UNIX 시스템 프로그래밍

- 3장. 파일과 디렉토리

파일 정보의 획득

- 파일 관련 각종 정보를 알아볼 수 있는 system call

- 사용법 :

```
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
int stat (const char *pathname, struct stat *buf);
int fstat (int filedes, struct stat *buf);
```

  - fstat는 open된 file에 대한 정보를 획득;
  - buf엔 file 정보가 저장;

파일 정보의 획득 (2)

- buf 에 채워진 내용은 ?
  - st_dev, st_ino : identifier (논리적 장치번호와 inode 번호)
  - st_mode : permission mode
  - st_nlink : link의 수
  - st_uid, st_gid : user의 uid와 gid
  - st_rdev : file이 장치인 경우만 사용
  - st_size : 논리적 크기
  - st_atime, st_mtime, st_ctime : file의 최근 access time, update time, stat 구조의 update time
  - st_blksize : I/O block 크기
  - st_blocks : 파일에 할당된 block의 수

access permission

- file mode :

  0764=0400+0200+0100+0040+0020+0004

  - owner:group:others:
  - read(4)+write(2)+execution(1)

- 상징형 모드 : page 136, 표 3.9

  0764 = S_IRUSR | S_IWUSR | S_IXUSR |
  | S_IRGRP | S_IWGRP | S_IROTH

그 밖의 permission

- 04000 S_ISUID : 실행이 시작되면, 소유자의 uid 가 euid가 된다.
- 02000 S_ISGID : 실행이 시작되면, 소유자의 gid 가 egid가 된다.

사용자와 소유권

- 소유권자 : 파일을 생성한 사람, user-id인 uid로 구분 (gid)
- 사용자 : 파일을 사용하는 사람, effective user-id 인 euid로 구분 (egid)

permission 확인

- if (s.st_mode&S_IRUSR)
  printf("소유자 읽기 권한 설정!!!");
  - 파일 접근 권한 확인 상수 : page 136, 표3-9

- if (S_ISREG(s.st_mode))
  printf("일반 파일!!!");
  - 파일 종류 확인 상수 : page 132, 표3-6

access 시스템 호출

- 특정 파일에 대한 읽기/쓰기/실행이 가능한지 확인하는 system call
- 사용법 :

```
#include <unistd.h>
int access(const char *pathname, int amode);
```

- 인수 사용법 :
  - amode : R_OK , W_OK, X_OK, 또는 F_OK
  - return 값 : 0 or -1
  - euid가 아니라 uid에 근거하여 process가 file에 접근 가능한지를 표현;

chmod 시스템 호출

- 특정 파일의 access permission을 변경하는 system call

- 사용법 :

```
#include <sys/types.h>
#include <sys/stat.h>
int chmod(const char *pathname, mode_t mode);
int fchmod(int fd, mode_t mode);
```

- 소유자만 사용 가능;

link 시스템 호출

- 기존 파일에 새로운 이름을 부여하는 system call
  - 한 파일에 하나 이상의 이름 : 각 이름을 "hard link" 라 부름
  - link count : link의 수
- 사용법:

  ```
  #include <unistd.h>
   int link (const char *orginal_path, const char *new_path);
  ```

  - return값 : 0 or -1 (new_path가 이미 존재하면.)
- unlink의 재검토 : unlink("a.out");
  - 실은 link를 제거 ➔ link_count--
  - 만약, link_count가 0이 되면 실제로 제거 (free block으로);

symbolic link

- symbol형 link ;
  - link의 제한점 :
    - directory에 대한 link 생성은 불가;
    - 다른 file system에 있는 file에 대해서는 link 생성 불가;

  ➔ symbolic link (자체가 file; 그 안에 다른 file에 대한 경로 수록;)

- 사용법 :

  #include <unistd.h>

  int symlink (const char *realname, const char *symname);
  - return 값 : 0 or -1 (symname이 이미 존재 하면)

symbolic link (2)

- open system call에 의해 open 가능; 어떤 파일이 open 되는지? realname? or symname?
- readlink의 사용 : symname안에 있는 내용을 보고 싶으면.

```
#include <unistd.h>
int readlink (const char *sympath, char *buffer,
size_t buffsize);
```

symbolic link (3)

- 사용법 :

```
#include <sys/types.h>
#include<sys/stat.h>
int lstat (const char *linkname, struct stat *buf);
```

- symbolic link가 가리키는 파일이 아닌 symbolic link 자체의 파일 정보를 전달

direcroty 생성 및 제거

- directory 생성: directory file에 .과 ..을 넣어서 생성;

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

- directory 제거: directory가 비어 있을 경우에만 성공;

```
#include <unistd.h>
int rmdir(const char *pathname);
```

rename 시스템 호출

- 사용법 :

  #include <stdio.h>

  int rename (const char *oldpathname, const char *newpathname);

  - file과 directory 둘 다 rename 가능;
  - newpathname이 존재해도 -1이 아님. 기존 file을 제거하고 새 이름 부여;

getcwd 시스템 호출

- 사용법 :

  #include <unistd.h>
  char *getcwd(char *name, size_t size);

  - return 값 : current working directory의 경로 이름
  - size는 실제 길이 보다 +1이 커야 한다.
  - 성공 시 directory의 경로 이름이 name에 copy; 실패 시 null pointer return;

chdir 시스템 호출

- 사용법 :

```
#include <unistd.h>
int chdir(const char *path);
```

- 예:

```
fd1=open("/usr/ben/abc", O_RDONLY);
fd2=open("/usr/ben/xyz", O_RDWR);
```

➔ chdir("/usr/ben");

```
fd1=open("abc", O_RDONLY);
fd2=open("xyz", O_RDWR);
```

directory 열기 및 닫기

- directory 열기:

  ```
  #include <sys/types.h>
  #include <dirent.h>
  DIR *opendir(const char *dirname);
  ```

  ➔ DIR형의 data structure에 대한 pointer를 return; 실패 시 null pointer return;

- directory 닫기:

  ```
  #include <dirent.h>
  int closedir(DIR *dirptr);
  ```

directory 읽기

- 사용법:

```
#include <sys/types.h>
 #include <dirent.h>
 struct dirent *readdir (DIR *dirptr);
```

  - return 값: dirptr이 가리키는 DIR 구조내의 한 항;
  - struct dirent : ino_t d_ino;
    char d_name[];
  - pointer dirptr은 read 후 다음 항을 가리킨다.
  - directory의 끝에 도달하면 null pointer를 return;

directory file pointer 이동

- 사용법 :

```
#include <sys/types.h>
#include <dirent.h>
void rewinddir(DIR *dirptr);
```