유닉스 프로그래밍 및 실습

7장. 파일과 디렉토리 관리

1. 파일과 메타자료 (1)

- 파일과 i-node
  - ls –i
- stat 함수군
  - 파일 메타 자료를 얻기 위한 함수군

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

int stat(const char *path, struct stat *buf);
int fstat(int fd, struct stat *buf);
int lstat(const char *path, struct stat *buf);
```

- struct stat
  - 각 필드 내용 확인
  - 실제로 /usr/include 밑의 헤더 파일들 확인할 것!

1. 파일과 메타자료 (2)

- 예제 코드

```
 1 #include <sys/types.h>
 2 #include <sys/stat.h>
 3 #include <unistd.h>
 4 #include <stdio.h>
 5
 6 int main(int argc, char *argv[])
 7 {
 8         struct stat sb;
 9         int ret;
10
11         if (argc < 2) {
12                 fprintf(stderr, "usage: %s <file>\n", argv[0]);
13
14                 return 1;
15         }
16
17         ret = stat (argv[1], &sb);
18         if (ret) {
19                 perror("stat");
20                 return 1;
21         }
22
23         printf("%s is %ld bytes\n", argv[1], sb.st_size);
24
25         return 0;
26 }
```

1. 파일과 메타자료 (3)

- 권한
  - 파일 권한 값 설정 함수

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

int chmod(const char *path, mode_t mode);
int fchmod(int fd, mode_t mode);
```

    - 파일 권한을 mode로 설정 (p.58 권한 값 상수 재확인)
- 소유자
  - 소유자와 그룹 설정 함수

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

int chown(const char *path, uid_t owner, gid_t group);
int lchown(const char *path, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
```

1. 파일과 메타자료 (4)

- 확장속성
  - xattrs
  - 원래 설계에 들어있지 않은 새로운 기능 지원
  - 응용 프로그램에서는 표준인터페이스를 사용하므로 파일시스템에 독립적으로 사용 가능
  - 키와 값
    - 키 : UTF-8 문자열로 이름공간.속성 형태
      - 예) user.mime_type
      - 이름공간 : system | security | trusted | user
  - POSIX의 확장속성 관련 연산
    - 파일과 키로 해당 값 반환
    - 파일과 키, 값을 주어 키에 해당 값 대입
    - 파일을 주면 모든 확장속성 키 목록 반환
    - 파일과 키로 파일에 있는 확장속성 삭제

1. 파일과 메타자료 (5)

- 확장속성 (계속)
  - 확장속성 조회

```
#include <sys/types.h>
#include <attr/xattr.h>

ssize_t getxattr(const char *path, const char *key, void *value, size_t size);
ssize_t lgetxattr(const char *path, const char *key, void *value, size_t size);
ssize_t fgetxattr(int fd, const char *key, void *value, size_t size);
```

  - 확장속성 설정

```
#include <sys/types.h>
#include <attr/xattr.h>

ssize_t setxattr(const char *path, const char *key, void *value,
                 size_t size, int flags);
ssize_t lsetxattr(const char *path, const char *key, void *value,
                 size_t size, int flags);
ssize_t fsetxattr(int fd, const char *key, void *value,
                 size_t size, int flags);
```

1. 파일과 메타자료 (6)

- 확장속성 (계속)
  - 확장속성 나열

```
#include <sys/types.h>
#include <attr/xattr.h>

ssize_t listxattr(const char *path, char *list, size_t size);
ssize_t llistxattr(const char *path, char *list, size_t size);
ssize_t flistxattr(int fd, char *list, size_t size);
```

  - 확장속성 삭제

```
#include <sys/types.h>
#include <attr/xattr.h>

ssize_t removexattr(const char *path, const char *key);
ssize_t lremovexattr(const char *path, const char *key);
ssize_t fremovexattr(int fd, const char *key);
```

2. 디렉토리(1)

- 유닉스의 디렉토리
  - 파일이름과 i-node 번호를 담고 있는 파일
  - 디렉토리 항목(directory entry -> dirent)
- 현재 작업 디렉토리
  - 부모에 상속받는 속성
  - 현재 작업 디렉토리 얻기

```
#include <unistd.h>

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

  - getcwd에서 buf, size를 각각 NULL, 0으로 주었을 때 동작
    - POSIX에서는 동작이 정의되지 않음
    - LINUX의 경우 충분한 버퍼 공간을 알아서 할당하고 그 주소 반환

2. 디렉토리(2)

