Error and Command Line Arguments

Jo, Heeseung

학습목표

오류처리 함수

동적 메모리 할당

명령행 인자

오류 처리 함수[1]

오류 메시지 출력 : perror(3)

```
#include <stdio.h>
void perror(const char *s);
```

```
01  #include <sys/errno.h>
02  #include <unistd.h>
03  #include <stdlib.h>
04  #include <stdio.h>
05
06  int main(void) {
07      if (access("unix.txt", R_OK) == -1) {
08          perror("unix.txt");
09          exit(1);
10      }
11
12      return 0;
13  }
```

```
# ex1_4.out
unix.txt: No such file or directory
```

동적 메모리 할당[1]

메모리할당 : malloc(3)

- 인자로 지정한 크기의 메모리 할당

```
#include <stdlib.h>
void *malloc(size_t size);
```

```
char *ptr
ptr = malloc(sizeof(char) * 100);
```

메모리할당과 초기화 : calloc(3)

- nelem * elsize 만큼의 메모리를 할당하고, 0으로 초기화

```
#include <stdlib.h>
void *calloc(size_t nelem, size_t elsize);
```

```
char *ptr
ptr = calloc(10, 20);
```

동적 메모리 할당[2]

메모리 추가 할당: realloc(3)

```
#include <stdlib.h>
void *realloc(void *ptr, size_t size);
```

- 이미 할당받은 메모리(ptr)에 size 크기의 메모리를 추가로 할당

```
char *ptr, *new;
ptr = malloc(sizeof(char) * 100);
new = realloc(ptr, 100);
```

메모리 해제 : free(3)

```
#include <stdlib.h>
void free(void *ptr);
```

- 사용을 마친 메모리 반납

명령행 인자[1]

명령행 : 사용자가 명령을 입력하는 행

- 명령행 인자 : 명령을 입력할 때 함께 지정한 인자(옵션, 옵션인자, 명령인자 등)
- 명령행 인자의 전달 : main 함수로 자동 전달

```
int main(int argc, char *argv[])
```

```
01  #include <stdio.h>
02
03  int main(int argc, char *argv[]) {
04      int n;
05
06      printf("argc = %d\n", argc);
07      for (n = 0; n < argc; n++)
08          printf("argv[%d] = %s\n", n, argv[n]);
09
10      return 0;
11  }
```

```
# ex1_6.out -h 100
argc = 3
argv[0] = ex1_6.out
argv[1] = -h
argv[2] = 100
```

File I/O

Jo, Heeseung

학습목표

유닉스에서 파일 입출력의 특징을 이해

저수준 파일 입출력 함수를 사용

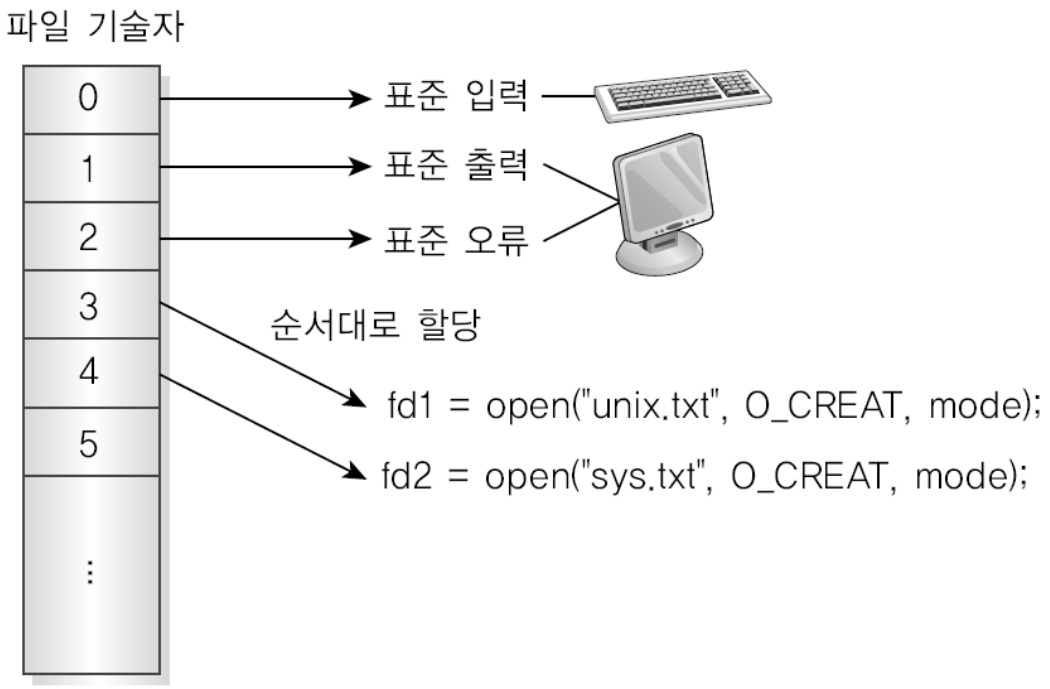
(고수준 파일 입출력)

