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

담당 교수 / 분반 : 김영재 / 2분반

이름 / 학번 : 김준우 / 20211524

개발 기간 : 10/16 ~ 10/29

1. **개발 목표**

File system에 관한 system call인 open, close, create, remove, read, write, seek, tell, file size 함수들을 구현하고, file system의 동기화 기능을 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

프로세스는 file descriptor를 이용해 파일에 접근해야 한다. 따라서 Pintos에서 각 프로세스 별로 file descriptor를 만들어 파일들을 관리할 수 있도록 구현한다. 이를 이용해 각 thread가 file에 접근해 이용할 수 있게 된다.

2. (이번 프로젝트에서 구현해야 하는) System Calls

File에 관련된 system call들로 user level에서 system call을 호출 했을 때 syscall handler를 통해 file system에 관련된 kernel 함수가 실행된다.

3. Synchronization in Filesystem

File을 여러 프로세스(thread)에서 접근해 이용할 수 있기 때문에, 동기화가 필요하다. File을 read/write하는 경우, 프로세스가 execute/exit 할 때에 발생하는 문제들에 대한 해결법이다. 여러 프로세스에서 한 번에 file에 접근해 변경하는 것을 방지할 수 있고, 프로세스가 execute ,exit 될 때 parent와 child 프로세스 상태를 확인해 오류가 발생하는 것을 방지한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. File Descriptor: 구현에 이용할 자료구조와 선택한 이유를 서술

File Descriptor의 구현에 자료구조로는 배열을 선택했다. System call 함수에서 fd를 인자로 받아올 때 정수로 받아오는데 이 정수를 이용해 가장 간단히 file descriptor를 이용 할 수 있는 자료구조가 배열이기 때문에 배열을 선택했다.

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 : 인자로 file의 이름과 크기를 받아와 filesys\_create 함수를 호출해 파일을 생성하고 생성되면 true 아니면 false를 return하는 함수이다.

- remove : 인자로 file의 이름을 받아와 filesys\_remove 함수를 호출해 파일을 삭제하고 삭제가 되면 true 아니면 false를 return하는 함수이다.

- open : 인자로 file의 이름을 받아와 현재 thread의 file descriptor의 빈 fd를 찾아 file pointer를 할당하고 fd를 return 해주는 함수이다.

- filesize : 인자로 open file의 fd를 받아와 현재 thread의 file descriptor의 해당 fd에 할당된 파일의 길이를 file\_length 함수를 호출해 return 해주는 함수이다.

- seek : 인자로 open file의 fd와 position을 받아 file\_seek 함수를 호출해 열린 파일의 offset으로 이동하는 함수이다.

- tell : 인자로 open file의 fd를 받아 file\_tell 함수를 호출해 열린 파일의 위치를 return해주는 함수이다.

- close : 인자로 open file의 fd를 받아 해당 file descriptor의 열린 파일을 닫아주고 초기화 해주는 함수이다.

- read : fd값이 3 이상인 경우에 file에서 읽어오는 작업을 해줘야 하기 때문에, file\_read 함수를 호출해 이 결과를 return 해준다.

- write : fd값이 3 이상인 경우에 file에 쓰는 작업을 해줘야 때문에, file\_write함수를 호출해 이 결과를 return 해준다.

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

File system에 사용되는 system call에서 lock을 이용해 file을 보호해준다. File\_lock이라는 lock 구조체를 선언해 lock\_acquire, lock\_release를 이용해 사용할 것이다. Lock을 이용해 여러 프로세스에서 같은 file에 접근해 읽기, 쓰기, 지우기를 하는 상황을 방지한다. Open, write, read 함수에서 이용한다.

