**Pintos Project 4: Virtual Memory**

담당 교수 / 분반 : 박성용 교수님

이름 / 학번 : 홍주표 / 20181702

개발 기간 : 2022.11.15 ~ 2022.12.07

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술

기존 프로젝트 1, 2, 3에서 page fault가 발생하면 프로그램은 종료되었다. 이번 프로젝트에서는, page fault에 대한 처리를 추가한다. 물리 메모리가 가득 찬 경우, swap을 할 수 있게 구현하고, page fault가 stack 영역에서 발생하면 stack growth를 구현함으로써 더 reliable하고 프로그램이 적절하게 실행되는 핀토스를 구현하는 것을 목표로 한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* 아래 각 항목 개발의 필요성 또는 개발 시 기대되는 결과를 간략히 서술
  + Page Table & Page Fault Handler

기존 page table에 기존 핀토스에는 구현되어 있지 않은 기능인 virtual address를 physical address로 변환하는 기능을 추가한다. 이 때, 특정 virtual address에서 page table 내의 physical address를 찾으려고 할 때 오류가 발생하면 page fault가 발생한다. 기존 핀토스는 page fault가 발생하면 프로그램을 종료했지만, page fault handler를 적절히 구현하여 단순히 종료시키는 것이 아닌, 메모리에서의 처리를 통해 프로그램이 계속 동작할 수 있게 구현한다.

* + Disk Swap

process에 할당할 physical memory가 부족할 때, swap disk에 사용하지 않는 frame을 넣고 새로운 frame을 할당받음으로써 오류가 나지 않게 구현한다. 이 때, LRU 알고리즘을 이용하여 어떤 frame을 swap disk에 넣을지 선택한다. 따라서 메모리에 용량이 가득 차더라도 프로그램을 계속 동작시킬 수 있게 된다.

* + Stack Growth

기존 핀토스에서는 stack의 크기가 4KB로 고정 할당되어 있다. 해당 stack의 용량을 초과하는 영역에 대한 접근이 발생하는 경우 stack을 확장시키는 동작을 구현한다. 이 때, stack을 확장시킬 것인지에 대한 유효성도 검사해야 하는데, 만약 segment 영역에 접근을 하는 경우에는 프로그램을 종료하는 등의 처리를 한다.

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

앞서 설명했듯이, 일정 virtual address에 접근할 때, 이에 해당하는 physical address가 존재하지 않는다면 page fault가 발생한다. 다시 말해, page table에서 virtual address에 해당하는 physical address를 발견하지 못한다면 page fault가 발생한다는 것이다.

이 때 page fault를 handle하려고 한다. 이는 해당 virtual address에 해당하는 내용을 disk에서 읽어와 memory에 frame으로 쌓는 과정이다. 이 과정에서 해당 virtual address가 segment 영역인지 아닌지 등의 유효성 검사를 해야한다. 만약 handle에 성공했다면, page table 또한 update하고, page table을 통해 앞서 접근했던 virtual address에 대응하는 memory address를 매핑한다.

* + Disk swap 발생 시 사용한 page replacement algorithm에 대해 서술

second chance 알고리즘을 사용한다. 메모리에 load되든 page의 circular list를 관리한다. 참조 비트 (reference bit)는 초기에 1로 설정된다. page fault handler에 의해 page가 교체되어야 한다면, 해당 page의 참조 비트를 확인한다. 그 값이 0이면 교체하고, 1이라면, 참조 비트를 0으로 설정, page를 떠난 후 다음 순서에 다시 확인하는 방식이다.

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

추가되는 page가 stack access로 나타나는 경우에만 페이지를 할당한다. 이는 스택 포인터 esp를 이용하여, user memory의 stack 영역인지, stack frame에 있는 주소인지를 판별하는 것이다. 만약 esp가 user memory의 stack 영역을 벗어나면, segmentation fault가 발생하기 때문에 이를 처리해야 한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* 개발 범위를 포함하여 구현 내용에 대한 일정 작성

22.11.15~22.11.20: Page table & Page fault handler 구현

22.11.21~22.12.02: Disk swap 및 Stack growth 구현

22.12.03~22.12.04: 보고서 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
  + Page Table & Page Fault Handler

기존 핀토스에서 page fault가 발생하면 pintos/src/userprog/exception.c 중 page\_fault 함수에 들어가고 에러 메시지 출력 후 프로세스를 kill 하는 것으로 구현되어 있다. 이를 이번 목표에 맞게 수정한다.

