**Pintos Project 4: Virtual Memory**

담당 교수 : 박성용 교수님

이름 / 학번 : 나호영/20211531

개발 기간 : 11/20 ~ 12/8

1. **개발 목표**Stack Growth와 Second Chance 기법을 이용하는 Disk Swap 기능을 구현하여 가상 메모리(Virtual Memory)를 개발한다.
2. **개발 범위 및 내용**
   1. **개발 범위**
   2. **Page Table & Page Fault Handler**  
      페이지 폴트가 발생하였을 때 원활히 처리하기 위해 기존 페이지 테이블에 저장하지 못하는 정보들을 담는 SPT(Supplement Page Table)를 구현한다.
   3. **Disk Swap**  
      페이지 폴트가 발생하여 물리 메모리에 사용하기 위한 페이지를 위해 프레임을 할당받으려고 할 때, 사용하고 있는 물리 메모리에 더 이상 할당할 수 없는 상태가 되면 하드디스크의 스왑 공간에 지금 사용 중인 페이지 중 하나를 골라 스왑 아웃시키고 필요한 페이지를 스왑 인시킨다. 이 때, Second Chance 기법을 사용하여 프레임들을 순회하며 accessed bit를 확인해 최초 발견 시 1이면 0으로 바꾸고 0이면 해당 페이지를 evict로 선정, 즉 스왑 아웃시킬 페이지를 선택하는 policy를 사용한다. 또한 스왑 테이블을 구현하여 스왑 디스크를 관리할 수 있게 한다.
   4. **Stack Growth**  
      스택이 할당되었던 공간보다 더 확장된 공간이 필요하다면 해당 스택 영역을 위해 페이지를 추가하여 할당해준다. 이 때, 스택이 확장될 수 있는 메모리는 8MB로 제한한다. 할당되었던 스택의 범위를 벗어난 공간에 대한 접근 요청이 발생하면, 해당 요청이 스택의 범위를 벗어나 스택의 확장을 요청하는 것인지 아니면 invalid access인지 확인해야 한다.
   5. **개발 내용**
   6. **Page fault가 발생하는 이유와 이를 handling하는 전반적인 과정을 서술**  
      물리 메모리에 올라가있지 않은 페이지에 접근하려고 할 때 페이지 폴트가 발생한다. 하지만 페이지 폴트가 발생하였다고 하여 모두 invalid access와 같은 오류가 발생하지 않는다. 정상적인 경우, 페이지 폴트가 발생하였을 때 먼저 물리 메모리에 필요한 페이지를 로드한다. 이후, 물리 메모리 공간이 부족해지면 위에서 설명한 Second Chance와 같은 다양한 정책에 따라 스왑 아웃시킬 페이지를 선정하고 스왑 디스크로 스왑 아웃시킨 후, 물리 메모리에 필요한 페이지를 로드한다. 또한, 구현할 SPT를 활용하여 해당 페이지가 디스크에 있는지 스왑 공간에 있는지 등을 확인하여 페이지 폴트 핸들링을 진행한다.
   7. **Disk swap 발생 시 사용한 page replacement algorithm에 대해 서술**  
      Second Chance 알고리즘을 사용하여 구현한다. Second Chance 알고리즘은 프레임들을 순회하며 accessed bit를 확인하여 0이라면 해당 프레임을 선정하여 디스크 스왑을 처리한다. 이 때, 최초 발견 시 1이면 0으로 한번 바꾸고 다음 프레임으로 넘어가는 과정이 존재한다. 알고리즘을 위해 순환 큐를 사용하여 구현한다.
   8. **Stack growth 구현 시 stack 확장 여부를 판단할 수 있는 방법에 대해 서술**  
      스택은 메모리 상에서 아래에서 위로 확장되며, 이는 8MB로 제한되어 있다. 따라서, valid한 스택 영역인지 확인하기 위해 가상 커널 메모리 공간이 시작되는 지점을 의미하는 PHYS\_BASE를 이용하여 페이지 폴트가 발생한 주소가 PHYS\_BASE - 8MB 부터 PHYS\_BASE 사이에 있는지 확인한다. 또한 f->esp - 32 (이전 스택 프레임의 끝) 보다 페이지 폴트가 일어난 주소가 더 큰지 확인해야 한다. 핀토스에서는 레지스터 값을 스택에 푸쉬할 때 32byte가 푸쉬되는데, 페이지 폴트가 일어난 주소가 이전 스택 프레임의 끝보다 더 작은 메모리 주소값을 가진다면 이는 valid한 스택 확장 요청이 아니기 때문이다.
3. **추진 일정 및 개발 방법**
   1. **추진 일정**

