**Pintos Project 2: User Program (2)**

담당 교수 / 분반 : 박성용 / 홀수분반

이름 / 학번 : 강민석 / 20161559

개발 기간 : 2020.11.09 ~ 2020.11.12

1. **개발 목표**

file system과 관련된 system call을 구현한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

1. File Descriptor

pintos에서는 UNIX와 유사하게 file descriptor를 통해 파일에 접근한다. 이로 인해 process에서 각종 파일에 접근할 때 fd를 이용하면 쉽게 해당 파일에 접근할 수 있다.

2. (이번 프로젝트에서 구현해야 하는) System Calls

create, remove, open, close, filesize, read, write, seek, tell system call을 구현해야 한다.

3. Synchronization in Filesystem

파일에 접근할 때, 하나의 파일에 대해 읽기/쓰기 혹은 쓰기/쓰기가 동시에 수행되면 안된다. 이를 위해 적절하게 synchronization을 구현하여 예기치 못한 결과를 막아준다.

* 1. **개발 내용**

1. File Descriptor: process당 여러 개의 file descriptor를 지닐 수 있으므로 thread 구조체 내부에 struct file\* 배열을 추가선언한다. 배열로 선언하는 이유는 이후 메모리 deallocate문제를 고려할 필요가 없어 안전하다고 판단하였기 때문이다.

2. System Calls: 구현해야 할 system call은 아래와 같다.

create : 파일을 생성한다. 정상적으로 생성되었으면 true를 반환하고 그렇지 않은 경우 false를 반환한다.

remove : 파일을 삭제한다. 정상적으로 삭제되었다면 true를 반환하고 그렇지 않은 경우 false를 반환한다.

open : 파일을 연다. 정상적으로 열렸다면 fd(file descriptor)를 반환하고 그렇지 않은 경우 -1을 반환한다.

filesize : 파일의 크기(byte)를 반환한다.

read : fd에 대응되는 파일을 읽는다. 실제로 읽은 크기(byte)를 반환하며 만약 읽지 못한 경우 -1을 반환한다.

write : fd에 대응되는 파일에 쓴다. 실제로 쓴 크기(byte)를 반환하며 만약 쓰지 못한 경우 -1을 반환한다.

seek : fd에 대응되는 파일에서 position을 변경한다.

tell : fd에 대응되는 파일에서 다음 읽거나 쓰여야할 byte의 현재 위치를 반환한다.

close : fd에 대응되는 파일을 닫는다.

3. Synchronization in Filesystem: 기본적으로 강의자료에 나와있는 first readers-writes problem을 해결하는 방식의 형태를 취한다. Lock 변수 wrt, Semaphore 변수 mutex를 userprog/syscall.c 파일에 전역변수로 설정하고, read/write system call 함수 내부에서 강의자료의 내용과 똑같이 구현한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

