OS Project-3 Report

mmap, munmap, freemem, pagefault

Software of Department

2016312568

Jung Hee Yoon

이번 Project 3는 mmap, munmap, freemem, pagefault을 구현하는 프로젝트다.

나는 이를 구현하기위해 mmap.c, mmap.h 파일들을 추가로 생성하였으며, Makefile에서 OBJS에 mmap.o를 추가하고 EXTRA에 mmap.c, mmap.h를 추가하였다. mmap.h에는 mmap_area구조체가 선언되어 있으며 mmap.c에 mmap_area 구조체 배열 areas[64]를 만들어 놓았으면 cur_area라는 현재 areas배열에 몇개의 정보가 들어가있는지를 나타내는 변수이다. 여기서 구현할때 주의점은 process -> sz라는 변수를 통해 부모가 자식에게 copyuvm함수를 통하여 pgdir정보를 전해주는 갯수가 정해지는 점, pgdir의 크기라는 점에서 필요한경우 mmap함수에서 sz사이즈를 늘려주고 allocuvm()을 해주며, munmap함수에서 sz를 줄여주고 deallocuvm()를 해줘야 한다. 이는 기본 pgdir크기(p—>sz)가 3이여서 3페이지 (4096*3)이하만큼 읽고 저장한다면 문제가 되지 않지만 그 이상의 크기를 저장한다면 문제가 생기기에 p—>sz를 늘려주고 늘어난 만큼 다시 할당해 주어야 한다. mmap.c에서 cprintf를 하면 xv6 build하는 과정에 무한루프가 생겨서 출력과 관련된 것은 file.c, proc.c, vm.c에다가 추가적으로 함수를 생성하였다. mmap.c에는 다음과 같은 함수들이 있다 : mmap(), munmap, mmap_fork(), fork_ck(), inc_fm(), dec_fm(), find_area(), fill_read(). 각각 설명을 해보자면, 다음과 같다.(소스첨부로 인해 페이지가 넘어갔습니다.) 또한 Pagehandler는 T_FPFLT플래그가 온 경우에 PFHandler()함수를 만들었으며 이또한 뒤에서 설명하겠습니다.

1) mmap(uint, int, int, int, int, int)

MMAP는 areas구조체 배열에 받은 인자들을 저장해주고 플래그에 따라 저장할 virtual address를 정하고, file을 읽을지를 정하고, mappages함수를 통해 피지컬메모리와 연결하고 pgdir을 추가할지 정한다. 소스는 다음과 같다.

```
uint mmap(uint addr, int length, int prot, int flags, int fd, int offset)
{
    struct proc *p = myproc();
    if((flags&MAP_PRIVATE)==0) return 0;

    areas[cur_area].f = p->ofile[fd];
    areas[cur_area].length = length;
    areas[cur_area].flags = flags;
    areas[cur_area].offset = offset;
    areas[cur_area].prot = prot;
    areas[cur_area].p = p;
    file_init_offset(p->ofile[fd],offset);
    uint allo_pos=0;

    if((flags&MAP_FIXED)!=0)
    {
        if(addr==0) return 0;
        for(int i=0;i<cur_area;i++)
        {
            int addr_area = areas[i].addr-MMAPBASE;
            if(areas[i].p->pid==p->pid&&((addr_area==addr)||(addr_area<addr&&(addr_area+areas[i].length>addr)))) return 0;
        }
        allo_pos=addr+MMAPBASE;
    }
    else
    {
        int ck=1;
        int ck_cnt=0;
        int ck_arr[64];
        uint addr_ck;
        for(int i=0;i<cur_area;i++)
        {
            if(areas[i].p->pid==p->pid)
            {
                ck_arr[ck_cnt]=i;
                ck_cnt++;
            }
        }
        for(int i=0;i<ck_cnt;i++)
        {
            addr_ck=areas[ck_arr[i]].addr+areas[ck_arr[i]].length;
            for(int j=0;j<ck_cnt;j++)
            {
                uint addr2 = areas[ck_arr[j]].addr;

                if((addr_ck==addr2)||((addr_ck>addr2)&&(addr_ck<addr2+areas[ck_arr[j]].length))) {
                    ck=0;
                    break;
                }
            }
            if(ck==1) {
                allo_pos=addr_ck;
                break;
            }
        }
        if(ck==0) return 0;
        if(ck_cnt==0) allo_pos=MMAPBASE;
    }

    areas[cur_area].addr=allo_pos;

    if((flags&MAP_POPULATE)!=0)
    {
        for(int i=0;i<length/PGSIZE;i++)
        {
            char* mem;
            mem = kalloc();
            if(mem==0) return 0;
            memset(mem, 0, PGSIZE);
            if((flags&MAP_ANONYMOUS)==0) fileread(areas[cur_area].f,mem,PGSIZE);

            int loc=i*PGSIZE+areas[cur_area].addr;
            if(mappages(p->pgdir,(void*)loc, PGSIZE, V2P(mem), PTE_W|PTE_U)<0)
            {
                kfree(mem);
                cprintf("fault mappages\n");
                return 0;
            }
            if(p->sz<=loc+PGSIZE) growproc(PGSIZE);
        }

    }
    cur_area++;
    p->map_count++;
    return areas[cur_area-1].addr;
}
```

