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

담당 교수 / 분반 : 박성용 / 3

이름 / 학번 : 조보현 / 20160641

개발 기간 : 2021/11/01~2021/11/14

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술

file system에 대한 system call들(create, remove, open, close, filesize,read,write,seek,tell)을 구현한다. 이때 Synchronization을 구현해서 reader/writer 문제 등이 발생하지 않도록 한다.

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 파일을 접근하기 위해서는 File Descriptor를 사용해야 한다. 따라서 File Descriptor를 구현하면 파일에 대한 읽기, 쓰기, 닫기 등의 작업을 수행 할 수 있다.

2. (이번 프로젝트에서 구현해야 하는) System Calls

파일 생성-create, 파일 삭제-remove, 파일 열기-open, 파일 사이즈 반환-filesize, 파일 읽기-read, 파일 쓰기-write, 파일 찾기-seek, 파일 주소 조회-tell 등의 작업을 수행 할 수 있게 된다.

3. Synchronization in Filesystem

어떤 프로세스가 파일을 읽거나 쓰고 있을 때 다른 프로세스가 해당 파일을 읽거나 쓰는 등의 동기화 문제가 발생할 수 있는데, 이를 방지하기 위해 Synchronization을 구현해야 한다. 따라서 Synchronization을 구현하면 critical section 문제를 방지 할 수 있다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. File Descriptor: 구현에 이용할 자료구조와 선택한 이유를 서술

File Descriptor는 간단하게 배열을 이용한다. Pintos Manual에 따르면 프로세스당 128개의 파일을 open 할 수 있으므로 배열의 크기를 130(stdin, stdout 포함)으로 고정시킬 수 있기 때문에 구현도 간단하고 O(1)에 접근 할 수 있는 배열을 사용한다.

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)

create : 파일 이름과 size를 파라미터로 받아 새로운 파일을 생성한다. 성공-true, 실패-false 반환

remove : 파일 이름을 통해 해당 파일을 삭제한다. 성공-true, 실패-false 반환

open : 파일 이름을 통해 해당 파일을 연다. 성공-file descriptor, 실패- -1 반환

close : 파일 descriptor를 통해 해당 파일을 닫는다.

filesize : 파일 descriptor를 통해 해당 파일의 size를 반환한다.

read : 파일 descriptor를 통해 해당 파일을 size만큼 buffer로 읽는다. 성공-읽은 바이트 수, 실패 - -1 반환

write : buffer에서 size만큼의 데이터를 해당 file descriptor에 해당하는 파일에 쓴다.

seek : 파일 descriptor를 통해 해당 파일의 읽거나 쓸 다음 위치를 pos로 변경한다.

tell : 파일 descriptor를 통해 해당 파일의 읽거나 쓸 다음 위치를 반환한다.

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

어떤 프로세스가 파일에 대하여 열기, 읽기, 쓰기 등의 작업을 하기 전에 lock을 얻고 작업이 끝나면 lock을 풀어주는 방법을 사용할 수 있다. 혹은 lock 대신 semaphore를 이용해서 lock과 비슷한 방식으로 동기화 문제를 해결 할 수 있다. 이는 sema\_down, sema\_up 을 이용하면 가능하다. reader/writer 문제를 해결하기 위해서는 wrt, mutex 두 개의 semaphore를 사용하면 될 것이다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

1. file Descriptor 구현 : 2021/11/01~2021/11/03

2. System Calls 구현 : 2021/11/04~2021/11/10

3. Synchronization 구현 : 2021/11/11~2021/11/13

4. Document 작성: 2021/11/14

* 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

File Descriptor 구현을 위해 src/thread/thread.h의 thread 구조체를 변경한다. file descriptor를 저장할 배열 file 배열을 핀토스 메뉴얼에 따라 사이즈 130으로 설정해서 추가한다. 또한 다음 fd를 가리킬 int fd를 추가해 새로운 파일을 열 때마다 fd를 통해 열린 파일의 포인터가 바로 배열에 추가될 수 있도록 한다. src/thread/thread.c에서 thread\_init(), thread\_create()함수 부분에서 file 배열과 fd 등을 초기화하는 코드를 추가한다. src/userprog/syscall.c 에는 fill이 NULL인지 check 하는 함수를 추가한다.