저수준 파일 입출력

파일 기술자 (file descriptor)

파일 기술자 (file descriptor)

- 현재 열려있는 파일을 구분하는 정수값
- 저수준 파일 입출력에서 열린 파일을 참조하는데 사용
  - 0 : 표준 입력
  - 1 : 표준 출력
  - 2 : 표준 오류



[그림 2-1] 파일 기술자 할당

파일 생성과 열고 닫기[1]

파일 열기: open(2)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *path, int oflag [, mode_t mode]);
```

- path에 지정한 파일을 oflag에 지정한 플래그 값에 따라 열고 파일기술자를 리턴
- oflag 값

종류	기능
O_RDONLY	파일을 읽기 전용으로 연다.
O_WRONLY	파일을 쓰기 전용으로 연다.
O_RDWR	파일을 읽기와 쓰기가 가능하게 연다.
O_CREAT	파일이 없으면 파일을 생성한다
O_EXCL	O_CREAT 옵션과 함께 사용할 경우 기존에 없는 파일이면 파일을 생성하지만, 파일이 이미 있으면 파일을 생성하지 않고 오류 메시지를 출력한다.
O_APPEND	파일의 맨 끝에 내용을 추가한다.
O_TRUNC	파일을 생성할 때 이미 있는 파일이고 쓰기 옵션으로 열었으면 내용을 모두 지우고 파일의 길이를 0으로 변경한다.
O_NONBLOCK/O_NDELAY	비블로킹(Non-blocking) 입출력
O_SYNC/O_DSYNC	저장장치에 쓰기가 끝나야 쓰기 동작을 완료

파일 생성과 열고 닫기[2]

파일 열기: open(2)

- mode: 파일 접근권한 지정, 0644같이 숫자나 플래그 값으로 지정 가능

플래그	모드	설명
S_IRWXU	0700	소유자 읽기/쓰기/실행 권한
S_IRUSR	0400	소유자 읽기 권한
S_IWUSR	0200	소유자 쓰기 권한
S_IXUSR	0100	소유자 실행 권한
S_IRWXG	0070	그룹 읽기/쓰기/실행 권한
S_IRGRP	0040	그룹 읽기 권한
S_IWGRP	0020	그룹 쓰기 권한
S_IXGRP	0010	그룹 실행 권한
S_IRWXO	0007	기타 사용자 읽기/쓰기/실행 권한
S_IROTH	0004	기타 사용자 읽기 권한
S_IWOTH	0002	기타 사용자 쓰기 권한
S_IXOTH	0001	기타 사용자 실행 권한

```
mode=S_IRUSR ¦ S_IWUSR;
```

파일 생성과 열고 닫기[3]

파일 닫기: close(2)

```
#include <unistd.h>
int close(int fildes);
```

- 프로세스에서 열 수 있는 파일 개수가 제한되어 있으므로 파일의 사용이 끝나면 닫아야 함

새 파일 열고 닫기