2) munmap

MUNMAP함수는 virtual address를 받으면 이에 해당하는 프로세스와 virtual address에 해당하는 것들 mmap_area구조체 배열에서 찾고, 있으면 메모리를 해제하고, mmap_area에 있는 내용을 제거한다. 이때 kfree()를 이용하여 원래 사용하던 메모리 (피지컬메모리) 에는 1이 저장된다. 이는 댕글링, 가비지 포인터를 예방하기 위함이다.

```
int munmap(uint addr)
{
    struct proc *p = myproc();
    int de_length=0;
    for(int i=0;i<cur_area;i++)
    {
        if(areas[i].p->pid==p->pid&&areas[i].addr==addr)
        {
            de_length=areas[i].length;
            for(int j=0;j<areas[i].length/PGSIZE;j++)
            {
                uint* pte_addr = walkpgdir(p->pgdir,(void*)(addr+j*PGSIZE),0);
                if(*pte_addr==0) break;
                if(*pte_addr!=0)
                {
                    kfree((char*)(P2V(PTE_ADDR(*pte_addr))));
                    *pte_addr=0;
                }
            }
            for(int j=i;j<cur_area-1;j++)
            {
                areas[j].f=areas[j+1].f;
                areas[j].addr=areas[j+1].addr;
                areas[j].length=areas[j+1].length;
                areas[j].offset = areas[j+1].offset;
                areas[j].prot=areas[j+1].prot;
                areas[j].flags=areas[j+1].flags;
                areas[j].p=areas[j+1].p;
            }
            areas[cur_area].f=0;
            areas[cur_area].addr=0;
            areas[cur_area].length=0;
            areas[cur_area].offset=0;
            areas[cur_area].prot=0;
            areas[cur_area].flags=0;
            areas[cur_area].p=0;
            areas[cur_area].f=0;

            cur_area--;
            break;
        }
    }
    growproc(-1*de_length);
```

3) mmap_fork : proc.c 에서 fork할때 copyuvm()을 하여 부모의 pgdir정보를 가져온뒤 mmap_fork를 실행한다. 이는 부모의 mmap_area구조체 배열에서 부모의 p와 맞는 것을 찾아 자식에게도 할당하는 함수이다.

4) fork_ck() : fork할때 mmap_fork가 불려졌는지 확인하는 함수이다. 디버깅용으로 작성하였다.

5) freemem() : free pages의 갯수를 리턴한다.

6) inc_fm, dec_fm : 각각 freemem_count변수를 하나씩 더하고 빼준다.

