**虚存管理模拟程序开发文档**

程序开发：田勇(39051718)

组员：王博弘，梁炯潜，郭莉莎

**TableofContents**

* [1需求说明：](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-1)
* [2主要数据结构](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2)
  + [2.1物理页面结构page\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.1)
  + [2.2快表项结构tlbe\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.2)
  + [2.3页表项pte\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.3)
  + [2.4程序结构program\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.4)
  + [2.5页表目录pgd\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.5)
  + [2.6进程控制块task\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.6)
  + [2.7指令command\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.7)
  + [2.8请求request\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.8)
  + [2.9外存块结构fs\_block\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.9)
  + [2.10mod\_t](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.10)
  + [2.11address\_tbyte\_tbool](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-2.11)
* [3主要模块](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3)
  + [3.1虚拟机模块:vmachine.c](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.1)
    - [3.1.1作用](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.1.1)
    - [3.1.2主要数据](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.1.2)
    - [3.1.3主要函数](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.1.3)
  + [3.2进程控制模块task.c](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.2)
    - [3.2.1作用](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.2.1)
    - [3.2.2数据](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.2.2)
    - [3.2.3主要函数](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.2.3)
  + [3.3页帧模块memmap.c](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.3)
    - [3.3.1作用](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.3.1)
    - [3.3.2数据](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.3.2)
    - [3.3.3主要函数](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.3.3)
  + [3.4快表模块tlb.c](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.4)
    - [3.4.1作用](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.4.1)
    - [3.4.2数据](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.4.2)
    - [3.4.3主要函数](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.4.3)
  + [3.5设备模块](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.5)
    - [3.5.1作用](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.5.1)
    - [3.5.2主要数据](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.5.2)
    - [3.5.3主要函数](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.5.3)
  + [3.6静态程序模块](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.6)
    - [3.6.1作用](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.6.1)
    - [3.6.2数据](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.6.2)
  + [3.7客户终端模块tty.c](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.7)
    - [3.7.1作用：](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.7.1)
    - [3.7.2主要数据](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.7.2)
    - [3.7.3主要函数](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-3.7.3)
* [4功能实现](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4)
  + [4.1快表](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.1)
  + [4.22级页表管理](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.2)
  + [4.3程序加载](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.3)
  + [4.4进程控制](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.4)
  + [4.5最近最少使用算法](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.5)
  + [4.6内存共享](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.6)
    - [4.6.1fork操作](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.6.1)
    - [4.6.2加载相同程序](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.6.2)
    - [4.6.3共享页面的交换处理](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.6.3)
  + [4.7缺页处理](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.7)
  + [4.8写时复制的实现](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.8)
  + [4.9多道程序并发及客户终端与虚拟机通信的实现](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.9)
    - [4.9.1多道程序实现](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.9.1)
    - [4.9.2通信](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-4.9.2)
* [5运行说明](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-5)
* [6测试说明](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-6)
* [7总结与感想](file:///D:\homework\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\README.html#sec-7)

**1需求说明：**

开发一个虚拟内存模拟程序，可以很好地模拟操作系统对虚拟内存的管理。该模拟程序实现以下模拟需求：

1. 基本需求
   1. 采用页式存储管理;
   2. 实现对缺页中断;
2. 高级需求
   1. 实现多道程序虚存管理.
   2. 实现２级页表
   3. 实现快表
   4. 通过FIFO管道实现多进程间通信。
3. 更高级需求（创新点)
   1. 实现内存共享
   2. 实现写时复制
   3. 模拟程序加载过程
   4. 实现最近最少使用替换算法

