**Pintos 프로젝트 3. Pintos Virtual Memory**

**(설계 프로젝트 수행 결과)**

**과목명 : [CSE4070-02] 임베디드시스템소프트웨어**

**담당교수 : 서강대학교 컴퓨터공학과 김 영 재**

**조원 : 28조 안진우**

**개발기간 : 11월30일 ~ 12월23일**

**최 종 보 고 서**

**프로젝트 제목: Pintos 프로젝트 3. Pintos Virtual Memory**

**제출일: 2017.12.23**

**참여조원: 28조 안진우**

**I. 개발 목표**

이번 프로젝트에서는 virtual memory에서 프로세스의 갯수와 크기 제한을 없애서 큰 용량의 여러개의 프로세스가 원활히 핀토스위에서 돌아가도록 하는 것이 그 목적이다. 그동안 user process, thread 프로젝트를 거치며 완성된 핀토스는 여러 thread의 동시 수행 및 동기화 기능이 향상 되었으나, physical memory의 size에 의해 크기 및 개수가 제한되는 약점이 있었다. 이번 프로젝트에서는 이를 해결하기 위해 virtual memory기능을 추가하여, swap disk를 통한 swap in and out, page table management, stack growth기능을 추가하고, page fault handler함수에서 physical memory에 할당되지 않은 영역을 자동적으로 disk로부터 load할 수 있게 하였다.

**II. 개발 범위 및 내용**

1. **개발 범위**

이번 개발 범위는 주로 page fault핸들러 함수에서의 memory와 disk간의 swap in and out이다. 이를 위해 page fault 핸들러 함수에 상당한 코드를 추가하였으며, vm 디렉토리에서 swap을 위한 여러 테이블과 함수를 새로 만들었다. 과정은 page fault handler함수의 코드추가를 시작으로 TOP DOWN방식으로 세부적 모듈을 개발해나갔다.

**나. 개발 내용**

이번 개발은 page fault함수에서 전체적인 flow를 짜고, 그에 맞춰 vm디렉토리에 새로운 모듈을 추가해 나가는 TOP DOWN방식으로 진행하였다.

1. page fault handler 함수에서의 전체 flow생성

page fault handler함수는 이번 프로젝트에서 핵심적인 영역이다. 유저 메모리에서 physcial memory에 접근할 수 없을 경우에 이 함수가 호출되며, 함수 내에선 invalid / valid 여부를 확인하여 invalid 할 경우에는 (기존 코드) process를 kill하여 종료시킨다. 중요한 것은 valid한 경우인데, 이 경우는 user process의 data가 physical memory가 아닌 swap disk 또는 file system에 저장되어 있는 경우이다. 따라서, 현재 메모리의 빈 공간을 확인하고, 없을 경우 새로이 할당하여 (swap out), 디스크로부터 해당 위치로 데이터를 가져오는 (swap in) 방식이 구현되어야 한다. 이와 관련된 세부 구현은 VM 디렉토리에서 진행하도록 하였다.

2. stack growth

stack growth는 유저 스택메모리의 제한을 훨씬 확장하는 것이 목적이다. 기존의 구현된 핀토스의 경우, stack공간은 프로세스가 생성될 때 단 한번의 page allocation을 하기 때문에 크기가 4KB로 크게 제한되는 문제가 있었다. 이를 해결하기 위하여 처음부터 stack공간을 크게 설정할 수 있겠으나, 사용하지 않는 스택공간에 대한 over head가 발생한다. 따라서, stack overflow때의 발생한 page fault에서 stack 을 reallocation시키는 것이 가장 바람직하다. 이를 위해 page fault 핸들러 함수에서는 발생한 page fault의 원인이 stack overflow인지 확인하고, stack에 새 페이지를 할당하도록 구현하였다. stack의 최대 크기는 8MB로 제한하였다.

