**Pintos Project 1: User Program (1)**

담당 교수 : 문의현 교수님

학번 / 이름 : 20181255 / 김기철

개발 기간 : 2022.10.26 ~ 2022.11.02

1. **개발 목표**

* **해당 프로젝트에서 구현할 내용을 간략히 서술.**

이번 프로젝트는 User Program을 실행시키기 위해 입력받은 명령어에 따라 올바른 시스템 콜이 호출되는 것을 목표로 한다. 세부적으로는 다음과 같은 4가지의 구현이 필요하다.

1. Arguement Passing

입력받은 명령을 Parsing하는 함수를 구현하고, 알맞게 Parsing한 명령들을 Stack에 저장하는 Passing 함수를 구현한다.

2. User Memory Access

사용자가 kernel 영역에 잘못 접근하는 것을 막고 user 영역에만 올바르게 접근할 수 있도록 확인해주는 유효성 검사 함수를 구현한다.

3. System Calls

System call handler 함수에서 입력받은 명령어에 따른 알맞은 시스템 콜이 호출되도록 다양한 시스템 콜 함수들을 구현한다.

4. Additional Implementation

3번에서 기본적인 시스템 콜 함수 이외에 추가로 구현하는 함수로, 피보나치 수를 반환하는 fibonacci 함수와 최댓값을 반환하는 max\_of\_four\_int 함수를 구현한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* **아래 항목을 구현했을 때의 결과를 간략히 서술**

1. Arguement Passing

사용가자가 입력받은 명령어를 공백(” “) 단위로 parsing한 후, parsing한 명령어들을 차례로 stack에 쌓아 저장한다.

2. User Memory Access

사용자가 user 영역에 접근하지 않고 kernel 영역에 잘못 접근하는 경우, 프로그램을 강제종료 시킨다.

3. System Calls

System call handler에서 구현한 다양한 종류의 시스템 콜 함수에 따라 사용자가 입력받은 명령어에 따른 알맞은 시스템 콜이 호출된다.

4. Additional Implementation

3번과 동일하다.

* 1. **개발 내용**
* **아래 항목의 내용만 서술 (기타 내용은 서술하지 않아도 됨.)**

1. Arguement Passing

명령어를 다른 문자열에 copy한 후, copy한 문자열을 parsing 함수를 통해 공백 단위로 쪼갠다. 쪼갠 명령어 중 첫 번째 원소가 파일이름이 되고 stack으로 넘겨줄 때는 4byte만큼씩 할당해준다.

2. User Memory Access

is\_user\_vaddr 함수를 통해 kernel 영역에 접근하지 못하도록 하고, 만약 접근한다면 exit 함수를 호출하여 강제종료시킨다.

3. System Calls

halt exit, exec, wait, read, write 함수들을 구현하고 syscall\_handler 함수에서 호출되도록 한다.

4. Additional Implementation

피보나치 함수와 최댓값 함수를 구현하고 syscall\_handler 함수에서 호출되도록 한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* **II. A.의 개발 범위를 포함하여 구현 내용에 대한 일정 작성**

10.26 ~ 10.29 : 온라인강의 수강 및 과제 내용 파악

10.29 ~ 11.1 : Argument Passing, User Memoery Access, System Calls, Additional Implementation 기능 함수 구현 및 테스트케이스 평가

11.2 : 최종 점검 및 보고서 작성

* 1. **개발 방법**
* **II. B.의 개발 내용을 구현하기 위해 어느 소스코드에 어떤 요소를 추가 또는 수정할 것인지 설명. (함수, 구조체 등의 구현이나 수정을 서술)**

1. Arguement Passing

process.c 파일에 parsing을 위한 parsing 함수와 passing을 위한 push\_stack 함수를 구현 process.h 파일에도 2개의 함수를 선언. 선언한 함수를 process.c 파일의 process\_execute 함수와 load 함수에 추가.

