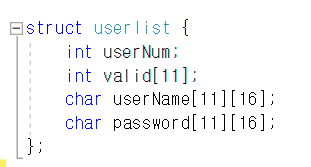
**Project 3**

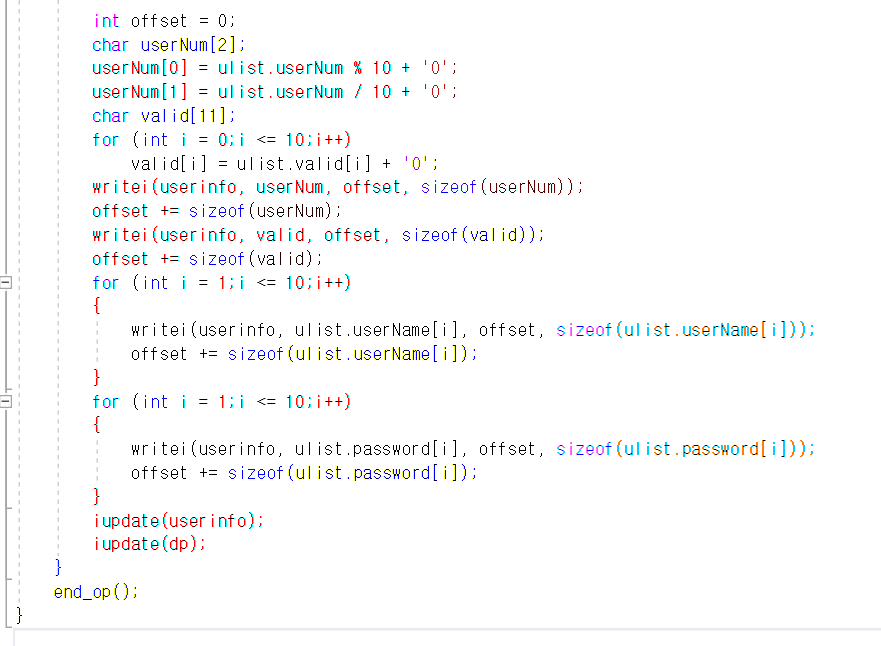
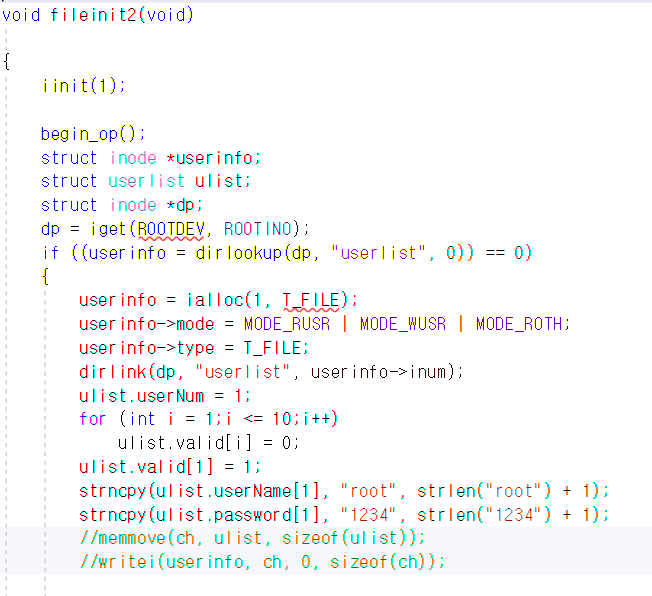
2018008613 안상욱

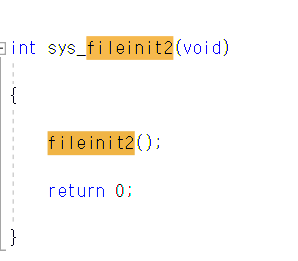
1. 프로그램

* User Accounts

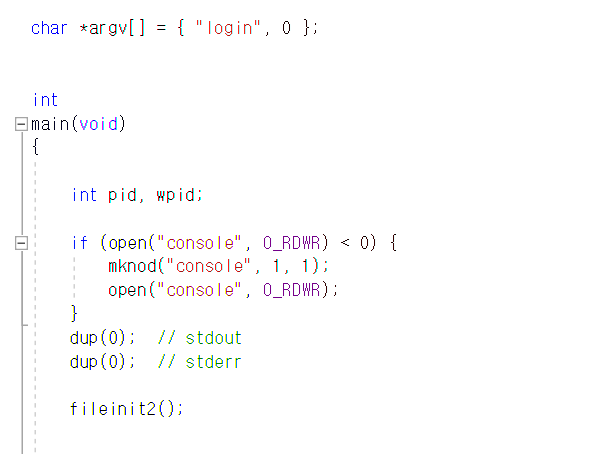


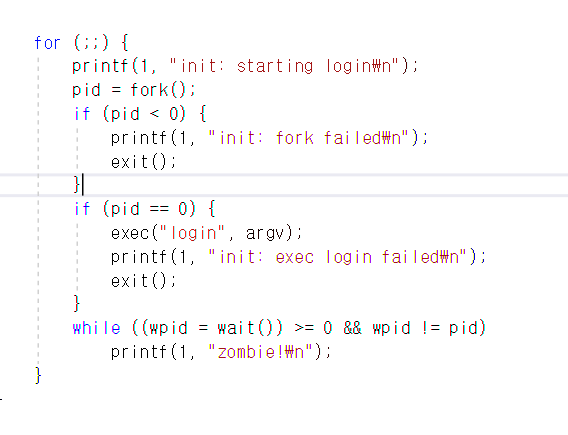
먼저 user 목록을 저장할 struct userlist를 만들었습니다.



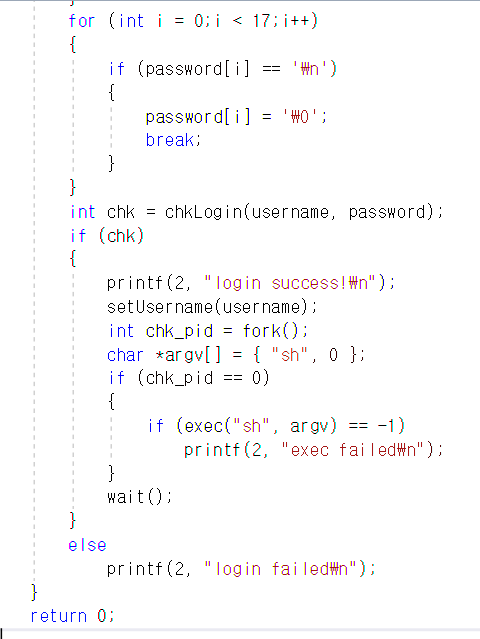


그리고 init.c에서 실행시킬 시스템 콜 함수인 fileinit2를 fs.c에 정의해주었습니다. Login이 시작되기 전 userlist라는 파일이 root directory에 있다면, 아무 작업도 하지 않고 만약 없다면, root, 1234라는 정보만을 가지고 있는 userlist 정보를 담고 있는 userlist 파일을 만들어서 root directory에 추가해주었습니다.

