System Programming

Exercise

Week 02. File I/O – part 1

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – open

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

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

- 성공 시 파일 디스크립터, 실패 시 -1 반환
- Flag 예시 (fcntl.h에 정의되어 있음)
  - O_RDONLY 읽기 전용
  - O_WRONLY 쓰기 전용
  - O_RDWR 읽기 쓰기
  - O_CREAT 파일이 존재 하지 않으면 생성
  - O_EXCL O_CREAT와 함께 사용되며 파일이 존재 시 에러 처리
  - O_TRUNC 파일이 존재 시 잘라 버림
  - O_APPEND 파일의 뒷부분에 추가

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – open /w O_RDWR

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

char *workfile = "test";

int main() {
  int fd;

  /* <fcntl.h>에 정의된 O_RDWR을 사용하여 파일을 읽기/쓰기로 개방한다 */

  if ((fd = open(workfile, O_RDWR)) == -1) {
    perror("open");
    exit(1);
  }

  /* 프로그램의 나머지 부분 */

  exit(0);      /* 정상적인 퇴장 */
}
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – open /w O_CREAT

```c
/* create new file */

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

int main(void){
 int fd;
 mode_t mode;
 mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH; /* 0644 */

 if((fd = open("test", O_CREAT, mode)) == -1){
  perror("creat");
  exit(1);
 }

 close(fd);
 return 0;
}
```

```
ls
main  main.c  test
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – open /w O_CREAT & O_EXCL

```c
/* create new file */

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

int main(void){
 int fd;
 mode_t mode;
 mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH; /* 0644 */

 if((fd = open("test", O_CREAT | O_EXCL, mode)) == -1){
  perror("creat");
  exit(1);
 }

 close(fd);
 return 0;
}
```

```
> ./main
creat: File exists
exit status 1
> rm test
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – open /w O_TRUNC

```
/* create new file */

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

int main(void){
 int fd;
 mode_t mode;
 mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH; /* 0644 */

 if((fd = open("test", O_TRUNC, mode)) == -1){
  perror("creat");
  exit(1);
 }

 close(fd);
 return 0;
}
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – creat

```
#include <fcntl.h>

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

- 새 파일 생성시 사용
- **open()** 함수에 O_CREAT | O_WRONLY | O_TRUNC flag를 사용한 것과 같음
- 초기 유닉스 구현에서는 **open()** 함수의 인자가 둘 뿐이었고, 새로운 파일을 만들 수 없었음
  - 그 당시에 파일을 만들기 위해서는 **creat()** 함수를 사용했음
  - 최근에는 **open()** 함수에 flag 조합에 따라 더욱 다양한 제어가 가능하기 때문에 잘 안쓰는 추세

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – read

```
#include <unistd.h>

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

- 디스크립터 *fd*가 가리키는 열려있는 파일에서 데이터를 읽음
- *count* 인자는 읽으려는 최대 바이트 수를 의미
- 읽은 데이터는 *buf* 인자에 저장됨
- Return value
  - 성공 시 : number of bytes read
  - End of file : 0
  - 실패 시 : - 1

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – read (example 1)

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

int main(void){
 int fd, n;
 char buf[10];
 mode_t mode;
 mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH; /* 0644 */

 if((fd = open("test", O_CREAT | O_TRUNC, mode)) == -1){
  perror("creat");
  exit(1);
 }
 if((n = read(fd, buf, 6)) == -1){
  perror("read");
  exit(1);
 }
 buf[n] = '\0';
 printf("n = %d, buf = %s\n", n, buf);
 close(fd);
 return 0;
}
```

```
> ./main
n = 6, buf = abcdef
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – read (example 2)

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

int main(void){
 int fd, n;
 char buf[10];
 mode_t mode;
 mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH; /* 0644 */

 if((fd = open("test", O_CREAT | O_TRUNC, mode)) == -1){
  perror("creat");
  exit(1);
 }
 if((n = read(fd, buf, 8)) == -1){
  perror("read");
  exit(1);
 }
 buf[n] = '\0';
 printf("n = %d, buf = %s\n", n, buf);
 close(fd);
 return 0;
}
```

```
> ./main
n = 7, buf = abcdefg
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – write

```
#include <unistd.h>

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

- 열려있는 파일에 데이터를 씀
- **read()**와 인자가 유사하지만, *buf*에는 쓸 데이터가 저장됨
- Return value
  - 성공 시 : number of bytes write
  - End of file : 0
  - 실패 시 : - 1

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – write (example)

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

int main(void){
  int fd, n, fd2;
  char buf[10];
  mode_t mode;
  mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH; /* 0644 */

  if((fd = open("test", O_CREAT, mode)) == -1){
    perror("creat");
    exit(1);
  }
  if((n = read(fd, buf, 8)) == -1){
    perror("read");
    exit(1);
  }
  buf[n] = '\0';
  printf("n = %d, buf = %s\n", n, buf);
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – write (example)

```
 if((fd2 = open("test2", O_CREAT | O_WRONLY, mode)) == -1){
   perror("creat2");
   exit(1);
 }

 if(write(fd2, buf, n) != n){
   perror("write");
   exit(1);
 }

 close(fd2);

 close(fd);
 return 0;
}
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – close

