

操作系统课程设计

设计与功能说明文档

学号	姓名	课号
1352956	鄞劭涵	42028702
1352958	金敏	42028702
1352960	张航	42028701

目 录

1	概述.....	3
1.1	项目目的.....	3
1.2	项目背景.....	3
1.3	开发环境.....	3
1.4	功能概述.....	3
2	程序使用.....	4
2.1	界面说明.....	4
2.2	使用示例.....	5
3	设计理论.....	6
3.1	界面实现.....	6
3.2	数据结构.....	6
3.3	调度算法.....	6
4	功能实现.....	6
4.1	界面实现.....	6
4.2	数据结构.....	6
4.3	调度算法.....	6

1.概述

1.1 项目目的

- 对参考源码的模块进行修改
- 实现系统级应用

1.2 项目背景

使用Linux 系统上 bochs 模拟器进行开发操作系统

1.3 开发环境

- VMware 虚拟机
- Ubuntu14.04
- Bochs-2.3.5

1.4 功能概述

(1)实现一个简易的命令行系统,支持对普通文件的创建/删除/写入/读出操作,以及文件读写保护限制。

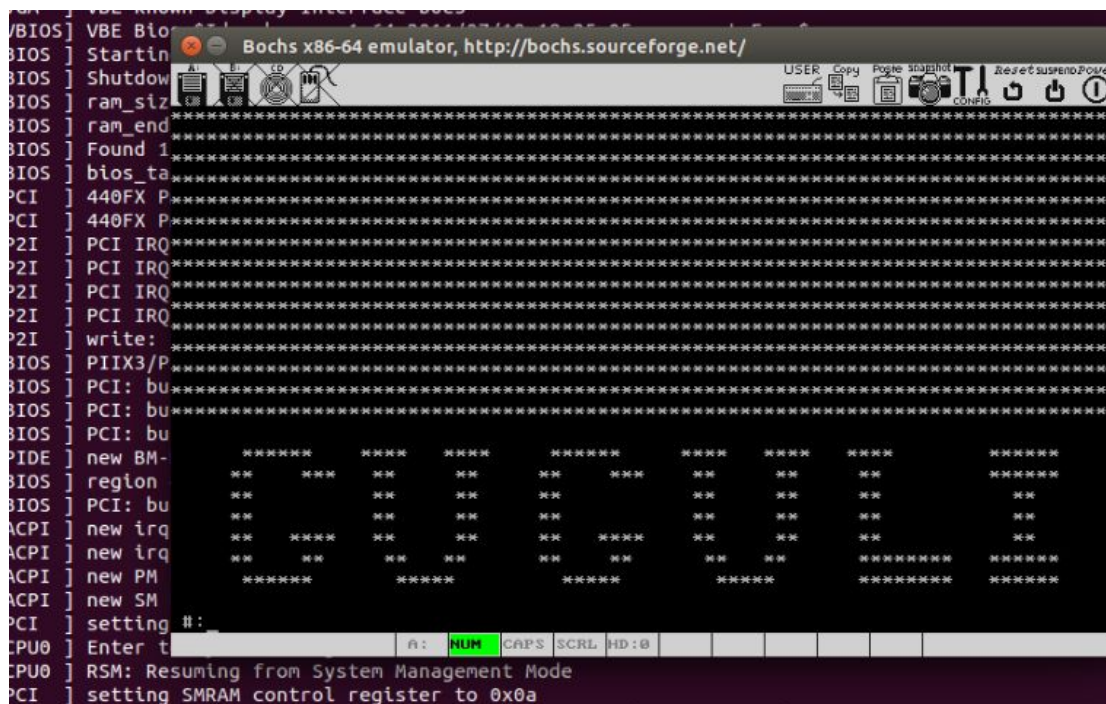
(2)对进程调度模块的丰富和优化,将原有按剩余 tick 数排序分配时间片改进为两级优先级反馈队列(按索引组织优先级队列的方法)

(3)对进程调度模块的丰富和优化,将原有按剩余 tick 数排序分配时间片改进为三级优先级反馈队列(直接对进程表 proc_table 调整顺序的方法)

2. 程序使用


2.1 界面说明

(1) 下图为我们的操作系统的初始界面，它的名字是 GUGULI OS，引导工作和内核加载过程完成之后，我们会进入内核 kernel/main.c 的命令行异步消息循环之中。'#'为命令提示符



