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

담당 교수 / 분반 : 문의현 교수님 / 2분반

이름 / 학번 : 김기철 / 20181255

개발 기간 : 2022.11.7~2022.11.14

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술

저번 프로젝트에 이어 filesystem 관련 시스템 콜 함수를 추가한다. 추가 구현할 함수로는 create, remove, open, close, filesize, seek, tell이 존재하고 file 관련 시스템 콜 함수인 read와 write 함수도 알맞게 수정한다. 이 과정에서 file descriptor와 synchronization 과정에서 문제가 발생하지 않도록 필요한 함수들을 알맞게 구현한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* 아래 각 항목을 구현해야 하는 이유, 혹은 구현 시 기대되는 결과를 간략히 서술

1. File Descriptor

프로세스가 파일에 접근하려면 fd(file descriptor) 값을 이용해야 하므로, file descriptor를 적절한 자료구조로 선언하고 프로세스가 알맞게 접근할 수 있도록 구현한다.

2. (이번 프로젝트에서 구현해야 하는) System Calls

파일 관련 시스템 콜인 create, remove, open, close, filesize, seek, tell을 추가 구현하여 파일이 올바르게 생성, 삭제 등의 기능을 하고, 표준입력이 아닌 파일이 입력되었을 때를 고려하여 저번 주차에 구현하였던 read, write 함수를 알맞게 수정하여 파일을 올바르게 읽고 쓰도록 한다.

3. Synchronization in Filesystem

동기화 과정에서 critical section에 race condition 등의 문제가 발생하지 않도록 lock 구조체와 함수를 이용하여 동기화가 올바르게 이루어지도록 구현한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. File Descriptor: 구현에 이용할 자료구조와 선택한 이유를 서술

프로세스가 파일에 접근하기 위해 만든 file descriptor 구현에 배열 자료구조를 이용하였다. thread 구조체 내에 프로세스가 접근할 수 있는 fd 배열을 선언하였고, 배열 크기는 메뉴얼에 따라 128개로 설정했다.

2. System Calls: 구현할 각 system call에 대해 간략히 서술 (하나의 system call 당 최대 3문장으로 간략히 설명; 3문장을 넘길 정도로 길게 작성하지 말 것)

1) create

파일 이름과 size를 입력받으면 filesys\_create 함수를 이용하여 size만큼의 새로운 파일을 생성한다. 성공적으로 파일이 생성되었다면 true를 반환한다.

2) remove

파일 이름을 입력받으면 filesys\_remove 함수를 이용하여 해당 파일을 삭제한다. 성공적으로 삭제하였다면 true를 반환한다.

3) open

파일 이름을 입력받으면 filesys\_open 함수를 이용하여 해당 파일을 open한다. 성공적으로 open하였다면 해당 파일의 fd값을 반환한다.

4) close

파일 이름을 입력받으면 file\_close 함수를 이용하여 해당 파일을 close한다. 반환값은 없다.

5) filesize

파일 이름을 입력받으면 file\_length 함수를 이용하여 해당 파일의 size를 반환한다.

6) seek

파일 이름과 위치 정보를 입력받으면 file\_seek 함수를 이용하여 해당 파일을 입력받은 위치로 이동시킨다.

7) tell

파일 이름을 입력받으면 file\_tell 함수를 이용하여 해당 파일의 위치를 반환한다.

8) read

표준입력이 아닌 파일이 입력된 경우를 추가 구현한다. 파일 이름과 buffer, size를 입력받으면 file\_read 함수를 이용하여 해당 파일을 size만큼을 read하여 buffer에 저장한다.

9) write

표준입력이 아닌 파일이 입력된 경우를 추가 구현한다. 파일 이름과 buffer, size를 입력받으면 file\_read 함수를 이용하여 buffer에서 size만큼을 해당파일에 wirte한다.