**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

|  |  |
| --- | --- |
| 11/30 ~ 12/5 | pintos 메뉴얼 분석 및 코드 이해 |
| 12/5 ~ 12/8 | page fault 플로우 생성 및 stack growth 구현 |
| 12/8 ~ 12/20 | 시험기간 - 구현 중단 |
| 12/21 ~ 12/23 | 주석 처리 및 최종 보고서 작성 |

**나. 개발 방법**

이번 프로젝트에서는 virtual memory management의 핵심이 되는 page fault handler함수로부터 시작하여 전체 flow를 짜고, TOP DOWN방식으로 세부적인 모듈을 생성하는 방법으로 구현하였다. page fault의 flow는 핀토스 실습자료를 참고하였다. 또 stack growth의 기능은 exception.c 에서 구현되도록 하였으며, 나머지 모든 VM관련 기능은 VM디렉토리에서 구현하였다.

1. page fault handler 함수에서의 전체 flow생성

page fault핸들러 함수는 이번 virtual memory에서 가장 핵심적이고 중추적인 기능을 하는 함수이다. 왜냐하면 모든 user memory의 memory 문제는 page fault함수에서 처리되기 때문이다. 특히, user memory에서 접근하는 데이터가 physical memory에 없을 경우 swap disk나 file system 으로부터 가져와야 하는데 이 기능의 시작점이 page fault handler라는 점에서 가지는 의의가 더욱 크다. 페이지 폴트 함수 flow는 아래 챕터4-1-1.1 프로그램 실행 구상도에 있으며 그 흐름은 다음과 같이 동작한다.

우선 페이지 폴트가 발생한 경우 가장 먼저 확인할 것은 valid / invalid 여부이다. page table의 page entry에는 error code가 저장되기 때문에 이를 통해 이를 알 수 있으며, invalid할 경우에는 종료시키고 valid할 경우에는 다음으로 stack 영역인지를 체크한다. stack 영역일 경우 stack에 한페이지를 더 할당하여 종료하며, 스택 영역이 아닐 경우에는 handler\_mm\_fault함수를 호출한다. 해당 함수에서는 swap in and out을 위한 기능이 구현된다. 우선 physical memory내에 empty frame이 존재하는 지 확인하고, 없을 경우 page replacement algorithm을 통해 swap out시킨다. 이후, 할당된 frame에 디스크로부터 해당 데이터를 가져오며, page table을 수정하여 리턴한다. 이 부분은 exception.c에서는 전체적인 flow를 잡고, 세부 모듈은 VM디렉토리에서 생성되도록 하였다.

2. stack growth

stack growth는 유저 접근이 스택 영역인 경우 새로운 page를 할당하여 확장된 스택공간을 유저에게 제공하는 것이 그 목적이다. 따라서 여기서는 기존의 4kb크기였던 스택 공간을 pintos 에서 명시한 8MB까지 dynamic하게 lazy한 방식으로 필요할때 할당하도록 구현하였다. palloc\_get\_page를 통해 할당된 스택 페이지는 install\_page를 통해 page table에 재등록 되도록 하였으며, 이를 위해 process.c에만 사용되도록 국한된 install\_page를 전역적으로 사용할 수 있도록 수정하였다. 스택 할당 공간을 8MB까지 제한하기 위하여 최초의 스택공간 PHYS\_BASE로부터 PHYS\_BASE\*8MB까지만 할당할 수 있도록 제한하였다.

