gggyyyyjjj L91 2015-08-11 19:05 Linux 计算机专业课

实验代码

“操作系统原理与实践”实验报告

地址映射与共享

在linux0.11中，test.c实际上打印的是逻辑地址，这个地址加上ds指示的基址才是线性地址，在现代linux内核中，由于段基址总是0，所以，线性地址就等于逻辑地址了。

在linux0.11中，每个进程分得独立的64M虚拟地址空间，因此ds基址就是从这64M*0/1/2/...开始，第一次跑test.c的时候，ds段基址是0x10000000也就是256M，结束后再跑一次，发现还是这个数值，估计是这段虚拟地址被收回后再次使用。如果，先跑test&，然后再跑test，当跟到第二个test进程的时候，会看到ds基址是0x14000000也就是256+64=320M的起始位置。

逻辑地址，是在编译后就确定了的，因此，不过在哪里跑，结果都是一样的。可以用objdump来看到编译结果。 test.c编译后，用objdump -d -s -x查看，可看到 (我将数字从0x12345678修改为0xaaaaaaaa更加明显一点） 00003004 g .data 0000 00 07 _i Contents of section .data: 3000 00000000 aaaaaaaa 84300000 00000000 .....0.....、

在地址实验映射中，重要的步骤有： 根据ds在GDT或者IDT中的offset，找到段基址，从而形成线性虚拟地址，再根据CR3的页目录表基址，找到相应页表基址，最后得到页架号从而形成物理地址。

由于是从信号量实验演变而来，因此，我没有先在ubuntu系统中测试，而是直接在linux0.11上写的应用程序。为了编写方便，直接写在一个.c文件中，然后通过命令行参数区分是producer还是consumer。代码内容见
http://git.shiyanlou.com/gggyyyjjj/shiyanlou_cs115/src/lab07/oslab/pc.c

用gcc pc.c即可编译通过 然后先运行 ./a.out & 启动生产者进程，再运行./a.out 1启动消费者进程。

一开始只实现了shmget和shmat函数，在最后进程退出的时候，会有一个free free page的错误，原因是，当进程退出的时候，操作系统会在背后帮忙释放所有的有效的线性地址所对应的物理页，那么，对这个共享的物理页，会被生产者和消费者两次释放，就产生了这样的错误。

简单的解决方法，是增加shmdt函数，将相应线性地址的页表项置为0，这样，操作系统就不会为此释放物理页了。这样做的后果是，用shmget分配的物理页永远无法被释放，可以增加一个系统调用，或者是用reference count机制进行处理，本试验没有做。

shm内核实现的主要代码见 http://git.shiyanlou.com/gggyyyjjj/shiyanlou_cs115/src/lab07/oslab/linux -0.11/mm/shm.c

虽然阅读了那个代码注释，也是看了别人的报告，才知道原来current->brk指示着空闲地址的开始。不过，还有一个疑问，那就是在shmdt的时候，将线性地址归还回去的时候，可以直接修改current->brk吗，因为，此时的值，可能不再是shmat时候的值了，简单的减去PAGE_SIZE就可以了吗，还是需要另外一个数据结构来保存记录的。

shiyanlou@185a31ebbd42: ~/Code/shiyanlou_cs115/oslab

```
void Produce()
{
        int i = 0;
        int shmid = shmget(SHMKEY, 4096, 0);
        int* shmaddr = (int*)shmat(shmid, NULL, 0);
        int writecount = 0;
        printf("shm address: 0x%x\n", shmaddr);

        /*init the memory, SIZE + write count + read count*/
        for (i = 0; i < SIZE + 2; ++i)
        {
                shmaddr[i] = 0;
        }

        for (i = 0; i < M; ++i)
        {
                sem_wait(freeRes);

                /*do not need lock here, always the different slot is accessed */
                writecount = shmaddr[SIZE];
                shmaddr[writecount] = i;
                shmaddr[SIZE] = (writecount+1) % SIZE;

                /*printf("write %d to buffer[%d]\n", i, writecount);
                fflush(stdout);*/
                sem_post(fillRes);
        }

        shmdt(shmaddr);
        printf("producer finished\n");
        fflush(stdout);
}

void Consumer()
{
        FILE* resultfp = fopen("result.txt", "w");
```

85,16 44%

应用程序菜单

shiyanlou@185a31ebbd42: ~/Code/shiyanlou_cs115/oslab

```
int sys_shmget(key_t key, size_t size, int shmflg)
{
        int i = 0;
        int freeSlot = -1;
        printk("in shmget\n");

        if (size > PAGE_SIZE)
        {
                return -EINVAL;
        }

        //do not lock shmArray here
        //since the application will invoke one by one
        for (i = 0; i < SHM_ARRAY_COUNT; ++i)
        {
                if (freeSlot == -1 && shmArray[i].magic != MAGIC)
                {
                        freeSlot = i;
                }
                if (key == shmArray[i].key)
                {
                        printk("shm key %d already exists\n", key);
                        return i;
                }
        }

        if (freeSlot == -1)
        {
                printk("too many shm created\n");
                return -1;
        }

        unsigned long page = get_free_page();
        if (page == 0)
        {
                return -ENOMEM;
```

31,9 40%

应用程序菜单