OS 1 실 습 과 제 (2 주 차)

전 공	학 번	학 년	이 름	제 출 일
융합전자공학부	2012002746	3	김현수	18/3/21

그림 1 Assignment1, Annotation

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

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

  char *c; //포인터 선언
  int fd0, fdl, sz; //파일명을 담을 변수, sz는 사이즈
  c = (char *)malloc(100 * sizeof(char)); //char 사이즈의 100배를 할당합니다.
  fd0 = open("printf.txt",O_RDONLY); //printf.txt 파일을 읽기전용 모드로 엽니다.
  fdl = open("printf-copy.txt",O_CREAT | O_RDWR | O_APPEND, 0644);
  //pnintf-copy.txt를 없으면 만들고(O_CREAT), 읽기쓰기가능(O_RDWR), 새로운 정보는 뒤에 붙이는(O_APPEND) 모드로 엽니다.

   if(fd0 < 0 || fdl < 0) //두 파일을 여는 때 에러가 났는지 확인합니다.
   {
     perror("Both files"); //에러를 출력합니다.
     return 1;
   }

  sz = read(fd0, c, 10); //10바이트 만큼 fd0에서 읽어서 c에 저장합니다.

  printf("read(%d, c, 10) : result = %d bytes read.\n", fd0, sz);
   c[sz] = '\0'; // 마지막 c에 null을 저장합니다.
  printf("Those bytes are as follows: %s\n", c);

   close(fd0); //fd1파일을 닫습니다.

  sz = write(fdl, c, strlen(c)); //c에 있는 문자열을 fd1에 저장합니다.
  printf("write(%d, c, strlen(c)) : result = %d bytes wrote.\n", fdl, sz);
  printf("These %d bytes are wrote to file : %s\n", sz, c);
  //몇바이트를 썼는지 출력하고, 문자열을 출력합니다.
  close(fdl); //fd1파일을 닫습니다.
  free(c);
  //malloc으로 할당한 c를 없앱니다.
}
```

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

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

  char *c; //포인터 선언
  int fd0, fdl, sz; //파일명을 담을 변수, sz는 사이즈
  c = (char *)malloc(100 * sizeof(char)); //char 사이즈의 100배를 할당합니다.
```

```
  fd0 = open("printf.txt",O_RDONLY); //printf.txt 파일을 읽기전용 모드로 엽니다.
  fdl = open("printf-copy.txt",O_CREAT | O_RDWR | O_APPEND, 0644);
  //pnintf-copy.txt를 없으면 만들고(O_CREAT), 읽기쓰기가능(O_RDWR), 새로운 정보는 뒤에 붙이는(O_APPEND) 모드로 엽니다.

   if(fd0 < 0 || fdl < 0) //두 파일을 여는 때 에러가 났는지 확인합니다.
   {
     perror("Both files"); //에러를 출력합니다.
     return 1;
   }

  sz = read(fd0, c, 10); //10바이트 만큼 fd0에서 읽어서 c에 저장합니다.

  printf("read(%d, c, 10) : result = %d bytes read.\n", fd0, sz);
   c[sz] = '\0'; // 마지막 c에 null을 저장합니다.
  printf("Those bytes are as follows: %s\n", c);

   close(fd0); //fd1파일을 닫습니다.

  sz = write(fdl, c, strlen(c)); //c에 있는 문자열을 fd1에 저장합니다.
  printf("write(%d, c, strlen(c)) : result = %d bytes wrote.\n", fdl, sz);
  printf("These %d bytes are wrote to file : %s\n", sz, c);
  //몇바이트를 썼는지 출력하고, 문자열을 출력합니다.
  close(fdl); //fd1파일을 닫습니다.
  free(c);
  //malloc으로 할당한 c를 없앱니다.
}
```

그림 2 Assignment1 Result

```
joseph@joseph-VirtualBox:~$ cat printf.txt
For the purposes of these operation is fishing
joseph@joseph-VirtualBox:~$ gcc aa.c
joseph@joseph-VirtualBox:~$ ./a.out
read(3, c, 10) : result = 10 bytes read.
Those bytes are as follows: For the pu
write(4, c, strlen(c)) : result = 10 bytes wrote.
These 10 bytes are wrote to file : For the pu
joseph@joseph-VirtualBox:~$ cat printf
printf-copy.txt  printf.txt
joseph@joseph-VirtualBox:~$ cat printf-copy.txt
 report holes, so
       these operationFor the pujoseph@joseph-VirtualBox:~$
```

그림 3 Assignment2, Annotation

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

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

char *c; //포인터 선언
int fd0, fdl, sz, seek_cnt; //파일명을 담을 변수, sz는 사이즈
c = (char *)malloc(100 * sizeof(char));//char 사이즈의 100배를 할당합니다.
fd0 = open("printf.txt", O_RDONLY); //printf.txt 파일을 읽기전용 모드로 엽니다.
fdl = open( "printf-copy.txt", O_CREAT | O_RDWR | O_APPEND, 0644);
//pnintf-copy.txt를 없으면 만들고(O_CREAT), 읽기쓰기가능(O_RDWR), 새로운 정보는 뒤에 붙이는(O_APPEND) 모드로 엽니다.

