Pintos 프로젝트 3. Pintos Virtual Memory

(설계 프로젝트 수행 결과)

과목명 : [CSE4070-01] 운영체제

담당교수 : 서강대학교 컴퓨터공학과 박성용

조원 : 83조 권명준, 전해성

개발기간 : 2017. 11. 29. - 2017. 12. 23.

최 종 보 고 서

프로젝트 제목: Pintos 프로젝트 1. Pintos Virtual Memory
제출일: 2017. 12. 23.
참여조원: 83조 권명준, 전해성

I. 개발 목표

물리적 메모리보다 크기가 큰 프로그램도 실행할 수 있도록 가상 메모리를 도입한다. 이를 위해 page table을 구성하고 page fault exception을 적절히 처리해 가상메모리와 물리적 메모리를 연결해 관리할 수 있게 한다.

II. 개발 범위 및 내용

가. 개발 범위

Page Fault Management - page fault exception을 적절히 처리해준다. 실제 물리적 메모리와 연결되지 않은 가상 메모리에 쓰려고 할 때, palloc을 통해 페이지를 할당받아 쓸 수 있도록 처리한다.

Swapping - swapping 상황이 발생할 때 이를 적절히 처리한다. Swap-out을 할 대상은 LRU second change algorithm을 사용하여 결정한다.

Stack Growth - user program이 PAGE_SIZE를 초과하여 stack을 사용하려 하면 페이지를 더 할당받아서 이를 지원한다.

나. 개발 내용

1. Page Fault Management

userprog/exception.c : page_fault() 함수가 처음에는 page fault시 바로 kill하는 방식이지만 이를 수정하여 page fault handling을 할 수 있게 한다.

2. Swapping

LRU(Least Recently Used) second change algorithm을 사용하여 가장 오랫동안 사용하지 않은 page를 swap out 시키고 필요한 페이지를 disk에서 swap in 하도록 한다.

3. Stack Growth

userprog/exception.c : page_fault() 함수가 page fault handling을 하면서 알아서 stach growth도 가능하게 한 번에 구현한다. page를 초과할 때 새로운 페이지를 할당해주면 된다.

III. 추진 일정 및 개발 방법

가. 추진 일정

- 11월 29일 ~ 12월 20일 : 매뉴얼 분석, 자료구조 논의
- 12월 21일 ~ 12월 23일 : page fault, stack growth 구현

나. 개발 방법

pintos manual의 요구사항에서 무엇이 빠졌는지를 파악한다. mmap 부분이 빠졌다. 이 부분을 제외하고 작업을 나눠 하려 했으나 page fault 함수는 기본적으로 먼저 구현되어 있어야 해서 이번 프로젝트는 같이 작업하기로 했다. page swapping의 경우 너무 많은 문제가 생겼다. 심지어 뭘 잘못 건드렸는지 이전 프로젝트인 userprog의 테스트가 전부 fail이 뜨는 경우도 발생하는 등 어려움을 겪었고 결국 swapping은 구현하지 못하였다. page fault handling과 stack growth를 같이 구현했다.