**2主要数据结构**

**2.1物理页面结构page\_t**

typedefstruct{

intlock;//equalstorefernumberminus1

boolaccessed;

byte\_tfreq;

pte\_t\*ref\_pte;

intblock;/\*blockinthe\*/

}page\_t;

|  |  |  |  |
| --- | --- | --- | --- |
| lock | 页面锁 | 记录当前物理页的引用 | -1:没有被引用;0:1个引用;>0:多个引用，页面被锁住，不允许交换 |
| accessed | 访问位 | 在一个刷新周期内是否被访问过 | 刷新周期到来时，将全部表项置FALSE,每次对页面访问，都将其置为True |
| freq | 最近使用频率 | 保存当页面被访问信息，用于页面替换 | 见[最近最少使用算法](file:///D:\\homework\\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\\README.html" \l "sec-2.1) |
| ref\_pte | 引用页表项指针 | 当有多个页表项引用时，只保留一个，如果没有引用，则置为NULL |  |
| block | 外存块号 | 指向外存中数据所在的块 |  |

**2.2快表项结构tlbe\_t**

/\*TLBentytype\*/

typedefstruct{

intvpage;//virtualpage

intppage;//physicpage

booledited;

mod\_tmode;

boolvalid;//Trueifit'sused.

intfreq;

}tlbe\_t;

|  |  |  |  |
| --- | --- | --- | --- |
| vpage | 虚页号 |  |  |
| ppage | 物理页号 |  |  |
| edited | 修改位 | 指向的页面是否被修改 |  |
| mode | 页面标志 | 页面的读、写、执行权限及是否写时复制 |  |
| valid | 有效位 | 当前表项是否有效 | TRUE:表项可用，FALSE:表项无用 |
| frep | 使用频率 | 记录表项访问频率，用于实现快表项替换 | 见[快表](file:///D:\\homework\\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\\README.html" \l "sec-1)的实现 |

**2.3页表项pte\_t**

typedefstruct{

boolvalid;//inmemory?

boolswapped;//TRUEifinswap;FALSEifinFS

booledited;

mod\_tmode;

//boolcopy;//writeoncopy

intpage\_ID;

}pte\_t;

|  |  |  |  |
| --- | --- | --- | --- |
| valid | 有效位 | 当前页是否在内存 |  |
| swapped | 交换位 | 当前页是否已经交换 |  |
| edited | 修改位 | 页面是否已经修改 |  |
| mode | 页面标志 | 页面的读、写、执行权限及是否写时复制 |  |
| page\_ID | 页号 | 物理页号或外存块号 | -1：指向全０页面;>0:页号或块号 |

**2.4程序结构program\_t**

typedefstruct{

intstart\_block;//thefirstblockinthedisk.

intdata;//thelengthofdataarea,whitchisreadonly

inttext;//lengthoftextarea,whitchisreadonlyandexcutable

}program\_t;

|  |  |  |
| --- | --- | --- |
| start\_block | 起始块号 | 程序在外存中第一块所在的块号 |
| data | 数据段大小（页） | 程序数据段所占的页数 |
| text | 程序段大小 | 程序程序段所占的页数 |

**2.5页表目录pgd\_t**

typedefpte\_t\*pgd\_t;

**2.6进程控制块task\_t**

typedefstruct\_task\_t{

pid\_tpid;

pgd\_t\*pgd;

program\_t\*program;

struct\_task\_t\*next;

intfifo\_fd;

}task\_t;

|  |  |  |
| --- | --- | --- |
| pid | 进程号 | 客户端进程的进程号，用于区分多个进程 |
| pgd | 页目录表 | 指向目录数组的指针 |
| program | 程序 | 进程对应的程序 |
| next | 下一个进程 |  |
| fifo\_fd | 管道描述符 | 该进程的输出管道 |

**2.7指令command\_t**

typedefenum{

NEW\_TASK,

READ,

WRITE,

EXEC,

RUN,

DISPLAY,

QUIT,

}command\_t;

|  |  |
| --- | --- |
| NEW\_TASK | 新进程指令 |
| READ | 读内存指令 |
| WRITE | 写内存指令 |
| EXEC | 执行内存指令 |
| RUN | 运行程序指令 |
| DISPLAY | 显示页表信息指令 |
| QUIT | 进程退出指令 |