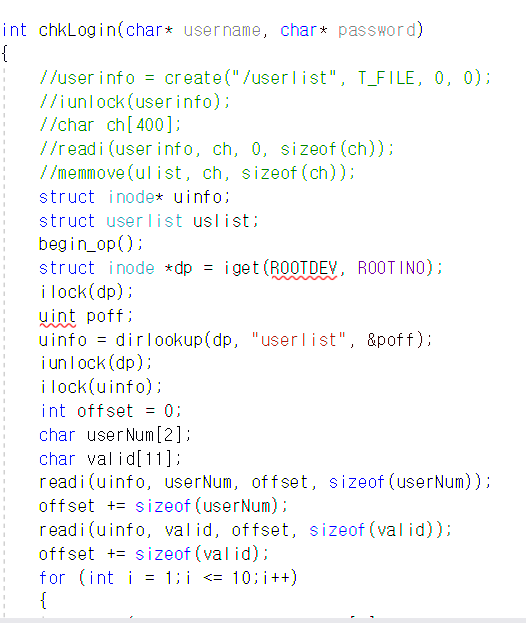




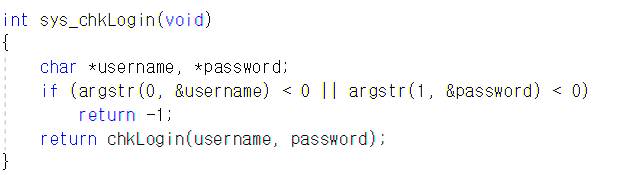
Init.c는 fileinit2() 시스템 콜을 호출해서 userlist가 없다면 userlist를 생성해주고, sh 대신 login.c라는 유저 프로그램을 만들어서 이 프로그램을 실행시켜주었습니다.



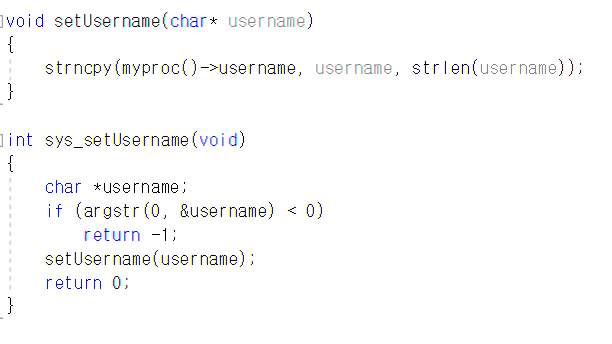
Login.c에서는 username, password를 입력받아서 이와 일치하는 유저가 userlist안에 있는지를 chkLogin 시스템콜 함수를 통해 확인해주고, 만약 있다면 sh.c를 실행해주고, 없다면 다시 while문을 돌아 username, password를 입력받도록 했습니다. 그리고 sh에서 logout을 입력해서 빠져나오는 경우에도 다시 while문을 돌아 username, password를 새로 입력받도록 했습니다. 그리고 setUsername 시스템 콜을 통해 sh를 실행하는 사람에게 username에 해당하는 사람의 권한을 주도록 했습니다.



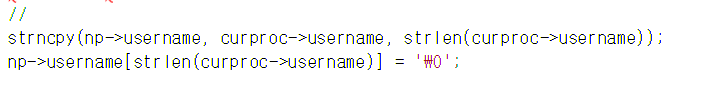




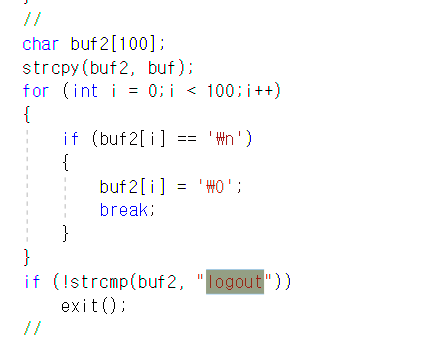
Fs.c 함수에 위치해있는 chkLogin 시스템콜 함수는 username과 password를 입력받아 이에 해당하는 유저가 userlist에 있는지를 확인해서 있다면 1을, 없으면 0을 return해줍니다.



그리고 proc.c 함수 내에 있는 setUsername을 통해 현재 실행하는 프로세스의 권한을 username으로 설정해서 sh를 실행하도록 했습니다.

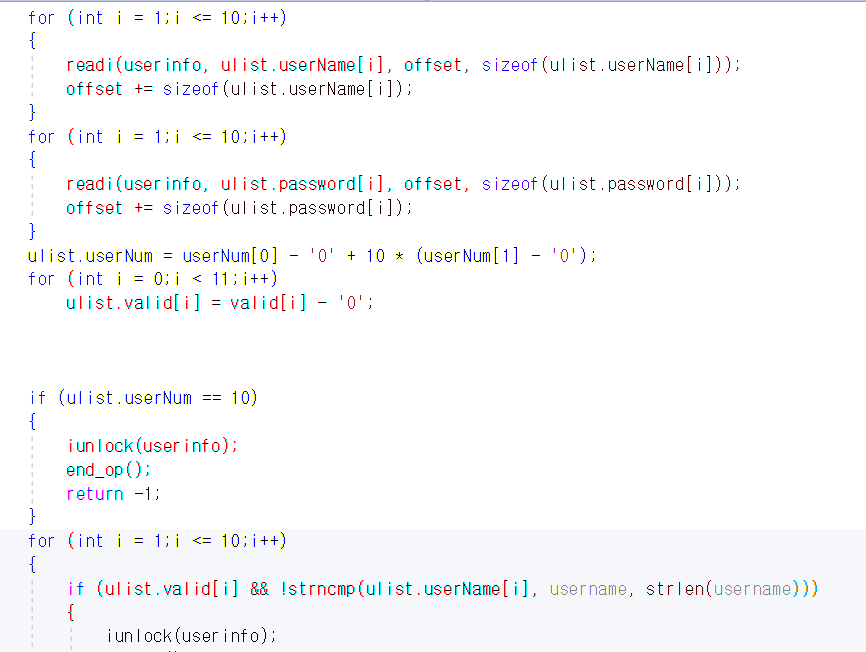


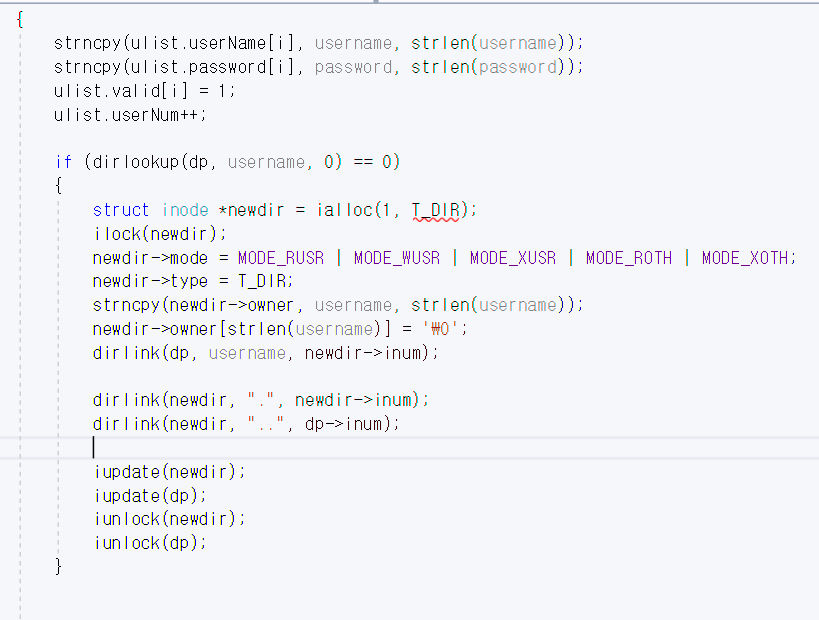
그리고 proc.c 안에 있는 fork 함수에서도 curproc->username을 복사해서 실행중인 유저의 이름을 저장하도록 해주었습니다.