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
VBE Bios
Startin
Shutdow
ram_siz
ram_end
Found 1
bios_ta
440FX P
440FX P
PCI IRQ
PCI IRQ
PCI IRQ
PCI IRQ
write:
PIIX3/P
PCI: bu
PCI: bu
PCI: bu
new BM-
region
PCI: bu
ACPI: new irq
ACPI: new irq
ACPI: new PM
ACPI: new SM
setting #
Enter t
CPU0: RSM: Resuming from System Management Mode
PCI: setting SMRAM control register to 0x0a
```

(1) we 命令可以显示我们的组员信息



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
VBE Bios
Startin
Shutdow
ram_siz
ram_end
Found 1
bios_ta
440FX P
440FX P
PCI IRQ
PCI IRQ
PCI IRQ
PCI IRQ
write:
PIIX3/P
PCI: bu
PCI: bu
PCI: bu
new BM-
region
PCI: bu
ACPI: new irq
ACPI: new irq
ACPI: new PM
ACPI: new SM
setting #
Enter t
CPU0: RSM: Resuming from System Management Mode
PCI: setting SMRAM control register to 0x0a

# : we
1352956 Yin Shaohan
1352958 Jin Min
1352960 Zhang Hang
```

2.2 使用示例

(1) 创建文件命令可以用来在 / 目录下创建一个文件

```
#####  
#:touch file1  
File created: file1 (fd 2)  
#####
```

(2) 模仿 linux 的重定向命令，实现对文件数据的导入

```
#####  
#:"abcde" > file1  
#####
```

(3) cat 命令可以用来显示指定文件中的前 20 个字节的信息，不足 20 个字节自动补上 EOF。

```
#####  
#:cat file1  
File opened. fd: 2  
20 bytes read: abcde  
#####
```

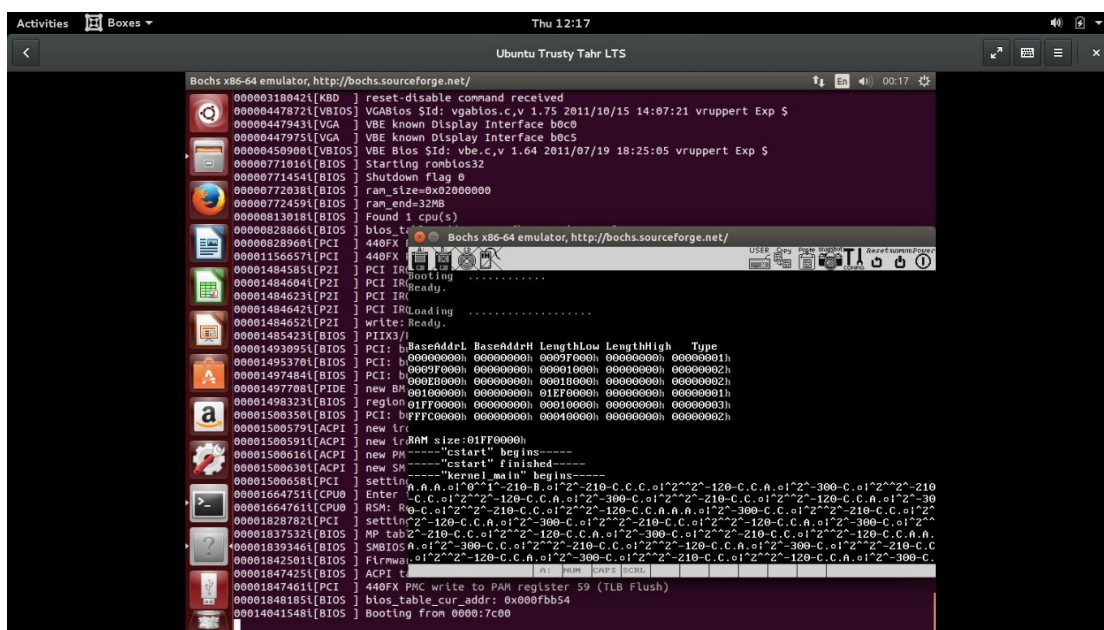
(4) 使用 rm 命令对文件进行删除

```
#####  
#:rm file1  
File removed: file1  
#####
```

