컴퓨터학부 20202920 조민혁

1.개요

교육용 운영체제인 xv6를 설치하여, 운영체제에 시스템 콜을 추가하고 크로스 컴파일 방식으로 리눅스 환경에서 xv6 프로그램을 작성하고 컴파일하여 xv6상에 수행하는 설계입니다. 또한 xv6의 커널을 이해하기 위해서 syscall.c, syscall.h, sysproc.c, user.h, usys.S, fcntl.h을 수정합니다. ‘Hello xv6 World’를 출력하는 helloxv6 응용프로그램을 helloxv6.c 코드를 통해 생성하고, xv6에 구현되어 있지 않는 lseek 시스템 콜을 구현하기 위해서 Makefile ,syscall.c, syscall.h, sysproc.c, user.h, usys.S, fcntl.h에 관련 내용들을 수정합니다. 또한 lseektest 쉘 프로그램을 만들어 lseek 시스템콜이 제대로 만들어졌는지 확인합니다. lseektest 쉘 프로그램은 ‘lseektest hello.txt 6 World’와 같이 입력하게 되면 lseektest를 호출하여 hello.txt 파일 안의 6번 인덱스에 값부터 수정하여 덮어써서 파일에 저장시키는 프로그램으로서, 기존의 hello.txt 파일의 내용을 ‘Before : Hello SSU’ 로 변경된 후의 hello.txt 파일의 내용을 ‘After : Hello World’와 같은 형식으로 출력하는 프로그램입니다.

2.상세설계

<함수 간의 호출 그래프>

helloxv6 프로그램의 함수 간의 호출 그래프

텍스트, 폰트, 도표, 스크린샷이(가) 표시된 사진

자동 생성된 설명

lseektest 프로그램의 함수 간의 호출 그래프

도표, 평면도, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명

<함수의 프로토타입 >

int sys\_lseek(void)

이는 sysfile.c에 있는 sys\_lseek함수의 프로토타입입니다. 이 함수는 lseek을 시스템 콜로 추가하기 위해서 사용되어지는 함수로서 파일 디스크립터, offset, whence 값을 int형으로 argint()함수를 사용하여 받습니다. 그렇게 입력받은 값들을 각각 fd, offset, whence 지역변수에 할당하고, fd를 통해서 열린 파일들 중 해당하는 파일을 가져옵니다. 또한 whence 값에 따라 fcntl.h에 미리 설정해둔 SEEK\_SET, SEEK\_CUR, SEEK\_END에 따라 switch문으로 분기하여 struct file \*f로 선언한 f에 입력받은 offset과 whenec를 조합하여 현재 파일의 offset값을 조정하고 리턴하는 함수입니다. 또한 에러가 발생하였을 경우 -1을 리턴합니다.

메인문으로 이루어져서 작동하기 때문에 파일명과 주된 작동원리를 설명하겠습니다.

helloxv6.c  
이는 Hello xv6 World를 출력하기 위한 c파일로 메인문에 printf(1,"Hello xv6 World\n"); 문이 있어서 해당기능을 수행할 수 있도록 합니다.

lseektest.c

lseek 시스템 콜이 제대로 구현되었는지 확인하기 위한 파일로서, 인자로 입력받은 파일을 열고 해당 파일의 내용을 출력한 후, 해당 파일에 마지막으로 출력되는 것이 개행문자인지 확인하고 개행문자인 경우에는 개행문자를 붙히지 않고, 그렇지 않은 경우에는 개행문자를 붙혀서 출력합니다. 그 후에 새롭게 입력받은 단어를 offset위치에 덮어써서 저장시킵니다. 그 후 파일을 다시 열어 저장한 내용을 출력시킵니다.

