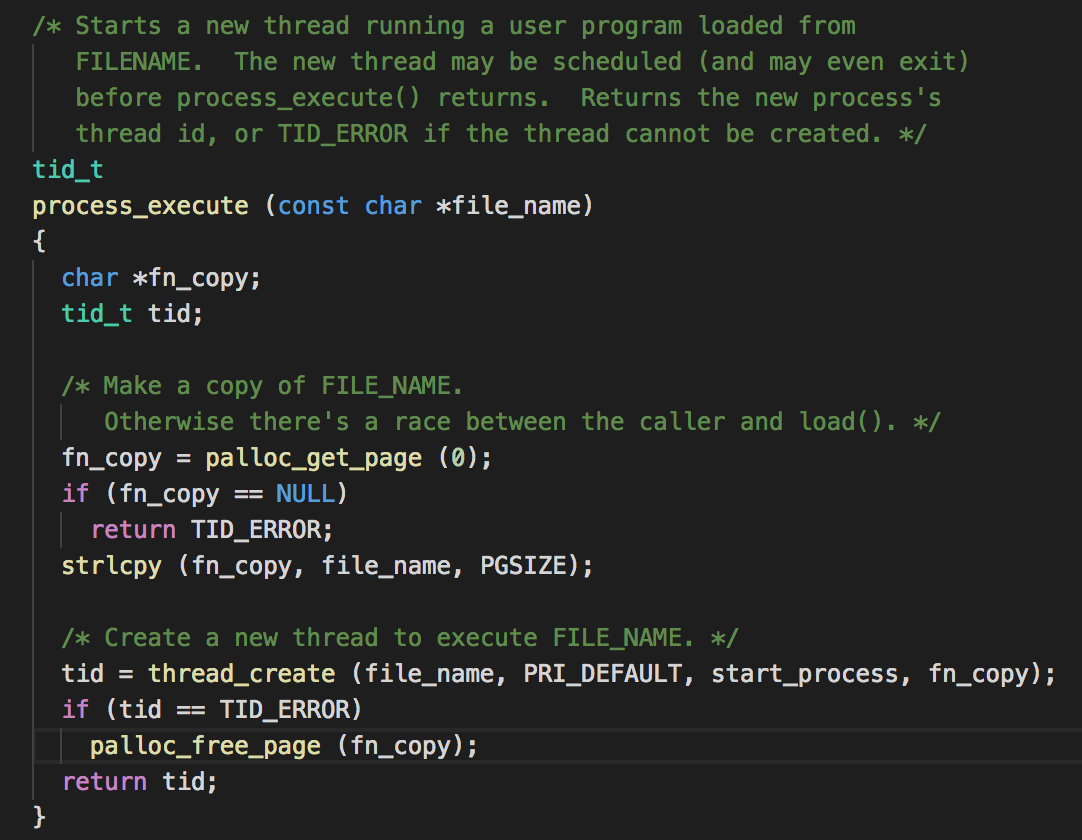
**Project 2 Design Report**

성해빈, 송창근

**1. 문제 해결**

1. User Process
   1. Argument Passing



process.c 에 있는 process\_execute 함수는 user program의 thread creation을 담당한다. 이때 argument를 따로 분리하지 않고 단순히 fn\_copy으로만 명령을 통째로 따내서 thread\_create의 aux, 즉 실행된 function의 인자로 전달한다. 이렇게 되면 실행할 각 user program function마다 argument parsing을 진행해야 하므로, 상당히 번거로워질 것이다. 미리 process\_execute function에서 char\*\* 타입을 만들어, argv를 따로 담는 2차원 배열을 만들자. “ls –l foo”가 커맨드라고 할 때, Fn\_copy[0] = “ls”, fn\_copy[1] = “-l”, fn\_copy[2] = “foo”가 되는 식으로 spacebar를 이용한 parsing을 수행한다. String parsing에는 string.h 에 있는 strtok\_r 을 이용하도록 한다.

* 1. Process Termination Messages

Thread termination이 일어날 때 메세지를 출력하는데, thread의 종말을 알리는 thread\_exit 함수에 삽입하는 것이 적절해 보인다. 여기서 출력하는 thread name은 process\_execute()에 전달된 full string에서 argument만 뺀 것이다. 한마디로 fn\_copy[0]에 해당하는 부분이다. 여기서 user process 에 해당하는 것만 메세지를 출력하고, halt system call이 발동된 경우라면 메시지를 출력하지 않는 것에 조심하자. Process load failure에도 메시지 출력을 하도록 한다.

1. System calls

System call 은 userprog디렉토리에 있는 syscall.c와 syscall.h를 수정하면 된다. Syscall은 각자에 해당하는 숫자가 있는데, lib/user/syscall.c에 syscall의 인자 개수마다 prototype이 만들어져 있고, 우리는 각 함수를 만들어주면 되는 식이다. Stack push와 ret으로 prototype이 정의되어있다.

* 1. User process manipulation

System call 중 user process 와 관련된 system call이다. Halt, exit, exec, wait 등이 해당된다.

* 1. File manipulation

System call 중 file 과 관련된 system call이다. File create, file remove, file open, file size, read, write, seek, tell, close등이 포함된다. Filesys directory와 filesys.h, file.h에 있는 함수들을 수정 없이 그대로 사용하라고 매뉴얼에 적혀있다.

1. Denying writes to executables
   1. 필요한 이유

어떤 thread가 현재 특정 파일을 이용하고 있는데 다른 thread가 파일을 수정하게 내버려둔다면 큰 문제가 생길 것이다. 이를 막기 위해 thread가 파일을 사용할 때 다른 thread가 수정하는 것을 방지해야하며, 이는 semaphore이나 lock을 사용할 필요가 있는 전형적인 상황이다.

**2. 현재 시스템 분석**

1. Process execution procedure

스레드 시스템은 thread.c 에 정의되어 있다. 스레드 구조체에는 다음과 같은 내용이 들어있다.

* tid: 스레드를 구별하기 위한 번호
* status: 스레드의 상태. RUNNING, 스레드는 다음과 같은 함수들이 처리한다.(미구현 함수 제외)
* thread\_init: 스레드를 초기화한다.
* thread\_start: 스스레드는 기본적으로 만들어지면 (우리가 수정할 방향으로는 우선순위에 따라) ready\_list에 삽입되며, 때가 되면 실행되었다가 yield되면 다시 ready\_list의 끝에 삽입되는 식으로 계속 돌아간다. block과 unblock으로 스레드의 실행을 잠시 정지시킬 수 있으며, 일을 다 끝내면 마지막에는 exit로 삭제된다.

1. System call procedure (for each system call)

동시에 서로 다른 스레드가 접근해서는 안 되는 코드 구역을 임계 구역이라고 하는데, 이러한 임계 구역의 처리를 하기 위한 처리가 동기화이다. 스레드가 임계 구역에 진입하면 lock을 걸고, 임계 구역을 나오면 unlock을 하는데, 이 과정에서 세마포어를 사용한다. 세마포어는 두 개의 원자적 함수로 조작되는 정수 변수로, sema\_down(P)은 임계 구역에 들어갈 때, sema\_up(V)은 임계 구역에서 나올 때 실행된다. 이 때 세마포어의 초기 value는 공유할 수 있는 자원의 총 수로, 세마포어의 value가 0이면 더 이상 사용할 수 있는 자원이 남아있지 않음을 의미한다. 이 경우 스레드는 해당 임계 구역에 접근할 수 없다. 다른 스레드가 사용하고 sema\_up을 해 줄 때까지 기다려야 한다.

1. File system

동시ㅈㅇㅈ