```
#include <unistd.h>

int close(int fd);
```

- 파일 사용을 끝냈다고 시스템에게 알림
- 열려 있는 파일 디스크립터를 닫고, 프로세스가 차후에 재사용할 수 있도록 해제함
- 프로세스가 종료되면 열려 있던 모든 파일 디스크립터가 자동으로 닫힘
- Return value
  - 성공 시 : 0
  - 실패 시 : -1

System call – lseek

```
#include <unistd.h>

off_t lseek(int fd, off_t offset, int whence);
```

- 열려있는 파일의 읽기 및 쓰기 위치를 변경함
- *offset*: 파일 포인터를 이동할 상대 위치
  - 음수: 왼쪽으로 이동
  - 양수: 오른쪽으로 이동
- *whence*
  - SEEK_SET: 파일의 시작에서 계산
  - SEEK_CUR: 파일의 현재위치에서 계산
  - SEEK_END: 파일의 끝에서 계산
- 반환 값
  - 성공 시: 파일 포인터의 새로운 위치 (SEEK_SET을 기준으로 계산)
  - 실패 시: -1

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – lseek (example 1)

```
/* 변수 선언 */
off_t fsize;
int fd;
fd = open(filename, O_RDWR);
…


/* 파일 크기 확인하기 */
          fsize= lseek(fd, (off_t)0, SEEK_END);
…


/* 기존 파일 끝에 내용 추가하기 */
          lseek(fd, (off_t)0, SEEK_END);
          write(fd, buf, BSIZE);
…


/* O_APPEND flag를 이용해 파일 끝에 내용 추가하는 방법 */
          fd = open("another", O_WRONLY | O_APPEND);
          write(fd, buf, BUFSIZE);
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – lseek (example 2)

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

char buf1[] = "abcdefghij", buf2[] = "ABCDEFGHIJ";

void fatal(const char *str, int errcode){
     perror(str);
     exit(errcode);
}

int main(void) {
  int fd;

  if ((fd = creat("file.hole", 0640)) < 0)
    fatal("creat error", 1);

  if (write(fd, buf1, 10) != 10)
    fatal("buf1 write error", 1);
  /* offset now = 10 */
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – lseek (example 2)

```
  if (lseek(fd, 40, SEEK_SET) == -1)
    fatal("lseek error", 1);
  /* offset now = 40 */

  if (write(fd, buf2, 10) != 10)
    fatal("buf2 write error", 1);
  /* offset now = 50 */

  exit(0);
}
```

```
abcdefghij                              ABCDEFGHIJ
```

```
> cat file.hole
abcdefghijABCDEFGHIJ
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – lseek (example 3)

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

int main(void){
  int fd, n;
  off_t start, cur;
  char buf[256];

  if((fd = open("test.txt", O_RDONLY)) == -1){
    perror("open");
    exit(1);
  }

  start = lseek(fd, 0, SEEK_CUR);
  n = read(fd, buf, 255);
  buf[n] = '\0';
  printf("Offset start = %d, Read str = %s, n = %d\n", (int)start, buf, n);
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – lseek (example 3)

```
  cur = lseek(fd, 0, SEEK_CUR);
  printf("Offset cur = %d\n", (int)cur);

  start = lseek(fd, 5, SEEK_SET);
  n = read(fd, buf, 255);
  buf[n] = '\0';
  printf("Offset start = %d, Read str = %s, n = %d\n", (int)start, buf, n);

  close(fd);
  return 0;
}
```

```
> cat test.txt
abcdefghijklmn
```

```
> ./main
Offset start = 0, Read str = abcdefghijklmn, n = 14
Offset cur = 14
Offset start = 5, Read str = fghijklmn, n = 9
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – unlink / remove

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

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

- unlink/remove : 파일을 제거한다

- pathname: 절대적 혹은 상대적 파일/디렉토리 경로명
- Return value
  - 성공 시: 0
  - 실패 시: -1

- 빈디렉토리를 제거할 때는 remove만 사용

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – unlink / remove

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

int main(void){
 int cnt;
 cnt = unlink("tmp");

 if(cnt == -1){
  perror("unlink tmp");
  exit(1);
 }

 printf("unlink tmp success\n");
 return 0;
}
```

```
touch tmp
```

```
./main
unlink tmp success
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – unlink / remove

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

int main(void){
 int cnt;
 cnt = remove("tmp");

 if(cnt == -1){
  perror("unlink tmp");
  exit(1);
 }

 printf("unlink tmp success\n");
 return 0;
}
```

```
> touch tmp
```

```
> ./main
unlink tmp success
```

```
> mkdir tmp
> touch ./tmp/a
```

```
> ./main
unlink tmp: Directory not empty
exit status 1
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – unlink / remove

