인덕대학교

파일 시스템 I
- 기본개념과 응용

컴퓨터소프트웨어학과

김병국 교수

학습목표

- 실습을 통해 파일 시스템 사용법을 익힌다.
- 파일 접근을 위한 기본적인 시스템함수를 안다.
- 저수준 파일 처리를 할 수 있다.
- 고수준 파일 처리를 할 수 있다.

목차

1. 응용 실습

인덕대학교

❑ 파일 처리 : 생성 및 시간변경

- 명령: touch [-t YYYYMMDDhhmm.ss] filename
  - 파일의 시간 정보 수정을 주목적으로 함
  - 지정한 파일이 없을 때에는 파일을 생성
  - 시간 생성을 위한 옵션으로 “-t”를 사용
    - 예: touch -t 202104200910.00
  - 일반적 옵션:
    - -a : 접근 시간만 변경
    - -m : 수정 시간만 변경

- 시간 확인
  - 명령 예:
    - $ ls -l --full-time --time=atime
    - $ ls -l --full-time --time=ctime
    - $ ls -l --full-time --time=birth

❑파일 처리 : 타입 보기

- 명령: file filename
  - 지정한 파일의 타입을 출력

```
[kali@kali: ~]$ls hello*
hello  hello.c
[kali@kali: ~]$cat hello.c
#include <stdio.h>

int main()
{
        printf("Hello World\n");
        return 0;
}

[kali@kali: ~]$file hello*
hello:   ELF 64-bit LSB pie executable, x86-64, ve
-x86-64.so.2, BuildID[sha1]=e905fd610172e547ea572f
hello.c: C source, ASCII text
[kali@kali: ~]$
```

1. 응용 실습

인덕대학교

❑ 파일 처리 : 복사

- 명령: cp [원본] [경로 또는 대상파일]
  - 지정한 “원본” 파일을 “대상파일” 또는 “경로” 하단에 복사
  - 대표적인 옵션:
    - -a : 속성까지 복사
    - -R, -r : 디렉터리 하단까지 복사
    - -b : 동일한 파일이 있으면 백업본 생성(~)
    - -i : 덮어쓰기를 할 지를 묻기

사용 형식	동 작
cp 파일1 파일2	파일1을 파일2로 복사.
cp 파일들 디렉터리	파일들을 디렉터리 밑에 같은 이름으로 복사.
cp -r 디렉터리1 디렉터리2	디렉터리1을 디렉터리2로 복사. 디렉터리1의 파일도 모두 복사됨.

1. 응용 실습

인덕대학교

❑ 파일 처리 : 링크

- 명령: ln [-s] [원본] [대상파일]
  - 원본 파일을 대상파일로 링크
  - 파일의 내용은 동일하며 껍데기만 복사되는 원리
    - 윈도우의 바로가기 아이콘과 유사한 개념
  - 옵션:
    - -s: 심볼릭 링크로 파일을 생성
    - “” : 하드 링크로 파일을 생성

- 링크의 타입
  - 하드 링크(hard link)
    - i-node는 동일하나 껍데기가 다른 파일
  - 심볼릭 링크(symbolic link)
    - i-node는 다르나 껍데기가 다른 파일의 이름을 참조

❑파일 처리 : 이동

- 명령: mv [원본] [대상]
  - 지정한 원본파일을 대상파일로 변경
  - 파일이름 변경 기능도 수행

❑파일 처리 : 삭제

- 명령: rm [옵션] [파일명 또는 디렉터리명]
- 지정한 파일이나 디렉터리를 삭제
- 옵션
  - -i: 확인 후 삭제
  - -f: 무조건 삭제
  - -r: 디렉터리 삭제

❑디렉터리 처리 : 현재 위치 확인

- 명령: pwd
  - Present Working Directory
  - 현재 작업하고 있는 디렉터리 위치를 출력

❑디렉터리 처리 : 위치 이동

- 명령: cd [경로: 절대/상대]
  - “.” 현재 위치
  - “..” 상위 위치
  - “” 자신의 홈 위치로 되돌아가기
  - “~계정이름 “ 지정한 계정의 홈 위치로 이동
  - “-” 직전 작업 위치로 이동

