|  |  |  |
| --- | --- | --- |
| 南农大  **计算机操作系统课程设计**  **实践报告**  XH2 | | |
|  | 题 目: | Linux2.6进程管理系统的仿真实现 |
|  | 姓 名: | 梁嘉文 |
|  | 学 院: | 信息科技学院 |
|  | 专 业: | 计算机科学技术系 |
|  | 班 级: | 计科161 |
|  | 学 号: | 19216126 |
|  | 指导教师: | 姜海燕 职称: 教授 |
| 2019年3月 15 日 | | |

**目录**

[1 程序结构说明 2](#_Toc3555056)

[1.1 程序结构图 2](#_Toc3555057)

[1.2 程序结构描述 2](#_Toc3555058)

[2 模块论述设计与实现 2](#_Toc3555059)

[2.1 内存模块 2](#_Toc3555060)

[2.1.1 读取内存数据 3](#_Toc3555061)

[2.1.2 写入内存数据 3](#_Toc3555062)

[2.2 硬盘设计 3](#_Toc3555063)

[2.2.1 读取硬盘数据 4](#_Toc3555064)

[2.2.2 写入硬盘数据 4](#_Toc3555065)

[2.3 cpu设计 5](#_Toc3555066)

[2.3.1 mmu设计 5](#_Toc3555067)

[2.3.2 计时器设计 5](#_Toc3555068)

[2.3.3 寄存器设计 6](#_Toc3555069)

[2.4 地址线和数据线 6](#_Toc3555070)

[2.5 页面管理 6](#_Toc3555071)

[2.5.1 伙伴算法分配内存 6](#_Toc3555072)

[2.5.2 回收内存空间 6](#_Toc3555073)

[2.5.3 虚存空间分配内存 7](#_Toc3555074)

[2.5.4 回收虚存空间 7](#_Toc3555075)

[2.5.5 页面生成 7](#_Toc3555076)

[2.5.6 页面修改 7](#_Toc3555077)

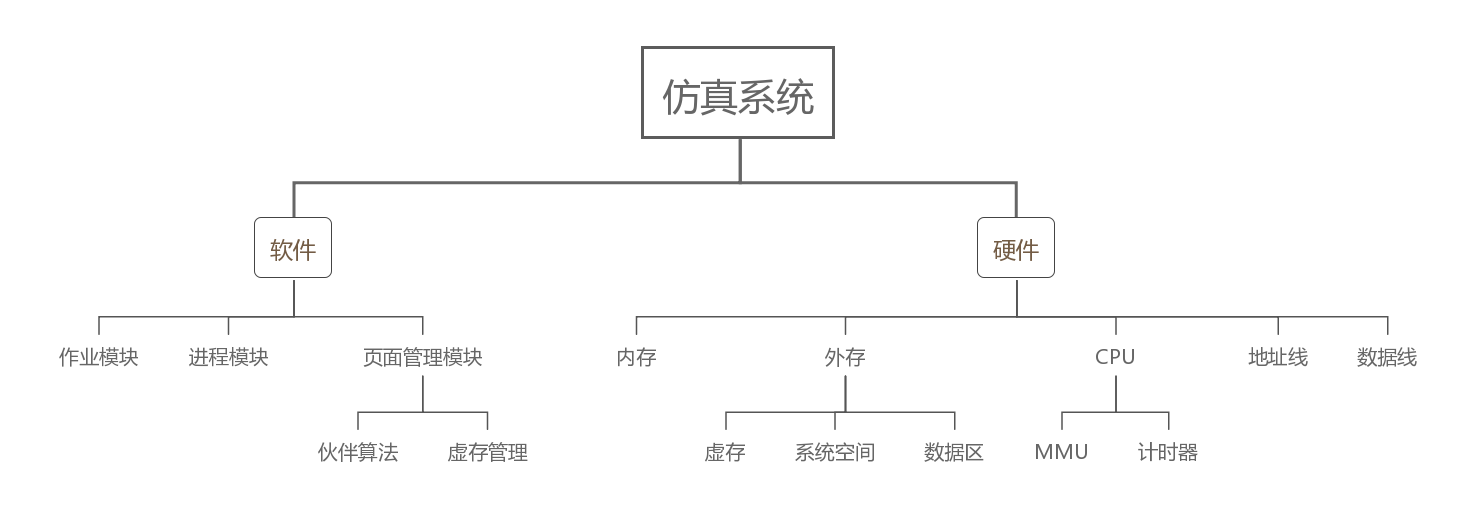
[3 技术问题分析报告 8](#_Toc3555078)