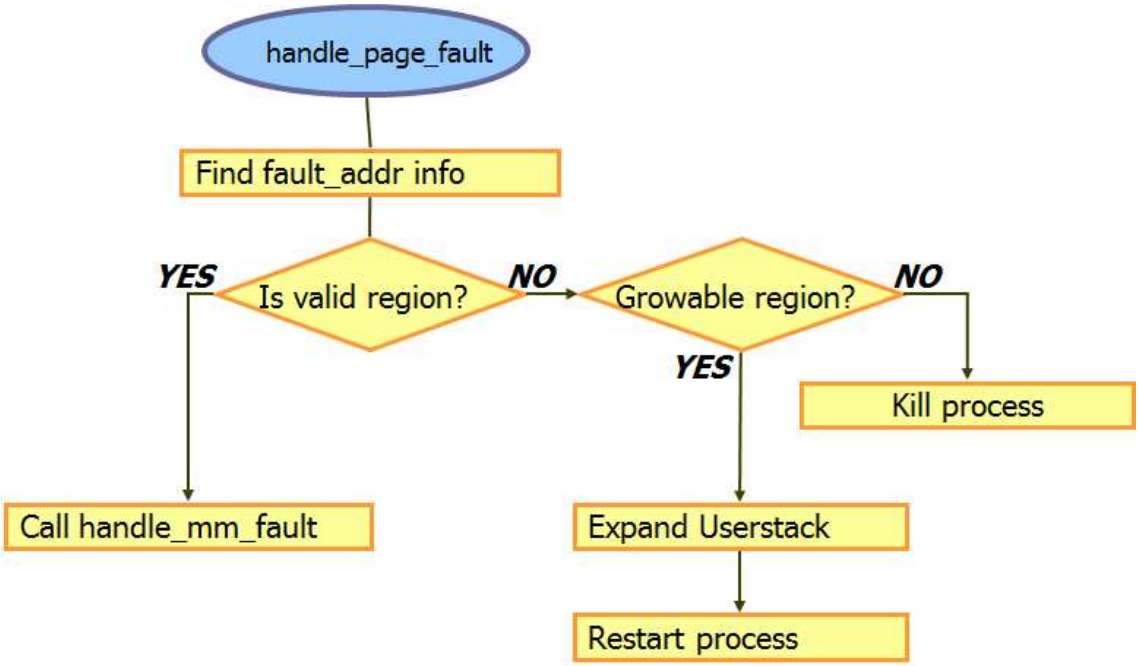
다. 연구원 역할 분담

- 권명준 : page fault, stack growth
- 전해성 : page fault, stack growth

IV. 연구 결과

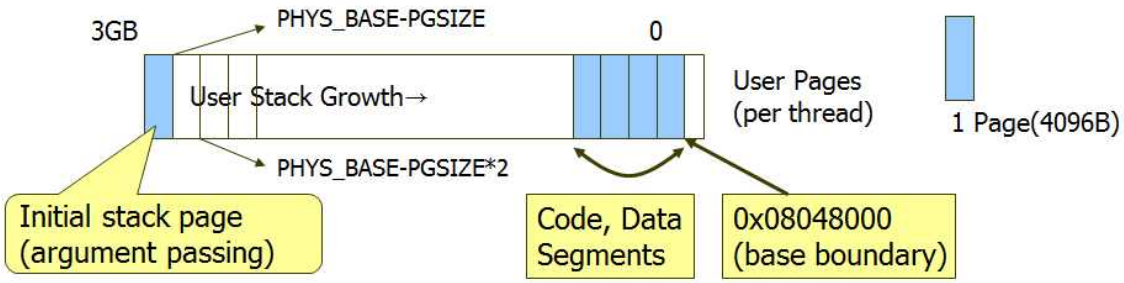
1. 합성 내용

1-1. Page_fault_Handling



위 순서도에서 page swapping 구현과 관련된 함수 handle_mm_fault는 구현하지 못했다.