**다. 연구원 역할 분담**

1. 핀토스 메뉴얼 분석 및 코드 이해 : 안진우 100%

2. 프로그램 생성 및 구현 : 안진우 100%

3. 테스팅, 에러 처리 및 주석, 보고서 : 안진우 100%

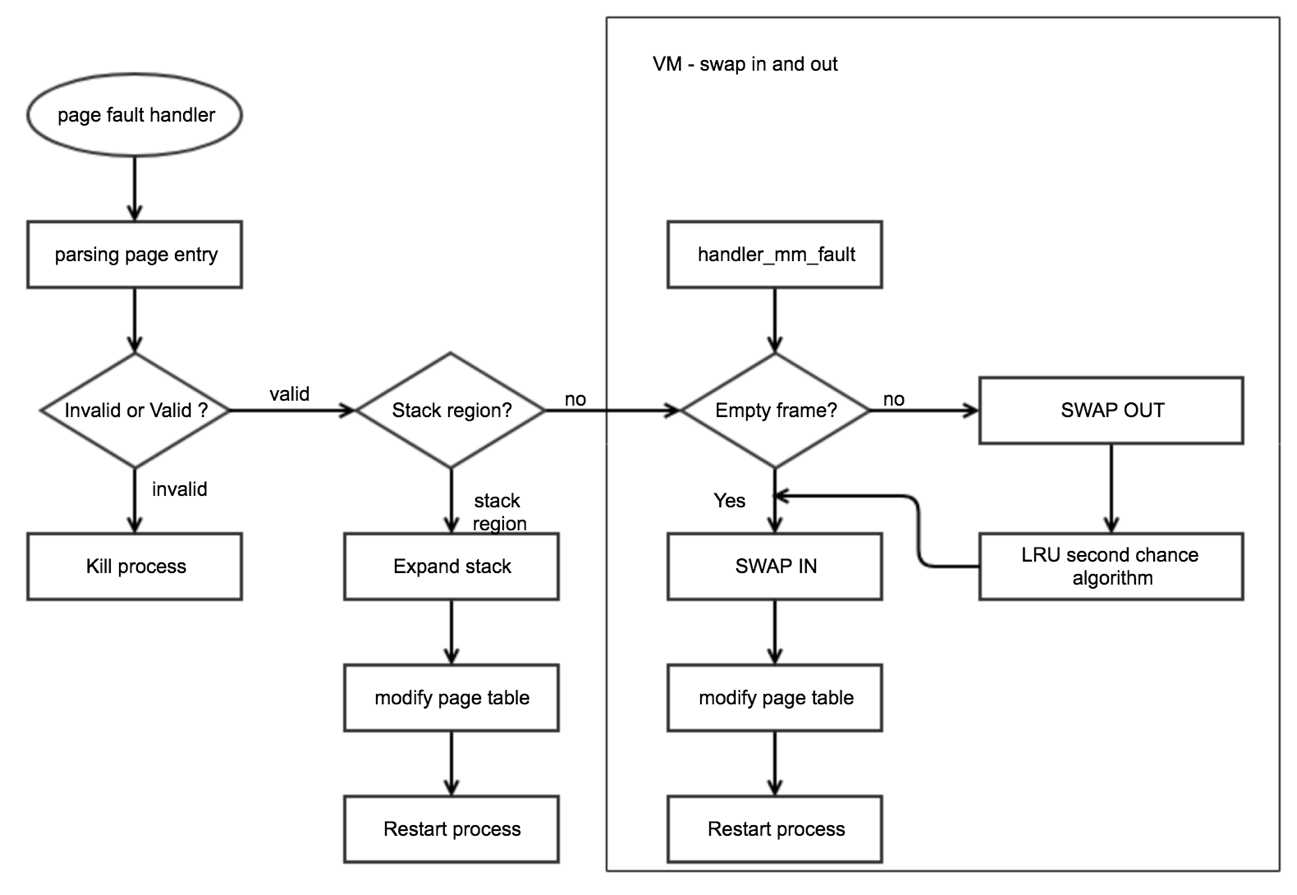
**IV. 연구 결과**

1. 합성 내용:

1.1. 프로그램 실행 구성도

1. Memory management flow

이번 프로젝트에서 구현한 memory management의 전체 flow는 다음과 같다. Page fault handler는 모든 메모리 관리의 시작점이자 중추이며, 해당 함수로부터 메모리관리 기능이 관리된다. 구현은 크게 두부분으로 page fault에서의 전체 flow관리 및 stack growth기능 / swap in and out기능으로 나뉜다. 앞의 기능은 exception.c에서 모듈을 통해 구현하였으며, 뒤의 기능은 vm디렉토리에서 새 파일을 생성하여 구현하였다.