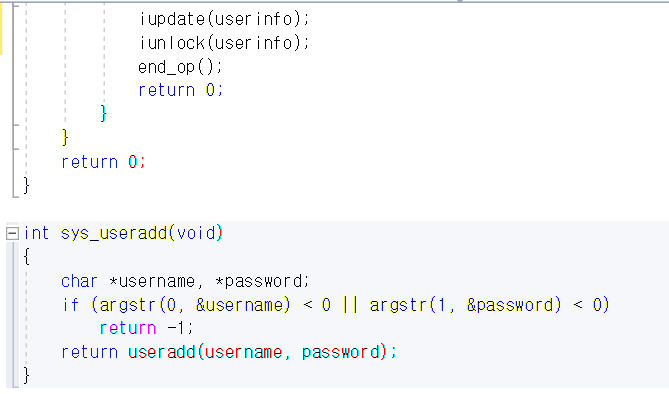
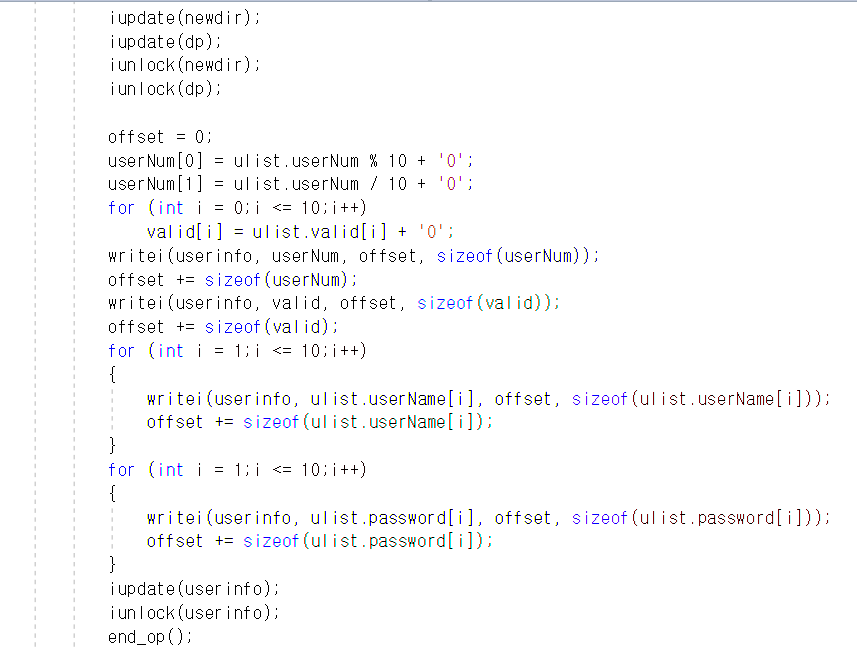


그리고 sh.c의 main 내부에 다음과 같은 코드를 추가해서 logout이 입력된다면, sh 프로그램을 종료하도록 했습니다.

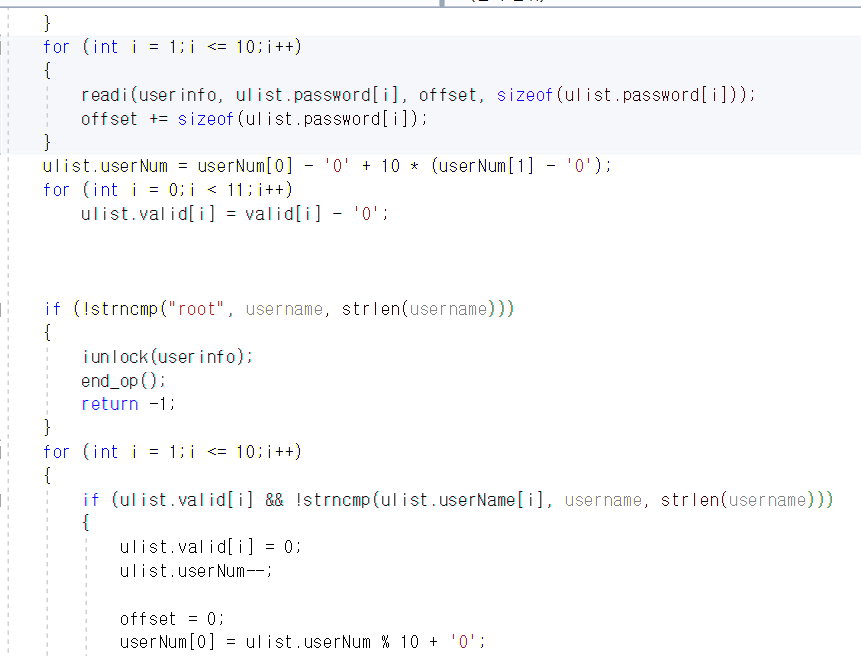


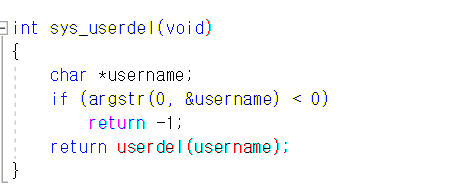
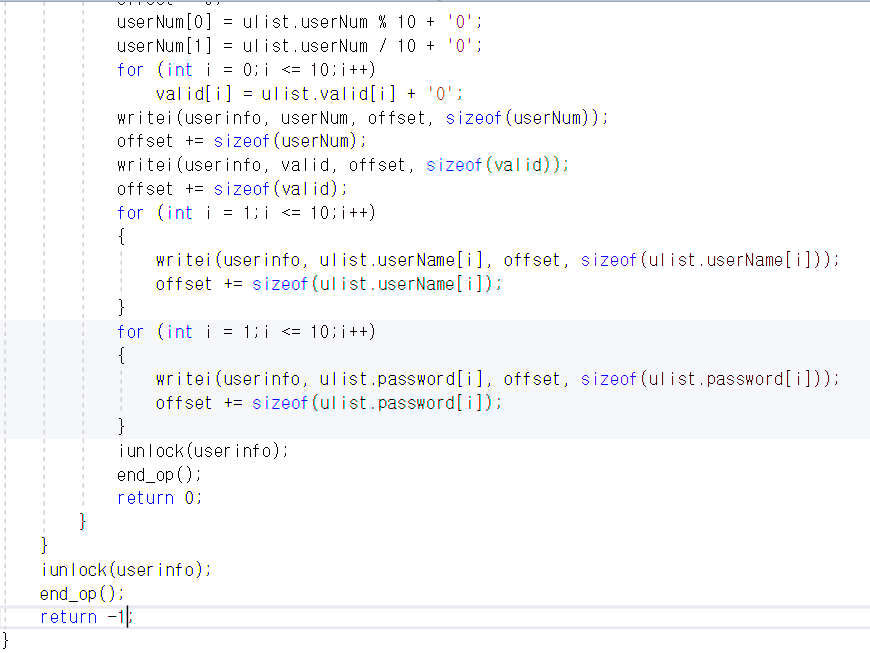






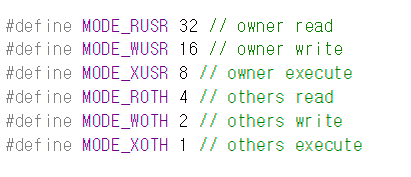




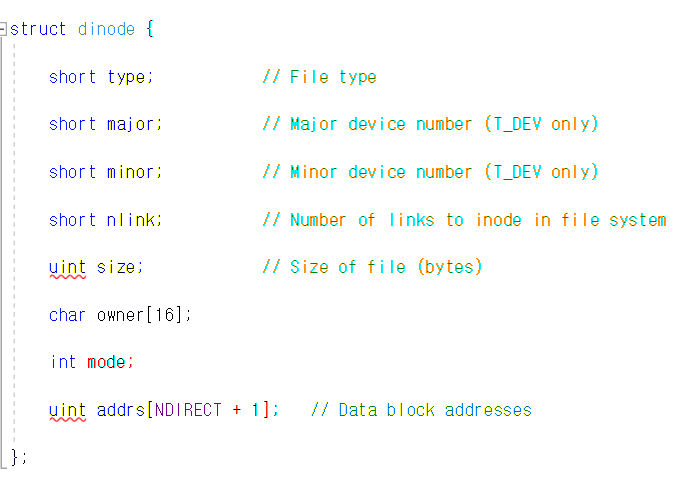


그리고 fs.c에 useradd, userdel 시스템 콜을 구현했습니다. 이 함수를 호출한 유저가 root가 아니면 실패하도록 했고, 이미 유저가 10명인데 useradd를 하는 경우, 중복된 이름을 add하는 경우, del할 user가 존재하지 않는 경우, root를 삭제하려는 경우에는 실패하도록 했습니다. Add가 성공한 경우 root directory에 username에 해당하는 directory가 존재하지 않는다면, directory를 생성해주었습니다.

