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

담당 교수 / 분반 : 박성용 교수님

이름 / 학번 : 홍주표 / 20181702

개발 기간 : 2022.10.05 ~ 2022.10.10

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술

프로젝트 1에서 구현하지 않은 system call인 create, remove, open, close, filesize, seek, tell을 구현하고 read와 write를 각각 STDIN과 STDOUT이 아닌 file에 접근하도록 수정한다. 또한 동기화 문제를 해결하여 80개의 테스트 중 지난 테스트 21개를 제외한 나머지 테스트도 통과하도록 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

system call을 통해 특정 file에 접근하기 위해서는 file descriptor가 필요하다. file descriptor를 구현함으로써 각 thread가 파일에 접근할 수 있게 된다.

2. (이번 프로젝트에서 구현해야 하는) System Calls

create, remove, open, close, filesize, seek, tell을 기능에 맞게 구현하고 프로젝트 1에서 구현한 read와 write를 각각 STDIN과 STDOUT이 아닌 file에 접근하도록 수정한다.

3. Synchronization in Filesystem

여러 process에서 하나의 파일에 접근하고 그 파일을 사용할 때 일어날 수 있는 문제를 해결한다. semaphore, lock과 같은 방식을 사용하여 이를 해결한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. File Descriptor: 구현에 이용할 자료구조와 선택한 이유를 서술

기존에 사용한 thread 구조체에 구조체 배열을 추가하여 file descriptor를 구현한다. 그리고 thread가 생성될 때 이 구조체 배열은 NULL로 초기화되고 이 구조체 배열을 사용하여 파일들에 접근할 수 있게 된다. 이 구조체 배열에 file descriptor를 저장함으로써 다른 동작을 실행할 때 저장한 file descriptor를 사용하여 해당 파일에 접근할 수 있게 된다.

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. SYS\_CREATE (bool create (const char \*file, unsigned initial\_size))

'initial\_size' byte 크기를 갖고 'file'의 파일명을 갖는 파일을 하나 생성한다. 성공이나 실패 여부에 따라 true, false를 반환한다. 이때 파일을 열지는 않는다.

1. remove (bool remove (const char \*file))

'file'의 파일명을 갖는 파일을 지운다. 성공이나 실패 여부에 따라 true, false를 반환한다. 파일은 열려 있든 닫혀 있든 지워져야 한다.

1. open (int open (const char \*file))

'file'의 파일명을 갖는 파일을 연다. 그 file descriptor를 반환하거나 open에 실패한 경우 -1을 반환한다.

1. close (void close (int fd))

'fd'의 file descriptor를 갖는 파일을 닫는다.

1. filesize (int filesize (int fd))

'fd'의 file descriptor를 갖는 파일의 사이즈를 byte 단위로 반환한다.

1. read (int read (int fd, void \*buffer, unsigned size))

기존에 구현했던 STDIN (fd == 0) 뿐만 아니라 다른 파일에 대해서도 구현한다. 'fd'의 file descriptor를 갖는 열린 파일에 'buffer'에 'size'만큼 읽는다. 실제로 읽은 byte 수를 반환하거나, 읽기에 실패한 경우 -1을 반환한다.

1. write (int write (fd, const void \*buffer, unsigned size))

기존에 구현했던 STDOUT (fd == 1) 뿐만 아니라 다른 파일에 대해서도 구현한다. 'fd'의 file descriptor를 갖는 열린 파일에 'size'만큼의 'buffer'로부터 쓴다. 실제로 쓴 byte 수를 반환한다.

1. seek (void seek (int fd, unsigned position))

'fd'의 file descriptor를 갖는 열린 파일에 읽거나 쓸 다음 바이트의 위치를 바꾼다.

1. tell (unsigned tell (int fd))

'fd'의 file descriptor를 갖는 열린 파일에 읽거나 쓸 다음 바이트의 위치를 반환한다.

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

system call 중 read, write 그리고 open에서 lock을 사용한다. 이때 lock\_acquire와 lock\_release를 사용하는데, 쉽게 생각해서 여럿이 접근을 하더라도 하나가 접근할 때 lock을 걸어 다른 이의 접근을 막고 그 하나가 동작을 끝낼 때 lock을 풀어 다른 이가 접근할 수 있게 한다. 이때 사용하는 lock과 관련된 함수는 내부적으로 semaphore를 사용하기에 semaphore(sema\_down, sema\_up)와 큰 차이가 없다.