2. User Memory Access

userprog/syscall.c 파일의 syscall\_handler 함수 내에 시스템 콜 함수들을 호출하기 전 유효성 검사를 위한 is\_user\_vaddr 함수를 추가.

exception.c 파일의 page\_fault 함수 내에 유효성 검사를 위한 is\_user\_vaddr 함수를 추가.

3. System Calls

userprog/syscall.h 파일에 구현하고자 하는 시스템 콜 함수들인 halt exit, exec, wait, read, write 함수들을 선언하고, userprog/syscall.c 파일의 syscall\_hadnler 함수 내에 각각의 함수를 구현.

thread.h 파일의 thread 구조체 내에 wait와 exec을 위한 멤버변수들을 추가.

thread.c 파일의 init\_thread 함수 내에 위에 추가한 멤버변수들을 초기화.

4. Additional Implementation

syscall\_nr.h 파일의 열거형 구조체 내에 SYS\_FIBO, SYS\_MAX를 추가.

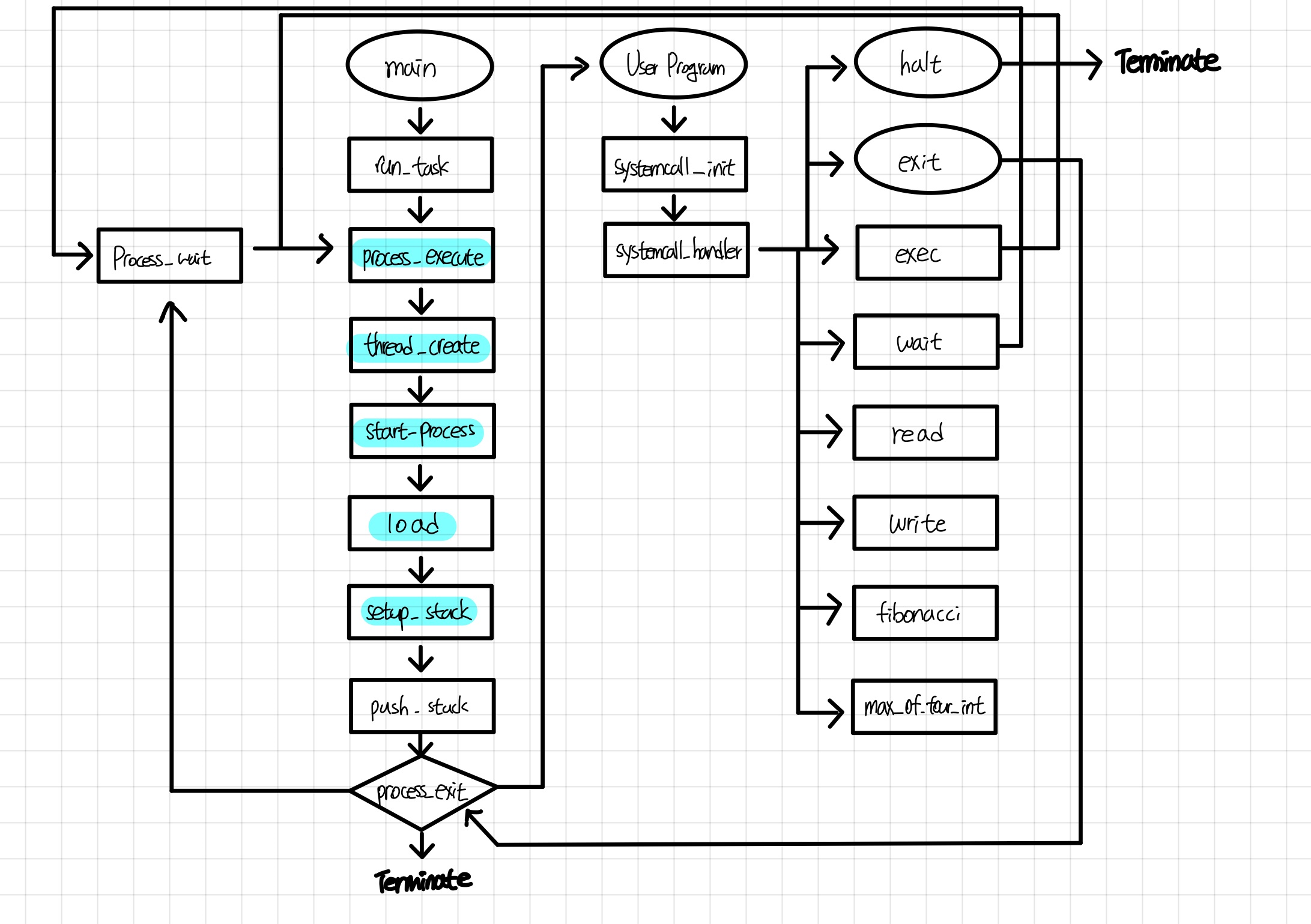
userprog/syscall.h 파일에 fibonacci 함수와 max\_of\_four\_int 함수를 선언하고, userprog/syscall.c 파일의 syscall\_hadnler 함수 내에 각각의 함수를 구현.

