

파일 종류 검색[2]

□ 매크로를 이용한 파일 종류 검색

매크로명	매크로 정의	기능
S_ISFIFO(mode)	$((mode) \& 0xF000) == 0x1000$	참이면 FIFO 파일
S_ISCHR(mode)	$((mode) \& 0xF000) == 0x2000$	참이면 문자 장치 특수 파일
S_ISDIR(mode)	$((mode) \& 0xF000) == 0x4000$	참이면 디렉토리
S_ISBLK(mode)	$((mode) \& 0xF000) == 0x6000$	참이면 블록 장치 특수 파일
S_ISREG(mode)	$((mode) \& 0xF000) == 0x8000$	참이면 일반 파일
S_ISLNK(mode)	$((mode) \& 0xF000) == 0xA000$	참이면 심볼릭 링크 파일
S_ISSOCK(mode)	$((mode) \& 0xF000) == 0xC000$	참이면 소켓 파일

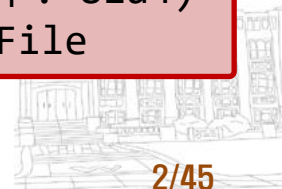
- 각 매크로는 인자로 받은 mode 값을 0xF000과 AND연산 수행
- AND 연산의 결과를 파일의 종류별로 정해진 값과 비교하여 파일의 종류 판단
- 이 매크로는 POSIX 표준



[예제 3-4] 매크로를 이용해 파일 종류 검색하기 (test1.c)

```
01 #include <sys/types.h>
02 #include <sys/stat.h>
03 #include <stdio.h>
04
05 int main(void) {
06     struct stat buf;
07
08     stat("unix.txt", &buf);
09     printf("Mode = %o (16 진 수 : %x)\n", (unsigned int)buf.st_mode,
           (unsigned int)buf.st_mode);
11
12     if(S_ISFIFO(buf.st_mode)) printf("unix.txt : FIFO\n");
13     if(S_ISDIR(buf.st_mode)) printf("unix.txt : Directory\n");
14     if(S_ISREG(buf.st_mode)) printf("unix.txt : Regular File\n");
15
16     return 0;
17 }
```

```
# ex3_4.out
Mode = 100644 (16진수: 81a4)
unix.txt : Regular File
```

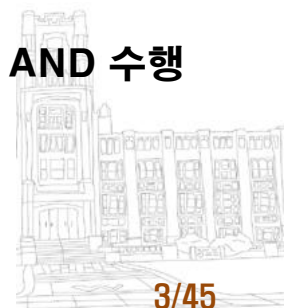


파일 접근 권한 검색[1]

□ 상수를 이용한 파일 접근 권한 검색

상수명	상수값	기능
S_ISUID	0x800	st_mode 값과 AND 연산이 0이 아니면 setuid가 설정됨
S_ISGID	0x400	st_mode 값과 AND 연산이 0이 아니면 setgid가 설정됨
S_ISVTX	0x200	st_mode 값과 AND 연산이 0이 아니면 스틱키 비트가 설정됨
S_IRREAD	00400	st_mode 값과 AND 연산으로 소유자의 읽기 권한 확인
S_IWRITE	00200	st_mode 값과 AND 연산으로 소유자의 쓰기 권한 확인
S_IEXEC	00100	st_mode 값과 AND 연산으로 소유자의 실행 권한 확인

- 소유자의 접근권한 추출과 관련된 상수만 정의
- 소유자 외 그룹과 기타사용자의 접근권한은?
 - st_mode의 값을 왼쪽으로 3비트 이동시키거나 상수값을 오른쪽으로 3비트 이동시켜 AND 수행
 - `st_mode & (S_IRREAD >> 3)`



파일 접근 권한 검색[2]

□ POSIX에서 정의한 접근권한 검색 관련 상수

상수명	상수값	기능
S_IRWXU	00700	소유자 읽기/쓰기/실행 권한
S_IRUSR	00400	소유자 읽기 권한
S_IWUSR	00200	소유자 쓰기 권한
S_IXUSR	00100	소유자 실행 권한
S_IRWXG	00070	그룹 읽기/쓰기/실행 권한
S_IRGRP	00040	그룹 읽기 권한
S_IWGRP	00020	그룹 쓰기 권한
S_IXGRP	00010	그룹 실행 권한
S_IRWXO	00007	기타 사용자 읽기/쓰기/실행 권한
S_IROTH	00004	기타 사용자 읽기 권한
S_IWOTH	00002	기타 사용자 쓰기 권한
S_IXOTH	00001	기타 사용자 실행 권한

시프트 연산없이 직접
AND 연산이 가능한 상수 정의



[예제 3-5] 상수를 이용해 파일 접근 권한 검색하기 (test2.c)

```
01 #include <sys/types.h>
02 #include <sys/stat.h>
03 #include <stdio.h>
04
05 int main(void) {
06     struct stat buf;
07
08     stat("unix.txt", &buf);
09     printf("Mode = %o (16진수: %x)\n", (unsigned int)buf.st_mode,
10         (unsigned int)buf.st_mode);
11
12     if ((buf.st_mode & S_IREAD) != 0)
13         printf("unix.txt : user has a read permission\n");
14
15     if ((buf.st_mode & (S_IREAD >> 3)) != 0)
16         printf("unix.txt : group has a read permission\n");
17
18     if ((buf.st_mode & S_IROTH) != 0)
19         printf("unix.txt : other have a read permission\n");
20
21     return 0;
22 }
```

```
# ex3_5.out
Mode = 100644 (16진수: 81a4)
unix.txt : user has a read permission
unix.txt : group has a read permission
unix.txt : other have a read permission
```

