操作系统原理实验报告

|  |
| --- |
| **实验报告**  学院：数据科学与计算机学院 专业：计算机科学与技术 年级：2016级  实验人姓名（学号）：王锡淮（16337236）  日期： 2018 年 3月 16日 |
| **实验2 加载用户程序的监控程序**  **[实验目的]**   1. 学习加载用户程序并执行程序、控制程序。 2. 学习BIOS的调用和使用原理。 3. 学习各种中断的使用。 4. 使用批处理系统的原理和实现。 5. 学习内存和外存的安排管理。   **[实验要求]**   1. 设计四个有输出的用户可执行程序。 2. 设计一个最简单的系统，允许键盘输入，用于指定运行这四个有输出的用户可执行程序之一。 3. 自行组织映像盘的空间存放四个用户可执行程序   **[实验内容]**   1. 对于用户可执行程序，基本来自实验1：用字符‘O’从屏幕左边第五行位置45度角下斜射出，保持一个可观察的适当速度直线运动，碰到屏幕的边后产生反射，改变方向运动，如此类推，不断运动；同时在左上角显示出姓名的拼音以及学号；将屏幕背景色色设置为紫色，并不断改变字符‘O‘的前景色，实现炫酷动态变色；同时增加交互式清屏功能，按下’c’键则将之前的轨迹清空。四个程序的出发相对位置不相同。 2. 在进入用户可执行程序之前，有字符画形式的‘王锡淮’显示出来；进入之后，有用户提示。 3. 在当前运行的程序执行完毕之后，通过按键1~4，分别选择执行左上角、右上角、左下角、右下角的用户程序。 4. 安排映像盘的空间如下：     各名字的含义见下面的**[实验方案]。**   1. 实现基于进程的操作系统。如下图所示（来自[1]）：   屏幕剪辑  将向屏幕上输出’O’并变换色彩、打印名字学号等信息并打印用户提示、检测是否应该清空屏幕、打印字符画名字等系统功能模块单独分离开来，作为进程运行，再单独定义进程切换函数，将系统作为一系列分离进程的集合来实现。   1. 使用多个文件配合。将4个用户程序和操作系统的例程以及引导盘分开，并使它们能够配合运行。   **[实验方案]**   * 实验工具以及环境­   实验器材：笔记本电脑一台。  实验环境：windows10系统。  实验工具软件：sublime text 3编辑程序代码=>nasm编译汇编代码=>bximage创建与修改软盘镜像（内置于bochs）=>dd for windows用于修改软盘镜像文件=>bochs用于虚拟仿真运行和调试程序=>vmware用于最后的展示。   * 虚拟机配置方法[2]   Vmware的配置方法在实验1时已经探究明白，所以只记录bochs的成套工具的配置以及使用方法。  使用bochs的配置方法是写好配置文件，而配置文件的具体写法已经包含在bochs安装的根目录下的’bochsrc-sample.txt’文件中。下面展示我所使用的最简单版本的配置文件：  屏幕剪辑   * 方案思想   该方案的设计来源于对课本中进程的状态，进程的控制还有操作系统的执行的学习。  **进程的状态：**  在本次实验中实现的是最简单的二态进程模型，即一个进程拥有运行态和未运行态，  屏幕剪辑  当收到派遣命令（这里是按下对应按键）时，转为运行态。  此外，还可以维护一个队列，其中记录着要按顺序运行的进程ID，适时调度。  **进程的控制：**  进程的创建在本实验中体现为对应的按键按下，程序被执行；而进程切换的场景本应有三种：  屏幕剪辑  但是在本实验中为了简化设计，只考虑陷阱和系统调用，没有考虑中断的原因是自定义int 9h的结果出现错误，无法实现在中断（按键按下）出现时返回监控程序。  **操作系统的执行：**  操作系统的执行有三种经典模式：  分离的内核、在用户进程内执行操作系统例程、操作系统作为分离的进程执行。  本实验中选择第三种，即将操作系统例程作为分离的进程执行。将本实验中系统的各个子功能分离出来，作为常驻例程存放在内存中。  **单道批处理系统：**  本次实验中实现的是最简单的单道批处理系统。  屏幕剪辑  **多文件的设计：**  在本实验中将各个程序，包括监控程序、系统例程、用户程序分开多个文件存放的原因是模仿现代操作系统，在这个设计中主要涉及的是对于程序位置、起始地址的控制。  **用户程序的设计：**  用户程序主要来源于实验1，但是修复了原来算法中存在的可能从四个顶点穿出去的bug，也重构代码使其能够复用。   * 实验原理   **关于在屏幕上显示：**  除了在实验1中用到的直接向0xB800后的4000个字节写入数据，还可以使用BIOS提供的int 10h/13h中断，该中断的使用方法如下：  屏幕剪辑  可以方便的在屏幕上显示字符串。  **对用户程序的修改：**   1. **如何修复穿墙bug：**   将在实验1中使用的代码重构，碰撞只分为碰撞上下边界和左右边界两种，便可以方便的判断出是否到达四个顶点。   