

UNIX 시스템 프로그래밍

» 2장. 파일 입출력

UNIX file 접근 primitives

- ▶ file : byte들의 linear sequence
- ▶ 파일 입출력
 - 저수준 파일 입출력 (open, close, read, write, dup, dup2, fcntl, lseek, fsync)
 - 고수준 파일 입출력 (fopen, fclose, fread, fwrite, fputs, fgets, fprintf, fscanf, freopen, fseek)

file descriptor

- ▶ 현재 open된 file을 구분할 목적으로 UNIX가 붙여 놓은 번호
- ▶ 표준 입출력
 - 0 : 표준 입력
 - 1 : 표준 출력
 - 2 : 표준 오류 출력
- ▶ 한 프로세스가 동시에 open 할 수 있는 file의 개수에는 제한이 있음.
 - close 사용

open 시스템 호출

- ▶ 기존의 file을 open 하거나, 새롭게 file을 생성하여 open하는 system call
- ▶ 사용법 :

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

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

```
#include <fcntl.h>
```

```
int open(const char *filename, int oflag, [mode_t mode]);
```

open 시스템 호출 (2)

▶ 인수 사용법 :

- filename : 파일 이름
- oflag : file을 access하는 방식
O_RDONLY 또는 O_WRONLY 또는 O_RDWR
그리고, file을 create하는 방식
O_CREAT 그리고 O_EXCL 또는 O_TRUNC
그리고,
O_APPEND
- mode : file을 새로 생성할때만 사용
0600 또는 0664 또는 0644 또는
- return 값 : 실행 성공 시 file descriptor (음이 아닌 정수);
실행 실패 시 -1;

creat 시스템 호출

- ▶ file을 생성하여 open하거나, 기존 file을 open하는 system call

- ▶ 사용법 :

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int creat(const char *filename, mode_t mode);
```

- ▶ 주의 사항 :
 - file을 쓰기 가능한 상태로 open;
 - file이 이미 존재하면: 두 번째 인자는 무시; 기존 file은 0으로 open!

close 시스템 호출

- ▶ open 된 file을 close 할때 사용
 - process 종료 시 open된 file들은 자동으로 close.
 - 그러나, 동시에 open할 수 있는 file 수 제한 때문에 close 사용;
- ▶ 사용법 :

```
#include <unistd.h>
int close(int filedes);
```
- ▶ 인수 사용법 :
 - filedes : open된 file의 file descriptor;
 - return 값 : 성공 시 0; 실패 시 -1;

read 시스템 호출

- ▶ open된 file로부터 지정한 byte수 만큼의 data를 읽어 지정된 저장장소에 저장하는 명령
 - file pointer or read-write pointer : 읽혀질 다음 byte의 위치를 나타냄;
- ▶ 사용법 :

```
#include <unistd.h>
ssize_t read(int fildes, void *buffer, size_t nbytes);
```


read 시스템 호출 (2)

▶ 인수 사용법 :

- `filedes` : open된 file의 file descriptor;
- `*buffer` : 읽은 data를 저장할 곳의 주소; data type은 상관 없음;
- `nbytes` : 읽을 byte 수; data type에 상관 없이 지정된 byte 수 만큼 읽음;
- return 값 : 성공 시, 실제 읽힌 byte 수; 실패 시, -1;
 - return 값 < nbytes이면, file에 남은 data가 nbytes보다 적을 때;
 - 그 다음은 더 이상 읽을게 없으면; return 값은 0;

write 시스템 호출

- ▶ 지정된 저장장소에서 지정한 byte수 만큼의 data를 읽어 open된 file에 쓰는 명령
 - file pointer or read-write pointer: 쓰여질 다음 byte의 위치를 나타냄;
- ▶ 사용법 :

```
#include <unistd.h>
ssize_t write(int fildes, const void *buffer, size_t nbytes);
```

write 시스템 호출 (2)

▶ 인수 사용법 :

- filedes : write를 할 file의 descriptor;
- *buffer : write 할 내용이 들어 있는 저장 장소의 주소;
- nbytes : write할 byte의 수;
- return 값 : 쓰여진 byte 수 or -1;
 - 보통은 $\text{return 값} = n$;
 - $\text{return 값} < n$ 이면, 쓰는 도중 media가 full;
 - 만약, 쓰기 전에 꽉 차면; -1 return;

write 시스템 호출 (3)

▶ 주의 사항

- 기존 file을 open system call로 open하고 바로 write를 하면???
- `fd = open("data", O_WRONLY|O_APPEND);`
→ open 하자마자, file pointer가 file의 끝으로 이동;

read/write의 효율성

- ▶ File을 copy하는 프로그램의 실행 시간
 - BUFSIZE가 512 (disk blocking factor)의 배수 일 때, 효율적;
 - system call의 수가 적을수록 효율적;

lseek와 임의 접근

- ▶ open된 file내의 특정 위치로 file pointer를 이동시키는 system call
- ▶ 사용법 :

```
#include <sys/types.h>
#include <unistd.h>
off_t lseek(int fildes, off_t offset, int whence);
```

lseek와 임의 접근 (2)

▶ 인수 사용법 :

- `filedes` : open된 file의 file descriptor
- `whence` : `whence`에서 `offset` 만큼 떨어진 위치로 이동
 - `whence` 위치는 아래 위치 중 선택:
 - `SEEK_SET` : file 시작 지점
 - `SEEK_CUR` : 현재 file 위치
 - `SEEK_END` : file의 끝 지점
- `offset` : `whence`에서 `offset` 만큼 떨어진 위치로 이동; `+/-` 로 방향 설정 가능;
- `return` 값 : 이동된 위치 (시작점부터의 위치); 이동 실패 시 `-1`;

file의 제거

- ▶ file을 삭제(?)하는 명령
- ▶ 사용법 :
 - `#include <unistd.h>`
`int unlink(const char *filename);`
 - `#include <stdio.h>`
`int remove(const char *filename)`

file의 제거 (2)

- ▶ 주의 사항 :
 - include file이 다르다!
 - file descriptor가 아니라, file 이름을 쓴다!
 - 성공시 0; 실패 시 -1 return;
 - file이 open 되어 있는 상태면?
- ▶ 차이점 : unlink는 file 만 삭제(?), remove는 file 과 빈 directory도 삭제;

표준 입력, 표준 출력, 표준 오류

- ▶ 표준 입력(키보드) : $fd=0$
- ▶ 표준 출력(터미널 화면) : $fd=1$
- ▶ 표준 오류(터미널 화면) : $fd=2$

표준 입출력의 변경

▶ redirection :

- `$./a.out < infile` (infile0| file descriptor 0.)
- `$./a.out > outfile` (outfile0| file descriptor 1.)
- `$./a.out < infile > outfile`

▶ pipe :

- `$./a.out1 | ./a.out2`
(a.out1의 표준 출력이 a.out2의 표준 입력으로)

오류 메시지 출력

- ▶ 예:
 - `perror("error ... ");`
→ `error ... : No such file or directory`