2. 제작 내용: 개발 결과

1. . page fault handler 함수에서의 전체 flow

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| exception.c 에서는 page fault에 관련된 전체 memory management의 flow을 구성하였다. 구성된 flow는 프로그램 구상도 1 과 같으며, page fault의 원인을 확인하기 위하에 user에서 보낸 page table entry를 확인하였다. 핀토스에서의 page table entry의 구조는 다음과 같다.   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Physical Address | AVL |  | D | A |  | U | W | P |   우선 page fault handler에서는 메모리 접근의 타당성을 검증해야 한다. 이를 위해 사용되는 entry bit가 U, W, P이다. 각 비트는 다음과 같은 의미를 가진다.  P : present bit. page table entry가 실제 메모리 주소를 가리키지 못하는 경우라고 보면 타당하다. swap disk / file system에 저장되어 발생하는 경우가 있고, invalid해서 발생하는 경우가 있다.  U : User/Privileged bit. 메모리 접근이 유저에 의한것인지 커널에 의한 것인지 나타내는 비트이다.  W : read / write bit. 현재 접근하려는 페이지가 writable한지, readible한지 나타내는 비트이다.  해당 비트는 page fault handler에서 interrupt변수인 intr\_frame에 error code로 존재한다. 따라서 핀토스에서는 다음과 같이 에러코드로부터 해당 엔트리를 불러온다.  스크린샷%202017-12-22%20오전%2010.55.11.png  그리고 가지고 온 엔트리는 다음과 같이 사용되어 페이지폴트 플로우를 구성한다.  스크린샷%202017-12-22%20오전%2010.54.05.png  우선 not present이지만, writable한 페이지고, user가 접근한 경우는 정상 접근으로 간주할 수 있다. 반대로 그렇지 않으면, 잘못된 접근으로써 process를 kill하고 종료해야 한다.  올바른 접근의 경우는 두가지 flow를 거치는데, stack영역인지 확인해야 한다. stack 영역일 경우 stack을 위한 page를 할당하여 종료하고, 그렇지 않으면 swap disk또는 filesystem 으로부터 swap in하는 flow를 구현한다. |

2. stack growth

|  |
| --- |
| stack growth에서는 유저가 스택영역의 접근을 요청했으나, 메모리 overflow가 발생한 경우에 의한 페이지폴트 처리를 구현한다. 그러기 위해선 가장 먼저, user의 접근이 stack영역인지 아닌지 구분해줘야 한다. 이를 구분하기 위해 stack\_growable함수를 생성하여 다음과 같이 처리하였다.  스크린샷%202017-12-22%20오전%2011.01.12.png  user stack영역의 최초 virtual address는 PHYS\_BASE이며 음의 방향으로 stack이 쌓인다. 스택공간은 최대 8MB로 제한되므로 해당 공간보다 작을 경우에 true를 리턴한다.  여기서 stack\_page는 현재 stack에 할당된 페이지수를 나타내는 중요한 역할을 하는 변수이다. 여기에서는 현재 stack주소범위를 잡아주기 위한 용도로 사용된다. 추가적인 설명은 아래에서 추가한다.  stack region으로 판명났을 경우 stack\_allocate함수를 이용하여 새로운 페이지를 할당하도록 다음과 같이 구현하였다.  스크린샷%202017-12-22%20오전%2011.13.57.png  우선 현재 stack에서 요구하는 페이지가 한개가 아닐 수 있음을 염두해야 한다. 스택에서는 거대한 array등으로 한번에 많은 페이지를 요구할 수 있는데, 이를 위해 할당해야할 페이지수를 먼저 구해야 한다. 여기서 사용되는 것이 stack\_page로, stack\_page로 이전 instruction까지 할당되어있는 현재 스텍 주소를 구하고, 이와 요구되는 새로운 스택공간의 주소의 차를 이용하여 할당할 페이지의 수를 정할 수 있다. 페이지 수를 일단 정하고나면, 새로운 페이지를 palloc\_get\_page를 이용하여 할당하고, 할당된 페이지를 page mapping table로 맵핑시킨다. 이를 위해 install\_page함수가 사용된다. install\_page는 기존 pintos의 경우 process.c에서만 사용할 수 있는 내부 함수로 정의되어 있다. 이를 수정하여 exception.c에서도 사용할 수 있도록 변경하였다. |