3. Synchronization in Filesystem: Lock, Semaphore를 어떻게 이용할 수 있는지 각각에 대해 설명 (다른 방법을 서술해도 되지만 lock과 semaphore는 반드시 포함해야 함)

1) lock

race condition 등의 문제점이 발생하지 않도록 lock 구조체를 선언하고 lock\_acquire, lock\_release 함수를 이용한다. 위에서 추가로 구현한 파일 관련 시스템 콜 함수들을 호출할 때, 작업이 수행되기 전에 lock\_acquire 함수를 호출하고, 작업이 마쳤다면 이후에lock\_release 함수를 호출하는 방식이다.

2) semaphore

thread 구조체에 file\_semaphore 변수를 선언하고 sema\_up, sema\_down 함수를 이용한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

**11/7~11/9 : file descriptor 및 시스템 콜 함수 구현**

**11/9~11/12 : synchronization 구현 및 테스트 케이스 확인**

**11/13~11/14 : 코드 최종 마무리 및 보고서 작성**

* 1. **개발 방법**
* II. B.의 개발 내용을 구현하기 위해 각각에 대해 다음 사항들을 포함하여 설명
  + 수정해야하는 소스코드
  + 수정하거나 추가해야 하는 자료구조
  + 수정하거나 추가해야 하는 함수

1. File Descriptor

thread.h 내에 fd값을 저장해줄 128크기의 배열을 선언하고 thread.c에서 초기화를 해준다.

2. System Calls

userprog/syscall.c의 syscall\_handler 함수내에 file관련 시스템 콜 명령어들에 따른 함수들을 추가 구현한다. 추가로 구현할 함수들을 userprog/syscall.h에서 선언을 해주고, 각각의 기능에 알맞게 시스템 콜이 작동되도록 구현한다.

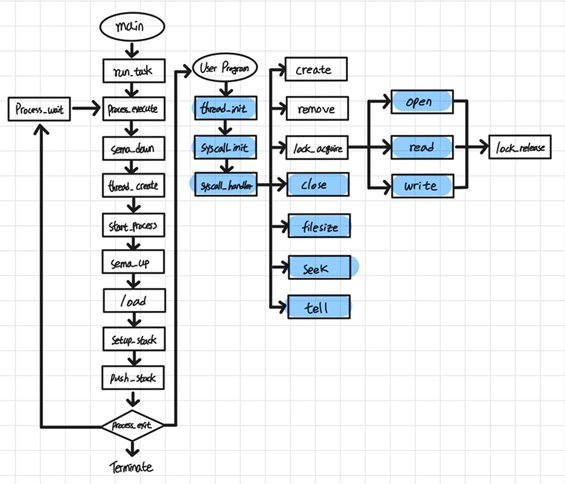
3. Synchronization in Filesystem

thread.h 내에 lock\_acquire과 lock\_release 함수를 실행하기 위한 lock 구조체 변수를 선언하고 thread.c 에서 lock\_init 함수를 통해 초기화를 해준다. 또한, thread 구조체 내에 동기화를 위한 구조체 변수들을 추가한다. 추가할 변수로는 프로세스의 상태를 위한 file\_semaphore 변수, 프로세스가 비정상적으로 종료되는지 확인하는 flag 변수, 현재 파일을 나타내는 curfile 변수, 현재 프로세스의 부모 프로세스를 나타내는 parent thread 변수가 있다. 이렇게 선언한 변수들을 모두 thread.c의 init\_thread 함수에서 초기화 해준다. 또한, exception.c의 page\_fault 함수 내에 not\_present 변수가 1일 때 프로그램을 종료시켜주는 조건을 추가한다. 2번에서 추가로 구현한 시스템 콜 함수 중, open, read, write 명령을 실행하기 전과 후에 lock\_acquire과 lock\_release 함수를 넣어 race conditon 등의 문제가 발생하지 않도록 해주고, process.c의 process\_execute, start\_process, load 함수에 공유 데이터를 위해 sema\_up, sema\_down 함수를 이용하여 file\_semaphore 변수를 처리해준다.