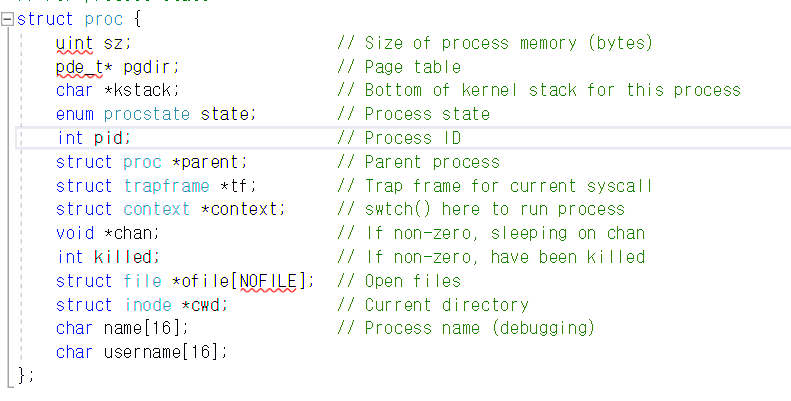
* File Mode



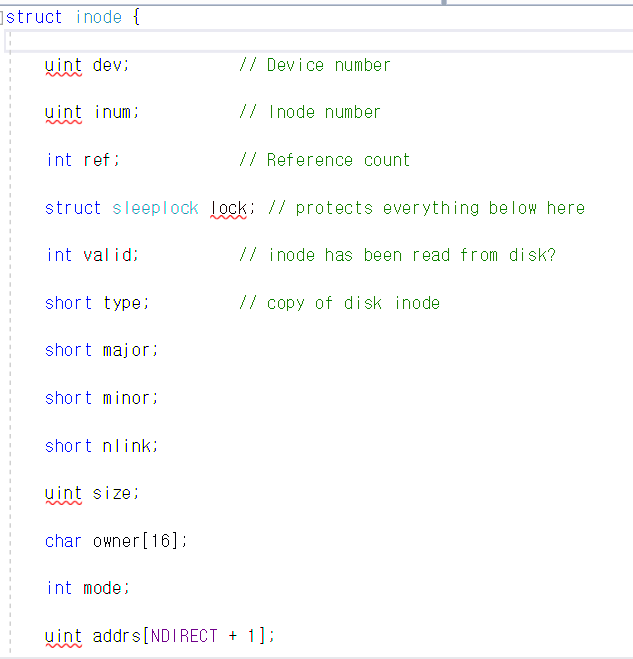




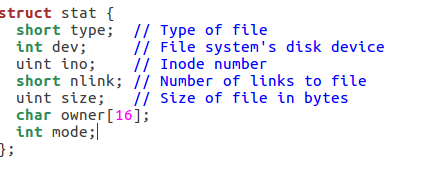
Fs.h에서는 MODE에 대한 define 목록을 추가해주었고, size와 owner 목록을 추가해주었습니다. Size, owner는 4 + 16 = 20바이트이므로, NDIRECT를 5개 줄여서 DINODE의 크기가 원래와 같은 64바이트로 유지시켜주었습니다.



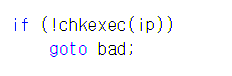
Proc.c의 struct proc에는 username 항목을 추가해주어 실행한 사람의 이름을 담고 있도록 해주었습니다.



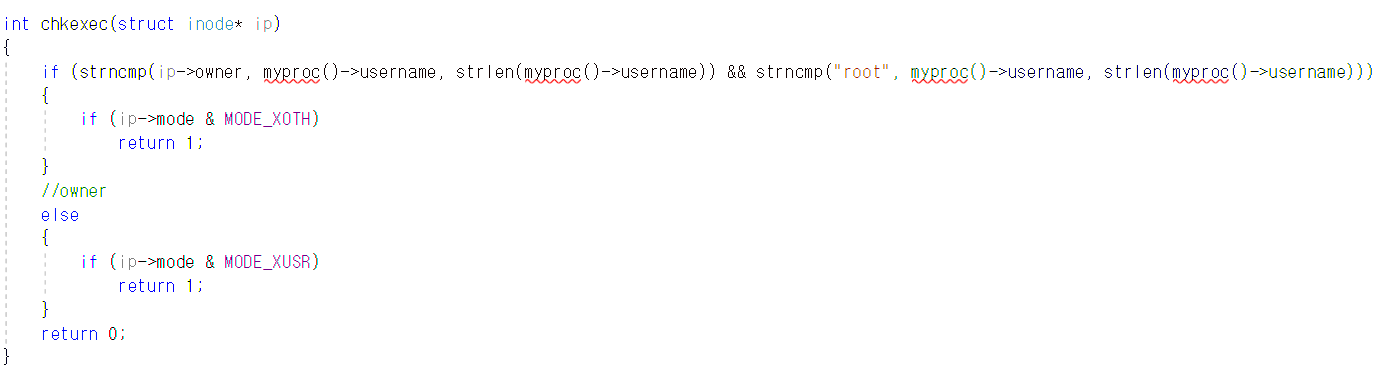
File.h의 inode에도 dinode처럼 owner, mode를 추가해주었습니다.



Stat.h에도 owner, mode 항목을 추가해주었습니다.

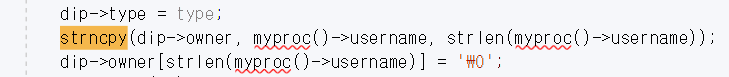


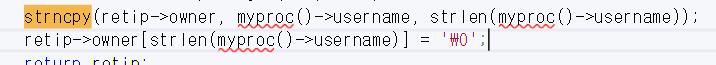
Exec.c에 fs.c에 정의되어있는 chkexec함수를 통해 실행 권한이 있는지를 판단합니다.



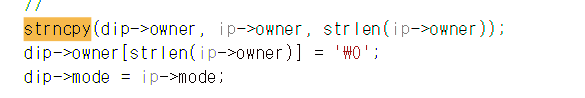
Fs.c에 정의되어있는 chkexec 함수에서는 만약 owner이거나 root인데 MODE\_XUSR 권한이 있다면 1을 RETURN, 그렇지않다면 0을 RETURN해줍니다.

만약 others인데 MODE\_XOTH 권한이 있다면 1을 RETURN, 그렇지 않다면 0을 RETURN해줍니다.

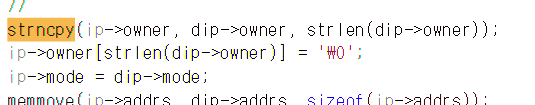




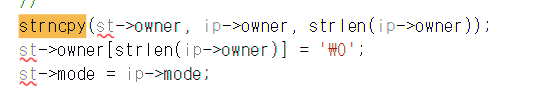
ialloc에서 inode와 dinode의 owner 정보를 myproc()->username을 통해 할당해줍니다.