lib/user/syscall.h 파일에 fibonacci 함수와 max\_of\_four\_int 함수를 선언하고, lib/user/syscall.c 파일에 syscall4 함수를 정의 및 위의 함수를 구현.

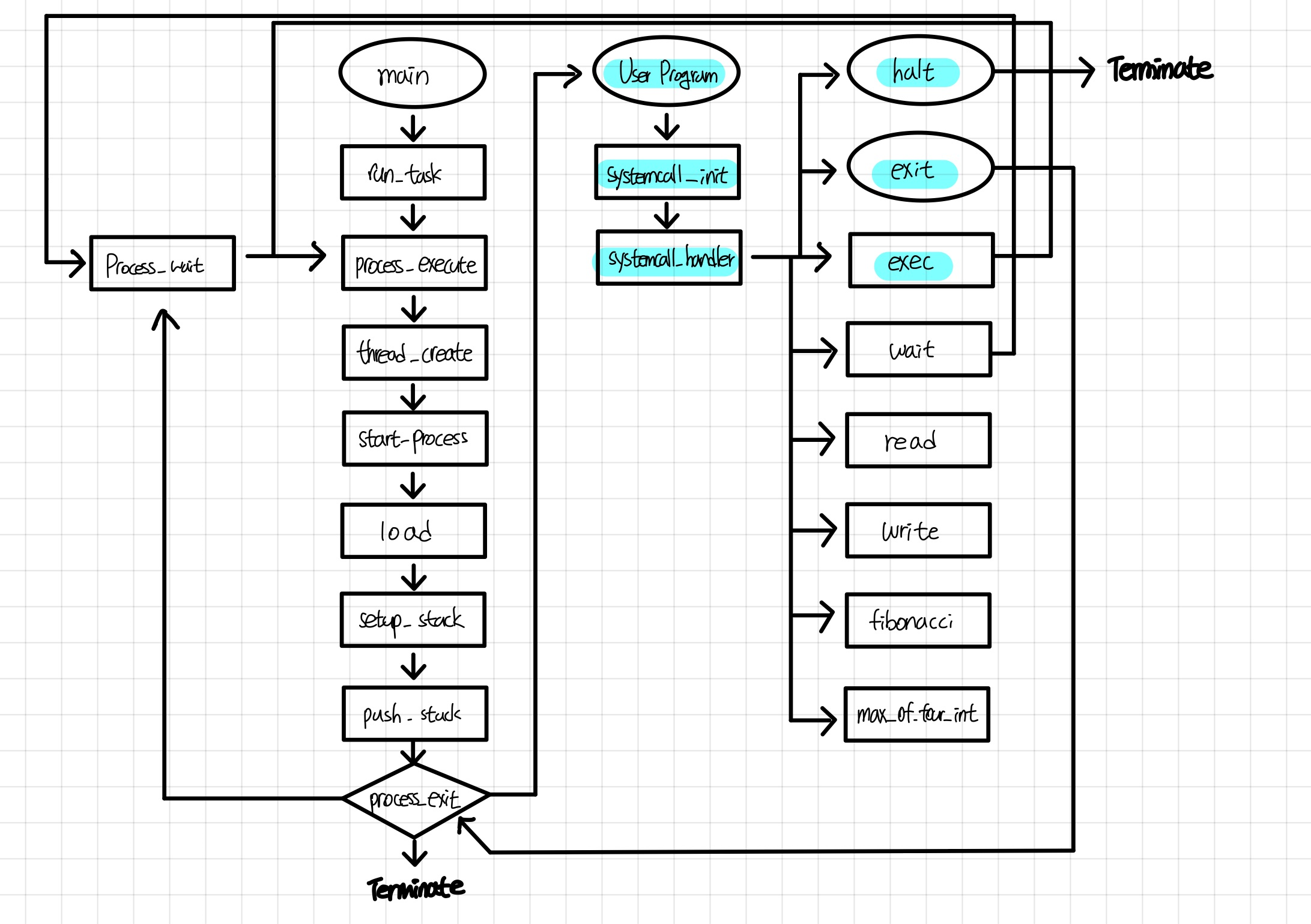
1. **연구 결과**
   1. **Flow Chart**

* **II. B. 개발 내용에 대한 Flow Chart를 작성**

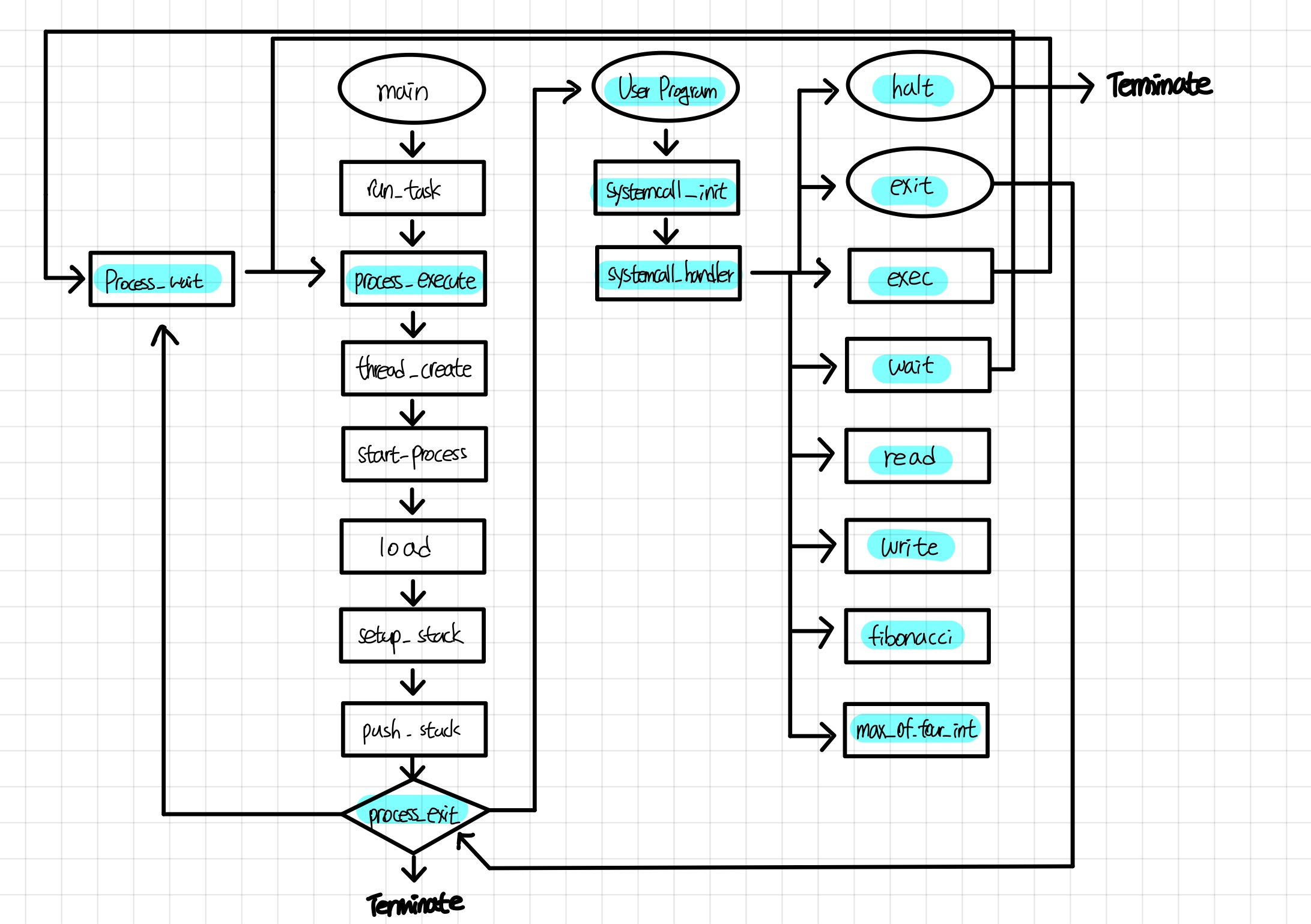
1. Argument Passing



1. User Memory Access



1. System Calls



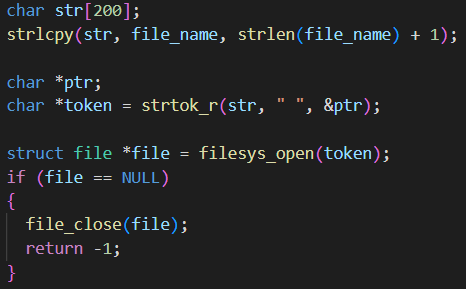
* 1. **제작 내용**
* **II. B. 개발 내용의 실질적인 구현에 대해 코드 관점에서 작성.**
* **구현에 있어 Pintos에 내장된 라이브러리나 자체 제작한 함수를 사용한 경우 이에 대해서도 설명.**
* **개발상 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결책에 대해 설명.**

