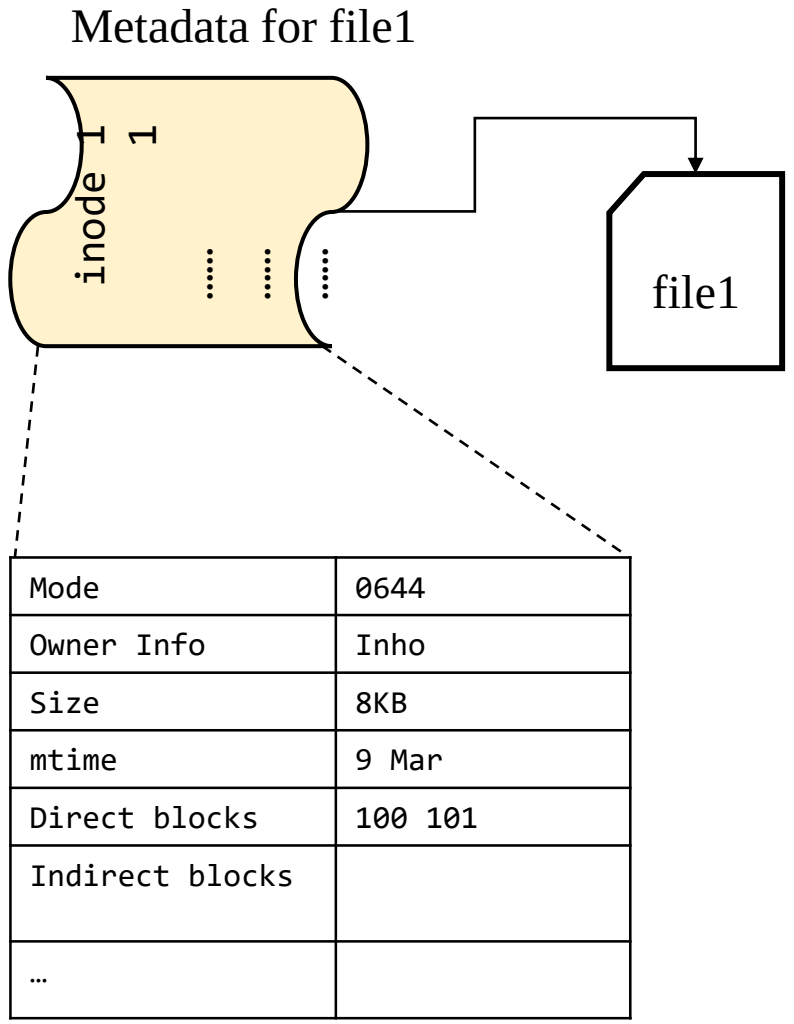
```
$ ./mycp origin dest; cat dest
```

```
inhoinno@inhoinno:~/TABA_OS_2023/mycp$ cat dest
mycpprogram
I am origin file
inhoinno@inhoinno:~/TABA_OS_2023/mycp$
```

개념설명 - 파일, 그리고 디렉토리

- 용어 정의 terminology
  - 파일 file
    1. user data
       - 실제 파일에 저장되는 데이터
    2. Metadata
       - File의 정보와 user data의 위치를 가리키는 데이터

➔ 데이터와 메타데이터의 차이는?



실습9: mycp-adv

- 프로그램 요구사항 기술 - mycp advanced
  - Input
    - USAGE : `./mycp origin_file_here dest_file_here`
    - 내용이 적혀있는 원본파일origin file
  - Output
    - 원본 파일origin file 의 user data가 적혀있는 복사본 파일destination file
    - **이때 파일의 속성 정보를 포함하여 완전 복사 (Metadata copy)**

실습9: mycp-adv

- mycp

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64

int main(int argc, char *argv[]){
      int fd_origin, fd_dest, read_size, write_size =0;
      char buf[MAX_BUF];

      struct stat* stat_origin=(struct stat*)malloc(sizeof(struct stat));
      mode_t flag_origin;

      if (argc!=3){
                  printf("USAGE: %s origin dest\n",argv[0]);
                  exit(-1);   }
      fd_orgin    = open(argv[1], O_RDONLY);

