**操作系统大作业**

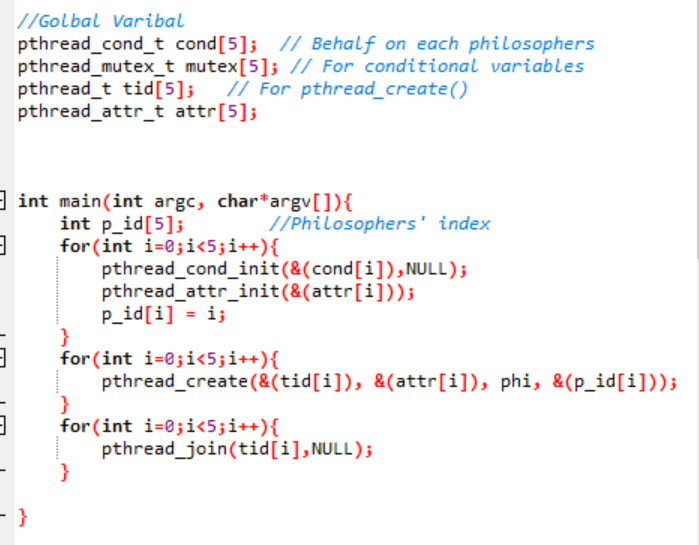
**17363104 朱韬正**

**1哲学家问题**

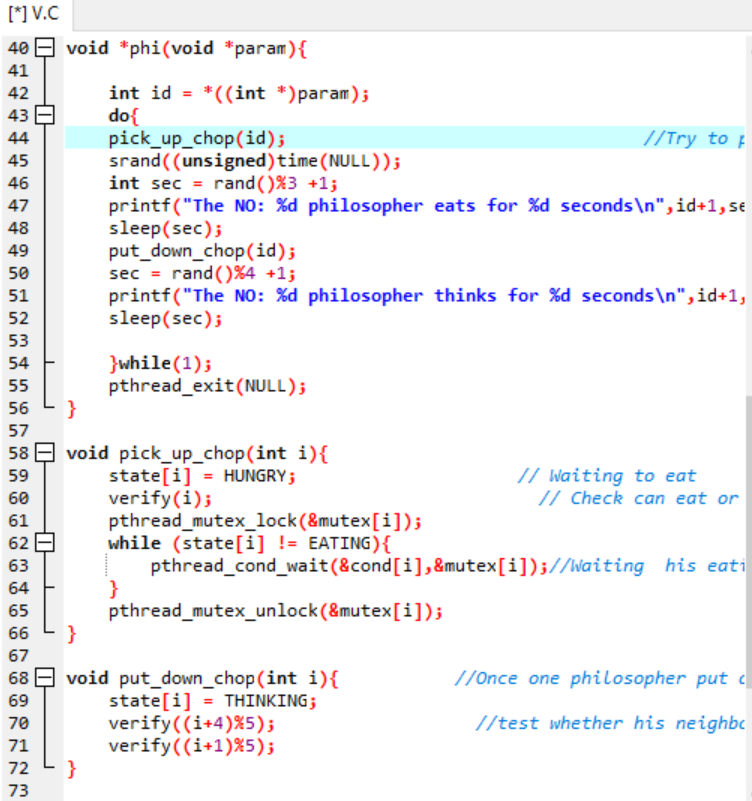
**a设置三个可能的状态：**

**1573111946(1)**

**b之后初始化五个线程代表五个哲学家，初始化五个互斥锁，五个条件变量。**

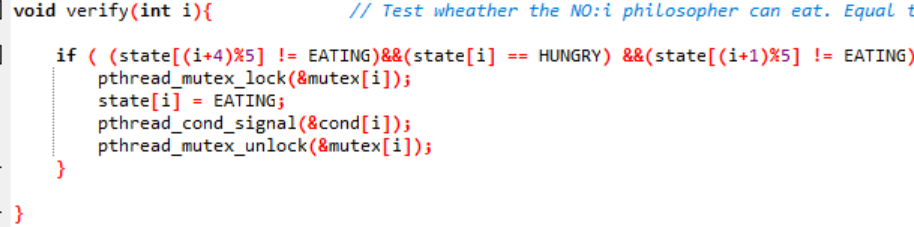
****

**c接下来完成哲学家函数与其动作：**

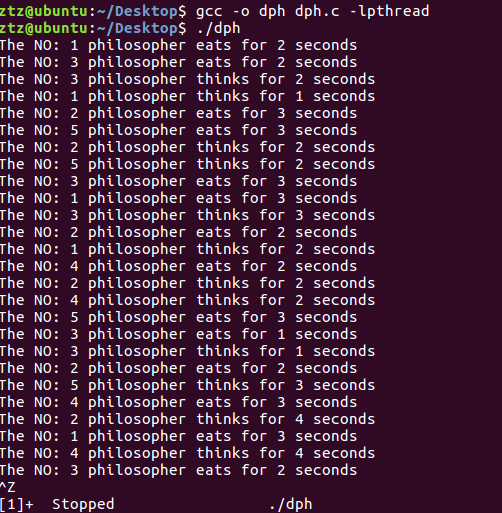
