**Pintos Project 4: Virtual Memory**

담당 교수 : 김영재 교수님

이름 / 학번 : 조현지 / C111181

개발 기간 : 12/17 ~ 12/22

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술

**가상 메모리 시스템과 page frame 관리 시스템을 구현한다. 기존의 순차적 page frame 할당 대신 프로세스의 page table을 바탕으로 page fault handling을 구현한다. 이때, page frame pool에 second change clock algorithm을 구현한다.**

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

프로세스에 대한 page table 구조체를 구현한다. 프로세스는 page table을 바탕으로 물리 메모리와 매핑이 된다. 프로세스가 page table을 통해 접근하려는 페이지가 존재하지 않는 경우 page fault가 발생하며, page fault handler를 바탕으로 swapping 등의 상황에서 자원을 할당할 수 있게 한다..

* 1. Disk Swap

프로세스가 요구하는 메모리 크기가 물리 메모리보다 클 경우에 해당한다. 자주 사용하지 않는 페이지를 page replacement algorithm 중 하나인 second change clock algorithm에 따라 선택한 뒤 페이지를 교체한다. Page fault handler가 담당하는 여러 동작 중 하나에 속한다.

* 1. Stack Growth

프로세스는 실행 도중 기존에 할당 받은 스택 페이지 크기보다 커질 수 있다. 따라서, 동적으로 스택 영역을 확장하기 위한 기능이 필요하다. Page fault handler가 담당하는 여러 동작 중 하나에 속하며 가상 메모리 시스템 구현을 위해 필요한 기능이다.

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

Page fault는 프로세스가 존재하지 않는 페이지에 접근할 경우 발생한다.

* + - * 발생 이유에는 크게 4가지가 있다. 첫번쨰, binary file이 아직 물리적 메모리(Disk)에 로드되지 않은 경우가 있다. 두번째, 프로세스가 요구하는 가상 메모리 공간이 물리적 메모리 공간의 크기를 넘어섰을 때 page replacement algorithm에 따라 스왑 영역으로 페이지가 swap out 됐을 수 있다. 이때 페이지를 다시 swap in 시켜주는 로직이 필요하게 된다. 세번째, 기존에 할당한 스택 영역보다 스택의 영역이 커졌을 경우에 대해 동적으로 페이지가 할당되지 않았을 경우이다. 마지막으로, 잘못 참조된 메모리 주소에 프로세스가 접근하려고 하는 경우이다.
  1. Disk swap 발생 시 사용한 page replacement algorithm에 대해 서술

Page frame pool의 Queue 구조체에 second chance clock algorithm을 구현한다.

* + - * Place replacement algorithm은 가상 메모리 영역이 물리적 메모리 영역의 크기를 넘어섰을 경우 필요하다. 프로세스가 필요한 페이지를 로드하려고 하지만 frame pool에 빈 페이지가 없을 경우, victim pointer가 가리키는 페이지를 swap out 시켜 페이지를 비운 뒤 새로운 페이지로 교체한다.
      * 각 페이지가 가지는 reference 1 bit를 가지며 디폴트는 1이다. Victim pointer는 원형 큐를 순회하며 pointer가 가리키는 페이지의 bit가 1이면 0으로 바꾼다. 만약 bit가 0일 경우, 새로운 페이지 할당을 위한 victim page로 선정돼 교체된다.
  1. Stack growth 구현 시 stack 확장 여부를 판단할 수 있는 방법에 대해 서술

프로세스가 참조하려는 주소가 사용자 스택 영역 안에 있는지 확인한다.

* + - * 스택은 동적으로 할당 가능한 영역이지만 PHYS\_BASE 이내의 주소에 대해서만 확장 가능하다. 따라서, 프로세스가 요청하는 추가적인 스택 페이지가 PHYS\_BASE(사용자 스택 영역)을 벗어나 커널 영역을 침범하는지 확인해야한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

|  |  |
| --- | --- |
| 2024.12.17 ~ 2024.12.20 | Page table, Page 구조체 구현 |
| 2024.12.20 ~ 2024.12.22 | Stack growth 구현 시도 |

* 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

|  |
| --- |
| **Pagedir.c** |
| **[page table - 수정 사항]**  - page table 구조체 구현  - page table에 hash 구조체를 추가해, page table 내의 모든 page table entry를 관리할 수 있게 함수를 수정  - page table entry는 page fault handling을 위해 page fault가 발생 가능한 4가지 type 의 상태 중 하나를 가짐 |
| **frame.c** |
| **[Disk swapping- 수정 사항]**  - frame pool 구현을 위해 전역 변수로 queue 구조체 구현  - queue에 대해 second chance clock algorithm 구현  - swap in & out 구현을 위해 swap file 구조체 구현  - block.c 파일의 함수를 이용해 block에 대해 rw를 진행 |
| **Exception.c** |
| **[Page fault handler - 수정 사항]**  - page fault가 발생 가능한 4자기 경우에 따라 각기 다르게 동작한다. Page가 어떤 경우에 해당하는지는 PTE의 page type 변수 값에 따라 판단한다.  - reference bit가 valid이면, disk swapping을 진행  - unvalid reference bit지만 PHYS\_BASE 이내의 page fault address 였다면 스택 확장으로 판단한 뒤, 동적으로 할당한다. |

