Linux System Programming #3-1

Spring 2020

School of Computer Science and Engineering,

Soongsil University, Seoul, Korea

Jiman Hong

jiman@acm.org

III. 파일 속성과 디렉토리

- **파일 속성 확인 변경 함수**
  - stat 구조체
- **디렉토리 제어 함수**

stat(), fstat(), lstat()

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

리턴 값: 성공 시 0, 에러 시 –1을 리턴/ errno 설정

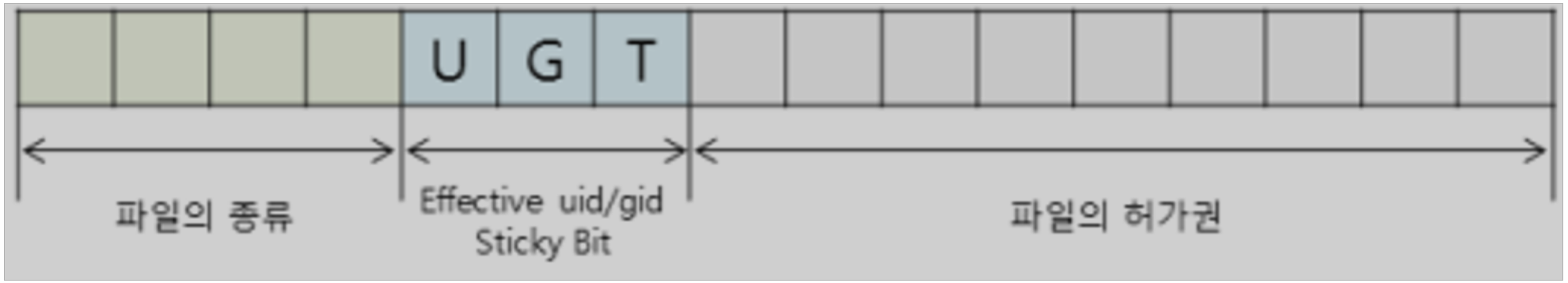
- Pathname/filesdes에 해당하는 파일에 대한 정보를 포함하고 있는 stat 구조체를 리턴하는 시스템호출
  - stat 구조체에 포함되어 있는 멤버 변수들은 운영체제마다 다르게 구현
  - ls 명령에서 사용
- stat()
  - 지정한 경로(pathname)에 해당하는 파일에 대한 정보를 넣은 구조체를 buf 통해서 리턴.
- fstat()
  - 지정한 경로(filedes)가 오픈되어 있는지 확인
- lstat(): 심볼릭 링크 파일 자체 리턴

stat 구조체

```
struct stat {
  //sys/stat.h
   dev_t          st_dev;/* 디바이스 번호 */
   ino_t          st_ino;/* inode 번호 */
   mode_t         st_mode;/* 모드 (접근 권한) */
   nlink_t        st_nlink;/* 하드 링크 수 */
   uid_t st       uid;/* 소유자의 사용자 ID */
   gid_t          st_gid;/* 소유자의 그룹 ID */
   rdev_t         st_rdev;/* 디바이스 ID (특수 파일인 경우) */
   off_t          st_size;/* 파일 크기(바이트) */
   blksize_t      st_blksize;/* 블록 크기 */
   blkcnt_t       st_blocks;/* 512바이트 블록 갯수 */
   time_t         st_atime;/* 최종 접근 시간 */
   time_t         st_mtime;/* 최종 수정 시간 */
   time_t         st_ctime;/* 최종 상태 변경 시간 */
};
```

```
struct statfs {
  //sys/stat.h
   long  f_type.; // 파일 시스템 종류
   long f_bsize; //블록 크기
   long f_blocks; //블록 갯수
   long f_bfree; //free 블록 수
   long f_avail; //슈퍼유저가 아닌 유저 위한 free 블록수
   long f_files; // 파일 시스템 내 총 inode 수
   long f_ffree; //파일 시스템 내  free inode 수
   fsid_t  f_fsid; // 파일시스템 ID
   long  f_namelen; // 파일 이름 최대 길이
   long  f_spare[6] ; //reserved
```

stat(), fstat(), lstat()



