

**操作系统课程设计**

2016.7.11~2016.7.15

学院： 软件学院

小组成员：陈冉冉 1452685

朱芳瑞 1452762

李威 1452714

目录

[一．设计主题 4](#_Toc461104520)

[1. 项目简介 4](#_Toc461104521)

[2. 参考源码 4](#_Toc461104522)

[3. 参考书目 4](#_Toc461104523)

[二．开发环境 4](#_Toc461104524)

[1. 运行环境： 4](#_Toc461104525)

[2. 开发语言 4](#_Toc461104526)

[3. 文件依赖 4](#_Toc461104527)

[三．系统设计 4](#_Toc461104528)

[1.系统启动 4](#_Toc461104529)

[（1）系统从磁盘中读出磁盘相关的信息。 4](#_Toc461104530)

[（2）系统将遍历所有根目录区的所有扇区，将每一个扇区都加载如内存，然后从中寻找文件名Loader.bin的文件，直到找到为止 5](#_Toc461104531)

[（3）引导扇区找到Loader后，就把控制权交给Loader。Loader的任务有两个，一个是加载内核到内存，即把kernel.bin找到，Loader每读取一个扇区，就会在Loading后增加一个. 6](#_Toc461104532)

[（4）Loader的第二个任务是跳入保护模式 6](#_Toc461104533)

[2. 进程调度 7](#_Toc461104534)

[3.文件系统 11](#_Toc461104535)

[4. Makefile 12](#_Toc461104536)

[四．用户应用 13](#_Toc461104537)

[1. 输入输出 13](#_Toc461104538)

[2. 创建文件 14](#_Toc461104539)

[3. 读取文件 14](#_Toc461104540)

[4. 写文件 15](#_Toc461104541)

[５. 删除文件 16](#_Toc461104542)

[6. 日期 16](#_Toc461104543)

[７．计时器 17](#_Toc461104544)

[五． 使用方法 18](#_Toc461104545)

