3주차 2차시 파일 접근

【학습목표】

1. 리눅스 디렉토리의 특징을 설명할 수 있다.
2. 디렉토리 생성, 삭제, 수정 등을 할 수 있다.

학습내용1 : 파일 사용 권한

1. 소유권

- 리눅스는 다중 사용자 지원 운영체제임
- 파일 접근 권한 보호

2. 파일 속성

```
user1@myubuntu:~$ ls -l /etc/hosts
-rw-r--r-- 1 root root 223 11월  8 23:13 /etc/hosts
user1@myubuntu:~$
```

; 총 10의 자리에 파일 종류, 소유자, 소유그룹, 사용자 권한을 표시

① st_mode 값의 구조

16비트												
4bit	3bit			3bit			3bit			3bit		
파일 종류	특수접근 권한			소유자 권한			소유그룹 권한			일반사용자 권한		
	s^1	s^2	t	r	w	x	r	w	x	r	w	x

② 파일 종류

종류	의미	종류	의미
d	디렉터리	-	일반 파일
l	심블릭 링크 파일	b	특수파일(블록 장치)
c	특수 파일(문자 장치)		

③ 특수 접근 권한

표시	의미	2진수표기	8진수표기
s^1	setUID, user 권한으로 실행	100	4
s^2	setGID, group 권한으로 실행	10	2
t	sticky bit, 누구든지 접근 가능	1	1

④ 읽기쓰기실행 권한

표시	권한	2진수표기	8진수표기
r	읽기 가능, 파일을 읽거나 복사 가능	100	4
w	쓰기 가능, 파일을 수정, 이동, 삭제가 가능	10	2
x	실행 가능, 파일을 실행 할 수 있다.	1	1

3. 파일 정보 검색

; i-node 정보 검색 : stat(), lstat(), fstat() 함수 사용

① 파일명으로 파일 정보 검색 : stat(2)
inode에 저장된 파일 정보 검색
path에 검색할 파일의 경로를 지정하고, 검색한 정보를 buf에 저장

```
#include <sys/types.h>
#include <sys/stat.h>
int fstat(int fd, struct stat *buf);
```

path : 정보를 알고자 하는 파일명
buf : 검색한 파일 정보를 저장할 구조체 주소

* stat 구조체

```
struct stat {
    dev_t          st_dev;
    ino_t          st_ino;
    mode_t         st_mode;
    nlink_t        st_nlink;
    uid_t          st_uid;
    gid_t          st_gid;
    dev_t          st_rdev;
    off_t          st_size;
         time_t    st_atime;
    time_t         st_mtime;
    time_t         st_ctime;
    blksize_t      st_blksize;
    blkcnt_t       st_blocks;
    char           st_fstype[_ST_FSTYPSZ];
};
```

* 파일명으로 i-node 정보 검색

```
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
10     printf("Inode = %d\n", (int)buf.st_ino);
11     printf("Mode = %o\n", (unsigned int)buf.st_mode);
12     printf("Nlink = %o\n",(unsigned int) buf.st_nlink);
13     printf("UID = %d\n", (int)buf.st_uid);
14     printf("GID = %d\n", (int)buf.st_gid);
15     printf("SIZE = %d\n", (int)buf.st_size);
16     printf("Atime = %d\n", (int)buf.st_atime);
17     printf("Mtime = %d\n", (int)buf.st_mtime);
18     printf("Ctime = %d\n", (int)buf.st_ctime);
19     printf("Blksize = %d\n", (int)buf.st_blksize);
20     printf("Blocks = %d\n", (int)buf.st_blocks);
21     printf("FStype = %s\n", buf.st_fstype);
22
23     return 0;
24  }
```

```
# ex3_1.out
Inode = 192
Mode = 100644
Nlink = 1
UID = 0
GID = 1
SIZE = 24
Atime = 1231397228
Mtime = 1231397228
Ctime = 1231397228
Blksize = 8192
Blocks = 2
FStype = ufs
```