2020. 11. 09 : file descriptor 구현

2020. 11. 10 : file system call 구현

2020. 11. 11 : synchronization 구현

2020. 11. 12 : 코드 최적화 및 test case 통과를 위한 코드 수정

* 1. **개발 방법**

1. File Descriptor

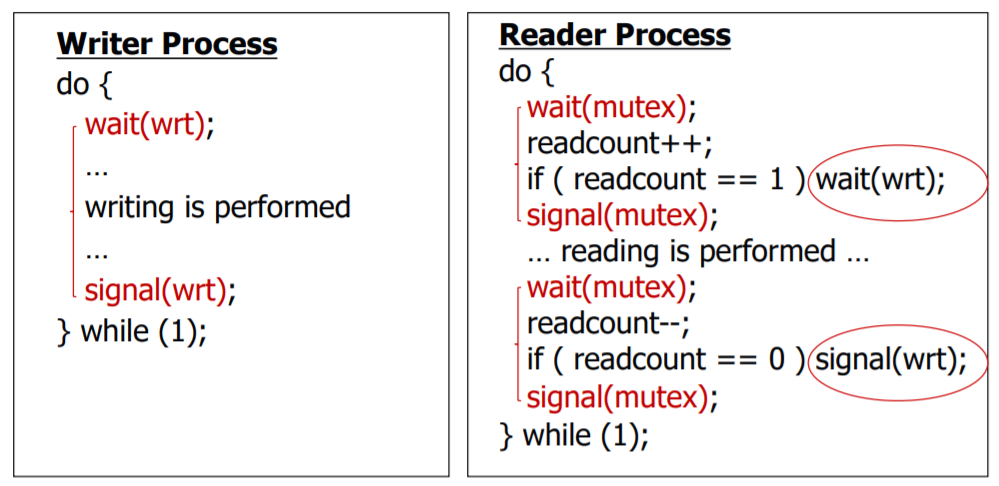
thread 구조체에 struct file\*의 배열형태를 추가한다. 이를 위해 userprog/thread에 있는 thread.h 파일을 수정해야한다.

2. System Calls

모든 file system call은 userprog/syscall.c에 각 syscall number에 대한 처리, 함수 호출 등을 구현한다. 이번 프로젝트에서 system call 구현은 모두 filesys 디렉토리에 있는 file.h 혹은 filesys.h에 선언된 함수를 호출한다. 따라서 userprog/syscall.c에 함수를 구현하되, filesys 디렉토리에 있는 함수를 호출하기 위해서 syscall.c 파일에 #include “filesys/filesys.h”, #include “filesys/file.h”를 추가해준다.

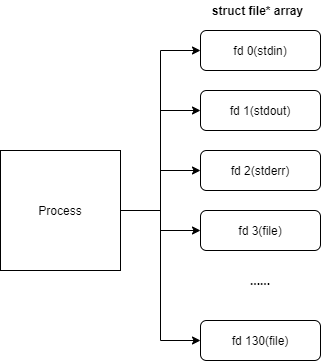
3. Synchronization in Filesystem

Synchronization을 구현하기 위해 userprog/syscall.c 파일에 1개의 semaphore, 1개의 lock, 1개의 integer 변수(readcount)를 전역 변수로 선언해준다. first readers-writers를 구현하기 위해 필요한 장치이며 아래 내용(강의자료)와 똑같이 구현해주었다. mutex를 semaphore로, wrt를 lock으로 선언하였다.



