System Programming

Exercise

Week 04. File I/O – Part 3

Directory

- 홈(home) 디렉토리
  - 로그인 후 사용자가 처음 접하게 되는 디렉토리
- 현재 작업 디렉토리 (current working directory)
  - 현재 일을 하고 있는 디렉토리 (상대적 경로명은 여기서 시작)
- 파일/디렉토리의 pathname
  - 절대(absolute) 경로 – 루트 디렉토리(/)에서부터 지정
    - 예) /home/park/book
  - 상대(relative) 경로 – 현재 작업 디렉토리에서부터 지정
    - 예) ./park/book

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Directory (Cont'd)

- 디렉토리에는 아래 정보들이 저장되어 있다.
  - 파일 또는 디렉토리의 inode 번호
  - 파일 또는 디렉토리의 리스트
- Inode 번호는 주어진 파일 또는 디렉토리의 상태 정보와 데이터 블록의 주소를 가지고 있다.
- 모든 디렉토리는 아래 두 디렉토리를 포함하고 있다.
  - 현재 디렉토리를 의미하는 . (점 한 개)
  - 부모 디렉토리를 의미하는 .. (점 두 개)
- 디렉토리는 다른 디렉토리를 포함할 수 있다.

Directory Access Mode

r

- 디렉토리에 포함된 파일 또는 디렉토리 들의 이름 열람 가능
  (단, 이 말이 디렉토리 내에 포함된 파일 또는 디렉토리들의 내용을 열람할 수 있다는 것은 아님)

w

- 디렉토리 내에 새로운 파일 또는 디렉토리를 생성하거나 생성된 파일 또는 디렉토리 제거 가능
  (단, 이 말이 디렉토리 내에 포함된 파일 또는 디렉토리에 어떤 정보를 저장할 수 있다는 의미는 아님)

x

- 명령어 cd 혹은 시스템콜 chdir을 사용하여 디렉토리로 이동할 수 있음
  (어떤 파일을 open 하거나 실행시키기 위해서는 그 파일의 절대 경로에 소속된 모든 리펠토리의 x 접근모드가 필요함)

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

System Call – mkdir, rmdir

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

- 디렉토리의 생성 및 제거
  - pathname : 생성 또는 제거 할 디렉토리 경로
  - mode : 디렉토리에 대한 접근 허가권
- Return value
  - 성공 시: 0
  - 실패 시: -1

- Directory 개념
- mkdir/rmdir
- opendir/
  closedir
- readdir/
  rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Example – mkdir, rmdir

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

int main(int argc, char **argv) {

if (argc!= 2) {
    fprintf(stderr, "Usage : %s dir_name \n", argv[0]);
    return 1;
  }

  if(mkdir(argv[1], 0755)) {
    perror("mkdir error");
    return 1;
  }

  return 0;
}
```

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

System Call – opendir, closedir

```
#include <sys/types.h>
#include <dirent.h>
DIR *opendir(const char *dirname);
int closedir(DIR *dirptr);
```

- 디렉토리 열기 및 닫기
- opendir
  - Return value
    - 성공 시: DIR 유형에 대한 포인터 반환
    - 실패 시: NULL
- closedir
  - Return value
    - 성공 시: 0
    - 실패 시: -1

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

System Call – readdir, rewinddir

```
#include <sys/types.h>
#include <dirent.h>
struct dirent *readdir(DIR *dirptr);
void rewinddir(DIR *dirptr);
```

- 디렉토리 읽기
- readdir
  - Return value
    - [성공 시: struct dirent*]
    - [실패 시: NULL]
- rewinddir
  - no return value

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Example – opendir, closedir, readdir (1)

```c
#include <dirent.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

typedef enum {FALSE, TRUE} Boolean;

static void listFiles (const char *dirpath){
  DIR *dirp;
  struct dirent *dp;
  Boolean isCurrent;

  isCurrent = strcmp(dirpath, ".") == 0;

  dirp = opendir(dirpath);

  if(dirp == NULL){
    fprintf(stderr, "opendir failed on '%s'", dirpath);
    return;
  }

  for(;;){
    errno = 0;
    if((dp = readdir(dirp)) == NULL)
      break;
```

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Example – opendir, closedir, readdir (2)