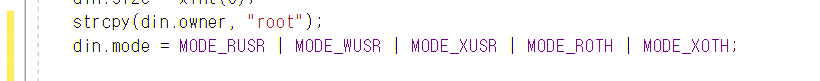
iupdate에서도 owner, mode정보를 함께 업데이트 시켜줍니다.



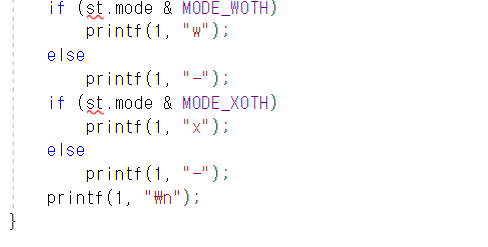
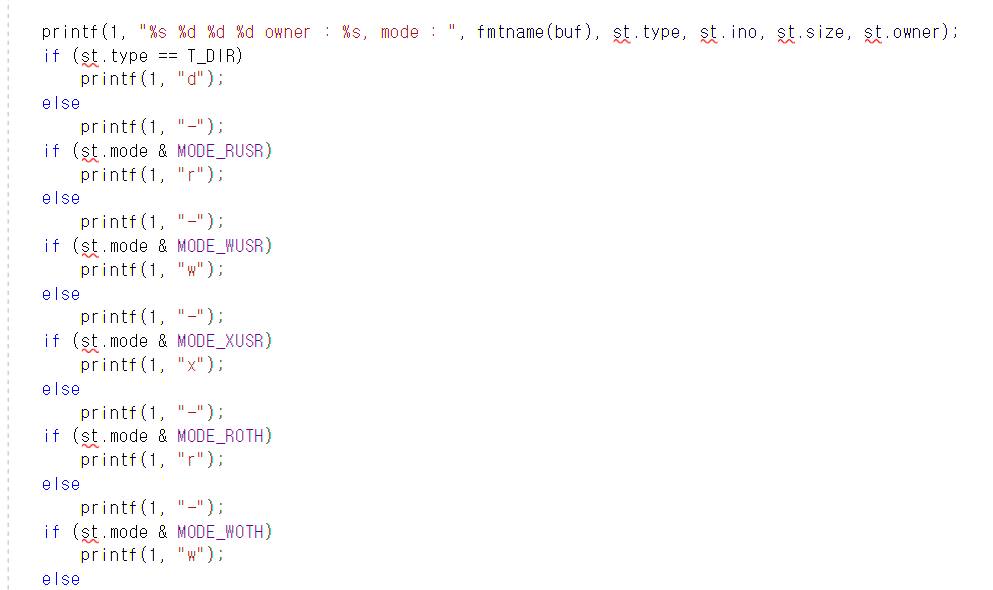
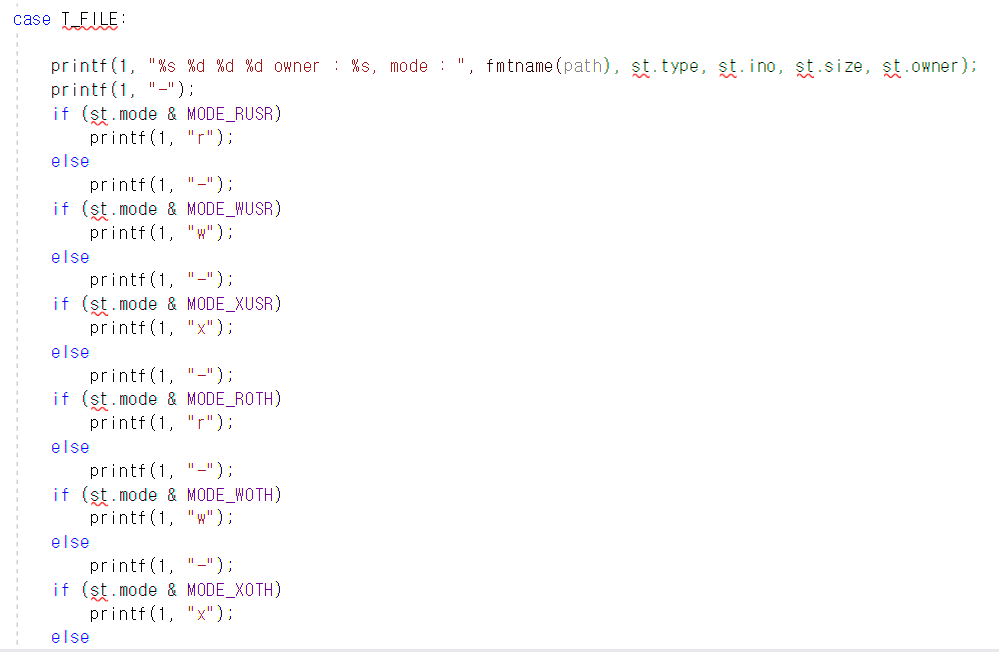
Ilock 함수에서도 owner, mode 정보를 함께 업데이트 시켜줍니다.



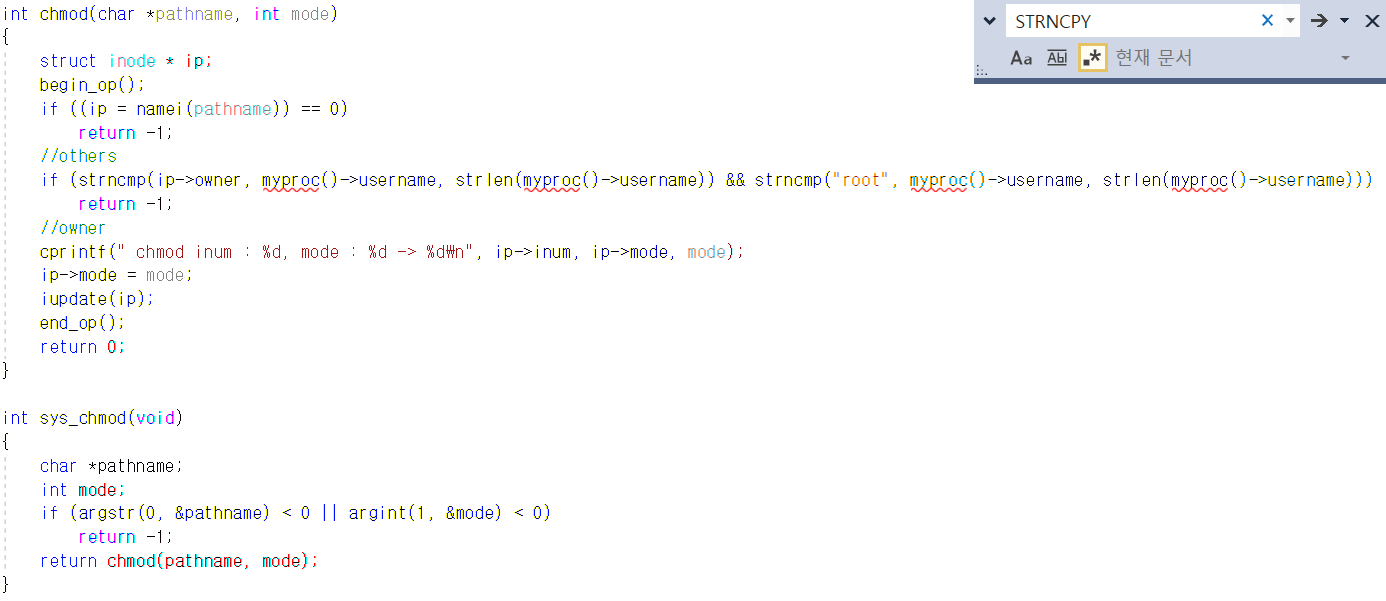
Stati 함수에서도 owner, mode 정보를 함꼐 업데이트 시켜줍니다.