우선 pintos/src/vm 디렉토리에 page.c와 page.h 파일을 생성한다. page.h에 page table과 이에 대한 엔트리 (vm\_entry)를 나타내는 구조체를 추가한다. 이 때, page table은 hash table로 구현한다. 이를 위해 pintos/src/threads/thread.h 중 thread 구조체에 hash 구조체 (vm)을 추가한다. 또한 hash 함수를 사용하여 hash table을 초기화하는 함수, 엔트리를 추가하는 함수, 엔트리를 제거하는 함수 등을 추가한다. 각 함수에 대한 설명은 후술하겠다.

그리고 pintos/src/userprog/process.c 중 load\_segment 함수를 수정한다. 기존의 물리 페이지를 할당하고 맵핑하는 부분을 주석 처리하고 vm\_entry를 생성, 필드 초기화, hash table에 추가하는 함수로 변경한다.

또한 pintos/src/userprog/process.c. 중 setup\_stack 함수를 수정한다. 기존에 존재하는 내용 (페이지 할당, 페이지 테이블 설정, 스택 포인터 설정) 아래에 vm\_entry를 생성, 필드 초기화, hash table에 추가하는 함수를 추가한다.

가상주소의 유효성을 검사하는 내용도 추가한다. 말 그대로, 가상주소에 해당하는 vm\_entry가 존재하는지를 검사하는 것이다. 이는 pintos/src/userprog/syscall.c에서 진행한다. 각 함수에 대한 설명은 후술하겠다.

최종적으로 page fault를 처리하기 위한 함수를 추가, 수정한다. 앞서 말했듯이, pintos/src/userprog/exception.c 중 page\_fault 함수에서 에러 메시지 출력 후 프로세스를 kill 하는 것을 주석처리 한 후, fault\_addr에 대한 유효성 검사를 한 후, page fault handler 함수 (handle\_mm\_fault)를 호출한다. 이 함수는 pintos/src/userprog/process.c에 구현한다. physical memory에 할당 완료 후, 실제 디스크의 파일을 physical page로 load한다.

이로써 page fault를 처리할 수 있게 된다.

* + Disk Swap

swap과 관련된 동작들은 pintos/src/vm 디렉토리의 swap.c와 swap.h 파일에 구현한다.

우선 pintos/src/vm/page.h에 유저에게 할당된 phycial page를 표현하는 자료구조인 page 자료구조를 추가한다.

그리고 swap.c와 swap.h에 LRU 리스트를 초기화 하는 함수, LRU 리스트 마지막에 유저 페이지를 삽입하는 함수, LRU 리스트에서 유저 페이지를 제거하는 함수를 추가한다.

또한 유저페이지의 할당/삭제에 대한 함수도 구현한다. 이 함수들은 pintos/src/vm 디렉토리의 frame.c와 frame.h 파일에 구현한다.

위에서 구현한 함수들을 기반으로 pintos/src/userprog/process.c 중 setup\_stack 함수를 수정한다. 기존에 사용하던 palloc\_get\_page 함수는 alloc\_page 함수로, palloc\_free\_page 함수는 free\_page 함수로 수정한다. 이 수정은 같은 파일 내의 handle\_mm\_fault 함수에서도 이루어진다.

* + Stack Growth

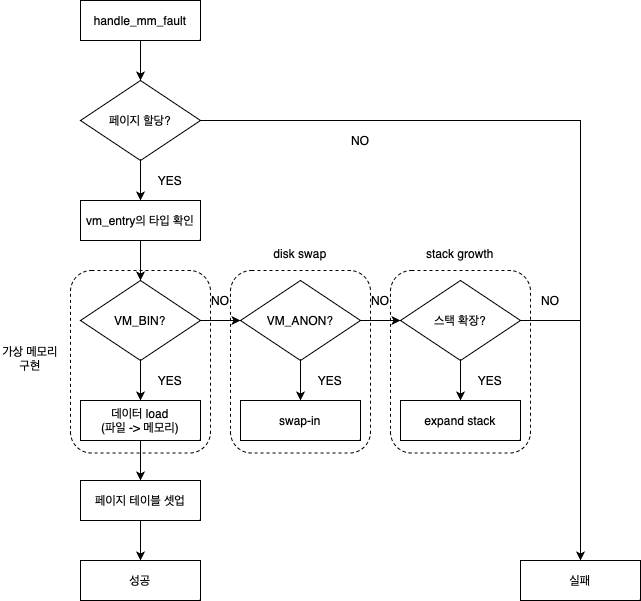
pintos/src/userprog/process.c 에 stack을 확장하는 함수 (expand\_stack)을 추가한다. 이 함수는 인자로 받는 addr 주소를 포함하도록 stack을 확장한다. 그 후 alloc\_page 함수를 통해 메모리를 할당하고, vm\_entry를 할당 및 초기화한다. 그리고 install\_page 함수를 호출하여 page table을 설정한 후, 성공 시 true를 return하도록 구현한다.