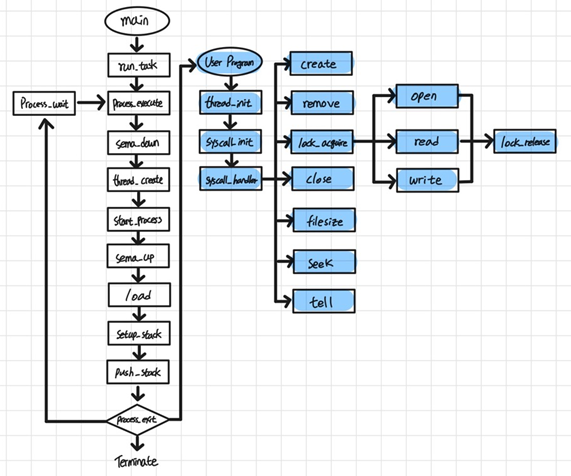
1. **연구 결과**
   1. **Flow Chart**

* II. B. 개발 내용의 각 3가지 항목에 대하여 Flow Chart 작성

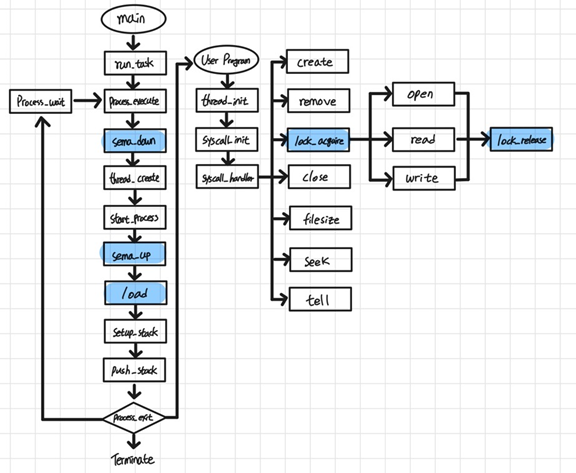
1. File Descriptor



2. System Calls



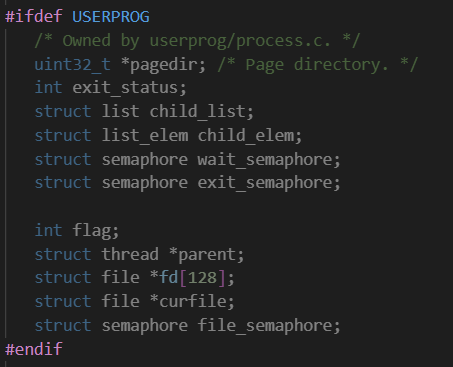
3. Synchronization in Filesystem



* 1. **제작 내용**
* II. B. 개발 내용의 각 3가지 항목에 대하여 실직적으로 구현한 코드의 관점에서 작성 (구현 내용, 알고리즘 등을 명확히 서술할 것)
* 구현에 있어 Pintos에 내장된 라이브러리나 자체 제작한 함수를 사용한 경우 이에 대해서도 설명
* 개발 중 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결한 방식에 대해 설명

1. File Descriptor

**1) thread.h**



thread 구조체 내에 fd값을 저장하는 배열 fd[128]을 선언한다.

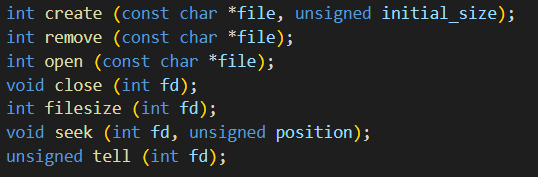
**2) thread.c**



thread 구조체에 선언한 fd 배열을 thread.c의 init\_thread에서 초기화한다.

2. System Calls

**1) uesrprog/syscall.h**

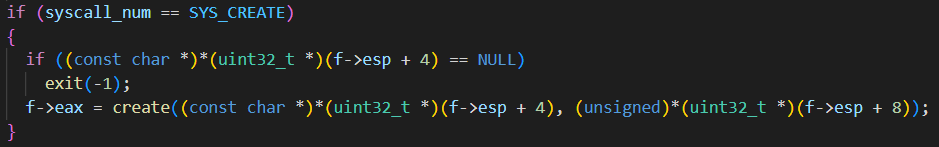


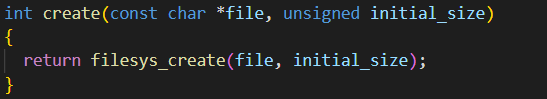
새로 구현할 시스템 콜 함수들을 syscall.h에 선언한다.