Mkfs.c의 ialloc 함수에서도 owner, mode 정보를 초기화 시켜줍니다.



그리고 ls.c의 ls 함수에서도 owner와 mode를 추가로 출력해주도록 바꾸어주었습니다.

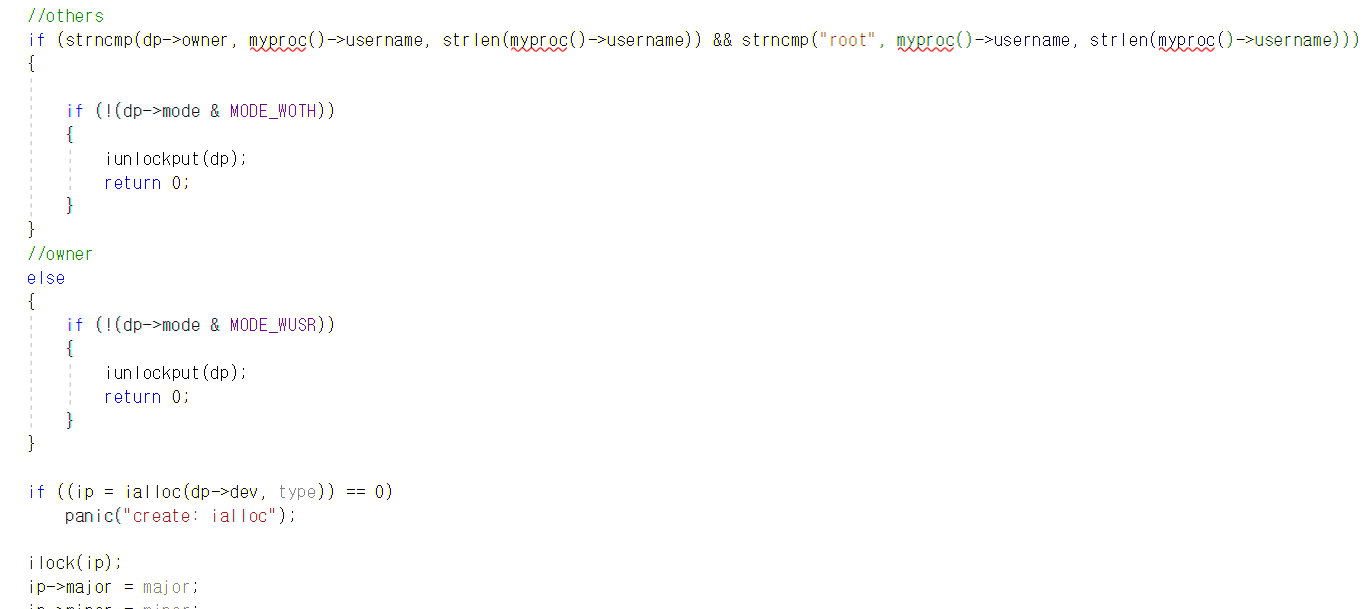
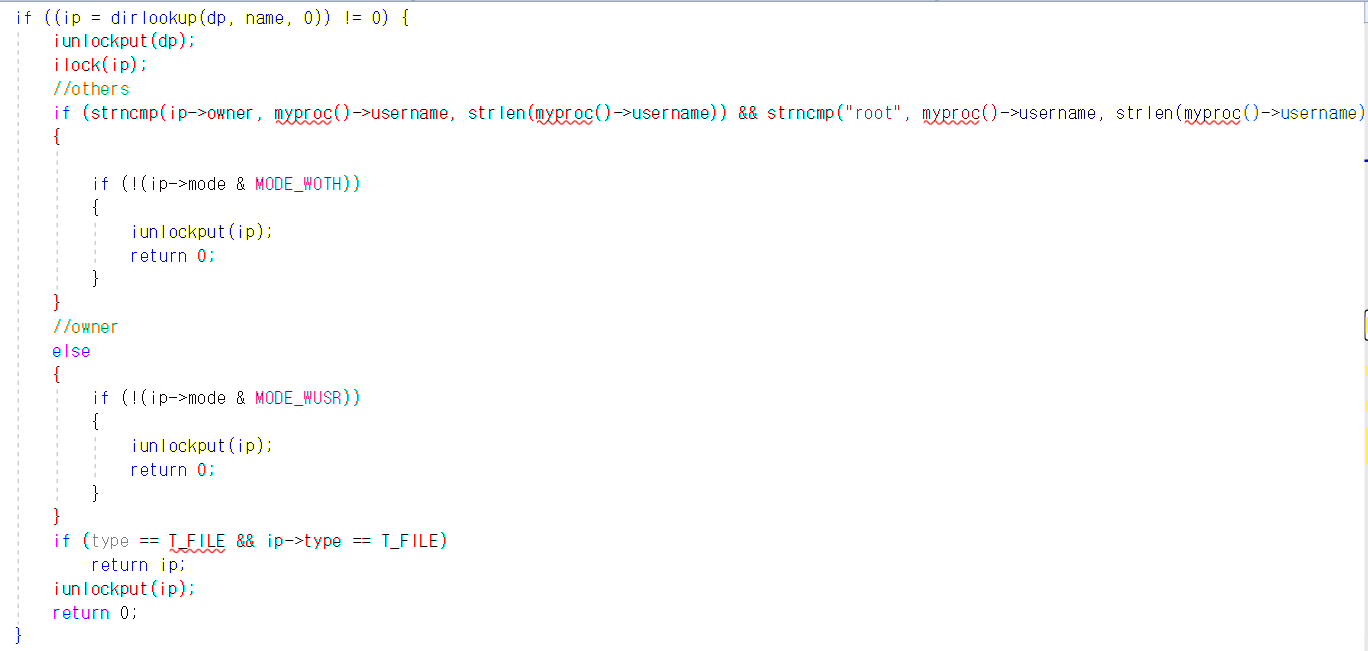


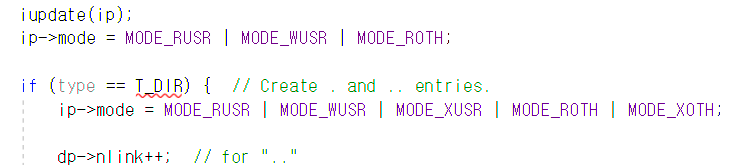
그리고 fs.c 내부에 chmod 시스템 콜 함수를 구현했습니다.

파일, 디렉토리의 owner나 root만이 권한을 변경할 수 있도록 해주었습니다. Namei를 통해 해당 inode를 읽어와서 mode를 변경한 뒤 update해주었습니다.