```
    if(strcmp(dp->d_name, ".") == 0 || strcmp(dp->d_name, "..") == 0)
        continue;
    if(!isCurrent)
      printf("%s/", dirpath);

    printf("%s\n", dp->d_name);
  }

  if(errno != 0) perror("readdir");
  if(closedir(dirp) == -1) perror("closedir");
}

int main(int argc, char *argv[]){
  if(argc > 1 && strcmp(argv[1], "--help") == 0)
     printf("%s [dir...]\n", argv[0]);
  if(argc == 1)
    listFiles(".");

  else
    for(argv++; *argv; argv++)
      listFiles(*argv);
  exit(0);
}
```

```
./main
foo
foo-d
bar-hl
bar2
foo-sh
main-sh
test
dir
file
prog
dir2
main.c
main
```

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

System Call – chdir, getcwd

```
#include <unistd.h>
int chdir(const char *dirpath);
char *getcwd(char *name, size_t size);
```

- chdir
  - 현재 작업 디렉토리를 변경한다
  - Return value
    - 성공 시: 0
    - 실패 시: -1
- getcwd
  - 현재 작업 디렉토리 경로 이름을 찾는다
  - name: 디렉토리 이름을 넣을 장소
  - size: *name의 크기
  - Return value
    - 성공 시: name
    - 실패 시: NULL

- Directory 개념
- mkdir/rmdir
- opendir/
  closedir
- readdir/
  rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Example – chdir, getcwd

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

int main(int argc, char **argv) {

char buf[255];

 if (chdir(argv[1])) {
    perror("error");
    exit(1);
  }

  getcwd(buf, 255);
  printf("현재 작업 디렉토리: %s\n", buf);
  exit(0);
}
```

```
> ./main ..
현재 작업 디렉토리: /home/runner
```

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

System Call – ftw (1)

`struct FTW {int base; int level;}`

```
#include <ftw.h>
int ftw(const char *path, int (*fn)(const char *pathname,
const struct stat *statbuf, int typeflag), int depth);
int nftw(const char *path, int (*fn)(const char *pathname,
const struct stat * statbuf, int typeflag, struct FTW *ftwbuf),
int depth, int flags);
```

- path에 명시된 디렉토리 트리를 재귀적으로 돌며 디렉토리 트리의 각 파일에 대해 한 번씩 fn 함수를 호출
  - path : 열고자 하는 경로
  - fn : 각 파일/디렉토리 탐색 시 불려지는 함수
  - depth : 사용 할 파일 descriptor 개수 (보통 1 사용)
    - 클 수록 디렉토리의 개방 횟수가 줄고 처리 속도가 빠름
    - 한 프로세스가 할당 가능한 최대 수 존재
  - Return value
    - 성공 시: 0
    - 실패 시: 함수 func가 반환하는-1 혹은 0 이 아닌 값

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

System Call – ftw (2)

```
#include <ftw.h>
int ftw(const char *path, int (*fn)(const char *pathname,
const struct stat *statbuf, int typeflag), int depth);
int nftw(const char *path, int (*fn)(const char *pathname,
const struct stat * statbuf, int typeflag, struct FTW *ftwbuf),
int depth, int flags);
```

- ftw()와 nftw()의 *flags*
  - FTW_CHDIR: 디렉토리 내용을 처리하기 전에 chdir() 실행
  - FTW_DEPTH: 후위 운행법 (postorder traversal) 사용
    - 기본 적으로는 전위 운행법 (preorder traversal)을 사용함
  - FTW_MOUNT: 다른 파일 시스템으로 변경하지 않음
    - 트리의 하부 디렉토리 중 하나가 마운트 지점, 즉 다른 파일 시스템일 경우, 해당 디렉토리는 탐색하지 않음
  - FTW_PHYS: 심볼릭 링크를 역참조하지 않음
    - 기본적으로는 역참조 하도록 설계되어 있음

