OS project3 wiki

2016024893 오성준

1. 디자인

1) Login / Logout

Init 할 때, sh.c 대신에, login.c를 실행하게 했습니다. 그런 이후에, login에서는 userinfo라는 파일을 없으면 만들면서 root에 관한 정보만 적어주고, 있다면, 그 파일의 정보를 받아서 사용합니다. 이 때, 파일은 한칸씩 띄어져 있는 형태로, “아이디 비밀번호 아이디 비밀번호 “의 형태로 저장됩니다. 그렇게 해서, 파일을 한문장으로 받은 후에, 지난번 과제처럼 파싱을 해서 각 정보들을 저장했습니다. 이 때도, 배열을 나눠서 저장하지 않고, 하나의 배열에 아이디와 비밀번호를 저장해서 하나의 계정당 2칸씩 차지할 수 있도록 했습니다. 그래서 정보를 확인할 때도 두칸씩 확인하는 것을 볼 수 있습니다. 무한루프를 돌면서 아이디와 비밀번호가 해당하는 게 존재한다면 몇 번째 user인지 kernel에 숫자로 적어주는 enrolluser라는 시스템콜을 실행하고 sh.c를 실행합니다.

2) User add / User delete

Useradd와 userdel은 둘 다 모두 fs.c에 있습니다. Useradd와 userdel 같은 경우에는 본격적으로 파일을 실행하기 전에, login처럼 값들을 미리 읽어오는 과정이 필요합니다. 왜냐하면, 제 코드에서는 한 번에 kernel메모리에서 유저 정보를 갱신해주지 않기 때문입니다. 그리고 fs.c에서 해당 프로그램들이 실행됬다는 것은 갱신이 필요하다는 의미이므로 할 때마다, 파일을 읽어서 user에 관한 정보를 갱신해줬습니다. 이 과정에서 readi를 사용해서 파일을 읽을 수 있었습니다. 그리고 enrolluser를 통해서, 갱신된 nuser는 몇번째 유저가 사용중인지 확인할 수 있는 번호로, 이 번호에 해당하는 user와 root가 strncmp를 통해서 같을 때만 실행할 수 있도록 했습니다.

useradd에서는 해당하는 정보를 받은 후에, writei를 써서 userinfo를 갱신해 줬고, directory를 만드는 것은 sysfile.c에서 create함수를 make\_inode란 함수로 이름만 바꿔서 집어넣은 후, 그 코드를 사용해서 만들었습니다.

userdel에서는 지울 정보를 받은 후에, 앞에서 받아 놓은 정보를 바탕으로 파일에 덮어쓰기를 해줍니다. 덮어써줬는데, ip->size값이 바뀌지 않았다면, 해당하는 정보가 없었던 것이기 때문에 -1을 return합니다.

3) File Mode

Fs.h에 해당하는 권한을 저장을 한 후에, inode와 dinode그리고 stat에서 각각의 구조체에, owner와 mode라는 int 값을 추가했습니다. 그 와 동시에 mkfs.c는 dinode의 값을 512의 약수가 되도록 요구하기 때문에, fs.h에서 NDIRECT값을 26으로 바꿔서 총 128의 크기를 가지도록 했습니다. 그리고 설정할 dinode의 값을 ialloc에서 추가해줘서 mode는 61을 owner는 0을 가리키도록 했습니다.

제가 구현한 권한이 작동하는 부분은 create함수, exec함수, open함수입니다. 명세대로 구현했고, 전부 sysfile.c에서 해결했습니다. exec함수도 sys\_exec에서 해결했습니다. 이 때 사용하는 t값은 nuser와 같은 개념으로 enrolluser를 사용할 때, 동시에 갱신해주는 부분입니다. 그래서 다음과 같이 살행합니다

1. root가 실행자면 t=0이고 이 때는 user권한을 확인한다.

2. 아니라면, t값이 ip->owner와 같은지 확인하고 같다면 user권한을 확인하고 다르다면 other 권한을 확인한다.

이 때, 권한의 비교는 and 연산자로 했으며, 권한을 확인했는데 0일 때, 예외처리를 하도록 구현했습니다.

4) Change Mode

chmod라는 시스템콜을 구현했고, fs.c에 구현했습니다. Nuser가 root이거나 ip->owner일 경우에 값을 바꿀 수 있도록 했고, 아니라면, -1을 return하게 했습니다.

