File I/O
part 1

한양대학교 소프트웨어학부

Dept. of Division of Computer Science
Hanyang University

File Operations

- create
- write
- read
- reposition within file
  - file seek
- delete
- open(Fi)
  - search the directory structure on disk for entry Fi, and move the content of entry to memory.
- close (Fi)
  - move the content of entry Fi in memory to directory structure on disk.

System Call - open

사용법

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

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

Return value – [성공시 : new file descriptor] [실패시 :- 1]

Flag <fcntl.h>

O_RDONLY	읽기 전용
O_WRONLY	쓰기 전용
O_RDWR	읽기 쓰기
O_CREAT	파일이 존재 하지 않으면 생성
O_EXCL	O_CREAT와 함께 사용되며 파일이 존재 시 에러처리
O_TRUNC	파일이 존재 시 잘라버림
O_APPEND	파일의 뒷부분에 추가

Open - example

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> // close
#include <fcntl.h> // O_RDWR

char *workfile = "test.txt";

int main()
{
        int fd;

        // <fcntl.h>에 정의된 O_RDWR을 사용하여 개방한다.
        // 파일을 읽기/쓰기 권한으로 개방한다.

        if ((fd = open(workfile, O_RDWR)) == -1) {
                printf("Couldn't open %d\n", workfile);
                exit(1);
        }

        /*
         * 프로그램의 나머지 부분
         */

        close(fd);
        exit(0);
}
```

System Call – creat

사용법

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

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

: 새 파일 생성시 사용
: open 에서 O_CREAT|O_WRONLY|O_TRUNC flag와 같음

System Call - close

사용법

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int close(int fd);
```

: 파일 사용을 끝냈음을 시스템에게 알림

Return value - [성공시 : 0]
[실패시 : -1]

System Call - read

사용법

```
#include <unistd.h>

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

Return value – [성공시 : number of bytes read]
[End of file : 0]
[실패시 :- 1]

: 파일로 부터 임의의 byte를 버퍼로 복사하는데 사용

System Call - write

사용법

```
#include <unistd.h>

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

Return value – [성공시 : number of bytes read]
[End of file : 0]
[실패시 :- 1]

: 버퍼로부터 임의의 byte를 파일에 쓰는데 사용

Example : simpleio.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>

#define BSIZE 1024
#define FPERM 0644

int main()
{
        int fd1, fd2, n;
        char buf[BSIZE];

        if ((fd1 = open("test.in", O_RDONLY)) < 0) {
                perror("file open error");
                exit(1);
        }

        if ((fd2 = creat("test.out", FPERM)) < 0) {
                perror("file creation error");
                exit(1);
        }

        while ((n = read(fd1, buf, BSIZE)) > 0) {
                // assume no read/write error
                write(fd2, buf, n);
        }

        close(fd1);
        close(fd2);

        exit(1);
}

// 실행시 반드시 test.in 파일이 존재해야함 !
```

System Call - lseek

사용법

```
#include <sys/types.h>
#include <unistd.h>

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

: 열린 파일의 읽기/쓰기 위치를 옮긴다
: offset : file pointer를 상대적으로이동할 바이트수
  - 양수: 뒤로, 음수: 앞으로
: whence
  - SEEK_SET : 처음에서
  - SEEK_CUR : 현재 위치에서
  - SEEK_END : 끝에서
: Return value
  - 성공시: file pointer의 새로운 위치(절대위치)
  - 실패시: -1

System Call - lseek (cont'd)

기존 파일의 끝에 추가하기

```
…
filedes= open(filename,O_RDWR);
lseek(filedes, (off_t)0, SEEK_END);
write(filedes, outbuf, OBSIZE);
```

파일의 크기를 알아볼때

```
…
off_t filesize;
int filedes;
…
filesize= lseek (filedes, (off_t)0, SEEK_END);
```

한 파일의 끝에 자료를 추가하기

1) lseek(filedes, (off_t)0, SEEK_END);
   write(filedes, appbuf, BUFSIZE);
2) filedes= open("yetanother", O_WRONLY | O_APPEND); -> 4page
   write(filedes, appbuf, BUFSIZE);