2. System Calls

src/userprog/syscall.c에 create, remove, open, close, filesize, seek, tell 함수를 추가한다. 이는 pintos에서 제공하는 filesys API를 이용해서 구현한다. 필요시 함수를 추가 구현한다. syscall\_handler 또한 각각의 SYSTEM CALL NUMBER에 맞게 작동할 수 있도록 수정한다. 또한 오류를 방지하기 위해 fd와 file 배열들의 유효성도 check 하는 함수를 작성한다. 기존에 있던 read(), write() 함수는 stdin, stdout 뿐만 아니라 파일에도 쓰거나 읽을 수 있도록 수정한다.

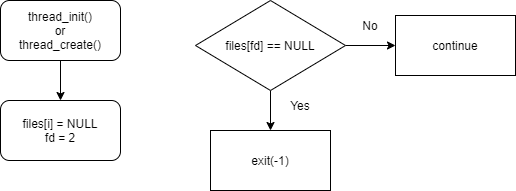
3. Synchronization

src/thread/thread.h의 thread 구조체에서 자식과 부모간의 동기화 문제를 해결하기 위해 thread 포인터 parent와 semaphore waitChild를 추가한다. src/userprog/syscall.c에서 reader/writer 문제를 해결하기 위해 semaphore wrt와 mutex를 src/userprog/syscall.h에 추가해놓고 이용한다. syscall\_init() 함수에서 sema\_init()을 이용해 둘을 초기화한다. read(), open(), write() 함수에서 위 두 semaphore를 이용해 동기화 문제를 처리한다.

1. **연구 결과**
   1. **Flow Chart**

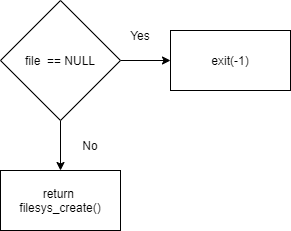
* II. B. 개발 내용의 각 3가지 항목에 대하여 Flow Chart 작성

