Operating Systems
Project 03

- Implementing User System in xv6 -

Due date : 11:59 pm ~~June 23~~
26일 금요일

프로젝트 목표

- xv6의 파일 시스템에는 유저의 개념이 없기 때문에, 어떤 파일이든 자유롭게 읽고, 쓰고, 그리고 실행하는 것이 가능합니다.
- 유저의 개념을 도입하고 파일들에 접근 권한을 둘 수 있도록 기능을 추가해 봅시다.

구현할 기능들

- User accounts
  - Login / Logout
  - Add user / Delete user
- File mode
  - Owner / Others
  - Read / Write / Execute
  - Change mode
- Modification of ls

주의사항

- 디바이스 (T_DEV)는 고려하지 않아도 됩니다. 단, 디바이스와 관련한 기존의 기능들은 모두 정상 동작해야 합니다.
- 이번 과제의 대략적인 구성은 리눅스에서의 동작과 유사합니다. 과제 명세에서 확실하게 이해되지 않는 부분이 있다면 리눅스 셸에서 기본 사용자와 다른 사용자 계정을 만들고, 파일의 권한을 바꾸면서 테스트해보시면 됩니다.
  - 단, 일부 사소한 차이는 있을 수 있으며, 해당 사항에 대해서는 후술하겠습니다.

User Accounts

- xv6에 하나 또는 다수의 유저가 등록될 수 있도록 기능을 추가합니다.
- 각 유저는 유저의 이름, 비밀번호, 그리고 root를 제외하고 자신의 디렉토리를 가지고 있습니다.
- 최초에는 root 유저 하나만이 존재합니다.
  - root 유저는 모든 파일에 대해 그 owner와 동일한 권한을 가집니다. (리눅스에서의 전지전능함과는 차이가 있을 수 있습니다.)
- 유저 이름 및 비밀번호는 모두 알파벳 대소문자 또는 숫자로만 이루어져 있고, 길이는 1 이상 15 이하로 가정합니다 (예외 경우를 생각하지 않아도 됩니다).
- 유저의 최대 수는 root를 포함하여 10입니다.

User Accounts

- 유저들의 이름 / 비밀번호 목록은 루트 디렉토리의 파일 하나에 저장합니다.
- 이 파일의 이름 및 구조는 자유입니다.
- 부팅 시에 이 파일을 읽어 유저 목록을 만들어야 합니다. 파일이 존재하지 않는 경우 새로 생성하고 root 계정에 대한 정보만을 저장합니다.
  - root의 비밀번호는 편의상 1234로 합니다.

Login / Logout

- 로그인을 위한 유저 프로그램을 작성합니다.
  - init 프로세스는 sh 대신 이 프로그램을 실행합니다.
  - 이 프로그램은 무한 루프를 돌며, 올바른 유저 이름 / 비밀번호가 입력되면 로그인하여 sh를 실행하고 sh가 종료될 때까지 wait 합니다.
- sh에 logout 내장 명령을 추가하여 이 명령이 입력되면 sh를 종료합니다.
  - 즉, sh가 종료되면 다시 로그인 화면으로 돌아가 다른 사용자로 로그인할 수 있어야 합니다.
- 특정 사용자로 로그인이 되었다는 것은 프로세스가 해당 유저의 권한으로 실행된다는 것을 의미합니다. 또한 그 프로세스가 fork나 exec을 사용하여 만들어내는 프로세스들 역시 같은 유저의 권한으로 실행됩니다.

Add / Delete User

- 유저를 생성하고 삭제하기 위한 다음의 두 시스템 콜을 추가합니다.
  - int useradd(char *username, char *password);
    - username의 이름과 password의 비밀번호를 가진 유저를 생성합니다.
    - 유저가 성공적으로 생성되면 0을 반환합니다.
    - 기존에 이미 존재하는 유저 이름인 경우, 더 이상 유저를 추가할 공간이 없는 경우 -1을 반환합니다.
  - int userdel(char *username);
    - username의 이름을 가진 유저를 삭제합니다.
    - root 계정은 삭제할 수 없습니다.
    - 성공적으로 삭제했다면 0을 반환합니다.
    - root 계정을 삭제하려 했거나, 없는 유저인 경우 -1을 반환합니다.
- 두 시스템 콜 모두 root로서만 호출 가능하며, root 계정이 아닌 유저가 호출한 경우 -1을 반환합니다.