3),4),5),6) 에 관한 소스는 다음과 같다.

```
void mmap_fork(struct proc* parent, struct proc* child)
{
    for(int i=0;i<cur_area;i++)
    {
        if(areas[i].p->pid == parent->pid)
        {
            fork_check++;
            areas[cur_area].addr=areas[i].addr;
            areas[cur_area].f=areas[i].f;
            areas[cur_area].length=areas[i].length;
            areas[cur_area].offset=areas[i].offset;
            areas[cur_area].prot=areas[i].prot;
            areas[cur_area].flags=areas[i].flags;
            areas[cur_area].p=child;
            cur_area++;
        }
    }
}
int fork_ck()
{
    return fork_check;
}
int freemem()
{
    return freemem_count;
}
void inc_fm()
{
    freemem_count++;
}
void dec_fm()
{
    freemem_count--;
}
```

7) find_area

이는 인자로 받은 프로세스와 주소로 mmap_area 구조체 배열에서 해당하는 값이 있는지 찾고, 있으면 반환해주는 함수이다.

```
struct mmap_area find_area(struct proc *p, uint addr)
{
    struct mmap_area ck;
    for(int i=0;i<cur_area;i++)
    {
        if(p->pid==areas[i].p->pid&&(areas[i].addr<=addr)&&(areas[i].addr+areas[i].length>addr))
        {
           check=1;
           return areas[i];
        }
    }
    return ck;
}
```

8) fill_read

이 함수는 pagefault handler에서 불려지며 이는 page fault가 일어난 위치와 프로세스를 인자로 받아 해당된 주소에 정보가 있는지 찾고, 없으면 return -1, 있으면 플래그에 따라 데이터를 가져올지 어떤 데이터를 가져올지 체크하고 물리주소와 연결시켜준다.

```
int fill_read(struct proc *p, uint fault_addr)
{
    struct mmap_area ck;
    check=0;
    ck=find_area(p,fault_addr);
    if(check==0)
    {
        return -1;
    }
    file_init_offset(ck.f,ck.offset);
    for(int i=0;i<ck.length/PGSIZE;i++)
    {
        char* mem;
        mem = kalloc();
        if(mem==0) return 0;
        memset(mem, 0, PGSIZE);
        if((ck.flags&MAP_ANONYMOUS)==0) fileread(ck.f,mem,PGSIZE);
        int loc=i*PGSIZE+ck.addr;
        if(mappages(p->pgdir,(void*)loc, PGSIZE, V2P(mem), PTE_W|PTE_U)<0)
        {
            kfree(mem);
            cprintf("fault mappages\n");
            return -1;
        }
        if(p->sz<=loc+PGSIZE) growproc(PGSIZE);
    }
    return 1;

}
```

9) Pagefault Handler

현재 page_fault가 난 주소를 fault_addr변수에 들고오며 mmap.c에 있는 fill_read 함수를 호출하여 상황에 맞게 프로세스를 죽이던, 페이지를 새로 mappages를 한다.

```
void PFHandler(struct trapframe *tf)
{
    struct proc *p=myproc();
    uint fault_addr = rcr2();
    fault_addr = PGROUNDDOWN(fault_addr);

    int ck=fill_read(p,fault_addr);
    if(ck==-1) {
        cprintf("cannot access (already munmap or wrong address) \n");
        p->killed=1;
        return;
    }
}
```

Test cases

#1 munmap한 주소값을 읽어올 경우 (text2)

```c
#include "types.h"
#include "user.h"
#include "param.h"
#include "fcntl.h"

int main(int argc, char** argv) {
    printf(1, "mmap test \n");
    int i;
    int size = 8192;
    int fd = open("README", O_RDWR);
    char* text = (char*)mmap(0, size, PROT_READ, MAP_PRIVATE, fd, 0);                          //File example
    char* text2 = (char*)mmap(0, size, PROT_WRITE|PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0);  //ANONYMOUS example
    char* text3 = (char*)mmap(4096, size, PROT_WRITE|PROT_READ, MAP_PRIVATE|MAP_POPULATE|MAP_FIXED, fd, 0); // FIXED example
    for (i = 0; i < size; i++)
        printf(1, "%c", text[i]);
    printf(1,"\n============file mmap end==========\n\n\n\n");
    munmap((uint)text2);
    if((uint)text2==0) printf(1,"mmap fault\n");
    else
    {text2[0] = 's';
    text2[4096] = 'Y';
    for (i = 0; i < size; i++)
        printf(1, "%c", text2[i]);
    }
    printf(1,"\n============anonymous mmap end==========\n");
    if((uint)text3==0) printf(1,"fixed mmap fault\n");
    else{
        for(int i=0;i<size;i++)
        {
            printf(1,"%c",text3[i]);
        }
    }
    printf(1,"\n============Fixed mmap end=============\n");
    munmap((uint)text);
    //munmap((uint)text2);
    munmap((uint)text3);

    exit();
}
```

