**Pintos Project 4 : Virtual Memory**

**(설계 프로젝트 수행 결과)**

과목 명 :운영체제

담당 교수 :김영재

조 / 조원 :10조/ 황순 허준형

개발 기간 :2018-12-21 2018-12-25

**프로젝트 제목 : pintos project 4 virtual memory**

**제출일 : 2018.12.26**

**참여 조원 :황순 허준형**

1. **개발 목표**

**현재 우리가 개발한 핀토스에서 돌아가는 프로그램의 개수와 크기는 머신의 메모리 사이즈에 따라 다르다. 우리는 이 제한을 돌파하여, virtual memory개념을 이용하여 원하는 사이즈로 조정할 수 있게 만든다.**

1. **개발 범위 및 내용**
   1. **개발 범위**

* **Page, Frames, Page Tables, Swap Slot을 개발한다. 각 메모리 구동 방식은 메뉴얼을 참조하여 개발한다.** 
  1. **개발 내용**
* **page table management**

**page table management하기 위해서는 3가지 table이 더 필요하다. 각각은 supplement page table,frame table, swap table이다.**

* **supplement page table은 process가 접근하는 모든 메모리 영역의 정보를 관리한다. supplemental page table에 fault된 page를 저장한다. 만약 그 메모리 참조가 유효하면 supplemental page table entry를 사용한다. 만약 공유를 수행하려면 그 page의 data가 이미 page frame에 존재하지만 그 해당 page table에는 존재하지 않는다. 만약 supplemental page table이 user process가 접근할 수 없는 주소를 가리키거나 page가 kernel virtual memory에 놓여있거나 읽기 전용 page에 대해 write를 시도하는 경우는 접근이 유효하지 않은 것이다. 따라서 유효하지 않은 접근은 exit(-1)함수를 호출한다. 그리고 page를 저장하기 위한 frame을 얻어오고 file system or swap, zeroing으로부터 읽을 data를 frame에 fetch한다. 마지막으로 page fault가 발생한 virtual address에서 physical page로 page table entry가 가리키도록 userprog/pagedir.c 함수를 이용한다.**
* **frame table의 가장 중요한 기능은 사용되지 않는 frame을 얻도록 하는 것이다. 비어있는 frame이 존재하면 이 기능은 쉽지만 비어있는 frame이 존재하지 않으면 한 개의 frame은 비어있어야만 한다.**
* **swap Disk에서 Virtual Memory를 사용하기 위해서는 Memory를 Backing Storage로, Main Memory를 Cache로 추상화하여, 작은 실제메모리를 이용해서 더 큰 가상메모리를 구현해야 한다. 이를 위해서 devices/block.c에 존재하는 block에 관한 API를 이용해서, Swap block으로 사용되는, Backing Storage의 4번째 가상 HDD를 활용하여 Paging의 Swap space로 사용한다.**

**- Paging implementation**

**위의 구현한 page table과 page table을 management하기 위해 구현한 3가지 table을 이용하여 page가 memory에 없는 경우 즉, page를 저장할 frame이 부족한 경우 swap disk를 활용한 page replacement algorithm을 구현한다. 이는 다음과 같은 단계를 가진다. page replacement algorithm을 이용하여 교체할 frame을 선택한다. 이 때, page table에 “accessed"와"dirty" bit을 표시한다. "accessed" bit은 page에 read 혹은 write를 하는 경우 1로 setting하고 write하는 경우에만 “dirty” bit을 1로 setting한다. 이 bit들을 상황에 맞게 표시한 후에 page table로부터 그 frame에 대한 reference를 제거한다. 또한, 필요에 따라서그 page를 file system 혹은 swap disk에 write한다.**

**- Stack growth**

**fault가 발생한 주소를 이용하여 그 주소가 stack growth를 통해 접근할 수 있는 주소이면 그만큼 page를 할당하고 stack이 자라날 수 없는 주소(최대 8MB)이면 user\_exit(-1)함수를 호출한다. page\_fault 처리시 not\_present, write, user라는 부울 타입의 변수가 있다. not\_present는 page fault가 일어난 해당 address에 page가 존재하는가를 나타내는 것이고 write는 해당 영역이 writable한지, user는 user space에서의 page fault인지 kernel 영역에서의 page fault인지를 나타내는 변수이다. 이것들을 이용해 page fault가 일어났을 때 이 page fault가 어떤 상태에서 일어난 page fault인지를 판단할 수 있다.**

