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

담당 교수 : 김영재 교수님

학번: 20202106

컴퓨터공학과

OSHIMA ASUKA

1. **개발 목표**

* 아래 pintos를 활용해서 os, system call의 역할에 대해 이해한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* **아래 항목을 구현했을 때의 결과를 간략히 서술**

1. Argument Passing

아래 사진과 같이 Argument passing가 성공했고 Echo x가 ASCII코드로 표시되어 있는 것을 확인할 수 있다.

テキスト

自動的に生成された説明

1. User Memory Access

　User memory에 접근을 관리되어 무효한 포인터들을 체크할 수 있게 되었다.

1. System Calls

만들었던 system call이 user program에서 작동하는 것을 확인했다.

テキスト

自動的に生成された説明

* 1. **개발 내용**
* **아래 항목의 내용만 서술 (기타 내용은 서술하지 않아도 됨.)**
* *Argument Passing*
  + *커널 내 스택에 argument를 쌓는 과정 설명*

파일명은 변경하지 않기 위해 파일명을 카피하고 복사한 파일명에서 argument들을 argv[]에 저장한다. Argument의 개수는 argc로 카운트한다. Argument를 반대순서로 stack에 쌓였다. 이러므로 함수 안에서 순서대로 접근할 수 있기 때문이다. Argument 리스트의 끝을 인식하기 위해 위해 NULL pointer를 push했다. Argv의 요소들의 주소과 그 위치를 가리키는 주소 그리고 argc의 개수도 스택의 저장한다.

* User Memory Access
  + Pintos 상에서의 invalid memory access 개념을 간략히 설명

invalid memory access란 access가 허용되지 않은 메모리 영역에 접근하려고 하는 것을 의미한다. User process가 kernel mode에 접근하려고 하는 것은 허용되지 않은 동작이며, user memory가 kernel memory에 접근하려고 하는 것도 invalid memory access의 일부다. 또 할당되지 않은 메모리에 접근하는 것, 다른 process의 메모리에 접근하려고 하는 것도 invalid memory access고, 이러한 invalid memory access를 시도하면 security가 위협해지기 때문에 invalid memory access를 막하야 한다.

* memory access를 어떻게 막을 것인지 설명

system call가 user memory에 접근하기 전에 Syscall.c에서 user pointer가 접근 가능한 위치를 가리키는지 확인해야 한다. 즉 null아닌 포인터이고 user address를 가리키는지, 그리고 virtual 공강에 mapping되어 있는지를 확인한다. Page\_fault()함수에서 User pointer가 PHYS\_BASE이하를 가리키고 있는지를 확인하고 그 포인터를 참조한다. 포인터가 invalid인 경우 page fault가 발생한다.

* System Calls
  + 시스템 콜의 필요성에 대한 간략한 설명

프로그램이 커넬에 접근하기 위한 interface를 제공한다. OS는 시스템 콜을 이용해서 데이터를 load하거니 프로그램을 실행한다. 또 물리적인 메모리 공간에 접근을 시도한다.

* + 이번 프로젝트에서 개발할 시스템 콜에 대한 간략한 설명 (하나의 시스템 콜 당 최대 3문장으로 간략히 설명; 3문장을 넘길 정도로 길게 작성하지 말 것)

Halt(): pintos 종료

Exit() : 현제 실행중인 프로그램을 종료하고 커널에 현재 상태를 리턴한다.

Wait():

Child process가 종료될 때 까지 parent process를 block상태로 한다. Child process가 종료되면 parent process에 종료상태를 리턴한다.

Exec() :

Child process를 만들다. 지정된 프로그램을 로드하고 실행한다.

Read():

표준 입력부터 입력을 읽은다.

Write():

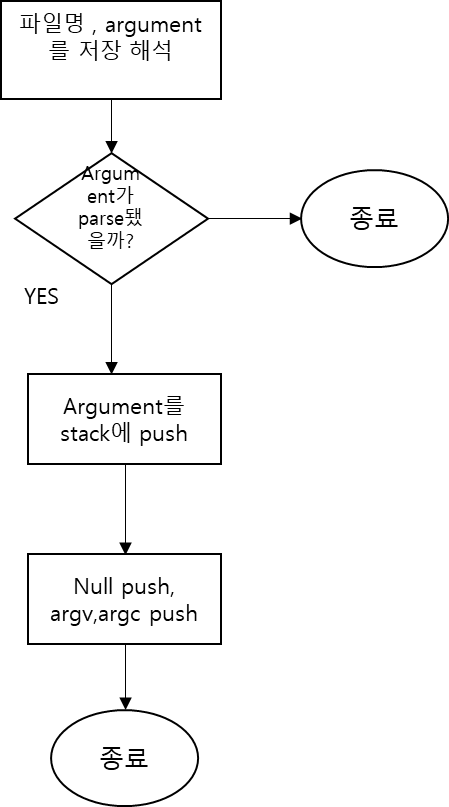
표준 입력에서 출력한다.