**2.8请求request\_t**

typedefstruct{

pid\_tpid;

command\_tcommand;

address\_taddress;

byte\_tvalue;

}request\_t;

|  |  |
| --- | --- |
| pid | 进程id |
| command | 请求命令 |
| address | 请求地址 |
| value | 请求值 |

**2.9外存块结构fs\_block\_t**

typedefstruct\_block\_t{

intpage\_ID;//ifloaded,thephisicpage

intcount;//referencecount

}fs\_block\_t;

|  |  |  |
| --- | --- | --- |
| page\_ID | 页号 | 如果当前页被装入内存，其所在的页面号 |
| count | 引用数 | 使用当前块的进程数 |

**2.10mod\_t**

typedefunsignedcharmod\_t;

#defineMODE\_RWX0x07

#defineMODE\_RW0x06

#defineMODE\_W0x02

#defineMODE\_R0x04

#defineMODE\_X0x01

#defineMODE\_C0x08

|  |  |
| --- | --- |
| MODE\_RWX | 可读、可写、可执行 |
| MODE\_RW | 可读、可写 |
| MODE\_W | 只写 |
| MODE\_R | 只读 |
| MODE\_X | 只执行 |
| MODE\_C | 写时复制 |

**2.11address\_tbyte\_tbool**

typedefunsignedshortaddress\_t;

typedefunsignedcharbyte\_t;

typedefenum{

TRUE=1,

FALSE=0,

}bool;

|  |  |  |  |
| --- | --- | --- | --- |
| address\_t | 虚拟程序的地址 | １６位 | 采用１６进制表示，如2a7b |
| byte\_t | 虚拟程序的字节 | 8位 | １６进制，如3e |

**3主要模块**

**3.1虚拟机模块:vmachine.c**

**3.1.1作用**

虚拟机模块是整个模拟程序的核心，负责对请求的应答和从虚地址空间到物理地址之间的转换，在转化期间发生的页表的动态生成、管理、释放，也由虚拟机模块管理。

**3.1.2主要数据**

* Memory  
  Memory本模拟程序所模拟的内存，共MEM\_SIZE字节。通过对MEM\_SIZE的调节，可以控制物理内存的大少。MEMSIZE应当是PAGESIZE的整数倍。
* Address和AddrMode  
  Address模拟地址寄存器，保存对虚拟地址的进行转化后得到的实地址。

