**Pintos Project 4: Virtual Memory**

담당 교수 : 김영재 교수님

이름 / 학번 : 김효림 / 20221549

개발 기간 : 2024 12/20~12/22

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술

pintos에서 virtual memory가 작동할 수 있도록 page와 frame 그리고 swap 파일을 작성한다. Page fault를 처리할 수 있도록 하고, stack growth, LRU 정책이 적용된 디스크에서의 paging 처리가 가능할 수 있도록 한다.

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

Disk에 page가 있는지 확인하는 로직을 추가하고 이를 불러오는 작업을 한다. 즉 CPU에서 참조한 주소가 물리 메모리에 존재하는지 확인한 후, 없을 경우 디스크에서 필요한 데이터를 가져온다. 해당 내용을 구현하게 되면 page fault를 더 reliable할 수 있게 한다.

* 1. Disk Swap

물리 메모리가 부족한 상황에서 swap-out을 통해 page를 얻어 온다. 이를 통해 메모리 관리의 효율성을 높일 수 있다.

* 1. Stack Growth

stack에 접근할 때 page fault가 일어난다면 stack growth를 통해 크기를 조절하여 적절한 page를 할당할 수 있도록 한다. 이를 통해 동적으로 메모리를 관리할 수 있게 된다.

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

접근이 불가능한 영역에 접근을 시도할 때, 물리 메모리에 존재하지 않고 disk에 저장된 페이지에 접근할 때 page fault가 발생한다. 즉, CPU가 가상주소를 참조했을 때 해당 주소가 물리 메모리에 올라오지 않는 경우이다. Page fault가 발생하면 page fault handler는 disk에 있는 경우 해당 페이지를 검색한다.

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

LRU 알고리즘을 사용해 Least Recently Used page를 page replacement 대상으로 선정한다. LRU list를 사용해서 교체 대상의 페이지를 탐색하고, page가 dirty인 경우 swap 영역에 저장한다. 그리고 새로운 페이지를 물리 메모리에 로드해 준다.

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

Page fault가 발생했을 때 넘겨 받은 faulting address가 기존 stack pointer 아래 일정 오프셋 내에 있을 경우에는 스택을 확장하고, faulting address가 범위를 초과한 경우 확장하지 않는다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

12/20~12/21 : page.h, page.c, frame.h, frame.c, swap.h, swap.c 구현

* 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
* <Page fault> Pintos manual에 따라 page.h, page.c를 통해 supplemental page table을 구현하여 virtual address와 physical memory 간의 mapping을 관리한다. 여기에 page fault 처리 로직을 포함한다. 이때 struct hash을 이용해 hash map 자료구조를 기반으로 structure를 구성한다. 해당 방식으로 struct spt\_entry 및 struct page를 작성한다. 이때 spt에 page를 추가하거나 삭제하는 함수, 디스크에서 데이터를 읽어 메모리에 로드하는 함수, spt를 탐색하는 함수 등을 작성해야 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* <Disk swap> Pintos manual에 따라 frame.h, frame.c를 통해 LRU 정책에 기반해 frame table을 구현한다. 메모리가 부족할 경우 페이지 교체를 위해 frame\_evcit 함수를 작성한다. 또한 swap.c, swap.h를 작성하여 swap 공간을 관리하여 페이지를 디스크로 내보내거나 복원하는 swap\_in, swap\_out 함수를 작성한다. Frame에 관련해서는 struct frame를 선언해 물리 메모리의 프레임 정보와 연결된 페이지를 관리한다. 또한 list 자료구조를 활용해 lru\_list를 선언하여 LRU 정책에 따라 순환 탐색을 위한 리스트를 사용한다. Swap 영역 관리를 위해서는 bitmap 자료구조를 활용한다. 이때 물리 메모리에서 새로운 프레임을 할당하는 함수, LRU 정책에 따라 교체할 프레임을 선택하는 함수(evict), 사용하지 않는 프레임을 해제하는 함수를 구현해야 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* <Stack growth> : /userprog/process.c에서 Stack growth를 지원할 수 있도록 함수를 작성하거나 코드를 수정한다. 해당 부분을 수정하지 못하였다.

1. **연구 결과**
   1. **Flow Chart**

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

텍스트, 도표, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 친필, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