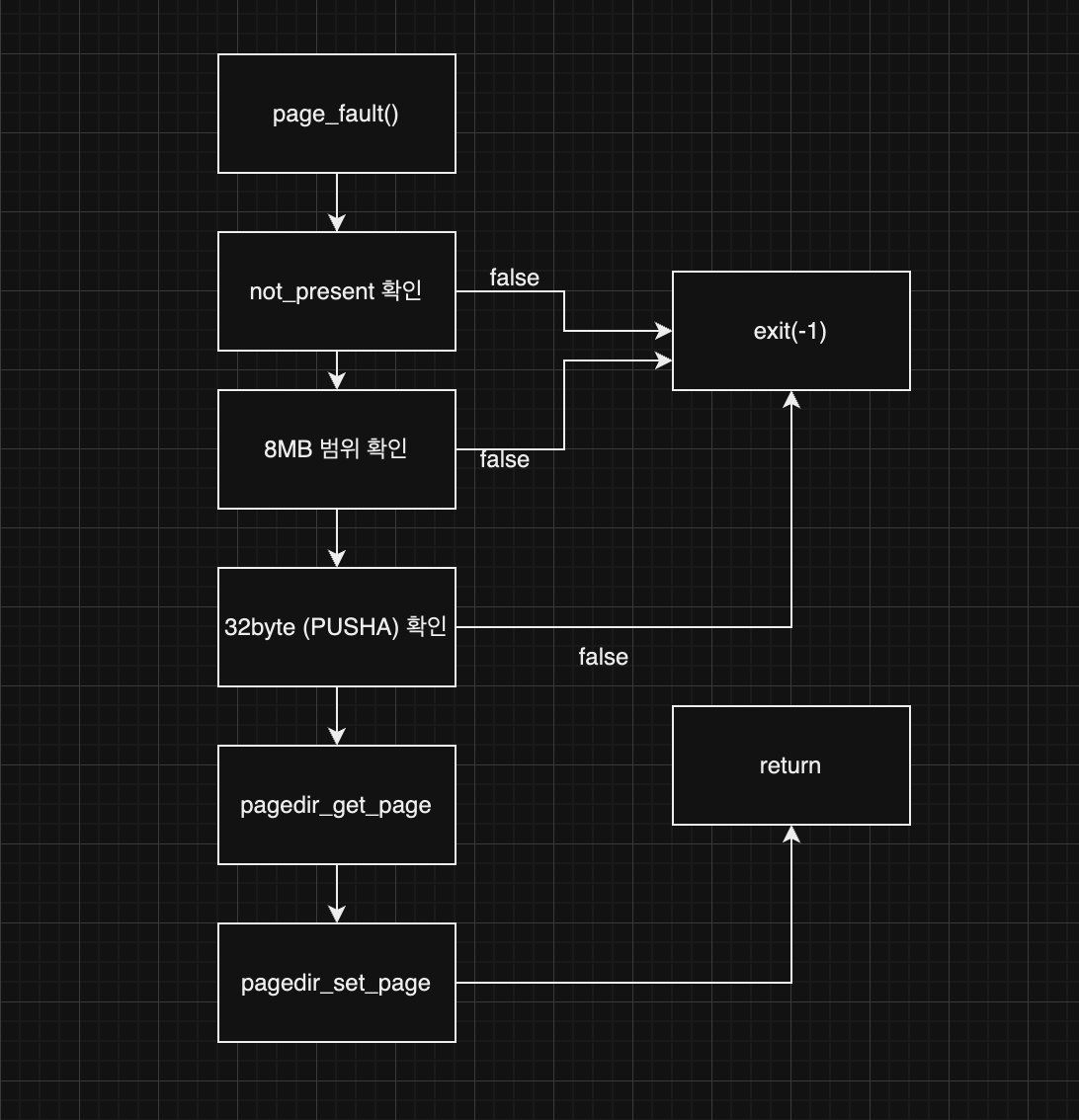