      /* [1] fall through. get file attribute structure from fd_origin */
      flag_origin = stat_origin->/* [2] fall through. let's get member from struct stat "stat_origin->field_here" */;

      fd_dest     = open(argv[2], O_RDWR|O_CREAT|O_EXCL|O_SYNC, flag_origin);
      //open error handling
      //read from the origin file
      while((read_size = read(fd_origin, buf, MAX_BUF))!= 0){
                  //write to the dest file
                  write_size = write(fd_dest, buf, read_size);
      }
      close(fd_origin); close(fd_dest);
}
```

hint

```
[1] $man fstat
```

실습9: mycp-adv

- mycp 결과 화면

before

```
$ ls -l
inhoinno@inhoinno:~/TABA_OS_2023/mycp-adv$ ls -l
total 52
-rw-rw-r-- 1 inhoinno inhoinno   226  3월   7 23:54 Makefile
-rwxrwxr-x 1 inhoinno inhoinno 20872  3월   7 23:55 mycp
-rw-rw-r-- 1 inhoinno inhoinno  1162  3월   7 23:54 mycp-answer.c
-rw-rw-r-- 1 inhoinno inhoinno  9320  3월   7 23:55 mycp-answer.o
-rw-rw-r-- 1 inhoinno inhoinno   894  3월   7 23:16 mycp.c
-rw-r--r-- 1 inhoinno inhoinno    30  3월   7 23:22 origin
```

```
$ cat origin
inhoinno@inhoinno:~/TABA_OS_2023/mycp-adv$ cat origin
mycp program
I am origin file
```

after

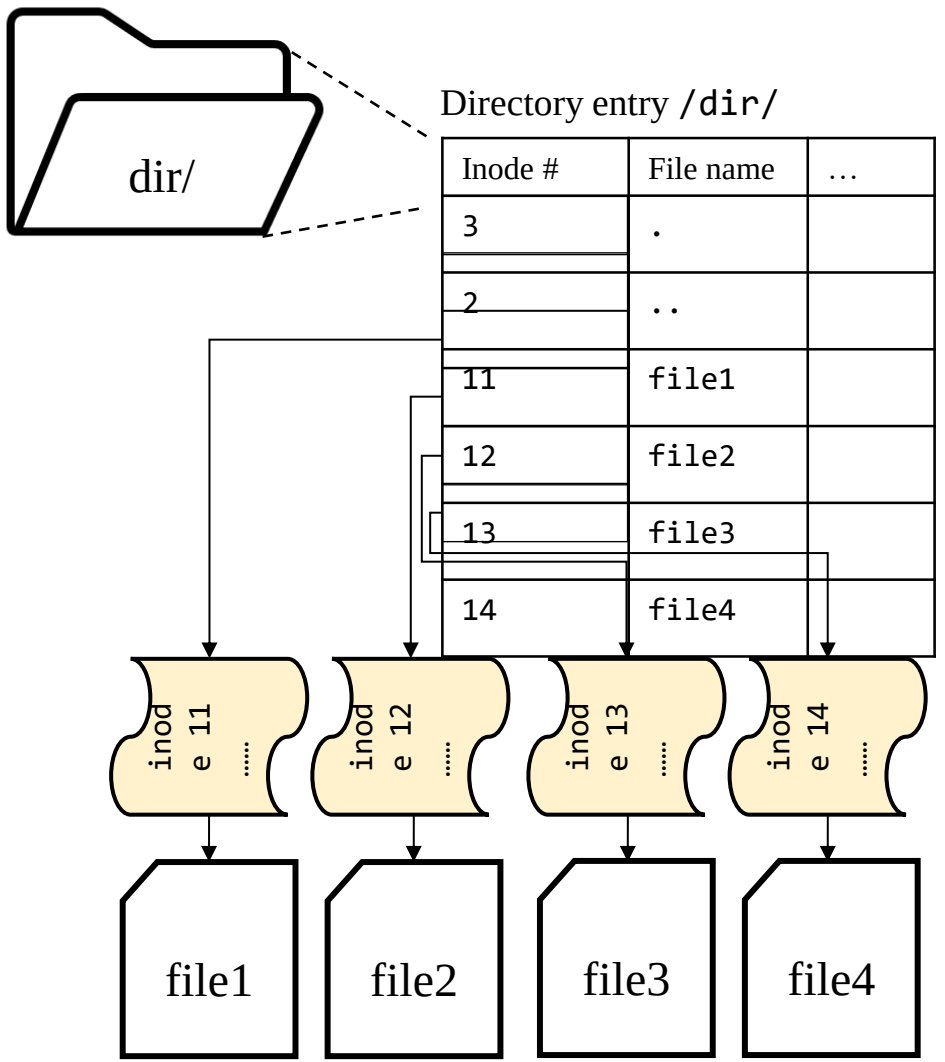
```
$ make
$ ls -l
inhoinno@inhoinno:~/TABA_OS_2023/mycp-adv$ ./mycp origin dest
flag origin 100644
inhoinno@inhoinno:~/TABA_OS_2023/mycp-adv$ ls -l
total 56
drwxrwxr-x 2 inhoinno inhoinno  4096  3월   8 00:04 answer
-rw-r--r-- 1 inhoinno inhoinno    30  3월   8 00:05 dest
-rw-rw-r-- 1 inhoinno inhoinno   219  3월   8 00:00 Makefile
-rwxrwxr-x 1 inhoinno inhoinno 20872  3월   8 00:04 mycp
-rw-rw-r-- 1 inhoinno inhoinno  1072  3월   8 00:04 mycp.c
-rw-rw-r-- 1 inhoinno inhoinno  9232  3월   8 00:04 mycp.o
-rw-r--r-- 1 inhoinno inhoinno    30  3월   7 23:22 origin
```

