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

**(설계 프로젝트 수행 결과)**

과목 명 : 운영체제

담당 교수 : 김영재

조 / 조원 : 66조 / 류인석(20121586), 조형준(20141086)

개발 기간 : 2019.10.20 ~ 2019.11.03

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

**제출일 : 2019.11.03**

**참여 조원 : 류인석(20121586), 조형준(20141086)**

1. **개발 목표**

* **본 프로젝트에서는 Pintos 운영체제 상에서 User Program을 수행하기 위한 System Call을 구현하여야 하며, Argument Passing을 구현하여 입력 받은 커맨드를 적절하게 파싱하여 argument로 넘겨주어 적절한 프로세스를 실행하도록 한다.**

1. **개발 범위 및 내용**
   1. **개발 범위**

* **System call, System call handler : User Program 구동 시 필요로 하는 kernel functionality와, 이에 접근하기 위한 System call handler를 syscall.c에서 구현한다.**
* **Argument passing : 입력된 커맨드를 적절한 기준으로 나누어 구동시킬 User program의 Argument로 passing을 process.c에서 구현한다.**
* **User stack : Argument passing으로 넘겨받은 Argument를 스택에 쌓아 저장하고 접근할 수 있게 함.**
  1. **개발 내용**
* **System call : user program 구동 시 필요로 하는 기본적인 kernel function에 대한 정의. User memory access기능과 system call interrupt를 이용하여 syscall.c에서 syscall\_handler 함수를 정의하며, 각 경우에 필요한 system call을 선택하여 호출한다**
  + **Halt : shutdown\_power\_off()를 이용해 pintos 프로그램을 종료한다.**
  + **Exit : 현재 진행중인 user program을 종료하고 kernel에 user program의 status를 반환한다.**
  + **Exec : 파라미터로 주어진 이름의 프로그램을 run하고 pid를 return한다.**
  + **Wait : 자식 프로세스가 종료될 때까지 부모 프로세스를 대기시킨다.**
  + **Read : buffer에 내용을 입력 받는다.**
  + **Write : buffer의 내용을 putbuf를 이용하여 출력한다.**
  + **Fibonacci : 입력 받은 k번째 피보나치 수를 출력한다.**
  + **Sum of four int : 입력 받은 4개의 정수의 합을 출력한다.**
* **Argument passing : User program 실행 시 입력 받은 command line을 argument로 변환하여 user stack에 저장하고, user program에 argument를 전달 할 수 있도록 한다.**
* **Thread : 스레드 간의 관계를 고려하여, 부모로부터 파생된 자식 스레드가 실행이 완료되기 전에 부모 스레드가 종료되지 않도록 대기시켜야 하며, 이를 위해 부모 스레드가 자식 스레드의 status를 알 수 있도록 해야 한다.**