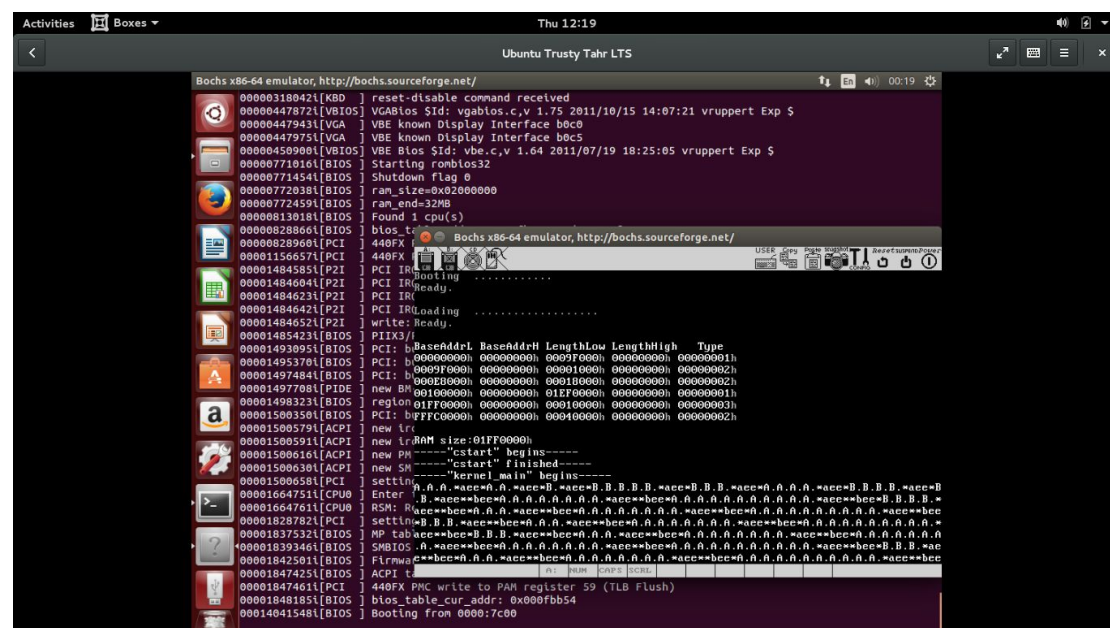
(5) 使用 rm 对特殊文件进行删除，触发文件保护机制

```
#####  
#:rm /dev_tty0  
cannot remove file /dev_tty0, because it is not a regular file.  
Failed to remove file: /dev_tty0  
#####
```

(1) 使用两级优先级反馈队列进行用户进程的调度



(2) 使用三级优先级反馈队列进行用户进程的调度



3. 设计理论

3.1 界面实现

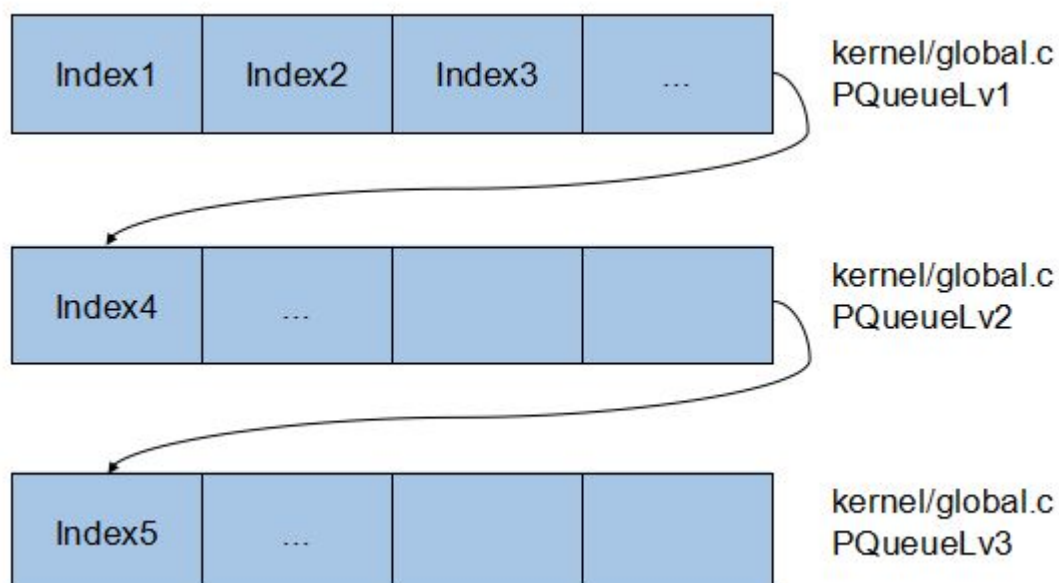
界面保留了源代码的大部分界面，采用的是 `boot/boot.asm` 中定义的 BIOS 基础文字布局：

```
mov ax, 0600h
mov bx, 0700h
mov cx, 0
mov dx, 0184fh
int 10h
```

黑白模式，屏幕字符分辨率为 `80*50`，代码的改进部分包括字符图案输出同样是包装 BIOS 10h 中断指令提供的输出指令，其显存根据 `loader.asm` 中的定义基址为 `0b8000h`，并在此基础上通过计算偏移来指定屏幕准确位置上的字符值，字符集为 `ascii`。