**2) userprog/syscall.c**

새로 선언한 시스템 콜 함수들을 syscall.c의 syscall\_handler 함수에 추가한다.

**(1) create**

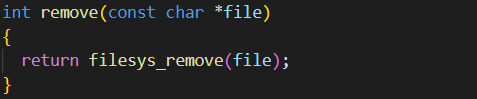




파일 이름과 size를 입력받으면 filesys\_create 함수를 이용하여 size만큼의 새로운 파일을 생성한다. 성공적으로 파일이 생성되었다면 true를 반환하여 eax에 넣어준다.

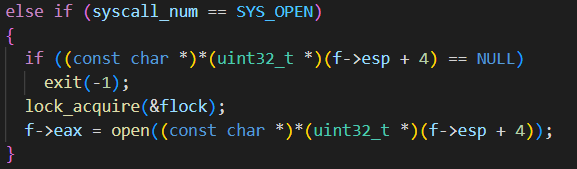
**(2) remove**

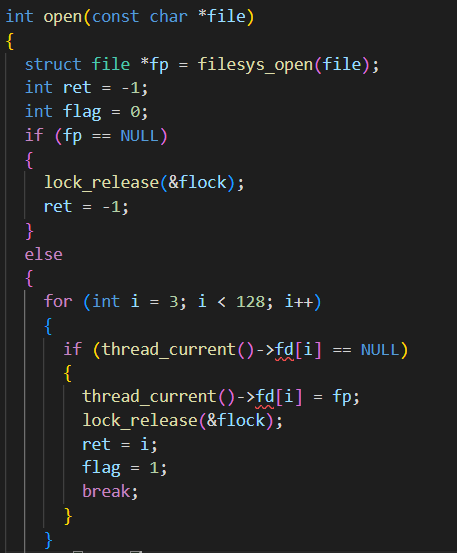


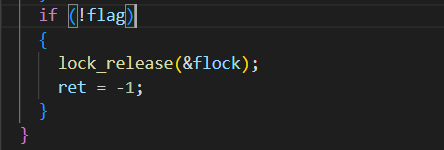


파일 이름을 입력받으면 filesys\_remove 함수를 이용하여 해당 파일을 삭제한다. 성공적으로 삭제하였다면 true를 반환하여 eax에 넣어준다.

**(3) open**

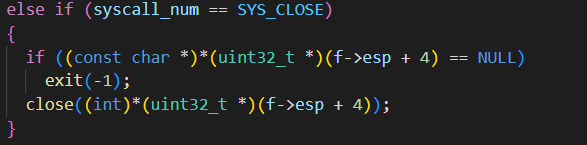


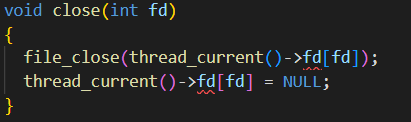




파일 이름을 입력받으면 filesys\_open 함수를 이용하여 해당 파일을 open한다. 성공적으로 open하였다면 해당 파일의 fd값을 반환하여 eax에 넣어준다.

**(4) close**

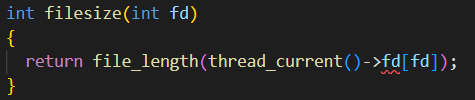




파일 이름을 입력받으면 file\_close 함수를 이용하여 해당 파일을 close한다. 반환값은 없다.

**(5) filesize**

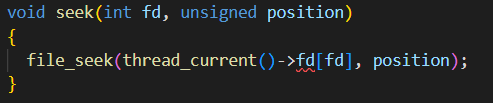




파일 이름을 입력받으면 file\_length 함수를 이용하여 해당 파일의 size를 반환하여 eax에 넣어준다.

