컴퓨터학부 20202920 조민혁

1.소스코드와 실행화면

1.ssu\_fddup.c

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <stdio.h>

int main(void)

{

int testfd;

int fd;

//test.txt파일을 생성하고 열어줍니다.

fd = open("test.txt", O\_CREAT);

//fcntl을 통해서 파일 디스크립터를 복사해줍니다.

testfd = fcntl(fd, F\_DUPFD, 5);

printf("testfd :%d\n", testfd);

//다시 한 번 파일디스크립터를 복사해줍니다.

testfd = fcntl(fd,F\_DUPFD, 5);

printf("testfd :%d\n", testfd);

getchar ();

}

실행화면

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

2.ssu\_fdcopy.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <string.h>

#include <fcntl.h>

#define MSG "message will be written to Terminal\n"

int main(void)

{

int new\_fd;

//표준 출력을 3으로 복사시켜줍니다.

if ((new\_fd = fcntl(STDOUT\_FILENO, F\_DUPFD, 3)) == -1){

fprintf(stderr, "Error : Copying File Descriptor\n");

exit(1);

}

//기존의 표준 출력을 닫아줍니다.

close(STDOUT\_FILENO);

//3으로 표준 출력의 파일 디스크립터를 복사시켜놨기 때문에

//표준 출력에 메세지를 출력시켜줍니다.

write(3, MSG,strlen(MSG));

exit(0);

}

실행화면

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

3.ssu\_closeonexec\_3.c (fcntl예제 3번)

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <fcntl.h>

int main(void)

{

int flag;

//fcntl을 사용하여 파일 디스크립터의 복사본을 만드는데 인자가 부족하게 만듭니다.

if((flag = fcntl(STDOUT\_FILENO, F\_DUPFD)) == -1){

fprintf(stderr, "Error : Checking CLOSE\_ON\_EXEC\n");

exit(1);

}

//flag값을 출력해줍니다.

printf("CLOSE ON EXEC flag is = %d\n", flag);

exit(0);

}

실행화면

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

4.ssu\_nonblock1.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <fcntl.h>

//함수원형들입니다.

void set\_flags(int fd, int flags);

void clr\_flags(int fd, int flags);

char buf[500000];

int main(void)

{

int ntowrite, nwrite;

char \*ptr;

//표준입력으로 읽이서 buf에 저장시켜줍니다.

ntowrite = read(STDIN\_FILENO, buf ,sizeof(buf));

fprintf(stderr, "reading %d bytes\n", ntowrite);

//set\_flags함수를 호출해줍니다.

set\_flags(STDOUT\_FILENO, O\_NONBLOCK);

ptr = buf;

while (ntowrite > 0){

errno = 0;

nwrite = write(STDOUT\_FILENO,ptr,ntowrite);

fprintf(stderr, "nwrite = %d, errno = %d\n", nwrite, errno);

//ptr에서는 write한 문자열의 길이만큼을 이동시키고 ntowrite는 뺴줍니다.

if(nwrite > 0) {

ptr += nwrite;

ntowrite -= nwrite;

}

}

//함수를 호출시켜줍니다.

clr\_flags(STDOUT\_FILENO, O\_NONBLOCK); /\*nonblocking 해제 \*/

exit(0);

}

void set\_flags(int fd, int flags)

{

int val;

//파일 디스크립터의 파일 접근 권한과 상태 플래그를 리턴해줍니다.

if((val = fcntl(fd, F\_GETFL, 0)) < 0){

fprintf(stderr, "fcntl F\_GETFL failed");

exit(1);

}

val |= flags;

//flag를 설정해줍니다.

if(fcntl(fd, F\_SETFL, val) < 0){

fprintf(stderr,"fcntl F\_SETFL failed");

exit(1);

}

}

void clr\_flags(int fd, int flags)

{

int val;

//현재 파일디스크립터의 플래그를 가져옵니다.

if((val = fcntl(fd, F\_GETFL, 0 )) < 0){

fprintf(stderr, "fcntl F\_GETFL failed");

exit(1);

}

val &= ~flags;

//플래그를 설정해줍니다.

if(fcntl(fd, F\_SETFL, val) < 0){

fprintf(stderr, "fcntl F\_SETFL failed");

exit(1);

}

}

실행화면

스크린샷, 텍스트, 패턴, 패브릭이(가) 표시된 사진

자동 생성된 설명

스크린샷, 패턴, 직사각형, 패브릭이(가) 표시된 사진

자동 생성된 설명

5.ssu\_nonblock2.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <errno.h>

#include <fcntl.h>

#define MAX\_CALL 100

#define MAX\_BUF 20000

void serror(char \*str)

{

fprintf(stderr,"%s\n",str);

exit(1);

}

int main(void)