<구현한 내용과 수정한 내용>

(1)xv6 설치 및 컴파일을 해결하기 위해서 AWS의 EC2를 사용하여 ubuntu x86체제의 인스턴스를 생성하여 수행하였습니다. 우선적으로 ‘git clone <https://github.com/mit-pods/xv6-public>’명령어를 사용하여 xv6-public을 클론받았습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그 후에 xv6-public을 수행하기 위해서 ‘sudo apt-get install qemu-kvm’명령어를 통해서 qemu 에뮬레이터를 설치하였습니다. xv6-public에 있는 Makefile을 통해 make 하려했는데 Werror와 관련된 항목으로 인해 컴파일 에러가 발생하여 Makefile에서 CFLAGS 부분에 Werror 플래그를 삭제하여 해당 컴파일 에러를 없애고 warning으로 바꿀 수 있었습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그 후 make qemu 명령어를 통해서 qemu 에뮬레이터를 수행하여 에뮬레이터 환경에서 ls 명령어를 통해서 xv6에 현재 있는 파일과 디렉토리를 확인할 수 있었습니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

(2)helloxv6 응용프로그램을 구현하기 위해서 우선 helloxv6.c 파일을 생성하고 아래와 같이 구현하였습니다.

#include "types.h"

#include "stat.h"

#include "user.h"

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

{

//print Hello xv6 World

printf(1,"Hello xv6 World\n");

exit();

}

그 후에 helloxv6.c 파일을 생성하는 것만으로는 xv6에서 실행시킬 수 없기 때문에 Makefile의 UPROGS에 해당 내용을 추가해주었습니다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

\_helloxv6\로 추가해줌으로써 xv6 빌드 과정에서 유저 프로그램으로 컴파일될 수 있게 만들어주었습니다. 또한 이렇게 포함시킬 경우 컴파일된 실행 파일은 xv6의 파일 시스템 이미지에 포함되게 만듭니다. 그후 EXTRA부분에도 helloxv6.c를 포함시킵니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

EXTRA 부분에 파일을 포함시킴으로써 해당 파일들이 빌드 과정에서 추가적으로 관리되거나 처리될 파일에 추가시킵니다. 이렇게 Makefile에 포함시킨 후 make clean 명령어를 통해 기존의 컴파일된 파일들을 지우고 새롭게 make 명령어를 통해 컴파일 시킨 후 make qemu를 통해 qemu에뮬레이터를 수행 시킨 후 helloxv6 명령어를 수행시켜서 ‘Hello xv6 World’가 출력되는 것을 확인해볼 수 있었습니다.

(3)lseek() 시스템 콜 추가를 하기 위해서 우선 적으로 user.h파일에

(기존코드)

.

.

.

// system calls

int fork(void);

int exit(void) \_\_attribute\_\_((noreturn));

int wait(void);

int pipe(int\*);

int write(int, const void\*, int);

int read(int, void\*, int);

int close(int);

int kill(int);

int exec(char\*, char\*\*);

int open(const char\*, int);

int mknod(const char\*, short, short);

int unlink(const char\*);

int fstat(int fd, struct stat\*);

int link(const char\*, const char\*);

int mkdir(const char\*);

int chdir(const char\*);

int dup(int);

int getpid(void);

char\* sbrk(int);

int sleep(int);

int uptime(void);

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

.

.

.

(기존코드)

의 마지막 부분에 추가하였습니다. int lseek(int fd, int offset, int whence);로 사용하였습니다. 그 후에 usys.S에 lseek을

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

와 같이 마지막에 SYSCALL(lseek)을 추가하여 주었습니다. 그 후 syscall.h에   
// System call numbers

#define SYS\_fork 1

#define SYS\_exit 2

#define SYS\_wait 3

#define SYS\_pipe 4

#define SYS\_read 5

#define SYS\_kill 6

#define SYS\_exec 7

#define SYS\_fstat 8

#define SYS\_chdir 9

#define SYS\_dup 10

#define SYS\_getpid 11

#define SYS\_sbrk 12

#define SYS\_sleep 13

#define SYS\_uptime 14

#define SYS\_open 15

#define SYS\_write 16

#define SYS\_mknod 17

#define SYS\_unlink 18

#define SYS\_link 19

#define SYS\_mkdir 20

#define SYS\_close 21

#define SYS\_lseek 22  
와 같이 마지막에 #define SYS\_lseek 22로 22번 시스템콜로서 콜 번호를 부여하여주었습니다. 그 후에 syscall.c 파일에

(기존 코드)

.