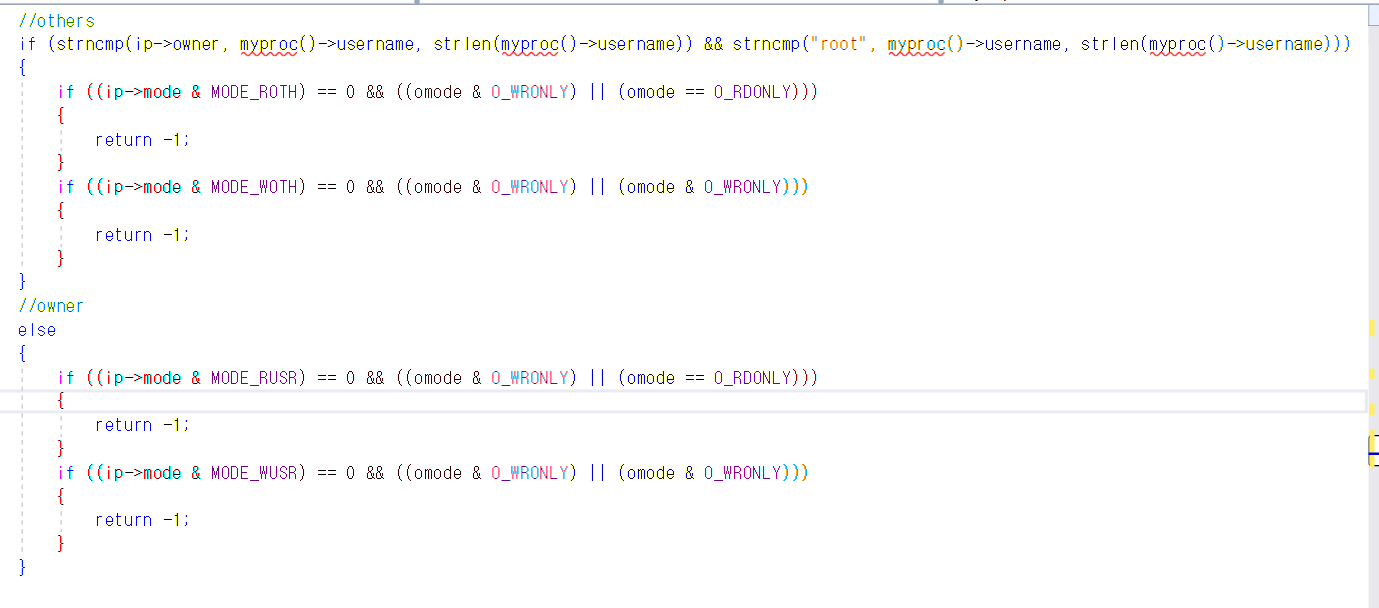


그리고 fs.c 안의 namex 함수에서 경로를 한 단계씩 따라갈 때마다 해당 directory에 실행 권한이 있는지 체크해주었습니다.





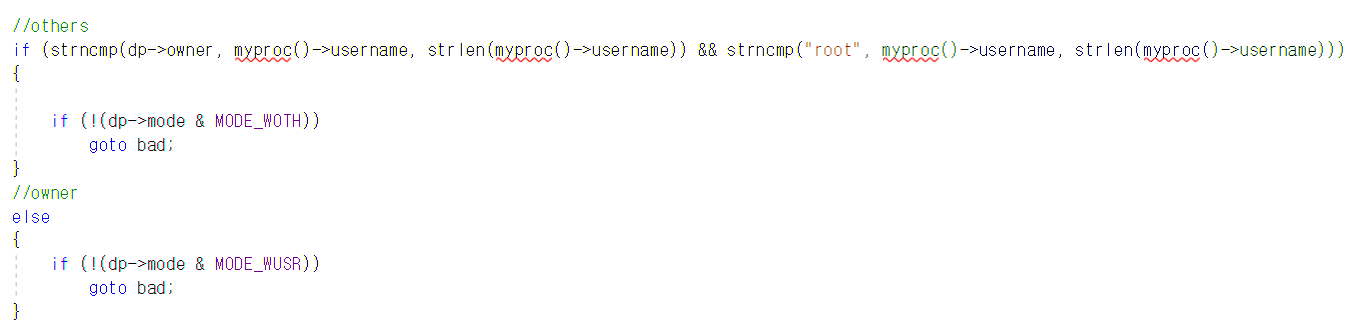
sysfile.c 안의 create 함수에서는 생성하려는 파일이 존재하는 경우, 존재하지 않는 경우에 write 권한이 있는지 확인해주었고, file을 처음 만들 때에는 MODE\_RUSR | MODE\_WUSR | MODE\_ROTH 권한을, directory를 처음 만들 때에는 MODE\_RUSR | MODE\_WUSR | MODE\_XUSR | MODE\_ROTH | MODE\_XOTH 권한을 주었습니다.



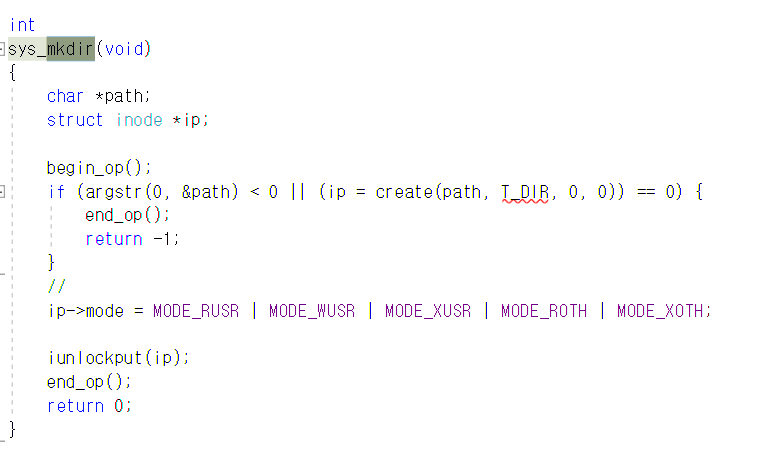
sysfile.c 안의 sys\_open 함수에서는 열기 모드가 O\_RDONLY 또는 O\_RDWR일 때 read 권한이, O\_WRONLY 또는 O\_RDWR일 때 write 권한이 있는지를 체크해주었습니다.



sysfile.c 안의 sys\_chdir 함수에서 실행 권한이 있는지를 확인해주었습니다.



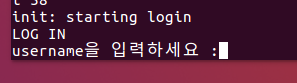
sysfile.c 안의 sys\_unlink 함수에서 삭제하려는 파일이 있는 디렉토리에 write 권한이 있는지를 확인해주었습니다.



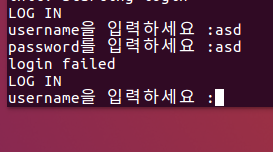
sysfile.c 안의 sys\_mkdir 함수에서는 mode를 MODE\_RUSR | MODE\_WUSR | MODE\_XUSR | MODE\_ROTH | MODE\_XOTH로 초기화해주었습니다.

2. example 코드 실행

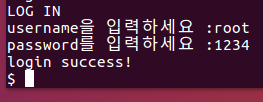
영상을 따라가며 테스트를 해 보았습니다.



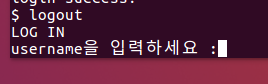
처음 프로그램을 실행했을 때, login 화면이 나옵니다.



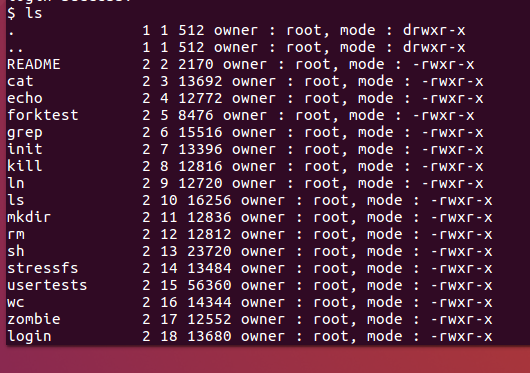
잘못된 아이디와 비밀번호를 입력했다면, login failed라는 메세지가 뜨고 다시 로그인 창이 나옵니다.



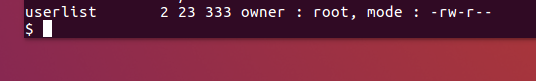
올바른 아이디와 비밀번호를 입력했다면, login success라는 메세지가 뜨고 shell이 나오게 됩니다.



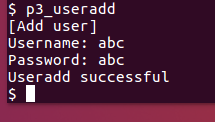
logout을 하게되면 다시 login 창이 뜨게 됩니다.