AddrMode保存对物理地址所在页的标志，包括读、写、执行权限和写时复制标志。

* Buffer  
  Buffer作为缓冲区，用于与客户端程序进行通信。

**3.1.3主要函数**

* voidmmap\_refresh\_handler(intsig)  
  处理时钟信息，用于实现最近最少使用原则算法。当时钟信号到来时，调用memap\_refresh函数，对物理页面更新，并重新设置时钟来激发下一次更新。
* voidstart\_machine()  
  起动虚拟机，调用各个模块的初始化函数对各个模块进行初使化。
* voidshutdown\_machine()  
  关闭虚拟机，释放资源。
* voiddisplay(char\*info),voiddispaly\_apgetable(task\_t\*task)  
  显示进程页表信息，并将信息通过FIFO传给客户端程序。
* do\_response(request\_t\*request)  
  对请求进行回应。
  1. 如果请求类型为NEW\_TASK,则创建一个新的进程
  2. 如果请求的进程不是当前进程，则将进程切换到当前进程，切换过程要将快表清空。
  3. 如果请类型是WRITE,READ,EXEC则将地址转化为物理地址，转化过程中处理缺页、置换、加载等情况。
  4. 如果请求类型是RUN,则进行加载新程序替换当前程序。
  5. 如果请求类型是QUIT,则释放当前进程资源。
* pte\_t\*lookup\_pgt(address\_tvaddr)  
  查找页表，将物理地地和物理页面的标志分别存入Address和AddrMode。在此过程中进行缺页处理、页面管理、加载、共享等操作。
* boolaccess\_addr(address\_tvaddr,command\_tcmd)  
  对READ,WRITE,EXEC进行操作，并判断是否有进行该操作的权限。该函数先查找TLB是否有要查找的页面，如果没有，则查找页表，并将结果存入TLB中。对权限进行验证后，执行操作，并返回结果。
* voidvm\_fork(pidtchild\_id,task\_t\*ptask)  
  对ptask进程进行fork,产生一个进程id为child\_id的子进程。子进程并不立刻分配内存空间，而是和父进程共享内存，但此时父进程和子进程的页面全部标记为写时复制。
* voidpgd\_deep\_clone(task\*task)  
  产生一个和task共享内存的进程的内表项。
* exec\_program(task\_t\*task,program\_t\*pro),load\_program(task\_ttask,program\_t\*pro)  
  用一个新程序替换进程的程序，重新分配页表，将其定位到对应的外存或内存页面。
* lazy\_load(task\_t\*task,intpage)  
  对进程task的page页面进行“懶加载“：只将对应的外存块号写入页表，而外存中的页面并不加载到内存，留到访问到该页面时再进行加载。
* resetpte(constpte\_t\*pte),free\_pgd(pgd\_tpgd)  
  释放对应的页目录和页表。
* main()  
  程序入口。

**3.2进程控制模块task.c**

**3.2.1作用**

控制进行程的创建、切换、终止。

**3.2.2数据**

* TaskInit  
  模拟Linux系统中的init进程。在本模拟程序中，任何其它进行都由TaskInit进程fork出来，然后调用EXEC命令加载其它程序。
* CurTask  
  当前进程的指针。

**3.2.3主要函数**

* voidinit\_task()  
  初始化进程模块，设定TaskInit进程，加载其程序。
* task\_t\*new\_task(pid\_tpid,pgd\_t\*pgd)  
  生成一人进程号为pid,页目录为pgd的新进程。
* boolswitch\_task(pid\_tpid)  
  将当前进程切换到pid进程。
* task\_t\*free\_task(task\_t\*task)  
  释放task进程的资源
* voidkill\_task(pid\_tpid)  
  删除进程号为pid的进程。

**3.3页帧模块memmap.c**

**3.3.1作用**

控制物理页面的加载、交换、保护、共享，维护空闲页面清单。

**3.3.2数据**

* MemMap[MMapSIZE]  
  MemMap的每一个页面对应着一个物理页面，负责维护该页面的信息。

**3.3.3主要函数**

* voidpage\_in(pte\_t\*pte)  
  处理缺页。先在页面中找出一个空页表（如果没有，调用page\_out换出一个),然后根据pte中的信息将页面加载入内存。
  1. 如果swapped为True，则从交换区中将对应页面加载入内存
  2. 如果swapped为False，则从FSBlock中将页面加载到内存
  3. 如果pageID为-1，则直接使用一块空页面，并将该页面全部置零。
* voidpage\_out(intpage\_id)  
  将page\_id交换出内存，如果页面没有修改过，直接丢弃该页面，并调用lazy\_load设置该页的外存块号。
* intmempty\_page()  
  查找所有页面，如果lock为－１,则说明此页为空，返回该页号；如果没有，则在lock为0的页面中，寻找最近最少使用的（即freq最小的)的页面将其交换出去，并返回该页页号。
* voidmemmap\_refresh()  
  扫描所有的页面，将页面的freq右移一位，如果页面的accessed位为TRUE，则freq最高位置１，否则置０。

**3.4快表模块tlb.c**

**3.4.1作用**

模拟快表.

**3.4.2数据**

* TLB{TLBSIZE]  
  快表数据.