또한 stack 확장이 끝난 후에도 stack이 유효한지를 검사하는, 다시 말해서 sp 주소가 포함되어 있는지 확인하는 함수인 verify\_stack 함수를 구현한다. sp 주소가 존재할 시 handle\_mm\_fault 함수를 호출한다.

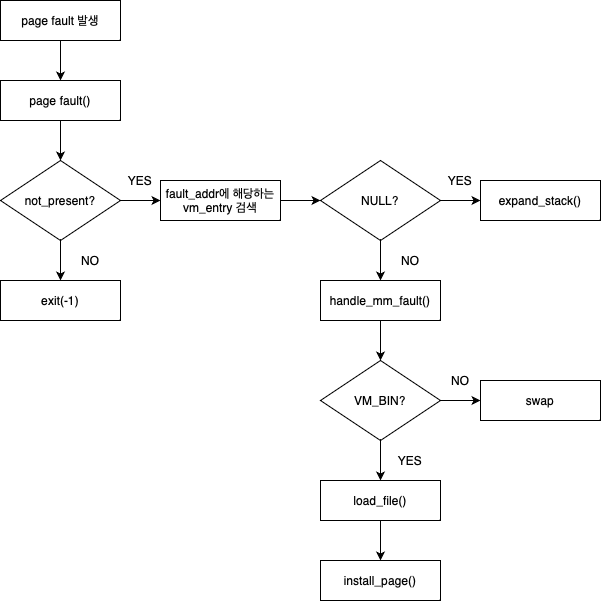
1. **연구 결과**
   1. **Flow Chart**

* 개발 내용의 각 항목에 대하여 Flow Chart 작성

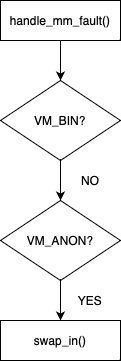
<전체적인 Flow Chart>



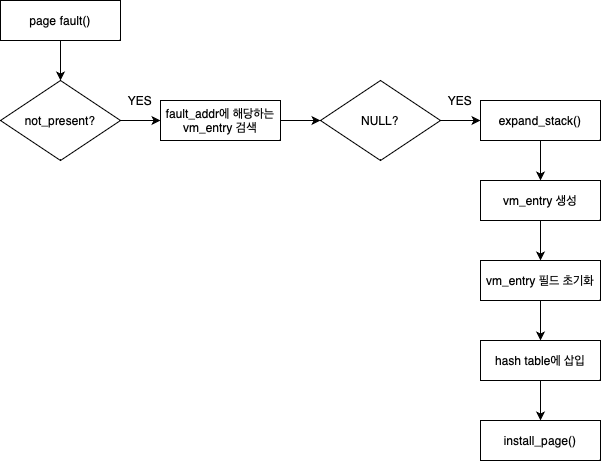
* Page Table & Page Fault Handler



* Disk Swap



* Stack Growth



* 1. **제작 내용**
* 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 table & Page fault Handler

텍스트이(가) 표시된 사진

자동 생성된 설명

위 코드는 pintos/src/threads/thread.h 파일에 구현되어 있는 코드이다.