.

.

extern int sys\_chdir(void);

extern int sys\_close(void);

extern int sys\_dup(void);

extern int sys\_exec(void);

extern int sys\_exit(void);

extern int sys\_fork(void);

extern int sys\_fstat(void);

extern int sys\_getpid(void);

extern int sys\_kill(void);

extern int sys\_link(void);

extern int sys\_mkdir(void);

extern int sys\_mknod(void);

extern int sys\_open(void);

extern int sys\_pipe(void);

extern int sys\_read(void);

extern int sys\_sbrk(void);

extern int sys\_sleep(void);

extern int sys\_unlink(void);

extern int sys\_wait(void);

extern int sys\_write(void);

extern int sys\_uptime(void);

extern int sys\_lseek(void);

static int (\*syscalls[])(void) = {

[SYS\_fork] sys\_fork,

[SYS\_exit] sys\_exit,

[SYS\_wait] sys\_wait,

[SYS\_pipe] sys\_pipe,

[SYS\_read] sys\_read,

[SYS\_kill] sys\_kill,

[SYS\_exec] sys\_exec,

[SYS\_fstat] sys\_fstat,

[SYS\_chdir] sys\_chdir,

[SYS\_dup] sys\_dup,

[SYS\_getpid] sys\_getpid,

[SYS\_sbrk] sys\_sbrk,

[SYS\_sleep] sys\_sleep,

[SYS\_uptime] sys\_uptime,

[SYS\_open] sys\_open,

[SYS\_write] sys\_write,

[SYS\_mknod] sys\_mknod,

[SYS\_unlink] sys\_unlink,

[SYS\_link] sys\_link,

[SYS\_mkdir] sys\_mkdir,

[SYS\_close] sys\_close,

[SYS\_lseek] sys\_lseek,

};

.

.

.

(기존 코드)

와 같이 마지막에 extern int sys\_lseek(void); 로 외부에 선언된 sys\_lseek을 참조하여 [SYS\_lseek] sys\_lseek으로, 시스템콜 테이블에 lseek시스템 콜 번호와 이를 처리하는 sys\_lseek 함수를 연결해줍니다.

그 후에 sysfile..c에 sys\_lseek을 구현하기 위해서 fcntl.h 헤더에   
#define O\_RDONLY 0x000

#define O\_WRONLY 0x001

#define O\_RDWR 0x002

#define O\_CREATE 0x200

#define SEEK\_SET 0

#define SEEK\_CUR 1

#define SEEK\_END 2  
와 같이 SEEK\_SET, SEEK\_CUR, SEEK\_END를 정의하여 주었습니다. 마지막으로 sysfile.c 파일에   
(기존코드)

.

.

.

//sys\_lseek

int

sys\_lseek(void)

{

int fd;

int offset;

int whence;

struct file \*f;

// Use argint to retrieve and validate system call arguments

// argint retrieves an integer argument passed by the user program.

// If retrieving any of the arguments (fd, offset, whence) fails, return -1.

if (argint(0, &fd) < 0 || argint(1, &offset) < 0 || argint(2, &whence) < 0)

return -1;

// Retrieve the file structure pointer for the given file descriptor (fd) from the current process

f = myproc()->ofile[fd];

if (f == 0)

return -1;

//Match cases with whence

switch (whence) {

case SEEK\_SET:

f->off = offset;

break;

case SEEK\_CUR:

f->off += offset;

break;

case SEEK\_END:

f->off = f->ip->size + offset;

break;

default:

return -1;

}

//return current file's offset

return f->off;