또한 process 간의 synchronization에서 semaphore를 사용하여 문제를 해결한다. 간단히 설명하자면 process\_execute에서 sema\_down을 호출하고 start\_process에서 sema\_up을 호출하는 방식으로 synchronization 문제를 해결한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

22.10.05 ~ 22.10.06: create, remove, open, close, filesize, seek, tell system call 구현

22.10.07 ~ 22.10.09: read, write system call 기존 함수에 기능 추가 및 synchronization 문제 해결

22.10.10: 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. pintos/src/threads/thread.h에 file descriptor를 위한 변수를 추가한다.

thread 구조체 내에 struct file \*fd[128]; 을 추가하여 각 프로세스에서 고유한 file descriptor 테이블을 가질 수 있게 한다.

1. pintos/src/threads/thread.c에 위에서 생성한 fd를 모두 NULL로 초기화하는 코드를 추가한다.
2. pintos/src/userprog/syscall.c에 추가로 구현해야 하는 system call을 구현한다.

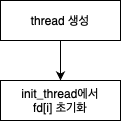
create, remove, open, close, filesize, seek, tell 함수를 구현하고 기존의 read와 write 함수를 수정한다. 그리고 syscall\_handler 함수에 SYS\_CREATE, SYS\_REMOVE, SYS\_OPEN, SYS\_CLOSE, SYS\_FILESIZE, SYS\_SEEK, SYS\_TELL을 추가하여 각각에 맞 는 함수를 호출한다. 이때 프로젝트 1과 마찬가지로 is\_user\_vaddr 함수를 사용 하여 invalid memory access를 처리한다. 또한 synchronization 문제를 해결하기 위해 open, read, write 함수에서 lock에 관련된 함수(lock\_acquire, lock\_release)를 사용한다.

자세한 코드에 대한 설명은 아래에서 하겠다.

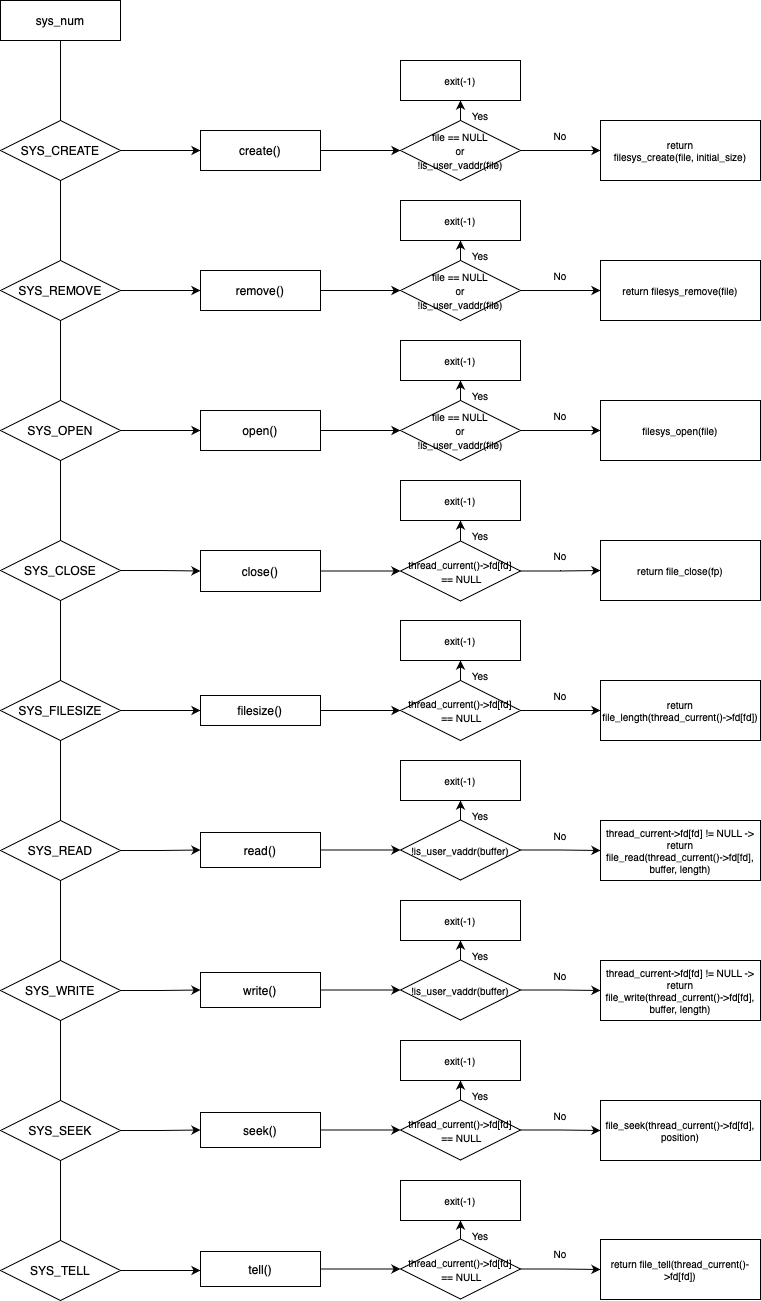
1. pintos/src/userprog/syscall.h에 추가로 구현한 system call 함수의 prototype을 선언한다.
2. pintos/src/threads/thread.h에 동기화를 해결하기 위해 struct semaphore load\_lock을 추가한다. 또한 이를 pintos/src/threads/thread.c에서 sema\_init으로 초기화하고 pintos/src/userprog/process.c의 process\_execute 함수와 start\_process 함수에 사용한다.
3. **연구 결과**
   1. **Flow Chart**