텍스트, 친필, 도표, 폰트이(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

-Page Fault Handling

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

두 함수는 spt에서 spt\_entry를 저장할 위치를 결정하기 위해 해시 값을 생성(page\_hash)하고, spt에서 spt\_entry를 정렬하기 위해 비교(page\_less)한다. Page\_hash 함수는 hash\_bytes 함수를 이용해 spt\_entry의 가장 주소인 vaddr을 해싱한다. Page less 함수의 경우 해시 테이블 내에서 충돌이 발생했을 때 정렬하기 위해 사용한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위 세 함수는 spt를 초기화하고, 새로운 spt\_entry를 삽입하거나 삭제하는 동작을 한다. Spt\_init의 경우 hash\_init을 통해 spt의 해시 테이블을 초기화한다. Spt\_insert의 경우 입력으로 받은 vaddr이 이미 spt에 존재하는지 확인한 후에, 존재하지 않으면 hash\_insert를 통해 spt\_entry를 추가한다. Spt\_delete는 spt\_entry를 찾아 hash\_delete를 호출하며 제거한 후, 해당 메모리를 free를 통해 할당을 해제한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Spt\_destroy는 hash\_destroy 함수를 호출하여 테이블의 모든 항목에 대해 spt\_entry\_destroy\_func를 실행하여 spt\_entry에 할당된 리소스를 해제하고, spt를 제거한다. Spt\_entry\_destroy\_func는 spt\_entry가 물리 메모리에 로드된 상태인지 확인하고, 로드된 경우 페이지를 해제한 후 page directroy에서 매핑을 해제한다. 그후 마지막으로 spt\_entry 메모리를 해제한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Spt\_find 함수는 전달받은 vaddr을 기반으로 spt에서 해당 virtual memory를 가지는 spt\_entry를 찾는다. Hash\_find 함수를 호출하여 spt에서 해당 항목을 검색한다.

Load\_page 함수는 디스트에 저장된 페이지 데이터를 물리 메모리에 로드하는 역할을 한다. 파일이 BIN 혹은 FILE 타입인 경우 file\_read\_at 함수를 이용해 데이터를 읽어 물리 메모리에 복사한 후, 읽지 않은 공간은 0으로 채운다.

<pagedir\_get\_page : vaddr이 매핑된 물리 주소를 반환한다. Page\_clear\_page : page directory에서 매핑을 제거한다. Palloc\_free\_page : 메모리를 해제한다.>

-Page replacement

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

Frame\_allocate 함수는 페이지를 물리 메모리에 할당한다. 메모리가 부족한 경우 page replacement 실행을 위해 해당 함수를 사용한다. Palloc\_get\_pace를 통해 메모리 할당을 시도한 후, kaddr이 NULL인 경우(할당 실패) frame\_evict를 호출해 교체할 페이지를 선택한다. 새로운 frame 구조체를 생성한 후에 LRU list에 생성된 frame을 lru list에 추가한다.

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Frame evict 함수는 LRU 방식을 기반으로 교체할 페이지를 선택하게 된다. Page\_is\_accessed 함수를 사용해 최근에 접근된 페이지를 확인하고, page\_is\_dirty를 이용해 dirty 상태를 확인한다. Page\_is\_dirty 함수는 page directory를 확인해 해당 페이지가 변경된 적이 있는지를 확인한 후, 변경된 경우 디스크나 swap partition에 저장한다. 만약 접근한 page가 anonymous 페이지인 경우 swap\_out이 호출되도록 조건문을 설정한다. 만약 type이 FILE이었다면 변경된 내용을 원래 파일에 저장해 준다. 그 후 file\_write\_at 함수를 사용해 데이터를 파일애 기록한다..

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Frame\_free 함수는 frame을 해제하고 LRU list에서 제거한다. LRU list를 순회하여 해당 kva에 해당하는 frame을 찾고, palloc\_free\_page를 이용해 물리 메모리를 해제한다. 그 후 free를 통해 frame 구조체 메모리를 해제한다.

<pagedir\_set\_accessed : page directory의 accessed 비트를 설정한다. 이 함수는 페이지가 accessed 상태인지 확인한 후, 해당 비트를 리셋하여 LRU 알고리즘에서 최근 접근 여부를 확인할 수 있도록 한다.>

-Swap In/Out

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Swap in/ swap out 함수는 물리 메모리의 페이지를 swap partition에 저장하거나 해당 부분에 저장된 페이지를 물리 메모리로 복원하는 역할을 한다. Swap\_out 함수는 비트맵에서 빈 슬롯을 검색하여 저장한 후(bitmap\_scan\_and\_flip) 페이지 데이터를 해당 swap slot에 저장한다.(block\_write). 마지막으로 swap slot 인덱스를 반환한다. Swap int 함수는 bitmap\_test를 이용해 swap slot 상태를 확인하고, block\_read를 통해 swap slot 데이터를 물리 메모리로 복사한다. 그후 bitmap\_flip을 이용해 사용 가능 상태로 상태를 변경해 준다.

<bitmap\_test : bitmap 구조체를 확인해, 특정 위치가 사용 중인지 아닌지 확인한다. 사용 중이라면 true, 아니라면 false가 반환된다. Block\_read : 지정된 블록 디바이스에서 데이터를 읽어 온다. 각 파라미터는 block device에 대한 포인터, 읽어들일 sector의 번호, 데이터를 저장할 buffer를 의미한다. Bitmap\_flip : 비트뱁에서 특정 위치의 비트를 반전시킨다. Swap slot의 사용 상태를 전환하기 위해 사용한다. Bitmap\_scan\_and\_filp : 비트맵에서 연속된 빈 슬롯을 검색한 후, 해당 슬롯의 상태를 반전시킨다. Block\_write: block\_read와는 반대로 블록 디바이스의 특정 섹터에 데이터를 기록한다.

* 1. **시험 및 평가 내용**
* (채점 대상 테스트 케이스에 해당하는) make check 수행 결과를 캡처하여 첨부
* 텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

  자동 생성된 설명 텍스트, 스크린샷, 패턴, 타이포그래피이(가) 표시된 사진

  자동 생성된 설명