❑디렉터리 처리 : 생성

- 명령: mkdir [디렉터리명]
  - 새로운 디렉터리 생성
  - 옵션
    - -p: 디렉터리 생성에 명시된 하위 디렉터리도 함께 생성

❑디렉터리 처리 : 삭제

- 명령: rmdir [디렉터리명]
  - 지정한 디렉터리를 삭제
  - 지정한 디렉터리에는 엔트리가 없어야 함

1. 응용 실습

인덕대학교

❑ 디렉터리 처리 : 사용량 확인

- 명령: du [경로]
  - 지정한 경로의 사용량을 출력
  - 일반적 옵션:
    - -h : 사용자 인식을 고려한 사이즈로 표시(1K, 234M, 2G 등)
    - -k : 킬로바이트(kilobyte) 단위로 표시
    - -m : 메가바이트(megabyte) 단위로 표시
    - -s : 요약정보(summary)만 표시

```
[kali@kali: ~]$ls
Desktop    Downloads  hello.c  Music    OperatingSystem  Public     Videos
Documents  hello      ls.txt   Network  Pictures         Templates
[kali@kali: ~]$du OperatingSystem/ -h
76K     OperatingSystem/04_1
40K     OperatingSystem/05_1
52K     OperatingSystem/07_2
28K     OperatingSystem/04_3
92K     OperatingSystem/07_1
184K    OperatingSystem/06_1
28K     OperatingSystem/03_2
504K    OperatingSystem/
[kali@kali: ~]$
```

❑ 디스크 처리 : 사용량 확인

- 명령: df
  - 디스크의 사용량을 출력
  - 일반적 옵션:
    - -h : 사용자 인식을 고려한 사이즈로 표시(1K, 234M, 2G 등)
    - -k : 킬로바이트(kilobyte) 단위로 표시
    - -m : 메가바이트(megabyte) 단위로 표시

```
[kali@kali: ~]$df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           394M  2.4M  391M   1% /run
/dev/sda1        78G  8.8G   65G  12% /
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           394M   56K  394M   1% /run/user/1000
[kali@kali: ~]$
```

❑디스크 처리 : 블록장치 정보 출력

- 명령: lsblk
  - 시스템의 블록장치의 정보를 출력
  - 일반적 옵션 :
    - -f : 파일시스템을 같이 표시

```
[kali@kali: ~]$lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   80G  0 disk
├─sda1   8:1    0   79G  0 part /
├─sda2   8:2    0    1K  0 part
└─sda5   8:5    0  975M  0 part [SWAP]
sr0     11:0    1 1024M  0 rom
[kali@kali: ~]$
```

2. 저수준 파일 접근

인덕대학교

❑ 유닉스 파일시스템

- 유닉스는 프로세스, 장치 및 파일들에 대하여 모두 파일시스템으로 관리
- 시스템의 모든 자원의 제어는 파일접근으로 가능
- 파일 입출력(I/O)를 위한 기본적인 시스템 함수들을 제공
- 저수준 파일 처리: 파일 기술자를 통해 접근하는 방식

- 대표적 파일 처리용 시스템 함수들
  - open() : 파일 및 장치 접근
  - close() : 접근 해제
  - read() : 파일이나 장치로부터 읽기
  - write() : 파일이나 장치에 쓰기
  - lseek() : 지정한 위치로 오프셋을 이동
  - creat() : 파일 또는 장치를 생성
  - unlink() : 파일이나 장치를 삭제
  - ioctl() 또는 fcntl() : 파일이나 장치의 속성을 제어

3. 저수준 파일 접근과 해제 (1/5)

❑ 파일 접근

- 함수: open()
  - 파일 및 장치 접근
  - 프로세스를 위한 파일 기술자를 생성
- 인자
  - *pathname : 접근할 파일 또는 장치명
  - flags : 접근 모드(예: O_RDONLY, O_WRONLY, O_RDWR, O_CREAT 등)
  - (옵션) mode : 생성 권한(파일 생성 시)
- 결과 값:
  - 성공 : 양수
  - 실패 : -1

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

int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

[open() 함수의 프로토타입]

3. 저수준 파일 접근과 해제 (2/5)