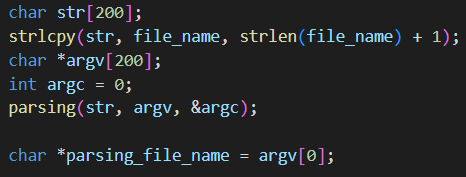
1. Argument Passing

1) process.h



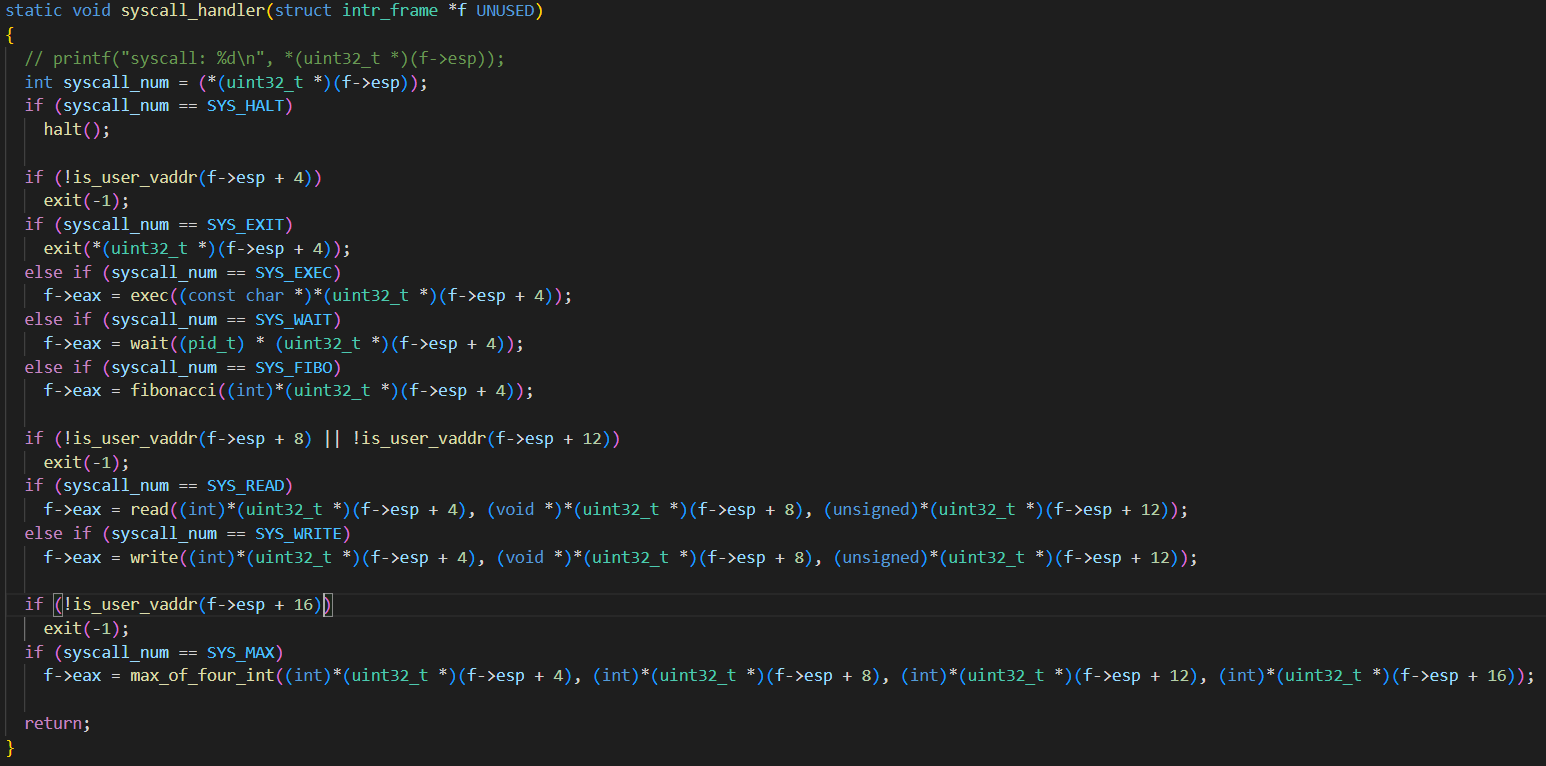
2) process.c





1. User Memory Access

1) userprog/syscall.c



