**Pintos Project 4: Virtual Memory**

담당 교수 : 박성용

이름 / 학번 : 김태곤 / 20191583

개발 기간 : 23.11.22 ~ 23.12.10

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술

이 프로젝트의 목표는 가상 메모리 관리 및 페이지 폴트 처리를 개선하는 것이다. 이를 위해 보조 페이지 테이블을 도입하여 기존 페이지 테이블을 보완하고, 페이지 폴트 발생 시 프로그램이 종료되지 않고 제대로 실행될 수 있도록 페이지 폴트 처리 메커니즘을 강화한다. 또한, 메모리 부족 상황에서 디스크로 페이지를 swap out하는 기능을 추가하고, 이를 위해 pseudo-LRU 페이지 교체 알고리즘을 구현한다. 마지막으로, 페이지 폴트가 발생했을 때 필요에 따라 프로세스 스택을 동적으로 성장시킬 수 있는 메커니즘을 개발하여, 스택 공간의 유연한 확장을 지원한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* 아래 각 항목 개발의 필요성 또는 개발 시 기대되는 결과를 간략히 서술
  1. Page Table & Page Fault Handler

Supplemental page table을 추가하고 page fault handling을 구현한다. 기존에는 page fault가 발생하면 exit(-1)이 수행되어 프로그램이 종료되었지만, page fault handler의 작동으로 page fault가 발생한 page가 할당되어 새 page table에 추가된다. 즉, Page fault가 발생할 경우 프로그램이 종료되지 않고 메모리 처리 후 계속 동작을 이어나간다.

* 1. Disk Swap

Physical memory가 실행 중인 프로그램에 필요한 모든 데이터를 담을 수 없을 때 disk swap이 필수적이다. Implement pseudo-LRU policies (second chance)를 이용하여 disk swap을 구현하면 메모리를 보다 효율적으로 관리하는데 도움이 된다.

* 1. Stack Growth

많은 운영 체제에서 각 thread는 동적으로 성장하고 축소하는 스택을 가지고 있다. Stack growth를 올바르게 처리하는 것은 프로그램에서 재귀 함수 호출과 동적 메모리 할당을 지원하는데 중요하다. Stack growth를 효과적으로 구현하면 메모리 사용을 개선하고 stack overflow를 방지할 수 있다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술
  1. Page fault가 발생하는 이유와 이를 handling하는 전반적인 과정을 서술

CPU가 physical memory에 없는 page에 접근할 때 page fault가 발생한다. 이때, 시스템은 fault\_addr을 확인하여 참조의 유효성을 판단한다. 유효하지 않은 참조인 경우, 해당 영역이 확장 가능한지 검토하고, 아니라면 프로세스를 종료하고 할당을 해제한다. 반면, 확장 가능한 영역이거나 유효한 참조라면, stack growth를 하거나 빈 frame을 확보하여 page swap을 하고 메모리에 로드한 후 프로세스를 재시작한다.

* 1. Disk swap 발생 시 사용한 page replacement algorithm에 대해 서술

Disk swap을 관리하기 위해, 이 프로젝트는 second chance algorithm을 사용한다. 이는 pseudo-LRU policy의 변형으로, page를 교체하기 전에 두번째 기회를 준다. 각 page 항목에는 추가 비트인 참조 비트가 있다. 프레임 리스트를 순회하면서 접근된 비트가 1로 설정된 페이지에게 두 번째 기회를 주고, 0으로 설정된 페이지를 교체 대상으로 선택한다. 이때 원본과 달라진 page는 swap out을 통해 업데이트된다.

* 1. Stack growth 구현 시 stack 확장 여부를 판단할 수 있는 방법에 대해 서술

페이지 폴트가 발생한 주소가 스택 접근인지 확인한다. 주소가 physical Base Address에서 8MB 이내에 있고, 스택 포인터(esp)에서 32 바이트 이내인 경우에 스택 접근으로 간주한다. 이 조건을 충족할 경우 스택을 확장한다. 이는 새 페이지를 할당하고 페이지 디렉토리에 추가하여 스택 공간을 늘린다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

11.22 : Project 요구사항 파악

11.23 ~ 11.26 : Pintos Manual 공부

11.27 : Kernel Panic 문제 해결

11.28 ~ 12.9 : Project 요구사항 구현(실패…)

* 1. **개발 방법**
* II. B.의 개발 내용을 구현하기 위해 각각에 대해 다음 사항들을 포함하여 설명
  + 수정해야하는 소스코드
  + 수정하거나 추가해야 하는 자료구조
  + 수정하거나 추가해야 하는 함수
  1. Page Table & Page Fault Handler

page.c page.h 를 추가한다. 이후 page fault handler를 추가 작성하여 page fault 가 일어났을 때 어떤 페이지 폴트인지 확인하고, 유효성을 판단해 추가 작업을 해주는 함수를 작성해야한다. 이번 프로젝트4를 진행하면서 이부분은 구현에 성공하지 못하였다.