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

int main(void){
 int cnt;
 cnt = unlink("tmp");

 if(cnt == -1){
  perror("unlink tmp");
  exit(1);
 }

 printf("unlink tmp success\n");
 return 0;
}
```

```
> touch tmp
```

```
> ./main
unlink tmp success
```

```
> mkdir tmp
> touch ./tmp/a
```

```
> ./main
unlink tmp: Directory not empty
exit status 1
```

```
> ./main
unlink tmp: Is a directory
exit status 1
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – fcntl

```
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
int fcntl(int fd, int cmd, ...);
```

- fcntl: 열린 파일의 속성(attribute)을 제어한다
  - fd: open 혹은 creat이 반환한 파일 descriptor
  - cmd
    - F_GETFL : flag를 통한 파일 상태 표시기를 되돌려준다
    - F_SETFL : 파일 상태 표시기를 세번 째 변수의 값으로 정한다
  - O_APPEND, O_NONBLOCK, O_SYNC, O_ASYNC 만 가능
- Return value
  - 성공 시: 자연수(>=0)
    - F_GETFL 사용 시 "반환 값 & O_ACCMODE"를 통해 open()의 flag 확인
  - 실패 시: -1
    - F_SETFL 사용시 0

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – fcntl (example)

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

void fatal(const char *str, int errno){
  perror(str);
  exit(errno);
}

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

  int accmode, val, fd;
  if((fd = open("tmp", O_RDWR)) == -1)
    fatal("open", 1);

  if ((val = fcntl(fd, F_GETFL, 0)) < 0)
    fatal("fcntl error for fd", 1);

  accmode = val & O_ACCMODE;
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – fcntl (example)

```
if (accmode == O_RDONLY)
  printf("read only");
else if (accmode == O_WRONLY)
  printf("write only");
else if (accmode == O_RDWR)
  printf("read write");
else {
   fprintf(stderr, "unkown access mode");
  exit(1);
}

val |= O_APPEND;
if(fcntl(fd, F_SETFL, val) == -1)
  fatal("fcntl setfl", 1);
if(write(fd, "abcdefghij", 10) != 10)
  fatal("write", 1);

if (val & O_APPEND)
  printf(", append");
if (val & O_NONBLOCK)
  printf(", nonblocking");
if (val & O_SYNC)
  printf(", synchronous writes");
putchar('\n');
close(fd);
}
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – dup

```
#include <unistd.h>

int dup(int fd);
```

- 파일 디스크립터를 인자로 받아 새로운 파일 디스크립터 반환
- 현재 할당할 수 있는 파일 디스크립터 중 가장 작은 값 할당

- Return value
  - 성공 시 새로운 파일 디스크립터
  - 실패 시 -1

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – dup (example)

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

int main(void){
  int fd, fd1;

  if((fd = open("tmp", O_CREAT | O_WRONLY | O_TRUNC, 0644)) == -1){
    perror("creat");
    exit(1);
  }

  close(1);

  fd1 = dup(fd);

  printf("Ori FD = %d\n", fd);
  printf("DUP FD = %d\n", fd1);
  printf("Standard Output Redirection\n");
  close(fd);

  return 0;
}
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – dup2

```
#include <unistd.h>

int dup2(int oldfd, int newfd);
```

- 새로운 파일 디스크립터를 자동으로 할당하는 **dup()**과 달리 새로운 파일 디스크립터를 지정할 수 있음
- 만일 새로운 파일 디스크립터가 이미 열려있다면 해당 파일 디스크립터를 먼저 닫음
  - 다만 이 과정에서 오류가 일어나더라도 무시하고 진행함
  - 안전하게 닫고 나서 사용하기를 권장
- Return value
  - 성공 시 원하는 파일 디스크립터
  - 실패 시 -1

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

System call – dup2 (example)

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

int main(void){
  int fd;

  if((fd = open("tmp", O_CREAT | O_WRONLY | O_TRUNC, 0644)) == -1){
    perror("creat");
    exit(1);
  }

  dup2(fd, 1);

  printf("DUP2: Standard Output Redirection\n");
  close(fd);

  return 0;
}
```

- open
- creat
- read
- write
- close
- lseek
- unlink/remove
- fcntl

실습 문제

한 파일의 내용을 다른 파일로 복사하는 함수

copy_file()을 작성하여라.

단, Hole이 있는 경우도 복사할 수 있어야 한다.

prototype

- `int copy_file(const char *name1, const char *name2);`

1. 첫 번째 파일을 개방한다.
2. 두 번째 파일을 생성한다.
3. 첫 번째 파일을 읽어 두 번째 파일에 쓴다.
4. 첫 번째 파일의 끝에 도달할 때까지 3번 동작을 반복한다.
5. 두 파일을 닫는다.

THANK YOU

Presented by Hasoo Eun