System Programming

Exercise

Week 03. File I/O – Part 2

허가와 파일모드

- Permission: 상이한 사용자들이 파일에 접근할 권한
  - 파일의 소유자가 permission결정
- 사용자 유형
  1) 파일의 소유자
  2) 파일에 연관된 그룹에 속하는 사용자
  3) 1),2)에 속하지 않는 사용자
- 파일 permission의 세가지 그룹
  1) 파일을 읽기
  2) 파일에 쓰기
  3) 파일의 수행

파일 permission변경

- 팔진수 사용법
  - chmod 666 foo : foo 파일을 모든 사용자가 읽고 쓸 수 있게 한다
  - 0700 + 050 + 05 = 0755
- 상징형 모드
  - S_IRUSR: 소유자에게 읽기 허가
  - S_IRGRP: 그룹에 대해 읽기를 허가
  - or 연산가능: S_IRUSR | S_IRGRP

System Call – umask

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

mode_t umask(mode_t newmask);
```

- umask: 파일 사용 권한을 제한하기 위한 mask를 지정
- newmask: 8진수 형식, 다음 페이지 참조
- Return value: 항상 성공하며 이전에 설정된 mask값을 리턴

ex) umask 값을 022로 했을 경우 mode를 0666으로 했다면

-> 0666 & -022 = 0644 = rw-r--r-- 권한을 가지는 파일생성

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

newmask 값의 의미

자릿수	값	의미
1	0	모든 사용자 권한이 허용된다
	4	사용자 읽기 권한이 금지된다
	2	사용자 쓰기 권한이 금지된다
	1	사용자 실행 권한이 금지된다
2	0	모든 그룹 권한이 허용된다
	4	그룹 읽기 권한이 금지된다
	2	그룹 쓰기 권한이 금지된다
	1	그룹 실행 권한이 금지된다
3	0	모든 기타 권한이 허용된다
	4	기타 읽기 권한이 금지된다
	2	기타 쓰기 권한이 금지된다
	1	기타 실행 권한이 금지된다

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

Example – umask

```
1   #include <stdio.h>
2   #include <fcntl.h>
3   #include <unistd.h>
4   #include <sys/stat.h>
5   #include <sys/types.h>
6
7   int main() {
8     int fd;
9     mode_t old_mask, new_mask;
10    old_mask=umask(022);
11    fd = open("test", O_CREAT|O_WRONLY, 0666);
12    printf("old_mask=%o\n",old_mask);
13
14    new_mask = umask(old_mask);
15    printf("new_mask=%o\n",new_mask);
16    close(fd);
17    return 0;
18  }
```

System Call – access

```
#include <unistd.h>

int access(const char *pathname, int amode);
```

- access
  - 지정된 파일에 대해서 권한을 가지고 있는지 체크
- int amode
  - R_OK : 읽기 여부 체크
  - W_OK : 쓰기 여부 체크
  - X_OK : 실행 여부 체크
  - F_OK : 파일의 존재 유무
- Return value
  - [성공 시: 0]
  - [실패 시: -1]

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

Example – access

```c
#include <stdio.h>
#include <unistd.h>
int main(){
  if(access("test", F_OK) != 0){
    perror("파일이 존재하지 않음");
  } else {
    printf("파일이 존재함\n");
    if(access("test", R_OK | W_OK) != 0){
      perror("읽기 쓰기 권한이 없음");
    } else {
      printf("읽기 쓰기 권한이 있음\n");
    }
    if(access("test", X_OK) != 0){
      perror("실행 권한이 없음");
    } else {
      printf("실행 권한이 있음\n");
    }
  }
  return 0;
}
```

System Call – chmod

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

int chmod(const char *pathname, mode_t newmode);
```

- chmod
  - 파일의 모드를 newmode 모드로 변경한다.
- mode_t *newmode*
  - 새로운 파일 접근 모드 (팔진수, 상징형 모드)
- Return value
  - [성공 시: 0]
  - [실패 시: -1]