[4 参考文献 8](#_Toc3555079)

[5 个人总结 9](#_Toc3555080)

**1 程序结构说明**

**1.1 程序结构图**



**1.2 程序结构描述**

1.2.1 在仿真系统的软件部分中，页面管理部分涉及到了对于内存的管理和对于虚存空间的管理。在讲所有空间进行分页之后，进行内存管理时，采用linux2.6所使用的伙伴算法；在管理虚存空间时，使用简单页式存储管理进行统一管理。

1.2.2 在硬件部分中，对内存、外存、cpu三个部件分别建立文件作为仿真设备，系统中的类对象体现为文件的映射；对地址线和数据线两个部件只进行类对象的实例化，模拟系统中实际存在的硬件设备，但并没有对应的文件仿真。

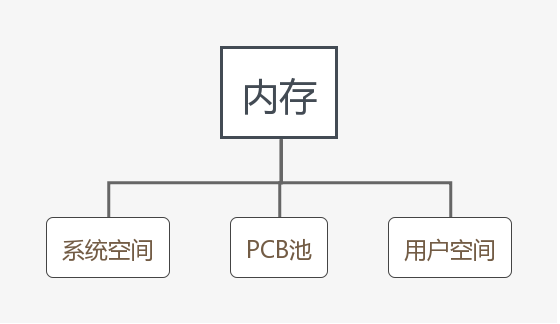
1.2.3 在CPU类中设计MMU单元和计时器单元，负责模拟CPU中的地址变换机构和CPU内的计时器部件。

**2 模块论述设计与实现**

**2.1 内存模块**

内存作为操作系统中最核心的部件之一，需要频繁与CPU进行数据交互，频繁进行读写操作。而内存本身作为存储器，并没有任何可编程的能力和执行操作的能力。故在裸机设计中，对于内存模块的功能定位为空间有限的高速存储设备，在设计时着重体现其读写功能。

由于内存类的实例对象是内存文件的映射，故设置内存类对象在加载时将文件中所有数据导入。



内存空间共32KB，用户空间共16KB，按照每512B一页的设计，占据页号32-63，分配时使用伙伴算法，回收时以单个页面为单位进行回收。

**2.1.1 读取内存数据**

GetData(地址address)

{

将address信息写入地址线address\_line

读取地址线信息address\_line作为偏移地址在内存类的数据中进行寻址

将寻址到的内存单元中的数据组合成short型数据并写入数据线data\_line中

读取data\_line中的数据并作为返回值返回

}

**2.1.2 写入内存数据**

WriteData(地址address，数据data)

{

将address信息写入地址线address\_line

将data信息写入数据线data\_line中

读取地址线信息address\_line作为偏移地址在内存类的数据中进行寻址

向寻址到的内存单元中写入data\_line中的数据，并将该short型数据拆分成两个字节分别存储

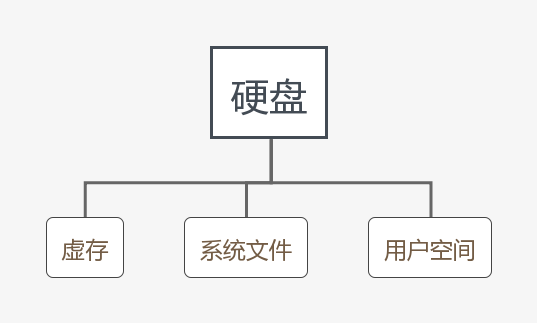
}

**2.2 硬盘设计**

硬盘作为计算机系统中存储数据的中药设备，需要存储大量数据信息，以及存储程序文件。但硬盘本身作为存储器，同样并没有可编程的能力和执行操作的能力。故在裸机设计中，对于硬盘的功能定位为空间足够的长期存储设备，在设计时需要体现其读写功能。