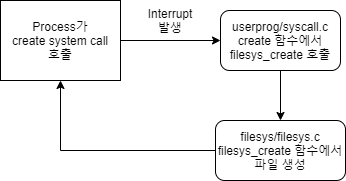
1. **연구 결과**
   1. **Flow Chart**

**1. file descriptor**

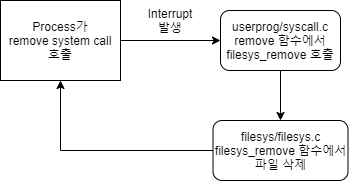
****

**2. System Calls**

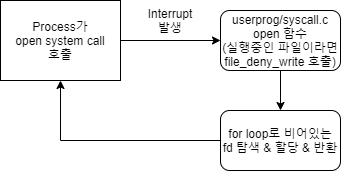
**- create**

****

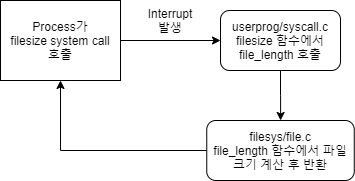
**- remove**

****

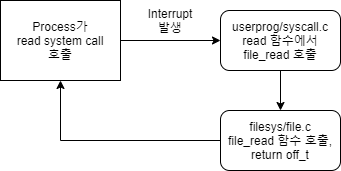
**- open**

****

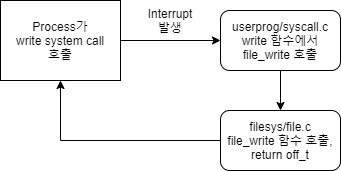
**- filesize**

****

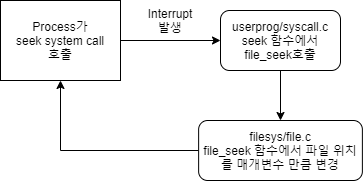
**- read**

****

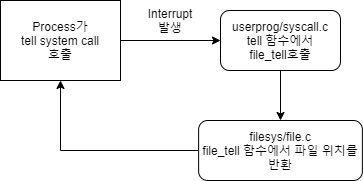
**- write**

****

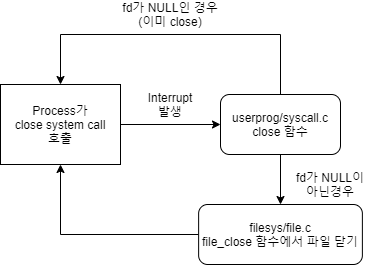
**- seek**

****

**- tell**

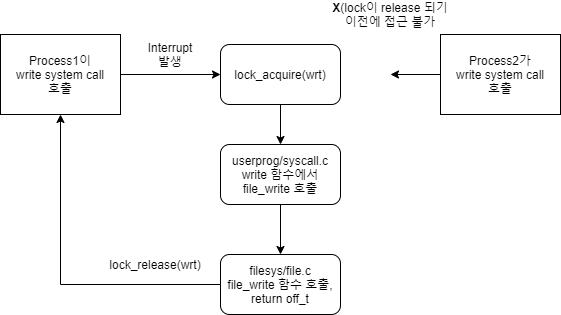
****

**- close**

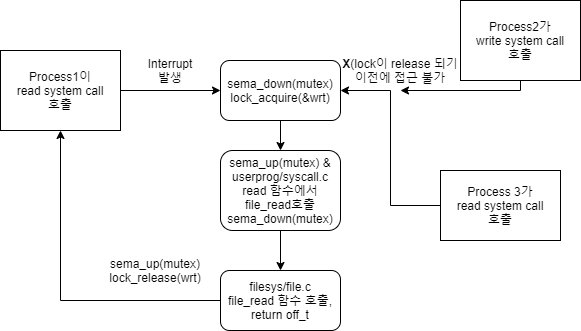
****

**3. Synchronization in Filesystem**

**- write**

****

**- read**

****

* 1. **제작 내용**

**1. File Descriptor**

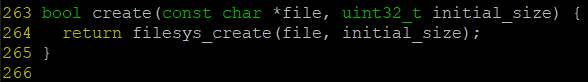
**C:\Users\seven\Desktop\2.PNG**

(threads/thread.h struct thread 내부)

우선 위와 같이 thread.h의 thread 구조체 내부에 파일 포인터 배열을 131개 선언하였다. 여기서 131개를 선언한 이유는 다음과 같다. 우선 fd 0, 1, 2는 stdin, stdout, stderr 등 그 용도가 정해져있다. 그리고 pintos manual에서는 제한이 없도록 구현하는 것이 좋지만 하나의 process 당 128개의 파일을 열 수 있도록 설정하면 된다라고 명시되어있다. struct file은 filesys/file.c에 정의되어 있는 파일 구조체로, struct inode, off\_t pos, bool deny\_write 변수를 포함하는 구조체이다.

2. System calls

- create



(userprog/syscall.c)

syscall.c 내부에 create 함수를 위와 같이 정의한다. 생성할 file 이름과 초기 크기는 매개변수와 같으며, 해당 매개변수를 filesys\_create의 argument로 넘겨주면서 함수를 호출한다. filesys\_create는 filesys/filesys.h에 정의되어 있는 함수다.

