컴퓨터학부 20142385 엄재식

1. 과제 개요
2. lseek 구현
3. lseek의 옵션 구현
4. 상세설계
5. syscall.c

int lseek(int fd, int offset, int whence, char \*opt)

opt를 strcmp를 이용해 읽고 옵션마다 다른 값을 whence에 더해서

syscall3(SYS\_LSEEK, fd, offset, whence)를 호출함

1. do\_syscall.c

do\_lseek 함수

whence의 값을 비교해 옵션에 맞게 함수를 진행한다.

cur\_process->file[fd]->pos 값을 whence와 offset을 이용해 구한다.

1. e 옵션

파일 크기를 넘어가는 값만큼 끝에서부터 write로 0을 채워준다

1. a 옵션

write로 파일 끝에서부터 whence까지 offset만큼 옮기고 빈공간을 offset만큼 write로 0으로 채운다.

1. re 옵션

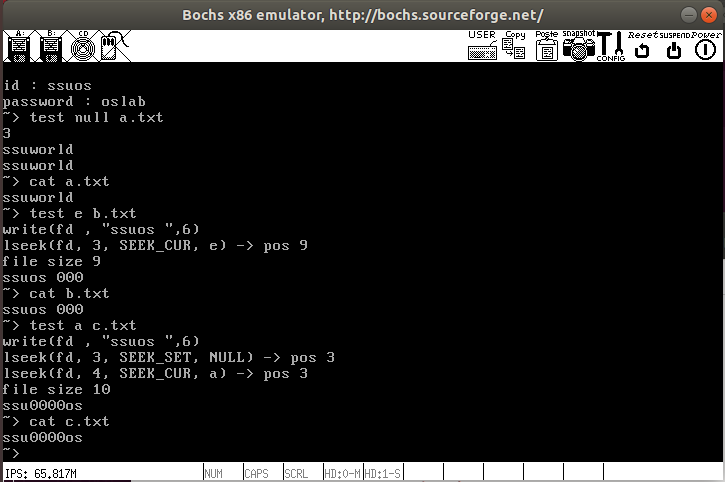
write로 파일 끝에서부터 시작까지 offset만큼 옮기고 처음부분을 offset만큼 write로 0으로 채운다.

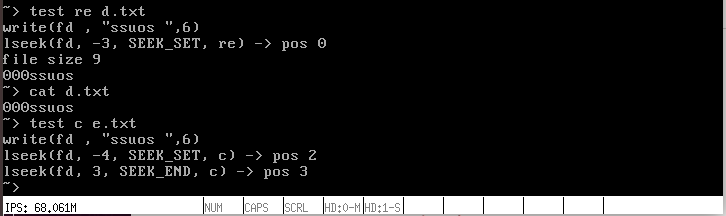
1. c 옵션

whence에서 offset만큼 이동한 값이 음수면 양수가 될 때까지 파일사이즈만큼 더해줘서 양수로 만들고

그 값을 파일사이즈로 모듈러 연산을 하여 pos값을 구한다.

1. 실행결과





1. 소스코드
2. syscall.c

void init\_syscall(void)  
{  
    REGSYS(SYS\_FORK, do\_fork, 3);  
    REGSYS(SYS\_EXIT, do\_exit, 1);  
    REGSYS(SYS\_WAIT, do\_wait, 1);  
    REGSYS(SYS\_SSUREAD, do\_ssuread, 0);  
    REGSYS(SYS\_SHUTDOWN, do\_shutdown, 0);  
    REGSYS(SYS\_OPEN, do\_open, 2);  
    REGSYS(SYS\_READ, do\_read, 3);  
    REGSYS(SYS\_WRITE, do\_write, 3);  
    REGSYS(SYS\_LSEEK, do\_lseek, 3);  
}

int lseek(int fd, int offset, int whence, char \*opt)  
{  
    if (!strcmp(opt, "e")) //e opt  
        whence += 10;  
    if (!strcmp(opt, "a")) //a opt  
        whence += 100;  
    if (!strcmp(opt, "re"))//re opt  
        whence += 1000;  
    if (!strcmp(opt, "c")) //c opt  
        whence += 10000;      
    return syscall3(SYS\_LSEEK, fd, offset, whence);  
}

