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

담당 교수 / 분반 : 김영재 교수님 / 1반

이름 / 학번 : 이도훈 / 20171663

개발 기간 : 10/05 – 10/16

1. **개발 목표**

저번 프로젝트에 추가로 파일 시스템 관련된 system call을 만든다. Create, remove, open, close, filesize, read, write 등의 system call을 추가로 구현하고 filesystem의 동기화를 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 통하여 특정 파일에 access한다. 따라서 각각의 프로세스가open한 파일들을 관리하기 위해서 file descriptor를 만든다. 이를 통해 각각의 thread가 파일에 access할 수 있다.

2. (이번 프로젝트에서 구현해야 하는) System Calls

명령어를 입력 받았을 때 system call handler를 사용해 명령어를 stack에서 load해서 명령어에 적합한 행동을 실행한다.

3. Synchronization in Filesystem

Synchronization in filesystem은 file을 read, write 할 시, process를 execute, exit할 때 사용된다. 파일이 여러 process에서 이용되는 것과, process가 execute, exit 종료될 때 오류가 발생하는 것을 방지한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. File Descriptor: 구현에 이용할 자료구조와 선택한 이유를 서술

Process가 서로 독립적이어야 하므로 struct thread 안의 fd를 배열로 선언했다.

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 : 새로운 file을 만드는데 사용하고 성공하면 true를 return하고 실패하면 false를 return한다.
2. remove : file을 삭제하는데 사용하고 성공하면 true를 return하고 실패하면 false를 return한다.
3. open : File을 open한다. open하는 것에 실패하거나 file descriptor를 return하면 -1을 return한다. File이 여러 번 open 되면 새로운 file descriptor를 return한다.
4. close : file descriptor를 close한다.
5. filesize : 입력받은 fd로 open하는 file의 size를 byte 단위로 return한다.
6. read : 입력받은 fd로 open하는 file을 read한다. 파일을 read하지 못하면 -1을 return한다.
7. write : 입력받은 fd로 open하는 file의 buffer size byte 만큼 write를 수행한다.
8. seek : 입력받은 fd로 open하는 file에서 다음 byte를 해당 위치로 옮긴다
9. tell : 입력받은 fd로 open하는 file에서 read, write하는 위치를 return한다.

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

File system에서의 system call에서 lock을 사용한다. 한 파일에서 read 와 write가 동시에 발생되는 경우를 방지하기 위해 file\_lock이라는 struct lock을 선언해서 사용했다. Read, write 명령어를 수행하기 전과 후 모두에 lock을 사용하여 동시에 발생되는 경우를 방지했다.

Process 사이의 synchronization에서 semaphore를 사용한다. Process가 execute, start 될 때 synchronization 해주었다. Struct thread에 새로운 구조체인 load\_sema를 선언하여 사용했다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

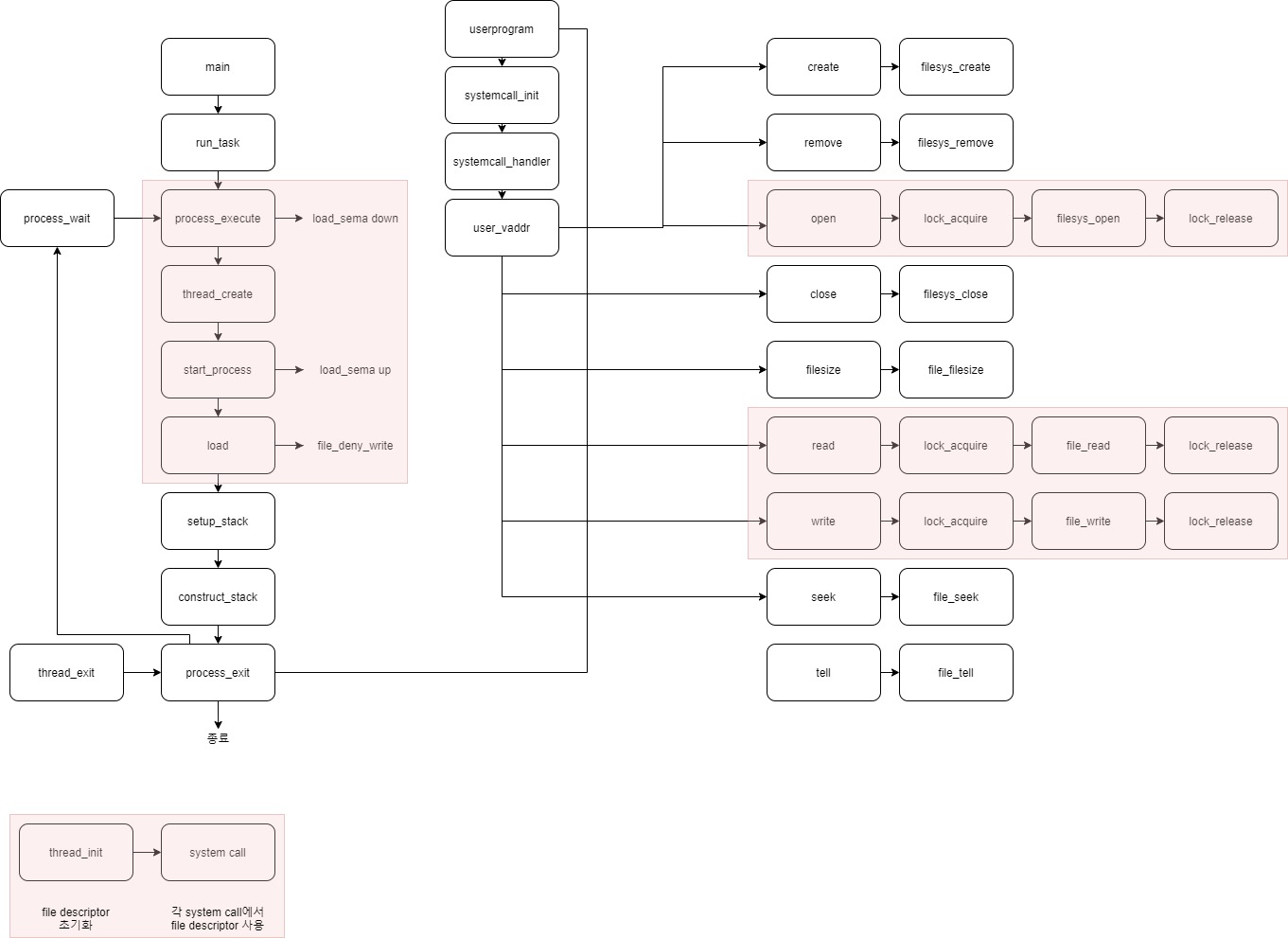
* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