Example : lseek (makehole.c)

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

char buf1[] = "abcdefghij";
char buf2[] = "ABCDEFGHIJ";

int main()
{
        int fd;

        if ((fd = creat("file.hole", 0640)) < 0) {
                perror("creat error");
                exit(1);
        }

        if (write(fd, buf1, 10) != 10) {
                perror("buf1 write error");
                exit(1);
        }
        /* offset now = 10 */

        if (lseek(fd, 40, SEEK_SET) == -1) {
                perror("lseek error");
                exit(1);
        }
        /* offset now = 40 */

        if (write(fd, buf2, 10) != 10) {
                perror("buf2 write error");
                exit(1);
        }
        /* offset now = 50 */

        exit(1);
}
```

System Call - unlink/remove

사용법

```
#include <unistd.h>
int unlink(constchar *pathname);

#include <stdio.h>
int remove(constchar *pathname);
```

unlink/remove : 파일을 제거한다
- pathname: 절대적 혹은 상대적 파일/디렉토리 경로명
- Return value [성공시: 0] [실패시: -1]
- 빈 디렉토리를 제거할 때는 remove만을 사용한다

System Call - fcntl

사용법

```
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
int fcntl(int fd, int cmd, ...);
```

fcntl: 열린 파일의 속성(attribute)을 제어한다
- –fd: open 혹은 creat가 반환한 파일 descriptor
- –cmd:
  - F_GETFL : flag를 통한 파일 상태 표시기를 되돌려준다
  - F_SETFL : 파일 상태 표시기를 세번째 변수의 값으로 정한다
    - – O_APPEND, O_NONBLOCK, O_SYNC, O_ASYNC만 가능
- – Return value
  - [성공시: 자연수(>=0)]
    - – F_GETFL사용시는(반환값&O_ACCMODE)가open의flag임
  - [실패시: -1]
    - – F_SETFL사용시는0#

Example : fcntl (filestatus.c)

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <stdlib.h>

void filestatus(int fd)
{
        int accmode, val;
        if ((val = fcntl(fd, F_GETFL, 0)) < 0) {
                perror("fcntl error for fd");
                exit(1);
        }


        printf("fd = %d: ", fd);
        accmode = val & O_ACCMODE;
        if (accmode == O_RDONLY)
                printf("read only");
        else if (accmode == O_WRONLY)
                printf("write only");
        else if (accmode == O_RDWR)
                printf("read/write");
        else {
                fprintf(stderr, "unknown access mode");
                exit(1);
        }

        if (val & O_APPEND)
                printf(", append");
        putchar('\n');
}

int main()
{
        filestatus(open("test.in", O_RDWR));
        filestatus(open("test.in", O_RDONLY));
        filestatus(open("test.in", O_WRONLY | O_APPEND));

        exit(1);
}
```

errno & perror

- 파일 접근 system call
  - 실패시: -1 return
- errno
  - 오류변수, <errno.h>에 포함
  - system call동안 발생했던 오류의 마지막 type 기록
- perror 서브루틴
  - 문자열 인수, 콜론, errno변수의 현재값의 메시지를 출력
  - perror("error opening nonesuch");
  - 만일 nonesuch가 존재하지 않으면
  - error opening nonesuch: No such file or directory 출력

실습

한 파일의 내용을 다른 파일로 복사하는
함수 copyfile(src*, dest*)을 작성하여
간단한 cp 프로그램 구현하시오.

1. 원본 파일을 개방 (RDONLY로 open)
2. 대상 파일을 생성 (creat대신 open사용)
3. 원본 파일의 끝에 도달할 때까지 파일을 읽어 대상 파일에 기록
4. 두 파일을 모두 닫음

실행파일이 simplecp 일 경우
실행예) $ ./simplecp [source] [destination]
source파일을 destination으로 복사

Q & A

- Thank you :)