{

int nread, nwrite, val, i=0;

char \*ptr;

char buf[MAX\_BUF];

int call[MAX\_CALL];

//표준 입력을 입력받아 buf에 저장합니다.

nread = read(STDIN\_FILENO, buf ,sizeof(buf));

fprintf(stderr, "read %d bytes\n",nread);

//FLAG정보를 전달받아줍니다.

if((val = fcntl(STDOUT\_FILENO, F\_GETFL, 0)) < 0)

serror("fcntl F\_GETFL error");

val |= O\_NONBLOCK;

//플래그를 설정해줍니다.

if(fcntl(STDOUT\_FILENO, F\_SETFL,val) < 0)

serror("fcntl F\_SETFL error");

for(ptr = buf; nread > 0; i++){

errno = 0;

//표준 출력에 출력시켜줍니다.

nwrite = write(STDOUT\_FILENO,ptr, nread);

if(i < MAX\_CALL)

call[i] = nwrite;

fprintf(stderr, "nwrite = %d, errno = %d\n", nwrite, errno);

if(nwrite > 0){

ptr += nwrite;

nread -= nwrite;

}

}

//플래그 정보를 가져옵니다.

if((val = fcntl(STDOUT\_FILENO, F\_GETFL, 0)) < 0)

serror("fcntl F\_GETFL error");

//플래그를 NONBLOCK으로 설정해줍니다.

val &= ~O\_NONBLOCK;

//플래그를 설정해줍니다.

if(fcntl(STDOUT\_FILENO, F\_SETFL, val) < 0)

serror("fcntl F\_SETFL error");

for(i = 0; i < MAX\_CALL; i++)

fprintf(stdout ,"call[%d] = %d\n",i,call[i]);

exit(0);

}

실행화면

패턴, 스크린샷, 패브릭, 스티치이(가) 표시된 사진

자동 생성된 설명

스크린샷, 패턴, 직사각형, 사각형이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

6.ssu\_open\_on\_exec.c과 ssu\_close\_on\_exec.c

ssu\_open\_on\_exec.c의 소스코드

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

int main(void)

{

int fd;

int val;

//txt파일을 생성해줍니다.

fd = open("exec\_copy.txt",O\_CREAT);

//exec함수로 무한 루프 프로그램을 실행시킵니다.

execl("/home/minhyuk/Example19/loop","./loop",NULL);

exit(0);

}

실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

ssu\_close\_on\_exec.c의 소스코드

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

int main(void)

{

int fd;

int val;

//텍스트 파일을 생성시켜줍니다.

if((fd = open("exec\_copy.txt",O\_CREAT)) < 0){

fprintf(stderr, "open error for %s\n","exec\_copy.txt");

exit(1);

}

//파일디스크립터 값을 가져와서 할당해줍니다.

val = fcntl(fd, F\_GETFD,0);

if(val & FD\_CLOEXEC)

printf("close-on-exec bit on\n");

else

printf("close-on-exec bit off\n");

//비트를 켜줍니다.

val |= FD\_CLOEXEC;

if(val & FD\_CLOEXEC)

printf("close-on-exec bit on\n");

else

printf("close-on-exec bit off\n");

//파일디스크립터를 설정해줍니다.

fcntl(fd, F\_SETFD, val);

//무한루프프로그램을 실행시켜줍니다.

execl("/home/minhyuk/loop","./loop",NULL);

}

실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

7.ssu\_fcntl\_7.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

int main(void)

{

char \*filename = "ssu\_test.txt";

int fd1 , fd2;

int flag;

//파일을 오픈시켜줍니다.

if((fd1 = open(filename, O\_RDWR | O\_APPEND, 0644)) < 0){

fprintf(stderr, "open error for %s\n", filename);

exit(1);

}

//파일 디스크립터의 플래그를 설정할 수 있도록 해줍니다.

if(fcntl(fd1, F\_SETFD, FD\_CLOEXEC) == -1){

fprintf(stderr, "fcntl F\_SETFD error\n");

exit(1);

}

//플래그 정보를 가져옵니다.

if((flag = fcntl(fd1, F\_GETFL,0)) == -1){

fprintf(stderr,"fcntl F\_GETFL error\n");

exit(1);

}

//O\_APPEND플래그가 있는 경우와 없는 경우를 나눕니다.

if(flag & O\_APPEND)

printf("fd1 : O\_APPEND flag is set.\n");

else

printf("fd1 : O\_APPEND flag is NOT set.\n");

if((flag = fcntl(fd1, F\_GETFD, 0)) == -1){

fprintf(stderr, "fcntl F\_GETFD error\n");

exit(1);

}

//flag에 FD\_CLOEXEC이 설정되어 있는지 확인합니다.