❑파일 접근

- 함수: open()
  - 접근 모드(flags)

Flags	기능
O_RDONLY	읽기 전용 접근
O_WRONLY	쓰기 전용 접근
O_RDWR	읽기/쓰기 접근
O_APPEND	파일 기록(write() 함수 호출)시 가장 마지막에 추가됨
O_CREAT	파일이 없으면 생성
O_SYNC	쓰기 명령 수행해 운영체제 캐쉬가 아닌 I/O에 바로 기록
O_TRUNC	이미 파일에 내용이 있으면, 모두 지우고 새로 작성

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

int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

[open() 함수의 프로토타입**]**

3. 저수준 파일 접근과 해제 (3/5)

❑파일 접근

- 함수: open()
  - 생성 권한(mode)

mode	값	기능
S_IRUSR	0400	사용자 읽기
S_IWUSR	0200	사용자 쓰기
S_IXUSR	0100	사용자 실행
S_IRGRP	0040	그룹 읽기
S_IWGRP	0020	그룹 쓰기
S_IXGRP	0010	그룹 실행
S_IROTH	0004	기타 읽기
S_IWOTH	0002	기타 쓰기
S_IXOTH	0001	기타 실행

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

int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

[open() 함수의 프로토타입**]**

3. 저수준 파일 접근과 해제 (4/5)

❑ 접근 해제

- 함수: close()
  - 지정한 파일 기술자에 해당하는 접근을 해제
- 인자
  - fd : 접근 해제할 파일 기술자
- 결과 값:
  - 성공 : 0
  - 실패 : -1

```
#include <unistd.h>

int close(int fd);
```

[close() 함수의 프로토타입**]**

3. 저수준 파일 접근과 해제 (5/5)

❑실습

- 파일 접근 및 생성

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

int
main(int argc, char* argv[])
{
    int nFd = -1;

    if (argc != 2)
    {
        perror("we need an argument.\n");
        return -1;
    }

    nFd = open(argv[1], O_RDWR | O_CREAT, 0644);

    printf("FD is %d\n", nFd);

    close(nFd);

    return 0;
}
```

1

2

4. 저수준 파일 파일 읽기 (1/2)

❑ 파일 읽기

- 함수 : read()
  - 파일의 내용을 주어진 길이(개수)만큼 읽음
- 인자 :
  - fd : 읽어 들일 파일의 기술자
  - *buf : 읽어 들인 파일의 내용
  - count : 읽을 최대 길이(일반적으로 buf의 크기)
- 결과 값:
  - 성공 : 읽어 들인 내용의 길이
  - 실패 : -1

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
```

[read() 함수의 프로토타입**]**

4. 저수준 파일 파일 읽기 (2/2)

❑ 실습

- 파일 내용 읽기

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

int
main(int argc, char* argv[])
{
    int nFd = -1;
    char pBuf[BUFSIZ] = { 0, };
    int nLen = 0;

    if (argc != 2)
    {
        perror("we need an argument.\n");
        return -1;
    }
```

①

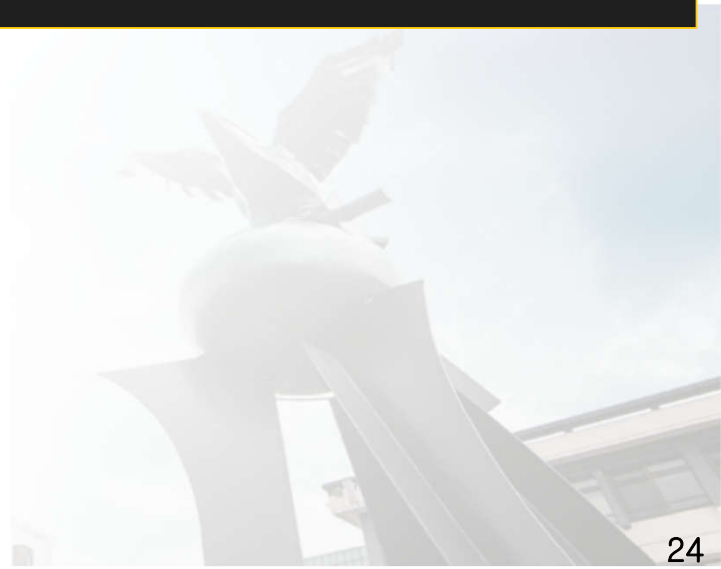
```c
    nFd = open(argv[1], O_RDONLY);