기존 thread 구조체에 hash 구조체 vm을 추가한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 코드는 위 코드는 pintos/src/vm/page.h에 구현한다.

vm\_entry 구조체를 선언한다.

type: VM\_BIN, VM\_ANON의 타입

vaddr: vm\_entry가 관리하는 가상페이지 번호

writable: true일 경우 해당 주소에 write 가능, false일 경우 불가능

is\_loaded: 물리 메모리의 탑재 여부를 알려주는 flag

file: 가상주소와 맵핑된 파일

offset: 읽어야 할 파일의 offset

read\_bytes: 가상 페이지에 쓰여져 있는 데이터의 크기

zero\_bytes: 0으로 채울 남은 페이지의 바이트

elem: hash table element

텍스트이(가) 표시된 사진

자동 생성된 설명

위 코드는 pintos/src/vm/page.c 파일에 구현한다.

vm\_init(): hash\_init()을 사용하여 hash table을 초기화하는 함수이다.

insert\_vme(): hash\_insert()를 사용하여 vm\_entry를 hash table에 삽입하는 함수이다.

성공 시 true를, 실패 시 false를 반환한다.

delete\_vme(): hash\_delete()를 사용하여 ve\_entry를 hash table에서 제거하는 함수이다.

find\_vme(): 인자로 받은 vaddr에 해당하는 vm\_entry를 검색 후 반환하는 함수이다.

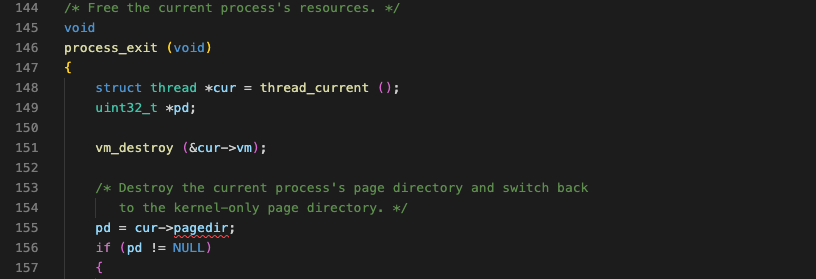
vm\_destroy(): hash\_destroy()를 사용하여 hash table의 버킷 리스트와 vm\_entry를 제거하는 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 코드는 pintos/src/userprog/process.c에 있는 코드이다.

start\_process 함수에 vm\_init()을 사용하여 hash table을 초기화한다.



process\_exit 함수에 vm\_destroy()를 사용하여 프로세스 종료 시 vm\_entry들을 제거하도록 코드를 수정한다.

텍스트이(가) 표시된 사진

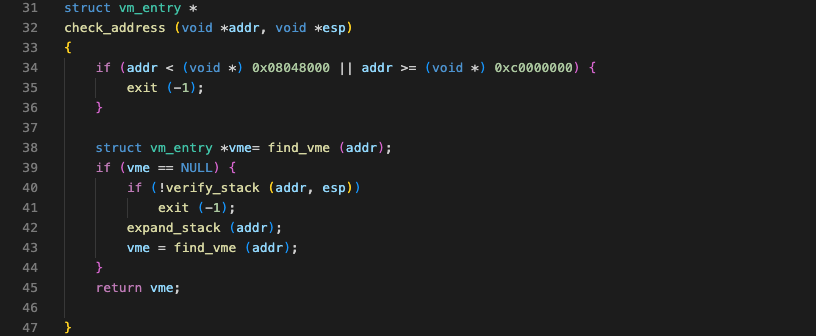
자동 생성된 설명

load\_segment 함수에 기존에 존재하던 물리 페이지를 할당하고 맵핑하는 부분을 삭제하고, vm\_entry 생성, vm\_entry 필드 초기화, vm\_entry를 hash table에 삽입하는 과정을 추가한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

setup\_stack 함수에 기존에 존재하던 페이지 할당, 페이지 테이블 설정, 스택 포인터 설정 부분 이후에, 4KB 스택의 vm\_entry 생성, 생성한 vm\_entry 필드값 초기화, hash table에 삽입하는 과정을 추가한다.



위 코드는 pintos/src/userprog/syscall.c에 존재하는 코드이다.

check\_address 함수를 구현한다. vm\_entry를 사용하여 유효성 검사 작업을 수행하도록 한다. 또한 vm\_entry를 반환하는데, 만약 addr에 해당하는 vm\_entry가 NULL이라면 stack에 대한 유효성 검사를 진행한 후, stack을 expand하는 과정 또한 추가한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

check\_valid\_buffer 함수를 구현한다. buffer를 사용하는 read() system call의 경우 buffer의 주소가 유효한 가상 주소인지 아닌지를 검사하는 함수이다. 이 때, check\_address 함수를 사용한다.

check\_valid\_string 함수를 구현한다. 이는 system call에서 사용할 인자의 문자열의 주소값이 유효한 가상주소인지 아닌지를 검사하는 함수이다.