if(flag & FD\_CLOEXEC)

printf("fd1 : FD\_CLOEXEC flag is set.\n");

else

printf("fd1 : FD\_CLOEXEC flag is NOT set.\n");

//파일 디스크립터를 복사해서 넣어줍니다.

if((fd2 = fcntl(fd1, F\_DUPFD, 0)) == -1){

fprintf(stderr, "fcntl F\_DUPFD error\n");

exit(1);

}

//flag정보를 가져와서 할당시켜줍니다.

if((flag = fcntl(fd2, F\_GETFL, 0)) == -1){

fprintf(stderr, "fcntl F\_GETFL error\n");

exit(1);

}

if(flag & O\_APPEND)

printf("fd2 : O\_APPEND flag is set.\n");

else

printf("fd2 : O\_APPEND flag is NOT set.\n");

//파일 디스크립터 정보를 가져와서 저장시켜줍니다.

if((flag = fcntl(fd2, F\_GETFD, 0)) == -1){

fprintf(stderr, "fcntl F\_GETFD error\n");

exit(1);

}

if(flag & FD\_CLOEXEC)

printf("fd2 : FD\_CLOEXEC flag is set.\n");

else

printf("fd2 : FD\_CLOEXEC flag is NOT set.\n");

exit(0);

}

실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

8.ssu\_fcntl\_8.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include "ssu\_employee.h"

#define DUMMY 0

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

{

struct ssu\_employee record;

int fd;

int flags;

int length;

int pid;

//인자가 적게 들어온 경우에 대한 예외처리입니다.

if (argc < 2){

fprintf(stderr,"Usage : %s file\n",argv[0]);

exit(1);

}

//파일을 열어줍니다

if((fd = open(argv[1], O\_RDWR)) < 0){

fprintf(stderr, "open error for %s\n", argv[1]);

exit(1);

}

//플래그 정보를 가져와줍니다.

if((flags = fcntl(fd, F\_GETFL, DUMMY)) == -1){

fprintf(stderr, "fcntl F\_GETFL error\n");

exit(1);

}

//O\_APPEND를 켜줍니다.

flags |= O\_APPEND;

//플래그를 구한 값으로 설정해줍니다.

if(fcntl(fd, F\_SETFL, flags) == -1){

fprintf(stderr, "fcntl F\_SETFL error\n");

exit(1);

}

//현재 프로세스의 pid값을 구합니다.

pid = getpid();

while(1){

printf("Enter employee name : ");

scanf("%s", record.name);

if(record.name[0] == '.')

break;

printf("Enter employee salary : ");

scanf("%d", &record.salary);

record.pid = pid;

length = sizeof(record);

//파일에 작성해줍니다.

if(write(fd, (char \*)&record, length) != length){

fprintf(stderr, "record write error\n");

exit(1);

}

}

close(fd);

exit(0);

}

실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

9.ssu\_fcntl\_lock1.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <fcntl.h>

#define NAMESIZE 50

#define MAXTRIES 5

//employee

struct employee{

char name[NAMESIZE];

int salary;

int pid;

};

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

{

struct flock lock;

struct employee record;

int fd, sum = 0,try = 0;

//10초를 기다려줍니다

sleep(10);

//파일을 오픈해줍니다.

if((fd = open(argv[1],O\_RDONLY)) == -1){

perror(argv[1]);

exit(1);

}

//flock구조체의 Lock변수를 설정해줍니다.

lock.l\_type = F\_RDLCK;

lock.l\_whence = 0;

lock.l\_start = 0L;

lock.l\_len = 0L;

//파일디스크립터의 lock을설정해줍니다.

while(fcntl(fd, F\_SETLK, &lock) == -1){

if(errno == EACCES){

if(try++ < MAXTRIES){

sleep(1);

continue;

}

printf("%s busy -- try later\n", argv[1]);

exit(2);

}

perror(argv[1]);

exit(3);

}

sum = 0;

//read해서 record에 넣어줍니다.

while(read(fd, (char \*)&record, sizeof(record)) > 0){

printf("Employee: %s, Salalry: %d\n",record.name,record.salary);

sum+=record.salary;

}

printf("\nTotal salary: %d\n", sum);

//lock 타입을 설정해줍니다.

lock.l\_type = F\_UNLCK;

//lock을 해제해줍니다.

fcntl(fd, F\_SETLK, &lock);

close(fd);

}

실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

10.ssu\_fcntl\_lock2.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <fcntl.h>

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

{

struct flock lock;

int fd;

char command[100];

//파일을 오픈해줍니다.

if((fd = open(argv[1],O\_RDWR)) == -1){

perror(argv[1]);

exit(1);

}

//lock변수를 설정해줍니다.