파일 접근 권한 검색[3]

□ 함수를 사용한 파일 접근 권한 검색 : access(2)

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

- path에 지정된 파일이 amode로 지정한 권한을 가졌는지 확인하고 리턴
- 접근권한이 있으면 0을, 오류가 있으면 -1을 리턴
- 오류메시지
 - ENOENT : 파일이 없음
 - EACCESS : 접근권한이 없음
- amode 값
 - R_OK : 읽기 권한 확인
 - W_OK : 쓰기 권한 확인
 - X_OK : 실행 권한 확인
 - F_OK : 파일이 존재하는지 확인



[예제 3-6] access 함수를 이용해 접근 권한 검색하기(test3.c)

```
01  #include <sys/errno.h>
02  #include <unistd.h>
03  #include <stdio.h>
04
05  extern int errno;
06
07  int main(void) {
08      int per;
09
10      if (access("unix.bak", F_OK) == -1 && errno == ENOENT)
11          printf("unix.bak: File not exist.\n");
12
13      per = access("unix.txt", R_OK);
14      if (per == 0)
15          printf("unix.txt: Read permission is permitted.\n");
16      else if (per == -1 && errno == EACCES)
17          printf("unix.txt: Read permission is not permitted.\n");
18
19      return 0;
20  }
```

```
# ls -l unix*
-rw-r--r--  1 root other 24  1월   8일   15:47 unix.txt
# ex3_6.out
unix.bak: File not exist.
unix.txt: Read permission is permitted.
```

파일 접근권한 변경

□ 파일명으로 접근권한 변경 : chmod(2)

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

- path에 지정한 파일의 접근권한을 mode값에 따라 변경
- 접근권한을 더할 때는 OR연산자를, 뺄 때는 NOT연산 후 AND 연산자 사용
 - chmod(path, S_IRWXU);
 - Chmod(path, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH);
 - Mode |= S_IWGRP;
 - Mode &= ~(S_IROTH);
 - chmod(path, mode);

mode 값 설정 후
chmod(path, mode) 잊지말기!

□ 파일 기술자로 접근 권한 변경 : fchmod(2)

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



[예제 3-7] chmod 함수 사용하기 (test4.c)

```
01 #include <sys/types.h>
02 #include <sys/stat.h>
03 #include <stdio.h>
04
05 int main(void) {
06     struct stat buf;
07
08     chmod("unix.txt", S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH);
09     stat("unix.txt", &buf);
10     printf("1.Mode = %o\n", (unsigned int)buf.st_mode);
11
12     buf.st_mode |= S_IWGRP;
13     buf.st_mode &= ~(S_IROTH);
14     chmod("unix.txt", buf.st_mode);
15     stat("unix.txt", &buf);
16     printf("2.Mode = %o\n", (unsigned int)buf.st_mode);
17
18     return 0;
19 }
```

mode값에 따라
권한이 어떻게 바뀌었나?

```
# ls -l unix.txt
-rw-r--r--  1 root  other 24  1월  8일  15:47 unix.txt
# ex3_7.out
1.Mode = 100754
2.Mode = 100770
# ls -l unix.txt
-rwxrwx---  1 root  other 24  1월  8일  15:47 unix.txt
```

- 파일의 정보를 추출하는 프로그램을 작성하라. 정보를 알고 싶은 파일의 이름은 명령행 인자로 받는다.

\$/test unix.txt

File Name : unix.txt

Inode Number :

File Type : Regular File

Permission : rw-r—r—

UID :

Size :

**** if ((buf.st_mode & S_IRUSR) != 0) mode[0] = 'r' ;**



파일 접근권한 변경

□ 파일명으로 접근권한 변경 : umask – test5.c

```
#include <sys/types.h>
#include <sys/stat.h>
mode_t umask(mode_t newmask);
```

- 각 프로세스에는 파일 생성 마스크(file creation mask)가 설정
- 파일이 생성될 때 어떤 모드가 주어지든지 자동적으로 허가 비트를 0으로 만들기 위해 사용 (지정된 허가가 우연히 켜지는 것을 방지)

fd = open(filename, O_CREAT, mode) 는

fd = open(filename, O_CREAT, (~mask)&mode) 와 동일

예) fd = open(filename, O_CREAT, 0644)에 umask = 007을 적용하면
기타의 100 & 000 (~111) = 000

=> fd = open(filename, O_CREAT, 0640)

fd = open(filename, O_CREAT, 0666)에 umask = 022를 적용하면

그룹의 110 & 101 (~010) = 100

기타의 110 & 101 (~010) = 100

=> fd = open(filename, O_CREAT, 0644)