Add / Delete User

- 유저가 생성되면 루트 디렉토리에 해당 유저 이름과 같은 이름의 디렉토리를 하나 생성합니다.
  - 이 디렉토리의 owner는 생성된 그 유저로 합니다.
  - 해당 이름의 디렉토리나 파일이 이미 존재하는 경우에는 아무것도 하지 않습니다.
  - 유저가 삭제될 때에는 별도의 처리를 하지 않습니다.
- 유저가 생성 / 삭제된 후에는 유저 목록을 저장하는 파일이 업데이트되어야 하며, xv6를 종료한 후 재부팅했을 때에도 반영이 되어있어야 합니다.

File Mode

- 파일 (디렉토리 포함)에 owner / others에 대한 read / write / execute 권한을 부여할 수 있게 합니다.
- Owner: 파일을 생성한 유저입니다.
- Others: Owner를 제외한 다른 유저들입니다.
- 최초의 파일 시스템에서의 모든 파일 및 디렉토리의 owner는 root로 합니다.

File Mode

- 파일 (T_FILE)의 경우 다음과 같은 사항이 적용됩니다.
  - Read: 파일을 읽을 수 있는 권한입니다. 구체적으로, read 권한이 없다면 open 시스템 콜에 O_RDONLY 또는 O_RDWR 옵션이 들어간 경우 실패해야 합니다.
  - Write: 파일을 쓸 수 있는 권한입니다. 구체적으로, write 권한이 없다면 open 시스템 콜에 O_WRONLY 또는 O_RDWR 옵션이 들어간 경우 실패해야 합니다.
  - Execute: 파일을 exec를 통해 실행할 수 있는 권한입니다. 다시 말해, execute 권한이 없는 파일을 실행하려 하면 exec이 실패해야 합니다.

File Mode

- 디렉토리 (T_DIR)의 경우 다음과 같은 사항이 적용됩니다.
  - Read 권한이 없는 경우 디렉토리 내의 파일 목록을 볼 수 없습니다. 파일의 open시 read 권한과 같은 방법으로 처리하면 됩니다.
  - Write 권한이 없는 경우 디렉토리 내의 파일을 삭제하거나, 새로운 파일을 생성할 수 없습니다. 구체적으로, create 함수에서 새로운 파일을 생성하려 하는 디렉토리에 write 권한이 없다면 생성에 실패해야 하고, sys_unlink 함수에서 파일을 삭제하려 하는 디렉토리에 write 권한이 없다면 삭제에 실패해야 합니다.
  - Execute 권한이 없는 경우 디렉토리 내의 파일에 접근할 수 없습니다. 구체적으로, namex를 통해 경로를 탐색하는 과정에 execute 권한이 없는 디렉토리가 하나라도 있다면 실패해야 합니다. 또한, execute 권한이 없는 디렉토리로 current working directory를 설정할 수도 없습니다. (이미 cwd 상태인 디렉토리에서 execute 권한이 없어지는 것은 가능합니다.)

File Mode

- open 시스템 콜을 통해 새로 만들어지는 파일 (T_FILE)은 최초에 다음과 같은 권한을 가집니다.
  - MODE_RUSR | MODE_WUSR | MODE_ROTH
- mkdir 시스템 콜을 통해 새로 만들어지는 디렉토리 (T_DIR)는 최초에 다음과 같은 권한을 가집니다.
  - MODE_RUSR | MODE_WUSR | MODE_XUSR | MODE_ROTH | MODE_XOTH
  - 최초의 파일 시스템에서의 모든 파일 및 디렉토리도 이와 같은 권한으로 합니다.

File Mode – Tricky Cases

- 파일의 경로에 접근할 때는 그 경로상에 있는 모든 디렉토리에 execute 권한이 있어야만 합니다. 즉, 같은 파일이라고 해도 사용한 경로에 따라 접근이 가능할 수도, 가능하지 않을 수도 있습니다.
  - ex) 현재 작업중인 디렉토리의 부모 디렉토리에 execute 권한이 없는 경우, 현재 디렉토리에서 직접적으로 작업하는 것은 가능하지만, 루트로부터 절대경로를 따라 내려오거나, "../현재디렉토리" 등과 같이 부모 디렉토리를 거쳐오게 할 수 없습니다.
- 리눅스에서는 파일의 권한만을 보는 것은 read와는 달라 권한에 관계 없이 볼 수 있지만, 이번 과제에서는 read 권한이 없는 경우 세부 정보를 보지 못해도 괜찮은 것으로 하겠습니다. 즉, ls를 했을 때 read 권한이 없는 파일의 stat이 실패해도 됩니다.