}

를 구현합니다. 위의 함수는 argint()함수를 사용하여 시스템 콜에 전달된 정수형 인자를 각각 fd, offset, whence에 대입시킵니다. 그 후 f = myproc()->ofile[fd]; 를 사용하여 현재 프로세스의 열린 파일들의 목록들 중 fd와 인덱스가 일치하는 파일을 f에 할당시킵니다. 그리고 만약에 f가 아무것도 리턴받지 않아 0일 경우에는 -1을 리턴합니다. whence값에 따라 fcntl.h에 정의된 대로 switch문을 구성하여 해당하는 case에 대해서 다르게 처리해줍니다. SEEK\_SET의 경우에는 입력된 offset값을 파일의 offset으로 넣어서 리턴시키고, SEEK\_CUR의 경우에는 현재 offset에 입력받은 offset을 더하여 리턴시키며, SEEK\_END 의 경우에는 파일의 크기인 마지막 위치에 입력받은 offset크기 만큼 더하여 리턴시킵니다. 그 외의 whence값이 들어오는 경우에는 에러가 발생한 것이기에 -1을 리턴하게끔 함수를 구현하였습니다.

(4)lseek() 시스템 콜을 호출하는 간단한 쉘 프로그램을 구현하기 위해서 lseektest.c를 생성하였습니다.

#include "types.h"

#include "stat.h"

#include "user.h"

#include "fcntl.h"

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

int fd, offset,len;

int buffersize = 512;

int readBuffer;

int fileLength = 0;

//because xv6 block is 512

char buffer[buffersize];

if(argc < 4){

printf(1,"usage : lseektest <filename> <offset> <string>\n");

exit();

}

//open file for print

fd = open(argv[1],O\_RDWR);

if(fd < 0){

printf(1,"File open error!\n");

exit();

}

//print for before

printf(1,"Before : ");

//print until all comment is done

while((readBuffer = read(fd,buffer, buffersize))>0){

//print to screen

fileLength += write(1,buffer,readBuffer);

}

//move to end-1

if(lseek(fd,fileLength-1,SEEK\_SET)<0){

printf(1,"lseek error\n");

exit();

}

//read last word

if(read(fd,buffer,1)<0){

printf(1,"read error\n");

exit();

}

//if last word is \n don't print \n

if(buffer[0] != '\n'){

printf(1,"\n");

}

//move for print lseek

offset = atoi(argv[2]);

if(lseek(fd,offset,SEEK\_SET)<0){

printf(1,"lseek error\n");

exit();

}

len = strlen(argv[3]);

if(write(fd,argv[3],len)<0){

printf(1,"write error\n");

exit();

}

//close file

close(fd);

//reopen file

fd = open(argv[1],O\_RDWR);

if(fd < 0){

printf(1,"File reopen error\n");

exit();

}

//print after data

printf(1,"After : ");

//read file data

while((readBuffer = read(fd,buffer, buffersize))>0){

write(1,buffer,readBuffer);

}

printf(1,"\n");

//close file

close(fd);

exit();

}

와 같이 구현하였습니다. 첫번째 인자로 open할 파일을 입력 받습니다. 그 후에 offset 과 바꿀 문장을 입력받습니다.

먼저 인자가 4개보다 작은 경우에는 에러를 출력하고 어떻게 사용할 수 있을지에 대한 예외처리를 진행해줍니다. 그 후에 인자로 받은 파일을 open하여 줍니다. 이때 에러가 발생하여 open에 -1이 리턴될 경우에 대한 예외처리를 진행해줍니다. 그 후에 read함수로 파일을 읽은 후에 write함수로 파일 내용을 표준 출력으로 출력시켜줍니다. 그 후에 fileLength로 write할 때마다 기록한 바이트 수를 이용하여 파일의 마지막 위치로 이동하여 마지막 위치에 해당하는 것이 \n인 경우에는 \n을 한 번 더 출력하지 않지만, \n이 아닌 경우에는 문장의 분리를 위하여 \n을 출력시켜줍니다.  
그 후에 offset값으로 입력받은 위치로 오프셋을 이동시킨 후 strlen으로 새롭게 수정할 문장의 길이를 구하고 이를 이용하여, 파일에 write함수로 작성하여 줍니다. 그 후 파일을 한 번 close한 후에 다시 파일을 열어서 After : 를 출력하고 변경된 파일 내용을 read함수와 write함수를 통해 출력합니다. 그 후 다시 파일을 닫고 프로그램을 종료합니다. 이때 hello.txt를 xv6에 포함시키기 위해서 Makefile을 수정하였습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위와 같이 fs.img에 hello.txt를 포함시켰습니다. 또한

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

에도 hello.txt를 포함시키고, EXTRA에도 hello.txt를 포함시켰습니다. 이 뿐만 아니라 lseektest가 컴파일되어서 실행할 수 있도록 UPROGS와 EXTRA에도 포함시켰습니다.