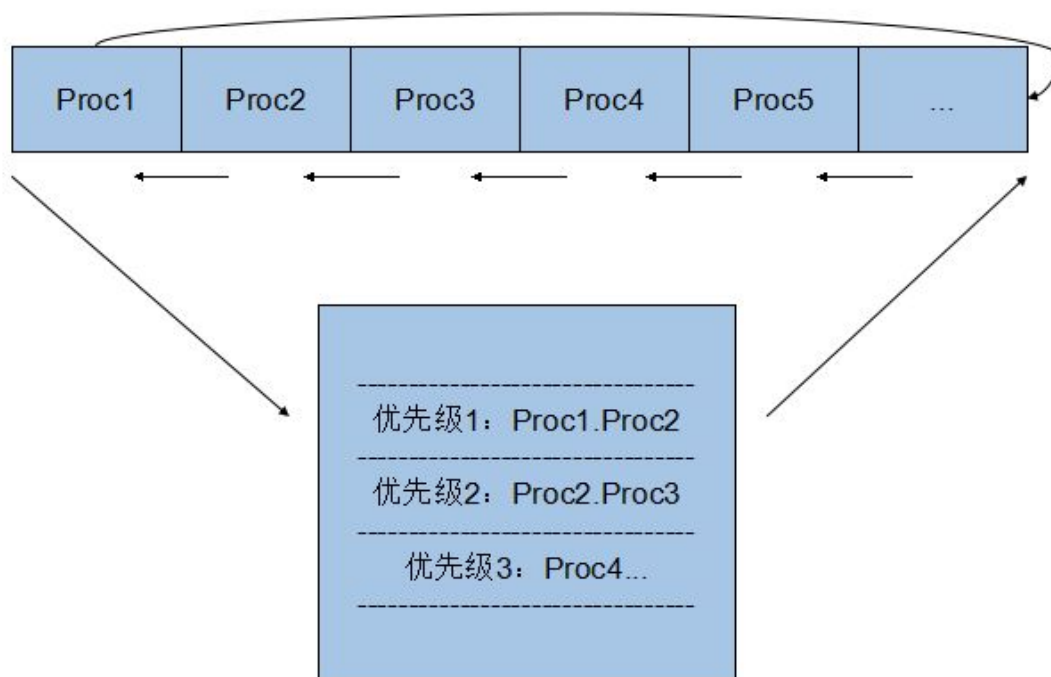
3.2 数据结构及调度算法

(1) 对于两级优先级反馈队列



在 `kernel/global.c` 同样有用来监视三个队列长度的全局变量 `Lv1Size, Lv2Size, Lv3Size`，通过长度变化来对队列进行不同范围的循环调整，在每个队列中储存的是 `proc_table` 中每个进程的索引值，通过计算下一步进程的索引值来给 `P_proc_ready` 赋值。实际上是给进程调度算法设计成了一个黑匣子，第三级队列采用的是纯粹的循环策略，即进程进入了第三级队列就不再降级，队列的等级越低分配到的时间片的长度就越长，等待的优先级就越低。

(2) 对于三级优先级反馈队列



和之前的两级优先级反馈队列不同之处在于，优先级队列并不显式创立，而是通过默认的规则对进程的优先级和计数进行计数。按顺序方法扫描进程表找到整个表中优先级最高的进程并且使其分配得时间片，但是马上调整到进程表的最后位置，使其最后一个被扫描到，从而实现优先级队列的方法，于前者相比节省了空间，但是牺牲了一定速度，因为要对整个进程表的 PCB 都进行调整。

4.功能实现

4.1

在对命令行系统的实现过程，我们实际上做的工作是在 `kernel/global.c` 中创建了一个永久的用户级进程，因为源代码中的内核是微内核的实现思想，所以我们的命令和内核交互的方法就是使用内核对外部提供的 `send_recv` 方法，这也是种异步方法。

我们的命令系统是在文件系统的支持之下创建起来的，操作系统和用户的交互是通过内核对 `/dev_ttyx` 设备文件的读写进行的，而对用户来讲是打开设备文件拿到的 0 号文件描述符 `std_in` 和界面进行交互，以及 1 号文件描述 `std_out`，我们对用户输入命令的反馈也是通过对 `std_out` 1 号文件描述符的读写操作来进行的。

主要代码如下所示的文件中：

```
Kernel/main.c
Kernel/tty.c
Kernel/global.c
Include/sys/fs.h
Lib/*.c
```

4.2

对进程的调度方法的改进对程序员和用户都是透明的，所以我们使用了 chapter6 中的源代码来对进程的调度方法进行调整，具体的调度策略已经在上文中说明。在演示中的 `-%d%d%d-` 分别代表了优先级队列中的进程个数。在所有的例子我们设立了 `NR_TASKS` 个进程。具体的定义在 `proc.h` 中。

主要代码如下所示的文件中：

```
Kernel/clock.c
Kernel/proc.c
Kernel/main.c
Kernel/global.c
```