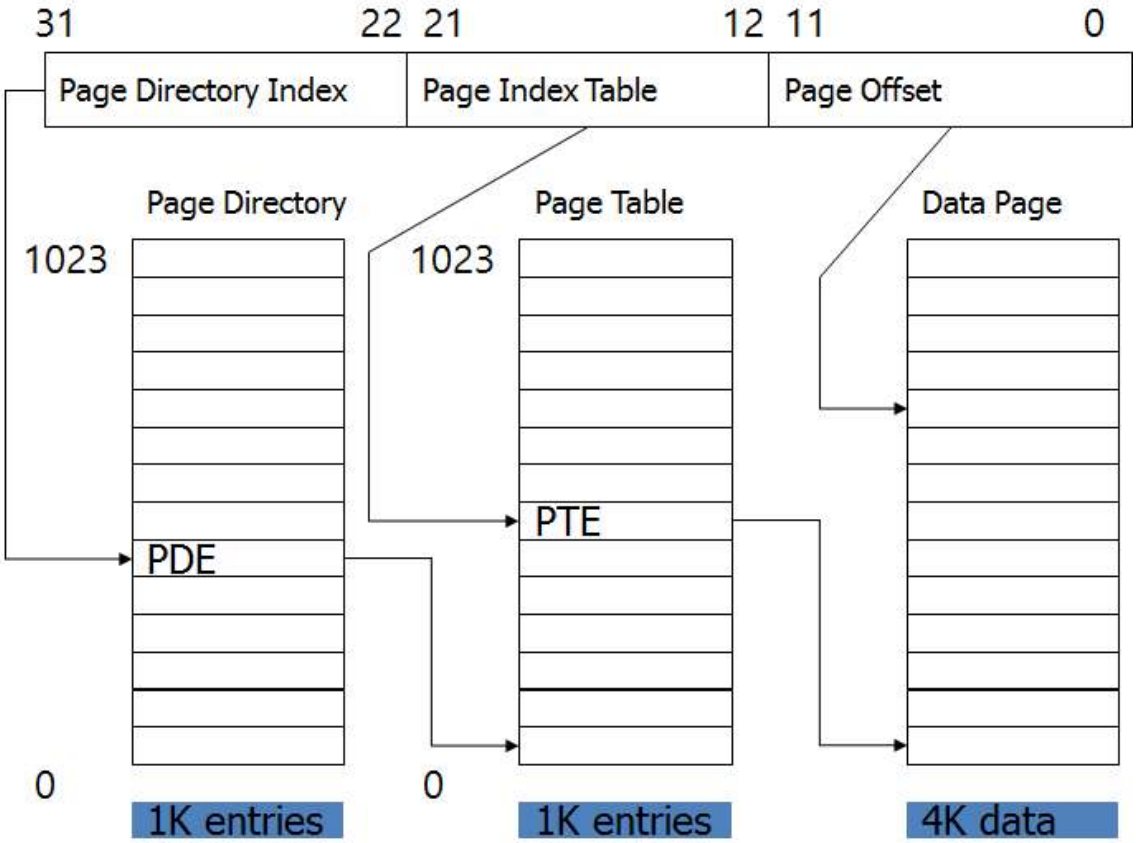
1-2. Stack Growth



2. 제작 내용

2-1. 자료구조

핀토스에서 기본적으로 제공하는 page directory 관련 자료구조와 API들을 이용하여 page swapping을 제외한 Page Fault Handling을 구현하였다.



핀토스에서 페이지를 사용할 때, 32bit 주소를 이용하며 위 그림과 같이 구성되어 있다. 스레드별로 자신의 pagedir를 가지고 있으며 이는 스레드가 이용하고 있는 page directory 이다. 페이지를 해당 페이지 디렉토리에 맵핑시킬 때는 pagedir_set_page와 같은 함수를 사용한다. 또한 해당 페이지 디렉토리에 이미 존재하는 페이지를 찾을 때는 pagedir_get_page를 사용한다.

2-2. 알고리즘

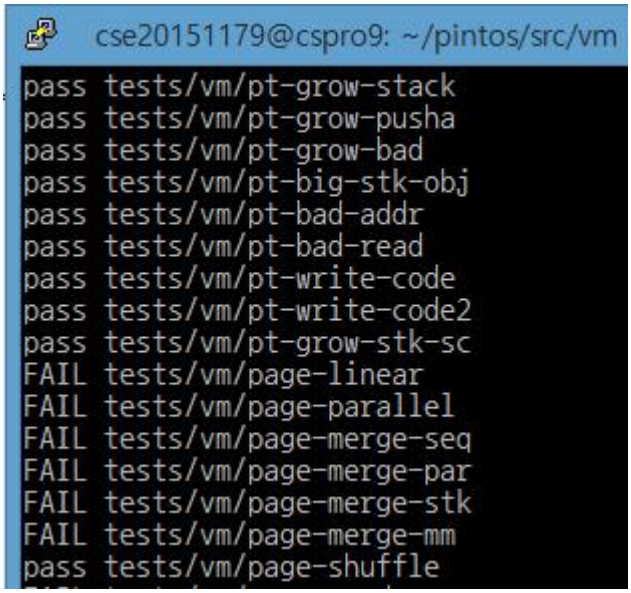
Page fault handling의 경우 위의 순서도를 그대로 따르도록 한다. 이 때, Valid Region 인가는 접근하고자 하는 주소인 fault_addr이 user adrress space에 있는지 확인하여 판단한다. 그리고 invalid access인지는 page_fault 함수 내의 Boolean 변수 not_present, write, user를 이용하여 판단하도록 한다. not_present는 현재 fault_addr이 접근하고자 하는 페이지가 존재하지 않는다면 true이다. false라면 페이지는 존재하지만 쓰기 전용 페이지임을 알려준다. 따라서 not_present가 false라면 잘못된 접근이므로 프로세스를 종료시킨다. write는 그 주소를 읽으려는지 쓰려고 하는지를 가르쳐 주는데, 만약 존재하지 않는 페

이지를 읽으려고 한다면 프로세스를 종료시킨다. 그리고 user는 접근하는 주체가 kernel인지 user인지를 나타내주는데 kernel에서 접근하고자 한다면 프로세스를 종료시키도록 한다.

Stack Growth는 fault_addr이 최대 스택사이즈인 8M안에 있는지 확인한 뒤에, Growable하다면 fault_addr에 맞춰 page-aligned된 주소까지 스택을 키워주도록 한다.