st_mode 구조 – 비트 마스크

stat(), fstat(), lstat() –

테스트 파일종류	8진수	2진수	테스트 매크로	파일종류	비고
S_IFMT	0170000	1111 000 000000000		파일 타입 비트 필드 위한 비트 마스크	16bit st_mode에서 상위 4bit 속성값 추출을 위한 마스크
S_IFREG	0100000	1000 000 000000000	S_ISREG()	일반 파일	가장 흔한 종류의 파일로, 파일에 포함되는 데이타의 형태가 특별하게 정해져 있지 않음
S_IFDIR	0040000	0100 000 000000000	S_ISDIR()	디렉토리 파일	다른 파일들의 이름과 그 파일들에 대한 정보를 가리키는 포인터들을 포함하는 파일
S_IFCHR	0020000	0010 000 000000000	S_ISCHR()	문자 디바이스	장치에 대한 가변 크기 단위의 언버퍼링(Unbuffered) 입출력 접근을 제공하는 파일
S_IFBLK	0060000	0110 000 000000000	S_ISBLK()	블록 디바이스	디스크 드라이브 같은 장치에 대한 고정 크기 단위의 버퍼링 입출력 접근을 제공하는 파일
S_IFIFO	0010000	0001 000 000000000	S_ISFIFO()	FIFO, Pipe 파일	프로세스간단 방향 통신에 사용되는 파일
S_IFSOCK	0140000	1100 000 000000000	S_ISSOCK()	소켓	프로세스간 통신에 사용되는 파일
S_IFLNK	0120000	1010 000 000000000	S_ISLNK()	심볼릭 링크	lstat()에 의해서만 리턴, 윈도우의 단축아이콘과 유사
S_IFDOOR	0150000	1101 000 000000000	S_ISDOOR()	Door 파일 (IPC를 위한 특수 파일)	Solaris

stat(), fstat(), lstat()

테스트 매크로	8진수	2진수	내용
S_ISUID	0004000	0000 100 000000000	set-user-id
S_ISGID	0002000	0000 010 000000000	set-group-id
S_ISVTX	0001000	0000 001 000000000	sticky bit
S_IRWXU	0000700	0000 000 111000000	파일 소유자의 권한 마스크
S_IRUSR	0000400	0000 000 100000000	파일 소유자의 읽기 권한
S_IWUSR	0000200	0000 000 010000000	파일 소유자의 쓰기 권한
S_IXUSR	0000100	0000 000 001000000	파일 소유자의 실행 권한
S_IRWXG	0000070	0000 000 000111000	그룹 사용자의 권한 마스크
S_IRGRP	0000040	0000 000 000100000	그룹 사용자의 읽기 권한
S_IWGRP	0000020	0000 000 000010000	그룹 사용자의 쓰기 권한
S_IXGRP	0000010	0000 000 000001000	그룹 사용자의 실행 권한
S_IRWXO	0000007	0000 000 000000111	다른 사용자의 권한 마스크
S_IROTH	0000004	0000 000 000000100	다른 사용자의 읽기 권한
S_IWOTH	0000002	0000 000 000000010	다른 사용자의 쓰기 권한
S_IXOTH	0000001	0000 000 000000001	다른 사용자의 실행 권한

stat(), fstat(), lstat() 예제 1

<ssu_stat_1.c>

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

int main(int argc, char *argv[]) {
   struct stat statbuf;

   if (argc != 2) {
      fprintf (stderr, "usage: %s <file>\n", argv[0]);
      exit(1);
   }
   if ((stat(argv[1], &statbuf)) < 0 ) {
      fprintf(stderr, "stat error\n");
      exit(1);
   }

   printf("%s is %ld bytes\n", argv[1], statbuf.st_size);
   exit(0);
}
```

실행 결과

```
root@localhost:/home/oslab# ./ssu_stat_1 ssu_test.txt
ssu_test.txt is 74 bytes
```

stat(), fstat(), lstat() 예제 2

<ssu_stat_2.c>

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

struct stat statbuf;

void ssu_checkfile(char *fname, time_t *time);
int main(int argc, char *argv[])
{
   time_t  intertime;
   if (argc != 2) {
      fprintf(stderr, "usage: %s <file>\n", argv[0]);
      exit(1);
   }

   if (stat(argv[1], &statbuf) < 0) {
      fprintf(stderr, "stat error for %s\n", argv[1]);
      exit(1);
   }
 intertime = statbuf.st_mtime;
   while (1) {   ssu_checkfile(argv[1], &intertime);
     sleep(10);
   }
}

void ssu_checkfile (char *fname, time_t *time) {
   if (stat(fname, &statbuf) < 0) {
      fprintf(stderr, "Warning : ssu_checkfile() error!\n");
      exit(1);
   }

   else
      if (statbuf.st_mtime != *time) {
         printf("Warning : %s was modified!.\n", fname);
         *time = statbuf.st_mtime;
      }
}
```