5) Modification of ls

주로 ls.c를 고쳤고, 위와 같이 받은 mode를 32부터 서서히 나눠서 해당 값에 맞게 rwx를 쓰도록 만들었습니다. 그리고 그 앞에서 type을 확인하고 d값을 쓰기도 합니다. 앞에서 말했다시피 ls.c도 userinfo라는 파일을 읽어서 user에 저장하고, 각각의 ip의 owner를 읽어서 프린트하도록 만들었습니다.

2. 구현

1) login.c

int fd, n;

char \*buf="root 1234 ";

if((fd=open("userinfo",O\_RDONLY))<0) ***-> file이 있는지 확인합니다.***

{

close(fd);

fd=open("userinfo",0x202); ***-> 없다면 파일을 만듬과 동시에 buf에 해당하는 정보를 씁니다***.

int w=write(fd,buf,strlen(buf));

if (w<0) printf(1,"write error in login\n");

close(fd);

}

fd=open("userinfo",O\_RDONLY);

while((n = read(fd, ref, sizeof(ref))) > 0) {

user[0]=strtok(ref," ");

user\_cnt++;

for(int i=1;;i++)

{

user[i]=strtok(0," ");

if(user[i]==0) break;

user\_cnt++;

} ***-> 파일을 읽고, user정보를 받아옵니다.***

}

close(fd);

while(1)

{

int pass=0;

printf(1,"username: ");

char ID[15], pwd[15];

getcmd(ID, sizeof(ID));

ID[strlen(ID)-1]=0;

printf(1,"password: ");

getcmd(pwd, sizeof(pwd));

pwd[strlen(pwd)-1]=0;

for(int i=0;i<user\_cnt;i+=2)

{

//printf(1,"id:%s passwd:%s\n", user[i], user[i+1]);

if(strcmp(ID,user[i])==0&&strcmp(pwd,user[i+1])==0)

{

pass=1;

enrolluser(i/2);

break;

}

}

if(pass==1) break;

}

***-> while문을 돌면서 입력 받은 값이 있나 살피는데, 아이디와 비밀번호가 연속적으로 저장되므로 한번에 i값을 2씩 키워야 합니다.***

***그 이후의 코드들은 sh를 실행하는 코드이므로 생략하겠습니다.***

2) fs.c

char buf[512]; ***-> 파일을 읽어들일 때 사용하는 buffer입니다.***

char \*user[20]; ***-> 이 곳에 아이디와 비밀번호가 순차적으로 저장됩니다.***

int user\_cnt=0, isin=0;

int nuser=0; ***-> 현재 사용중인 user의 번호입니다.***

void enrolluser(int user)

{

nuser=user;

}

int chmod(char \*pathname, int mode)

{

//struct inode \*ip=namei("userinfo");

//ilock(ip);

//cprintf("%d\n", mode);

//if(mode==MODE\_WUSR)

//cprintf("nuser: %d\n", nuser);

begin\_op();

struct inode \*ip=namei(pathname);

ilock(ip);

//cprintf("nuser: %d owner: %d\n", nuser, ip->owner);

if(nuser&&nuser!=ip->owner)  ***-> 여기서 root가 아닌데 owner도 아니면 -1을 return 합니다***.

{

iunlockput(ip);

end\_op();

return -1;

}

//cprintf("1\n");

//begin\_op();ls

//cprintf("2\n");

ip->mode=mode;

//cprintf("3\n");

iupdate(ip);

//cprintf("4\n");

iunlockput(ip);

//cprintf("5\n");

end\_op();

return 0;

}

int userdel(char \*username)

{

if(strncmp("root", username, strlen(username))==0) return -1; ***-> root인지 확인합니다***.

***;;;;;;;;;;;;;;;;;; -> 중간에 값을 입력받는건 login에 있는 코드와 동일하므로 생략합니다.***

begin\_op();

ilock(ip);

int len=0;

int s=ip->size;

//cprintf("ip->size: %d\n", s);

for(int i=0;i<user\_cnt;i+=2)

{

if(strncmp(user[i],username,strlen(username))==0) continue;

***-> 지우는 것이면 파일에 덮어쓰지 않습니다***

writei(ip, user[i], len, strlen(user[i]));

len+=strlen(user[i]);

writei(ip, " ", len, strlen(" "));

len++;

writei(ip, user[i+1], len, strlen(user[i+1]));

len+=strlen(user[i+1]);

writei(ip, " ", len, strlen(" "));

len++;