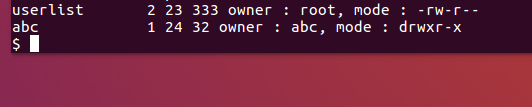
다시 로그인해서 ls를 치면 owner와 mode 정보가 추가로 출력되도록 했습니다.



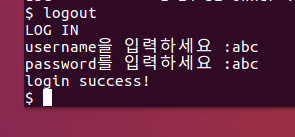
userlist에 대한 정보도 들어있습니다.



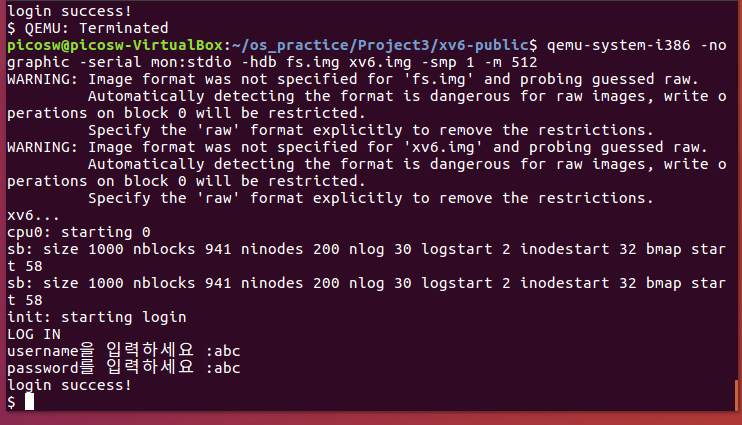
useradd를 실행하여 abc라는 유저를 추가해주었습니다.



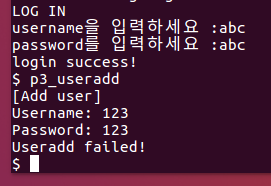
ls를 하게 된다면 그 username과 같은 이름의 directory가 생성됨을 볼 수 있습니다.



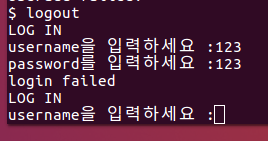
logout한 뒤 추가된 유저인 abc로 로그인하면 성공하는 것을 볼 수 있습니다.



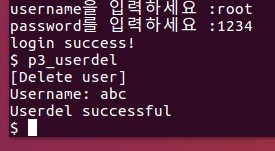
프로그램을 종료시키고 다시 실행한 뒤 abc로 로그인해도 정상적으로 로그인되는것을 볼 수 있습니다.



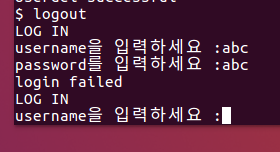
root가 아닌 사용자가 useradd 사용을 할 시 실패하는 것을 볼 수 있습니다.



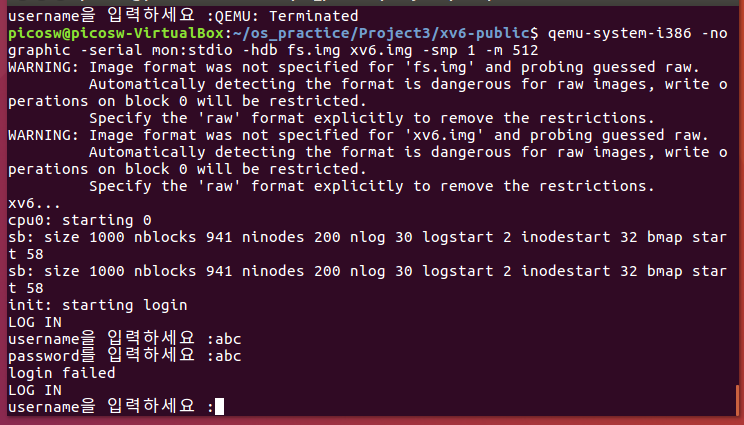
add를 실패했으니 해당 아이디로 로그인을 시도했을 때 실패하는 것을 볼 수 있습니다.



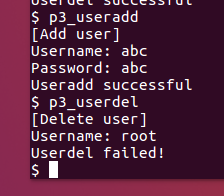
다시 root로 로그인해서 아까 생성했던 abc라는 user를 삭제했을 때 정상적으로 삭제된 것을 볼 수 있습니다.



로그아웃한 뒤 삭제된 아이디로 로그인을 시도하면 실패하는 것을 볼 수 있습니다.

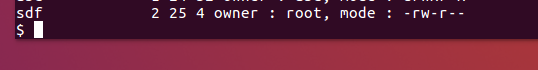


재부팅한 뒤 삭제된 아이디로 로그인을 시도해도 실패하는 것을 볼 수 있습니다.

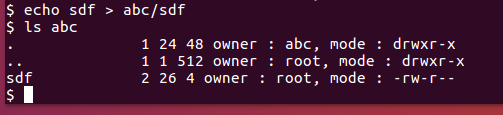


abc유저를 다시 추가하고, root를 삭제하려했지만 root에 대한 삭제는 실패하는 것을 볼 수 있습니다.

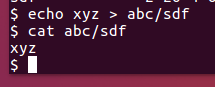




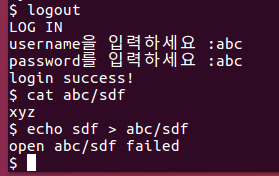
sdf라는 파일을 생성한뒤 ls를 통해 sdf라는 파일이 생성됨을 볼 수 있습니다.



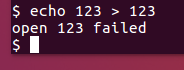
abc directory안에 sdf라는 파일을 생성한뒤 ls abc를 하면 sdf라는 파일이 생성됨을 알 수 있습니다.



읽고 쓰기 권한이 있는 owner인 root는 파일 내용을 읽고 쓰는 것을 자유롭게 할 수 있습니다.



logout하여 해당 파일에 대해 others인 abc로 로그인한다면 읽기권한은 있어서 파일을 읽을 수 있지만, 쓰기 권한이 없어서 쓰기를 시도한다면 실패하게됩니다.



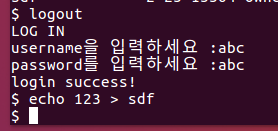
root directory에 쓰기를 시도했을 경우에는 쓰기 권한이 없으므로, 실패하게 됩니다.



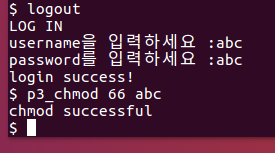
다시 root로 돌아가서 chmod를 사용해서 sdf의 mode를 66으로 바꾸어주면 성공하게 됩니다.



ls를 통해 확인해보면 mode가 rw-rw-로 바뀐 것을 확인해 볼 수 있습니다.



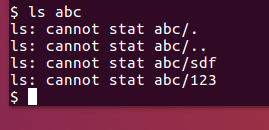
logout한 뒤 abc로 login한 후 sdf에 write를 해 주면 others에 대한 write 권한을 얻었으므로 , write가 되는 것을 확인할 수 있습니다.



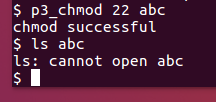
abc에서 abc폴더에 대해 owner 권한으로 chmod를 성공하는 것을 확인할 수 있습니다.



ls를 통해 abc의 모드가 바뀌었음을 알 수 있습니다.



그 후 ls abc를 했을 때 abc의 실행 권한이 없어 다음과 같은 결과가 나타남을 볼 수 있습니다.



abc에서 read 권한까지 뺀 뒤 ls abc를 실행하면 다음과 같은 결과가 나타남을 볼 수 있습니다.

이처럼 동영상에서 나와 있는 테스트들을 모두 수행해 보았습니다.