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

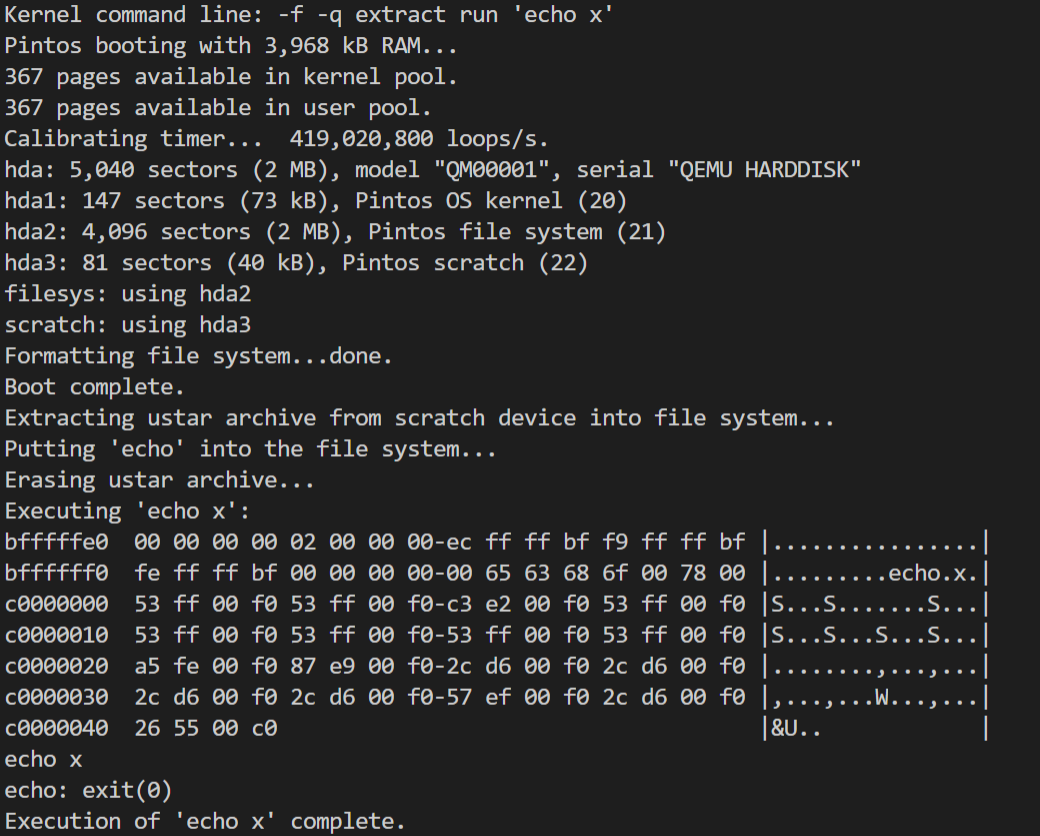
* **10.20 ~ 10.22 : 매뉴얼 이해**
  + **프로젝트 구현의 이해도를 높이기 위하여 핀토스 매뉴얼의 정독 및 이해를 높이고자 함.**
* **10.23 ~ 10.25 : 개발 역할 분담 및 코드 이해**
  + **프로젝트 소스의 전체적인 흐름의 이해**
  + **업무의 효율성을 위하여 구현할 기능의 분배를 논의**
* **10.26 ~ 10.27 : argument passing 및 user stack 구현**
  + **Process.c에서 command line으로 입력된 argument를 나눠 user stack에 쌓고 제대로 저장이 되는지 확인**
* **10.28 ~ 10.31 : System call 및 thread 구현**
  + **User program 실행 시 필요한 system call의 기능을 구현**
  + **Thread의 process\_wait를 중점적으로 연구하여 스레드 간의 동기화 (synchronization) 기능 구현에 초점**
* **11.1 ~ 11.2 : 추가 기능 구현 및 test case check 및 디버깅**
  + **피보나치, 4개 정수 합 함수를 구현**
  + **채점과정 확인을 통한 세부 오류 수정**
* **11.3 : 보고서 작성**
  1. **개발 방법**
* **개발환경 : putty의 불편성으로 인해 vs code를 통해 원격 가상환경으로 개발.**
* **깃허브 협업 : 각자 개발한 코드의 합병이 용이하기 때문에 git hub로 협업 개발.**
  1. **연구원 역할 분담**
* **조형준 : argument passing 구현, system call 구현, 보고서 작성**
* **류인석 : user stack 구현, thread 구현, 디버깅 및 테스트**