1. **연구 결과**
   1. **Flow Chart**

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

Page table & Page fault handler

|  |  |
| --- | --- |
| 1 | Page fault가 발생하여, page fault handler에 진입해 프로세스의 page table에 접근한다. |
| 2 | Page table 내의 PTE 중 fault가 발생한 fault address가 포함된 페이지가 있는지 확인한다. |
| 3 | 해당되는 페이지가 존재하지 않는 경우, 스택 확장 여부를 확인한다. 만약 스택 확장에 해당된다면 스택 영역에 대한 추가적인 페이지를 동적으로 할당한다. 아니라면 잘못된 주소에 접근했다고 판단하고 프로세스를 강제 종료한다. |
| 4 | 해당되는 페이지가 존재하는 경우, 페이지의 reference bit가 valid한지 확인한다. unvalid하다면, 프로세스가 요구하는 페이지가 현재 메모리에 없기에 frame pool에서 second chance clock algorithm을 바탕으로 자주 사용되지 않은 페이지 영역과 교체한다. |
| 5 | PTE의 데이터를 update한 다음 프로세스가 다음 명령어를 실행하게 한다. |

Disk Swap

|  |  |
| --- | --- |
| 1 | Frame pool에 새로운 물리적 페이지에 대해 할당 시도 |
| 2 | 만약 frame pool에 비어 있는 페이지가 존재할 경우, 새로운 frame를 할당한다. Frame pool에 빈 공간이 없다면 page replacement algorithm 중 하나인 second chance clock algorithm을 수행한다. |
| 3 | Frame pool의 victim pointer가 큐를 원형 순회하며 reference bit를 확인한다. |
| 4 | 만약 pointer가 가리킨 frame의 reference bit가 1이면 0으로 교체한 뒤 다음 frame으로 넘어간다. Reference bit가 0일 경우에는, 해당 frame에 대한 페이지를 스왑 영역으로 swap out 한 뒤, 새로운 페이지를 빈 frame에 할당한다. |
| 5 | 스왑된 페이지는 swap table에 기록해 이후에 있을 swap in에 대비한다. |
| 6 | 이후 프로세스를 재개한다. |

Stack Growth

|  |  |
| --- | --- |
| 1 | Page fault가 발생한 주소가 프로세스의 사용자 스택 영역에 해당되는지 확인한다. |
| 2 | PHYS\_BASE 이내에 존재할 경우, 스택 확장이 가능한 가상 메모리 영역이기에 새로운 페이지를 할당한다. |
| 3 | 이때도, 물리 메모리 할당을 위한 frame pool 영역에 빈 공간이 없다면 disk swapping을 진행한다. |
| 4 | 이후 PTE 구조체를 생성한 뒤 이를 해당 프로세스의 page table에 추가한다. |
| 5 | 프로세스를 재개한다. |

* 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

|  |  |
| --- | --- |
| **Pagedir.c** | |
| **[page table - 수정 사항]**  - page table 구조체 구현  - page table에 hash 구조체를 추가해, page table 내의 모든 page table entry를 관리할 수 있게 함수를 수정  - page table entry는 page fault handling을 위해 page fault가 발생 가능한 4가지 type 의 상태 중 하나를 가짐 | - enum pte\_type : page 타입  - struct page\_table : PT 구조체  - struct page\_table\_entry : PTE 구조체  : 프로세스 별로 page table을 할당해준다. 이후, page table에 유효한 접근이 요청될 경우 새로운 페이지를 할당해준다. |
| **frame.c** | |
| **[Disk swapping- 수정 사항]**  - frame pool 구현을 위해 전역 변수로 queue 구조체 구현  - queue에 대해 second chance clock algorithm 구현  - swap in & out 구현을 위해 swap file 구조체 구현  - block.c 파일의 함수를 이용해 block에 대해 rw를 진행 | - static struct list frame\_table : frame pool 변수  - static struct list\_elem \*clock pointer : 새로운 페이지를 위해 교체될 페이지를 가리키는 victim pointer 역할  : frame pool에 빈 공간이 있으면 frame\_allocate 함수를 통해 물리적 페이지를 할당한다. 만약 frame pool에 빈 공간이 없을 경우 clock pointer를 이용해 frame pool 내의 큐를 원형 순회한다. Second chance clock algorithm에 의해 reference bit를 1->0을 교체해나가면서 reference bit가 0인 페이지 중 가장 먼저 pointer에 의해 지목된 페이지의 frame 영역을 free 시킨다. |
| **Exception.c** | |
| **[Page fault handler - 수정 사항]**  - page fault가 발생 가능한 4자기 경우에 따라 각기 다르게 동작한다. Page가 어떤 경우에 해당하는지는 PTE의 page type 변수 값에 따라 판단한다.  - reference bit가 valid이면, disk swapping을 진행  - unvalid reference bit지만 PHYS\_BASE 이내의 page fault address 였다면 스택 확장으로 판단한 뒤, 동적으로 할당한다. | 페이지 폴트가 난 주소를 가져온다. 폴트가 난 주소가 포함된 PTE가 해당 프로세스의 page table에 존재하는지 해시 구조체에서 확인한다. 만약 존재할 경우, page의 valid bit가 0이라면 swap table에 P페이지가 존재하기 때문에 disk swapping을 진행한다.  PTE가 존재할 경우, 페이지 폴트 발생 주소가 사용자의 스택 영역 내인지 커널 영역인지 확인한다. 스택 영역은 동적으로 페이지 할당이 가능하지만 사용자 스택 영역인(PHYS\_BASE)를 넘을 수 없기에, 이에 대해 주소가 유효한지 확인하는 절차가 필요하다.  그외의 page fault는 잘못 접근된 주소로 판단하고 프로세스를 강제 종료시킨다. |

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