# 一．设计主题

1. 项目简介： 我们的项目是一个微型的操作系统，从软盘启动，能进行进程调度，内存分配，应用 程序选择等功能。

2. 参考源码： 《Orange’s 一个操作系统的实现》随书附带源码

3. 参考书目： 《Orange’s 一个操作系统的实现》

# 二．开发环境

1. 运行环境：linux系统下Bochs虚拟机。

2. 开发语言：C语言，汇编语言。

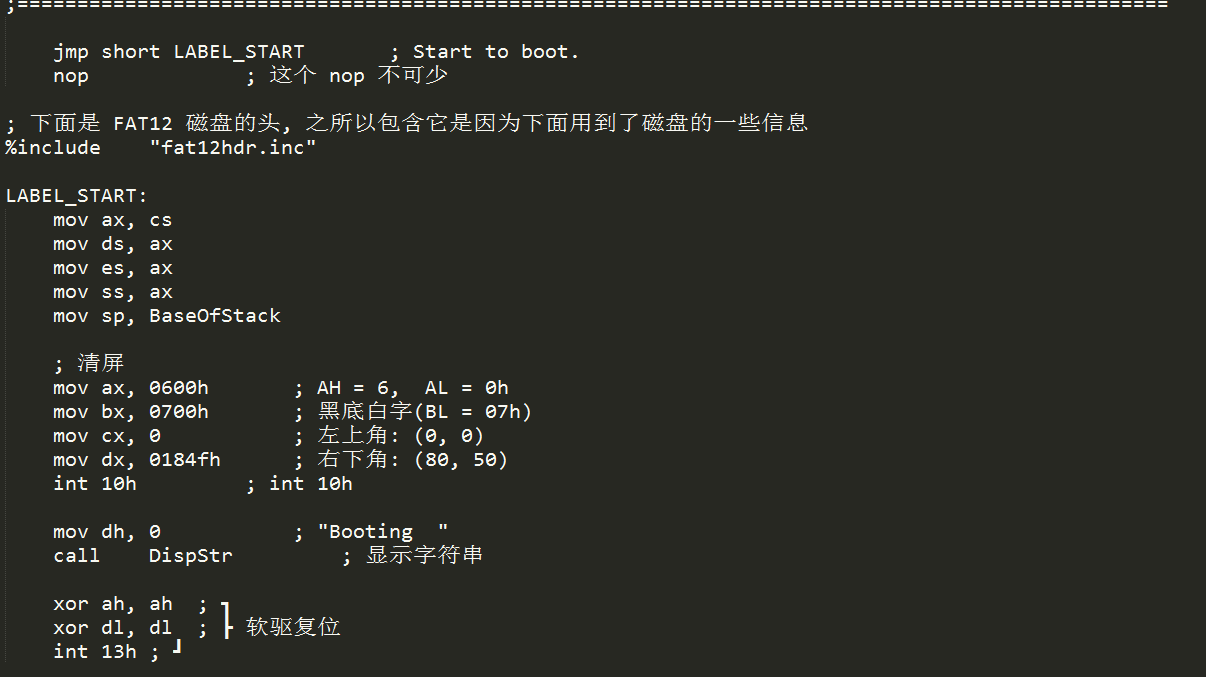
3. 文件依赖：OS文件夹下的各个文件。

# 三．系统设计

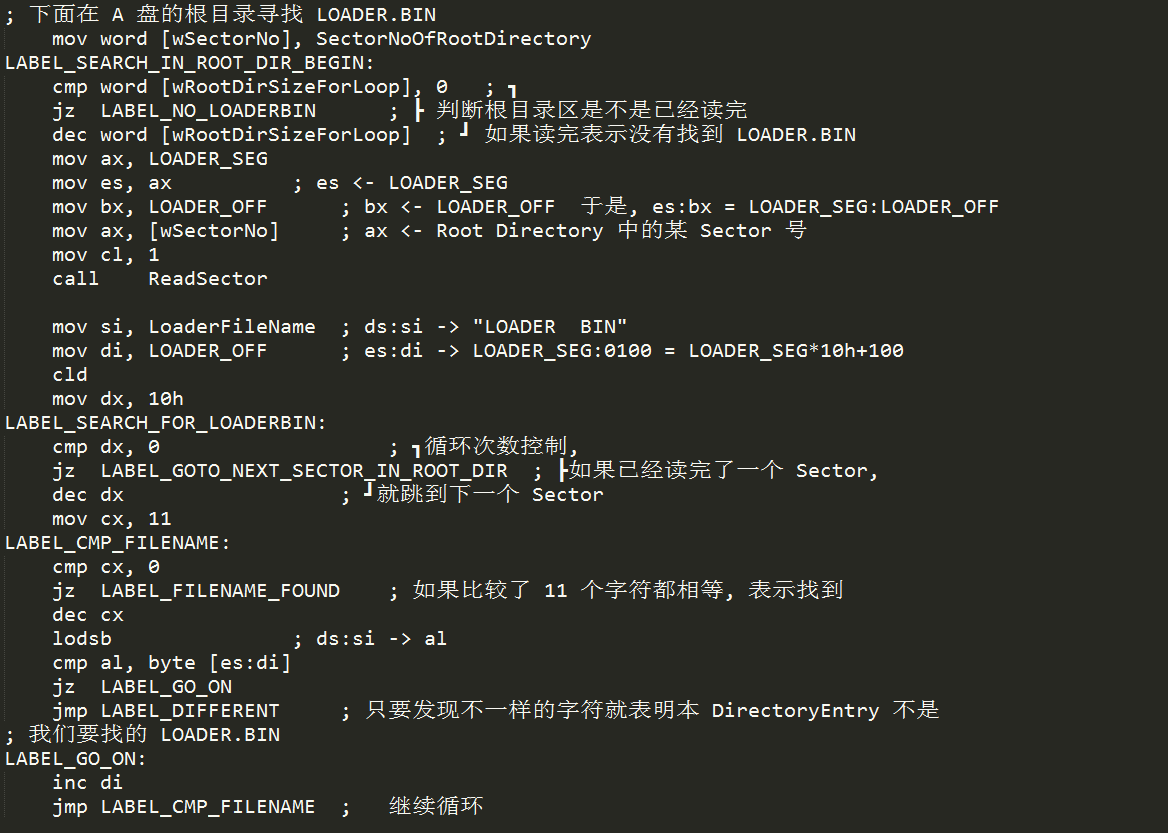
### **1.系统启动**

首先系统选择从软盘启动，进入软盘后开始进行初始设置，将软盘读到07c00h位置。

### （1）系统从磁盘中读出磁盘相关的信息。



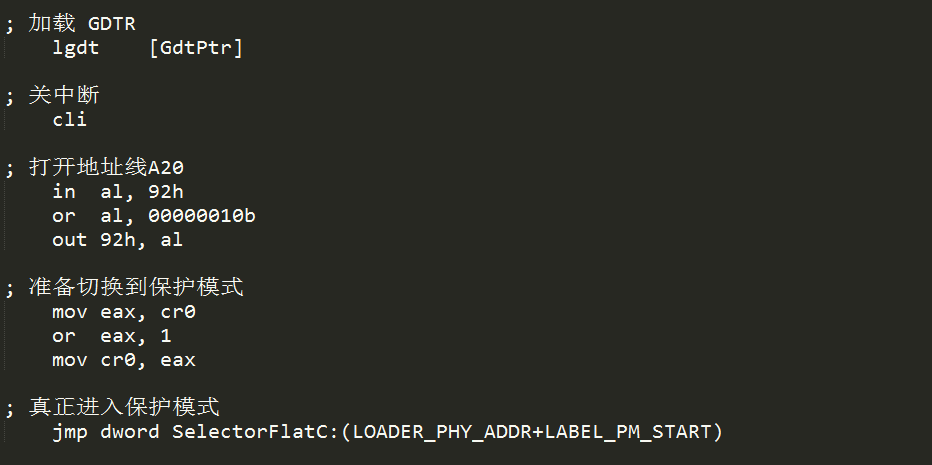
### （2）系统将遍历所有根目录区的所有扇区，将每一个扇区都加载如内存，然后从中寻找文件名Loader.bin的文件，直到找到为止



### （3）引导扇区找到Loader后，就把控制权交给Loader。Loader的任务有两个，一个是加载内核到内存，即把kernel.bin找到，Loader每读取一个扇区，就会在Loading后增加一个.

### （4）Loader的第二个任务是跳入保护模式

在实模式下，程序的寻址只能在1MB内，而这显然不能满足我们的需求，所有我们要跳入保护模式。在保护模式下，程序的寻址能力达到了4GB。



## 2. 进程调度

我们将中断、进程调度等放在系统ring0上，将终端等任务放在ring1上，讲用户进程放在ring3上。按照时间顺序，进程执行过程大致可以概括为：

1. 进程A执行。
2. 发生时钟中断，跳入ring0，时钟中断处理程序启动。
3. 进行进程调度，下一个就绪进程被指定（如B）。
4. 进程B被恢复，ring0跳出。
5. B运行中。

进程调度的方式主要分为两大类：抢占式和非抢占式。抢占式的调度就是调度程序一旦把处理机分配给某个进程后便一直让这个程序一直运行下去，直到这个进程运行完成或者发生某个事件而阻塞的时候，这个进程才会把处理机让出来供另一个进程使用。而抢占式的调度是指当一个进程正在运行时，系统可以根据某种原则，剥夺已分配给它的处理机，将处理机再分配给其他的进程。

在我们这个操作系统中采用的是抢占式的。是一个基于时间片原则的抢占式调度。主要是运用系统的定时器，给进程分配一定的时间片数，当进程运行的时间超过这个时间片的时候，就会把CPU的控制权让出来供其他进程使用。

当一个进程的时间片用完了之后，系统就要根据调度算法从就绪队列里边选出来一个进程并将处理机的控制权交给它。在本操作系统中采用的是“多级队列调度”的调度算法。

1. 基本思想：

·该调度算法将就绪队列分成2个队列，分别是任务队列和用户进程队列，对应的优先级是Priority(Q1)、Priority(Q2)，所有位于Q1的进程的优先级都比位于Q2的所有进程的优先级要高。