由于硬盘类的实例对象是硬盘文件的映射，故设置硬盘类对象在加载时将文件中所有数据导入。

由于硬盘空间大小超出16位地址线寻址空间，故对地址线采取分时复用的方式，分三次传输物理地址的偏移量、扇区号、磁道号，并按照特定方式进行组合，使三维地址与一维地址之间建立映射关系，既满足系统中对于地址统一管理的需要，又满足硬件条件的物理限制。



硬盘空间共1MB，前64KB为硬盘与内存的交换区，即虚存空间，中间16KB存储系统文件，在开机时调入内存的系统区，其余空间均为用户空间，可存储作业信息和数据。在管理虚存空间时，对于虚存空间的申请以页为单位，在可用空间中分配足够数量的页后停止。释放时可对单独的页进行释放。

**2.2.1 读取硬盘数据**

GetData(地址address)

{

将address信息写入地址线address\_line

读取地址线信息address\_line作为偏移地址在硬盘类的数据中进行寻址

将寻址到的硬盘单元中的数据组合成short型数据并写入数据线data\_line中

读取data\_line中的数据并作为返回值返回

}

**2.2.2 写入硬盘数据**

WriteData(地址address，数据data)

{

将address信息写入地址线address\_line

将data信息写入数据线data\_line中

读取地址线信息address\_line作为偏移地址在硬盘类的数据中进行寻址

向寻址到的硬盘单元中写入data\_line中的数据，并将该short型数据拆分成两个字节分别存储

}

**2.3 cpu设计**

Cpu作为系统的核心部件，包括寄存器堆、mmu、计时器等几个部件，用于支持系统的各个功能模块的工作，在各个功能模块中协调工作。

**2.3.1 mmu设计**

Mmu负责将虚拟地址转换为实际地址，由于所有空间均采用页式空间管理，故在mmu中的转换逻辑应为虚拟地址转换为页号，页号转换为页框号，再根据页框号找到对应的实际地址。同时在mmu中，设置TLB用于快速查询页号所对应页框号，提高CPU利用率。在查询页号对应地址时，若TLB命中则直接返回结果；若TLB未命中且内存命中，则将该块信息添加至TLB中，替换从TLB中删除LRU算法选择的块；若TLB与内存均为命中，则产生缺页中断，同时更新TLB与内存，均使用LRU算法选择被替换的单元。

在保证该页位于内存中时，可使用以下函数进行地址转换

VirtualAddressToRealAddress (pcb块，虚拟地址virtual\_address)

{

提取虚拟地址的偏移地址

在TLB中查找virtual\_address对应的页号

If(存在) 使用页号查询物理地址

Else

{

根据逻辑地址计算页号

将虚拟页号与物理页号信息添加至TLB被替换单元中

根据物理页号查询物理地址

}

获取物理地址作为基地址，加上偏移地址后作为实际的物理地址返回

}

PageToRealAddress(物理页号page\_num)

