**Pintos Project 4: Virtual Memory**

담당 교수 : 박성용 교수님

이름 / 학번 : 김채연 20181617

개발 기간 : 2020.12.05~202012.23

1. **개발 목표**

이전 프로젝트에서는 page fault가 일어나면 종료가 되었지만 이번 프로젝트에서는 page fault에 대한 handling을 추가하여 조금 더 안전성 있고 적절하게 실행되도록 핀토스를 구현하는 것을 목표로 한다. 또한 process의 수와 크기가 memory size에 제한되는 것을 virtual memory를 이용하여 제한을 없앤다.

1. **개발 범위 및 내용**
   1. **개발 범위**
      1. Page Table & Page Fault Handler

기존 page table에 필요한 정보를 더 추가한다. 정보가 추가된 page table을 통해 page fault가 발생했을 때 kernel이 page table에서 page fault가 발생한 virtual page를 찾아서 어떤 데이터가 있는지 알아내고 또한 kernel이 process가 종료될 때 free할 resource를 결정할 수 있다.

그리고 page fault handler는 memory reference가 valid하다면 page를 저장할 frame을 얻은 다음 file system이나 swap을 통해 data을 읽거나 zeroing을 통해 frame으로 가져온다. access가 유효하지 않을 때는 process를 종료하고 모든 resource를 free한다.

* + 1. Disk Swap

Swap을 구현한다. process에 할당해 줄 physical memory가 부족할 때 disk로 swap out이 일어난다. 이때 swap될 page는 page replacement algorithm을 사용한다. Swap table을 통해 사용중인 slot과 free한 slot을 관리하여 필요할 때 선택하여 사용하고 다시 free를 한다.

* + 1. Stack Growth

이전 프로젝트에서의 stack은 user virtual address 공간의 단일 page이기 때문에 programe의 stack은 제한이 되었다. 따라서 stack growth를 구현하여 현재 크기 이상으로 stack이 커지면 필요에 따라 페이지를 추가로 할당한다. 이때 stack access로 나타나는 주소에서만 page fault가 발생하면 다른 stack page를 할당하도록 한다.

* 1. **개발 내용**
     1. Page fault가 발생하는 이유와 이를 handling하는 과정

CPU가 invalid page(physical memory상에 존재하지 않는 page)에 접근을 하면 page fault가 발생한다. 따라서 page fault가 발생하면 fault\_addr을 얻어서 invalid reference인지 먼저 확인을 한다. invalid reference라면 growable region인지 다시 확인 후 아니라면 process를 종료시키고 resource를 free시킨다. growable region이라면 user stack을 확장하고 process를 다시 시작한다. 만약 valid reference라면 empty frame을 얻어서 page를 swap하여 memory로 올린 다음에 다시 process를 restart한다. 이때 remaining frame이 없다면 process replacement algorithm을 실행한다.

* + 1. Disk swap 발생시 사용한 page replacement algorithm

second chance 알고리즘을 사용한다. 이 알고리즘은 LRU 알고리즘과 비슷한 알고리즘으로 메모리에 로딩된 page의 circular list로 관리한다. 페이지가 처음 로딩된다면 1로 set 되어있다가 page fault handler에 의해서 replace되어야 할 때 해당 page의 value가 0이라면 replace한다. page가 1인 경우에는 0으로 다시 set을 한다. 그러고 circular list의 다음 page로 이동하여 value값을 다시 확인한다.

* + 1. stack growth 구현시 stack 확장 여부를 판단할 수 있는 방법

추가되는 page가 stack access로 나타나는 경우에만 페이지를 할당한다. 따라서 page fault가 일어났을 때 fault\_addr이 esp 보다 크거나 esp-4 또는 esp-32일 때 stack 이 확장되어야 한다. esp-4와 esp-32는 push 명령어에 의해 발생한 값이다. 그리고 base boundary 내에 위치 해야한다. 기본적으로 default limit은 8MB이다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

2020.12.05~2020.12.23

12/19 ~ 12/21 stack grow 구현

12/22 ~ 12/23 오류 수정 및 보고서 작성

* 1. **개발 방법**

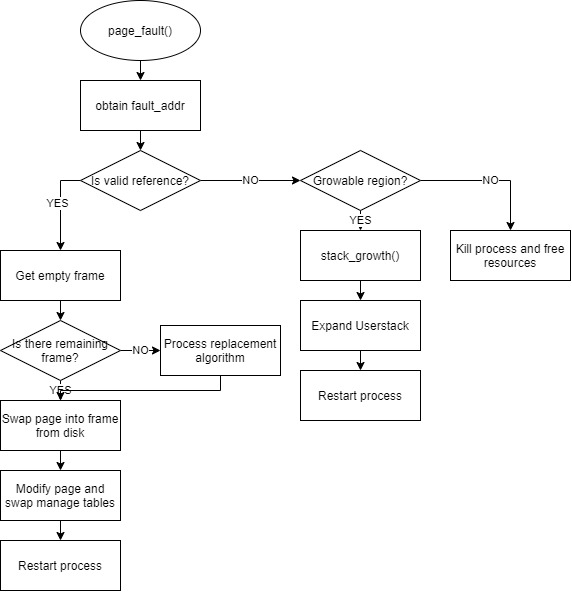
먼저 이전 project3에서 vm에서 make check를 돌렸을 때 발생하는 에러코드를 삭제한다. thread\_prior\_aging 변수와 thread\_yield() 함수에서 에러가 발생한다.

vm폴더 내에 새로운 파일을 만들고 Makefile.build에 새로 만든 파일을 추가한다.

exception.c에서 page\_fault 함수 내에서 프로세스를 바로 종료하도록 하는 코드를 지우고 page fault handling을 하는 코드를 새로 추가한다. 그리고 stack grow 함수를 호출하여 stack 확장이 필요할 때 확장시킨다.