File Mode – Tricky Cases

- 디렉토리에 execute 권한만 없더라도 디렉토리 내의 파일을 읽거나 새 파일을 생성하는 것이 불가능할 수 있습니다. 이는 그 파일이 위치할 디렉토리를 얻어오는 과정 (namex)에 execute 권한을 요구하는 작업이 포함되어 있기 때문입니다.
- 리눅스에서는 디렉토리에 execute 권한이 없어도 cd 명령어가 . (현재 디렉토리) 및 .. (부모 디렉토리)로의 이동은 허용해 주는 것으로 보이지만, 과제에서는 이를 허용하지 않아도 괜찮습니다.

Change Mode

- 파일이나 디렉토리의 owner, 또는 root만이 파일이나 디렉토리의 권한을 변경할 수 있습니다.
- 주의: 파일의 권한에 따른 것이 아닌, 단순히 해당 파일의 소유자 여부를 통해서만 change mode를 수행할 권한이 결정됩니다.
  - 단, 그 파일의 경로를 탐색하는 과정에서 execute 권한이 없어 실패할 수는 있습니다.
- 구현할 시스템 콜
  - int chmod(char *pathname, int mode);
  - pathname은 권한을 변경할 파일의 경로입니다. 절대경로, 상대경로 모두 들어올 수 있습니다.

Change Mode

- int chmod(char *pathname, int mode);
  - mode는 다음의 값들을 비트 OR을 수행하여 만들어내는 값입니다.
    #define으로 fs.h에 선언해야 합니다.
    - #define MODE_RUSR 32 // owner read
    - #define MODE_WUSR 16 // owner write
    - #define MODE_XUSR 8 // owner execute
    - #define MODE_ROTH 4 // others read
    - #define MODE_WOTH 2 // others write
    - #define MODE_XOTH 1 // others execute
  - 해당 비트들을 제외한 나머지 비트는 무시합니다.
  - 경로가 올바르지 않거나, 수행할 권한이 없는 경우 -1을 반환합니다.
  - 올바르게 수행되었다면 0을 반환합니다.

Modification of ls

- ls 명령어가 각 파일의 권한과 소유자의 유저 이름을 출력하게 합니다.
  - 해당 정보들은 fstat 시스템 콜이 추가로 제공할 수 있도록 합니다.
- 파일의 권한은 drwxrwx 형식으로 출력합니다.
  - 가장 왼쪽은 T_FILE인 경우 -, T_DIR인 경우 d로 표시합니다.
  - 그 후 세 글자는 각각 소유자의 read, write, execute 권한, 그 후 세 글자는 다른 유저들의 권한입니다. 해당 권한이 없는 경우 그 자리를 -로 표기합니다.
  - ex) -rw-r-x: Owner가 읽고 쓸 수 있으며, others가 읽고 실행할 수 있는 파일 (T_FILE)
- 그 외의 출력 형식은 자유입니다.

Guidelines

- 파일에 대한 정보를 추가하는 것이므로 inode 및 dinode 구조가 함께 그에 맞게 변경되어야 합니다.
- xv6의 파일 시스템이 저장되는 fs.img의 처음 상태도 변경된 inode의 구조를 따라야 합니다. fs.img의 첫 구성은 mkfs.c에서 이루어지므로, mkfs.c를 일부 수정해야 합니다.
  - 주로 수정해야 할 곳은 ialloc 함수입니다. 추가된 정보들도 함께 기록될 수 있도록 바꾸어 줍니다.
- mkfs.c에서는 assert로 한 블록의 크기가 dinode의 크기의 배수일 것을 요구합니다. 따라서 dinode의 내용을 바꿀 때 전체 바이트 수가 512의 약수가 되도록 NDIRECT 등의 값을 함께 조정해야 합니다.

Guidelines

- 유저 정보를 불러오는 작업은 부팅이 모두 끝나고 최초의 유저 프로세스가 만들어진 후, 그 유저 프로세스로부터 시스템 콜을 통해 커널 모드에 진입하여 진행하는 것이 좋습니다.
  - 파일 시스템 내에서 sleeplock 등이 사용되는데, sleeplock은 현재 실행 중인 유저 프로세스에 묶이기 때문에 유저 프로세스가 없는 상태인 부팅 과정에서 사용하면 문제가 발생합니다.
- 파일 입출력과 관련된 내용은 inode, directory, pathname layer에서 수행하는 것을 추천합니다. File descriptor는 시스템 콜을 동반하는 레이어이기 때문에 커널에서 사용하기에 적합하지 않을 수 있습니다.