{

If(page\_num范围在内存中)根据页号计算物理地址

Else

{

根据页号计算物理地址

将物理地址转换成硬盘的三维地址

将三维地址按照硬盘的地址转换方式重新组合成一维地址进行返回

}

}

**2.3.2 计时器设计**

计时器类中运行一个线程负责产生时间戳，对时间戳进行自增处理模拟系统内核时间的自增。同时每隔一段时间使中断信号发生改变，产生一次中断信号，

**2.3.3 寄存器设计**

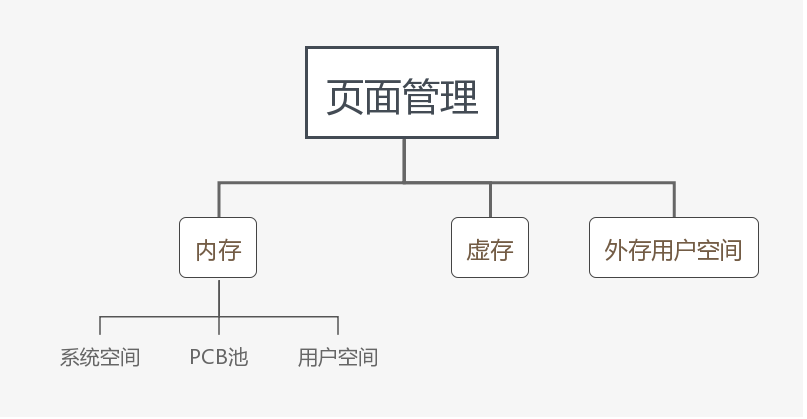
在cpu中设置PC、IR、CR3、PSW等寄存器，存储程序的执行情况。同时设置pcb指针current\_pcb用于定位当前执行的进程的pcb块，在切换进程之后负责恢复现场。

**2.4 地址线和数据线**

由于地址线与数据线并不是独立存在的硬件单元，但在硬件仿真中不可缺少，故对这两个对象分别抽象为类，设置对应的读写函数用于操作，但不设置文件作为硬件的虚拟实现方式。

**2.5 页面管理**

将所有存储空间（32KB内存+1MB硬盘）统一按照512B每页的形式进行划分，共计2112页，第0-63页为内存空间，64-192页为虚存空间，第224-2111页为硬盘的用户空间。对于不同区间的页面，采取不同的管理方法。



由于虚存空间相对而言足够大，故选择在高级调度时将作业的所有信息全部调入至虚存中，再由缺页中断调入内存；相应的，在内存空间不足时，也会使用LRU算法将某些进程中最久未使用的页中的数据调回至虚存中，增加内存可用空间。

**2.5.1 伙伴算法分配内存**

Applymemory(需求的页号数量page\_need)

{

向上取整计算page\_need最接近的2的整数次幂k

在空闲链表中找到大小为k的链中的第一个空闲块号

if(还有符合条件的空闲块)

{将空间分配给申请的进程，并自上而下修改bitmap中记录数据|

else

{返回空，分配内存失败}

}

**2.5.2 回收内存空间**

Recyclepage(需要回收的页号page\_num)

{

将page\_num页所对应内存空间填充为全0

递归修改page\_num所在bitmap图，若有伙伴块被一起释放则修改上层bitmap图状态  
}

**2.5.3 虚存空间分配内存**

ApplyDisk(需求的页号数量Page\_need)

{

While(Page\_need>=0)

{

在虚存中顺序查找未使用的块号

向申请的进程分配一个空闲块

继续查找下一个空闲块，并将Page\_need-1

}

返回分配情况，使用字符串存储分配信息  
}

**2.5.4 回收虚存空间**

Recyclepage(需要回收的页号page\_num)

{

将page\_num页所对应内存空间填充为全0

将page\_num页的使用情况修改为未使用

在申请该页的进程中删除对该页的占用信息  
}

**2.5.5 页面生成**

在新建页对象时，使用构造函数生成页，数据产生于内存中页面对应地址。

Page(页号num)

{

调用mmu中的功能，将页号转换为物理地址address

For(i=0;i<512;i+=2)

{

将地址address+i处short类型数据复制到page类数据对象中  
}

}

**2.5.6 页面修改**

在修改页中数据时，不仅修改页面中数据成员，同时重新寻址并修改对应内存中数据。

在交换操作时，由于页对象已生成，对其数据无法进行修改，故需要重新生成页查看修改后的信息。

**3 技术问题分析报告**

问题1 在页中设置数组存放页内数据，在对页面数据进行修改时无法同步对内存中数据进行修改。

解决方法： 取消页中存放数据的数组，在修改页数据时新建页对象，直接使用内存空间作为页面数据存放的空间，在修改页面时直接修改内存数据。符合页面作为虚拟逻辑结构的实际情况。