```
01  #include <sys/types.h>
02  #include <sys/stat.h>
03  #include <fcntl.h>
04  #include <unistd.h>
05  #include <stdlib.h>
06  #include <stdio.h>
07
08  int main(void) {
09      int fd;
10      mode_t mode;
11
12      mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
13
14      fd = open("unix.txt", O_CREAT, mode);
15      if (fd == -1) {
16          perror("Creat");
17          exit(1);
18      }
19      close(fd);
20
21      return 0;
22  }
```

```
# ls unix.txt
unix.txt: 해당 파일이나 디렉토리가 없음
# gcc -o ex2_1.out ex2_1.c
# ex2_1.out
# ls -l unix.txt
-rw-r--r--   1 root   other   0   1월   6일   13:10 unix.txt
```

접근권한:644

O_EXCL 플래그 사용하기

```
01  #include <sys/types.h>
02  #include <sys/stat.h>
03  #include <fcntl.h>
04  #include <unistd.h>
05  #include <stdlib.h>
06  #include <stdio.h>
07
08  int main(void) {
09      int fd;
10
11      fd = open("unix.txt", O_CREAT | O_EXCL, 0644);
12      if (fd == -1) {
13          perror("Excl");
14          exit(1);
15      }
16      close(fd);
17
18      return 0;
19  }
```

```
# ls unix.txt
unix.txt
# ex2_2.out
Excl: File exists
```

```
# rm unix.txt
# ex2_2.out
# ls unix.txt
unix.txt
```

파일 읽기와 쓰기

파일 읽기 : read(2)

```
#include <unistd.h>
ssize_t read(int fildes, void *buf, size_t nbytes);
```

- 파일에서 nbytes로 지정한 크기만큼 바이트를 읽어서 buf에 저장
- 실제로 읽어온 바이트 개수를 리턴
- 리턴값이 0이면 파일의 끝에 도달했음을 의미
- 파일의 종류에 상관없이 무조건 바이트 단위로 읽어 옴

파일 쓰기 : write(2)

```
#include <unistd.h>
ssize_t write(int fildes, const void *buf, size_t nbytes);
```

- buf가 가리키는 메모리에서 nbytes로 지정한 크기만큼 파일에 기록
- 실제로 쓰기를 수행한 바이트 수를 리턴

파일 읽고 쓰기

```
...
06  int main(void) {
07      int rfd, wfd, n;
08      char buf[10];
09
10      rfd = open("unix.txt", O_RDONLY);
11      if(rfd == -1) {
        perror("Open unix.txt");
13          exit(1);
14      }
15
16      wfd = open("unix.bak", O_CREAT | O_WRONLY | O_TRUNC, 0644);
17      if (wfd == -1) {
18          perror("Open unix.bak");
19          exit(1);
20      }
21
22      while ((n = read(rfd, buf, 6)) > 0)
23          if (write(wfd, buf, n) != n) perror("Write");
24
```

파일기술자 2개 선언

6바이트씩 읽어온다

파일 읽고 쓰기

```
25      if (n == -1) perror("Read");
26
27      close(rfd);
28      close(wfd);
29
30      return 0;
31  }
```

```
# ls unix.bak
unix.bak: 해당 파일이나 디렉토리가 없음
# ex2_5.out
# cat unix.bak
Unix System Programming
```

파일 오프셋 지정

파일 오프셋 위치 지정 : lseek(2)

```
#include <sys/types.h>
#include <unistd.h>
off_t lseek(int fildes, off_t offset, int whence);
```

- offset으로 지정한 크기만큼 오프셋을 이동
- offset의 값은 whence값을 기준으로 해석

```
lseek(fd, 5, SEEK_SET);
lseek(fd, 0, SEEK_END);
```

파일의 시작에서 5번째 위치로 이동

파일의 끝에서 0번째, 즉 끝으로 이동

값	설명
SEEK_SET	파일의 시작 기준
SEEK_CUR	현재 위치 기준
SEEK_END	파일의 끝 기준

- 파일 오프셋의 현재 위치를 알려면?

