山东大学 计算机科学与技术 学院

操作系统 课程实验报告

学号：202100130022	姓名：郭家宁	班级：21 数据
实验题目：存储管理		
实验学时：2	实验日期：2023-6-5	

实验目的：

通过获取进程空间，理解 Linux 中进程的存储管理方式与进程地址空间的组成。

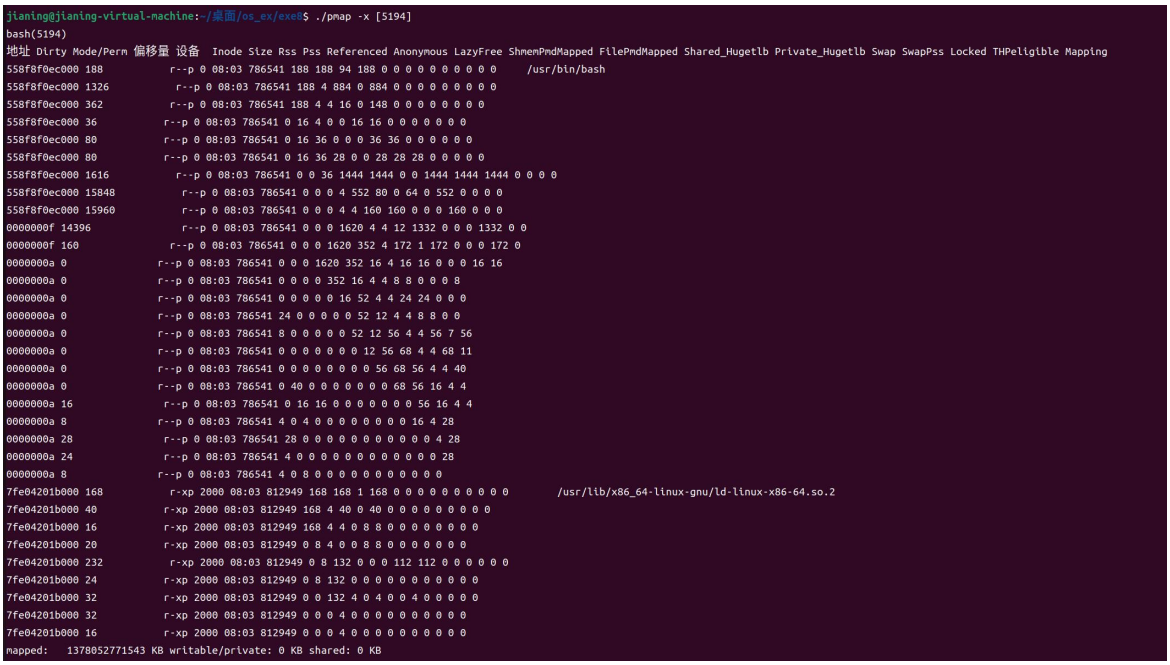
查阅资料，至少实现 linux 命令 pmap [-x -X] 的功能，其它功能自选。

实验结果：

1. Pmap -X(-x) 指令，-X 参数表示展示详细信息，包含全项内容、ELF 头、加载段等信息。

该命令可以用来查看当前正在运行的进程占用的内存情况，包括虚拟地址空间的分布，执行文件和共享库、堆、栈的分配情况等。

2. 查看 bash 的地址空间。



可以发现，能够查询出 bash 的地址，所在文件地址，已经偏移量等等。

```
5194:   bash
地址           Kbytes     RSS   Dirty Mode  Mapping
0000558f8f0ec000     188     188       0 r---- bash
0000558f8f11b000     892     884       0 r-x-- bash
0000558f8f1fa000     232     148       0 r---- bash
0000558f8f235000      16      16      16 r---- bash
0000558f8f239000      36      36      36 rw--- bash
0000558f8f242000      44      28      28 rw---   [ anon ]
0000558f907e5000    1564    1444    1444 rw---   [ anon ]
00007fe040e00000   14236     552       0 r---- locale-archive
00007fe041c00000     160     160       0 r---- libc.so.6
00007fe041c28000    1620    1332       0 r-x-- libc.so.6
00007fe041dbd000     352     172       0 r---- libc.so.6
00007fe041e15000      16      16      16 r---- libc.so.6
00007fe041e19000       8       8       8 rw--- libc.so.6
00007fe041e1b000      52      24      24 rw---   [ anon ]
00007fe041fd3000      12       8       8 rw---   [ anon ]
00007fe041fd6000      56      56       0 r---- libtinfo.so.6.3
00007fe041fe4000      68      68       0 r-x-- libtinfo.so.6.3
00007fe041ff5000      56      40       0 r---- libtinfo.so.6.3
00007fe042003000      16      16      16 r---- libtinfo.so.6.3
00007fe042007000       4       4       4 rw--- libtinfo.so.6.3
00007fe042010000      28      28       0 r--s- gconv-modules.cache
00007fe042017000       8       8       8 rw---   [ anon ]
00007fe042019000       8       8       0 r---- ld-linux-x86-64.so.2
00007fe04201b000     168     168       0 r-x-- ld-linux-x86-64.so.2
00007fe042045000      44      40       0 r---- ld-linux-x86-64.so.2
00007fe042051000       8       8       8 r---- ld-linux-x86-64.so.2
00007fe042053000       8       8       8 rw--- ld-linux-x86-64.so.2
00007fff5f921000     132     112     112 rw---   [ stack ]
00007fff5f9f2000      16       0       0 r----   [ anon ]
00007fff5f9f6000       8       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 --x--   [ anon ]
---------------- ------- ------- -------
total kB           20060    5584    1736
```