(기존코드)

.

.

.

UPROGS=\

\_cat\

\_echo\

\_forktest\

\_grep\

\_init\

\_kill\

\_ln\

\_ls\

\_mkdir\

\_rm\

\_sh\

\_stressfs\

\_usertests\

\_wc\

\_zombie\

\_helloxv6\

\_lseektest

.

.(기존코드)

.

EXTRA=\

mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\

ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\

printf.c umalloc.c\

README hello.txt dot-bochsrc \*.pl toc.\* runoff runoff1 runoff.list\

.gdbinit.tmpl gdbutil\

helloxv6.c\

lseektest.c\

.

.

.

(기존코드)

와 같이 수정하여서 사용하였습니다.

3.결과

(1)xv6설치 및 컴파일을 만족시키기 위해서 git clone <https://github.com/mit-pods/xv6-public> 명령어를 수행하여 xv6-public을 클론받았습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그 후에 qemu 에뮬레이터를 사용하여 실행시키기 위해서 sudo apt-get install qemu-kvm 명령어를 수행하였습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

make의 실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

xv6-public 디렉토리에서 make 명령어를 통해서 Makefile에 있는 명령어들이 실행될 수 있도록 하였고 그 결과를 캡처하였습니다.

make qemu의 실행결과

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

make qemu 명령어를 수행하여 qemu를 실행시켜서 가상화된 환경에서 xv6를 실행시킵니다.

ls 실행결과

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

ls 명령어를 수행하여, xv6 내부에 있는 파일과 디렉토리 목록을 출력하였습니다.

(2)helloxv6 응용 프로그램 구현을 위해서 실행한 결과입니다.

helloxv6.c로 만든 helloxv6응용프로그램의 결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

helloxv6 응용 프로그램을 사용하여 화면상에 ‘Hello xv6 World’를 출력하였습니다.

lseek 시스템 콜의 구현을 확인하기 위해서 시행하는 lseektest의 실행결과입니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

lseektest hello.txt 6 World를 입력하게 되면 기존에 hello.txt에 있던 내용을 ‘Before : Hello SSU’와 같은 형식으로 출력하고, offset 6에 해당하는 위치에 가서 World로 입력하였기 때문에 World로 덮어쓰게 됩니다. 따라서 ‘After : Hello World’와 같이 변한 내용을 파일에 저장하고 출력하게 됩니다.

4.소스코드

Makefile 의 소스코드입니다.

(기존코드)

.

.

.

CC = $(TOOLPREFIX)gcc

AS = $(TOOLPREFIX)gas

LD = $(TOOLPREFIX)ld

OBJCOPY = $(TOOLPREFIX)objcopy

OBJDUMP = $(TOOLPREFIX)objdump

CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -fno-omit-frame-pointer

CFLAGS += $(shell $(CC) -fno-stack-protector -E -x c /dev/null >/dev/null 2>&1 && echo -fno-stack-protector)

ASFLAGS = -m32 -gdwarf-2 -Wa,-divide

# FreeBSD ld wants ``elf\_i386\_fbsd''

LDFLAGS += -m $(shell $(LD) -V | grep elf\_i386 2>/dev/null | head -n 1)

.

.(기존코드)

.

UPROGS=\

\_cat\

\_echo\

\_forktest\

\_grep\

\_init\

\_kill\

\_ln\

\_ls\

\_mkdir\

\_rm\

\_sh\

\_stressfs\

\_usertests\

\_wc\

\_zombie\

\_helloxv6\

\_lseektest

fs.img: mkfs README hello.txt $(UPROGS)

./mkfs fs.img README hello.txt $(UPROGS)

.