·而在某个特定的队列中，各个进程之间是采用时间片轮转法的。当该队列中一个进程时间片到了之后就把CPU的控制权交给该进程的下一个进程。

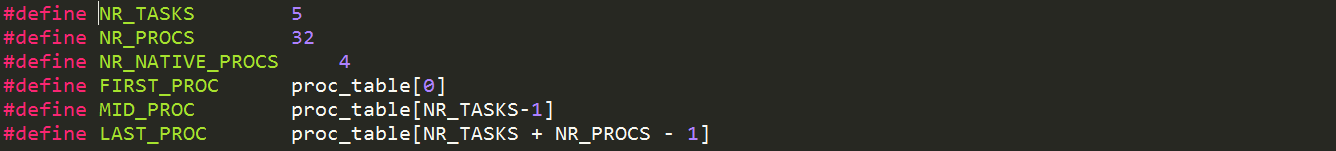
·此外，在多级队列调度算法中，每一个队列设定的时间片也是不一样的。一般各个队列的时间片是随着优先级的增加而减少的，也就是说优先级越高的队列中它的时间片是越短的。

当进程进行切换的时候，多级队列调度算法就会先从所有的队列中选出当前优先级最高的队列，然后在该队列中再采用时间片轮转的算法再选择一个进程。一般是当优先级更高的队列中所有的进程都执行完毕之后，才能轮到下一级的队列的进程开始运行。

（2）“多级队列调度”实现：

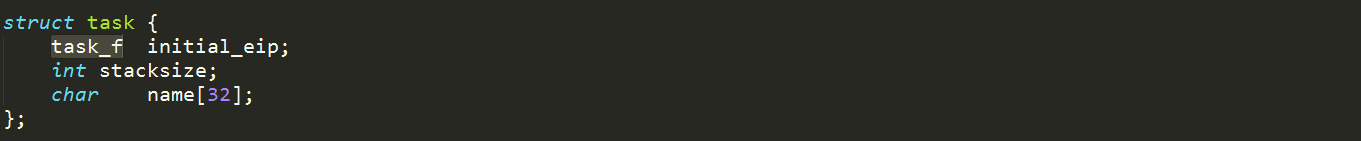
1）数据结构

1. 宏



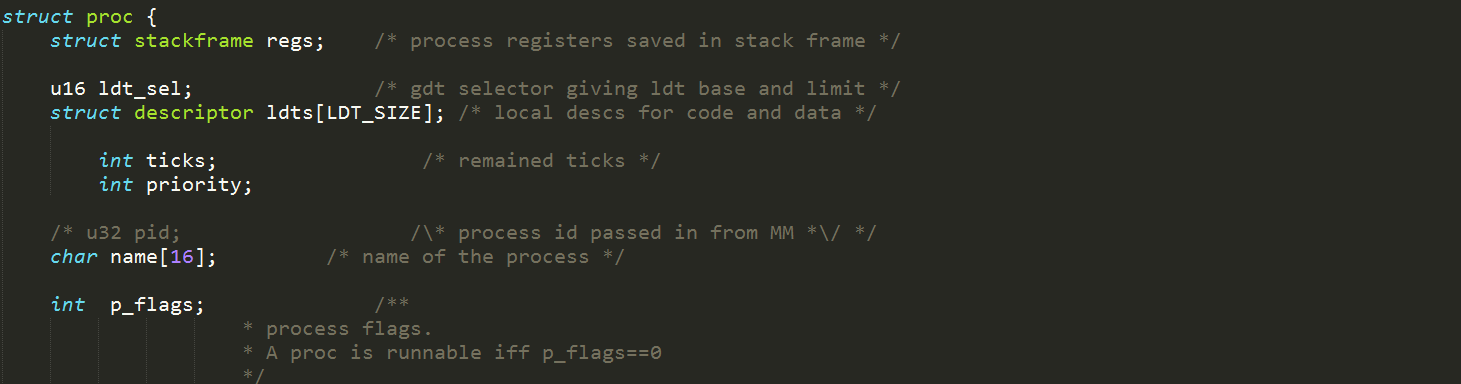
系统的任务有5个，用户进程最大为32个，系统进程是4个，proc\_table是由系统任务和用户进程组成的。

1. 结构体task



每个任务保存它的名字，栈空间的大小，以及入口地址等信息。

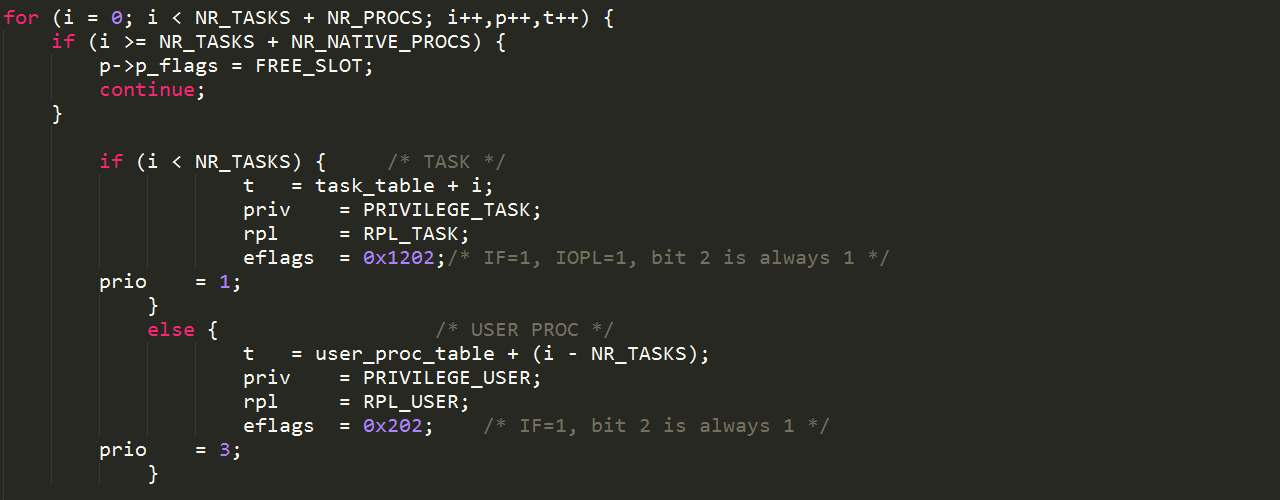
1. 结构体proc



这个结构体是用来存放进程信息，记录进程的选择子，描述符等信息。ticks的作用是记录进程的时间片数，priority记录进程的优先级。在本系统中，我们定义处于ring1的优先级为1，处于ring3的优先级为3，优先级越小，级别越高。