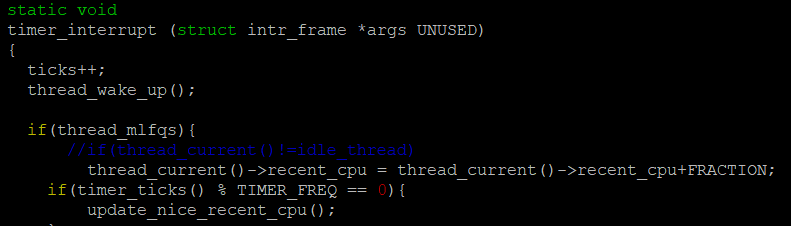
page.c page.h 에서 stack을 확장시키는 stack grow함수를 작성한다.

1. **연구 결과**
   1. **Flow Chart**



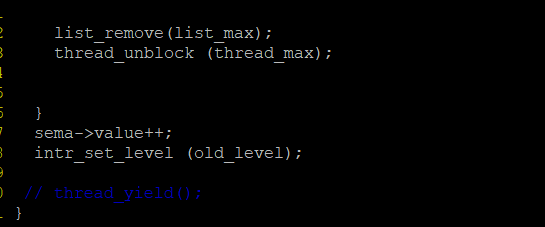
* 1. **제작 내용**
     1. device/timer.c

timer\_interrupt 함수에서 if문에서 thread\_prior\_aging 변수가 에러를 발생하므로 이 변수를 지운다.



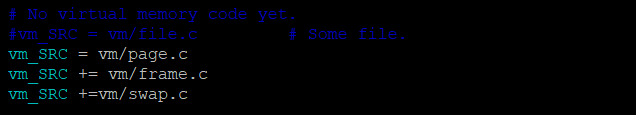
* + 1. threads/synch.c

sema\_up 함수에서 thread\_yield()를 마지막에 호출했을 때 에러가 발생하므로 호출을 지운다.



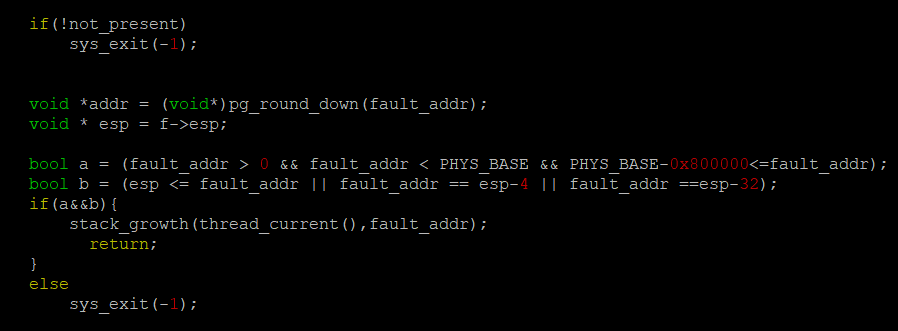
* + 1. Makefile.build

vm폴더에 새로 파일을 생성하고 해당 파일을 추가한다.



* + 1. userprog/exception.c

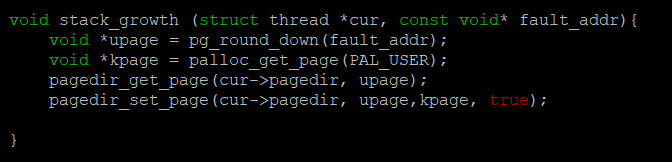
page\_fault 함수에서 sys\_exit(-1)로 프로세스를 종료하는 코드를 지우고 새로 handling 하는 코드를 추가한다.



먼저 not\_present가 false라면 writing r/o page라는 뜻이고 true 라면 not-present page라는 뜻이다. 따라서 false라면 sys\_exit함수를 호출하여 바로 종료한다. 다음 pg\_round\_down 함수를 호출하여 fault\_addr 근처의 page boundary 내림을 addr에 저장하고고 frame의 esp는 esp에 저장한다. fault\_addr에 대해서 0보다 크고 PHYS\_BASE보다 작고 base boundary보다 작은지 확인을 한다. 그리고 esp가 fault\_addr보다 작거나 fault\_addr이 esp보다 4또는 32가 작은지 확인을 한다. 이 두개의 조건이 모두 true라면 stack\_growth 함수를 호출하여 stack을 확장시키고 아니라면 sys\_exit함수를 호출하여 종료한다.

* + 1. vm/page.h, page.c

stack을 확장하는 함수를 선언한다.



현재 thread와 fault\_addr을 parameter로 받는다. pg\_round\_down 함수를 호출하여 fault\_addr 근처의 page boundary 내림을 upage에 저장하고 palloc\_get\_page 함수를 호출하여 새 single free page를 얻어서 kernel virtual address를 kpage에 저장한다. 다음 pagedir\_get\_page, pagedir\_set\_page 함수를 순서대로 호출하여 user virtual address에 일치하는 physical address를 찾고 page directory에 매핑을 추가하여 stack을 확장한다.

* 1. **시험 및 평가 내용**