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

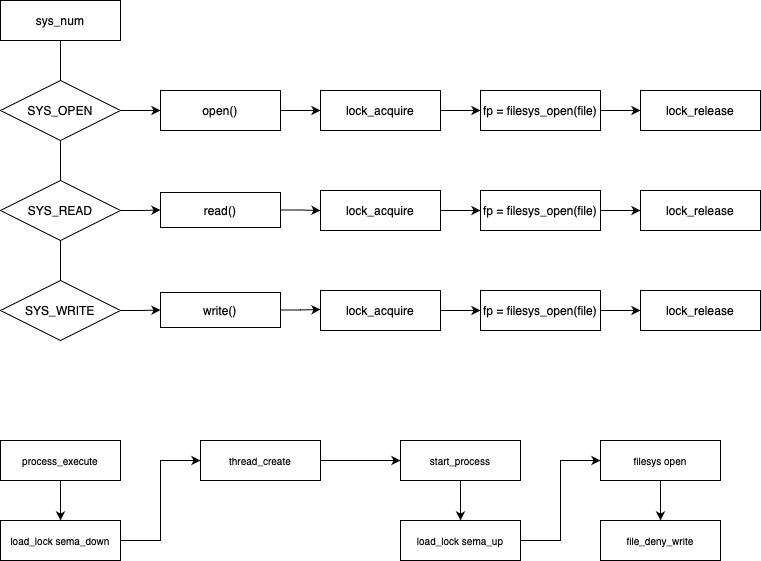
1. File Descriptor



1. System Calls



1. Synchronization in File System



* 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. pintos/src/threads/thread.h 중 struct thread에서

텍스트이(가) 표시된 사진

자동 생성된 설명

기존에 있던 내용에 file\_descriptor 사용을 위해 struct file \*fd[128]을 추가한다.

또한 pintos/src/threads/thread.c 중 init\_thread 함수에서

텍스트이(가) 표시된 사진

자동 생성된 설명

다음과 같은 코드를 추가하여 thread가 생성될 때 모든 fd[i]를 NULL로 초기화한다.

이로써 file descriptor를 사용할 준비를 마친다.

1. pintos/src/userprog/syscall.c에서 system call 함수를 구현한다.
   * + 1. bool create (const char \*file, unsigned initial\_size)

텍스트이(가) 표시된 사진

자동 생성된 설명

SYS\_CREATE에서 create 함수를 호출한다.

fie == NULL 이거나 !is\_user\_vaddr(file) 인 경우 exit(-1)을 한다.

create 함수는 filesys\_create(file, initial\_size) 함수를 호출하고 그 결과를 반환한다.

* + - 1. bool remove (const char \*file)

텍스트이(가) 표시된 사진

자동 생성된 설명

SYS\_REMOVE에서 remove 함수를 호출한다.

file == NULL 이거나 !is\_user\_vaddr(file) 인 경우 exit(-1)을 한다.

remove 함수는 filesys\_remove(file) 함수를 호출하고 그 결과를 반환한다.