② 파일 기술자로부터 파일 정보 검색 : fstat(2)
fd로 지정한 파일의 정보를 검색하여 buf에 저장

```
#include <sys/types.h>
#include <sys/stat.h>
int fstat(int fd, struct stat *buf);
```

fd : 열려 있는 파일의 파일 기술자
buf : 검색한 파일 정보를 저장할 구조체 주소

[예제 3-2] 명령행 인자 출력하기 ex3_2.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
10      printf("Inode = %d\n", (int)buf.st_ino);
11      printf("Mode = %o\n", (unsigned int)buf.st_mode);
12      printf("Nlink = %o\n",(unsigned int) buf.st_nlink);
13      printf("UID = %d\n", (int)buf.st_uid);
14      printf("GID = %d\n", (int)buf.st_gid);
15      printf("SIZE = %d\n", (int)buf.st_size);
16      printf("Atime = %d\n", (int)buf.st_atime);
17      printf("Mtime = %d\n", (int)buf.st_mtime);
18      printf("Ctime = %d\n", (int)buf.st_ctime);
19      printf("Blksize = %d\n", (int)buf.st_blksize);
20      printf("Blocks = %d\n", (int)buf.st_blocks);
21      printf("FStype = %s\n", buf.st_fstype);
22
23      return 0;
24  }
```

```
# ex3_2.out
Inode = 192
UID = 0
```

4. 파일 종류 검색

상수 이용한 파일 종류 검색
파일 종류 검색시 상수와 매크로 이용

상수명	상수값(16진수)	기능
S_IFMT	0xF000	st_mode 값에서 파일의 종류를 정의한 부분을 가져옴
S_IFIFO	0x1000	FIFO 파일
S_IFCHR	0x2000	문자 장치 특수 파일
S_IFDIR	0x4000	디렉토리
S_IFBLK	0x6000	블록 장치 특수 파일
S_IFREG	0x8000	일반 파일
S_IFLNK	0xA000	심볼릭 링크 파일
S_IFSOCK	0xC000	소켓 파일

* 상수 이용 파일 검색

```
01  #include <sys/types.h>
02  #include <sys/stat.h>
03  #include <stdio.h>
04
05  int main(void) {
06      struct stat buf;
07      int kind;
08
09      stat("unix.txt", &buf);
10
11      printf("Mode  = %o (16진수: %x)\n", (unsigned  int)buf.st_mode,
(unsigned int)buf.st_mode);
12
13      kind = buf.st_mode & S_IFMT;
14      printf("Kind = %x\n", kind);
15
16      switch (kind) {
17          case S_IFIFO:
18              printf("unix.txt : FIFO\n");
19              break;
20          case S_IFDIR:
21              printf("unix.txt : Directory\n");
22              break;
23          case S_IFREG:
24              printf("unix.txt : Regular File\n");
25              break;
26      }
27
28      return 0;
29  }
```

```
# ex3_3.out
Mode = 100644 (16진수: 81a4)
Kind = 8000
unix.txt : Regular File
```

② 매크로 이용 파일 종류 검색
각 매크로는 인자로 받은 mode 값을 0xF000과 AND연산 수행
AND 연산의 결과를 파일의 종류별로 정해진 값과 비교하여 파일의 종류 판단
이 매크로는 POSIX 표준

매크로명	매크로 정의	기 능
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

```
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 : Regualr File\n");
15
16      return 0;
17  }
```

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

5. 파일 접근 권한 검색

① 상수 이용
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_IREAD >> 3)

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

* 파일 접근 권한 검색

```
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,
              (unsigned int)buf.st_mode);
10
11      if ((buf.st_mode & S_IREAD) != 0)
12          printf("unix.txt : user has a read permission\n");
13
14      if ((buf.st_mode & (S_IREAD >> 3)) != 0)
15          printf("unix.txt : group has a read permission\n");
16
17      if ((buf.st_mode & S_IROTH) != 0)
18          printf("unix.txt : other have a read permission\n");
19
20      return 0;
21  }
```

```
# 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
```

6. 함수 사용 파일 접근 권한 검색 : access(2)

① path에 지정된 파일이 amode로 지정한 권한을 가졌는지 확인하고 리턴
② 접근권한이 있으면 0을, 오류가 있으면 -1을 리턴
③ 오류메시지
ENOENT : 파일이 없음
EACCESS : 접근권한이 없음

④ amode 값
R_OK : 읽기 권한 확인
W_OK : 쓰기 권한 확인
X_OK : 실행 권한 확인
F_OK : 파일이 존재하는지 확인