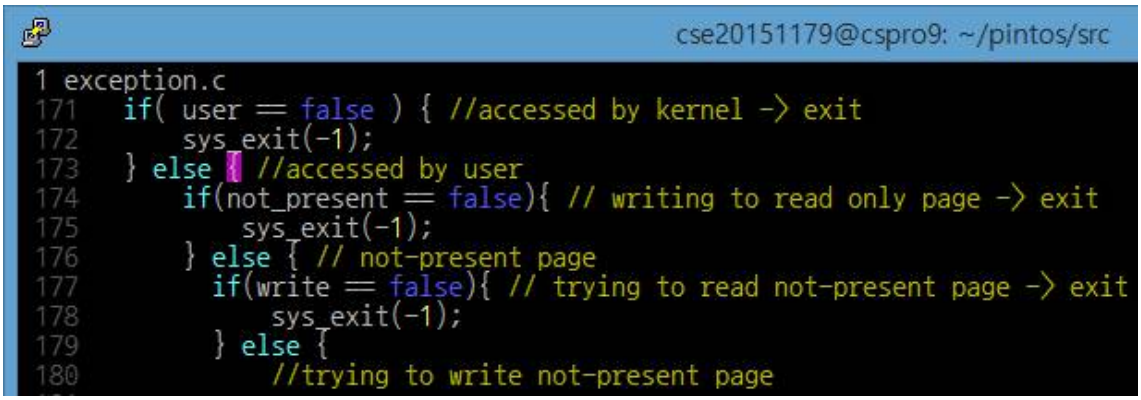
3. 시험 및 평가 내용:

3-1. tests



이번 프로젝트에서 요구하는 테스트 16개 중 10개를 통과하였다. page swapping 관련 테스트를 실패하였고, page fault handling과 stack growth를 통과하였다.

3-2. 보건 및 안정성



userprog/exception.c의 static void page_fault(struct intr_frame *f)에서 다양한 예외상황을 처리해주었다. kernel이 page fault를 호출했다면 치명적인 메모리 오류로 판단하여 바로 프로세스를 종료해버린다. user program이 호출했을 때 만약 read only page에 쓰기작업을 하려고 page fault를 호출한 것이면 이는 paging으로 해결할 수 없기 때문에 프로그램을 종료해버린다. 또한 존재하지 않는 페이지를 읽으려고 들 때도 종료해버린다. 존재하지 않는 페이지에 쓰려고 했을 때에만 page를 할당해 돌려주는 등의 핸들링을 한다. 이 외에도 palloc fail 등 모든 예외를 처리했다. 이런 예외 처리를 통해서 안정적으로 원하는 기능이 동작하도록 구현했다.

3-3. 생산성 및 내구성

불필요한 코드 없이 간결하게 원하는 기능만 딱 작성하였다. 할당하는 페이지 크기는 늘 일정하고 PGSIZE는 상수이므로 page fault 루틴은 항상 같은 동작을 한다. 오로지 변수가 있다면 palloc_get_page인데, 여기서 물리적 공간이 부족하면 프로세스에게 페이지를 줄 수 없다. 이 경우를 제외하면 다른 변수가 없으므로 같은 결과를 낸다. 따라서 여러 번 실행시켜도 전혀 문제없이 같은 결과를 내 놓으므로 내구성에 결함은 없다.

V. 기타

1. 연구 조원 기여도: 권명준(50%) 전해성(50%)

2. 기타 본 설계 프로젝트를 수행하면서 느낀 점을 요약하여 기술하라. 내용은 어떤 것이든 상관이 없으며, 본 프로젝트에 대한 문제점 제시 및 제안을 포함하여 자유롭게 기술할 것.

(느낀점)

- 권명준: 이번 프로젝트는 너무 어렵다. 매뉴얼에도 1532 lines addition이 필요한 것으로 되어있다. 이번에 프로젝트에서 빠진 mmap을 제외한다고 해도 상당한 양이다. 운영체제 과목을 통해 이제는 정말 컴퓨터가 무엇인지에 대해 알게 되었다고 말할 수 있게 되어 기쁘다.

- 전해성 : page swapping을 구현하기 위해서 추가적인 자료구조를 설계하고 핀토스 위에 적용할 때, 디버깅 작업이 너무 오래 걸려 포기한 것이 아쉽다. 시간이 좀만 더 있었으면 완성이 가능했을 거 같아 큰 아쉬움이 남는다.