#HW3

김태우 & 원상혁

1. 설명
2. Virtual Memory
   1. 과제 목표

현재 핀토스는 실행파일이 탑재될 때 전체 프로세스 주소공간이 할당되고, 주소 공간 각 페이지의 물리적 주소가 결정되는데, 그 결과 Swap과 Demand paging 사용이 불가능하다. 이러한 기능들을 가능하게 만들기 위한 Virtual Memory를 구현하는 것이 이번 과제의 목표이다.

* 1. 개념

가상주소 공간은 세그먼트들의 집합으로, 세그먼트는 VM페이지들의 집합이다.

Demand Paging 과정은 1.명령을 수행하면, 2.가상주소를 물리주소로 변환시키고, 3. 해당 page가 존재하지 않아 페이지 폴트를 발생시켜, 4. Page를 디스크에서 4KB만큼 읽어 물리메모리에 탑재하는 순서로 이루어진다.

* 1. 함수 구현
     + bool handle\_mm\_fault ( struct vm\_entry \*vme )

물리메모리를 할당해주고, vm\_entry의 타입 중 VM\_BIN의 경우를 처리해주었습니다.

load\_file함수를 사용하여 물리메모리에 로드하고, install\_page 함수를 이용해 물리페이지와 가상페이지의 맵핑을 해주었고, 성공 여부를 반환해주도록 구현했습니다.

* + - bool load\_file ( void\* kaddr, struct vm\_entey \*vme)

file\_read\_at함수를 사용해 물리페이지에 read\_bytes만큼 데이터를 써주고, zero\_bytes만큼 남는 부분을 memset함수를 사용해 0으로 패딩해주고, 정상적으로 로드되었는지 여부를 반환해주도록 구현했습니다.

* + - static bool load\_segment ( struct file \*file, off\_t ofs, unit8\_t \*upage, uint32\_t read\_bytes, uint32\_t zero\_bytes, bool writable )

물리 페이지를 할당하고 맵핑하는 부분을 삭제하고, vm\_entry 를 malloc을 사용해 생성하고, vm\_entry의 각 멤버들을 설정하도록 구현해주었고, insert\_vme 함수를 사용해 생성한 vm\_entry 를 해시테이블에 삽입해주었습니다.

* + - static bool setup\_stack ( void \*\*esp )

load\_segment 함수와 마찬가지로 vm\_entry 를 생성하고, 각 멤버를 설정하고, insert\_vme 함수를 사용해 해시테이블에 삽입해주었습니다.

1. Memory Mapped File
   1. 과제 목표

프로세스 주소공간에 파일을 맵핑, 메모리 접근(load/store)을 통해 파일을 접근하도록 구현한다.

현재 핀토스에는 mmap 함수와 munmap 함수가 구현되어 있지 않기에 이번 과제에서 함수를 구현한다.

* 1. 개념

파일 맵핑순서는 1. vm\_entry를 생성하고 2. 해시테이블에 삽입하고, 3. Demand paging 하여 파일을 맵핑한다.

* 1. 함수 구현
     + int mmap ( int fd, void \*addr )

인자 체크 후 가능 여부에 따라 가능한 경우, file\_reopen 함수를 사용하여 파일을 열고, 현재 mapid를 할당시켜주었습니다. 그리고 mmap\_file 을 생성하고 파일에 길이에 동안 vm\_entry 를 생성시켜주고 mapid를 리턴시켜주도록 구현했습니다.

* + - void munmap ( mapid\_t mapid )

mmap\_list 를 순회하면서 인자로 받아온 mapid와 일치하는 것이 있는 지 체크하고 일치하는 경우, 해당 vm\_entry 와 mmap\_file를 제거하고, file\_close 함수를 사용하여 해당 파일을 닫아주도록 구현했습니다.

1. Swapping
   1. 과제 목표

페이지 교체 정책을 구현하고 스와핑을 구현한다.

본 과제에선 pseudo LRU 방식인 clock 알고리즘을 사용하여 페이지 교체를 구현한다.