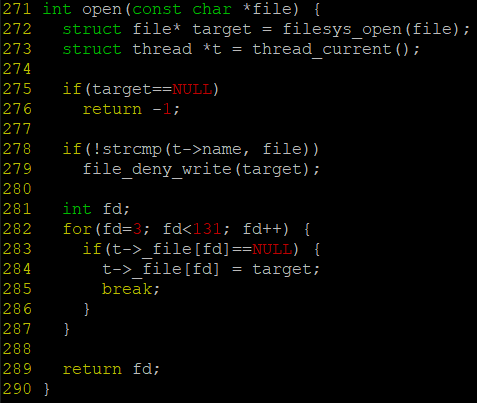
- remove

C:\Users\seven\Desktop\4.PNG

(userprog/syscall.c)

syscall.c 내부에 remove 함수를 위와 같이 정의한다. 삭제할 file 이름은 매개변수와 같으며, 해당 매개변수를 filesys\_remove의 argument로 넘겨주면서 함수를 호출한다. filesys\_remove는 filesys/filesys.h에 정의되어 있는 함수다.

- open



(userprog/syscall.c)

매개변수 file이름을 갖는 파일을 연다. 우선 filesys/filesys.h에 정의되어 있는 filesys\_open함수를 이용하여 파일을 연다. 만약 열기에 실패한다면(275줄) -1을 return하여 실패하였음을 알린다. pintos에서는 executable한 파일에 대하여 수정하는 것을 방지하게끔 구현해야 하므로, 278~279줄에서 실행중인 thread의 이름과 열려는 파일의 이름이 동일한 경우 filesys/file.h에 정의되어 있는 file\_deny\_write함수를 호출하여 write가 수행되지 못하도록 막아준다. 282 ~ 287의 코드는 process에서 비어있는 fd를 탐색한 후, fd를 할당하는 작업을 반복문으로 구현한 것이다.

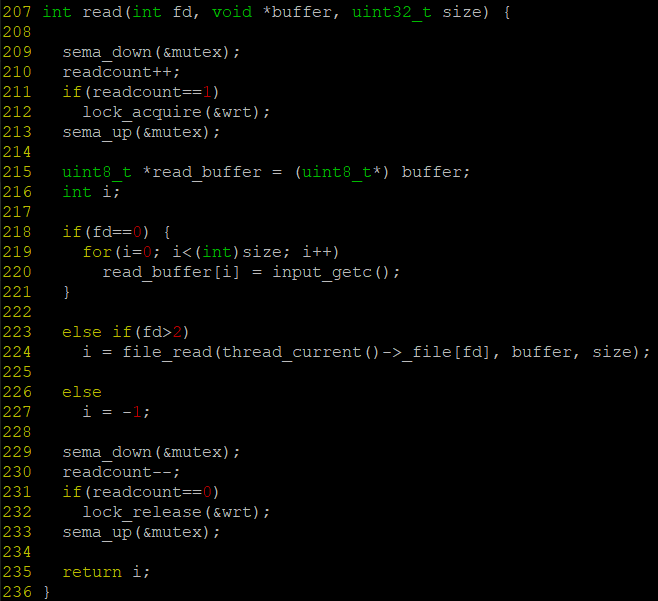
- filesize

C:\Users\seven\Desktop\6.PNG

(userprog/syscall.c)

filesys/file.h에 정의되어 있는 file\_length 함수를 호출한다. 대상 파일은 현재 process의 매개변수 fd에 대응되는 파일이다.

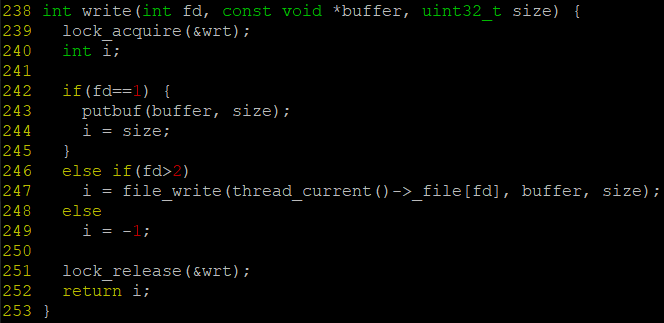
- read



(userprog/syscall.c)

file에 대한 read를 수행할 때는 filesys/file.h에 정의되어 있는 file\_read를 호출하였고(224줄), stdin 같은 경우 input\_getc()함수를 호출(220줄)하도록 하였다. (synchronization 부분은 따로 기술) 만약에 비정상적인 fd가 들어올 경우 -1을 반환함으로써 오류를 알려주었다.