    while ((nLen = read(nFd, pBuf, BUFSIZ)) > 0)
    {
        printf("%s", pBuf);
    }

    close(nFd);

    return 0;
}
```

②



5. 저수준 파일 파일 쓰기 (1/2)

❑ 파일 쓰기

- 함수 : write()
  - 내용을 주어진 길이(개수)만큼 파일에 기록
- 인자 :
  - fd : 파일의 기술자
  - *buf : 기록할 내용
  - count : 기록할 내용의 길이(일반적으로 buf의 크기)
- 결과 값:
  - 성공 : 기록한 내용물 길이
  - 실패 : -1

```
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);
```

[write() 함수의 프로토타입]

5. 저수준 파일 파일 쓰기 (2/2)

❑ 실습

- 파일 복사

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

int
main(int argc, char* argv[])
{
    int nFd1 = -1;
    int nFd2 = -1;
    char pBuf[BUFSIZ] = { 0, };
    int nLen = 0;

    if (argc != 3)
    {
        perror("we need two arguments.\n");
        return -1;
    }
    nFd1 = open(argv[1], O_RDONLY);
    nFd2 = open(argv[2], O_WRONLY | O_CREAT, 0644);

    while ((nLen = read(nFd1, pBuf, BUFSIZ)) > 0)
    {
        if (write(nFd2, pBuf, nLen) < 0)
        {
            perror("Writting Failed.\n");
            break;
        }
    }
    close(nFd2);
    close(nFd1);

    return 0;
}
```

6. 저수준 파일 파일 생성과 삭제 (1/3)

❑ 파일 생성

- 함수: creat()
  - 지정한 이름의 파일을 생성
- 인자:
  - *pathname : 생성할 파일의 이름
  - mode : 접근 권한(open() 함수의 mode와 동일)
- 결과 값:
  - 성공: 파일 기술자
  - 실패: -1

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

int creat(const char *pathname, mode_t mode);
```

[creat() 함수의 프로토타입]

mode	값	기능
S_IRUSR	0400	사용자 읽기
S_IWUSR	0200	사용자 쓰기
S_IXUSR	0100	사용자 실행
S_IRGRP	0040	그룹 읽기
S_IWGRP	0020	그룹 쓰기
S_IXGRP	0010	그룹 실행
S_IROTH	0004	기타 읽기
S_IWOTH	0002	기타 쓰기
S_IXOTH	0001	기타 실행

[mode 접근권한]

6. 저수준 파일 파일 생성과 삭제 (2/3)

❑ 파일 삭제

- 함수: unlink()
  - 지정한 이름의 파일을 생성
- 인자:
  - *pathname : 삭제할 파일의 이름
- 결과 값:
  - 성공: 0
  - 실패: -1

```
#include <unistd.h>

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

[unlink() 함수의 프로토타입**]**

6. 저수준 파일 파일 생성과 삭제 (3/3)

❑실습

1

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

int
main(int argc, char* argv[])
{
    int nFd = -1;

    if (argc != 2)
    {
        perror("we need an argument.\n");
        return -1;
    }
```

2

```c
    nFd = creat(argv[1], 0644);

    write(nFd, "Hello", 5);

    close(nFd);

    printf("Press Enter Key to continue.\n");

    getchar();

    printf("Deleting %s\n", argv[1]);

    unlink(argv[1]);

    printf("Done.\n");