//cprintf("len?: %d\n", len);

}

//cprintf("ip->size: %d\n", ip->size);

//for(int i=len;i<s;i++)

//{

// writei(ip, " ", i, strlen(" "));

//}

ip->size=len;

iunlock(ip);

end\_op();

if(s==ip->size) return -1; ***-> size가 같다면, 지우고 싶었던 게 없던 것이므로 -1을 return합니다***

return 0;

}

int useradd(char \*username, char\*password)

{

***;;;;;;;;;;;;;;;;;; -> 중간에 값을 입력받는건 login에 있는 코드와 동일하므로 생략합니다***

if(strncmp("root", user[nuser\*2], strlen(user[nuser\*2]))!=0) return -1; ***-> root인지 확인***

if(user\_cnt/2>10) return -1;

***-> user\_cnt는 user와 함께 커지므로 이 값을 2로 나눈 값이 10보다 커지지 않게 관리합니다***.

//cprintf("before\n");

for(int j=0;j<user\_cnt;j+=2)

{

//cprintf("user: %s\n", user[j]);

if(strncmp(user[j],username, strlen(username))==0)

{

//cprintf("in\n");

return -1;

}

}

//cprintf("after\n");

begin\_op();

ilock(ip);

writei(ip, username, ip->size, strlen(username));

writei(ip, " ", ip->size, strlen(" "));

writei(ip, password, ip->size, strlen(password));

writei(ip, " ", ip->size, strlen(" "));

iunlock(ip);

end\_op();

***-> 해당하는 값을 써줍니다. 저는 띄어쓰기 단위로 파싱하므로 띄어쓰기도 집어넣어 줍니다***.

//cprintf("before\n");

struct inode \*id;

begin\_op();

if((id = make\_inode(username, T\_DIR, 0, 0)) == 0) -> ***make\_inode= create함수 입니다***

{

end\_op();

//cprintf("in if sementics\n");

return -1;

}

//cprintf("afer\n");

//ilock(id);

id->owner=user\_cnt/2;

id->mode=(63-MODE\_WOTH);

//cprintf("1-1\n");

iupdate(id);

//cprintf("1-2\n");

iunlockput(id);

//cprintf("1-3\n");

end\_op();

//char \*a="test";

//int m=3;

//cprintf("chmod: %d\n", chmod(a,m));

return 0;

}

***이 외에도 iupdate와 ilock, stati에서 owner와 mode를 주고 받는 것도 구현되어있습니다.***

3) fs.h

#define NDIRECT 26

#define NINDIRECT (BSIZE / sizeof(uint))

#define MAXFILE (NDIRECT + NINDIRECT)

// On-disk inode structure

struct dinode {

short type; // File type

short major; // Major device number (T\_DEV only)

short minor; // Minor device number (T\_DEV only)

short nlink; // Number of links to inode in file system

uint size; // Size of file (bytes)

uint addrs[NDIRECT+1]; // Data block addresses

uint owner;

uint mode;

};

***NINDIRECT값을 바꿔줬고, struct안에 추가했습니다. Owner와 mode를 추가한건 inode와 stat struct에서도 해주었습니다***.

4) sysfile.c

***우선 이 곳에 구현한 시스템콜들의 wrapper function들이 존재합니다. 하지만 간단하니까 설명을 제하고 제가 create, open, exec에 구현한 것을 보여드리겠습니다.***

static struct inode\*

create(char \*path, short type, short major, short minor)

{

//cprintf("in create\n");

struct inode \*ip, \*dp;

char name[DIRSIZ];

if((dp = nameiparent(path, name)) == 0)

return 0;

ilock(dp);

if((ip = dirlookup(dp, name, 0)) != 0){

iunlockput(dp);

ilock(ip);

if(type == T\_FILE && ip->type == T\_FILE)

{

***if(t==0 && !(MODE\_WUSR & ip->mode))***

***{***

***//cprintf("1\n");***

***iunlockput(ip);***

***return 0;***

***}***

***if(t>0 && ip->owner==t && !(MODE\_WUSR & ip->mode))***

***{***

***//cprintf("2\n");***

***iunlockput(ip);***

***return 0;***

***}***

***else if(t>0 && ip->owner!=t && !(MODE\_WOTH & ip->mode))***

***{***

***//cprintf("3\n");***

***iunlockput(ip);***

***return 0;***

***}***

***-> 만약 파일이 있다면, 쓰기 권한이 있는지 확인합니다. Root는 owner와 동일한 권한을 검사합니다.***

iunlockput(ip);

return ip;