**3.4.3主要函数**

* voidinit\_tlb()  
  初始化快表.
* intlookup\_tlb(address\_tvaddress)  
  在TLB中查找vaddress的页面，如果没有，则对返回-1,否则返回对应的下标，并将Address和AddrMode设置为物理地址和页面标志。
* voidtlb2pgt(out),voidpgt2tlb(intvpage,pte\_t\*pte,intin)  
  控制在快表和页表之间相互传递。
* inttlb\_swap(pte\_t\*pte,intvpage)  
  在快表中交换出一个可用项，并填入pte的信息。
* voidtlb\_clear()  
  将所有快表项置为无效，进程切换时进行调用。

**3.5设备模块**

**3.5.1作用**

模拟文件系统，管理磁盘文件和交换区资源。

**3.5.2主要数据**

* SwapFile,FSFile  
  交换区文件和磁盘文件。当内存中页面使用完时，将会有页面被交换到SwapFile中，而在本模拟程序中，由于程序指令由用户通过终端手工输入，所以FSFile并无实际作用。
* SwapCount[SWAPSIZE]  
  记录交换区中块引用次数。
* FSBlock[FSSIZE]  
  管理磁盘块的信息，具体字段含义见[外存块结构fs\_block\_t](file:///D:\\homework\\%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F\\39051718_%E7%94%B0%E5%8B%87_%E8%99%9A%E6%8B%9F%E5%86%85%E5%AD%98\\README.html" \l "外存块结构fs\\_block\\_t)

**3.5.3主要函数**

* intempty\_swap\_block()  
  在交换区中寻找一个空闲的块并返回其下标。
* voidswap\_in(byte\_t\*mem,intblock)  
  将swap区中的一块加载到mem指向的内存。
* voidload\_block(intmem,intblock)  
  将文件系统中的一块加载到内存。
* voidswap\_out(byte\_t\*mem)  
  将内存中的一页数据写入到交换区。

**3.6静态程序模块**

**3.6.1作用**

提供多个程序的段信息，以便模拟程序加载过程序，而不是随机生成每一页的控制位。

**3.6.2数据**

* Program[PROG\_N]  
  程序的数组。

**3.7客户终端模块tty.c**

**3.7.1作用：**

模拟一个进程的终端，通过终端像虚拟机发送指令。

**3.7.2主要数据**

* Buffer[BUFF\_SIZE]  
  用于与虚拟机通信，接收消息及输出指令的缓冲区。
* Request  
  当前的请求。
* Continue  
  是否继续，当用户输入q指令时，Continue将被置为FALSE,程序将释放资源准备退出。

**3.7.3主要函数**

* boolparse\_command()  
  从终端中读一行指令，进其进行解析，将解析的结果保存在Request中。
* intmain()  
  程序入口，初始化资源，并向虚拟机发出NEW\_TASK指令，程序进入解析指令循环。

**4功能实现**

**4.1快表**

通过tble\_tTLB[TLB\_SIZE]进行模拟快表结构。虚拟机启动时，快表进行初始化，将所有表项置为无效。每次访问地时，先从块表中查找有效且虚页号匹配的表项，并对所有表项的freq信息进行更新。当没有找到时，再查找页表，并且根据查找到信息对快表进行更新。

快表只对当前进程有效，所以当程序进行切换进程时，快表将进行清空操作，将所有快表项置为无效。

**4.22级页表管理**

2级页表的好处是操作系统只需对需要的页表分配内存，从而可以节省大量内存。本模拟程序对此功能进行了很好的模拟。

当新建一个进程时，页目录将会动态地被分配，然后只分配程序的data段页表和text段页表，对于堆栈断并不分配其页表，只有当虚拟机需要访问该段时，才动态地分配该页表，并将其指内存的０块。