Guidelines

- 유저 정보를 저장하는 형식은 저장하는 배열 전체를 통째로 저장하면 편리합니다.
  - inode 레이어에서 파일 입출력을 하는 경우 편리하게 형식을 지정할 수 있는 상위 레벨의 기능들을 사용하기 어렵고, 파일의 크기를 줄이는 기능이 없기 때문에 항상 고정된 크기의 내용을 한 번에 쓰고 한 번에 읽어들이는 편이 간결합니다.
- 파일의 권한을 체크할 때 디바이스는 예외로 항상 허용하도록 처리해야 합니다. 콘솔 입출력이 동작하지 않는다면 console 디바이스에 파일 입출력을 수행하는 과정에서 문제가 발생했을 수 있습니다.

Guidelines

- 파일의 권한과 관련하여 체크해야 할 곳은 다음과 같습니다.
  - namex 함수에서 경로를 한 단계씩 따라갈 때마다 각 디렉토리에 execute 권한이 있는지 확인
  - exec 함수에서 실행하려는 파일에 execute 권한이 있는지 확인
  - create 함수에서 생성하려는 파일이 이미 존재하는 경우, 해당 파일에 write 권한이 있는지 확인
  - create 함수에서 생성하려는 파일이 존재하지 않는 경우, 그 디렉토리에 write 권한이 있는지 확인
  - sys_open 함수에서 열기 모드가 O_RDONLY 또는 O_RDWR일 때 read 권한이 있는지 확인
  - sys_open 함수에서 열기 모드가 O_WRONLY 또는 O_RDWR일 때 write 권한이 있는지 확인
  - sys_chdir 함수의 목적지에 execute 권한이 있는지 확인
  - sys_unlink 함수에서 삭제하려는 파일이 있는 디렉토리에 write 권한이 있는지 확인

Guidelines

- 파일을 다루는 것은 운영체제 상에서도 매우 민감한 부분에 속하기 때문에, inode 관련 함수들의 사용 원칙을 항상 준수할 수 있도록 주의를 기울여야 합니다.
  - 사용 전 begin_op, 사용 후 end_op
  - 어떤 함수들은 호출 시점에 ilock을 먼저 걸어놓을 것을 요구하기도 합니다.
  - iget을 호출하거나 다른 함수에 의해 간접적으로 호출된 경우 상응하는 iput이 반드시 있어야 합니다.
  - 중간에 예외 처리로 조기 종료하는 경우에도 lock을 해제하거나 end_op를 하는 등의 처리가 필요한 경우 반드시 수행되어야 합니다.
- xv6가 무한 재부팅이 되거나 panic이 발생하더라도 panic하지 마시고, 자신의 디버깅 능력을 믿고 문제의 발생 지점을 찾아 원인을 분석해 봅시다.

평가

- **Completeness**: 명세의 요구조건에 맞게 xv6가 올바르게 동작해야 합니다.
- **Defensiveness**: 발생할 수 있는 예외 상황에 대처할 수 있어야 합니다.
- **Wiki&Comment**: 테스트 프로그램과 위키를 기준으로 채점이 진행되므로 위키는 최대한 상세히 작성되어야 합니다.
- **Deadline**: 데드라인을 반드시 지켜야 하며, 데드라인 이전 마지막으로 commit/push된 코드를 기준으로 채점됩니다.
- **Do NOT copy or share your code.**

평가지표

평가 기준은 다음과 같으며, 각 명세의 일부분을 완성하였다면 부분점수가 주어집니다.
실패하는 테스트가 있을 시 해당 부분에서 감점이 이루어집니다.

평가 기준	배점	비고
Login / Logout	15	대략적인 배점이며, 실제 채점 시에는 경계를 명확하게 구분 짓기 어려운 것들이 많아 세부 테스트 단위로의 배점이 부여될 예정입니다.
User add / User delete	15	
File mode	25	
Change mode	15	
Modification of ls	10	
Wiki	20	
총점	100	

제출

- 제출은 Hanyang gitlab을 통해 이루어져야 합니다.
- 제출된 repository는 반드시 그림의 디렉토리 구조를 가져야 합니다. (os_practice가 repo폴더)
- xv6-public폴더의 이름이 지켜지면 되며, 파일이나 폴더가 추가되어도 상관없습니다.
- 채점은 과제 마감 시간 전에 마지막으로 commit 및 push된 버전을 기준으로 합니다. 만일을 위해 commit 내역을 삭제하지 않도록 해주시기 바랍니다.