**.- Demand Paging**

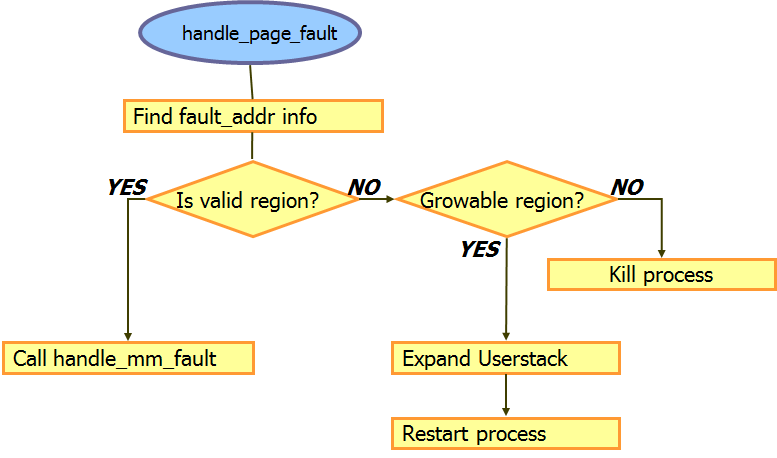
**기존 Process는 단일 페이지내에 적재되어 PintOS에서 구현되었지만, 이제 부터는 그보다 큰 프로세스 또한, Lazy Swapping 방식을 이용해서, 필요한 Segment만 Paging하여 메모리에 적재하는 방식을 사용하여, 더 큰 프로세스와 Frame관리를 유연하게 한다..**

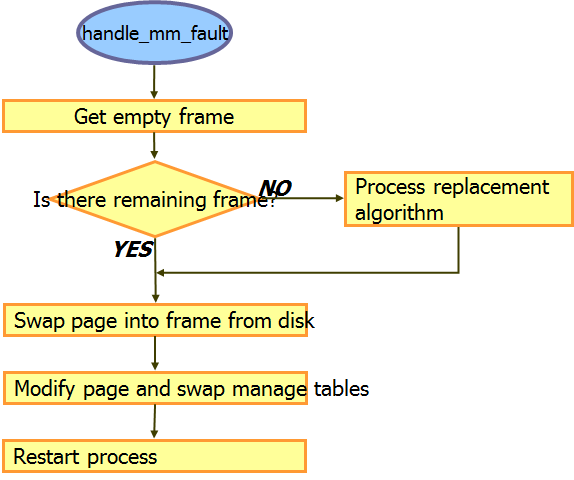
1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* **개발 순서는 위와 같다. 자료구조를 만든후, 우리의 userprog에서 이제 페이지 폴트를 어떤 경우일 때 발생시키는지 바꾼다. 그리고 각 테스트의 경우를 신경써서 한다.**
  1. **개발 방법**
* **깃허브를 이용한 방식으로 역할을 분담하였다. 그리고 매번 만나서 같이 이야기를 하였다.**
  1. **연구원 역할 분담**
* **황순 – 개발 담당**
* **허준형 – 보고서 제작**