1. do\_syscall.c

int do\_lseek(int fd, int offset, int whence)  
{  
    struct ssufile \*cursor;  
    int temp;  
    int i, n, start, size;  
    char c;  
    if( (cursor = cur\_process->file[fd]) == NULL)  
        return -1;  
    //null opt  
    if (whence >= -1 && whence <= 1)  
    {  
        //whence에서 offset만큼 이동  
        switch (whence) {  
            case SEEK\_SET:  
                temp = offset;  
                break;  
            case SEEK\_CUR:  
                temp = cursor->pos + offset;  
                break;  
            case SEEK\_END:  
                temp = cursor->inode->sn\_size + offset;  
                break;  
        }  
        //음수이거나 파일크기를 넘는 값은 -1 리턴  
        if (temp < 0)  
            return -1;  
        else if (temp > cursor->inode->sn\_size)  
            return -1;  
        else  
        {  
            //정상값이면 cursor->pos 변경 후 리턴  
            cursor->pos = temp;   
            return cursor->pos;  
        }  
    }  
    //e opt  
    if (whence >= 9 && whence <= 11)  
    {  
        //whence에서 offset만큼 이동  
        switch (whence-10) {  
            case SEEK\_SET:  
                temp = offset;  
                break;  
            case SEEK\_CUR:  
                temp = cursor->pos + offset;  
                break;  
            case SEEK\_END:  
                temp = cursor->inode->sn\_size + offset;  
                break;  
        }  
        //음수 값 -1 리턴  
        if (temp < 0)  
            return -1;  
        else if (temp > cursor->inode->sn\_size)  
        {      
            //파일 크기를 넘어간 값만큼 0으로 집어넣음  
            cursor->pos = cursor->inode->sn\_size;  
            n = temp - cursor->inode->sn\_size;  
            for (i = 0; i < n; i++)  
                write(fd, "0", 1);  
            return cursor->pos;  
        }       
        else  
        {  
            cursor->pos = temp;   
            return cursor->pos;  
        }  
    }  
    //a opt  
    if (whence >= 99 && whence <= 101)  
    {  
        //whence에서 offset만큼 이동한 값  
        switch (whence-100) {  
            case SEEK\_SET:  
                start = 0;  
                break;  
            case SEEK\_CUR:  
                temp = cursor->pos + offset;  
                if (offset < 0)  
                    start = cursor->pos + offset;  
                else   
                    start = cursor->pos;  
                break;  
            case SEEK\_END:  
                temp = cursor->inode->sn\_size + offset;  
                if (offset < 0)  
                    start = cursor->pos + offset;  
                else   
                    start = cursor->pos;  
                break;  
        }  
        if (offset < 0)  
            offset = -offset;  
        if (start < 0)  
            start = 0;  
        //옮겨야할 시작점  
        n = cursor->inode->sn\_size - start;  
        //파일사이즈 늘리기 전의 사이즈저장  
        size = cursor->inode->sn\_size;  
  
        //파일사이즈 offset만큼 늘리기          
        cursor->pos = cursor->inode->sn\_size;  
        for (i = 0; i < offset; i++)  
            write(fd, "\0", 1);  
        //파일 끝에서부터 옮겨야할 시작점까지 offset만큼 옮기기  
        for (i = 0; i < n; i++)  
        {  
            cursor->pos = size - 1 - i;  
            read(fd, &c, 1);  
            cursor->pos += offset - 1;  
            write(fd, &c, 1);      
        }  
        //offset만큼 0채우기  
        cursor->pos = start;  
        for(i = 0; i < offset; i++)  
            write(fd, "0", 1);  
        cursor->pos = start;  
        return cursor->pos;  
    }  
    //re opt      
    if (whence >= 999 && whence <= 1001)  
    {  
        //whence에서 offset만큼 이동한 값  
        switch (whence-1000) {  
            case SEEK\_SET:  
                temp = offset;  
                break;  
            case SEEK\_CUR:  
                temp = cursor->pos + offset;  
                break;  
            case SEEK\_END:  
                temp = cursor->inode->sn\_size + offset;  
                break;  
        }  
        //음수 값  
        if (temp < 0)  
        {  
            if (offset < 0)  
                offset = -offset;  
              
            size = cursor->inode->sn\_size;  
            //파일사이즈 offset만큼 늘리기  
            cursor->pos = cursor->inode->sn\_size;  
            for (i = 0; i < offset; i++)  
                write(fd, "\0", 1);  
            //끝부터 처음까지 offset만큼 옮기기  
            for (i = 0; i < size; i++)  
            {  
                cursor->pos = size - 1 - i;  
                read(fd, &c, 1);  
                cursor->pos += offset - 1;  
                write(fd, &c, 1);  
            }  
            //offset만큼 0으로 채우기  
            cursor->pos = 0;  
            for (i = 0; i < offset; i++)  
                write(fd, "0", 1);  
            cursor->pos = 0;  
            return cursor->pos;      
              
