운영체제 과제
KU MMU

컴퓨터공학과
201511243
김동언

KU MMU

- Basic Design
  - 운영체제가 프로세스 마다 가상의 주소 공간을 제공하기 위한 페이징 기법을 C언어를 통해 구현.
  - run_proc가 호출되면 프로세스가 존재하지 않는 시점에는 Page Directory를 새로 생성하고 해당 프로세스의 PDBA를 저장하고, 존재하는 경우 PDBA를 불러와서 ku_cr3의 위치를 PDBA가 위치하는 메모리상의 위치로 변경합니다. (Context Switch)
  - page_fault가 발생하면 해당 프로세스의 virtual address에 따라 PD, PMD, PT의 Entry에 페이지를 할당 및 스왑해줍니다.
  - 최종적으로, ku_traverse가 두 번째 호출 되었을 때 물리 메모리상에 virtual address에 맞는 엔트리 할당이 되어서, Address translation(Virtual address to Physical address)이 이루어 질 수 있도록 해줍니다.

- Description for important functions

int ku_h_get_firstin_page	**Functionality**	Swap의 FIFO 정책에 따라 스왑 될 수 없는 페이지를 제외하고 **가장 먼저 할당된 페이지**를 찾아주는 함수입니다. page_fault가 발생하고 모든 페이지가 가득 찼을 때, swap out 하기 위해 사용하는 함수입니다. 스왑 가능한 페이지가 존재하지 않으면 -1을 반환합니다.
	Parameters	
	Return Value	First-In Page Frame Number

int ku_h_get_page	**Functionality**	실제로 페이지 할당을 담당하는 함수입니다. 비어있는 페이지 또는 스왑 가능한 **페이지 번호**를 반환합니다. Swappable 매개변수에 따라 스왑 불가능 또는 들어온 순번에 대한 정보를 ku_h_swappable 변수에 저장합니다. 페이지 할당을 할 수 없으면 -1을 반환합니다.
	Parameters	char swappable
	Return Value	Page Frame Number

void * ku_mmu_init	**Functionality**	자료구조들을 초기화 시키는 함수입니다.
		• **Physical Memory** : struct ku_pte *ku_h_memory
		  • mem_size 만큼 할당
		• **Page free list** : int *ku_h_swappable
		  • mem_size / 4만큼 할당
		• **Swap Space** : int *ku_h_swapspace
		  • swap_size 만큼 할당
		• **Process List** : ku_h_linkedlist *ku_h_processes
		  • 링크드리스트 초기화

	Parameters	unsigned int mem_size, unsigned int swap_size
	Return Value	ku_h_memory의 주소값

int ku_run_proc	**Functionality**	Process List에 pid가 존재하는지 확인하고, 처음 들어온 pid이면 프로세스 생성하고 Page Directory를 할당합니다. 프로세스를 처음 생성할 때, Page Directory를 할당해야 하기 때문에 ku_h_get_page(-1)을 호출했을 때 할당될 페이지가 없으면 프로세스 생성에 실패하고 -1을 반환합니다. 페이지 할당에 성공하면 Process List에 pid와 pfn*4정보를 담아 추가합니다. 포인터로 전달받은 ku_cr3이 가리키는 값을 ku_h_memory + pdba(pfn*4)로 바꿔줍니다.
	Parameters	char pid, struct ku_pte **ku_cr3
	Return Value	Success 0, Failure -1
int ku_page_fault	**Functionality**	각각의 프로세스의 Page Directory로부터 시작하여 VirtualAddress에 따라 Page Directory, Middle Directory, Table를 순회 하면서 주소에 해당하는 **페이지를 할당**해줍니다. 각각 PD, PMD, PT에서 va를 비트마스킹 해서 offset을 구하고 PDE, PMDE, PTE 위치를 찾아 각각 Entry의 값을 이용합니다. Present bit가 0일 때, 스왑 되었는지 확인하고, ku_h_get_page 함수로 페이지 하나를 가져와서 스왑된 상황에서는 swap in을 시줍니다. 가져온 페이지의 PFN을 해당 Entry의 값으로 설정하고, present bit도 1로 바꿔줍니다. Present bit가 1일 때, 다음 페이지는 ku_h_memory + pfn * 4의 위치에 존재하므로 동일한 순서대로 작업을 반복합니다. Page Table에서 페이지 할당이 성공했으면 page_fault 함수가 정상적으로 실행 되었으므로 0을, pmd, pt, page 중 한 곳이라도 ku_h_get_page 함수의 결과값이 -1이면 마찬가지로 -1을 반환합니다.
	Parameters	char pid, char va
	Return Value	Success 0, Failure -1