}

//cprintf("4\n");

iunlockput(ip);

return 0;

}

//ilock(dp);

***if(t==0 && !(MODE\_WUSR & dp->mode))***

***{***

***//cprintf("1\n");***

***iunlockput(dp);***

***return 0;***

***}***

***if(t>0 && dp->owner==t && !(MODE\_WUSR & dp->mode))***

***{***

***iunlockput(dp);***

***return 0;***

***}***

***else if(t>0 && dp->owner!=t && !(MODE\_WOTH & dp->mode))***

***{***

***iunlockput(dp);***

***return 0;***

***}***

***-> 파일이 없는 상황에서 해당 디렉토리에 쓰기 권한이 있는지 확인합니다.***

if((ip = ialloc(dp->dev, type)) == 0)

panic("create: ialloc");

//cprintf("1\n");

ilock(ip);

ip->major = major;

ip->minor = minor;

ip->nlink = 1;

//cprintf("2\n");

if(t>=0)

{

//cprintf("%d %d\n", ip->owner, t);

ip->owner = t;

}

else ip->owner = 0; ***-> 이 코드는 밑에 t값이 사용되는데, 음수일 때가 있어서 해준 예외처리 입니다. 참고로 t값은 nuser와 같은 역할을 합니다. create함수에서 파일을 만들 때 owner와 mode를 설정해 줍니다.***

ip->mode = MODE\_RUSR+MODE\_WUSR+MODE\_ROTH;

iupdate(ip);

//cprintf("3\n");

if(type == T\_DIR){ // Create . and .. entries.

dp->nlink++; // for ".."

//id->owner=user\_cnt/2;

ip->mode=(63-MODE\_WOTH);

iupdate(dp);

// No ip->nlink++ for ".": avoid cyclic ref count.

if(dirlink(ip, ".", ip->inum) < 0 || dirlink(ip, "..", dp->inum) < 0)

panic("create dots");

}

if(dirlink(dp, name, ip->inum) < 0)

panic("create: dirlink");

//cprintf("4\n");

iunlockput(dp);

//cprintf("5\n");

return ip;

}

int

sys\_open(void)

{

char \*path;

int fd, omode;

struct file \*f;

struct inode \*ip;

if(argstr(0, &path) < 0 || argint(1, &omode) < 0)

return -1;

begin\_op();

//cprintf("in open %s\n", path);

if(omode & O\_CREATE){

ip = create(path, T\_FILE, 0, 0);

if(ip == 0){

end\_op();

return -1;

}

}

else {

if((ip = namei(path)) == 0){

end\_op();

return -1;

}

***ilock(ip);***

***if(omode == O\_RDONLY || omode == O\_RDWR){***

***if(t==0 && !(MODE\_RUSR & ip->mode))***

***{***

***//cprintf("1\n");***

***iunlockput(ip);***

***end\_op();***

***return 0;***

***}***

***if(t>0 && ip->owner==t && !(MODE\_RUSR & ip->mode))***

***{***

***iunlockput(ip);***

***end\_op();***

***return -1;***

***}***

***if(t>0 && ip->owner!=t && !(MODE\_ROTH & ip->mode))***

***{***

***iunlockput(ip);***

***end\_op();***

***return -1;***

***}***

***}***

***if(omode == O\_WRONLY || omode == O\_RDWR){***

***if(t==0 && !(MODE\_WUSR & ip->mode))***

***{***

***//cprintf("1\n");***

***iunlockput(ip);***

***end\_op();***

***return 0;***

***}***

***if(t>0 && ip->owner==t && !(MODE\_WUSR & ip->mode))***

***{***

***iunlockput(ip);***

***end\_op();***

***return -1;***

***}***

***if(t>0 && ip->owner!=t && !(MODE\_WOTH & ip->mode))***

***{***

***iunlockput(ip);***

***end\_op();***

***return -1;***

***}***

***}***

***-> 각각의 상황에 맞게 권한을 검사하는 모습입니다. 형태는 create함수와 같습니다.***

if(ip->type == T\_DIR && omode != O\_RDONLY){

iunlockput(ip);

end\_op();

return -1;

}

}

if((f = filealloc()) == 0 || (fd = fdalloc(f)) < 0){

if(f)

fileclose(f);

iunlockput(ip);

end\_op();

return -1;

