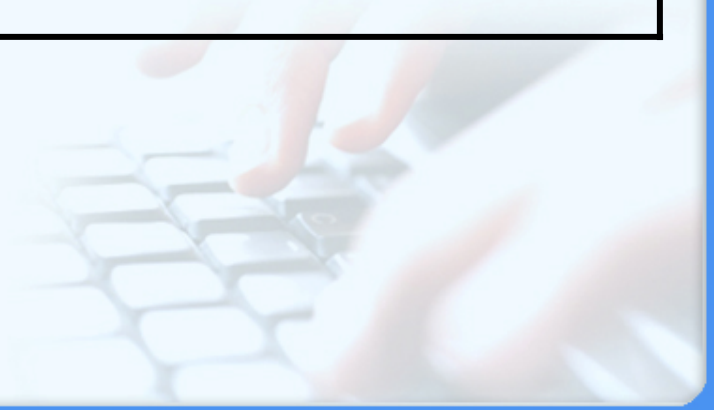
5장 디렉터리 다루기

- 서론
- 예제 프로그램
- 함수
  - mkdir
  - rmdir
  - opendir
  - closedir
  - readdir
  - rewinddir
  - chdir
  - getcwd

한빛미디어(주)

디렉터리를 관리하는 시스템 호출/표준 라이브러리 함수

함수	의미
mkdir	새로운 디렉터리를 작성한다.
rmdir	디렉터리를 삭제한다.
opendir	디렉터리를 파일처럼 개방한다.
closedir	개방한 디렉터리를 닫는다.
readdir	개방된 디렉터리로부터 디렉터리 항목을 읽어온다.
rewinddir	개방된 디렉터리 스트림을 초기화한다.
chdir	디렉터리 경로를 변경한다.
getcwd	현재 작업 디렉터리를 구한다.



【예제】 예제 프로그램(1/3)


```
01 #include <unistd.h>
02 #include <sys/types.h>
03 #include <sys/stat.h>
04 #include <fcntl.h>
05 #include <dirent.h>
06
07 int main()
08 {
09     char buffer[256];
10     DIR *dirp;
11     struct dirent *dentry;
12
13     getcwd(buffer, 256);
14     printf("%s₩n", buffer);
15
16     mkdir("apple", 0755);
17     mkdir("banana", 0755);
18
19     chdir("apple");
```

10 디렉터리 파일 기술자 (파일 기술자와 비슷한 개념이다.)

13 현재 작업 디렉터리를 알아온다

16 새로운 디렉터리를 작성한다.

19 현재 작업 디렉터리를 변경한다.

표준입력 스트림

【예제】 예제 프로그램(2/3)


```
20    getcwd(buffer, 256);
21   printf("%s₩n", buffer);
22
23    close(open("test.txt", O_CREAT | O_RDWR, 0644));
24
25    chdir("..");
26
27    rmdir("apple");
28    rmdir("banana");
29
30    dirp = opendir("apple");
31
32    while(dentry = readdir(dirp))
33        if(dentry->d_ino != 0)
34            printf("%s₩n", dentry->d_name);
```

23 새로운 파일을 생성하고 바로 닫는다.

27 디렉터리를 삭제한다.

30 디렉터리 파일을 개방한다

32 디렉터리 항(directory entry)을 하나씩 읽어서 표준 출력한다.

표준입력 스트림

【예제】 예제 프로그램(3/3)


```
35    rewinddir(dirp);
36
37
38    while(dentry = readdir(dirp))
39        if(dentry->d_ino != 0)
40            printf("%s₩n", dentry->d_name);
41
42
43    closedir(dirp);
44 }
```

35 개방된 디렉터리 파일 내의 읽기 포인터를 최초의 위치로 옮긴다.

38 다시 한번 디렉터리 항을 하나씩 읽어서 표준 출력한다.

