운영체제(3200, 월수(07-09)) HOMEWORK 2

KU_MMU

201510346 *수학과 민진홍*

---

1. Introduction

지금까지 수업 동안 Address translation, Segmentation, Paging, TLB, Multi-level page table, swapping등에 대해 공부를 하였다. 이번 과제는 프로그램들을 실행하면서 가상 주소를 접근하는데 필요한 페이지를 할당하는 과정에서 운영체제가 작동하는 과정을 실습하는 과제이다. 필요에 따라 컨텍스트 스위치를 발생시켜 pid를 변경해주고, 할당이 안된 페이지들에 대해 page fault를 발생시켜서 페이지 할당 및 Swapping을 해준다. 따라서 이러한 전반적인 과정에 대하여 구현을 하였다. 이번 과제에서 paging 및 swapping에 대한 가정은 다음과 같다.

- Page replacement policy is FIFO.
- 8-bit addressing with 4 bytes page size. And each PTE/PDE is 1 byte;
- In PDE/PTE, the format is PFN[7:2]Unuse[1]Present[0] or Swapoffset[7:1]Present[0].
- In KU_MMU, We use 3-level page tables.

  (i.e. the format of va is PDIndex[7:6] PMDIndex[5:4] PTIndex[3:2] Offset [1:0])
- Only page frame for page(i.e. not page table) is swapped-out

위의 가정에 따라서, 필요한 전역 변수들을 할당해주고, 경우에 따라 context switch를 해줌으로써 pid 를 변경하고 필요에 따라 해당 프로세스의 PDBR 을 할당한다.. 그리고 주어진 process의 page에 접근을 시도할 때, memory에 존재하지 않으면 page fault를 발생시켜서 원하는 page를 memory에 mapping해준다. 물론 memory에 존재하면 바로 접근을 시도한다. Section 2에는 전반적인 접근 방식을, Section3에는 과제에서 제시한 3가지의 함수에 자세한 설명을 적었다. 마지막으로 Section 4에는 소스 파일의 중요 함수들의 대한 description이 되어 있다.

2. The Approaches

전체적인 ku_cpu.c의 프로그램 구조는 다음과 같다.

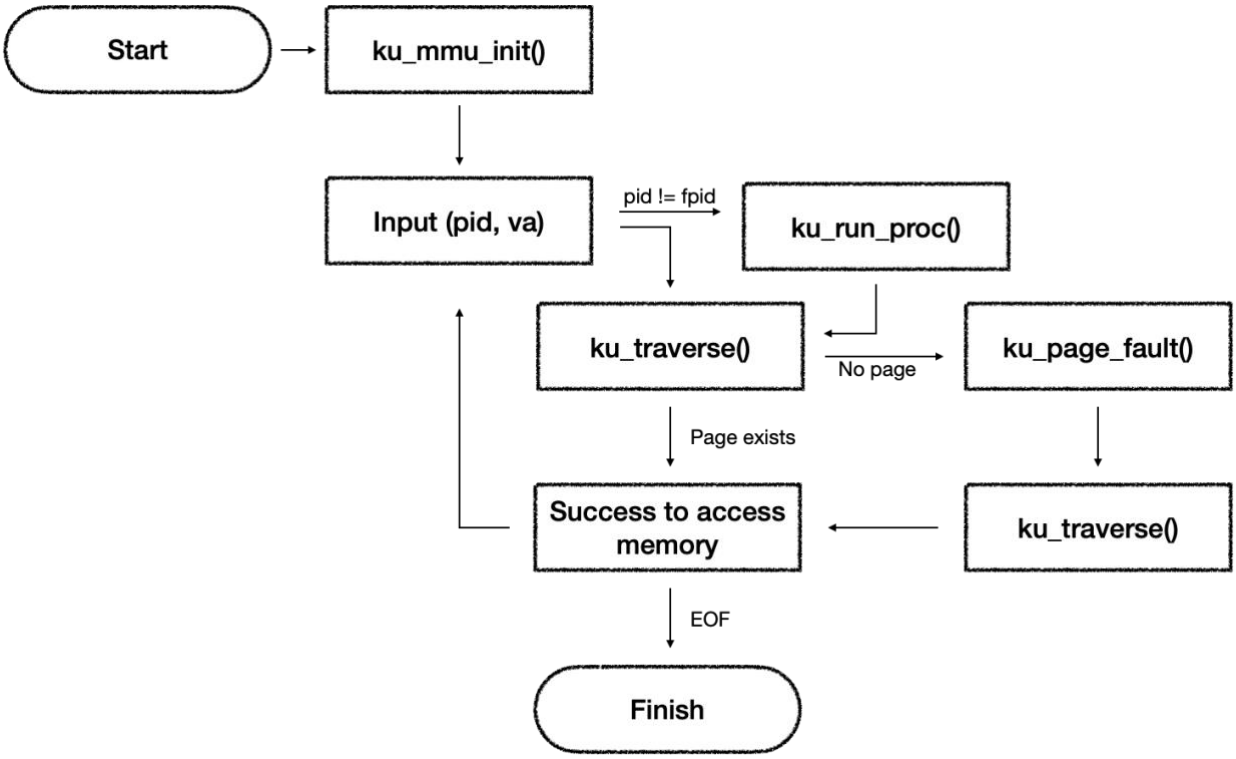

그림 1. The Outline of ku_cpu.c. In this picture, fail case isn't considered.