System Call – ftw (3)

```
#include <ftw.h>
int ftw(const char *path, int (*fn)(const char *pathname,
const struct stat *statbuf, int typeflag), int depth);
int nftw(const char *path, int (*fn)(const char *pathname,
const struct stat * statbuf, int typeflag, struct FTW *ftwbuf),
int depth, int flags);
```

- 콜백함수 fn의 *typeflag*
  - FTW_F: 일반 파일 형식
  - FTW_D: 디렉토리
  - FTW_DNR: 읽을 수 없는 디렉토리
    - 따라서 nftw()가 관련된 하부 디렉토리를 탐색할 수 없음
  - FTW_DP: 디렉토리의 후위 운행 (FTW_DEPTH)을 실행
    - 현재 파일과 하부 디렉토리는 이미 처리된 디렉토리로 취급
  - FTW_NS: 심볼릭 링크가 아닌 파일에 대해 stat()호출했지만 실패함
    - statbuf는 정의되지 않음
  - FTW_SL: 심볼릭 링크. nftw()가 FTW_PHYS 플래그로 호출된 경우에만 리턴
  - FTW_SLN: 댕글링 심볼릭 링크. FTW_PHYS 플래그를 사용하지 않은 때만 리턴

- Directory 개념
- mkdir/rmdir
- opendir/
  closedir
- readdir/
  rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Example – ftw

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

int list (const char *name, const struct stat *status, int type) {
  if (type == FTW_NS)
    return 0;
  if (type == FTW_F)
    printf("%-30s\t0%3o\n", name, status->st_mode&0777);
  else
    printf("%-30s*\t0%3o\n", name, status->st_mode&0777);
  return 0;
}

int main(int argc, char **argv) {

  if (argc==1)
     ftw(".", list, 1);

  else
    ftw(argv[1], list, 1);

  return 0;
}
```

```
› ./main
.                              * 0755
./foo                            0666
./foo-d                        * 0755
./bar-hl                         0600
./bar2                           0600
./foo-sh                         0666
./main-sh                        0755
./test                           0644
./dir                          * 0400
./file                           0400
./prog                           0500
./dir2                         * 0755
./main.c                         0644
./main                           0755
```

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Example – nftw (1)

```
#define _XOPEN_SOURCE 600
#include <ftw.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

static void usageError(const char *progName, const char *msg) {
  if (msg != NULL)
    fprintf(stderr, "%s\n", msg);
  fprintf(stderr, "Usage: %s [-d] [-m] [-p] [directory-path]\n", progName);
  fprintf(stderr, "\t-d Use FTW_DEPTH flag\n");
  fprintf(stderr, "\t-m Use FTW_MOUNT flag\n");
  fprintf(stderr, "\t-p Use FTW_PHYS flag\n");
  exit(EXIT_FAILURE);
}

static int dirTree(const char *pathname, const struct stat *sbuf, int type,
struct FTW *ftwb) {
  if (type == FTW_NS) {
    printf("?");
  } else {
    switch (sbuf->st_mode & S_IFMT) {
    case S_IFREG: printf("-"); break;
    case S_IFDIR: printf("d"); break;
```

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Example – nftw (2)

```
    case S_IFCHR: printf("c"); break;
    case S_IFBLK: printf("b"); break;
    case S_IFLNK: printf("l"); break;
    case S_IFIFO: printf("p"); break;
    case S_IFSOCK: printf("s"); break;
    default: printf("?"); break;
    }
  }

  printf(" %s ", (type == FTW_D) ? "D " : (type == FTW_DNR) ? "DNR" :
  (type == FTW_DP) ? "DP " : (type == FTW_F) ? "F " :
  (type == FTW_SL) ? "SL " : (type == FTW_SLN) ? "SLN" :
  (type == FTW_NS) ? "NS " : " ");

  if (type != FTW_NS)
    printf("%7ld ", (long) sbuf->st_ino);
  else
    printf(" ");