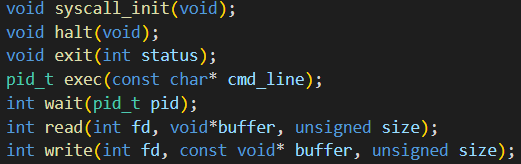
2) exception.c



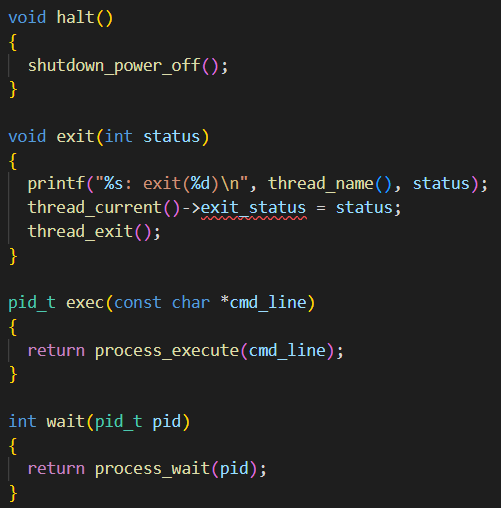
1. System Calls

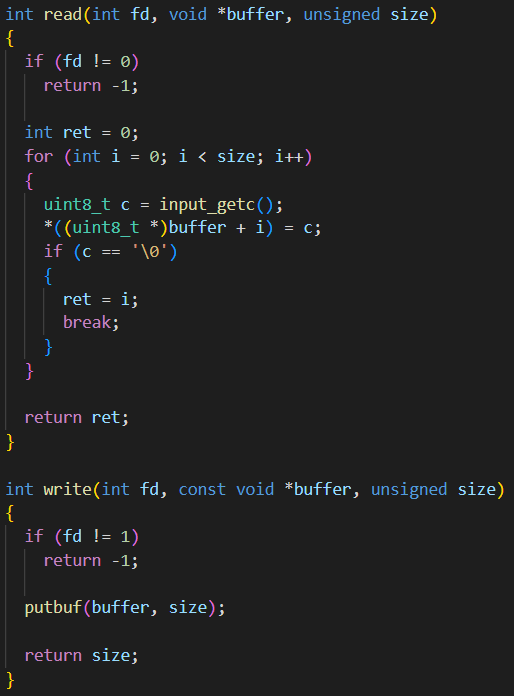
* **이번 프로젝트에서 개발한 시스템 콜을 구현 관점에서 상세히 서술.**