우선적으로, ku_mmu.h에 필요한 전역 변수 및 상수가 필요하다. 전역 변수는 physical memory, swap memory, free list for physical memory에 대한 변수 및 페이지 교체를 위한 큐, 각 프로세스의 PDBR을 저장하는 리스트 등이 있다.

Physical memory는 PDE/PTE 가 한 객체이기 때문에 그것에 대한 구조체(ku_pte)가 필요하다. 과제의 가정에 의해 크기는 1바이트이므로 unsigned char 형식으로 선언했다. 그리고 메모리공간은 이 구조체 배열을 가리키는 포인터 변수(ku_mmu_pmem)를 선언했다. 마지막으로 메모리 크기를 저장하는 변수(ku_mmu_mem_size)를 선언했다.

Swap memory는 과제 가정에서 실제 데이터에 대한 할당을 하지 않으므로 swap 공간에 대한 할당 여부만 표현해주면 된다. 또한 swap 공간에서 단위는 페이지 크기이므로 4바이트이다. 프로그램 실행 시에 인자로 전달 받는 swap_size는 바이트 단위이므로 메모리 할당 시에 (swap_size / 4)의 크기로 할당했다. 그리고 0,1로 할당 유무만 표현하면 되므로 char 형식을 가리키는 포인터 변수(ku_mmu_s_space)로 선언했다. 마지막으로 swap 공간의 크기를 저장하는 변수(ku_mmu_swap_size)를 선언했다.

Free list의 경우 단위는 페이지 크기이고 physical memory에 대한 할당 유무를 표현하므로 swap 공간과 마찬가지로 char 형식을 가리키는 포인터 변수(ku_mmu_free_p)를 선언해서 (mem_size / 4)의 크기로 할당했다.

페이지 교체를 위해서는 page 할당 시마다 할당했던 순서를 저장하기 위한 큐(ku_mmu_queue)를 선언하여 (pid, va, next)의 정보를 갖는 노드들을 삽입(Enqueue()) 및

삭제(Dequeue())한다. 단, 이 과정에서 과제의 가정에 의해 page directory, page middle directory, page table에 대한 page 를 할당 시에는 삽입을 하지 않고 오로지 page에 대한 page를 할당 시에 삽입한다.

또한 프로세스마다 pdbr를 저장하기 위한 리스트의 헤드 부분(ku_mmu_pdbr)을 선언했다. 이 리스트에는 (pid, pdbr, next)의 정보를 갖는 노드가 삽입(add_PDBR)된다.

주어진 Virtual address에 대해 각 page table에 대한 index를 얻기 위해 shift 연산을 해야한다. 따라서 상수에 대해서는 shift 연산을 위해 필요한 상수(MASKs and SHIFTs)들을 선언하였다.

따라서, ku_mmu_init()을 통해 위의 전역 변수들에 대해 초기화를 해준다. 그리고 실행 시에 전달 받은 텍스트 파일에서 (pid, va)의 정보를 읽은 다음, 현재 실행 중인 pid랑 비교한다. 만약 같으면 그대로 ku_traverse()를 실행한다. 같지 않으면 ku_run_proc()함수를 실행한다. 이 함수에서는 우선 ku_mmu_pdbr을 통해 pid에 대한 pdbr정보가 있는지 찾아보고 없으면, 새로 PFN을 할당하여 pdbr을 만들어준다. ku_traverse()를 실행하는데 필요한 페이지가 할당이 되어 있지 않으면 page fault가 발생하여 ku_page_fault()함수가 실행된다. 이 함수를 통해 할당이 안됐거나, swap-out된 페이지들에 대해서 memory 상에 할당해준다. 즉, 필요한 페이지를 접근하는 경로 상에 memory 에 존재하지 않는 Page Directory, Page Middle Directory, Page Table, Page 가 있다면 memory 상에 할당해준다. 그리고 텍스트 파일의 모든 정보를 읽었으면 해당 프로그램은 성공적으로 끝난다.