进程退出时，将先释放页表的指向的页面资源，再释放页表空间，最后释放目录空间。

**4.3程序加载**

本模拟程序对实际程序的结构进行了简化，我们假设：

1. 程序分为３个段：data段，只读;text；可读可执行；堆栈段，可读可写，不可执行。
2. 程序中不包含共享库。
3. 程序从地址０开始加载。

当程序加载时，data段和text段的页表立刻分配内存，通过lazy\_load将页表指内文件系统中的块地址，但是此时并不将其数据加载到内存，而是延迟到对该数据进行操作时再加载。对于堆栈段内的页表，如果此时已经在页目录中分配了，则将其指向０区，即将其page\_ID置为－１,否则不分配内存,留待使用时分配。

**4.4进程控制**

虚拟机启动时，自动生成１号进程TaskInit，以后每启动一个客户终端，都将向虚拟机发送一个NEW\_TASK请求。虚拟机接收到该请求后，通过模拟的fork操作生成一个新的Task，新的Task和TaskInit共享地址空间，它们的页表指内相同的物理页。

当调用RUN请求时，虚拟机加载指定的新程序，替原进程的地址空间。

每次虚拟机接收到新请求时，都会检查请求是不来自当前进程，如果不是，调用switch\_task进行切换，同时清空快表。

收到退出的请求时，虚拟机会释放该进程所占有的所有资源，并将当前进程切换到TaskInit进程。

**4.5最近最少使用算法**

每当访问一个内存时，虚拟机都将物理页面的access位置为TRUE。同时，虚拟机维护一个时间信号，每当时钟信号到来时，都将调用memmap\_refresh，将所有的表项的freq字段右移一位，且最高位置为accessed。这样，freq越大，表示最近使用次数越多，反之越少。当选择牺牲页时，选择一个最小的即可。

**4.6内存共享**

内存共享是本模拟程序提供的高级功能,通过使用共享内存，可以大幅度提高内存使用用效率。

**4.6.1fork操作**

新进程建立过程中，其通过TaskInit的fork操作产生了与TaskInit指向相同页面的页目录和页表，直接共享父进程的地址空间。

**4.6.2加载相同程序**

如果两个进程加载了相同的程序，则它们的页表在分配时会指向相同的外存块。此时调用page\_in作，会查找对应块的引用情况，如果引用数大于０，则查找其引用的页表，检查该页是否已经被修改。如果已经修改，则重新分配一段内存页，从外存块中将数据加载到内存；否则将新进程的页表指向对应的页面，并将进程的写时复制置１，实现共享内存。

**4.6.3共享页面的交换处理**

本程序对共享页面进行加锁，一率不允许交换。

**4.7缺页处理**

在任一时刻，一个进程的特定页面只可能存在４个地方：

|  |  |  |
| --- | --- | --- |
| 1 | 外存块 | valid=FALSE,swapped=FALSE |
| 2 | 内存中，即物理页面 | valid=TRUE |
| 3 | 交换区中 | valid=TRUE,swapped=TRUE |
| 4 | 0区(一个虚拟的页面，其数据全部为０) | valid=FALSE,page\_ID=-1 |

在进行缺页处理时，虚拟机会判断该页面是否可以共享页面，如果不可以，则选择一个没有使用的页面。如果所有页面都已经使用，根据最近最少使用原则，选择一个页将其交换出内存，然后根据页面所在的位置，调用相应的函数将数据加载到选择的空闲页面中。

**4.8写时复制的实现**

在每个页表项中，都有写时复制位，如果为１则表示该页处于写时复制状态。当一个进程的某个页面要和其它进程共享页面时，所有共享该页面的进程对应的页表项都要将写时复制位置１。

进行写操作时，必须先检查该页的写时复制位，如果为０，表示没有页面共享，继续执行操作即可。如果为１，表示\*可能\*还有其它进程在使用该页，进一步检查物理页面的lock字段，如果lock大于０，则该页面正被多个进程引用，必须先将该页面复制到一个新的页面，将\*当前进程\*的写时复制位置０，然后才可以进行写操作；如果lock位<1,则共享页面已经复制，只要将当前进行的该页表项写时复制位置０，就可以进写操作。