* 1. Disk Swap

Disk swap을 관리하기 위해, second chance algorithm을 사용한다. 이는 pseudo-LRU policy의 변형으로 프레임 리스트를 순회하면서 접근된 비트가 1로 설정된 페이지에게 두 번째 기회를 주고, 0으로 설정된 페이지를 교체 대상으로 선택한다. 이때 원본과 달라진 page는 swap out을 통해 업데이트된다. 해당 내용을 구현하기 위해 swap.c swap.h를 추가한다. 이번 프로젝트4를 진행하면서 이 부분은 구현에 성공하지 못하였다.

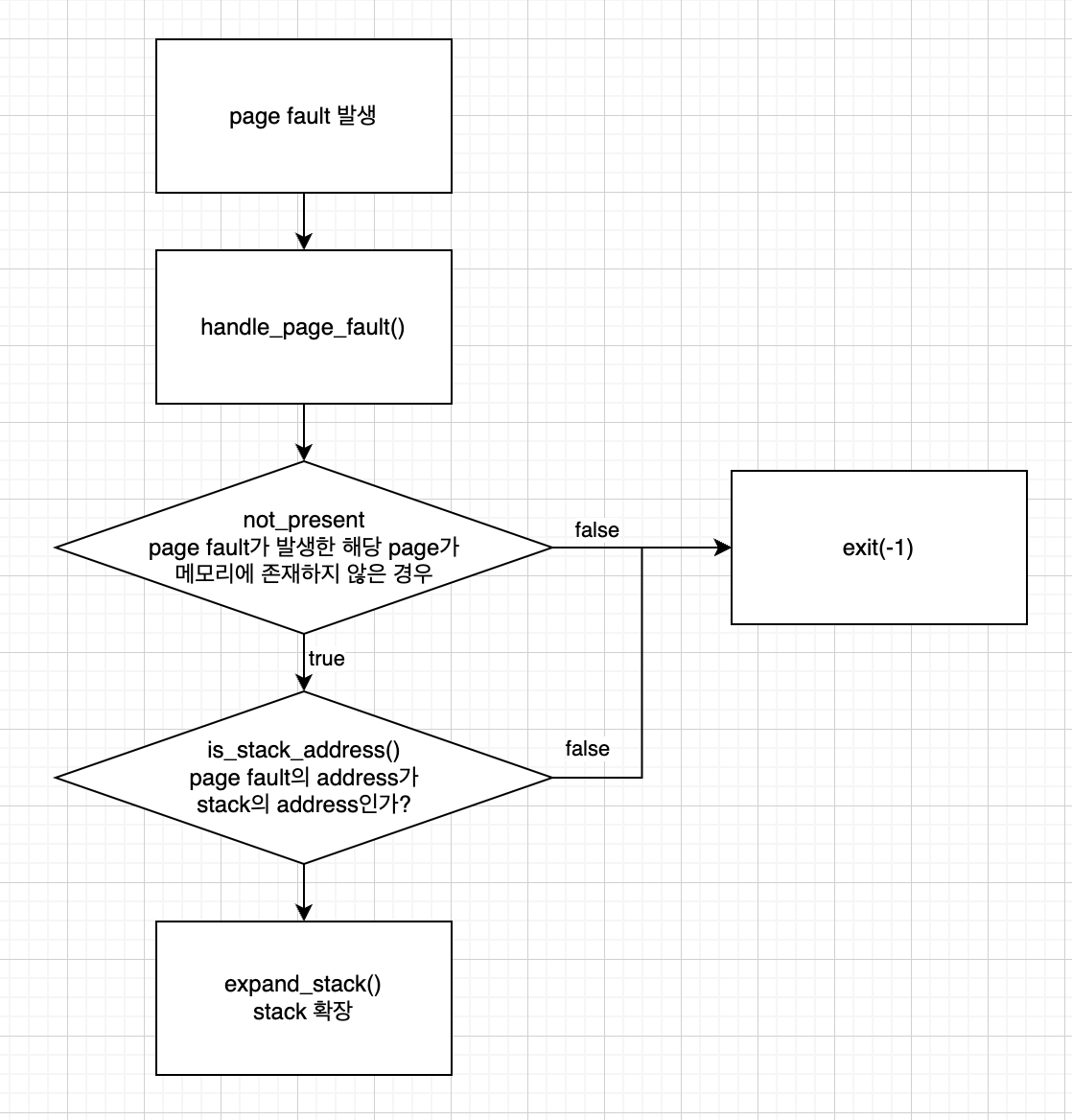
* 1. Stack Growth

Stack growth를 수행하기 위해 handle\_page\_fault, expand\_stack, is\_stack\_address 세 함수를 exception.c 파일 내에 선언하였다. Stack growth는 handle\_page\_fault 함수로 페이지 폴트 처리 과정에서 시작한다. 이 함수는 페이지 폴트가 스택 확장과 관련 있는지 확인한다. 