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

Example – chmod

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

int main(){
  if(chmod("test", S_IRUSR | S_IRGRP | S_IROTH | S_IXUSR) < 0 ){
    perror("chmod error: ");
  }
  return 0;
}
```

System Call – chown

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

int chown(const char *pathname, uid_t owner_id, gid_t group_id);
```

- chown
  - 파일에 대한 소유권을 바꾸기 위해서 사용
  - 파일의 소유자나 수퍼 사용자만이 이용가능
- uid_t *owner_id*
  - 새 소유자
- gid_t *group_id*
  - 새 그룹
- Return value
  - [성공 시: 0]
  - [실패 시: -1]

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

Example – chown

```c
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
#include <pwd.h>
#include <grp.h>

int main() {
  struct passwd *u_info;
  u_info = getpwnam("testuser");
  chown("test", u_info->pw_uid, -1);
  return 0;
}
```

System Call – rename

```
#include <stdio.h>

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

- rename
  - 파일의 이름이나 위치를 변경
- Return value
  - [성공 시: 0]
  - [실패 시: -1]

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

Example – rename

```c
#include <stdio.h>

int main() {
  if (rename("test", "test2") < 0)
    perror("rename : ");
  else
    printf("success\n");
  return 0;
}
```

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

System Call – link

```
#include <unistd.h>

int link(const char *orginal_path, const char *new_path);
int unlink(const char *pathname);
```

- link
  - 존재하는 파일에 대해서 새로운 연결(하드 링크)을 만듦
  - 만약 *new_path* 가 이미 존재한다면, 덮어쓰지 않음
- unlink
  - 지명된 링크를 제거하고, 파일의 링크카운트를 하나 줄임
- char **new_path*
  - 새로운 링크 또는 이름
- Return value
  - [성공 시: 0]
  - [실패 시: -1]

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

Example – link

```c
#include <unistd.h>
#include <stdio.h>

int main(){
  if(access("test", F_OK) < 0){
    perror("access: ");
    return -1;
  }

  if(link("test", "testl") < 0){
    perror("link: ");
    return -1;
  }
  printf("success\n");
  return 0;
}
```

System Call – symlink

```
#include <unistd.h>

int symlink(const char *oldpath, const char *newpath);
```

- symlink
  - 파일에 대한 심볼릭 링크를 만든다.
  - 심볼릭 링크 *newpath*가 이미 존재하면 덮어쓰지 않음
- Return value
  - [성공 시: 0]
  - [실패 시: -1]

- 주의할 점
  - realname 파일 삭제 시 : 심볼형 링크 파일 사용불가

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

System Call – stat, fstat

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

int stat(const char *pathname , struct stat *buf);
int fstat(int filedes , struct stat *buf);
```

- stat, fstat
  - 파일의 상태를 얻어온다.
  - 성공적으로 수행될 경우 파일의 정보를 stat구조체에 복사
- struct stat *buf
  - stat 구조를 가리키는 포인터
- Return value
  - [성공 시: 0]
  - [실패 시: -1]

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

struct stat의 구성

```
struct stat {
    dev_t       st_dev;     /* 파일이 들어있는 논리적 장치를 기술 */
    ino_t       st_ino;     /* 파일의 inode번호 */
    mode_t      st_mode;    /* 파일 모드 */
    nlink_t     st_nlink;   /* 하드 링크의 수 */
    uid_t       st_uid;     /* 파일 사용자 식별번호 */
    gid_t       st_gid;     /* 파일 그룹 식별번호 */
    dev_t       st_rdev;    /* 논리적 장치 의  type */
    off_t       st_size;    /* 파일의 크기(byte로 표시) */
    blksize_t   st_blksize; /* 파일 시스템 고유의 I/O 블록 크기 */
    blkcnt_t    st_blocks;  /* 파일에 할당된 물리적 파일 시스템 블록의 수 */
    time_t      st_atime;   /* 파일의 자료가 마지막으로 읽혔던 시간 */
    time_t      st_mtime;   /* 파일의 자료가 마지막으로 변경된 시간 */
    time_t      st_ctime;   /* stat 구조체가 변경된 시간*/
};
```