}

iunlock(ip);

end\_op();

f->type = FD\_INODE;

f->ip = ip;

f->off = 0;

f->readable = !(omode & O\_WRONLY);

f->writable = (omode & O\_WRONLY) || (omode & O\_RDWR);

return fd;

int

sys\_exec(void)

{

char \*path, \*argv[MAXARG];

int i;

uint uargv, uarg;

if(argstr(0, &path) < 0 || argint(1, (int\*)&uargv) < 0){

return -1;

}

memset(argv, 0, sizeof(argv));

for(i=0;; i++){

if(i >= NELEM(argv))

return -1;

if(fetchint(uargv+4\*i, (int\*)&uarg) < 0)

return -1;

if(uarg == 0){

argv[i] = 0;

break;

}

if(fetchstr(uarg, &argv[i]) < 0)

return -1;

}

begin\_op();

struct inode \*ip=namei(path);

if(ip==0)

{

//cprintf("???????\n");

end\_op();

return -1;

}

ilock(ip);

//cprintf("%s %d %d %d\n",path, ip->owner, t, (ip->mode));

//cprintf("!(ip->mode&MODE\_XUSR): %d\n", (ip->mode&MODE\_XUSR));

//cprintf("!(ip->mode&MODE\_XOTH): %d\n", (ip->mode&MODE\_XOTH));

***if(t<=0)***

***{***

***iunlockput(ip);***

***end\_op();***

***return exec(path, argv);***

***}***

***else if(ip->owner==t && !(ip->mode&MODE\_XUSR))***

***{***

***//cprintf("1 ip:%d t:%d\n", ip->owner, t);***

***iunlockput(ip);***

***end\_op();***

***return -1;***

***}***

***else if(ip->owner!=t && !(ip->mode&MODE\_XOTH))***

***{***

***//cprintf("2 ip:%d t:%d\n", ip->owner, t);***

***iunlockput(ip);***

***end\_op();***

***return -1;***

***}***

***-> exec에서는 약간 다른데 t의 초기값을 음수로 잡은 이유는 적어도 login까지는 안전하게 실행하기 위함입니다. 로그인을 하게 된다면, t값이 양수가 되서 그 이후에 권한들을 비교할 수 있습니다.***

iunlockput(ip);

end\_op();

//cprintf("final exec\n");

return exec(path, argv);

}

5) mkfs.c

uint

ialloc(ushort type)

{

uint inum = freeinode++;

struct dinode din;

bzero(&din, sizeof(din));

din.type = xshort(type);

din.nlink = xshort(1);

din.size = xint(0);

din.owner = xint(0);

//printf("mode: %d\n", MODE\_RUSR+MODE\_WUSR+MODE\_ROTH);

din.mode = xint(61);

//din.mode =

winode(inum, &din);

return inum;

}

***-> ialloc을 바꿔서 있는 파일의 root는 owner, 권한은 WOTH빼고 전부 가지도록 했습니다.***

6) ls.c

case T\_DIR:

if(strlen(path) + 1 + DIRSIZ + 1 > sizeof buf){

printf(1, "ls: path too long\n");

break;

}

strcpy(buf, path);

p = buf+strlen(buf);

\*p++ = '/';

while(read(fd, &de, sizeof(de)) == sizeof(de)){

if(de.inum == 0)

continue;

memmove(p, de.name, DIRSIZ);

p[DIRSIZ] = 0;

if(stat(buf, &st) < 0){

printf(1, "ls: cannot stat %s\n", buf);

continue;

}

printf(1, "%s ", fmtname(buf));

int isconsole=0;

if(st.type==T\_FILE) printf(1, "-");

else if(st.type==3)

{

isconsole=1;

printf(1, "c");

}

else printf(1, "d"); ***-> 파일이 디렉토리면 d를 먼저 프린트합니다.***

int tmp=st.mode;

//printf(1, "%d ", tmp);

int k=32;

while(k)

{

if(tmp/k==1&&!isconsole)

{

if(k==32||k==4) printf(1, "r");

if(k==16||k==2) printf(1,"w");

if(k==8||k==1) printf(1,"x");

tmp-=k;

}

else printf(1,"-");

k/=2;

}

***-> mode를 받아서 32부터 나누면서 몫이 있다면 그에 해당하는 문자를 프린트합니다.***

if(!isconsole) printf(1, " %s %d %d %d\n", user[st.owner\*2], st.type, st.ino, st.size);

else printf(1, " %d %d %d\n", st.type, st.ino, st.size);