**10/05 – 10/15 file descriptor & system call & synchronization 구현**

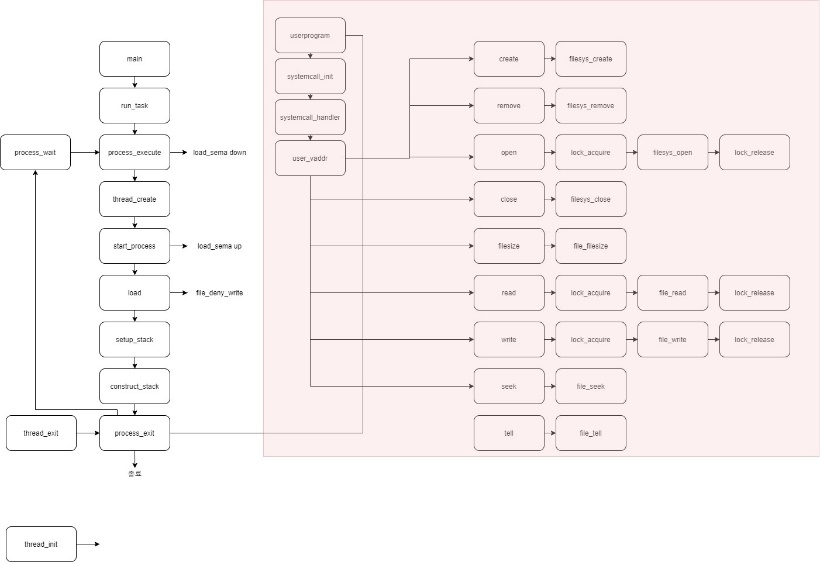
**10/15 – 10/16 보고서 작성**

1. **연구 결과**
   1. **Flow Chart**

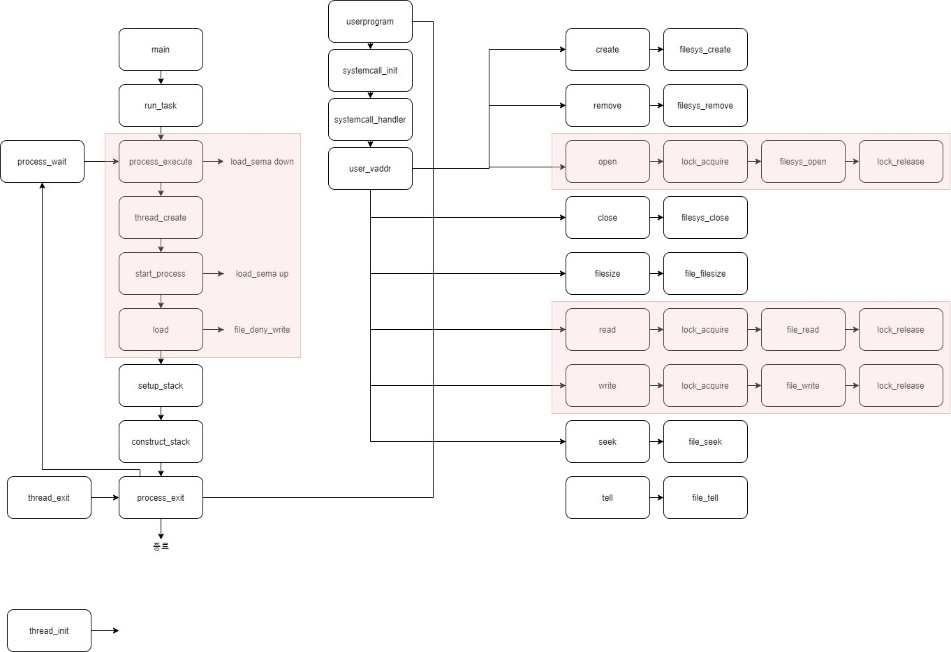
* II. B. 개발 내용의 각 3가지 항목에 대하여 Flow Chart 작성
  + 1. **File descriptor**



* + 1. **Systemcall**



* + 1. **Synchronization**



* 1. **시험 및 평가 내용**
* make check 수행 결과를 캡처하여 첨부
* **텍스트이(가) 표시된 사진

  자동 생성된 설명텍스트이(가) 표시된 사진

  자동 생성된 설명**