****

**注意到，每次放下筷子，都是资源的释放，验证周围有没有阻塞的哲学家。**

**d验证函数负责说明能不能吃，并且释放互斥锁，使阻塞的哲学家开始吃饭。**

****

**e最后试运行的结果，模拟了五个哲学家。**

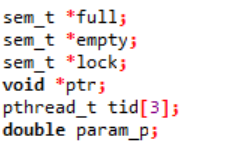
****

1. **生产者与消费者**

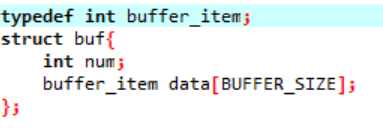
**生产者：**

**a 申明全局变量 ，以便线程操作**

**因为哲学家问题用过了mutex,这次以信号量lock为锁作为练习**

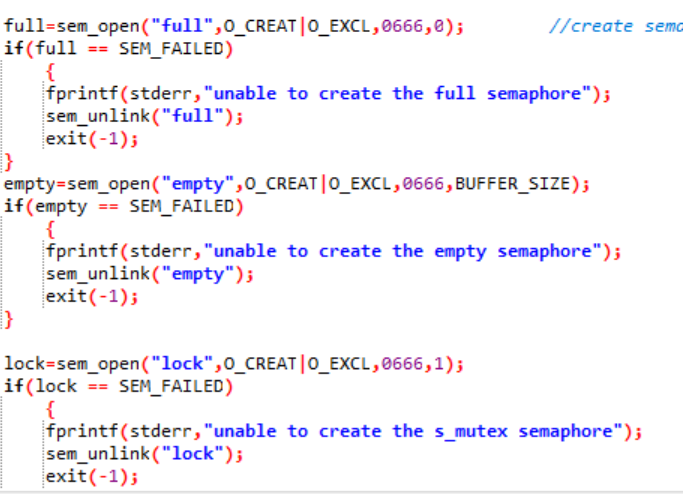