```
cur_offset = lseek(fd, 0, SEEK_CUR);
```

파일 삭제

unlink(2)

```
#include <unistd.h>
int unlink(const char *path);
```

- path에 지정한 파일의 inode에서 링크 수를 감소
- 링크 수가 0이 되면 path에 지정한 파일이 삭제
- 파일 뿐만 아니라 디렉토리(빈 디렉토리 아니어도 됨)도 삭제

remove(3)

```
#include <stdio.h>
int remove(const char *path);
```

- path에 지정한 파일이나 디렉토리를 삭제
- 디렉토리인 경우 빈 디렉토리만 삭제

unlink 함수로 파일 삭제하기

```
01  #include <unistd.h>
02  #include <stdlib.h>
03  #include <stdio.h>
04
05  int main(void) {
06      int cnt;
07
08      cnt = unlink("tmp.aaa");
09      if (cnt == -1) {
10          perror("Unlink tmp.aaa");
11          exit(1);
12      }
13
14      printf("Unlink tmp.aaa success!!!\n");
15
16      return 0;
17  }
```

tmp.aaa 파일 삭제

```
# ls -l tmp*
-rw-r--r--   1 root     other         37  1월  6일  17:50 tmp.aaa
-rw-r--r--   1 root     other         35  1월  6일  18:06 tmp.bbb
# ex2_10.out
Unlink tmp.aaa success!!!
# ls -l tmp*
-rw-r--r--   1 root     other         35  1월  6일  18:06 tmp.bbb
```

Binary data read/write

데이터 생성하고 저장

```
int main() {
    int fd;
    const char *filename = "array.bin";

    const int NUM_DATA = 1000;
    int *numbers = malloc(sizeof(int) * NUM_DATA);
    if (!numbers) {
        perror("malloc");
        return 1;
    }

    for (int i = 0; i < NUM_DATA; i++) {
        numbers[i] = i * 10;   // 0, 10, 20, ... 9990
    }

    fd = open(filename, O_WRONLY ¦ O_CREAT ¦ O_TRUNC, 0644);
    if (fd == -1) {
        perror("open for write");
        free(numbers);
        return 1;
    }
```

Binary data read/write

```
    ssize_t written = write(fd, numbers, sizeof(int) * NUM_DATA);
    if (written == -1) {
        perror("write");
        close(fd);
        free(numbers);
        return 1;
    }
    close(fd);
    printf("배열 %d개 저장 완료 (%zd 바이트)\n", NUM_DATA, written);

    free(numbers);
    return 0;
}
```

Binary data read/write

데이터 읽고 출력

```
int main() {
    int fd;
    const char *filename = "array.bin";
    const int NUM_DATA = 1000;

    int *buffer = malloc(sizeof(int) * NUM_DATA);
    if (!buffer) {
        perror("malloc");
        free(buffer);
        return 1;
    }

    fd = open(filename, O_RDONLY);
    if (fd == -1) {
        perror("open for read");
        free(buffer);
        return 1;
    }
```

Binary data read/write

```
    ssize_t read_bytes = read(fd, buffer, sizeof(int) * NUM_DATA);
    if (read_bytes == -1) {
        perror("read");
        close(fd);
        free(buffer);
        return 1;
    }
    close(fd);
    printf("배열 읽기 완료 (%zd 바이트)\n", read_bytes);

    // 읽은 데이터 일부 출력 (앞 10개만 확인)
    for (int i = 0; i < 10; i++) {
        printf("buffer[%d] = %d\n", i, buffer[i]);
    }

    free(buffer);
    return 0;
}
```

Exercise

Ex.

- 파일을 복사하는 mycp 프로그램 작성
- ./mycp a.txt b.txt

Ex.

- 파일을 복사하는 mycp2 프로그램 작성
- ./mycp a.txt b.txt c.txt d.txt ...
- a.txt를 이후 파일명 들로 모두 복사해서 생성
- a.txt는 한번만 open close 할 것