## Operating System

## Programming Assignment #5: Memory sharing across fork()

**2013-11826 임주경**

**- Summary of modifications**

코드 파일 기준으로 분류해서 수정사항을 요약하자면 다음과 같다.

- defs.h : kalloc.c 와 vm.c 에서 새롭게 정의된 함수에 대한 prototype을 추가하였다.

- exec.c : page를 uvmalloc 함수 호출로 할당하는 과정에서 ph.flag로 해당하는 program section의 write bit를 체크하는 분기점을 추가하였다. 그에 알맞은 flag로 page table에 mapping 되도록 구현하였다.

- kalloc.c : 각 Physical memory를 reference 하고 있는 page table entry를 count 하기 위해서, 일부 space를 할당하였다. 이 reference 공간의 시작 포인터를 글로벌 변수 ref_start로 정의하였고, count 변수는 uint32 type으로 정의하였으며, kernel 뒤의 주소부터 PHYSTOP 주소까지의 공간을 page의 크기 4KB로 나눈 총 page의 개수에 uint32 type의 크기를 곱한만큼 할당해주었다. Reference space의 뒤는 physical address space의 시작 포인터가 되며, 글로벌 변수 pa_start로 정의하였다.

각 Physical address의 reference count 값을 사용하기 위해 편의상 함수 3개를 구현하였다. 각각 get_refP는 physical address를 인자로 받아 해당하는 reference count 값을 저장하는 주소를 반환한다. 이때, page의 크기 4KB를 고려하여, reference space의 index는 (pa – pa_start) / 4KB 로 구현한다.

ref_up 과 ref_down 은 physical adress를 인자로 받아 해당 reference space에 접근하고, count 값을 변화시킨 뒤, 그 값을 반환한다. kfree와 kalloc 함수 내부에는 상황에 맞게 ref_up, ref_down을 추가하였다.

- trap.c : Page fault가 발생할 경우, r_scause 가 12, 13, 15 값을 갖게 된다. 해당 값을 check 하여, 새로 구현한 pagefaultHandler 함수를 호출하는 코드를 추가하였다. 이때, 해결할 수 없는 문제가 발생하면, "Page Fault"를 출력하고 해당 프로세스를 종료한다.

- vm.c : uvmalloc 함수의 인자를 하나 추가하여 exec의 segment의 write flag를 확인해서 적절한 flag를 mapping 할 수 있게 구현하였다. uvmcopy에서 자식 프로세스는 부모 프로세스와 같은 physical memory를 공유하도록 구현하였고, writable한 memory의 flag는 모두 read-only로 변경한다. 또한, ref_up 함수를 호출한다. copyout 함수의 내부에 분기문으로 해당 memory가 write가 불가능할 경우, pagefaultHandler 함수를 호출하도록 추가하였다. pagefaultHandler 함수를 새롭게 추

가하여, copy-on-write를 수행하거나, page fault가 발생했음을 체크하도록 구현하였다.

**- How to catch the page fault**

trap.c 에서 r_scause 값이 12, 13, 15 일 경우 page fault이므로, 이를 catch 한다. Catch 된 상황에서 pagefaultHandler 함수를 호출하고, 해당하는 page가 copy-on-write가 가능한지 flag를 통해 확인한다. 가능하다면, copy-on-write를 수행하여 문제를 해결하고, 그것이 아닌 Invalid virtual address로의 접근이나, read-only에 write를 하는 등의 fault가 발생할 경우, 실행중인 프로세스가 종료된다.

**- How to implement code segment sharing**

Fork가 호출되면, uvmcopy 함수 호출로 자식 프로세스는 새로운 physical memory를 할당 받게 된다. 이때, 새로운 physical memory를 할당하지 않고 부모의 code segment가 차지하는 physical memory를 mapping 하도록 구현하였다. 또한, 해당 physical address에 대한 reference count를 증가시킨다.

**- How to implement copy-on-write on data/stack/heap segment**

Data/stack/heap segment는 write가 가능하다. Uvmcopy 함수에서 해당 segment의 flag를 확인해 write가 가능하다면, 해당 page를 read-only로 변경하며, flag에서 사용되지 않는 8번째 bit (dirty bit)를 copy-on-write checker bit로 사용하였다. PagefaultHandler 함수에서는 이 copy-on-write checker bit를 확인하여, 해당하는 memory의 reference count가 1이 아니라면, 새로운 page를 할당해 copy-on-write를 수행하고, 해당 memory의 reference count가 1이라면, 이 page를 다시 write한 page로 변경하여 진행하고자 하였던 명령을 수행하도록 한다.

**- When is a page frame released and how?**

Page frame release는 kfree 내부에서 구현된다. kfree에서 해당 page의 reference count값을 check 하여 2 이상이라면, count를 하나 감소시키고 마친다. 이때, 단 하나의 참조만이 존재한다면, 그제서야 page frame은 released된다.