실행 결과

```
root@localhost:/home/oslab# ./ssu_stat_2 ssu_hole.txt
Warning : ssu_hole.txt was modified!.
Warning : ssu_hole.txt was modified!.
Warning : ssu_hole.txt was modified!.
Warning : ssu_checkfile() error!
```

파일의 종류

- **st_mode**
  - 일반 파일 : _
  - 디렉토리 파일 : d
  - 특수 파일
    - 소켓 : s
    - (명명, named) 파이프 = FIFO : p
    - 장치 파일 – Block Oriented (블록형), Character Oriented (문자형) : b / c
    - Door 파일 (Solaris) : D
  - 기타
    - 심볼릭링크 파일 : l

```
oslab@minipc:~/test$ ls -al
합계 28
drwxrwxr-x  4 oslab oslab 4096  3월 20 17:28 .
drwxr-xr-x 28 oslab oslab 4096  3월 20 17:25 ..
drwxrwxr-x  2 oslab oslab 4096  3월 20 17:25 backup
drwxrwxr-x  2 oslab oslab 4096  3월 20 17:27 original
-rw-rw-r--  1 oslab oslab    3  3월 20 17:24 ssu_report.c
-rw-rw-r--  1 oslab oslab    3  3월 20 17:28 ssu_result.c
-rw-rw-r--  1 oslab oslab    3  3월 20 17:24 ssu_test.c
```

파일의 종류 확인

- **상수 S_IFMT 사용**

```
stat(pathname, &statbuf);
if ((statbuf.st_mode & S_IFMT) == S_IFREG)
   printf("regular file\n");
```

- **표준 매크로 사용**

```
stat(pathname, &statbuf);
if (S_ISREG(statbuf.st_mode))
   printf("regular file\n");
```

파일의 종류 확인 – stat 구조체의 st_mode 필드 값 이용

표준 테스트 메크로	
#define S_ISREG(statbuf.st_mode)	((statbuf.st_mode) &S_IFMT) == S_IFREG)
#define S_ISDIR(statbuf.st_mode)	((statbuf.st_mode) &S_IFMT) == S_IFDIR)
#define S_ISCHR(statbuf.st_mode)	((statbuf.st_mode) &S_IFMT) == S_IFCHR)
#define S_ISLNK(statbuf.st_mode)	((statbuf.st_mode) &S_IFMT) == S_IFLNK)
#define S_ISFIFO(statbuf.st_mode)	((statbuf.st_mode) &S_IFMT) == S_IFIFO)
#define S_ISBLK(statbuf.st_mode)	((statbuf.st_mode) &S_IFMT) == S_IFBLK)
#define S_ISSOCK(statbuf.st_mode)	((statbuf.st_mode) &S_IFMT) == S_IFSOCK)

파일 매크로 예제

<ssu_file_macro.c>

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>

int main(int argc, char *argv[])
{
   struct stat file_info;
   char *str;
   int i;

   for (i = 1; i < argc; i++) {
      printf("name = %s, type = ", argv[i]);
      if (lstat(argv[i], &file_info) < 0) {
         fprintf(stderr, "lstat error\n");
         continue;
      }
```

```
 if (S_ISREG(file_info.st_mode))
         str = "regular";
      else if (S_ISDIR(file_info.st_mode))
         str = "directory";
      else if (S_ISCHR(file_info.st_mode))
         str = "character special";
      else if (S_ISBLK(file_info.st_mode))
         str = "block special";
      else if (S_ISFIFO(file_info.st_mode))
         str = "FIFO";
      else if (S_ISLNK(file_info.st_mode))
         str = "symbolic link";
      else if (S_ISSOCK(file_info.st_mode))
         str = "socket";
      else
         str = "unknown mode";
      printf("%s\n", str);
   }
   exit(0);
}
```

실행 결과

```
root@localhost:/home/oslab# ./ssu_file_macro /etc/passwd /home /dev
/dev/cdrom /dev/sda
name = /etc/passwd, type = regular
name = /home, type = directory
….
```

access()

```
#include <unistd.h>

int access(const char *pathname, int mode);