3. 시험 및 평가 내용:

-평가방법

1. 테스트 케이스 수행하기 (make clean 상태일때)

1. $cd pintos/src/vm
2. $make
3. $cd build
4. $make check
5. 22개의 테스트케이스가 통과하였는지 확인한다.
6. run -v -- -q run priority-lifo를 통해 적절한 출력인지 확인한다. (뒤에 내용 참조)

-보건 및 안정, 생산성과 내구성

이번 프로젝트에서는 기존에 제약되어 있던 process의 메모리 크기와 갯수를 해제하여 다양한 크기의 많은 프로세스가 동시에 메모리를 점유할 수 있도록 하여 생산성을 확장시키면서 동시에, 이로인해 발생할 수 있는 여러 에러에 대처하여 보건성 및 내구성을 향상시키고, 안정성있는 핀토스를 구현하는 것이 가장 큰 목적이었다.

이를 위해 기존의 무조건 process를 kill하도록 설계된 page fault함수를 수정하여 stack growth기능을 구현하여 스택 사이즈의 제한을 4kb에서 8MB로 크게 확장하여 생산성을 크게 높이고, 스택 오버플로우에 대한 내구성을 향상 시켰다.

다음으로 중요한 것이, 이렇게 확장된 기능을 가진 핀토스가 안정성과 보건성을 가지고 지속적으로 돌아가도록 하는것이었는데, 이를 위해 page fault handler에서는 page table entry의 error code를 이용하여 각 case를 세세하게 나누고 그에 따른 모듈을 구현하였다. 이를 테스트 한 결과 stack과 관련된 모든 모듈이 정상적으로 작동하였으며, 이를 통해 보건성과 안정성의 향상을 도모하였다.

**V. 기타**

- 기타 관련 내용을 기술할 것.

1**. 연구 조원 기여도:**

안진우: 100%

2. 기타 본 설계 프로젝트를 수행하면서 느낀 점을 요약하여 기술하라.

VM은 이번 핀토스의 마지막 과제였다.

그동안 핀토스 프로젝트를 통해 OS를 이론적으로 배울 뿐 아니라, 코드레벨로 분석하고, 수정하고, 생성하는 과정으로 OS의 내부 깊은 면 까지 이해할 수 있는 좋은 기회를 가졌다고 생각한다. 그리고 핀토스 프로젝트가 서강대학교에 있다는 사실만으로 서강대가 시스템 교육에 있어서 어떤 학교와 비교해도 더 양질의 교육을 제공하고 있다고 생각한다. 핀토스 프로젝트는 시스템 개발자가 되고 싶은 입장에서, 양질의 입문이었고 단기간에 실력향상을 꾀하기 가장 좋은 방법이었다고 생각한다.

하지만 아무것도 모르는 3학년이 핀토스 프로젝트를 시작할 경우 커다란 장벽을 넘기기 아주 어렵다는 사실은 아쉬운 점이라 생각한다. 실제로 동기 친구들의 경우, 본인이 핀토스에서 커널을 만졌는지조차 모르는 경우가 다수였다. 이 문제를 해결하기 위해서는 시스템 프로그래밍 수업등으로 미리 커널을 맛보게 하는 기회를 제공하거나, 임베디드 시스템 수업을 운영체제 이후의 수강과목으로 두지 않는 것도 좋은 방법이 될 것 같다는 생각이 들었다. 핀토스는 에뮬레이터이기 때문에 사실상 커널을 다루면서도, 임베디드 소프트웨어 수업에서처럼 커널을 다룬다는 직관이 크게 들지 않는다. 학생들이 os를 수강하기 전에 어떤 방식으로든, 실제 커널을 조금이나마 다뤄본다면 흥미가 생겨서 핀토스 프로젝트를 진행할 원동력이 커질 것이란 생각이 들었다.