스택 주소에서 페이지 폴트 발생 시, expand\_stack 함수를 호출해 스택을 확장한다. expand\_stack은 해당 주소를 포함하는 페이지를 스택에 추가하며, pg\_round\_down 함수로 주소를 페이지 경계로 내림 처리한다. is\_stack\_address 함수는 주소가 스택 주소인지 확인하며, 스택의 유효 범위를 결정하는 데 사용된다. 이 과정은 사용자 프로세스의 스택이 동적으로 확장될 수 있게 한다.

1. **연구 결과**
   1. **Flow Chart**

* II. B. 개발 내용의 각 항목에 대하여 Flow Chart 작성

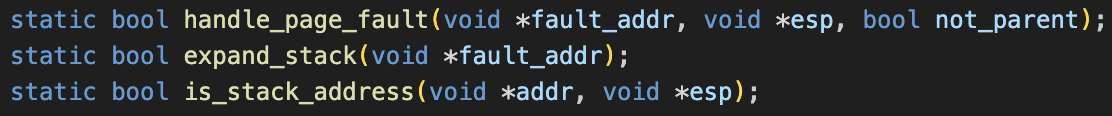
- Stack Growth



* 1. **제작 내용**
* II. B. 개발 내용의 각 항목에 대하여 실질적으로 구현한 코드의 관점에서 작성 (구현 내용, 알고리즘 등을 명확히 서술할 것)
  + 구현에 있어 Pintos에 내장된 라이브러리나 자체 제작한 함수를 사용한 경우 이에 대해서도 설명
* 개발 중 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결한 방식에 대해 설명