  printf(" %*s", 4 * ftwb->level, "");
  printf("%s\n", &pathname[ftwb->base]);
  return 0;
}
```

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Example – nftw (3)

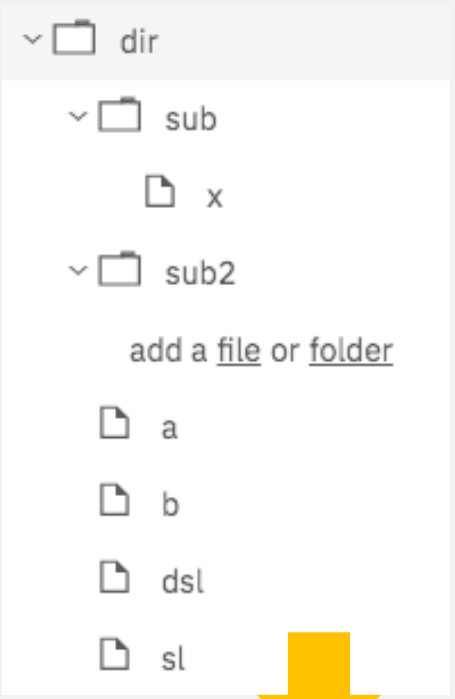
```
int main(int argc, char *argv[]) {
  int flags, opt;

  flags = 0;
  while ((opt = getopt(argc, argv, "dmp")) != -1) {
    switch (opt) {
    case 'd': flags |= FTW_DEPTH; break;
    case 'm': flags |= FTW_MOUNT; break;
    case 'p': flags |= FTW_PHYS; break;
    default: usageError(argv[0], NULL);
    }
  }

  if (argc > optind + 1)
    usageError(argv[0], NULL);

  if (nftw((argc > optind) ? argv[optind] : ".", dirTree, 10, flags) == -1) {
    perror("nftw");
    exit(EXIT_FAILURE);
  }
  exit(EXIT_SUCCESS);
}
```

- Directory 개념
- mkdir/rmdir
- opendir/ closedir
- readdir/ rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

Example – nftw (4)

```
> mkdir dir
> touch dir/a dir/b
> ln -s a dir/sl
> ln -s x dir/dsl
> mkdir dir/sub
> touch dir/sub/x
> mkdir dir/sub2
> chmod 0 dir/sub2
```

디렉토리 생성
일반 파일 생성
심볼릭 링크 생성
댕글링 심볼릭 링크 생성
하부 디렉토리 생성
하부 디렉토리 내부에 파일 생성
하부 디렉토리2 생성
하부 디렉토리2 접근금지 설정




```
> ./main dir
d D         298  dir
- F         299      a
- F         300      b
- F         299      sl
- SLN       299      dsl
d D         303      sub
- F         304          x
d DNR       305      sub2
```

인자 없이 호출한 경우

```
a  b  dsl  sl  sub  sub2
```

```
> ./main -p -d dir
- F         299      a
- F         300      b
l SL        301      sl
l SL        302      dsl
- F         304          x
d DP        303      sub
d DNR       305      sub2
d DP        298  dir
```

후위 탐색, 심볼릭 링크를 따라가지 않음

- Directory 개념
- mkdir/rmdir
- opendir/
  closedir
- readdir/
  rewinddir
- chdir/getcwd
- ftw/nftw
- 실습문제

실습문제

- getcwd()와 동일한 동작을 실행하는 함수를 구현하라.
- 단, getcwd()의 성공/실패 여부와 상관 없이, 탐색을 시작한 것과 동일한 디렉토리에 존재해야 함
- (힌트)
  - 부모 디렉토리(..)의 각 엔트리를 돌면서 opendir()과 readdir()을 사용하면 현재 작업 디렉토리의 이름을 알 수 있음
  - 현재 작업 디렉토리와 동일한 i-노드와 디바이스 번호를 가진 엔트리를 찾으면 현재 디렉토리가 무엇인지 알 수 있음
  - 한 번에 한 단계씩 살펴보고, 스캔을 통해 디렉토리 경로를 구축해볼 것
  - 부모 디렉토리가 현재와 동일하다면(루트 디렉토리의 경우), 검색 종료

THANK YOU

Presented by Hasoo Eun