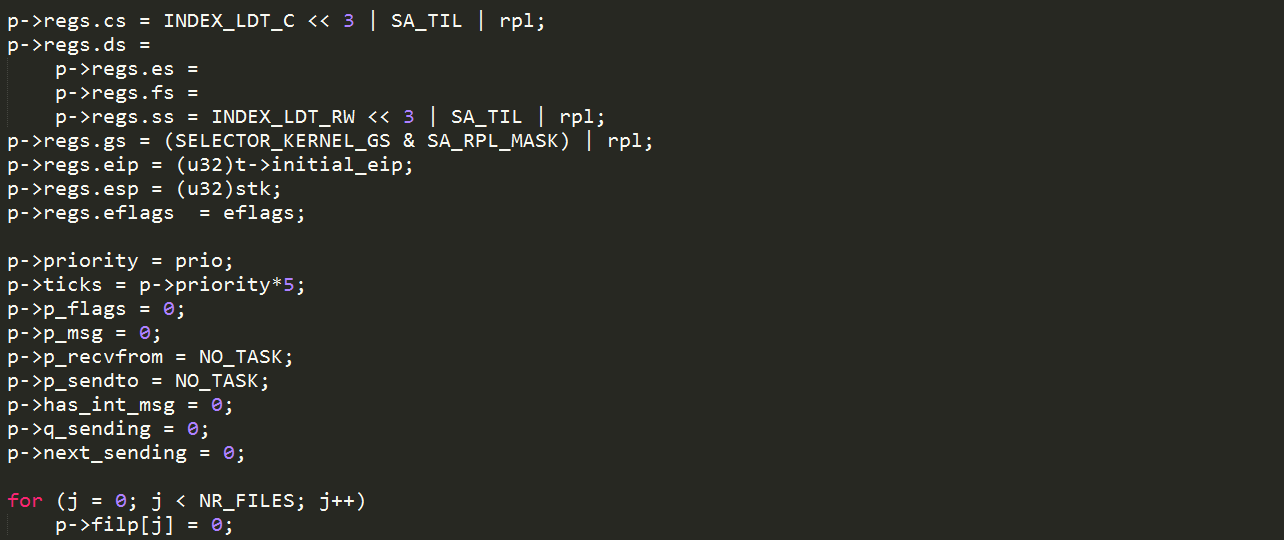
2）实现函数

①函数名：kernel\_main

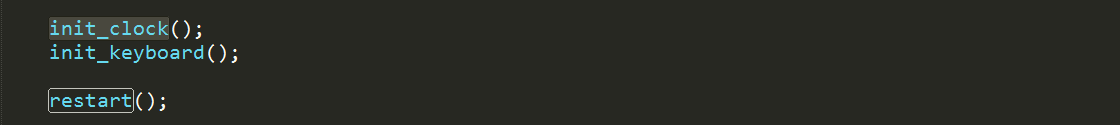


功能：初始化进程表，进程表中包含任务和用户进程，因此要分别初始化。

其中prio表示的是此任务或进程运行在系统的第几个环上。

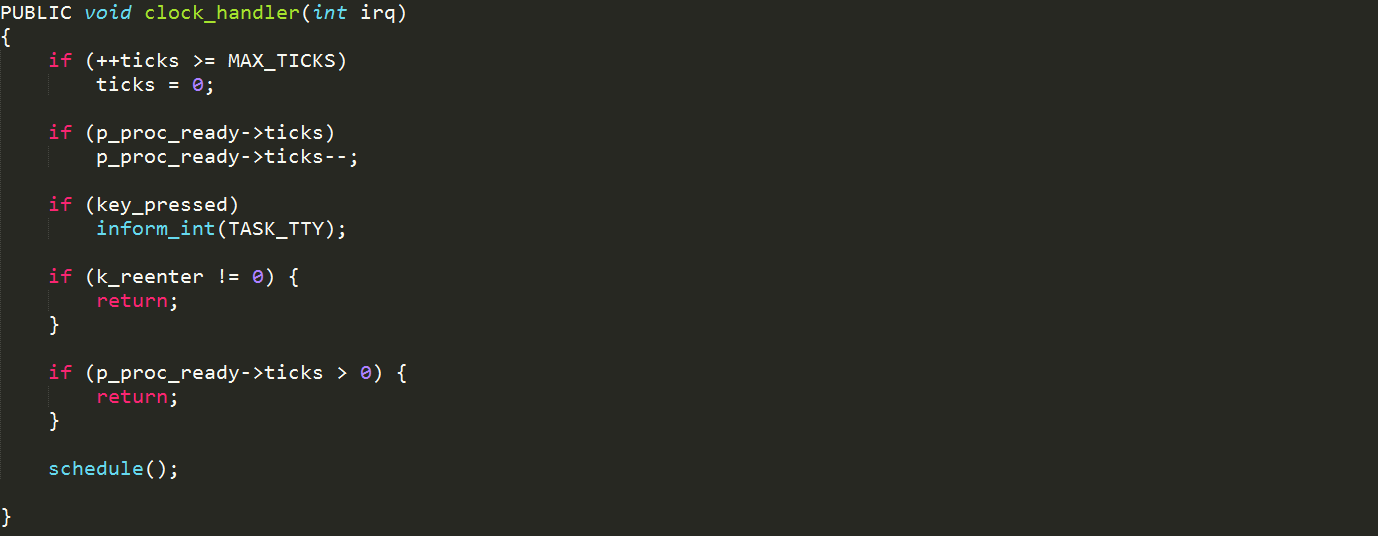


对进程或任务的寄存器、状态、时间片赋值。



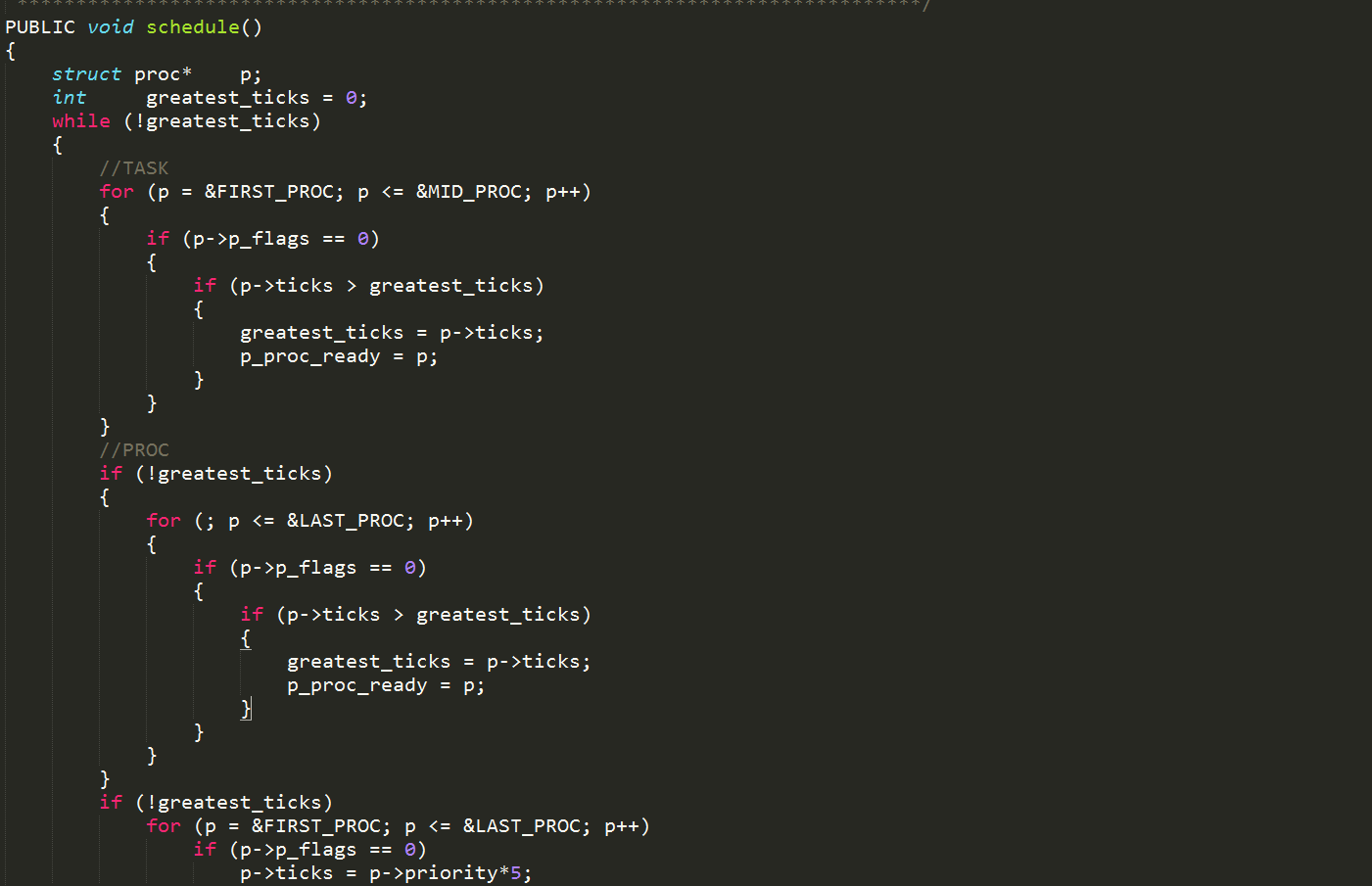
初始化时钟中断函数、键盘中断函数，restart函数的作用是选择第一个就绪函数，开始执行。

②函数名：clock\_handler