****

**b根据题意，定义buffer:带cursor的数组**

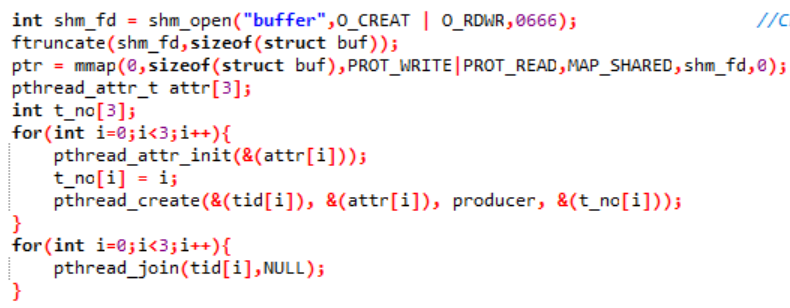
****

**c初始化具名信号量：**

**增加错误的提示**

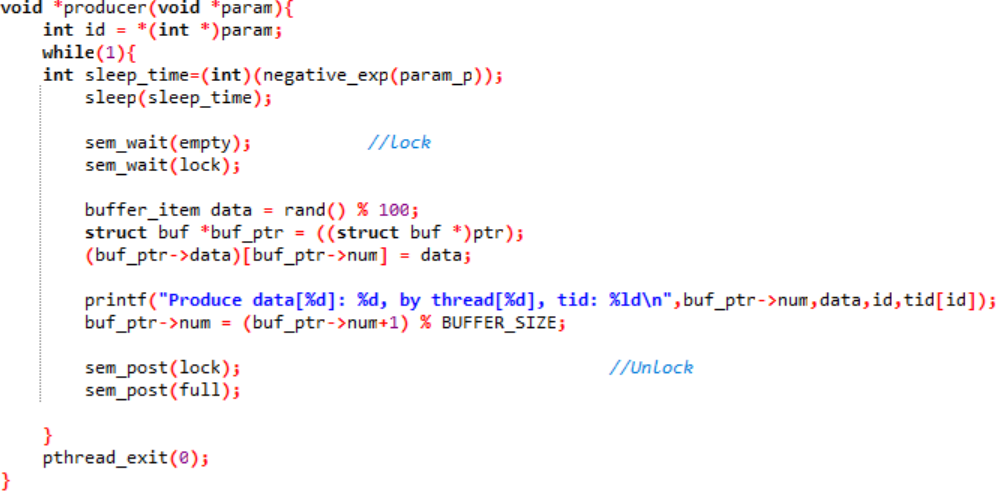
****

**d 分配共享内存，创建并加入线程**

****

**f生产者函数：**

**使empty信号量减一，full信号量加一，防止cons越界**

****

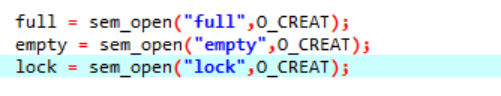
**e负指数的时间（double）的产生函数，经多次实验，输入参数param\_p 与param\_c 在9-21都可以。**