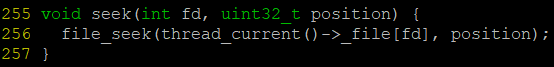
- write



(userprog/syscall.c)

file에 대한 write를 수행할 때는 filesys/file.h에 정의되어 있는 file\_write를 호출하였고(247줄), stdout 같은 경우 putbuf 함수를 호출(243줄)하도록 하였다. 비정상적인 fd가 들어올 경우 -1을 반환함으로써 오류를 알려주었다.

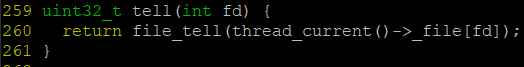
- seek



(userprog/syscall.c)

filesys/file.h에 정의되어 있는 file\_seek 함수를 호출한다. 대상 파일은 현재 process의 매개변수 fd에 대응되는 파일이며, position으로 위치를 변경한다.

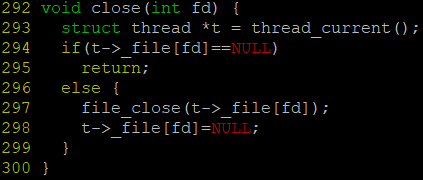
- tell.



(userprog/syscall.c)

filesys/file.h에 정의되어 있는 file\_tell 함수를 호출한다. 대상 파일은 현재 process의 매개변수 fd에 대응되는 파일이며, 반환 값으로 위치를 반환한다.

- close



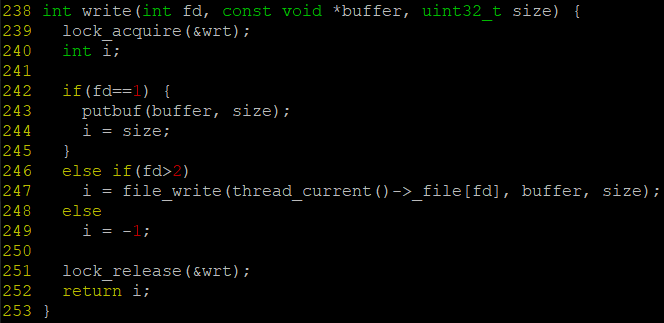
(userprog/syscall.c)

filesys/file.h에 정의되어 있는 file\_close 함수를 호출한다. 대상 파일은 현재 proess의 매개변수 fd에 대응되는 파일이다. 이 때, 이미 닫혀있는 경우 fd에 대응되는 struct file\*이 NULL이므로 이런경우 별도로 close 하지 않고 return(294줄) 한다. 그렇지 않은 경우 filesys/file.h에 정의되어 있는 file\_close함수를 호출하고, 해당 fd에 대응되는 struct file\*을 NULL로 변경해준다.

3. Synchronization in Filesystem

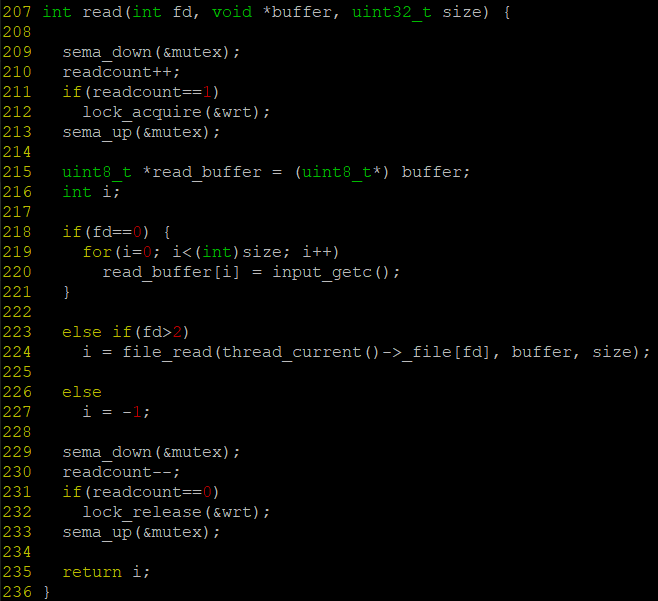
이번 프로젝트를 진행하면서 read/write에 대한 synchronization을 구현하였다.