1. **연구 결과**
   1. **합성내용**

**- Page fault handler**



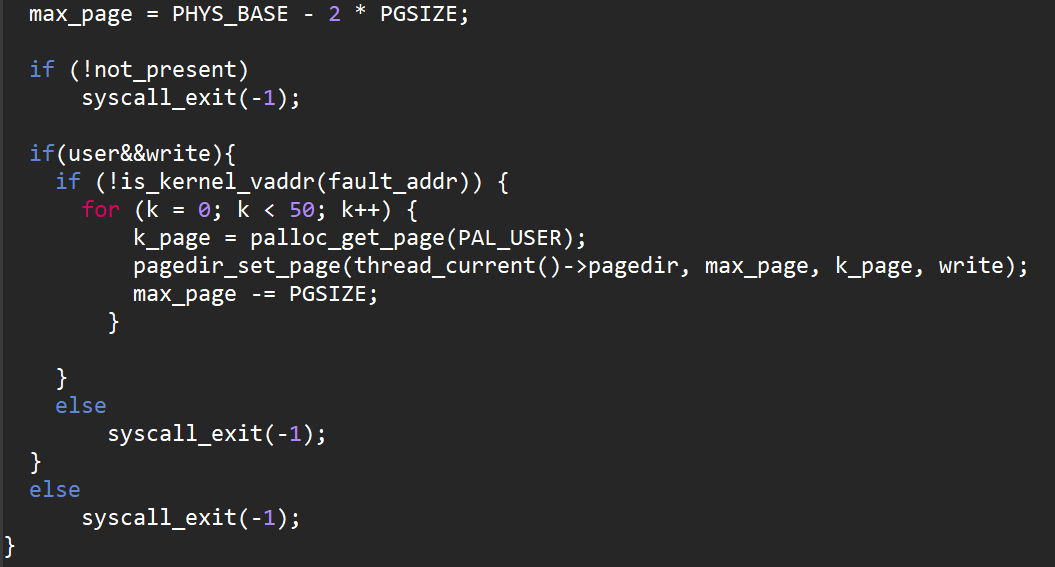


\*위의 순서도들은 page fault handler에 관한 순서도이다. 첫 번째 순서도에서 valid region이 뜻하는 바는 virtual memory주소가 현재 할당된 page들의 주소 영역안쪽인지를 판별해주는 부분이다. yes인 경우 handle\_mm\_fault함수를 call하는데 두 번째 순서도가 이 함수의 수행순서를 나타낸다. no인 경우 fault가 발생한 주소가 stack growth가 가능한 주소인지 판단하고 가능하다면 page를 더 할당하고 불가능하다면 강제종료 시킨다.

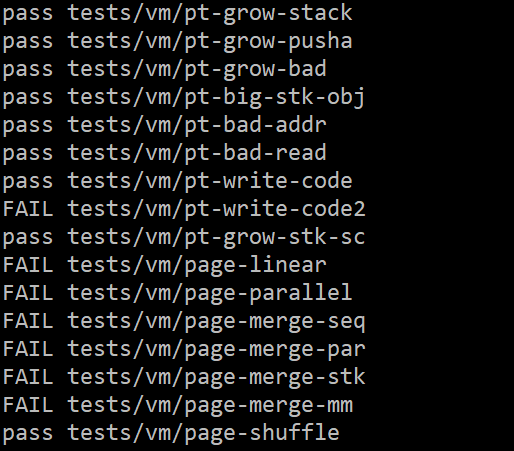
* 1. **제작 내용**

**Stack growth**

* **. exception.c 의 page\_fault 함수 수정**



**다음과 같이 수정하여서 stack이 커질수 있는 상황에 대해서는 수정하여 스택을 늘려줄 수 있도록 하였다. 예외의 경우 종료를 시킨다.**

* 1. **시험 및 평가 내용**
* **평가항목의 테스트 패스 점수는 다음과 같다.**
* 
* **위와 같이 16개의 testcase중 9개의 testcase를 통과하는 것을 알 수 있었다.**

1. **기타**
   1. **연구 조원 기여도**

* **황순(50%), 허준형(50%)**
  1. **소감**
* **황순- 운영체제 수업을 하면서 많은 공부가 됐다. 힘든 일도 많았지만 결과적으로는 리눅스와 운영체제의 동작에 대해서 어느정도 알게 되었다. 그렇지만 실제로 virtual memory를 구현하는 데에는 한계가 있었고, 더 구현해보고 싶었지만 시간적 제약이 커서 더 이상 해보지 못해서 아쉬웠다. 핀토스 프로젝트를 하면서 배운 것들을 직접 코딩해보면서 운영체제가 어떻게 동작하는지에 대해서 경험해 볼 수 있어서 좋은 기회였던 것 같다.**
* **허준형- 운영체제 수업을 이해할 수 있는데 도움이 조금 됐다. 실제 코드가 어떻게 돌아가는지 알 수 있어서 다행이다. 실제 코딩을 하면서 수업시간에 알게된 내용이 많은 도움이 되었고, 수업과 핀토스가 서로 상호작용을 잘 해서 서로를 이해하는데 많은 도움이 되었다. 핀토스 프로젝트를 하면서 힘들었지만 운영체제를 이해하는 좋은 경험을 한 것 같다.**