**4.9多道程序并发及客户终端与虚拟机通信的实现**

**4.9.1多道程序实现**

虚拟机程序通过一个进程模拟，多道程序通过多个客户终端进程模拟。用户通过客户终端向虚拟机发送指令请求。请求中包含有该进程的进程号，虚拟机程序借此对客户终端进行判别和进行进程切换操作。

**4.9.2通信**

虚拟机进程创建一个服务管道，客户终端向其中写请求，虚拟机从中读取请求并处理请求。每个客户终端有一个命名管道，虚拟机进程处理完请求后，将应答写入对应客户进程的命名管道，客户终端读取该管道并显示。

**5运行说明**

本程序的开发环境为Ubuntu11.10操作系统，采用c语言开发，使用gcc4.6.1编译。程序分为２个部分，分别为vmachine和tty。tty可以有多个实例用以模拟多个进程，在运行tty之前，必须先运行vmachine。

本程序的操作命令均从tty输入，tty将请求传给vmachine进行处理，vmachine会输出详细的处理过程，并结果发回到tty显示.

命令:

Q:退出程序。

Rxxxx:从地址xxxx读一字节。

Wxxxxxx:将值xx写入地址xxxx.

Xxxxx:执行地址xxxx的指令（实际上只做可执行权限的检查).

En:加载程序n,替换当前程序。

D:显示当前进程的页表。

S:关闭模拟程序。

**6测试说明**

本程序在Ubuntu11.10环境下测试。bash版本为4.2.10.在debug目录下可以运行debug.sh脚本自动测试，该脚本会按照debug.dat和debug2.dat中的指令向tty输入测试指令。用户也可以手工测试，方法为:

1. 运行vmachine
2. 运行一个或多个tty程序
3. 在tty中输入特定指令，观察程序输出。

在debug目录下的img目录中，给出了一些测试用例截图，可供参考。

**7总结与感想**

本程序是北京航空航天大学计算机学院操作系统第３次课程设计。从编写第一行伪代码，到撰写本文档，历时２个月。代码由田勇(39051718)完成，王博弘、梁炯潜、郭莉莎参与了讨论设计，并给出了一些很好的建议。

在设计本程序时，最初参考了实验教程上给出的程序，但很快发现该程序不能很好模拟操作系统对虚拟内存的管理。于是决定另起炉灶，从新设计一个更加接近真实系统的模拟程序。

最初的设计是用２个文件分别模拟磁盘文件和交换区。程序中有一解释程序，可以执行事先写入磁盘文件中的由一系列指令构成的虚拟程序。客户终端只用来控制执行哪一个程序。这样，实际上是要实现一个虚拟机。但后来发现其工作量超出预计数倍，不得不忍痛删除大量代码，只保留与虚存管理相关的部分。从主程序名vmachine、进程控制块task\_t、设备控制模块等处，仍可以看出当初设计的痕迹。

精简后的设计并没有简化对虚存管理的模拟。除了实现一些基础的功能以外，本程序还实现了共享内存、写时复制等高级功能，不仅如此，对于进程的生存周期，本程序也进行了很好模拟。但同时，复杂的功能也带来了复杂的代码和调试的困难。在本文档中，各个功能是分开描述的，但在代码中却难以分开实现。大多数情况下，多个功能都杂糅在一起，使得对数据的操作过程十分繁琐复杂，牵一发而动全身。这样，不仅查找错误十分困难，修改起来也不简单，往往改动一处，又导致另一处错误。因此，在提交本次作业时，可能仍然存在未发现的错误。

通过本次作业及小组讨论，本组成员对操作系统虚存管理有了更加深刻的理解，同时也掌握了程序运行的机理，并且对Linux编程环境更加熟悉，很好的达到了本次大作业的目的。