功能：此函数的主要作用发生时钟中断后，检查当前的任务或进程，将它的的时间片数减1，判断是否有键盘中断产生，如果有，则相应键盘中断，否则判断当前任务或进程的时间片数是否为0，如果不是，则继续执行，否则调用schdule选择下一个就绪进程。

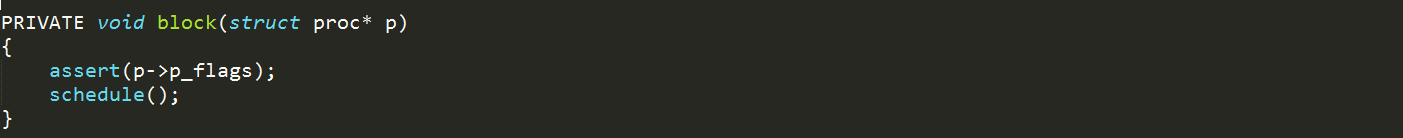
1. 函数名：schedule



功能：选择一个就绪的任务或进程。

指针遍历进程表，在[FIRST\_PROC，MID\_PROC]中是任务，在[MID\_PROC,LAST\_PROC]中是进程。其中flags为0是表明任务或进程处于就绪状态。指针如果在任务中发现还有未执行的任务，就返回。如果所有的任务的时间片数都为0，则在进程中选择。原理同任务。如果进程中也没有要执行的进程。会将进程表中所有的项按照各自的优先级重新赋时间片数。