리턴 값: 성공 시 0, 에러 시 –1을 리턴하고 errno 설정
```

- **인자로 지정한 파일(pathname)에 대해 사용자 또는 프로세스 접근가능한지를 판단하는 시스템호출**
- **Linux·Unix에서 프로세스가 파일에 접근하기 위해서 먼저 파일 접근 권한을 검사 => 유효 사용자 ID와 유효 그룹 ID**
- **프로세스의 실제 사용자 ID와 실제 그룹 ID에 의한 접근 가능 여부를 확인하기 위해 사용**
- **<unistd.h>에 정의된, mode 인자로 사용되는 상수**

상수	설명
R_OK	읽기 권한 판정
W_OK	쓰기 권한 판정
X_OK	실행 권한 판정
F_OK	파일 존재 여부 판정

access() 예제 1

<ssu_access_1.c>

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

int main(int argc, char *argv[])
{
   int i;

   if (argc < 2) {
      fprintf(stderr, "usage: %s <file1> <file2> .. <fileN>\n",
argv[0]);
      exit(1);
   }

   for (i = 1; i < argc; i++) {
      if (access(argv[i], F_OK) < 0) {
         fprintf(stderr, "%s doesn't exist.\n", argv[i]);
         continue;
      }
```

```
 if (access(argv[i], R_OK) == 0)
         printf("User can read %s\n", argv[i]);

 if (access(argv[i], W_OK) == 0)
         printf("User can write %s\n", argv[i]);

 if (access(argv[i], X_OK) == 0)
         printf("User can execute %s\n", argv[i]);
   }

   exit(0);
}
```

실행 결과

```
root@localhost:/home/oslab# ./ssu_access_1   /home/oslab    oslab
User can read /home/oslab
User can write /home/oslab
User can execute /home/oslab
oslab doesn't exist.
```

access() 예제. 2

```
<ssu_access_2.c>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define TABLE_SIZE (sizeof(table)/sizeof(*table))

int main(int argc, char *argv[])
{
   struct {
      char *text;
      int mode;
   } table[] = {
      {"exists", 0},
      {"execute", 1},
      {"write", 2},
      {"read", 4}
   };

   int i;
   if (argc < 2) {
      fprintf(stderr, "usage : %s <file>\n", argv[0]);
      exit(1);
   }
```

```
 for (i = 0; i < TABLE_SIZE; i++) {
       if (access(argv[1], table[i].mode) != -1)
          printf("%s -ok\n", table[i].text);
       else
          printf("%s\n", table[i].text);
   }
   exit(0);
}
```

실행 결과

```
root@localhost:/home/oslab# ./ssu_access_2   ssu_access_2
exists -ok
execute -ok
write -ok
read -ok.
```

umask()

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

mode_t umask(mode_t cmask);
리턴 값: 이전의 파일 모드 생성 마스크
```

- **프로세스의 파일 생성 마스크를 생성하고 이전 값을 리턴하는 시스템호출**
- **프로세스 속성 중 umask 값 => 파일 접근 모드 생성 마스크(file mode creation mask) =>시스템의 약한 보안장치**

umask()

플래그	설명
S_IRWXU	파일 소유자의 읽기, 쓰기, 실행 권한을 지정 / 00700
S_IRUSR	파일 소유자의 읽기 권한을 지정 / 00400
S_IWUSR	파일 소유자의 쓰기 권한을 지정 / 00200
S_IXUSR	파일 소유자의 실행 권한을 지정 / 00100
S_IRWXG	그룹 사용자의 읽기, 쓰기, 실행 권한을 지정 / 00070
S_IRGRP	그룹 사용자의 읽기 권한을 지정 / 00040
S_IWGRP	그룹 사용자의 쓰기 권한을 지정 / 00020
S_IXGRP	그룹 사용자의 실행 권한을 지정 / 00010
S_IRWXO	다른 사용자의 읽기, 쓰기, 실행 권한을 지정 / 00007
S_IROTH	다른 사용자의 읽기 권한을 지정 / 00004
S_IWOTH	다른 사용자의 쓰기 권한을 지정 / 00002
S_IXOTH	다른 사용자의 실행 권한을 지정 / 00001

umask() 예제

<ssu_umask.c>

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

#define RW_MODE
(S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP|S_IROTH|S_IWOTH)

int main(void)
{
   char *fname1 = "ssu_file1";
   char *fname2 = "ssu_file2";

   umask(0);

   if (creat(fname1, RW_MODE) < 0) {
      fprintf(stderr, "creat error for %s\n", fname1);
      exit(1);
   }

 umask(S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH);
  if (creat(fname2, RW_MODE) < 0) {
      fprintf(stderr, "creat error for %s\n", fname2);
      exit(1);
   }

   exit(0);
}
```

실행 결과

```
root@localhost:/home/oslab# umask 002
root@localhost:/home/oslab# umask
0002
root@localhost:/home/oslab# ./ssu_umask
root@localhost:/home/oslab# ls -l ssu_file1 ssu_file2
-rw-rw-rw- 1 root root 0 Jan  8 22:09 ssu_file1
-rw------- 1 root root 0 Jan  8 22:09 ssu_file2
root@localhost:/home/oslab# umask
0002
```