**1. File Descriptor**

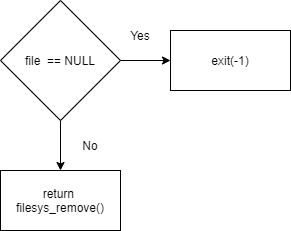


**2. System Calls**

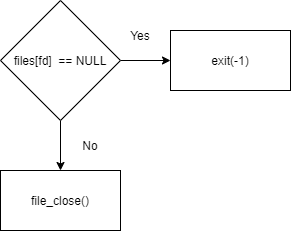
**1) create**

****

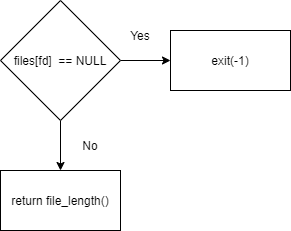
**2) remove**

****

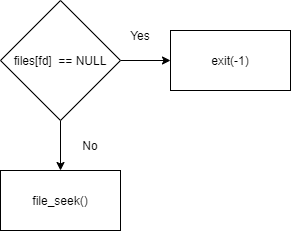
**3) close**

****

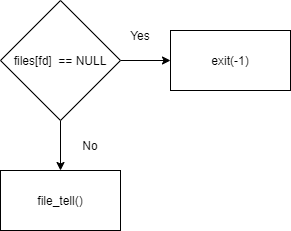
**4) filesize**

****

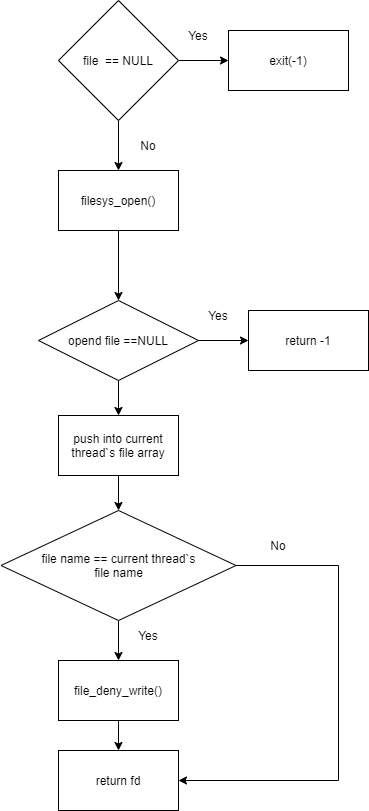
**5) seek**

****

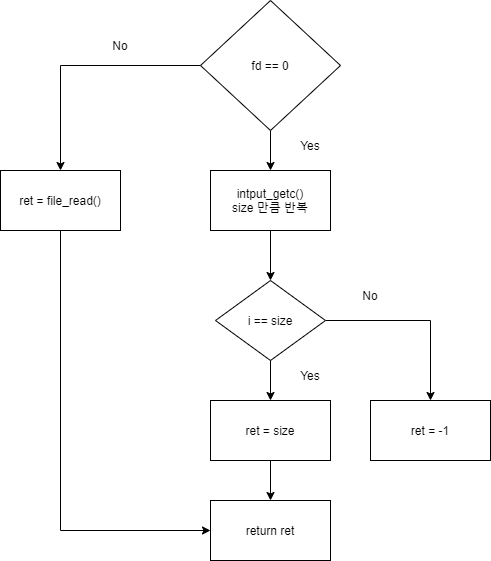
**6) tell**

****

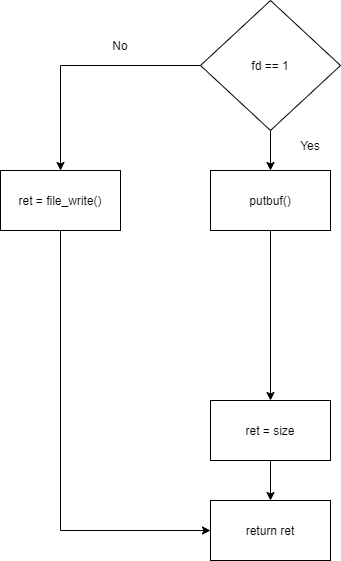
**7) open**

****

**8) read**

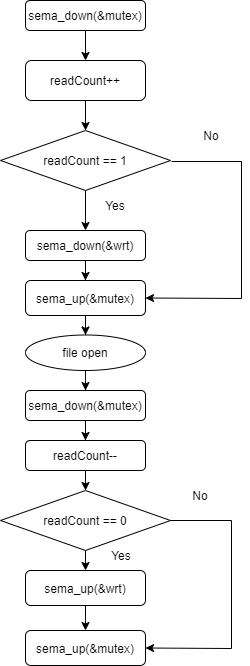
****

**9) write**

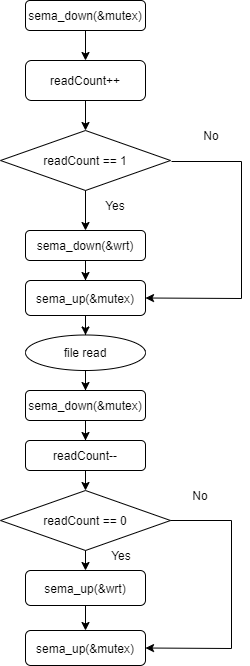
****

**3. synchronization**

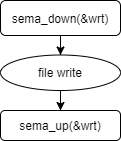
**1) open**

****

**2) read**

****

**3) write**

****

* 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**

텍스트이(가) 표시된 사진

자동 생성된 설명

src/threads/thread.h에서 thread 구조체에 thread가 open한 파일들을 가리킬 포인터들을 저장할 배열 files를 추가했고, 다음 file descriptor를 가리킬 int fd를 추가했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

src/thread/thread.c에서 thread\_init() 함수에서 새로 추가한 file 포인터 배열과 fd배열을 초기화하는 코드를 추가했다. fd를 2로 초기화한 이유는 0과 1이 STDIN, STDOUT을 가리키기 때문이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

src/userprog/syscall.c에서 fd가 가리키는 file 포인터가 NULL인지 체크하는 함수 checkFd()를 추가한다.

**2. System Calls**

**1) create**

텍스트이(가) 표시된 사진

자동 생성된 설명

src/userprog/syscall.c 에서 파일을 생성하는 filesys\_create() 함수를 이용해 파일을 생성한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

syscall\_handler에도 SYS\_CREATE를 처리하기 위해 위와 같은 코드를 추가한다. file이 NULL인지도 체크해주고, file의 유효성도 검사하고, check함수를 통해 유효성을 체크한다.

**2) remove**

텍스트이(가) 표시된 사진

자동 생성된 설명

src/userprog/syscall.c 에서 파일을 삭제하는 filesys\_remove() 함수를 이용해 파일을 삭제한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마찬가지로 syscall\_handler에서 유효성 검사를 해주고, remove함수를 수행하도록 한다.

