**Pintos Project 4: Virtual Memory**

담당 교수 : 김영재 교수님

이름 / 학번 : 이도훈 / 20171663

개발 기간 : 11/14 – 11/27

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술

Page fault가 일어나도 종료가 되는 것이 아닌 page fault handler를 통해 handling이 가능하게 되고 stack growth, paging, swap disk, page table management 등을 이번 프로젝트에서 구현한다.

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

기존 page table에 필요한 정보를 추가한다. 이때 주로 page fault handling을 위한 정보를 추가한다. 추가된 table을 통해 fault 발생한 kernel이 fault가 일어난 virtual page를 탐색해서 data를 알아낸다.

Page fault handler의 경우에는 CPU가 page fault를 일으키고, control이 kernerl로 pass 된다. 만약 memory reference가 valid하면 page를 frame에 저장하고 data를 frame에 fetch한다. 만약 invalid하다면 process를 종료한다.

* 1. Disk Swap

Process에 할당해 줄 physical memory가 부족할 때 disk로 swap out이 발생한다. Swap할 page 결정은 page replacement algorithm을 사용한다. Swap disk가 현재 사용하고 있는 슬롯과 빈 슬롯을 관리하는 swap table을 작성해야 한다.

* 1. Stack Growth

스택 영역이 두개 이상의 page에 걸치게 되더라도, 메모리 에러가 나지 않고 자동으로 stack 영역에 할당된 page를 할당하여 stack이 자동으로 grow할 수 있도록 한다

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

Invalid한 page에 access할 시, page fault가 발생한다. Page fault가 발생하면 fault가 발생

한 주소를 알아내서 invalid한 reference인지 여부를 확인한다. Invalid reference라면

Growable 한지 확인하고 가능하면 stack 을 확장시키고 process를 재시작한다. Valid

Reference인 경우에는 page swap을 통해 frame을 memory로 올리고 process를 재시작한다.

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

LRU policy와 유사한 second chance policy를 사용한다. Page가 처음 load됐을 때 1로 설정되었다가 page fault handler를 통해 replace 될 때 0이라면 replace한다. 1인 경우에는 0으로 설정해서 기회를 한 번 더 준다.

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

page fault가 일어났을 때 fault가 일어난 주소가 esp 보다 크거나 esp보다 4만큼 작거나이거나 esp보다 32만큼 작을 때 stack growth가 일어나야 한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

11/14 – 11/27 page table management, swap disk, stack growth 구현

11/27 – 11/27 보고서 작성

* 1. **개발 방법**
* II. B.의 개발 내용을 구현하기 위해 각각에 대해 다음 사항들을 포함하여 설명
  + 수정해야하는 소스코드

Makefile.build & devices/timer.c & threads/init.c, interrupt.c, thread.c, thread.h

Userprog/exception.c, pagedir.c, process.c, syscall.c, syscall.h

Vm/frame.c, frame.h, page.c, page.h, swap.c, swap.h

* + 수정하거나 추가해야 하는 자료구조

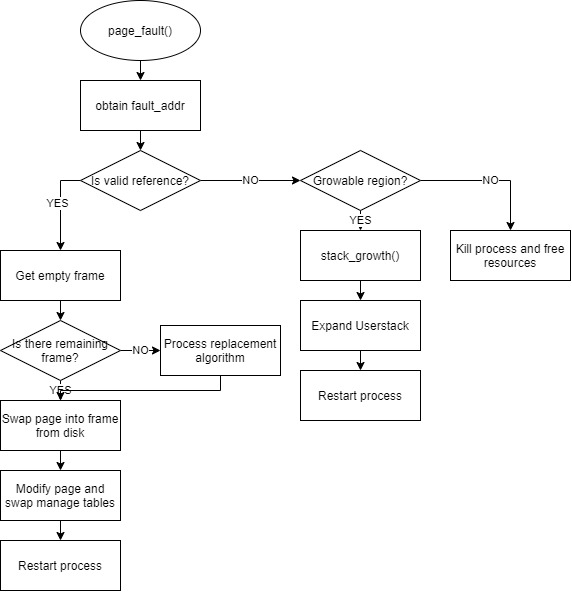
Struct thread, struct vm\_entry

* + 수정하거나 추가해야 하는 함수

vm\_init, vm\_hash\_func, stack\_growth, start\_process, page\_fault …

1. **연구 결과**
   1. **Flow Chart**

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



* 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
  1. Makefile.build

텍스트이(가) 표시된 사진

자동 생성된 설명

Vm 폴더에 파일을 생성하고 새로 생성한 파일을 추가한다.

* 1. Devices/timer.c

텍스트이(가) 표시된 사진

자동 생성된 설명

기존 if문에서 thread\_prior\_aging이 오류를 발생시키므로 삭제한다.

* 1. Threads/synch.c

텍스트이(가) 표시된 사진

자동 생성된 설명

기존 코드에서 thread\_yield 부분이 오류를 일으키므로 userprog 내에서만 구동하도록 변경한다.

* 1. Userprog/exception.c텍스트이(가) 표시된 사진

     자동 생성된 설명

Not\_present가 false라면 write page라는 것이므로 sys\_exit 함수를 통해 함수를 종료한다.

그 이후 addr변수에 fault가 일어난 주소 근처의 page boundary를 저장한다. Fault\_addr이 0보다 크고 phys\_base보다 작은지 확인하고 추가적으로 esp-32와 esp-4와도 비교한다. 두 조건이 모두 만족되면 stack growth를 통해 stack을 확장시키고 아니라면 sys\_Exit함수를 통해 함수를 종료한다.

* 1. Vm/page.h, page.c

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

Current thread와 fault\_addr을 파라미터로 받아온다. Palloc\_get\_page를 통해 new page를 얻고, pg\_round\_down을 통해 fault가 일어난 주소 근처의 boundary를 저장한다. 그 이후 user virtual address와 일치하는 address를 찾고 mapping해서 stack을 확장시킨다.

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

  자동 생성된 설명