```c
#include <sys/errno.h>
#include <unistd.h>
#include <stdio.h>

extern int errno;

int main(void) {
    int per;

    if (access("unix.bak", F_OK) == -1 && errno == ENOENT)
        printf("unix.bak: File not exist.\n");

    per = access("unix.txt", R_OK);
    if (per == 0)
        printf("unix.txt: Read permission is permitted.\n");
    else if (per == -1 && errno == EACCES)
       printf("unix.txt: Read permission is not permitted.\n");

    return 0;
}
```

```
# 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.
```

7. 접근 권한 변경

① 파일명으로 접근 권한 변경 : chmod(2)
* path에 지정한 파일의 접근권한을 mode값에 따라 변경
* 접근권한을 더할 때는 OR연산자를, 뺄 때는 NOT연산 후 AND 연산자 사용
chmod(path, S_ORWXU);
chmod(path, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH);
mode |= S_IWGRP;
mode &= ~(S_IROTH);

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

path : 접근 권한을 변경하려는 파일 경로
mode : 접근 권한

② 파일 기술자로 접근 권한 변경 : fchmod(2)

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

fd : 열려 있는 파일의 파일 기술자
mode : 접근 권한

학습내용2 : 심볼릭 링크 및 하드 링크 파일

1. 링크

이미 있는 파일이나 디렉터리에 접근할 수 있는 새로운 이름
같은 파일/디렉터리지만 여러 이름으로 접근할 수 있게 한다
하드링크 : 기존 파일과 동일한 i-node 사용, i-node에 저장된 링크 개수 증가
심볼릭 링크 : 기존 파일에 접근하는 다른 파일 생성(다른 i-node 사용)

2. 하드 링크

① 하드링크 특성
파일에 접근할 수 있는 파일명을 새로 생성
기존 파일과 동일한 i-node 사용
i-node에 저장된 링크 개수 증가

② 하드링크 생성 : link(2)

두 경로는 같은 파일시스템에 존재해야 함

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

existing : 기존 파일의 경로

new : 새로 생성할 링크의 경로

```
01  #include <sys/types.h>
02  #include <sys/stat.h>
03  #include <unistd.h>
04  #include <stdio.h>
05
06  int main(void) {
07      struct stat buf;
08
09      stat("unix.txt", &buf);
10      printf("Before Link Count = %d\n", (int)buf.st_nlink);
11
12      link("unix.txt", "unix.ln");
13
14      stat("unix.txt", &buf);
15      printf("After Link Count = %d\n", (int)buf.st_nlink);
16
17      return 0;
18  }
```

```
# ls -l unix*
-rwxrwx---   1 root     other   24  1월   8일   15:47 unix.txt
# ex3_8.out
Before Link Count = 1
After Link Count = 2
# ls -l unix*
-rwxrwx---   2 root     other   24  1월   8일   15:47 unix.ln
-rwxrwx---   2 root     other   24  1월   8일   15:47 unix.txt
```

3. 심볼릭 링크

① 특성
기존 파일에 접근할 수 있는 다른 파일을 만듬
기존 파일과 다른 i-node 사용

② 생성 : symlink(2)
성공 시 0 리턴, 실패하면 -1 리턴

```
#include <unistd.h>
int symlink(const char *name1, const char *name2);
```

name1 : 기존 파일의 경로
name2 : 새로 생성할 링크의 경로

```
01  #include <sys/types.h>
02  #include <sys/stat.h>
03  #include <unistd.h>
04
05  int main(void) {
06      symlink("unix.txt", "unix.sym");
07
08      return 0;
09  }
```

```
# ls -l unix*
-rwxrwx---    2 root     other         24  1월   8일   15:47 unix.ln
-rwxrwx---    2 root     other         24  1월   8일   15:47 unix.txt
# ex3_9.out
# ls -l unix*
-rwxrwx---    2 root     other         24  1월   8일   15:47 unix.ln
lrwxrwxrwx    1 root     other          8  1월 11일   18:48 unix.sym  ->
unix.txt
-rwxrwx---    2 root     other         24  1월   8일   15:47 unix.txt
```

③ 심볼릭 링크 정보 검색 : lstat(2)
lstat(2) : 심볼릭 링크 자체의 파일 정보 검색
cf) stat() 검색시 원본파일 검색

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

path : 심볼릭 링크의 경로
buf : 검색한 파일 정보를 저장한 구조체 주소