**3) open**

텍스트이(가) 표시된 사진

자동 생성된 설명

src/userprog/syscall.c에서 파일을 여는 filesys\_open() 함수를 이용해 파일을 연다. 이 때 파일을 여는 것이 실패할 경우에는 -1을 반환하고 성공한다면 현재 쓰레드의 fd를 이용하여 열린 파일이 file pointer 배열에 저장되도록 한다. 또한 현재 쓰레드의 파일 이름과 열린 파일의 이름을 비교하여 같다면 해당 파일에 쓰지 못하도록 file\_deny\_write() 함수를 사용한다. 마지막으로 fd를 1증가시켜 fd가 null인 공간을 가리킬 수 있도록 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마찬가지로 syscall\_handler 에서 유효성 검사를 하고 open()함수를 수행하도록 한다.

**4) close**

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 thread에서 해당 fd가 가리키는 파일을 file\_close()함수를 이용해 닫는다. 또한 해당 포인터가 가리키는 부분을 NULL로 초기화 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마찬가지로 syscall\_handler 에서 유효성검사를 하고 close() 함수를 수행하도록 한다.

**5) filesize**

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 thread의 fd가 가리키는 파일의 size를 file\_length() 함수를 통해 반환하도록 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마찬가지로 syscall\_handler 에서 유효성검사를 하고 getFileSize() 함수를 수행하도록 한다.

**6) seek**

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 thread의 fd가 가리키는 파일에 접근하여 file\_seek()함수를 이용해 현재 위치를 파일의 시작지점에서 pos만큼의 위치로 바꿔준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마찬가지로 syscall\_handler 에서 유효성 검사를 하고 seek() 함수를 수행하도록 한다.

**7) tell**

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 thread의 fd가 가리키는 파일에 접근하여 file\_tell()함수를 이용해 해당 파일안의 다음으로 읽거나 쓸 현재 위치를 반환하여 준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마찬가지로 syscall\_handler 에서 유효성 검사를 하고 tell() 함수를 수행하도록 한다.

**8) write**

텍스트이(가) 표시된 사진

자동 생성된 설명fd가 1이면 buffer의 내용을 size만큼 읽어 stdout에 출력한다. 1이 아니라면 file\_write()함수를 이용해 해당 파일에 size만큼 내용을 출력하도록 한다.

syscall\_handler는 수정할 부분이 없다.

**9) read**

텍스트이(가) 표시된 사진

자동 생성된 설명

fd가 0이라면 STDIN으로부터 내용을 읽고, 0이아니라면 file\_read() 함수를 이용해 해당 파일로부터 내용을 읽어들인다.

syscall\_handler는 수정할 부분이 없다.

**3. Synchronization**

텍스트이(가) 표시된 사진

자동 생성된 설명

src/threads/thread.h 에서 thread 구조체에 부모를 가리킬 parent 포인터와 자식들을 기다리는 semaphore waitChild를 추가한다.

텍스트, 오렌지, 장치, 설정이(가) 표시된 사진

자동 생성된 설명

src/userprog/syscall.h에 readers/writers 문제를 해결하기 위한 wrt, mutex와 readCount를 전역변수로 선언해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

src/userprog/syscall.c에서 syscall\_init() 함수에서 sema\_init() 함수를 이용해 mutex, wrt를 1로 초기화 해준다. readCount는 0으로 초기화한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

파일을 여는 open() 함수에서 동기화 문제를 해결하기 위해 sema\_down(), sema\_up() 함수를 이용해 mutex, wrt로 critical section을 감싸준다. 해당 코드는 강의자료의 내용을 참고하여 작성했다. 위와 같은 방법으로 동기화 문제를 해결했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

read() 함수도 똑같은 방법으로 동기화 문제를 해결했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

write() 함수는 wrt 세마포어를 이용하여 동기화 문제를 해결했다.

**#문제 발생 ( multi-oom 오류 )**

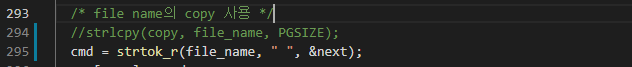
텍스트, 오렌지, 스크린샷이(가) 표시된 사진

자동 생성된 설명

모든 테스트 케이스를 통과했지만, multi-oom 테스트를 계속해서 통과하지 못했다. 오류 메시지를 보면 strlcpy() 함수에서 오류가 발생하는 것을 확인 할 수 있었다.



src/userprog/process.c 에서 load() 함수에서 file\_name의 변경을 막고자 해당 copy 버전을 사용했으나, 이 부분이 문제가 되었다.



위와 같이 copy 버전이 아닌 file\_name 원본을 사용하도록 바꾸었더니 문제없이 테스트를 통과할 수 있었다.

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

텍스트이(가) 표시된 사진

자동 생성된 설명