lock.l\_type = F\_WRLCK;

lock.l\_whence = 0;

lock.l\_start = 0l;

lock.l\_len = 0l;

//설정했던 lock을 fcntl을 통해서 설정해줍니다.

if(fcntl(fd,F\_SETLK, &lock) == -1){

if(errno == EACCES) {

printf("%s busy --try later\n",argv[1]);

exit(2);

}

perror(argv[1]);

exit(3);

}

//commad 변수를 설정해줍니다.

sprintf(command, "vim %s\n",argv[1]);

system(command);

lock.l\_type = F\_UNLCK;

//lock을 설정해줍니다.

fcntl(fd,F\_SETLK,&lock);

close(fd);

return 0;

}

실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

11.ssu\_fcntl\_lock3.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#define NAMESIZE 50

struct employee{

char name[NAMESIZE];

int salary;

int pid;

};

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

{

struct flock lock;

struct employee record;

int fd, recnum, pid;

long position;

//파일을 오픈시켜줍니다.

if((fd = open(argv[1], O\_RDWR)) == -1){

perror(argv[1]);

exit(1);

}

//현재 프로세스의 pid 값을 구합니다.

pid = getpid();

for(;;){

//recordnumber를 입력받습니다.

printf("\nEnter record number: ");

scanf("%d",&recnum);

if(recnum < 0)

break;

position = recnum \* sizeof(record);

lock.l\_type = F\_WRLCK;

lock.l\_whence = 0;

lock.l\_start = position;

lock.l\_len = sizeof(record);

//해당 recnum위치를 WRLCK으로 설정해줍니다.

if(fcntl(fd,F\_SETLKW, &lock) ==-1){

perror(argv[1]);

exit(2);

}

lseek(fd, position,0);

//파일을 읽어서 record에 저장해줍니다.

if(read(fd,(char\*)&record, sizeof(record)) == 0){

printf("record %d not found\n",recnum);

lock.l\_type = F\_UNLCK;

fcntl(fd,F\_SETLK,&lock);

continue;

}

printf("Employee: %s, salary: %d\n",record.name,record.salary);

record.pid = pid;

printf("Enter new salary: ");

scanf("%d", &record.salary);

lseek(fd,position,0);

//파일에작성해줍니다.

write(fd, (char\*)&record, sizeof(record));

lock.l\_type = F\_UNLCK;

fcntl(fd,F\_SETLK,&lock);

}

close(fd);

}

실행결과

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

12.ssu\_fcntl\_lock4.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#define NAMESIZE 50

struct employee{

char name[NAMESIZE];

int salary;

int pid;

};

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

{

struct flock lock;

struct employee record;

int fd, recnum, pid;

long position;

char ans[5];

//인자로 받은 파일을 오픈시켜줍니다.

if((fd = open(argv[1], O\_RDWR)) == -1){

perror(argv[1]);

exit(1);

}

//현재 프로스세의 pid를 가져옵니다.

pid = getpid();

//무한루프를 실행시켜줍니다.

for(;;){

printf("\nEnter record number: ");

scanf("%d",&recnum);

if(recnum < 0)

break;

position = recnum \* sizeof(record);

//lock변수에 대한 설정들을 진행해줍니다.

lock.l\_type = F\_RDLCK;

lock.l\_whence = 0;

lock.l\_start = position;

lock.l\_len = sizeof(record);

//만들어둔 lock을 설정시켜줍니다.

if(fcntl(fd, F\_SETLKW, &lock) == -1){

perror(argv[1]);

exit(2);

}

lseek(fd, position, 0);

//파일을 읽어서 레코드에 저장시켜줍니다.

if(read(fd, (char \*)&record, sizeof(record)) == 0){

printf("record %d not found\n",recnum);

lock.l\_type = F\_UNLCK;

fcntl(fd, F\_SETLK, &lock);

continue;

}

printf("Employee: %s, salary: %d\n", record.name, record.salary);

printf("Do you want to update salary (y or n)? ");

scanf("%s",ans);

//y가 아닌 경우

if(ans[0] != 'y'){

lock.l\_type = F\_UNLCK;

fcntl(fd, F\_SETLK, &lock);

continue;

}

//wr에 lock을 설정해둡니다.

lock.l\_type = F\_WRLCK;

if(fcntl(fd, F\_SETLKW, &lock) == -1){

perror(argv[1]);

exit(3);

}

record.pid = pid;

printf("Enter new salary: ");

scanf("%d", &record.salary);

//position 만큼 파일 포인터를 이동시켜줍니다.

lseek(fd, position, 0);

//파일에 record내용을 작성해줍니다.

write(fd, (char\*)&record, sizeof(record));

lock.l\_type = F\_UNLCK;

fcntl(fd, F\_SETLK, &lock);

}

close(fd);

}

실행결과

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명