    return 0;
}
```

❑ 고수준 파일 처리

- FILE 구조체를 통하여 파일을 접근하는 방식

- 대표적 파일 처리용 시스템 함수들
  - fopen() : 파일 및 장치 접근
  - fclose() : 접근 해제
  - fread(), fgets() : 파일이나 장치로부터 읽기
  - fwrite(), fputs() : 파일이나 장치에 쓰기
  - fseek() : 지정한 위치로 오프셋을 이동
  - fgetpos() : 현재 파일의 오프셋을 추출
  - fsetpos() : 현재 파일의 오프셋을 설정

8. 고수준 파일 처리 함수 (1/4)

❑ 파일 접근

- 함수: fopen()
  - 고수준 방식의 파일 및 장치 접근
  - 파일 접근을 위한 FILE 구조체 포인터 생성
- 인자
  - *pathname : 접근할 파일 또는 장치명
  - *mode : 접근 모드
- 결과 값:
  - 성공 : FILE 구조체 포인터
  - 실패 : NULL

mode	기능
“r”	읽기 전용 접근(시작점: 처음)
“r+”	읽기 & 쓰기 접근(시작점: 처음)
“w”	쓰기 전용 접근(시작점: 처음)
“w+”	읽기 & 쓰기 접근(시작점: 처음), 파일이 없으면 생성
“a”	쓰기 전용 접근(시작점: 끝), 파일이 없으면 생성
“a+”	읽기 & 쓰기 접근(시작점: 읽기->처음, 쓰기->끝), 파일이 없으면 생성

```
#include <stdio.h>

FILE *fopen(const char *pathname, const char *mode);
```

[fopen() 함수의 프로토타입**]**

8. 고수준 파일 처리 함수 (2/4)

❑ 접근 해제

- 함수: fclose()
  - 지정한 파일 포인터에 해당하는 접근을 해제
- 인자
  - *stream : 접근 해제할 FILE 포인터
- 결과 값:
  - 성공 : 0
  - 실패 : EOF (-1)

```
#include <stdio.h>

int fclose(FILE *stream);
```

[fclose() 함수의 프로토타입]

8. 고수준 파일 처리 함수 (3/4)

❑ 파일 읽기

- 함수: fread()
  - 지정한 파일 포인터에서 데이터 읽기
- 인자
  - *ptr : 기록될 공간
  - size : 각 원소의 크기(바이트)
  - nmemb : 데이터(원소들)의 개수
  - *stream : 접근 할 FILE 포인터
- 결과 값:
  - 성공 : 읽을 데이터의 개수
  - 실패 : 0

❑ 파일 쓰기

- 함수: fwrite()
  - 지정한 파일 포인터에 데이터를 기록
- 인자
  - fread()의 패턴과 동일
- 결과 값:
  - 성공 : 기록한 데이터의 개수
  - 실패 : 0

```
#include <stdio.h>

size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);

size_t fwrite(const void *ptr, size_t size, size_t nmemb,
              FILE *stream);
```

[fread()와 fwrite() 함수의 프로토타입]

8. 고수준 파일 처리 함수 (4/4)

인덕대학교

❑ 파일 읽기(라인 단위)

- 함수: fgets()
  - 지정한 파일 포인터에서 데이터 읽기
- 인자
  - *s : 기록될 공간
  - size : 최대 크기
  - *stream : 접근 할 FILE 포인터
- 결과 값:
  - 성공 : 기록된 공간의 주소
  - 실패 : NULL

❑ 파일 쓰기(라인 단위)

- 함수: fputs()
  - 지정한 파일 포인터에 데이터를 기록
- 인자
  - fgets()의 패턴과 동일
- 결과 값:
  - 성공 : 기록한 데이터의 개수
  - 실패 : -1

```
#include <stdio.h>

char *fgets(char *s, int size, FILE *stream);

int fputs(const char *s, FILE *stream);
```

[fread()와 fwrite() 함수의 프로토타입]

9. 고수준 파일 처리 예

❑실습

```c
#include <stdio.h>

int
main(int argc, char* argv[])
{
    int i = 0;
    FILE* fp1 = NULL;   // file pointer 1
    FILE* fp2 = NULL;   // file pointer 2

    char buffer[BUFSIZ];
    int buffer_len = 0;

    if (argc != 3) {
        printf("Usage: %s file1 file2\n", argv[0]);
        return -1;
    }

    fp1 = fopen(argv[1], "r");
    fp2 = fopen(argv[2], "w");
```

1

```c
    while (fgets(buffer, BUFSIZ, fp1) != NULL) {
        printf("%02d: %s", i++, buffer);
        fputs(buffer, fp2);
    }

    fclose(fp2);
    fclose(fp1);

    return 0;
}
```

2

수고하셨습니다.