* 1. 개념

메모리 내용을 디스크로 방출 및 메모리에 탑재하는 기법으로, 핀토스는 스왑 파티션을 스왑 스페이스로 제공한다.

* 1. 함수 구현
     + struct page\* alloc\_page ( enum palloc\_flags flags )

palloc\_get\_page 함수를 사용해 페이지를 할당하고, page 구조체를 할당한다. add\_page\_to\_lru\_list 함수를 사용해 LRU리스트에 page구조체를 삽입하고 page구조체의 주소를 리턴하도록 구현했습니다.

* + - void free\_page ( void \*kaddr )

물리 주소 kaddr에 해당하는 page구조체를 LRU리스트에서 검색하고, 매치하는 경우 \_\_free\_page 함수를 호출하도록 구현했습니다.

* + - static struct list\_elem\* get\_next\_lru\_clock ( )

Clock알고리즘의 LRU리스트를 이동하는 작업을 수행하는, LRU리스트의 다음 노드의 위치를 반환해주도록 구현했습니다.

* + - void swap\_in ( size\_t used\_index, void\* kaddr)

used\_index의 swap slot의 저장된 데이터를 논리 주소 kaddr로 복사해주도록 구현했습니다.

* + - size\_t swap\_out ( void\* kaddr )

kaddr 주소가 가리키는 페이지를 스왑 파티션에 기록, 페이지를 기록한 swap slot번호를 리턴해주도록 구현했습니다.

1. Stack
   1. 과제 목표

현재 스택의 크기는 4KB로 고정된 상태이지만, 현재 스택의 크기를 초과하는 주소에 접근이 발생한 경우, 유효한 스택 접근인지, 세그멘테이션 폴트인지 판별하는 휴리스틱을 적용하도록 코드 수정

* 1. 개념

페이지 할당 여부를 확인하고, vm\_entry 의 type별로 체크를하여 스택 확장인 경우 expand\_stack 함수를 사용해 스택을 확장시킨다.

* 1. 구현 함수
     + 구현하지 못했습니다.

1. Priority Scheduling and Synchronization
   1. 과제 목표

여러 쓰레드가 lock, semaphore, condition variable 을 얻기 위해 기다릴 경우 우선 순위가 가장 높은 쓰레드가 CPU를 점유하도록 구현한다.

* 1. 개념

우선순위를 무시하고 waiters list 에 삽입되는 순서대로 lock를 획득하는 것이 아닌 Semaphore 를 요청할 때 대기 리스트를 우선순위로 정렬하도록 수정한다.

* 1. 구현 함수
     + 구현하였으나, vm구현이후 에러 발생…
     + void sema\_down ( struct semaphore \*sema )

semaphore를 얻고 waiter리스트 삽입 시에 list\_insert\_ordered 함수를 사용해 우선순위대로 삽입되도록 구현해주었습니다.

* + - void sema\_up ( struct semaphore \*sema )

waiter list에 있는 쓰레드의 우선순위가 변경되었을 경우를 고려하여 list\_sort함수를 사용하여 정렬을 해주었고, 세마포어 해제 후 test\_max\_priority 함수를 사용하여 선점이 가능하게 해주었습니다.

* + - bool cmp\_sem\_priority ( const struct list\_elem \*a, const struct list\_elem \*b, void \*aux )

두 인자의 쓰레드 디스크립터를 읽어와 쓰레드의 우선순위를 비교하여 첫 번째 인자의 우선순위가 높으면 1을, 낮으면 0을 반환하도록 구현했습니다.

1. Priority Inversion Problem
   1. 과제 목표

priority donation 과 multiple donation , nested donation 구현.

* 1. 개념

우선순위가 높은 쓰레드가 우선순위가 낮은 쓰레드를 기다리는 현상으로, donation으로 해결할 수 있다.

쓰레드가 두 개이상의 lock 보유 시 각 lock에 의해 기부가 발생가능 하는 데, 이 때 이전 상태의 우선순위를 기억하고 있어야 한다.

* 1. 구현 함수
     + 구현하지 못했습니다.

2. Make Check 결과