1. 函数名：block



功能：使处于消息等待的进程进入等待状态

1. 函数名：unblock



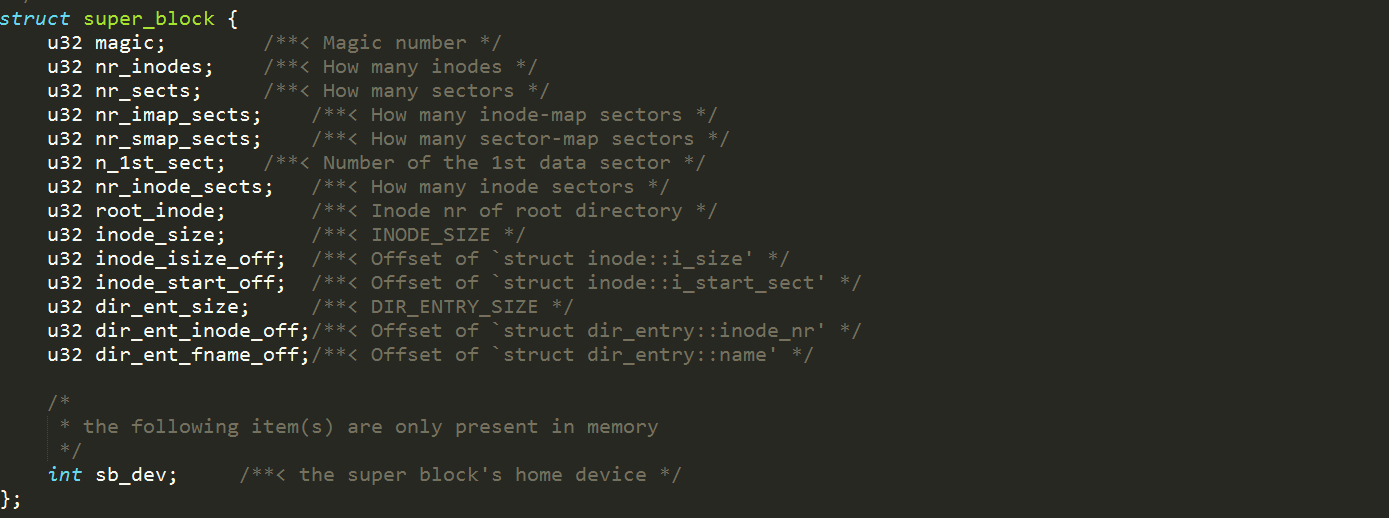
功能： 将等待的进程解除等待状态

## 3.文件系统

我们根据UNIX的文件原理，做了相应的简化设计。文件系统中有四个核心的数据结构

分别是：superblock、inode map、sector map、inode\_array。

1）superblock



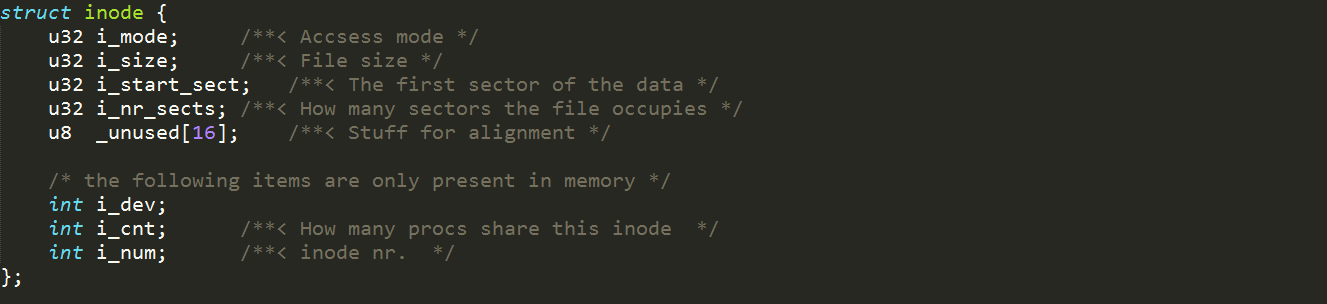
该结构体定义文件系统的标识、允许i-node的个数、inode\_array占用的扇区数、inode\_map占用的扇区数、sector\_map占用的扇区数，第一个数据扇区的扇区号等。

2）sector map和inode map

sector map表示扇区的使用情况。

inode map表示inode array的使用情况。

3）inode



Inode定义文件的起始扇区数，扇区大小，文件大小，文件的第一个扇区号。

## 4. Makefile

Makefile 可以理解为方便系统编译文件的工具，相当于批处理文件。首先生成一个不带扩展名的文件Makefile，在其中写入文件生成规则。有些场景下编译的程序是不能依赖OS和标准的C库的，并且需要C和汇编混合编译。

Make命令执行时，需要一个Makefile文件，以告诉make命令需要怎么样的去编译和链接程序。首先，我们用一个示例来说明Makefile的书写规则，在这个示例中，我们的工程有8个c文件，和3个头文件，我们要写一个Makefile来告诉Make命令如何编译和链接这几个文件。我们的规则是：

(1).如果这个工程没有编译过，那么我们的所有c文件都要编译并被链接。

(2).如果这个工程的某几个c文件被修改，那么我们只编译被修改的c文件，并链接目标程序。

(3).如果这个工程的头文件被改变了，那么我们需要编译引用了这几个头文件的c文件，并链接目标程序。

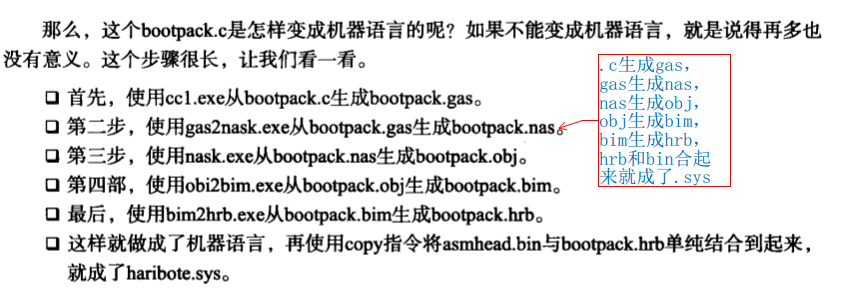
只要我们的Makefile写得够好，所有的这一切，我们只用一个Make命令就可以完成，Make命令会自动智能地根据当前的文件修改的情况来确定哪些文件需要重编译，从而自己编译所需要的文件和链接目标程序。