이를 바탕으로 syscall\_handler 함수에서 기존에 사용하던 is\_user\_vaddr 함수를 모두 주석처리 한 후, SYS\_EXEC, SYS\_READ, SYS\_WRITE, SYS\_CREATE, SYS\_REMOVE, SYS\_OPEN에 새로 생성한 유효성 검사를 해주도록 수정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

pintos/src/userprog/exception.c 중 page fault 함수를 수정한다.

기존 방식인 kill (-1)에 관한 코드를 주석 처리한 후, fault\_addr의 유효성 검사, page fault handler 함수 (handle\_mm\_fault)를 호출한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 코드는 process.c에 구현한 handle\_mm\_fault 함수이다. 이 함수는 페이지를 할당하고, 데이터를 파일에서 메모리로 load한 후, page table을 셋업하도록 구현한다. 또한 swap 기능을 구현할 때, palloc\_get\_page 함수를 alloc\_page 함수로, palloc\_free\_page 함수를 free\_page 함수로 변경한다.

텍스트, 모니터, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

위 코드는 vm/page.c에 구현한 load\_file 함수이다. 이 함수는 disk에 존재하는 page를 물리 메모리로 load하는 함수이다. 4KB를 전부 write하지 못한 경우에는 나머지를 0으로 채우도록 구현한다.

* + Swap disk

텍스트이(가) 표시된 사진

자동 생성된 설명

pintos/src/vm/page.h에 page 자료구조를 추가한다.

kaddr: 페이지의 물리 주소

vme: 물리 페이지가 매핑된 가상 주소의 vm\_entry 포인터

thread: 해당 물리 페이지를 사용 중인 thread의 포인터

lru: list 연결을 위한 필드

텍스트이(가) 표시된 사진

자동 생성된 설명

pintos/src/vm/frame.c 에 다음 함수들을 구현한다.

lru\_list\_init: lru\_list, lru\_list\_lock을 초기화하고, lru\_clock의 값을 NULL로 설정하는 함수이다.

add\_page\_to\_lru\_list: LRU list의 끝에 유저 페이지를 삽입하는 함수이다.

del\_page\_from\_lru\_list: LRU list에 유저 페이지를 제거하는 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

alloc\_page: palloc\_get\_page()를 사용하여 페이지를 할당한다. page 구조체를 할당 및 초기화하고, add\_page\_to\_lru\_list()를 사용하여 LRU 리스트에 page 구조체를 삽입한다.

free\_page: 물리 주소 kaddr에 해당하는 page 구조체를 lru 리스트에서 검색하고, 매치하는 항목을 찾으면 \_free\_page()를 호출하는 함수이다.

\_free\_page: LRU 리스트를 제거하는 함수이다. page 구조체에 할당받은 메모리 공간을 해제한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

second chance 알고리즘의 LRU 리스트를 이동하는 작업을 수행하는 함수이다.

위 함수들을 구현한 후, process.c에서 사용하던 palloc\_get\_page()와 palloc\_free\_page()를 각각 alloc\_page(), free\_page()로 변경한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

pintos/src/vm/swap.c 파일에 위 함수들을 구현한다.

swap\_init: swap 영역을 초기화하는 함수이다.

swap\_in: swap out된 페이지를 다시 메모리로 적재시키는 함수이다.

swap\_out: 메모리의 내용을 디스크의 swap 영역으로 방출시키는 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

pintos/src/userprog/process.c 파일 중 handle\_mm\_fault 함수에서 VM\_ANON case를 추가하여 swap\_in을 진행시킨다.

* + Stack growth

텍스트이(가) 표시된 사진

자동 생성된 설명

pintos/src/userprog/process.c 파일에 위 함수를 구현한다.

expand\_stack: addr 주소를 포함하도록 stack을 최대 8MB까지 확장시키는 함수이다.

verify\_stack: sp 주소가 포함되어 있는지 확인하는 함수이다.

텍스트, 모니터, 화면, 은색이(가) 표시된 사진

자동 생성된 설명

그리고 pintos/src/userporg/exception.c 파일 중 page\_fault 함수에 fault\_addr이 stack 영역인지 확인 후, expand\_stack()을 호출한다.

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

텍스트이(가) 표시된 사진

자동 생성된 설명