//break;

}

break;

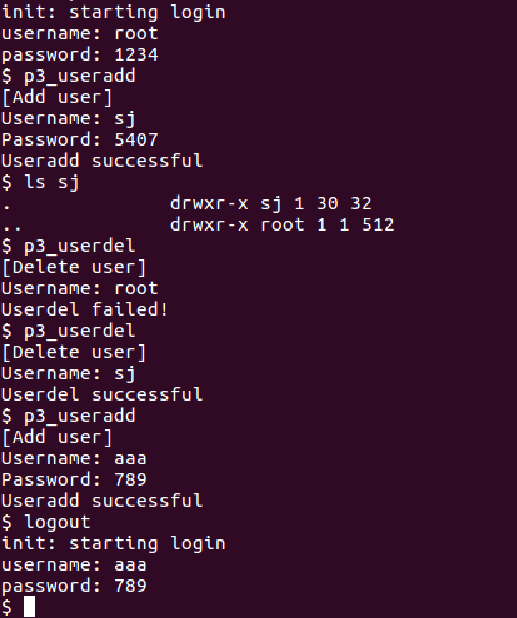
}

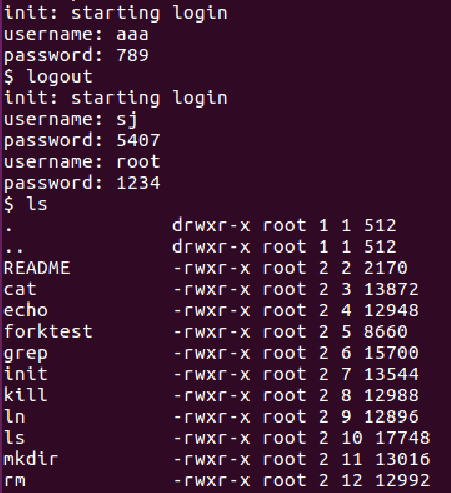
close(fd);

}

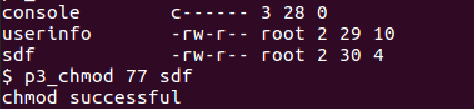
3. 실행 결과

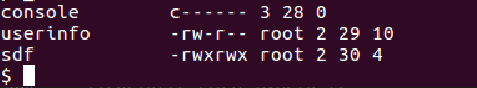
1) useradd and userdel

 -> sj란 계정을 만들고 지우고 aaa라는 계정을 새로 만들고 다시 시작했습니다.

-> 삭제된 계정으로의 로그인은 안되고 aaa로의 로그인은 여전히 가능한 것을 볼 수 있습니다.

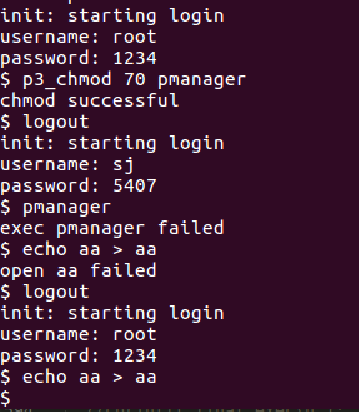
2) chmod





-> mode가 설정한대로 바뀐 것을 볼 수 있습니다.

3) mode implementation



-> pmanager의 권한을 바꾸자 sj가 pmanager를 실행시키지 못합니다 그리고 root의 주인인 sj가 아니므로 파일을 만들어내지도 못합니다 그에 반해 root는 sj가 못한 것들을 전부 할 수 있습니다.

4. 트러블 슈팅

1) begin\_op(), end\_op()

어느 함수에 이걸 써야되는지 몰라서 한참을 헤맸습니다. 제가 if문을 많이 써서 예외처리를 했는데 빠져버리면 qemu 자체가 터져버리는 현상이 많았습니다. Inode와 관련되서는 그냥 쓰고 보는게 마음이 편한 것 같기도 합니다.

2) ilock

Ilock은 좀 특별한 함수 인 것 같습니다. 그러니까 제가 ilock을 쓰지 않고 단순히 namei만 쓴다면 해당 inode의 정보가 제대로 존재하지 않더군요. 그래서 sys\_exec을 할 때, 권한을 비교한 값이 항상 0이 되는 일이 계속해서 생겼습니다. 다행히 ilock을 통해서 값들을 제대로 받아올 수 있어서 해결할 수 있었습니다.