- write



우선 write에서는 struct lock wrt를 선언하고, write 함수가 호출되면서 239줄에서 lock\_acquire를 통해 하나의 write가 수행될 때, 다른 write는 수행될 수 없도록 구현하였다. 모든 write가 수행되고 나면 lock\_release(251줄)를 통해 다른 write가 진입할 수 있도록 구현하였다. (write-write 문제 해결)

- read



read의 경우 추가로 struct semaphore mutex와 readcount 변수를 선언하였다. read의 경우 read-write, write-read, read-read 와 같이 수행되는 경우가 있다. 하나하나 살펴보면 다음과 같다.

(1) read-write

우선 read가 수행될 경우 210 줄에서 readcount가 1로 변경된 후, 212 줄에서 lock\_acquire(&wrt)를 통해 wrt에 lock이 걸린다. 따라서 write가 진입할 수 없게된다. write는 232줄에서 lock\_release가 수행된 후(CS가 끝난 후) 수행된다.

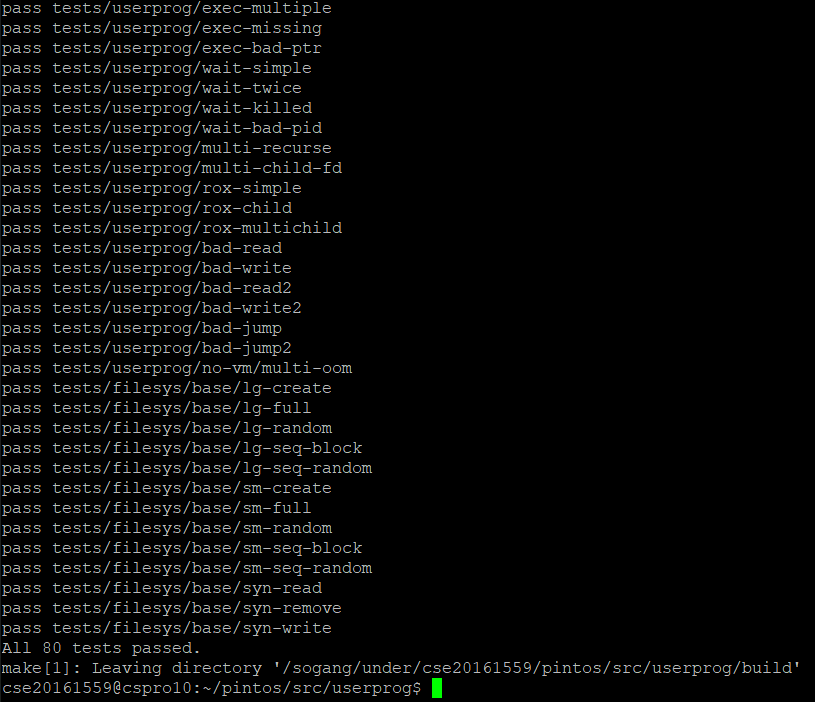
(2) write-read

우선 write가 수행될 경우 lock은 걸려있는 상태이므로 첫 번째 read가 수행될 경우 212줄에서 lock에 걸려 대기하며, 두 번째 이상의 read가 수행될 경우 첫 번째 read의 semaphore(209줄)에 걸려 대기상태로 들어간다.

(3) read-read

read-read의 경우 별도로 파일수정이 이뤄지지 않으므로 synchronization이 따로 필요없다. 불필요하게 lock이 걸리는 것을 방지하기 위해 readcount를 이용하여 여러 개의 read가 동시에 수행될 수 있도록 구현하였다.

* 1. **시험 및 평가 내용**

****