1) userprog/syscall.h

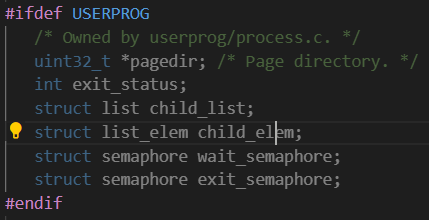


2) userprog/syscall.c

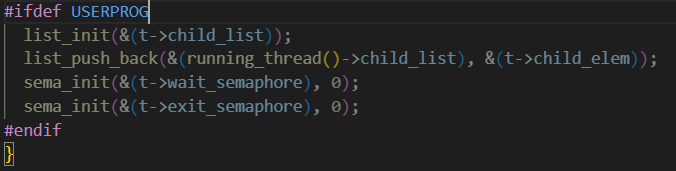




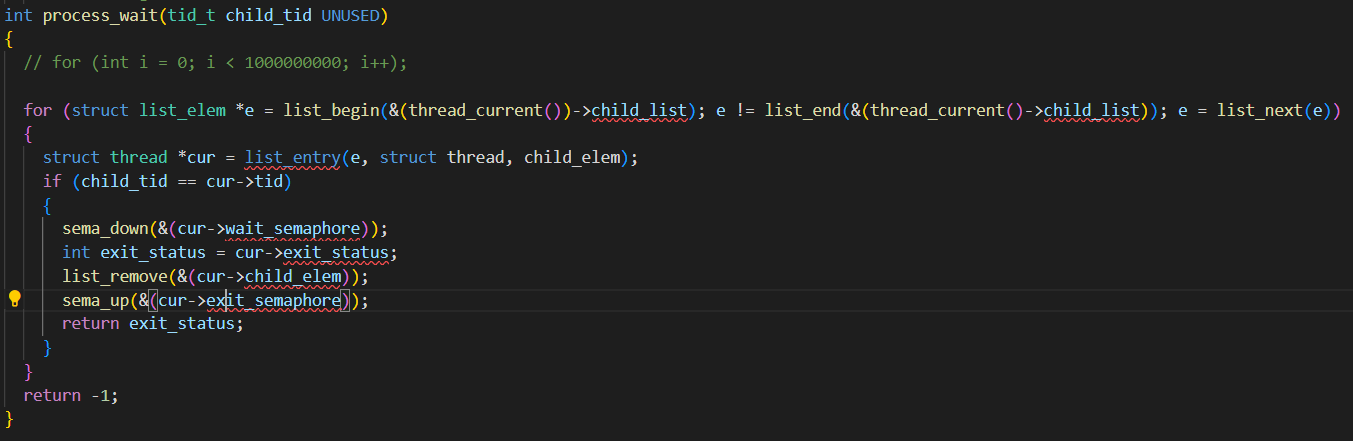
3) thread.h



4) thread.c



5) process.c





1. Additional System calls

* **새로운 시스템 콜(fibonacci, max\_of\_four\_int)을 구현하기 위해 수정하거나 작성한 코드에 대해 서술**

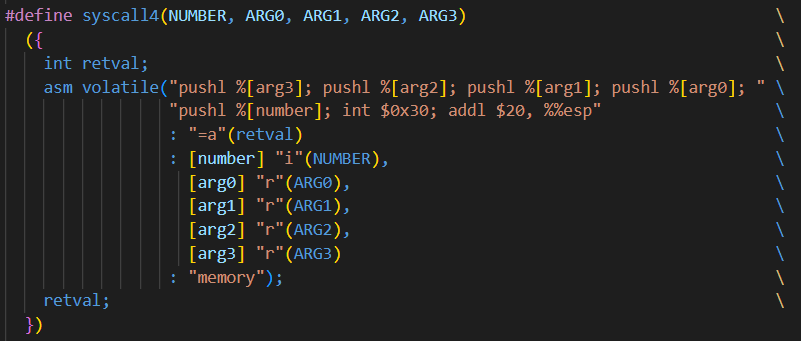
1) syscall-nr.h

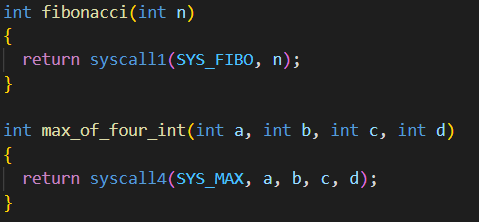