3. Important Functions for Homework

1. void *ku_mmu_init(unsigned int mem_size, unsigned int swap_size)

이 함수는 ku_mmu.h가 실행되는데 필요한 모든 전역 변수들에 대하여 초기화를 해주는 작업을 하는 함수이다. 우선 ku_mmu_mem_size와 ku_mmu_swap_size에 인자로 전달 받은 크기들을 대입한다. 그리고 physical memory, swap space, free list를 위한 공간을 calloc()함수를 통하여 할당한다. 여기서 calloc()으로 할당하는 이유는 PDE/PTE 및 사용 유무에 대해서 모두 처음에 0값을 가지고 시작하기 때문이다. 그리고 OS에 해당하는 공간을 위해 physical memory에서 0~3번째 pte는 1로, free list의 0번째 인덱스 값도 1로 바꿔준다. 단, swap space에 대해서는 개념적으로 할당되지 못하는 공간이므로, 0으로 남겨주는 대신, swap space를 탐색하는 과정에서 0번째 인덱스를 제외하였다. 마지막으로 PDBR의 헤드(ku_mmu_pdbr) 및 큐(ku_mmu_queue)에 대한 초기화를 해준다. 이 과정에서 동적 할당을 시도한 다음에는 제대로 할당이 되지 않은 경우에서는 0을 반환하고, 그 이외의 성공한 경우에 대해서는 ku_mmu_pmem, 즉, 물리 메모리 공간의 시작 주소를 반환한다.

2. int ku_run_proc(char pid, void **ku_cr3)

우선, 물리 메모리 공간과 swap 공간이 전부 할당된 상태인지 체크(check_full())를 해준다. 만약 전부 할당된 상태이면 실패, 즉, -1을 반환한다. 그렇지 않으면 우선 ku_mmu_pdbr을 통해서 주어진 pid에 대한 pdbr 정보들을 찾는다(get_pdbr()). 만약 찾을 수 없으면, 새로 PDBR을 할당해주기 위해 할당이 안된 PFN을 할당 받고(alloate_pfn()) 리스트에 추가한다. 이 경우는 PDBR을 위해 할당한 것이므로 큐에는 삽입하지 않는다. 마지막으로 형변환을 통해 ku_cr3에 해당 pdbr주소를 넣어주고 성공적으로 0을 반환한다.

3. int ku_page_fault(char pid, char va)

ku_run_proc과 마찬가지로 우선 메모리 공간과 swap 공간이 전부 할당된 상태인지 체크(check_full())한다. 그 다음 주어진 pid에 대하여 pdbr 정보를 받는다.(get_pdbr()) 이 경우에서 과제의 가정과 ku_run_proc()에 의해 무조건 PDBR 정보는 존재하기 때문에 성공적으로 반환이 된다. 그리고 주어진 va에서 각 page table에 대한 index 값(v0, v1, v2)을 shift연산을 통해 저장한다. 그 다음 page fault가 일어나는 원인에 대하여 경우를 생각해본다.

1) Page Middle directory가 할당되지 않은 경우(즉, 해당 pte가 0x0인 경우) PMD에 대한 PFN(pmd_pfn)을 할당 받고(allocate_pfn()) Page table을 찾는다.

2) Page table이 할당되지 않은 경우(즉, 해당 pte가 0x0인 경우) 1)과 마찬가지로 PT에 대한 PFN(pt_pfn)을 할당 받고(allocate_pfn()) Page를 찾는다.

3) Page 접근 시의 오류

A. Page가 할당되지 않은 경우, page에 대한 PFN(p_pfn)을 할당 받는다.(allocate_pfn()) 단, 여기서 page를 위한 page fame을 할당했기 때문에 큐에 삽입(Enqueue())를 해준다.

B. Page에서 해당 pte의 PFN은 0이 아닌데 present bit가 0이면 swap-out된 상태라고 볼 수 있다. 따라서 상위 7비트가 Swapoffset을 나타내므로 shift연산을 통해 swap space에 할당된(과제에서는 그렇다고 가정한) 공간의 인덱스(s_index)를 얻고 swap space(ku_mmu_s_space)의 해당 인덱스의 값은 0으로 바꿔준다. free list에 대해서는 swap-out되고 바로 다른 페이지가 할당되는 것이기 때문에 값은 그대로 1이다. 위와 마찬가지로 마지막에 큐에 해당 (pid, va) 노드를 삽입한다.

C. Page에서 해당 pte의 present bit가 1이면 이 경우는 어디선가 잘못된 것이다. 원하는 페이지가 할당이 되지 않았기 때문에 page fault가 발생했는데 해당 pte가 할당된 상태로 표현되어있기 때문이다. 따라서 -1를 반환한다.

4. Functions

ku_pte *get_pdbr	Functionality	Searches address of pdbr corresponding to a given pid on PDBR list and returns it. If there is no pdbr information in list, return NULL
	Parameters	char pid : process id for pdbr
int check_full	Functionality	If free list and swap space are full of 1, return 1. If not, return 0.
	Parameters	-
char allocate_pfn	Functionality	1) check_full() 2) Find index(=PFN) for 0 in free list and return it. 3) If there is no 0 in free list, Swap-out and return index for 0 in swap space.
	Parameters	-