43 개방된 디렉터리 파일을 닫는다

표준입력 스트림

실행 결과

- 실행 결과

```
$ ls -l
-rwxr-xr-x    1 usp      student     14931 Oct 19 21:18 ex05-01
$ ./ex05-01
/home/student/usp/
/home/student/usp/apple
.
..
test.txt
.
..
test.txt
$ ls -l
drwxr-xr-x    2 usp      student      4096 Oct 19 21:20 apple
-rwxr-xr-x    1 usp      student     14931 Oct 19 21:18 ex05-01
$ ls -l apple/
-rw-r--r--    1 usp      student         0 Oct 19 21:20 test.txt
$
```

- 새 디렉터리를 생성하고(mkdir), 지정한 빈 디렉터리를 삭제한다.(rmdir)

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

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

pathname	디렉터리의 경로 이름이다.
mode	생성하려는 디렉터리의 초기 접근 권한이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

  - **rmdir은 비어있는 디렉터리만 삭제할 수 있다.**

【예제 5-2】 ex05-02.c


```
01 #include <unistd.h>
02 #include <sys/types.h>
03 #include <sys/stat.h>
04
05 int main()
06 {
07     mkdir("test_dir1", 0755);
08     mkdir("test_dir2", 0755);
09
10     rmdir("test_dir2");
11 }
```

표준입력 스트림

```
$ ex05-02
$ ls -l
drwxr-xr-x    1 usp      student        20  3월 29일  17:24 test_dir1
$
```

- **파일을 개방하는 것처럼 디렉터리를 개방하고 닫는다.**

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

DIR *opendir(const char *name);
int closedir(DIR *dir);
```

name	개방하려는 디렉터리의 경로 이름이다.
dir	닫으려고 하는 개방된 디렉터리에 대한 포인터이다.
반환값	[opendir] 호출이 성공하면 디렉터리 스트림에 대한 DIR형 포인터를 반환하고, 실패하면 NULL을 반환한다. [closedir] 호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

open과 opendir

- 디렉터리는 파일의 특수한 형태
- open으로 디렉터리를 개방할 수 있으나 디렉터리 파일의 논리적인 구조를 알지 못하면 디렉터리 항을 읽어 오는 것이 쉽지 않다.
  - open으로 개방한 파일은 바이트 단위로 읽어온다. 즉, 논리적인 구조가 없다.
- opendir로 디렉터리를 개방하면 다음 절에서 배우게 되는 readdir로 그 내용을 쉽게 읽을 수 있다.

【예제 5-3】 ex05-03.c


```
01 #include <sys/types.h>
02 #include <dirent.h>
03 int main()
04 {
05     DIR *dirp;
06     if((dirp = opendir("test_dir1")) == NULL)
07     {
08         fprintf(stderr, "Error on opening directiry
           test_dir1₩n");
09         exit(1);
10     }
11
12    /* 디렉터리 목록을 읽어오는 부분 */
13
14    closedir(dirp);
15 }
```

개방된 디렉터리로부터 디렉터리 항(directory entry)을 읽어온다

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

struct dirent *readdir(DIR *dirp);
```

dirp	opendir로 개방한 디렉터리에 대한 포인터이다.
반환값	호출이 성공하면 struct dirent 포인터형의 디렉터리 항을 반환하고, 호출이 실패하면 NULL을 반환한다. 더 이상 읽을 디렉터리 항이 없을 경우에도 NULL을 반환한다.

- **readdir은 opendir로 개방한 디렉터리를 읽을 때 사용한다.**

디렉터리 항(directory entry)의 구조

- 아이노드 블록 번호 , 파일 이름 (null 문자로 끝난다.)

```
struct dirent
{
    long d_ino;
    char d_name[NAME_MAX +1];    /* null-terminated */
}
```

- 논리적인 구조

1020	.	₩0				
907	.	.	₩0			
1507	t	e	s	t	1	₩0
0	t	e	m	p	₩0	
1347	a	p	p	l	e	₩0

아이노드 블록 번호

파일 이름

【예제 5-4】 ex05-04.c


```
01 #include <sys/types.h>
02 #include <dirent.h>
03 #include <unistd.h>
04
05 int main()
06 {
07     DIR *dirp;
08     struct dirent *dentry;
09
10     if((dirp = opendir(".")) == NULL)
11         exit(1);
12
13     while( dentry = readdir(dirp)) {
14         if(dentry->d_ino != 0)
15             printf("%s₩n", dentry->d_name);
16     }
17
18     closedir(dirp);
19 }
```

14 d_ino가 0인 항은 삭제된 것이다.