问题2 伙伴算法的设计

解决方法：通过查询资料，使用bitmap算法，建立不同页面数大小的空闲队列，　使用位运算作为提取bit情况的工具。设计时考虑到设计的统一性，所有队列均采用最大空间即３２位设计，不应被使用的位补０，避免误操作。同时，在分配和回收页面时采用递归设计，分配时自上而下分配指定大小的页面空间给进程，且额度恒定位２的k次方，在可允许的页面浪费情况下减少了资源回收和分配的算法复杂度。ｋ较大的页所包含的页为k-1层两个相邻的页，即处于伙伴关系的页。

问题３ 分配页号时需考虑不同设备、不同逻辑区间的区别

解决方法：在系统区间中，允许根据页号进行内存单元的读取操作，但禁止访写入操作；在用户空间中，资源的申请和释放需要通过伙伴算法相关函数进行，通过页号可查询到是否处在使用中；在续存空间中，可直接申请需要的页面数量，并进行分配，释放时可直接根据页面序号进行释放。统一编排页号后在具体函数中添加条件判断，用于检测页号是否在合法空间中，实现对非法操作的拒绝处理。

问题４ 在设计页面监测的UI界面时需要对界面进行动态刷新

解决方法：经过查阅资料，对于不同功能的模块分别建立类并进行实例化，对每个类分别继承JPanel类，重载其paint函数，实现对于界面的重绘，完成功能需求。

问题５ 调用其他类的实例化对象时保证结果唯一

解决方法：对于每一个可能在外部进行调用的类，分别创建该类的静态对象，保证对象的唯一性，同时避免了在生成类对象时可能的系统情况差异，保证了程序逻辑的合理。

**4 参考文献**

[1] [27Up](https://me.csdn.net/lcl497049972). 伙伴算法buddy

[EB/OL].https://blog.csdn.net/lcl497049972/article/details/82954124

[2] [Unique-You](https://me.csdn.net/qq_22238021).伙伴(buddy)算法及位图

[EB/OL].https://blog.csdn.net/qq\_22238021/article/details/80208630

[3] [yilongdashi](https://me.csdn.net/yilongdashi).操作系统-虚拟存储管理技术之虚拟页式存储管理

[EB/OL].https://blog.csdn.net/yilongdashi/article/details/82728780

[4] [Givefine](https://home.cnblogs.com/u/wxd0108/).[Java中的多线程你只要看这一篇就够了](https://www.cnblogs.com/wxd0108/p/5479442.html)

[EB/OL].https://www.cnblogs.com/wxd0108/p/5479442.html

[5][大脑补丁](https://me.csdn.net/x541211190). Java用JFrame、JPanel、Graphics绘图案例讲解

[EB/OL].https://blog.csdn.net/x541211190/article/details/77414861

[6] Daniel P. Bovet,Marco Cesati.深入理解linux内核(第三版)[M].北京:中国电力出版社,2008

**5 个人总结**

经过本次操作系统课设，我对于操作系统的理解程度从纸上谈兵的阶段加深到了略懂一二的程度。可以说，操作系统的每一个部分都要经过精心的设计，才能在构筑成整体的操作系统时相互协作，更好的完成操作系统的任务。

在本次课程设计实验中，我们的选题是模拟仿真Linux2.6的系统中的部分模块，由于linux系统的设计十分巧妙，花费了许多时间阅读相关参考文献才在心中有了各个模块大致的实现方法。

在此次的课程设计中，我主要负责：CPU部件的仿真、内存空间的仿真实现、MMU地址变换、页表设计实现、页面调度算法、页面分配与回收算法、等几个方面内容的撰写，同时设计了内存检测窗口和CPU信息查看窗口。

由于功能较为底层，在设计时尽可能精简，只保留了该部分最核心的功能，以便搭建起逻辑较为简单的仿真系统运行的平台，在不影响平台运行的情况下降低逻辑复杂度。

在此基础上，对整个存储空间使用页式存储管理，既方便了伙伴算法的建立，又降低了逻辑复杂度。代价是对程序员较不友好，但在本次仿真中并不考虑程序员方的需求，故此代价可忽略。

由于能力不足，系统中尚有不足之处，需在今后的时间里逐步完善，最终完成一个较为贴近真实的linux2.6仿真系统的编写。