- 현재 작업 디렉토리 (계속)
  - 리눅스 C 라이브러리의 get_current_dir_name()
  - 오래된 BSD 시스템의 getwd()
- 현재 디렉토리 변경하기

```
#include <unistd.h>

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

  - p.306과 p.307 차이 이해하기

2. 디렉토리(3)

- 디렉토리 생성하기

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

int mkdir(const char *path, mode_t mode);
```

- 디렉토리 삭제하기

```
#include <unistd.h>

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

2. 디렉토리(4)

- 디렉토리 내용 읽기
  - 디렉토리 스트림 열기

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

DIR * opendir(const char *name);
```

    - 디렉토리 스트림을 가리키는 fd 얻기

```
#define _BSD_SOURCE /* 또는 _SVID_SOURCE */
#include <sys/types.h>
#include <dirent.h>

int dirfd(DIR *dir);
```

2. 디렉토리(5)

- 디렉토리 내용 읽기 (계속)
  - 디렉토리 스트림에서 읽기

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

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

    - struct dirent

```
struct dirent {
          ino_t d_ino;
          off_t d_off;
          unsigned short d_reclen;
          unsigned char d_type;
          char d_name[256];
};
```

      - POSIX는 d_name 필드만 요구

2. 디렉토리(6)

- 디렉토리 내용 읽기 (계속)
  - 디렉토리 스트림 닫기

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

int closedir(DIR *dir);
```

  - 예제(p.313)
  - 디렉토리 내용을 읽기 위한 시스템콜
    - readdir(), getdents()
    - C 라이브러리에서 제공하는 opendir(), readdir(), closedir() 사용 권고

3. 링크

- 디렉토리에서 이름과 i-node를 사상하는 개념
- 하드 링크

```
#include <unistd.h>

int link(const char *oldpath, const char *newpath);
```

- 심볼릭 링크
  - 전혀 새로운 파일 유형
  - 특수 파일로 다른 파일을 가리키는 경로 이름 포함

```
#include <unistd.h>

int symlink(const char *oldpath, const char *newpath);
```

- 링크 끊기

```
#include <unistd.h>

int unlink(const char *pathname);
```

4. 파일 복사와 이동

- 복사
- 이동
  - 파일이 존재하는 디렉토리 항목의 이름을 변경하는 행위

```
#include <stdio.h>

int rename(const char *oldpath, const char *newpath);
```

5. 디바이스 노드

- 개요
  - 특수파일이며, 응용과 디바이스 드라이버를 연결하는 인터페이스 제공
  - 응용이 디바이스 노드를 대상으로 열기, 읽기, 쓰기, 닫기 등을 요청하면 커널은 이들 요청을 디바이스 드라이버에 전달
  - 유닉스 시스템에서 하드웨어에 접근하는 표준 메커니즘
- 특수 디바이스 노드
  - /dev/null
  - /dev/zero
  - /dev/full
  - 테스트, 부하를 걸지 않고 원하지 않는 입출력 버리기 등에 활용
- 난수 발생기
  - /dev/random
    - 난수발생기에서 이용할 초기값, 키 생성, 강한 엔트로피가 필요한 과업 등에 적합
  - /dev/urandom
    - 강한 엔트로피가 필요 없는 일반적인 값

6. 대역외 통신

- 자료 외부 스트림에서 통신하고자 할 때 이용
  - 예: 직렬 포트 마지막 정보 읽기, 패리티 설정하기 등

```
#include <sys/ioctl.h>

int ioctl(int fd, int request, …);
```

  - P.327 예제 (CDROMEJECT 요청)

7. 파일 사건 감시하기

- 파일관련 사건을 다른 응용에 전달
  - 기존 시그널 기반 dnotify의 문제 개선
- inotify 초기화하기
- 감시
- 새로운 감시 추가
- inotify 사건
  - inotify 사건 읽기
  - 고급 inotify
  - 이동 사건을 하나로 엮기
- 고급 감시 옵션
- inotify 감시 삭제하기
- 사건큐 크기 얻기
- inotify 인스턴스 감시 소멸하기

과제

1. 명령줄에서 파일 이름을 받아 파일에 대한 정보를 자세히 출력하는 프로그램 작성
   - p.285 예제와 p.283 구조체 정보 활용
   - 시간 관련 함수 이용하고자 하는 경우 p.441 참조
2. 명령줄에서 디렉토리 이름을 받아 그 디렉토리 안의 파일에 대한 정보를 자세히 출력하는 프로그램 작성
   - 1번 문제 활용

- 제출기한 : 11월 8일 자정