**Pintos Project 4: Virtual Memory**

담당 교수 :박성용

이름 / 학번 :20191613 윤상현

개발 기간 :

1. **개발 목표**
2. Page Table Management를 위해서 Supplemental page table를 구현하고,

page fault handler를 수정.

1. page에서 disk로의 swap in/out 구현
2. Stack Growth 구현
3. **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

기존에 페이지 폴트가 발생할 경우에는, exit(-1)또는 kill을 호출할 뿐이었습니다.

위 항목의 page table 과 page fault handler를 구현한 이후로는, 적절하지 못한 경우에는 기존과 같이 exit(-1)을 호출하고, 그 이외의 page가 disk에 저장되어 있는 경우, 또는 stack의 크기가 모자라는 경우를 따로 처리해 주어서 프로그램의 메모리 관리의 유연성이 좋아집니다.

* 1. Disk Swap

기존에는 할당할 physical frame이 모자랄 경우에는 해결할 방법 없었습니다.

Disk swap을 구현한 이후에는, 별도의 디스크 파일에 physical frame의 정보를 저장해 놓음으로써, frame이 모자란 경우에 swap out하여 추가로 할당해줄 수 있습니다.

다시 그 정보가 필요한 경우에는 다시 swap in을 해주면 사용할 수 있습니다.

* 1. Stack Growth

기존의 스택은 크기가 1페이지로 할당된 이후에 크기를 따로 변경할 수단이 없어서, 스택 길이를 넘어서는 부분에 액세스할 경우 bad-ptr로 처리되어 exit(-1)을 호출했습니다. 이 부분을 구현할 경우에는 적당한 크기(8mb)의 스택 길이까지는 스택의 크기를 늘려주어서, 좀 더 유연한 프로그램의 작성이 가능합니다.

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

Page fault가 발생하는 이유에는, page의 present가 false이거나, 할당되지 않은 공간에 접근하거나, readnly page에 write를 시도하거나 등의 이유가 있습니다.

일단 page\_Fault내부의 not\_present가 false인 경우에는 read\_nly page에 write하려한 것이므로 exit(-1);

Fault\_addr에 page가 존재하는지 여부를 확인 합니다.

만약 page가 있다면, swap\_out된 것이므로 swap\_in해주면 됩니다.

만약 page가 없는데, fault\_addr이 stack의 길이와 관련하여 생긴 페이지 폴트라면,

Stack\_growth를 실행합니다. Fault\_addr이 더 이상 fault\_Addr이 아니게 될 때 까지 페이지를 새로 할당해줍니다. 그 과정에서 frame이 모자라다면 page replacement를 실행합니다.

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

Disk swap 발생 시에 사용한 알고리즘은 second chance입니다.

일단 메모리에 present한 페이지만 검색하고,

pagedir\_is\_Accessed로 체크해서 와본 적 있으면 즉시 리턴,

아니라면 pagedir\_set\_accessed로 와본적있다고 체크합니다.

단순하게 메모리에 present한 가장 처음 발견되는 2번쨰로 방문하는 페이지를 swap\_out하는 방법입니다.

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

일단 fault\_addr에 이미 page가 존재한다면, stack확장이 아닌 swap\_in이 필요합니다.

PHYS\_BASE 와 8mb보다 멀리 떨어져 있다면, stack growth가 불가능합니다. (한계치)

stack <= fault\_addr || fault\_addr == stack - 32 || fault\_addr == stack – 8 의 조건을 만족해야 합니다. (스택 위에 있거나, 스택 내부의 페이지 폴트인 경우)

fault\_Addr < PHYS\_BASE를 만족해야합니다 :스택의 크기를 늘려야만 하기때문

위 조건을 만족시 stack growth를 실행하고 아니면 그냥 exit(-1)을 하게 됩니다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

Supplenmental page table 작성

=> disk swap in/out 작성

=>page evict 작성

=>stack growth 작성

* 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

Src/userprog:

Process.c

Intall\_page(): pagedir에 추가할때에 spt에도 추가하는 부분 추가

Setup\_stack(): spt에 page 추가하는 부분 추가.

Load\_segment(): spt에 page 추가하는 부분 추가.

Exception.c

Page\_Fault(): swap\_in할지, 스택 크기를 늘릴지 exit(-1)할지 여부 구현

Src/threads

Thread.h

thread자료구조에 spt\* 추가

init.c

main(): disk파일 초기화관련 구문 추가.