* + - 1. int open (const char \*file)

텍스트이(가) 표시된 사진

자동 생성된 설명

SYS\_OPEN에서 open 함수를 호출한다.

file == NULL 이거나 !is\_user\_vaddr(file) 인 경우 exit(-1)을 한다.

synchronization 문제를 해결하기 위해서 lock\_acquire를 한다. 그리고 filesys\_open(file) 함수를 호출한다. fp == NULL 인 경우 lock\_release를 하고 -1을 반환한다. i=3부터 순회하면서 fd[i]가 NULL 인 경우 그 자리에 fp 값을 넣고 lock\_release를 한 후 그 fp 값을 반환한다. 그 전에 thread\_current의 name 변수 와 file 변수가 같으면 file\_deny\_write를 한다.

* + - 1. void close (int fd)

텍스트이(가) 표시된 사진

자동 생성된 설명

SYS\_CLOSE에서 close 함수를 호출한다.

현재 thread의 fd가 NULL 인 경우 exit(-1)을 한다.

그리고 fp에 현재 thread의 fd 값을 저장하고 해당 fd 값은 NULL로 바꾼다. 그 리고 file\_close(fp)를 호출하고 그 결과를 반환한다.

* + - 1. int filesize (int fd)

텍스트이(가) 표시된 사진

자동 생성된 설명

SYS\_FILESIZE에서 filesize 함수를 호출한다.

현재 thread의 fd가 NULL 인 경우 exit(-1)을 한다.

filesize 함수는 file\_length(thread\_current()->fd[fd]) 함수를 호출하고 그 결과를 반환한다.

* + - 1. void seek (int fd, unsigned position)

텍스트이(가) 표시된 사진

자동 생성된 설명

SYS\_SEEK에서 seek 함수를 호출한다.

현재 thread의 fd가 NULL 인 경우 exit(-1)을 한다.

seek 함수는 file\_seek(thread\_current()->fd[fd]) 함수를 호출하고 그 결과를 반환한다.

* + - 1. unsigned tell (int fd)

텍스트이(가) 표시된 사진

자동 생성된 설명

SYS\_TELL에서 tell 함수를 호출한다.

현재 thread의 fd가 NULL 인 경우 exit(-1)을 한다.

tell 함수는 file\_tell(thread\_current()->fd[fd]) 함수를 호출하고 그 결과를 반환한다.

* + - 1. int read (int fd, void \*buffer, unsigned length)

텍스트이(가) 표시된 사진

자동 생성된 설명

SYS\_READ에서 read 함수를 호출한다.

기존 코드는 fd가 0일 때에 대한 기능이므로 나머지 부분을 구현한다.

!is\_user\_vaddr(buffer) 이면 exit(-1)을 한다.

synchronization 문제를 해결하기 위해 lock\_acquire를 한다. 그리고 오류로 exit 을 하거나 return을 하기 전에 lock\_release를 한다. fd가 3 이상 (0은 STDIN, 1은 STDOUT, 2는 STDERR)인 경우에 현재 thread의 fd 값이 NULL이 아니라면 file\_read(thread\_current()->fd[fd], buffer, length) 함수를 호출하고 그 결과를 반환 한다.

* + - 1. int write (int fd, const void \*buffer, unsigned length)

텍스트이(가) 표시된 사진

자동 생성된 설명

SYS\_WRITE에서 write 함수를 호출한다.

기존 코드는 fd가 1일 때에 대한 기능이므로 나머지 부분을 구현한다.

!is\_user\_vaddr(buffer) 이면 exit(-1)을 한다.

synchronization 문제를 해결하기 위해 lock\_acquire를 한다. 그리고 오류로 exit 을 하거나 return을 하기 전에 lock\_release를 한다. fd가 3 이상 (0은 STDIN, 1은 STDOUT, 2는 STDERR)인 경우에 현재 thread의 fd 값이 NULL이 아니라면 file\_read(thread\_current()->fd[fd], buffer, length) 함수를 호출하고 그 결과를 반환 한다. 그 전에 현재 thread의 fd에 할당된 deny\_write 값이 0이 아니라면 file\_deny\_write를 한다.

그리고 syscall\_handler 함수에 다음과 같은 코드를 추가한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. pintos/src/userprog/process.c 중 process\_execute 함수에서 semaphore를 이용하여 process에 대한 synchronization 문제를 해결한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

또한 start\_process 함수에서도 같은 방식으로 문제를 해결한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

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

텍스트이(가) 표시된 사진

자동 생성된 설명