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：201705130120	姓名：苑宗鹤	班级：17 1 班
实验题目：实验七		
实验学时：2	实验日期：2020/5/1	

实验目的：

（1）通过考察系统加载应用程序过程，如何为其分配内存空间、创建页表并建立虚页与实页帧的映射 关系，理解 Nachos 的内存管理方法;

（2）理解系统如何对空闲帧的管理;

（3）理解如何加载另一个应 用程序并为其分配地址空间，以支持多进程机制;

（4）理解进程的 pid;

（5）理解进程退出所要完成的 工作;

实验环境：ubuntu18 x64 windows10 clion

实验步骤：

（1）阅读./prog/protest.cc,深入理解 Nachos 创建应用程序进程的详细过程

（2）阅读理解类 AddrSpace，然后对其进行修改，使 Nachos 能够支持多进程机制，允许 Nachos 同时运行多个用户线程;

（3）在类 AddrSpace 中添加完善 Print()函数(在实验 6 中已经给出)

（4）在类 AddrSpace 中实例化类 Bitmap 的一个全局对象,用于管理空闲帧;

（5）如果将 SpaceId 直接作为进程号 Pid 是否合适?如果感觉 不是很合适， 应该如何为进程分配相应的 pid?

（6）为实现 Join(pid),考虑如何在该进程相关联的核心线 程中保存进程号;

（7）根据进程创建时系统为其所做的工作，考虑进程退出时应该做哪些工 作;

（8） 考虑系统调用 Exec()与 Exit()的设计实现方案;

实验结果：

首先创建一个 bitmap 全局变量用来保存已经分配的

```
    void SaveState();           // Save
    void RestoreState();        // infc
    void Print();
    static BitMap * freeMap;
  private:
    TranslationEntry *pageTable;    //
```

然后在 AddrSpace 被调用时进行初始化

```
58  //----------------------------------------------------------------------
59
60  AddrSpace::AddrSpace(OpenFile *executable)
61  {
62
63      if ( AddrSpace::freeMap == nullptr ) {
64          //初始化
65          AddrSpace::freeMap=new BitMap(NumPhysPages);
66      }
67
```

分配时通过 Find 来分配

```
pageTable = new TranslationEntry[ numPages ];
for ( i = 0; i < numPages; i++ ) {
    int frame = AddrSpace::freeMap->Find();
    pageTable[ i ].virtualPage = frame; // for now, virtual page # = phys page #
    pageTable[ i ].physicalPage = frame;
    pageTable[ i ].valid = TRUE;
    pageTable[ i ].use = FALSE;
    pageTable[ i ].dirty = FALSE;
    pageTable[ i ].readOnly = FALSE; // if the code segment was entirely on
    // a separate page, we could set its
    // pages to be read-only
    bzero( machine->mainMemory + pageTable[ i ].physicalPage * PageSize, PageSize );
}
```

并且按页来进行 bzero 置零操作

可以将第一个分配的 spaceid 作为 pid 因为进程之间空间不重复

问题及收获：