```
$ cat dest
inhoinno@inhoinno:~/TABA_OS_2023/mycp-adv$ cat dest
mycp program
I am origin file
```

개념설명 - 파일, 그리고 디렉토리

- 용어 정의 terminology
  - 파일 file
    1. User data
       - 실제 파일에 저장되는 데이터
    2. Metadata
       - File의 정보와 user data의 위치를 가리키는 데이터
  - 디렉토리 directory
    1. User data
       - Directory entry

       :디렉토리 내 파일 및 하위 디렉토리를 가리키는 자료구조



Linux 에서 일반 파일과 비교했을 때 디렉토리가 가지는 특별한 차이점은?

실습 10: myls

- myls

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <dirent.h>
#include <sys/types.h>
int main(int argc, char *argv[]){
      //변수 선언
      DIR *dir=NULL;
      struct dirent* dentry=NULL;
      char *dir_name=".";
      //예외 처리
      if (argc == 1){ // args 없는 경우 현재 디렉토리 "." 내용을 보여줌.
                dir = opendir(dir_name);        }
      else if (argc == 2){
                dir_name = argv[1]; //warning.
                /* [1] fall through. fill out here using directory syscall.  */
      }else {printf("argc %d : We only accept 1 or 2 args for now\n", argc);
      exit(-1);  }

      while((dentry =/* [2] using dir syscall*/)!=NULL){
                printf("%s \n", dentry->d_name);
      }
      /* [3] fall through. close directory here.  */;
}
```

hint

```
[ ] $man opendir
[ ] $man readdir
[ ] $man closedir
```

struct dirent

```c
struct linux_dirent64 {
            u64                 d_ino;
            s64                 d_off;
            unsigned short      d_reclen;
            unsigned char       d_type;
            char                d_name[];
};
```

https://elixir.bootlin.com/linux/v5.15.98/source/include/linux/dirent.h

실습 10: myls

- myls 결과 화면

```
$ make
$ ./myls
```

```
inhoinno@inhoinno:~/TABA_OS_2023/myls$ ./myls
myls.c
myls
..
myls.o
.
Makefile
```

```
$ ./myls ..
```

```
inhoinno@inhoinno:~/TABA_OS_2023/myls$ ./myls ..
mycp
myls
mycp-adv
LICENSE
..
mycat
mycreat
.git
.
```

그 이외 파일 입출력 관련 시스템콜

- creat()
- mkdir(), readdir(), rmdir()
- pipe()
- mknod()
- link(),unlink()
- dup(),dup2()
- stat(),fstat()
- chmod(), fchmod()
- Ioctl(), fcntl()
- Sync(), fsync()