* + 유저 레벨에서 시스템 콜 API를 호출한 이후 커널을 거쳐 다시 유저 레벨로 돌아올 때까지 각 요소를 설명

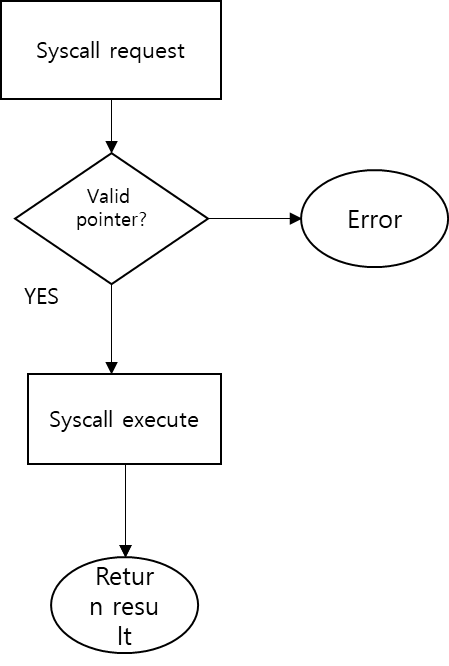
1. **추진 일정 및 개발 방법**
   1. **추진 일정**

프로젝트 pintos manual을 읽어 pintos 구성이 어떻게 되어 있는지 이해한다. 시스템콜을 이해하고 자료의 순서에 따라 실행한다.

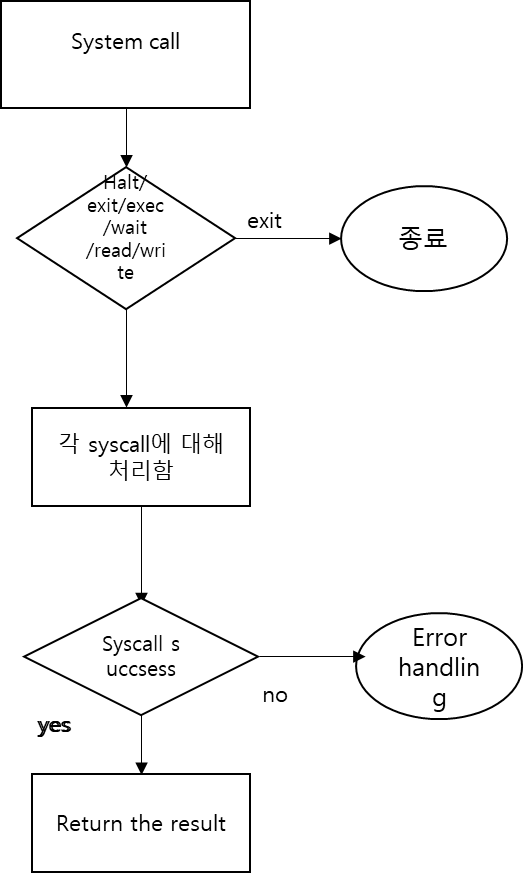
1. **연구 결과**
   1. **Flow Chart**
2. Argument Passing



1. User Memory Access



1. System Calls



* 1. **제작 내용**

1. Argument Passing

Argument passing를 하는 과정에서 null를 끝에 넣지 않아 argc test에서 결과가 잘 안 나왔다.

1. User Memory Access

무효한 access를 맡기 위해 is\_user\_vaddr()를 이용해서 syscall안에서 위반 access인지를 확인한다. 또 exceptorn.c에 pagefalutl함수에서 is\_kernel\_vaddr()를 이용했다

1. System Calls

Wait가 exec를 만드는 과정에서 child가 잘 만들지 못해 힘들었다. List를 활요하는 데 pagefalut가 발생했다.