```
01  #include <sys/types.h>
02  #include <sys/stat.h>
03  #include <unistd.h>
04  #include <stdio.h>
05
06  int main(void) {
07      struct stat buf;
08
09      printf("1. stat : unix.txt ---\n");
10      stat("unix.txt", &buf);
11      printf("unix.txt : Link Count = %d\n", (int)buf.st_nlink);
12      printf("unix.txt : Inode = %d\n", (int)buf.st_ino);
13
14      printf("2. stat : unix.sym ---\n");
15      stat("unix.sym", &buf);
16      printf("unix.sym : Link Count = %d\n", (int)buf.st_nlink);
17      printf("unix.sym : Inode = %d\n", (int)buf.st_ino);
18
19      printf("3. lstat : unix.sym ---\n");
20      lstat("unix.sym", &buf);
21      printf("unix.sym : Link Count = %d\n", (int)buf.st_nlink);
22      printf("unix.sym : Inode = %d\n", (int)buf.st_ino);
23
24      return 0;
25  }
```

```
# ls -li unix*
192 -rwxrwx---   2 root     other     24  1월   8일   15:47 unix.ln
202 lrwxrwxrwx   1 root     other      8  1월 11일   18:48 unix.sym->unix.txt
192 -rwxrwx---   2 root     other     24  1월   8일   15:47 unix.txt
# ex3_10.out
1. stat : unix.txt ---
unix.txt : Link Count = 2
unix.txt : Inode = 192
2. stat : unix.sym ---
unix.sym : Link Count = 2
unix.sym : Inode = 192
3. lstat : unix.sym ---
unix.sym : Link Count = 1
unix.sym : Inode = 202
```

④ 심볼릭 링크 내용 읽기 : readlink(2)

심볼릭 링크의 데이터 블록에 저장된 내용 읽기

```
#include <unistd.h>
ssize_t readlink(const char *restrict path, char  *restrict buf,
                 size_t bufsiz);
```

path : 심볼릭 링크의 경로

buf : 읽어온 내용을 저장할 버퍼

bufsiz : 버퍼의 크기

```
01  #include <sys/stat.h>
02  #include <unistd.h>
03  #include <stdlib.h>
04  #include <stdio.h>
05
06  int main(void) {
07      char buf[BUFSIZ];
08      int n;
09
10      n = readlink("unix.sym", buf, BUFSIZ);
11      if (n == -1) {
12          perror("readlink");
13          exit(1);
14      }
15
16      buf[n] = '\0';
17      printf("unix.sym : READLINK = %s\n", buf);
18
19      return 0;
20  }
```

```
# ex3_11.out
unix.sym : READLINK = unix.txt
# ls -l unix.sym
lrwxrwxrwx   1 root other 8  1월  11일    18:48 unix.sym ->unix.txt
```

⑤ 원본 파일 경로 읽기 : realpath(3)

심볼릭 링크가 가리키는 원본 파일의 실제 경로명 출력

```
#include <stdlib.h>
char *realpath(const char *restrict  file_name,
char *restrict resolved_name);
```

file_name : 심볼릭 링크명

resolved_name : 경로명을 저장할 버퍼 주소

```
01  #include <sys/stat.h>
02  #include <stdlib.h>
03  #include <stdio.h>
04
05  int main(void) {
06      char buf[BUFSIZ];
07
08      realpath("unix.sym", buf);
09      printf("unix.sym : REALPATH = %s\n", buf);
10
11      return 0;
12  }
```

```
# ex3_12.out
unix.sym : REALPATH = /export/home/jw/syspro/ch3/unix.txt
```

【학습정리】

1. 리눅스 파일 종류
- d : 디렉터리
- l : 심블릭 링크 파일
- - : 일반 파일
- b : 특수파일(블록 장치)
- c : 특수 파일(문자 장치)

2. 파일 권한
- r : 읽기 가능, 파일을 읽거나 복사 가능, 100(2진수표기), 4(8진수표기)
- w : 쓰기 가능, 파일을 수정, 이동, 삭제가 가능, 10(2진수표기), 2(8진수표기)
- x : 실행 가능, 파일을 실행 할 수 있음, 1(2진수표기), 1(8진수표기)

3. 하드 링크
- 파일에 접근할 수 있는 파일명을 새로 생성
- 기존 파일과 동일한 i-node 사용
- i-node에 저장된 링크 개수 증가

4. 심볼릭 링크
- 기존 파일에 접근할 수 있는 다른 파일을 만듬
- 기존 파일과 다른 i-node 사용