1. **연구 결과**
   1. **합성 내용**

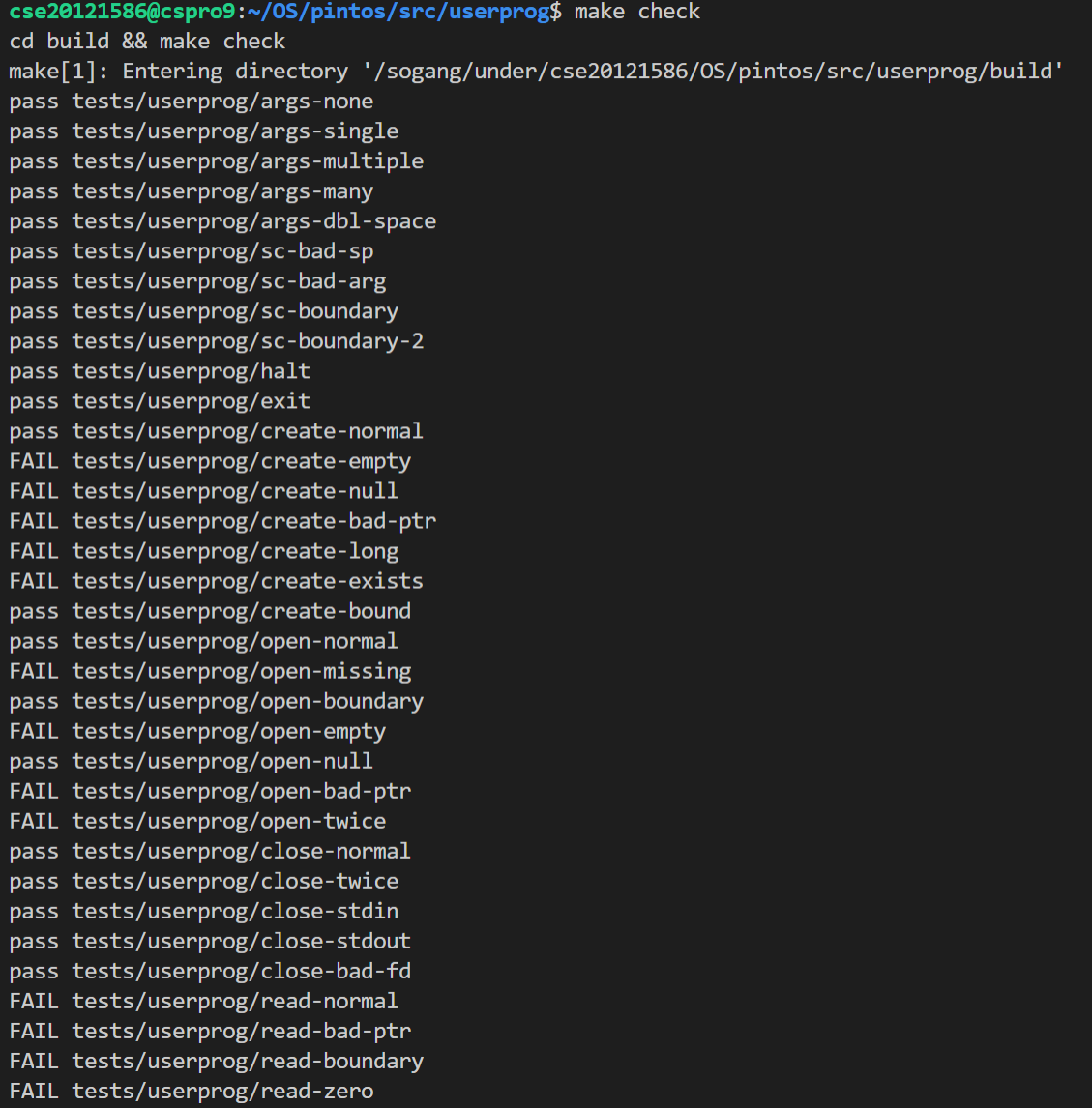
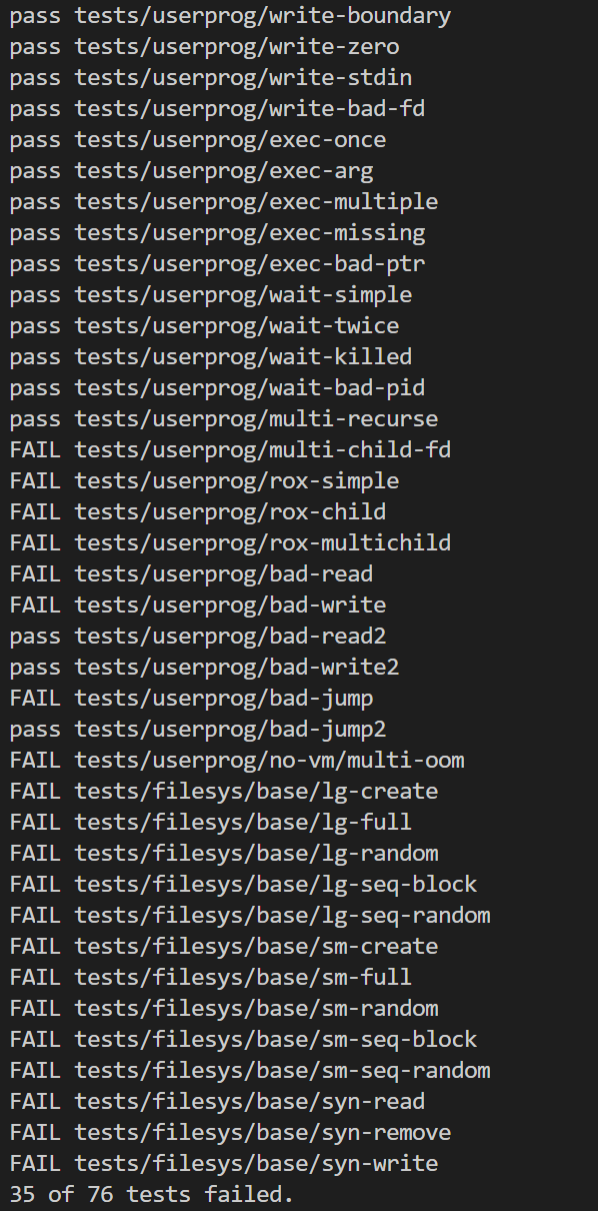
* **Init.c의 main함수에서 입력된 command line인 argv를 인자로 받아 run\_actions이 실행된다. Run\_actions 내에서는 run\_task를 통하여 process를 생성한다. Run\_task 내에서는 process\_execute에서 반환된 tid를 process\_wait에 전달하여 child process가 실행되는 동안 parent process를 대기시키도록 한다.**