표준입력 스트림

- opendir로 개방한 디렉터리 파일에서 읽기 포인터의 위치를 초기화한다.

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

void rewinddir(DIR *dir);
```

dir	읽기 포인터를 초기화하려는 개방된 파일의 포인터이다.
반환값	없음

- 마치, open으로 개방한 파일에 대해서 lseek(fd, 0, SEEK_SET)을 실행한 것처럼 개방된 디렉터리 파일 내에서 읽기 포인터를 초기화한다.
- opendir로 개방한 디렉터리에서 읽기 포인터는 디렉터리 항 단위로 이동한다.

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

int main()
{
    DIR *dirp;
    struct dirent *dentry;

    if((dirp = opendir(".")) == NULL)
        exit(1);

    printf("존재하는 파일들..₩n");
    while( dentry = readdir(dirp)) {
        if(dentry->d_ino != 0)
            printf("%s₩n", dentry->d_name);
    }

    rewinddir(dirp);

    printf("지워진 파일들..₩n");
    while( dentry = readdir(dirp)) {
        if(dentry->d_ino == 0)
            printf("%s₩n", dentry->d_name);
    }

    closedir(dirp);
}
```

디렉터리에 대한 읽기 포인터를 초기화한다.

삭제된 디렉터리 항은 출력하지 않는다.

현재 작업 디렉터리를 변경한다

```
#include <unistd.h>

int chdir(const char *path);
```

path	변경하려는 새로운 디렉터리 경로이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- **path**
  - **변경하려는 새로운 디렉터리 경로**
  - **절대 경로, 상대 경로 모두 가능하다.**

- **변경된 디렉터리 경로는 현재 프로세스에만 적용되며, 프로세스 종료 후 사용자가 보게 되는 셸에는 영향을 주지 않는다.**

프로세스가 작업 디렉터리를 변경하는 예

- 프로세스가 다른 디렉터리에 있는 다 수의 파일을 개방할 때, 작업 디렉터리를 변경하면 파일의 경로를 짧게 표현할 수 있다.

```
filedes1 = open("temp/file1.txt", O_RDONLY);
filedes2 = open("temp/file2.txt", O_RDONLY);
filedes3 = open("temp/file3.txt", O_RDONLY);
filedes4 = open("temp/file4.txt", O_RDONLY);
```

```
chdir("temp");
filedes1 = open("file1.txt", O_RDONLY);
filedes2 = open("file2.txt", O_RDONLY);
filedes3 = open("file3.txt", O_RDONLY);
filedes4 = open("file4.txt", O_RDONLY);
```

현재 작업 디렉터리를 변경한다

```
#include <unistd.h>

char *getcwd(char *buf, size_t size);
```

buf	현재 작업 디렉터리의 경로를 저장할 버퍼이다.
size	버퍼의 최대 크기이다.
반환값	호출이 성공하면 buf의 포인터를 반환하고, 실패할 경우 NULL을 반환한다.

- **현재 작업 디렉터리의 절대 경로를 지정한 버퍼에 저장한다.**

【예제 5-5】 ex05-05.c


```
01 #include <unistd.h>
02 #define BUF_SIZE    256
03
04 void printcwd()
05 {
06     char buffer[BUF_SIZE];
07
08     if(getcwd(buffer, BUF_SIZE) == NULL)
09         exit(1);
10
11     printf("%s₩n", buffer);
12 }
13 int main()
14 {
15     printcwd();
16     chdir("/usr/include");
17     printcwd();
18     chdir("..");
19     printcwd();
20 }
```

표준입력 스트림