3. 具体实现 pmap -x [pid]

（1） 首先使用字符串解析，把 pid 从字符串变成整形数据。 ‘

（2） 在/proc/pid/cmdline 下获取输出该进程的所有程序。

（3） 然后在/proc/pid/smaps 下获取该进程中所有程序的内存映射，地址 Dirty Mode/Perm 偏移量 设备等等信息。

```
pid_t StringToIntPid(char *p)
{
    while (!isdigit(*p) && *p)
        p++;
    return strtol(p, 0, 0);
}
```

```
void get_programe(pid_t pid)
{
    char name[PATH_MAX];
    int c, i = 0;
    FILE *f;

    sprintf(name, "/proc/%ld/cmdline", (long)pid);
    f = fopen(name, "r");
    if (!f)
        die("%s: %s\n", name, strerror(errno));
    while ((c = getc(f)) != EOF && c != 0)
        name[i++] = c;
    name[i] = '\0';
    printf("%s(%ld)\n", name, (long)pid); // 输出程序名称
    fclose(f);
}
```

```
void get_maps(pid_t pid)
{
    char fname[PATH_MAX];
    unsigned long writable = 0, total = 0, shared = 0;
    FILE *f;

    sprintf(fname, "/proc/%ld/smaps", (long)pid);
    f = fopen(fname, "r");

    if (!f)
        die("%s: %s\n", fname, strerror(errno));
    printf("地址 Dirty Mode/Perm 偏移量 设备 Inode Size Rss Pss Referenced Anonymous LazyFree ShmemPmdMapped FilePmdMapped Shared_Hugetlb Private_Hugetlb Swap SwapPss Locked THPeligible Mapping\n");
    while (!feof(f))
    {
        char buf[PATH_MAX + 100], perm[5], dev[6], mapname[PATH_MAX];
        // 地址、权限、映射文件所属设备号、映射的文件名
        unsigned long begin, end, size, inode, foo, Size[24];
        int n;
        for (int i = 1; i <= 23; i++)
        {
            if (fgets(buf, sizeof(buf), f) == 0)
                break;
            if (i == 1)
            {
                mapname[0] = '\0';
                sscanf(buf, "%lx-%lx %4s %lx %5s %ld %s", &begin, &end, perm,
                        &foo, dev, &inode, mapname); // *
                size = end - begin;
                total += size;
            }
            if (i >= 2 && i <= 21)
            {
                char tempchar[1000];
                sscanf(buf, "%s%ld%s", tempchar, &Size[i], tempchar);
            }
            if (i == 22)
            {
                char tempchar[1000];
                sscanf(buf, "%s%ld", tempchar, &Size[i]);
            }
            if (i == 23)
            {
                char tempchar[1000];
```

```
                    // sscanf(buf,"%s%ld",tempchar,&Size[i]);
                }
            }
            if (perm[3] == 'p')
            {
                if (perm[1] == 'w')
                    writable += size;
            }
            else if (perm[3] == 's')
                shared += size;
            else
                die("unable to parse permission string: '%s'\n", perm);
            unsigned long sumDirty = 0;
            for (int j = 7; j <= 10; j++)
                sumDirty += Size[j];
            n = printf("%08lx ", begin);
            n += printf("%ld %*s %s %lx %s %ld ", sumDirty, 22 - n, "", perm, foo, dev, inode);
            printf("%ld %ld %ld %ld %ld %ld %ld %ld %ld %ld %ld %ld %ld %ld %ld ", Size[2], Size[5],
Size[6], Size[11], Size[12], Size[13], Size[15], Size[16], Size[17], Size[18], Size[19], Size[20],
Size[21], Size[22]);
            printf("%*s %s\n", 44 - n, "", mapname);
        }

        printf("mapped:      %ld KB writable/private: %ld KB shared: %ld KB\n",
                total / 1024, writable / 1024, shared / 1024);
        fclose(f);
}
```

问题及收获：

对获取进程的内存映射有了更加深刻的理解与体会，而且对 pamp 命令也有了更熟悉的应用。