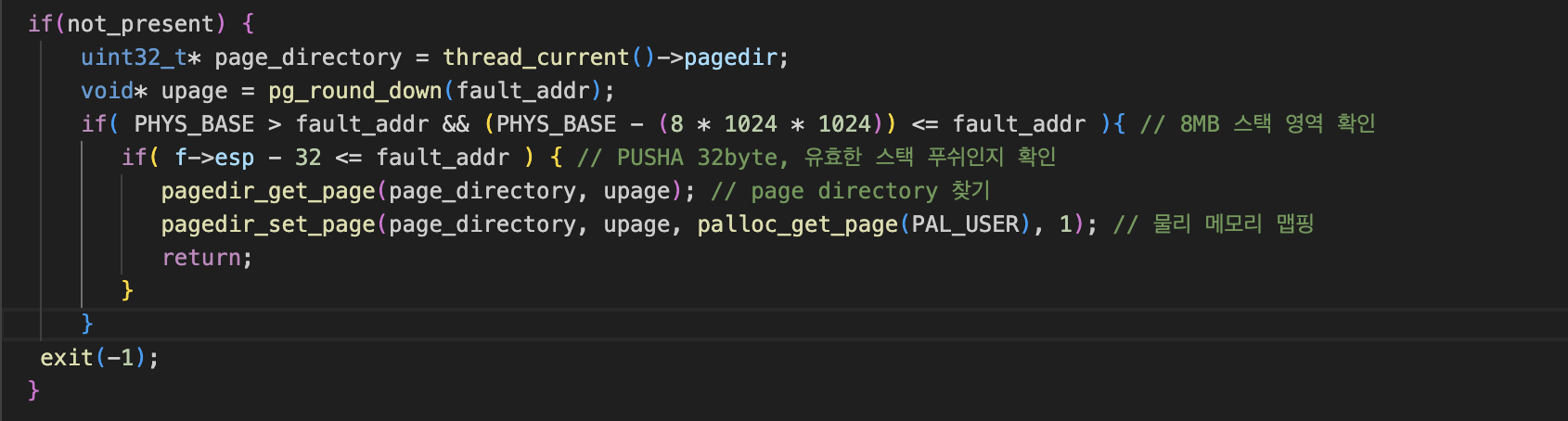
project 4 강의 수강 및 설계 : 11/20 ~ 11/22