**(6) seek**

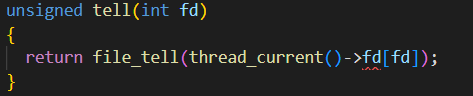
****



파일 이름과 위치 정보를 입력받으면 file\_seek 함수를 이용하여 해당 파일을 입력받은 위치로 이동시킨다. 반환값은 없다.

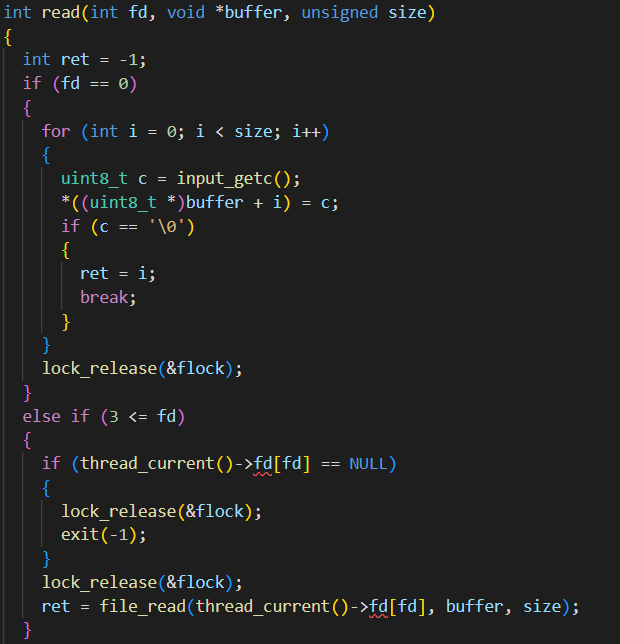
**(7) tell**

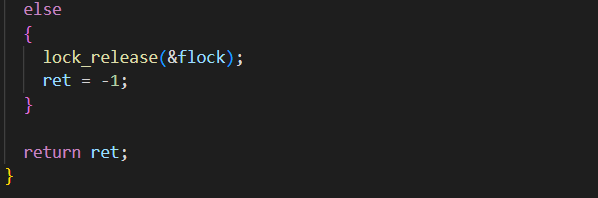
****



파일 이름을 입력받으면 file\_tell 함수를 이용하여 해당 파일의 위치를 반환하여 eax에 넣어준다.

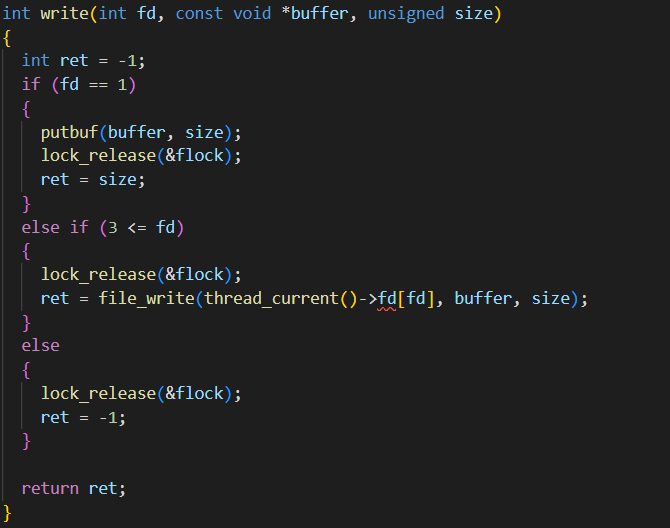
**(8) read**





표준입력이 아닌 파일이 입력된 경우를 추가 구현한다. 파일 이름과 buffer, size를 입력받으면 file\_read 함수를 이용하여 해당 파일을 size만큼을 read하여 buffer에 저장한다.

**9) write**



표준입력이 아닌 파일이 입력된 경우를 추가 구현한다. 파일 이름과 buffer, size를 입력받으면 file\_read 함수를 이용하여 buffer에서 size만큼을 해당파일에 wirte한다.