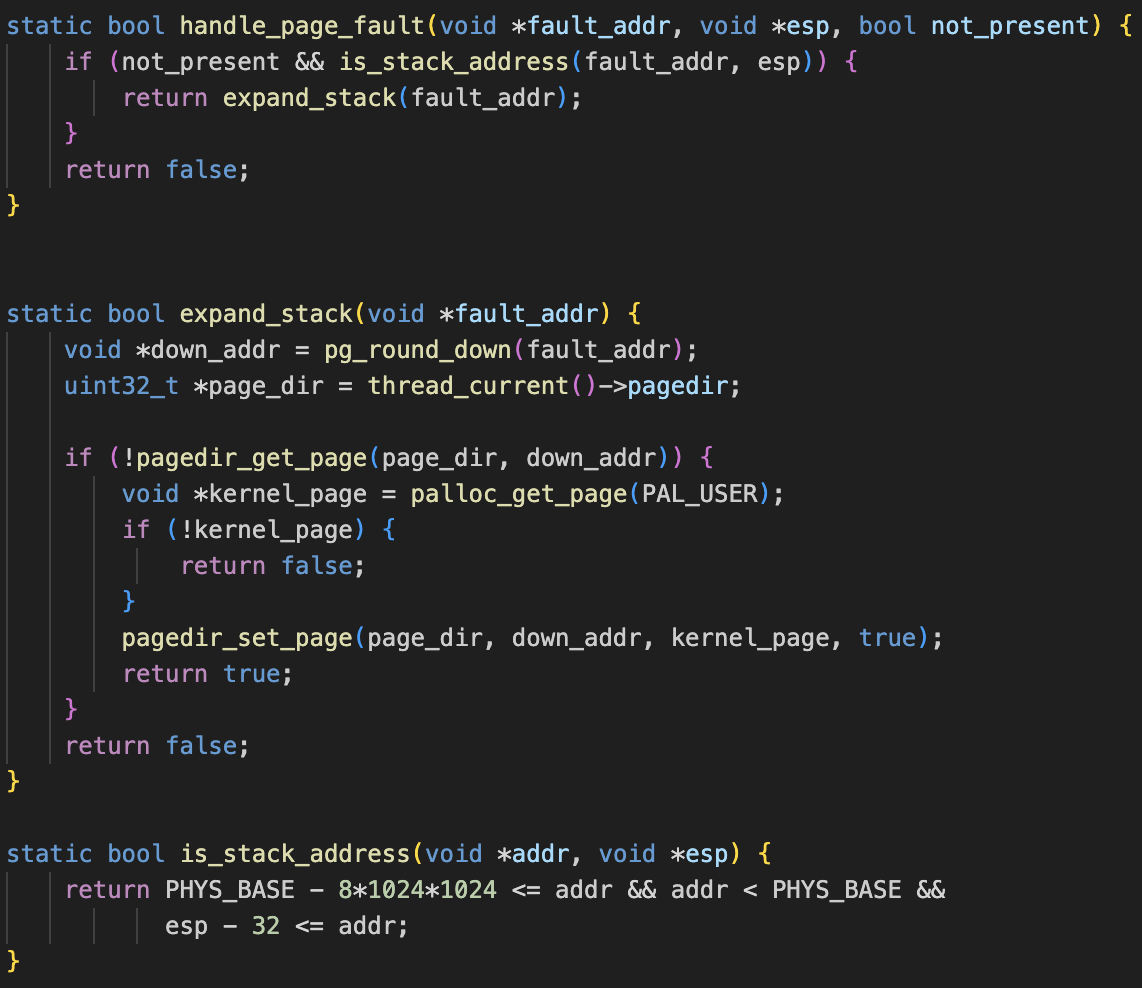
기존 project 3의 코드를 사용하여 project 4의 testcase를 실행시키면 all fail이 발생했었다. 이는 sema up 단계에서 thread\_yield가 발생해서 나타난 page fualt이다. 이를 해결하기 위해 ifndef를 통해 USERPROG일 때만 작동시키거나, thread\_start 이후 thread\_yield가 작동할 수 있게 flag를 추가하여 초기 작업 후에 thread\_yield가 작동하게 수정했었다. 이 작업을 통해 해결이 안되어, synch의 sema\_up에 thread\_yield를 제거하거나, intr\_context를 check하는 등 여러 시도를 해보았지만 고쳐지지 않아 project 2의 코드를 base로 작성하였다.

아래는 exception.c코드에서 stack growth작업을 추가하였다.

****

**텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명**

handle\_page\_fault 함수를 호출하여 페이지 폴트를 처리한다. 이 함수는 페이지 폴트가 스택 확장과 관련이 있는지 확인하고, 필요한 경우 스택을 확장한다. handle\_page\_fault 함수는 페이지 폴트가 발생한 주소가 스택의 일부인지 확인하고, 그렇다면 스택을 확장한다. 스택 확장이 필요한 경우 expand\_stack 함수를 호출한다. expand\_stack 함수는 스택을 확장하는 데 사용된다. 이 함수는 주어진 주소가 포함된 페이지를 스택에 추가한다. 이 작업은 사용자 프로세스의 스택이 동적으로 확장될 수 있도록 해준다. 이 함수 내 down\_addr은 페이지 폴트가 발생한 주소를 페이지 경계로 내림하여 얻어진 주소를 나타낸다. pg\_round\_down 함수는 주어진 주소를 가장 가까운 페이지 경계로 내림처리한다. 이렇게 함으로써, 함수는 페이지 단위로 메모리를 처리할 수 있다. is\_stack\_address 함수는 주어진 주소가 스택 주소인지 여부를 확인한다. 이 함수는 주소가 스택의 유효한 범위 내에 있는지 확인하여, 주소가 스택의 일부인지 아닌지를 결정한다. Ppt 14쪽에 default limit이 8MB로 제한되어 있어 fault address는 PHYS\_BASE-8MB보다 크고, PHYS\_BASE보다 작으며 esp-32보다 크거나 같은 경우 유효한 stack address로 설정하였다.

* 1. **시험 및 평가 내용**
* (채점 대상 테스트 케이스에 해당하는) make check 수행 결과를 캡처하여 첨부