2) lib/user/syscall.h



3) lib/user/syscall.c

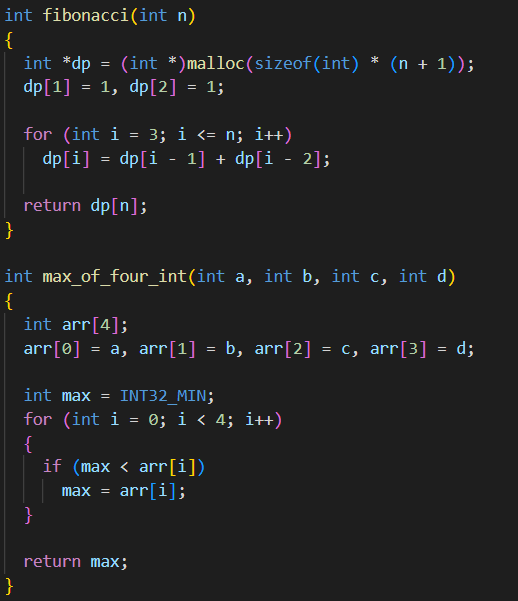




4) userprog/syscall.h

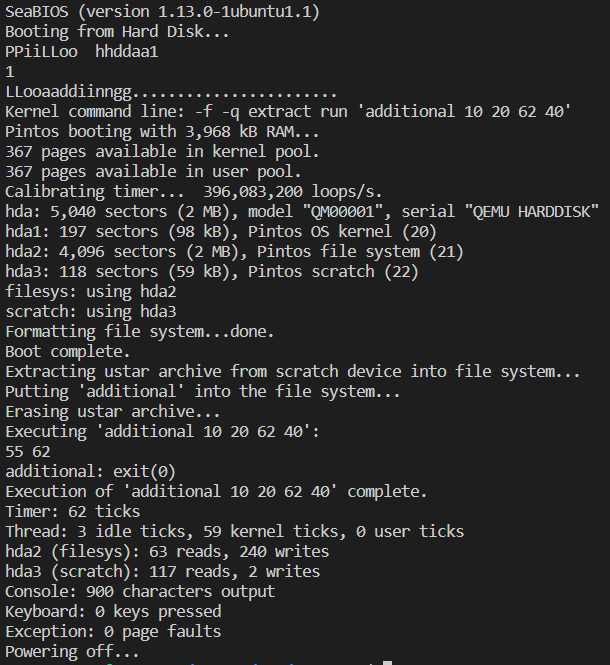


5) userprog/syscall.c



* 1. **시험 및 평가 내용**
* **fibonacci 및 max\_of\_four\_int 시스템 콜 수행 결과를 캡처하여 첨부.**

pintos --filesys-size=2 -p ../examples/additional -a additional -- -f -q run 'additional 10 20 62 40' 수행 결과

* ****

피보나치 수열의 10번째 값인 55와 10,20,62,40 중 최댓값인 62가 출력된 것을 확인할 수 있다.