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

Example – stat (1/3)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <pwd.h>
7  #include <grp.h>
8
9  int main(){
10   char *file_name = "test";
11   struct stat file_info;
12   struct passwd *my_passwd;
13   struct group *my_group;
14   mode_t file_mode;
15
16   if(stat(file_name, &file_info) < 0){
17     perror("stat: ");
18     return -1;
19   }
```

- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

Example – stat (2/3)

```
20
21    file_mode = file_info.st_mode;
22    printf("File name: %s\n", file_name);
23    printf("=====================================\n");
24    printf("File type: ");
25
26    if(S_ISREG(file_mode)) printf("Regular file\n");
27    else if(S_ISLNK(file_mode)) printf("Symbolic link\n");
28    else if(S_ISDIR(file_mode)) printf("Directory\n");
29    else if(S_ISCHR(file_mode)) printf("Character device\n");
30    else if(S_ISBLK(file_mode)) printf("Block device\n");
31    else if(S_ISFIFO(file_mode)) printf("FIFO file\n");
32    else if(S_ISSOCK(file_mode)) printf("Socket file\n");
33
34    my_passwd = getpwuid(file_info.st_uid);
35    my_group = getgrgid(file_info.st_gid);
```

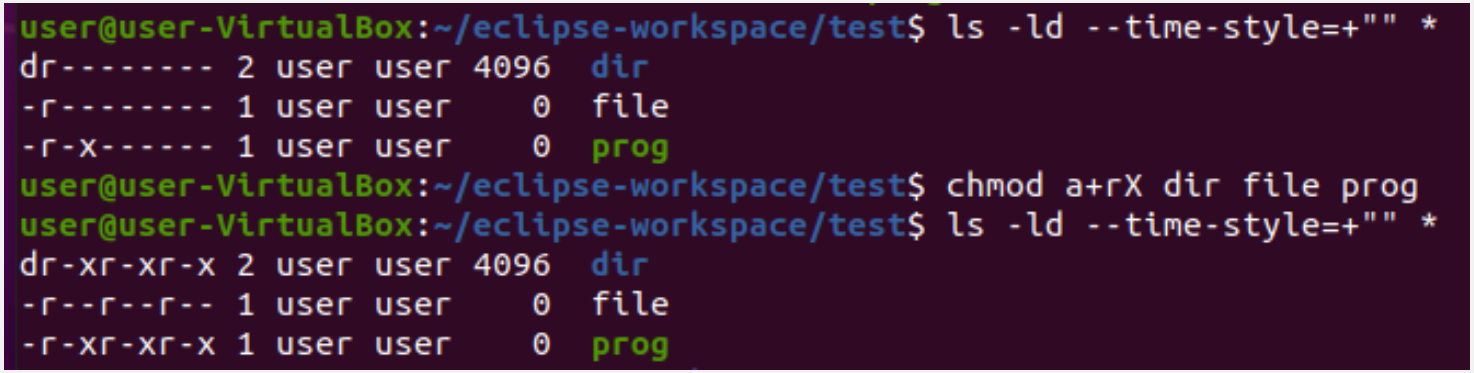
- Permission & mode
- umask
- access
- chmod
- chown
- rename
- link
- symlink
- stat

Example – stat (3/3)

```
36
37    printf("OWNER: %s\n", my_passwd->pw_name);
38    printf("GROUP: %s\n", my_group->gr_name);
39    printf("File size is: %ld\n", file_info.st_size);
40    printf("Number of hard links: %ld\n", file_info.st_nlink);
41
42    return 0;
43  }
```

실습 문제

- `chmod a+rX [path]` 명령은 다음과 같은 일을 수행한다.
  - 모든 읽기 권한 활성화
  - 디렉토리에는 모든 실행 권한을 활성화 한다.
  - 이미 실행 권한이 있는 파일에 대해 모든 실행 권한을 활성화 한다.
- 아래 그림은 위의 명령을 실행한 예제이다.



- `chmod a+rX`와 동일하게 동작하는 프로그램을 `stat()`과 `chmod()`를 이용하여 구현하라.

THANK YOU

Presented by Hasoo Eun