1. **代码模块化，明确接口定义：**   为了方便复用，将用户程序中的各种功能分离出来作为模块，并且定义接口。  **如何装载程序：**  使用int 13h/02h能够将指定扇区指定扇区数目的软盘内容拷贝到指定的内存位置。  屏幕剪辑  需要注意的是由于事先知道了将要拷贝到的目的位置，必须事先指定程序的起始地址，否则会出现偏移量不对应的问题。  **如何运行指定的程序：**  当程序需要被调用时，包括用户程序和系统例程，对于用户程序是按键按下，对于系统例程是产生系统调用，便使用call指令跳转到事先已经安排好的内存地址。  **用户程序何时会返回监控程序：**  如前所述，有三种情况下用户程序会返回监控程序，对于第一种中断返回，可以考虑自定义int 9h中断，无论何时按键被按下，int 9h被自动调用，而自定义的功能便可以使得按下的按键对应于一个进程ID，存储在队列中，当上一个进程执行完毕时，就调用队列中下一个进程。对于第二种陷阱返回，在本实验中可以指定为运行到一定次数后自动返回监控程序。对于第三种系统调用，由于在本实验中的系统调用都是为了配合用户查询的系统例程，这些例程运行完毕之后会返回用户程序。  **关于PC机键盘的处理过程以及int 9h在其中发挥的作用[3]：**  键盘上的每一个键相当于一个开关，键盘中有一个芯片对键盘上的每一个键的开关状态进行扫描。按下一个键时，开关接通，该芯片就产生一个扫描码，扫描码说明了按下的键在键盘上的位置。扫描码被送入主板上的相关接口芯片的寄存器中， 该寄存器的端口地址为60h。松开按下的键时，也产生一个扫描码，扫描码说明了松开的键在键盘上的位置。松开按键时产生的扫描码也被送入60h 端口中。  键盘的输入到达60h 端口时，相关的芯片就会向CPU 发出中断类型码为9 的可屏蔽中断信息。CPU 检测到该中断信息后，如果IF=1，则响应中断，引发中断过程，转去执行int 9 中断例程。  BIOS 提供了int 9 中断例程，用来进行基本的键盘输入处理，主要的工作如下：（1）读出60h 端口中的扫描码；  （2）如果是字符键的扫描码，将该扫描码和它所对应的字符码（即ASCII 码）送入内存中的BIOS 键盘缓冲区；如果是控制键（比如Ctrl）和切换键（比如CapsLock）的扫描码，则将其转变为状态字节（用二进制位记录控制键和切换键状态的字节）写入内存中存储状态字节的单元；  （3）对键盘系统进行相关的控制，比如说，向相关芯片发出应答信息。  从上面的内容中，可以看出键盘输入的处理过程：①键盘产生扫描码：②扫描码送入60h 端口：③引发9 号中断：④C PU 执行int 9 中断例程处理键盘输入。  所以我们能够改变的只有int 9h中断处理程序处理。  改变的方法为先编写自定义的int 9h中断，在这个中断中不仅要有新的功能，还要调用旧的int 9h中断来处理硬件细节，这就要求先将原来的int 9h中断在中断向量表的入口地址保存起来，而对于8086PC机中断向量表放在从内存0000:0000到0000:03FF的1024个单元，并且将新的int 9h中断的入口地址存放在中断向量表的对应位置。  要在新的int 9h中断中调用旧的int 9h中断，还要学会模拟中断的调用过程。也即模拟（1）标志寄存器入栈；（2）设置中断屏蔽位；（3）CS、IP入栈；（4）(IP)=(n\*4), (CS)=(n\*4+2)。以上过程可以通过（1）pushf；（2）设置IF=0，TF=0；（3）call dword来实现。   * 程序流程和算法思想   本程序的流程如下图所示：    左边的是整个系统的运行流程图，而右边的则是用户程序的流程图。  本实验中的算法思想是简单批处理操作系统的算法思想，即不断执行进程，当一个进程执行完毕再执行下一个进程。   * 数据结构与程序模块功能   **数据结构**  在设计中，完成加载后的内存组织和软盘的分布如下：    右图中每一个方块代表512字节。  **程序模块**   1. 加载程序   在本次实验中要加载的程序较多，但是基本上大同小异，所以选择加载用户程序的代码加以说明：  屏幕剪辑  屏幕剪辑  这两段代码的功能是从软盘中读取程序进入内存，根据cx计算出设计好的内存位置和软盘扇区位置。   1. 延时模块   屏幕剪辑  该模块在本实验中被多次使用，因为需要的延时时间不尽相同，所以没有将其作为系统例程。   1. 其余模块与实验1中类似。  * 代码文档组成说明   屏幕剪辑  ball\_A.asm、ball\_B.asm、ball\_C.asm、ball\_D.asm是四个用户程序，boot.asm是引导盘也即监控程序代码，checkInput.asm是检测有无键盘输入，cleanPrint.asm是清空屏幕，printBigName.asm是打印字符画名字，printnames.asm打印提示信息，run\_YouKnowMe.bat是脚本文件，包括整个流程，showBall.asm是再某个位置打印字符，boot.flp是软盘镜像文件。  **[实验过程]**   * 主要流程   **第一步：**使用bochs内置的bximage程序创建一个大小为1.44MB的软盘文件。注意定义要使用bximage，如果在外部创建，比如使用winhex进行创建，会导致bochs虚拟机提示”no bootable device”，导致错误。而具体过程如下图：  屏幕剪辑  **第二步：**使用dd for windows处理文件的拷贝。Dd是unix或unix-like上的一个文件复制工具，想要的windows上使用dd，可以选择下载dd for windows，或者使用Cygwin在windows下使用linux的命令行命令。本次实验一共需要使用10个文件，相对于的批处理代码如下：  屏幕剪辑  要注意的是seek选项可以选择将输入文件复制到跳过seek选项的参数多个block（大小由bs选项指定）的起始地址。  **第三步：**使用bochs（严格地说是其中的bochsdbg.exe）进行调试。  **第四步：**再次使用bximage将img后缀的镜像文件转化为flp后缀的VMware特定虚拟软盘格式，过程如下：  屏幕剪辑  **第五步：**使用vmware进行结果展示，在下面的“结果展示”部分详细说明。   * 输入输出说明   输入为在前一个进程退出之后的按键输入，该输入作为进程ID；  输出为向屏幕上打印的弹球程序。   * 结果展示   加载程序时的提示：    随后显示字符画界面和然后的初始界面：    分别按下1、2、3、4的运行界面：    以及在中途按下‘c’按键的情况：     * 遇到的问题以及解决情况  1. Bochs使用的.img文件不能是winhex创建的。如果使用winhex创建.img文件，bochs会显示’no bootable device’的提示信息；所以最好使用配套的bximage来创建.img文件。 2. 自定义的Int 9h无法使用。按照[3]所讲的方法自定义的int 9h中断出现了问题，在bochs下一步一步调试发现在调用原来的int 9h中断的时候，陷入了死循环，原因未知，所以只好放弃用户程序遇到中断返回监控程序的设计。 3. 文件所在内存位置与org指令。对org指令有了新的理解，org x指的不是将程序放在x处，而是事先知道了程序会被放在x处，通过org指令将所有的标签加上x。所以要将某个程序放在内存的x处，需要使用org x指令以避免标签偏移量出现不匹配的情况。 4. 子程序要注意保护上下文环境和利用接口。在编写代码的时候，由于不注意保护上下文，子程序中没有保存上下文，不经意地改变了一些寄存器，导致不必要的调试，降低了效率。   **[实验总结]**  在本次实验中要用到的原理有不少，尤其是关于进程和系统的理论，我通过这次实验对于这些内容也有了不少思考；还有的就是关于返回监控程序的方法，我也有不少思考。  在操作系统的发展过程中，操作系统从最开始的串行处理，发展到了简单批处理系统，再到后来的多道批处理系统和分时系统。在本次实验中我使用的是简单批处理系统的概念，但是我想的其实是能不能做成一个分时系统。这是一个多用户系统，通过设置进程的状态可以实现某几个用户程序“同时”进行，每一个时间间隔可以选择为移动并且打印一个字符所需的时间。但是我对于分时系统的学习还不够清楚，对于分时系统的实现要等到把理论弄清楚了。  而系统的执行有分为三种模式：无进程的内核、在用户进程中执行和基于进程的操作系统。在这三种模式中，基于进程的操作系统和微内核十分相似（或者说就是微内核吧，我不太确定），我认为这种模式不易出错，易于维护。所以在本实验当中也使用了这种思想。  我在用户程序怎么样返回监控程序这一方面做了不少思考。对于系统调用和用户程序中遇到陷阱（比如运行到一定次数就回到监控程序）是没有疑议的，但是在实现中断返回监控程序的时候发生了意外，我之前在上《计算机组成原理与设计》的时候完成过一个用x86汇编语言编写的小项目，自定义一个int 9h中断是一个有用的技巧，但是在这次实验中却没有成功，也没有找出原因来，苦思而不得其解。  还有的感言是一定要注意细节，比如我在处理内存位置和软盘位置的时候就错了挺久，原因是没有把对应关系明确地写下来。还有就是总是不小心就改了传递数据用的寄存器。  **[参考文献]**  [1] Operating Systems: Internals and Design Principles (7e).  [2] <https://www.jianshu.com/p/4ede2d1e467c>, 10分钟完成的操作系统(Bochs的使用--windows系统下).  [3] 汇编语言（第3版），王爽. |