        }  
        else if (temp > cursor->inode->sn\_size)  
            return -1;      
        else  
        {  
            cursor->pos = temp;   
            return cursor->pos;  
        }  
    }  
    //c opt      
    if (whence >= -9999 && whence <= 10001)  
    {  
        switch (whence-10000) {  
            case SEEK\_SET:  
                temp = offset;  
                break;  
            case SEEK\_CUR:  
                temp = cursor->pos + offset;  
                break;  
            case SEEK\_END:  
                temp = cursor->inode->sn\_size + offset;  
                break;  
        }  
        size = cursor->inode->sn\_size;  
        //음수값 양수로 바꿈          
        if (temp < 0)  
            while (temp < 0)  
                temp += size;  
        //모듈러          
        cursor->pos = temp % size;  
        return cursor->pos;  
          
    }  
              
}

1. proc.c

void lseek\_proc(void \*aux , void \*filename)  
{  
  
    char buf[BUFSIZ] = {0};  
    int fd;  
    fd = open(filename, O\_RDWR);  
    if (fd < 0 )return;  
    if (!strcmp((char \*)aux, "e"))  
    {  
        printk("write(fd , \"ssuos \",6)\n");  
        write(fd , "ssuos ",6);  
        printk ("lseek(fd, 3, SEEK\_CUR, e) -> pos %d\n", lseek(fd, 3, SEEK\_CUR, aux));   
        printk("file size %d\n", cur\_process->file[fd]->inode->sn\_size);   
        lseek(fd, 0, SEEK\_SET, NULL);          
        read(fd, buf, BUFSIZ);  
        printk("%s\n", buf);  
    }  
    else if (!strcmp((char \*)aux, "a"))  
    {  
        printk("write(fd , \"ssuos \",6)\n");  
        write(fd , "ssuos ",6);  
        printk("lseek(fd, 3, SEEK\_SET, NULL) -> pos %d\n", lseek(fd, 3, SEEK\_SET, NULL));   
        printk("lseek(fd, 4, SEEK\_CUR, a) -> pos %d\n", lseek(fd, 4, SEEK\_CUR, aux));  
        printk("file size %d\n", cur\_process->file[fd]->inode->sn\_size);   
        lseek(fd, 0, SEEK\_SET, NULL);              
        read(fd, buf, BUFSIZ);  
        printk("%s\n", buf);  
    }  
    else if (!strcmp((char \*)aux, "re"))  
    {  
        printk("write(fd , \"ssuos \",6)\n");          
        write(fd , "ssuos ",6);  
        printk("lseek(fd, -3, SEEK\_SET, re) -> pos %d\n", lseek(fd, -3, SEEK\_SET, aux));  
        printk("file size %d\n", cur\_process->file[fd]->inode->sn\_size);   
        lseek(fd, 0, SEEK\_SET, NULL);      
        read(fd, buf, BUFSIZ);  
        printk("%s\n", buf);  
    }  
    else if (!strcmp((char \*)aux, "c"))  
    {  
        printk("write(fd , \"ssuos \",6)\n");  
        write(fd , "ssuos ",6);                  
        printk("lseek(fd, -4, SEEK\_SET, c) -> pos %d\n", lseek(fd, -4, SEEK\_SET, aux));  
        printk("lseek(fd, 3, SEEK\_END, c) -> pos %d\n", lseek(fd, 3, SEEK\_END, aux));  
    }  
    else  
    {  
        write(fd , "ssuos ",6);  
        //if you add lseek() system call , remove the '//'  
        printk ("%d \n", lseek(fd, -3, SEEK\_CUR, aux));    
        write(fd, "world",5);  
        lseek(fd, 0, SEEK\_SET, aux);  
        read(fd , buf, 8);  
        printk("%s\n", buf);  
        lseek(fd, -9, SEEK\_END, aux);  
        read(fd, buf, 9);  
        printk("%s\n", buf);  
    }  
    //ssuworld 가 정확히 출력되어야 함  
    // 옵션에 대한 시나리오 및 검증할 코드 아래에 추가  
    // 각 옵션에 대해 파일 크기 및 내용이 정확하게 채워지는지 보여야 함  
    /\*   option  \*/  
}