.(기존코드)

.

# make a printout

FILES = $(shell grep -v '^\#' runoff.list)

PRINT = runoff.list runoff.spec README hello.txt toc.hdr toc.ftr $(FILES)

.

.(기존코드)

.

EXTRA=\

mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\

ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\

printf.c umalloc.c\

README hello.txt dot-bochsrc \*.pl toc.\* runoff runoff1 runoff.list\

.gdbinit.tmpl gdbutil\

helloxv6.c\

lseektest.c\

.

.

.

(기존코드)

syscall.h의 소스코드

// System call numbers

#define SYS\_fork 1

#define SYS\_exit 2

#define SYS\_wait 3

#define SYS\_pipe 4

#define SYS\_read 5

#define SYS\_kill 6

#define SYS\_exec 7

#define SYS\_fstat 8

#define SYS\_chdir 9

#define SYS\_dup 10

#define SYS\_getpid 11

#define SYS\_sbrk 12

#define SYS\_sleep 13

#define SYS\_uptime 14

#define SYS\_open 15

#define SYS\_write 16

#define SYS\_mknod 17

#define SYS\_unlink 18

#define SYS\_link 19

#define SYS\_mkdir 20

#define SYS\_close 21

#define SYS\_lseek 22

sysfile.c의 소스코드

(기존코드)

.

.

.

//sys\_lseek

int

sys\_lseek(void)

{

int fd;

int offset;

int whence;

struct file \*f;

// Use argint to retrieve and validate system call arguments

// argint retrieves an integer argument passed by the user program.

// If retrieving any of the arguments (fd, offset, whence) fails, return -1.

if (argint(0, &fd) < 0 || argint(1, &offset) < 0 || argint(2, &whence) < 0)

return -1;

// Retrieve the file structure pointer for the given file descriptor (fd) from the current process

f = myproc()->ofile[fd];

if (f == 0)

return -1;

//Match cases with whence

switch (whence) {

case SEEK\_SET:

f->off = offset;

break;

case SEEK\_CUR:

f->off += offset;

break;

case SEEK\_END:

f->off = f->ip->size + offset;

break;

default:

return -1;

}

//return current file's offset

return f->off;

}

user.h의 소스코드

struct stat;

struct rtcdate;

// system calls

int fork(void);

int exit(void) \_\_attribute\_\_((noreturn));

int wait(void);

int pipe(int\*);

int write(int, const void\*, int);

int read(int, void\*, int);

int close(int);

int kill(int);

int exec(char\*, char\*\*);

int open(const char\*, int);

int mknod(const char\*, short, short);

int unlink(const char\*);

int fstat(int fd, struct stat\*);

int link(const char\*, const char\*);

int mkdir(const char\*);

int chdir(const char\*);

int dup(int);

int getpid(void);

char\* sbrk(int);

int sleep(int);

int uptime(void);

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

.

.

.

(기존코드)

fcntl.h의 소스코드

#define O\_RDONLY 0x000

#define O\_WRONLY 0x001

#define O\_RDWR 0x002

#define O\_CREATE 0x200

#define SEEK\_SET 0

#define SEEK\_CUR 1

#define SEEK\_END 2

usys.S의 소스코드

#include "syscall.h"

#include "traps.h"

#define SYSCALL(name) \

.globl name; \

name: \

movl $SYS\_ ## name, %eax; \

int $T\_SYSCALL; \

ret

SYSCALL(fork)

SYSCALL(exit)

SYSCALL(wait)

SYSCALL(pipe)

SYSCALL(read)

SYSCALL(write)

SYSCALL(close)

SYSCALL(kill)

SYSCALL(exec)

SYSCALL(open)

SYSCALL(mknod)

SYSCALL(unlink)

SYSCALL(fstat)

SYSCALL(link)

SYSCALL(mkdir)

SYSCALL(chdir)

SYSCALL(dup)

SYSCALL(getpid)

SYSCALL(sbrk)

SYSCALL(sleep)

SYSCALL(uptime)

SYSCALL(lseek)

syscall.c의 소스코드

(기존코드)

.