**생성된 thread는 load를 통해 프로세스를 적재하여 실행이 가능하도록 해준다. 이 과정에서 필요한 시스템 콜을 적절히 호출하여 운영체제에서 유저프로그램이 작동하도록 한다**

* 1. **제작 내용**
* **Process Termination Messages**
  + **자료구조 : struct thread / int exit\_status : 쓰레드 상태 저장**
  + **알고리즘 : 유저 프로그램이 적재된 스레드에서 exit을 호출하여 exit\_status의 값을 확인하고 정상적으로 종료되었는지, 혹은 kernel에 의해 비정상적으로 종료되었는지 등을 확인할 수 있다.**
* **Argument Passing**
  + **자료구조 : char\* file\_name, void\*\* esp**
  + **알고리즘 : file\_name으로 넘어온 command line을 사용자 정의 함수 parse\_filename을 통해 strtok\_r을 이용한 파싱과정을 거쳐 esp가 가리키고 있는 스택에 argument로 하나씩 쌓으며 저장한다**



* **Thread(synchronization)**
  + **자료구조 :** 
    - **struct list child : 자식 쓰레드**
    - **struct list\_elem child\_elem : 자식 쓰레드 간을 연결하는 포인터**
    - **struct semaphore current\_lock : 현재 실행중인 스레드의 실행/중단 상태**
    - **struct semaphore temp\_lock : 메모리 삭제 대기를 위한 실행/중단 상태**
  + **알고리즘 : process\_wait는 argument로 받은 child\_tid가 죽을 때까지 기다렸다 죽으면 해당 쓰레드의 exit status를 리턴한다. semaphore value를 참조하여 process\_wait에서 child process를 기다린다. Sema\_down으로 대기, child 종료시 sema\_up으로 control 가능.**
* **System Calls**
  + **Syscall.c에서 switch문을 이용하여 userprogram을 실행하는데 필요한 syscall number의 경우에 따라 적절한 함수를 실행시켜주었다.**
  + **syscall에서 요구되는 argument는 user stack에서 참조하며, 사용자 정의 함수를 적절하게 사용하여 해당 시스템 콜이 발생할 수 있게 해주었다.**
* **Additional System Calls**
  + **자료구조 : Fibonacci, sum**
  + **알고리즘 : 제공된 pintos code는 argument를 최대 3개까지만 받을 수 있어서 lib/user/syscall.c에서 sum 실행에 필요한 syscall4를 구현하여 argument를 4개까지 받을 수 있게 코드를 추가하였다.**
  1. **시험 및 평가 내용**



1. **기타**
   1. **연구 조원 기여도**

* **조형준(45%) 류인석(55%)**
  1. **소감**
* **OS에대한 기본지식을 구현 할수 있는 기회를 가지게 되었다. 이번 프로젝트에서 User Program 실행과 System call 을 구현 하면서 더욱더 구조적으로 깊은 이해를 했다. 사실 학부생으로 여러 파일을 동시에 보면서 이들이 어떻게 구조적으로 돌아가는 지 볼 기회가 적었는데, 이번 프로젝트를 통해 이를 연습 했다. 또한 코드 내에서도 asm 코드들이 잠깐잠깐 있었는데, 이를 보면서 asssembly 언어가 어떻게 돌아가는지 훑어볼 수 있는 기회가 되었다.**
* **이번 프로젝트에서 가장 크게 배운 교훈은 명세서를 중시 해야 한다는 점이다. 사실 교수님꼐서 maual 을 천천히 읽어보는것이 중요하다고 했을때 "어떻게 100페이지가 되가는 명세서를 언제 다 읽지?"라는 생각을 했다. 그래도 교수님의 말씀을 따라 해봐야 겠다는 생각에 처음 3일간 다른건 하지 않고 명세서를 같이 정독하며 서로 이해를 도왔다. 이가 도움이 이렇게 크게 될 줄 몰랐다. 사실, README file 을 읽거나 명세서 를 작성하는 것이 개인적으로 가장 귀찮고 어려운 일이라 생각 했다. 그러나 이번 프로젝트를 하면서 명세서를 읽지 않았더라면 pintos 를 구현 하는데 훨씬 오랜기간이 걸렸을 것이다. 나중에 개발을 할 때도 남들에게 나의 코드를 보여줄때, 명확한 명세서를 만들수 있다는 점도 정말 중요하다는 생각을했다.**