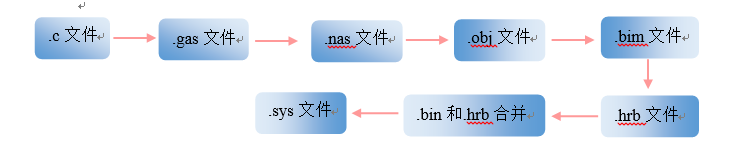
Make是如何工作的?在默认的方式下，也就是我们只输入Make命令。那么，Make会在当前目录下找名字叫“Makefile”或“makefile”的文件。如果找到，它会找文件中的第一个目标文件（target），在上面的例子中，他会找到“edit”这个文件，并把这个文件作为最终的目标文件。如果edit文件不存在，或是edit所依赖的后面的.o文件的文件修改时间要比edit这个文件新，那么，他就会执行后面所定义的命令来生成edit这个文件。如果edit所依赖的.o文件也不存在，那么Make会在当前文件中找目标为.o文件的依赖性，如果找到则再根据那一个规则生成.o文件。（这有点像一个堆栈的过程）当然，你的C文件和H文件是存在的啦，于是Make会生成.o文件，然后再用.o文件生成Make的终极任务，也就是执行文件edit了。这就是整个Make的依赖性，Make会一层又一层地去找文件的依赖关系，直到最终编译出第一个目标文件。在找寻的过程中，如果出现错误，比如最后被依赖的文件找不到，那么Make就会直接退出，并报错，而对于所定义的命令的错误，或是编译不成功，Make根本不理。Make只管文件的依赖性，即，如果在我找了依赖关系之后，冒号后面的文件还是不在，那么对不起，我就不工作啦。

通过上述分析，我们知道，像clean这种，没有被第一个目标文件直接或间接关联，那么它后面所定义的命令将不会被自动执行，不过，我们可以显示要Make执行。即命令——Make clean，以此来清除所有的目标文件，以便重编译。

于是在我们编程中，如果这个工程已被编译过了，当我们修改了其中一个源文件，比如file.c，那么根据我们的依赖性，我们的目标file.o会被重编译（也就是在这个依性关系后面所定义的命令），于是file.o的文件也是最新的啦，于是file.o的文件修改时间要比edit要新，所以edit也会被重新链接了（详见edit目标文件后定义的命令）。而如果我们改变了command.h，那么，kdb.o、command.o和files.o都会被重编译，并且，edit会被重链接。