3. Synchronization in Filesystem

**1) thread.h**



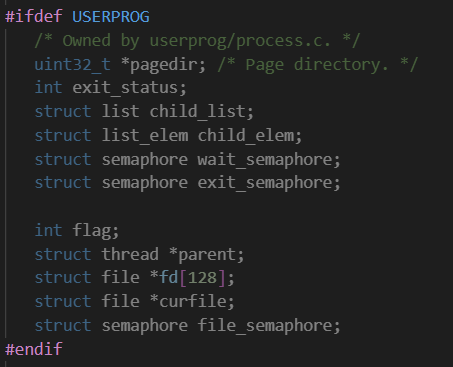
lock\_acquire, lock\_release 함수를 실행하기 위한 lock 구조체 변수 flock을 선언한다.

**2) thread.c**

****

선언한 구조체 변수를 thread.c의 thread\_init 함수에서 초기화해준다.

**3) thread.h**



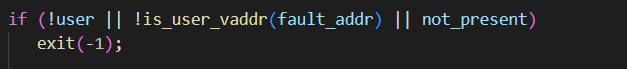
thread 구조체 내에 synchronization을 위한 구조체 변수를 선언한다.

**4) thread.c**



thread 구조체에 선언한 변수들을 thread.c의 init\_thread에서 초기화한다.

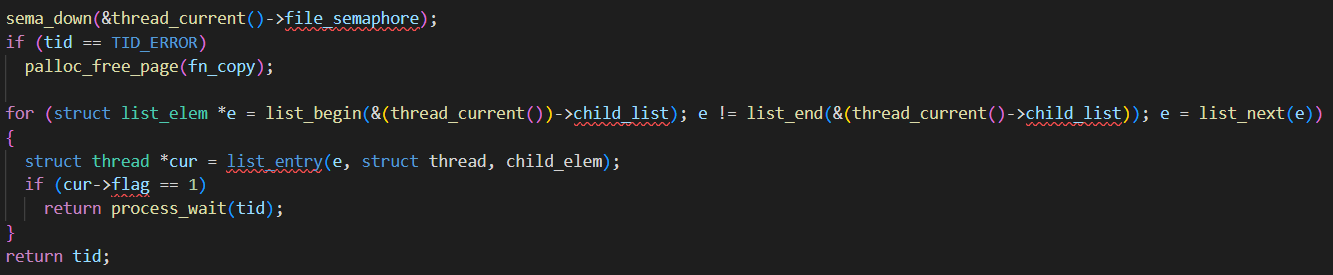
**5) exception.c**

****

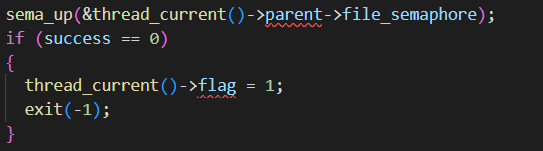
exception.c의 page\_fault 함수 내에 not\_present 변수가 1일 때 프로그램을 종료시켜주는 조건을 추가한다.

**6) process.c**

**(1) process\_execute**

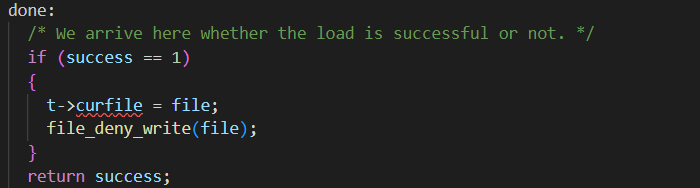
\process.c의 process\_execute 함수 내에 위와 같은 부분을 추가한다. 공유데이터 사용을 위해 sema\_down 함수를 이용하여 file\_semaphore을 처리해주고, 프로세스가 비정상적으로 종료되는 것을 막기 위해 flag가 1이라면 process\_wait를 실행시켜준다.