if(fd0 < 0 || fdl < 0) { //두 파일을 여는 때 에러가 났는지 확인합니다.
   perror("Both files"); //에러를 출력합니다.
   return 1;
}
/* This is new line; current offset */
seek_cnt = lseek(fd0, 0, SEEK_SET); //offset을 파일 시작으로부터 0바이트 떨어진곳으로 설정합니다.
printf("Now file offset is %d\n\n", seek_cnt);

sz = read(fd0, c, 40); //fd0에서 c에 40바이트를 읽어옵니다.
seek_cnt = lseek(fd0, 0, SEEK_CUR); //offset을 현재 offste으로부터 0바이트 떨어진곳으로 설정합니다.
printf("lseek(%d, 0, SEEK_CUR) returns the current offset = %d\n\n", fd0, seek_cnt);
/*offset 위치를 출력합니다.
fd0 파일 문자 길이가 39보다 작으면 들어있는 문자 개수+1을 출력합니다.
그 이상은 40을 출력합니다.*/
printf("Those bytes are as follows: %s\n\n", c); //fd0에서 읽어들인 문자열을 출력합니다

printf("Seek back to the beginning of the file, and call read()\n");
lseek(fd0, 0, SEEK_SET); //offset을 파일 시작으로부터 0바이트 떨어진곳으로 설정합니다.
seek_cnt = lseek(fd0, 200, SEEK_CUR); //현재 위치에서 200바이트 떨어진 곳으로 offset을 설정합니다.
printf("lseek(%d, 200, SEEK_CUR) returns the current offset = %d\n\n", fd0, seek_cnt);
//offset 위치를 출력합니다.

sz = read(fd0, c, 40); //fd0에서 c에 40바이트를 읽어옵니다.
printf("read(%d, c, 10) : result = %d bytes read.\n", fd0, sz);
 c[sz] = '\0';
printf("Those bytes are as follows: %s\n\n", c);
 close(fd0);
sz = write(fdl, c, strlen(c)); //c에 있는 문자열을 fd1에 저장합니다.
printf("write(%d, c, strlen(c)) : result = %d bytes wrote.\n", fdl, sz);
printf("These %d bytes are wrote to file : %s\n", sz, c);
  //몇바이트를 썼는지 출력하고, 문자열을 출력합니다.
close(fdl);
free(c);
//malloc으로 할당한 c를 없앱니다.
}
"bb.c" 53L, 2585C
```

```c
#include <stdio.h>

#include <stdlib.h>
```

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

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

char *c; //포인터 선언
int fd0, fdl, sz, seek_cnt; //파일명을 담을 변수, sz는 사이즈
c = (char *)malloc(100 * sizeof(char));//char 사이즈의 100배를 할당합니다.
fd0 = open("printf.txt", O_RDONLY); //printf.txt 파일을 읽기전용 모드로 엽니다.
fdl = open( "printf-copy.txt", O_CREAT | O_RDWR | O_APPEND, 0644);
//pnintf-copy.txt를 없으면 만들고(O_CREAT), 읽기쓰기가능(O_RDWR), 새로운 정보는 뒤에 붙이는(O_APPEND) 모드로 엽니다.

if(fd0 < 0 || fdl < 0) { //두 파일을 여는 때 에러가 났는지 확인합니다.
   perror("Both files"); //에러를 출력합니다.
   return 1;
}
/* This is new line; current offset */
seek_cnt = lseek(fd0, 0, SEEK_SET); //offset을 파일 시작으로부터 0바이트 떨어진곳으로 설정합니다.
printf("Now file offset is %d\n\n", seek_cnt);

sz = read(fd0, c, 40); //fd0에서 c에 40바이트를 읽어옵니다.
seek_cnt = lseek(fd0, 0, SEEK_CUR); //offset을 현재 offste으로부터 0바이트 떨어진곳으로 설정합니다.
printf("lseek(%d, 0, SEEK_CUR) returns the current offset = %d\n\n", fd0, seek_cnt);
/*offset 위치를 출력합니다.
fd0 파일 문자 길이가 39보다 작으면 들어있는 문자 개수+1을 출력합니다.
그 이상은 40을 출력합니다.*/
printf("Those bytes are as follows: %s\n\n", c); //fd0에서 읽어들인 문자
```

열을 출력합니다

```
printf("Seek back to the beginning of the file, and call read()\n");
lseek(fd0, 0, SEEK_SET); //offset을 파일 시작으로부터 0바이트 떨어진곳으로 설정합니다.
seek_cnt = lseek(fd0, 200, SEEK_CUR); //현재 위치에서 200바이트 떨어진 곳으로 offset을 설정합니다.
printf("lseek(%d, 200, SEEK_CUR) returns the current offset = %d\n\n", fd0, seek_cnt);
//offset 위치를 출력합니다.

sz = read(fd0, c, 40); //fd0에서 c에 40바이트를 읽어옵니다.
printf("read(%d, c, 10) : result = %d bytes read.\n", fd0, sz);
 c[sz] = '\0';
printf("Those bytes are as follows: %s\n\n", c);
 close(fd0);
sz = write(fdl, c, strlen(c)); //c에 있는 문자열을 fd1에 저장합니다.
printf("write(%d, c, strlen(c)) : result = %d bytes wrote.\n", fdl, sz);
printf("These %d bytes are wrote to file : %s\n", sz, c);
  //몇바이트를 썼는지 출력하고, 문자열을 출력합니다.
close(fdl);
free(c);
//malloc으로 할당한 c를 없앱니다.
}
```

그림 4 Assignment2 Result