result : 우측의 사진처럼 첫번째에 README를 읽고 munmap 되있는 주소를 읽었으므로 pagefault handler 에서 프로세스를 죽인다.

```
    JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)
    Plan 9 (entryother.S, mp.h, mp.c, lapic.c)
    FreeBSD (ioapic.c)
    NetBSD (console.c)

The following people have made contributions: Russ Cox (context switching,
locking), Cliff Frey (MP), Xiao Yu (MP), Nickolai Zeldovich, and Austin
Clements.

We are also grateful for the bug reports and patches contributed by Silas
Boyd-Wickizer, Anton Burtsev, Cody Cutler, Mike CAT, Tej Chajed, eyalz800,
Nelson Elhage, Saar Ettinger, Alice Ferrazzi, Nathaniel Filardo, Peter
Froehlich, Yakir Goaron,Shivam Handa, Bryan Henry, Jim Huang, Alexander
Kapshuk, Anders Kaseorg, kehao95, Wolfgang Keller, Eddie Kohler, Austin
Liew, Imbar Marinescu, Yandong Mao, Matan Shabtay, Hitoshi Mitake, Carmi
Merimovich, Mark Morrissey, mtasm, Joel Nider, Greg Price, Ayan Shafqat,
Eldar Sehayek, Yongming Shen, Cam Tenny, tyfkda, Rafael Ubal, Warren
Toomey, Stephen Tu, Pablo Ventura, Xi Wang, Keiichi Watanabe, Nicolas
Wolovick, wxdao, Grant Wu, Jindong Zhang, Icenowy Zheng, and Zou Chang Wei.

The code in the files that constitute xv6 is
Copyright 2006-2018 Frans Kaashoek, Robert Morris, and Russ Cox.

ERROR REPORTS

Please send errors and suggestions to Frans Kaashoek and Robert Morris
(kaashoek,rtm@mit.edu). The main purpose of xv6 is as a teaching
operating system for MIT's 6.828, so we are more interested in
simplifications and clarifications than new features.

BUILDING AND RUNNING XV6

To build xv6 on an x86 ELF machine (like Linux or FreeBSD), run
"make". On non-x86 or non-ELF machines (like OS X, even on x86), you
will need to install a cross-compiler gcc suite capable of producing
x86 ELF binaries (see https://pdos.csail.mit.edu/6.828/).
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC
simulator and run "make qemu".

============file mmap end==========


cannot access (already munmap or wrong address)
```

#2 fork 확인

길어서 사진으로 받아오지는 못했지만 결과는 README파일을 두번 읽어온다. 이는 부모 프로세스와 자식프로세스가 같은 물리주소를 공유함을 알 수있다.
또한 fwrite(현재는 지웠지만 원래는 구현했었습니다.) 를 통하여 부모와 자식이 같은 물리주소를 공유할 경우, 부모가 파일내용을 바꾸고 munmap하고 후에 자식이 파일내용을 본다면 바뀌어있는것을 통해 같은 주소를 가리키고 fwrite또한 구현되었음을 확인했습니다.

```
int main(int argc, char** argv) {
    printf(1, "mmap test \n");
    int size = 4096*5;
    int parent=getpid();
    int child;
    int fd = open("README", O_RDWR);

    char* text =(char*) mmap(0, size, PROT_READ, MAP_PRIVATE, fd, 0);                          //File example
    if((uint)text==0)
    {
        printf(1,"mmap failed\n");
        return 0;
    }
    if((child=fork())==0)
    {
         printf(1,"result address : %x\n",(uint)text);

        for(int i=0;i<size;i++)
            printf(1,"%c",text[i]);
        printf(1,"\nchild pid = %d\n", child);
        munmap((uint)text);
        exit();
    }
    else
    {
        wait();
        for(int i=0;i<size;i++)
            printf(1,"%c",text[i]);
        printf(1,"\nparent pid = %d\n",parent);
         printf(1,"result address : %x\n",(uint)text);

        munmap((uint)text);
    }
    exit();
}
```