Src/vm

page.c

Struct spt: page를 관리하는 hash\_Table 저장

Struct page: page의 user addr, kernel addr, present, 등등의 정보 저장

Struct frame:

Bool Swap\_space[1024]: disk의 섹터가 비어있는지 여부

Hash\_page(): page의 user addr을 hash 값으로 변형시켜줌

Hash\_less(): hash 비교 함수

Page\_table\_init(): spt초기화 함수

Page\_find():spt에서 page를 찾아줌

Page\_insert():spt에 page를 추가해줌

Page\_install():page를 pagedir에도 추가해줌

Evict\_page():frame을 swap\_out해서 메모리 확보

Swap\_init():swap을 하기위한 디스크 초기화

Swap\_in():disk에서 page로 정보를 읽어옴

Swap\_out():page에서 disk로 정보를 쓰고 page를 free

1. **연구 결과**
   1. **Flow Chart**

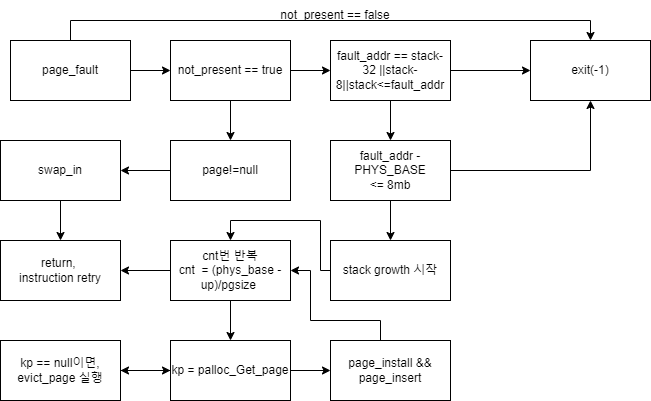
* II. B. 개발 내용의 각 항목에 대하여 Flow Chart 작성

Page fault handler / disk swap / stack growth

텍스트이(가) 표시된 사진

자동 생성된 설명





* 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

Src/userprog:

Process.c

=Intall\_page(): pagedir에 추가할때에 spt에도 추가하는 부분 추가. Page\_insert() 함수를 추가해서 spt의해시테이블에 삽입해줌

=Setup\_stack(): spt에 page 추가하는 부분 추가. Page\_insert를 추가한 install\_page를 호출

=Load\_segment(): spt에 page 추가하는 부분 추가. Page\_insert를 추가한 install page를 호출

Exception.c

=Page\_Fault(): swap\_in할지, 스택 크기를 늘릴지 exit(-1)할지 여부 구현

만약 not\_present == false면 readonly page에 작성한거니까 바로 exit(-1)

Fault\_Addr을 pg\_round\_down으로 user-addr로 변환해주고, page\_find함수로 spt에서 찾아본다. 그 후에 page가 NULL이 아닐경우에는, swap\_out당한 page이므로 swap\_in해주고 return해준다. (return하면 reinstruct된다)

Page가 NULL이고, PHYS\_BASE 와 8mb이하만큼 떨어져 있고, 스택 fault인 경우에는 stack의 길이를 늘려준다. (fault\_addr에 닿을떄까지 페이지를 추가로 할당해준다.)

어느 것에도 해당되지 않을 경우 exit(-1)해준다.

Src/threads

Thread.h

thread자료구조에 spt\* 추가

init.c

main(): disk파일 초기화관련 구문 추가. Init\_Swap을 추가해서, block\_get\_Role(SWAP)을 통해서 디스크의 위치를 저장하고, lock을 초기화한다.

Src/vm

page.c

Struct spt: page를 관리하는 hash\_Table 저장하고, lock들을 저장

Struct page: page의 user addr, kernel addr, present, 등등의 정보 저장

Struct frame:

Bool Swap\_space[1024]: disk의 섹터가 비어있는지 여부

Hash\_page(): page의 user addr을 hash 값으로 변형시켜줌

Page->ua를 hash\_int()함수로 변형한 값을 리턴

Hash\_less(): hash 비교 함수

Page->ua끼리 비교

Page\_table\_init(): spt초기화 함수 spt를 malloc해준 이후에, lock들을 초기화 해줌.

Page\_find():spt에서 page를 찾아줌. Hash\_find함수를 이용해서

Page\_insert():spt에 page를 추가해줌. Hash\_insert()함수를 이용

Page\_install():page를 pagedir에도 추가해줌. Install\_page함수가 process.c에서만 쓸 수 있는 함수라서, 같은 내용으로 작성

Evict\_page():frame을 swap\_out해서 메모리 확보. Second chance 알고리즘을 이용하여 page를 evict함. 고른 page는 swap\_out을 진행하고, 차지하고 있던 page, kaddr등을 해제해주고 pagedir에서도 삭제

Swap\_init():swap을 하기위한 디스크 초기화. Swap\_space를 false로 초기화해주고, block을 초기화

Swap\_in():disk에서 page로 정보를 읽어옴. Swap\_out을 진행한 후에, spt에저장 되어있던 block의 index에서 페이지에 대한 정보를 읽어옴. 그 후, swap\_space를 다시 false로 초기화해줌.

Swap\_out():page에서 disk로 정보를 쓰고 page를 free. Swap\_space에서 아직 비어잇는 곳의 인덱스를 찾고, block\_write함수로 디스크에 정보를 써줌. 그 후 그 인덱스를 page에 저장하고, page와 연결되어 있던 frame을 해제해줌. pagedir에서도 삭제함.

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

텍스트이(가) 표시된 사진

자동 생성된 설명