**(2) start\_process**



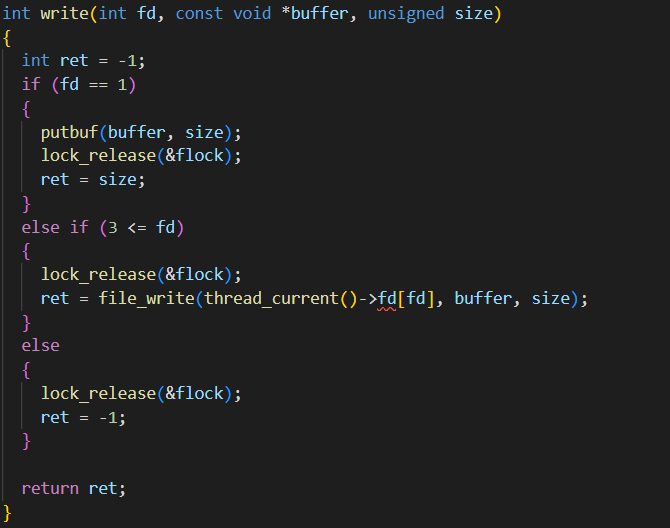
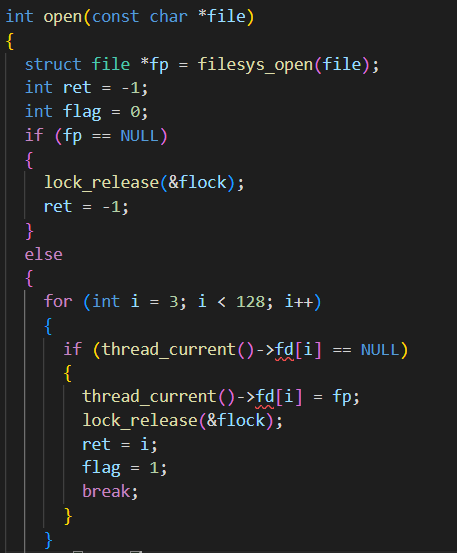
process.c의 start\_process 함수 내에서 위와 같은 부분을 추가한다. 공유데이터 사용을 위해 sema\_up 함수를 이용하여 parent thread의 file\_semaphore을 처리해주고, success가 0인 경우에 프로세스가 종료되도록 한다.

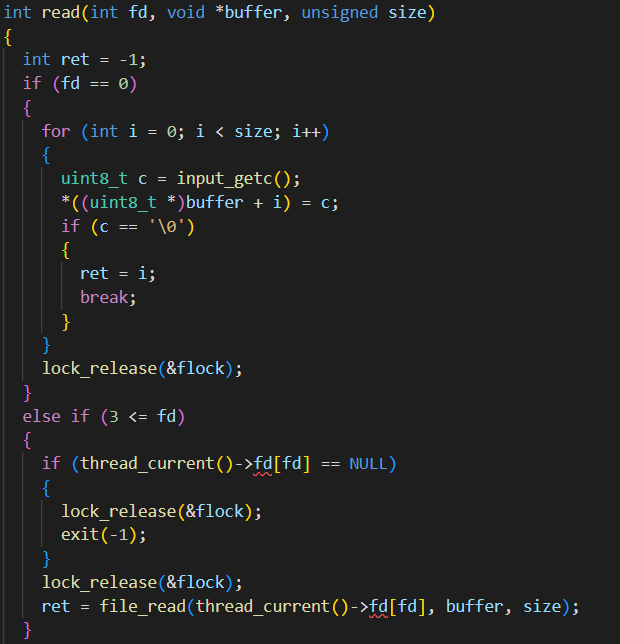
**(3) load**



process.c의 load 함수 내에서 위와 같은 부분을 추가한다. success가 1인 경우에 다른 wirte 작업이 실행되는 것을 막기 위해 file\_deny\_write 함수를 이용하여 처리해준다.

**7) userprog/syscall.c**





추가로 구현한 파일 관련 시스템 콜 함수들 중 open, read, write 작업을 할 때, 작업이 수행되기 전에 lock\_acquire 함수를 호출하고, 작업이 마쳤다면 이후에 lock\_release 함수를 호출하여 race condition 등의 문제점이 발생하지 않도록 한다.

* 1. **시험 및 평가 내용**
* make check 수행 결과를 캡처하여 첨부  
  