讲完makefile，我们再来说一下如何将c语言导入：



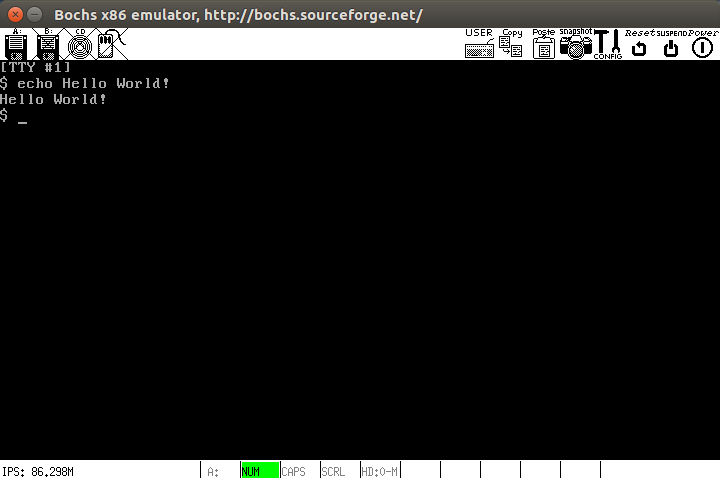


# 四．用户应用

为了使操作系统更好的和用户交互，我们丰富了终端操作，在shell中添加了实用且有趣的应用。

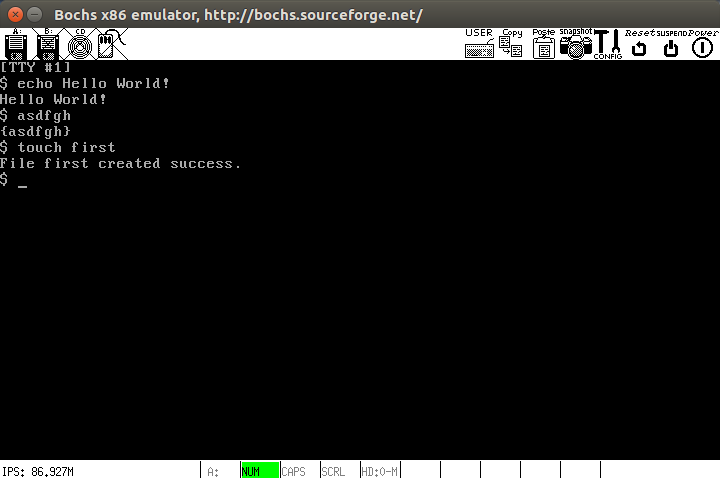
## 输入输出

echo命令的作用将后面的字符串全部显示



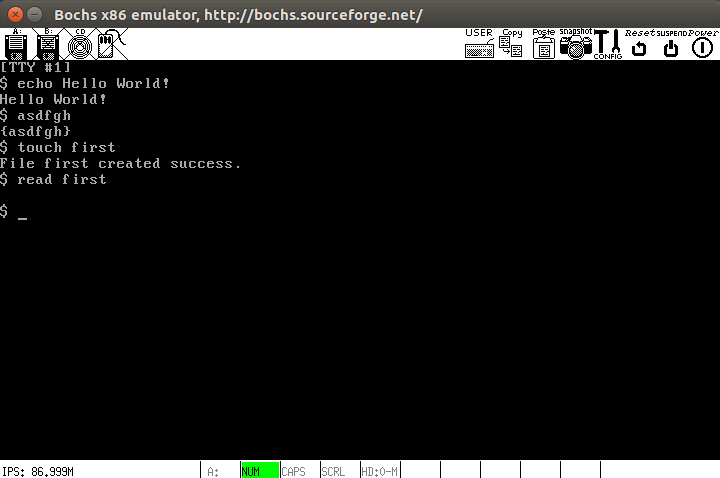
## 2. 创建文件

touch + 文件名 ：创建新的文件



## 3. 读取文件

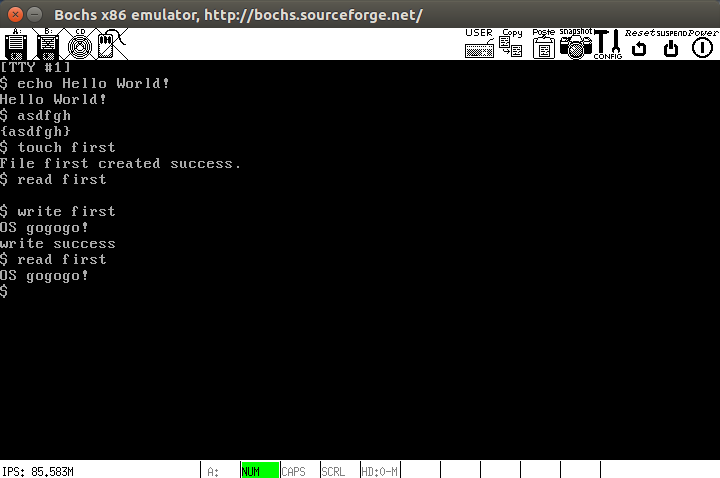
read+文件名

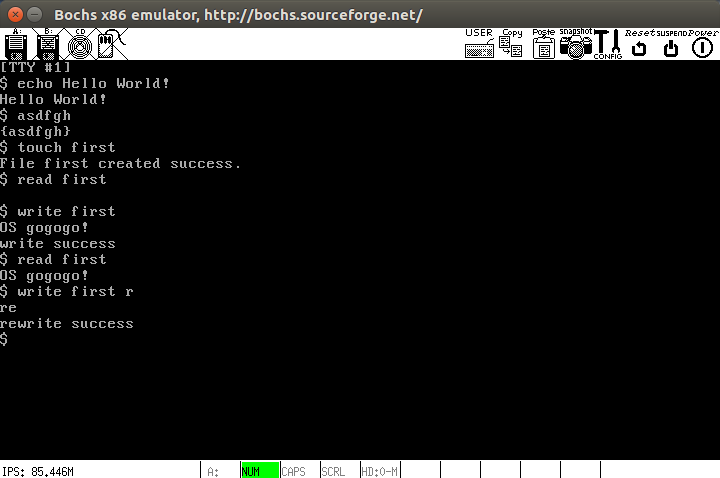


## 4. 写文件

　　　　write +文件名

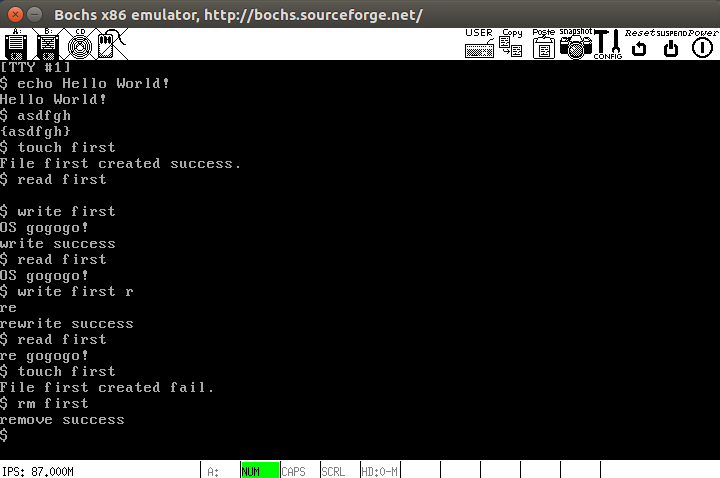
文件名后可跟随参数，ｒ表示从文件头开始写，默认是追加写。





## ５. 删除文件

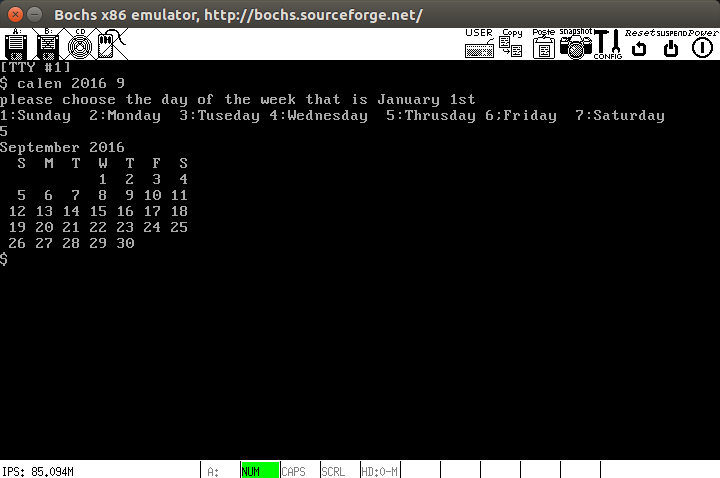
touch+文件名



## 6.日期

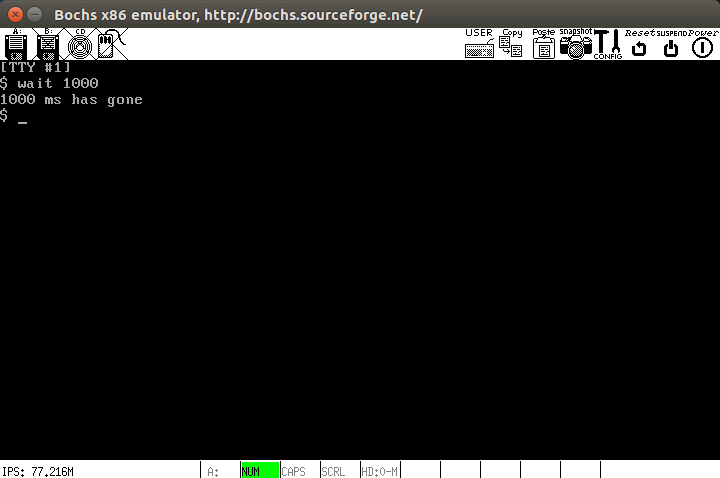
calen + 年份 +　月份

选择本年第一天是星期几



## ７．计时器

wait + 毫秒数



# 五． 使用方法

在os文件下打开终端，输入“bochs”，再输入“回车”即可看到操作系统的界面。

按住“shift+ctrl+F2”进入操作系统终端，可以输入各种终端命令。