.

.

extern int sys\_chdir(void);

extern int sys\_close(void);

extern int sys\_dup(void);

extern int sys\_exec(void);

extern int sys\_exit(void);

extern int sys\_fork(void);

extern int sys\_fstat(void);

extern int sys\_getpid(void);

extern int sys\_kill(void);

extern int sys\_link(void);

extern int sys\_mkdir(void);

extern int sys\_mknod(void);

extern int sys\_open(void);

extern int sys\_pipe(void);

extern int sys\_read(void);

extern int sys\_sbrk(void);

extern int sys\_sleep(void);

extern int sys\_unlink(void);

extern int sys\_wait(void);

extern int sys\_write(void);

extern int sys\_uptime(void);

extern int sys\_lseek(void);

static int (\*syscalls[])(void) = {

[SYS\_fork] sys\_fork,

[SYS\_exit] sys\_exit,

[SYS\_wait] sys\_wait,

[SYS\_pipe] sys\_pipe,

[SYS\_read] sys\_read,

[SYS\_kill] sys\_kill,

[SYS\_exec] sys\_exec,

[SYS\_fstat] sys\_fstat,

[SYS\_chdir] sys\_chdir,

[SYS\_dup] sys\_dup,

[SYS\_getpid] sys\_getpid,

[SYS\_sbrk] sys\_sbrk,

[SYS\_sleep] sys\_sleep,

[SYS\_uptime] sys\_uptime,

[SYS\_open] sys\_open,

[SYS\_write] sys\_write,

[SYS\_mknod] sys\_mknod,

[SYS\_unlink] sys\_unlink,

[SYS\_link] sys\_link,

[SYS\_mkdir] sys\_mkdir,

[SYS\_close] sys\_close,

[SYS\_lseek] sys\_lseek,

};

.

.

.

(기존코드)

helloxv6.c의 소스코드

#include "types.h"

#include "stat.h"

#include "user.h"

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

{

//print Hello xv6 World

printf(1,"Hello xv6 World\n");

exit();

}

lseektest.c의 소스코드

#include "types.h"

#include "stat.h"

#include "user.h"

#include "fcntl.h"

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

int fd, offset,len;

int buffersize = 512;

int readBuffer;

int fileLength = 0;

//because xv6 block is 512

char buffer[buffersize];

if(argc < 4){

printf(1,"usage : lseektest <filename> <offset> <string>\n");

exit();

}

//open file for print

fd = open(argv[1],O\_RDWR);

if(fd < 0){

printf(1,"File open error!\n");

exit();

}

//print for before

printf(1,"Before : ");

//print until all comment is done

while((readBuffer = read(fd,buffer, buffersize))>0){

//print to screen

fileLength += write(1,buffer,readBuffer);

}

//move to end-1

if(lseek(fd,fileLength-1,SEEK\_SET)<0){

printf(1,"lseek error\n");

exit();

}

//read last word

if(read(fd,buffer,1)<0){

printf(1,"read error\n");

exit();

}

//if last word is \n don't print \n

if(buffer[0] != '\n'){

printf(1,"\n");

}

//move for print lseek

offset = atoi(argv[2]);

if(lseek(fd,offset,SEEK\_SET)<0){

printf(1,"lseek error\n");

exit();

}

len = strlen(argv[3]);

if(write(fd,argv[3],len)<0){

printf(1,"write error\n");

exit();

}

//close file

close(fd);

//reopen file

fd = open(argv[1],O\_RDWR);

if(fd < 0){

printf(1,"File reopen error\n");

exit();

}

//print after data

printf(1,"After : ");

//read file data

while((readBuffer = read(fd,buffer, buffersize))>0){

write(1,buffer,readBuffer);

}

printf(1,"\n");

//close file

close(fd);

exit();

}

hello.txt의 내용

Hello SSU