프로세스 사이의 동기화에서 semaphore를 이용하는데 project 1에서 wait와 exit을 하는 경우에 semphore를 이용해 동기화 해주었고, 이번에는 sema\_load라는 새로운 semaphore를 선언해 파일을 이용하는 경우인 exec, start 에서 sema\_up과 sema\_down으로 관리 해주었다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

**10/16 ~ 10/18 : system call 함수 구현**

**10/26 : thread에 file descriptor, semaphore 관련 추가, init에 추가**

**10/27~10/28 : semaphore를 이용해 동기화 구현**

**10/29 : 오류 수정, 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

userprog/syscall.c에 file system과 관련된 systemcall들을 구현하고, syscall\_handler 함수에 system call number에 맞게 호출할 수 있도록 추가한다. 인자가 넘어가는 방식은 project 1과 같이 stack에 쌓인 argument를 넘긴다. Read와 write, exit 함수에도 file descriptor와 관한 부분을 추가하고 read, write ,open 함수에는 Lock을 이용해 함수 시작에서 lock\_acquire를 하고 return을 하기 전 lock\_release를 해주어 file을 보호해준다. 각 함수에서 fd나 file 이름이 NULL이면 exit(-1)로 예외처리를 해준다. userprog/syscall.h 에서 각 함수들을 추가해준다.

threads/thread.h 에 file descripter배열인 fd를 128의 크기로 선언해주고, child의 실행 여부를 관리하는 semaphore인 sema\_load, parent process를 가리키는 thread형 포인터 변수 thread, child가 종료되었는지 확인하는 flag를 선언해주었다. 그리고 이들도 thread가 만들어질 때 initialize 되어야 하기 때문에 threads/thread.c의 init\_thread 함수에서 file descriptor는 모두 NULL로, flag = 0, parent는 running\_thread()로 지정해주고 sema\_load도 init을 해준다.

userprog/process.c 에서 semaphore와 flag를 통해 여러 프로세스와 child를 관리 해준다. Process\_execute 함수에서 semphore를 사용해 child가 모두 종료 되도록 기다려줘 메모리를 정리한다. Process\_start에서는 load가 실패한 경우 flag를 1로 만들고 종료되도록 한다.

1. **연구 결과**
   1. **Flow Chart**

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

1. File\_descriptor 사용

**텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명**

1. System call (file system)

텍스트, 도표, 평면도, 평행이(가) 표시된 사진

자동 생성된 설명

3)

* 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

Userprog/process.c

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Process\_execute에서 sema\_load를 sema\_down해 list에 있는 elem들이 순차적으로 접근 하도록 해 parent가 child보다 먼저 종료되는 것을 막는다. flag로 child를 확인해 child list중 종료된것이 있으면 process\_wait함수를 호출해 return한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Process\_start에서 sema\_up해서 load가 정상적으로 종료되지 않은 경우를 보호한다. 그리고 load가 실패하면 flag를 1로 설정해 parent에서 확인할 수 잇게 하고 해당 thread를 exit으로 종료한다.

Userprog/syscall.c

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Syscall.c에서 lock을 이용하기 위해 새롭게 선언한 lock 구조체인 file\_lock을 lock\_init 함수를 통해 syscall이 init 될 때 같이 init 해줬다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Process를 종료하는 함수인 exit인데 종료하는 thread의 파일 포인터가 가리키는 file을 close 해준다. 3부터 하는 이유는 0,1,2는 각각 STDIN, STDOUT,STDERR기 때문이다.

텍스트, 스크린샷, 소프트웨어, 웹사이트이(가) 표시된 사진

자동 생성된 설명

create는 filesys/filesys.c에 정의된 함수인 filesys\_create를 통해 수행된다. inode를 생성, init 해주는 함수다. Free-map에서 inode 블록을 할당하고 file inode를 디크스에 기록하고 디렉토리 엔트리에 추가한다. 이 과정이 성공하면 bit map에 할당된 inode 블록을 해제하고 root dir에 inode메모리를 해제하고 success를 반환해준다.