Stack Growth 구현 : 11/23 ~ 11/26

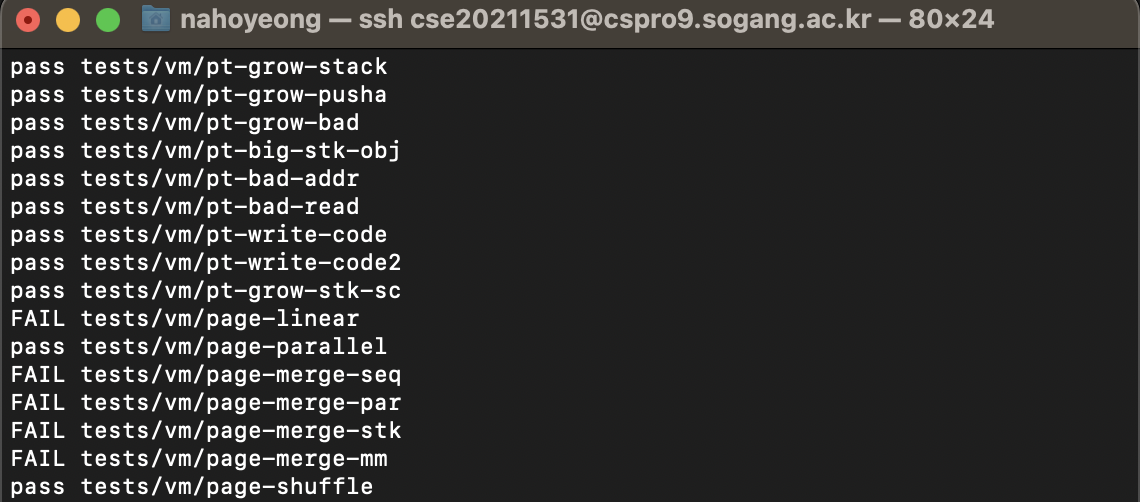
SPT, 페이지 폴트 핸들링 구현 : 11/27 ~ 12/05

디스크 스왑 구현 및 보고서 작성 : 12/06 ~ 12/08

* 1. **개발 방법  
       
     (1) Page Table**  
     먼저 현재는 비어있는 vm 디렉토리에 핀토스 메뉴얼에 나와있는 가이드처럼 page.c / page.h 파일을 만들고 SPT를 위한 spt\_entry 구조체를 선언한다. 해쉬 테이블 자료구조를 이용하여 SPT를 구현할 것이기 때문에 필요한 힘수들을 정의해야한다. spt를 초기화하는 spt\_init, spt\_entry를 추가하는 spt\_insert, 가상 주소를 받아 해당하는 spt\_entry를 찾는 spt\_lookup, spt\_entry를 제거하는 spt\_remove, spt를 제거하는 spt\_destory 함수를 작성한다. 그 후, userprog/process.c 의 load 함수에서 이를 초기화하는 코드를 추가해야 한다.  
       
     **(2) Disk Swap**  
     Second Chance 기법을 이용하여 디스크 스왑을 구현하기 위해 vm 디렉토리에서 swap.c /swap.h 파일을 만든다. 필요한 구조체인 swap\_block, 비트맵 자료구조를 활용하여 스왑 공간을 확인하기 위한 swap\_bitmap, 동기화 문제를 처리하기 위한 swap\_lock을 static으로 선언하여 준다. 또한 디스크 스왑을 초기화하기 위한 swap\_init 함수, 페이지를 스왑 아웃하기 위한 swap\_out, 페이지를 스왑 인하기 위한 swap\_in, 스왑 공간을 해제하기 위한 swap\_free함수를 추가한다.  
       
     **(3) Stack Growth**   
     페이지 폴트가 발생할 때, 스택 영역 확장을 체크하기 위해 userprog/exception.c 에 존재하는 page\_fault 함수에 PHYS\_BASE - 8MB 부터 PHYS\_BASE에 fault\_addr이 존재하는지 확인하고 f->esp - 32 (이전 스택 프레임의 끝) 보다 fault\_addr이 더 큰지 조건문을 통하여 순서대로 확인한 후, 스택 영역에 필요한 페이지를 할당한다. 현재 쓰레드의 pagedir 멤버를 확인하여 페이지 디렉토리를 얻고, pg\_round\_down 함수를 사용하여 해당 주소의 페이지 시작 주소를 받은 후 palloc\_get\_page를 이용하여 물리 메모리에 할당해야한다.

1. **연구 결과**
   1. **Flow Chart  
      (1) Page Table(구현 실패)  
        
      (2) Disk Swap(구현 실패)  
        
      (3) Stack Growth  
        
      **
   2. 제작 내용  
      (1) Page Table (구현 실패)  
        
      (2) Disk Swap (구현 실패)  
        
        
        
        
        
        
      (3) Stack Growth  
        
      

userprog/exception.c 의 page\_fault 내부에서 not\_present가 조건문에서 확인된다. 페이지 디렉토리인 page\_directory와 폴트가 일어난 주소의 페이지 시작 주소인 upage 를 선언한다. 그 후에 stack의 최대 확장 가능 영역인 8MB에 부합하는지 확인하고 이전 스택 프레임 끝을 의미하는 f->esp-32 보다 fault\_addr이 더 큰지 확인한다. 스택 확장 가능이라고 판정되면 pagedir\_get\_page를 통하여 페이지 디렉토리를 찾은 뒤 pagedir\_set\_page를 통해 메모리 맵핑을 진행하여 스택을 확장한다.

* 1. **시험 및 평가 내용  
       
       
       
       
       
     **