remove는 filesys/filesys.c에 정의된 filesys\_remove 함수를 호출해 수행한다. root dir에서 파일을 찾아 있으면 root dir에서 메모리를 해지해 remove하고 true를 return 하고 아니면 false를 return 한다.

filesize는 fd번째 포인터가 가리키는 파일의 크기를 file\_length함수를 호출해 return 한다.

open은 file을 열어줘야 한다. file에 직접 접근해야 하기 때문에 lock\_acquire을 해주고 filesys\_open으로 file 포인터 fp에 저장해주고 이게 NULL이면 종료해준다. 그게 아니면 fd를 탐색해 NULL을 가리키고 있는 포인터에 연결해줘야 하는데, 파일명과 thread의 이름이 같으면 안 되기 때문에 예외처리를 해주었다. 그것이 아니면 파일 포인터를 연결해주고 lock\_release를 해주고 file\_descriptor의 위치를 return 한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

fd가 0인 경우는 STDIN이기 때문에, fd 가 3 이상인 경우에만 file 을 read해와야 한다. 따라서 else if로 fd>2 인 경우를 예외처리 해주었고, 파일 포인터가 NULL을 가리키면 exit(-1)로 종료해줬다. 그리고 file\_read함수의 결과를 새로운 지역변수에 저장하고 return 해주었다. file\_read함수는 file에서 buffer에 size만큼 읽어오고 읽어온 크기를 byte로 return하는 filesys/file.c에 선언된 함수이다. file을 보호하는 lock을 read의 시작부분에 acquire 해주고 exit하거나 return을 해주기 전에 release를 해준다. 결과를 새로운 지역변수에 저장해 준 이유도 return을 해주기 바로 전에 lock을 release 해줘야 하기 때문이다. 처음에 인자로 넘기는 과정에서 check\_user\_vaddr을 이용해 인자들을 넘겨서 확인이 필요 없다고 생각했는데 이렇게 하면 read-bad-ptr이 fail이 나오게 된다. 그래서 buffer가 NULL인지 한번 더 확인해줘 해결해주었다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

fd가 1인 경우는 STDOUT이므로 read와 마찬가지로 fd>2인 경우에 file에 접근해야 한다. read와 마찬가지로 check\_user\_vaddr로 buffer를 한번 더 확인해주고 file\_lock을 acquire해준다. 그리고 exit을 하기 전이나 return을 하기 전 release를 해준다. write는 filesys/file.c에 있는 file\_write함수를 호출해 수행되는데, file\_write는 buffer를 size만큼 파일에 써주고 byte수를 return해주는 filesys/file.c의 함수다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

seek은 열린 파일의 위치로 이동하는 함수다. position 만큼 이동하고 fd[fd]가 NULL인 경우에 예외처리 해준다. file\_seek을 호출해 열린 파일 위치로 이동해준다.

tell은 파일의 위치를 알려주는 함수로 file\_tell를 호출해 수행한다.

close는 file\_close함수를 호출해 fd[fd]를 닫아주고 이 포인터가 NULL을 가리키도록 해준다.

Threads/thread.h

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Parent process를 가리키는 thread 구조체 \*parent, file descriptor를 위한 file 구조체 포인터 배열 fd, 새로운 semphore인 sema\_load 그리고 child 관리를 해줄 flag를 새롭게 선언해줬다.

threads/thread.c

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

thread.h에서 새로 선언한 것을 init\_thread 함수에서 초기화 해주는 과정이다. 파일 포인터들은 모두 NULL로 초기화 해주고 sema\_init 함수를 이용해 새로 선언한 semaphore인 sema\_load를 초기화 하고 flag는 0, parent는 thread가 생성될때 현재 running 중인 thread기 때문에 running\_thread로 정해준다.

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

패턴, 패브릭, 스크린샷, 모노크롬이(가) 표시된 사진

자동 생성된 설명텍스트, 패턴, 스크린샷, 패브릭이(가) 표시된 사진

자동 생성된 설명