****

**消费者：**

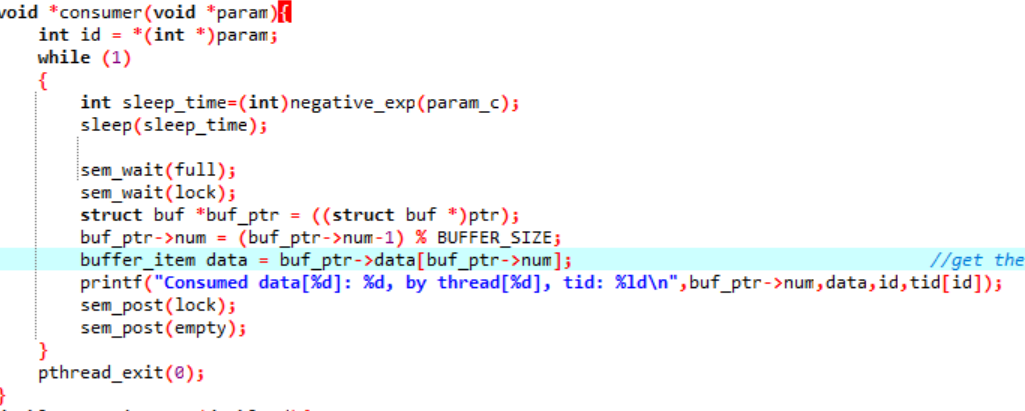
**a之前部分与生产者大致相同，略去**

**b信号量在另一个进程的打开：**

****

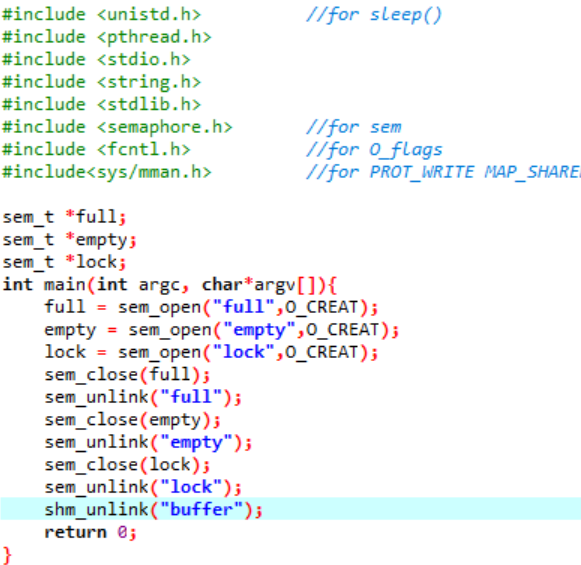
**返回已经打开信号量的指针。**

**c消费者函数：**

****

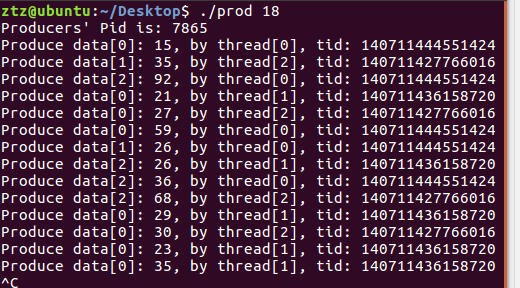
**其他：**

**因为在终端中ctrl+C中止程序不会自动sem\_unlink等,所以添加clr程序完成善后工作。（具名信号量会一直存在,导致下次打开程序出现问题（core dump））**

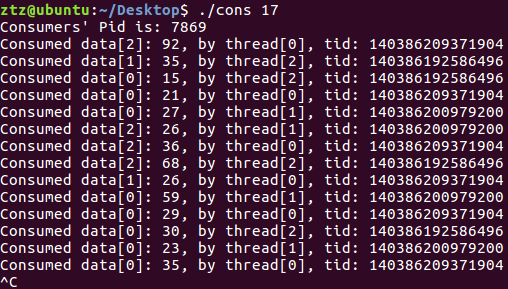
****

**结果示例：**

**生产者：**

****

**消费者：**

****

**3内核编译**

**CFS**

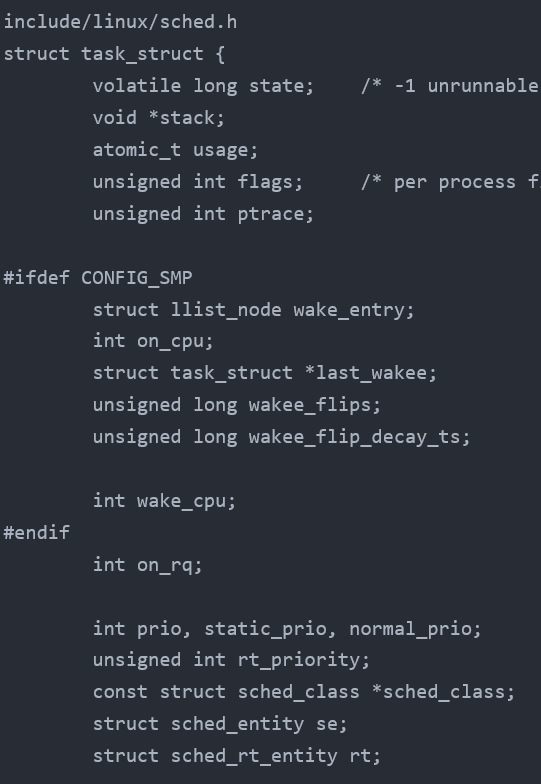
**Fair.c重点摘要**

1. **task\_struct进程描述符**

**state表示运行状态**

**Flags 退出等状态的标志**

**其中包含了sched\_entity se**

****

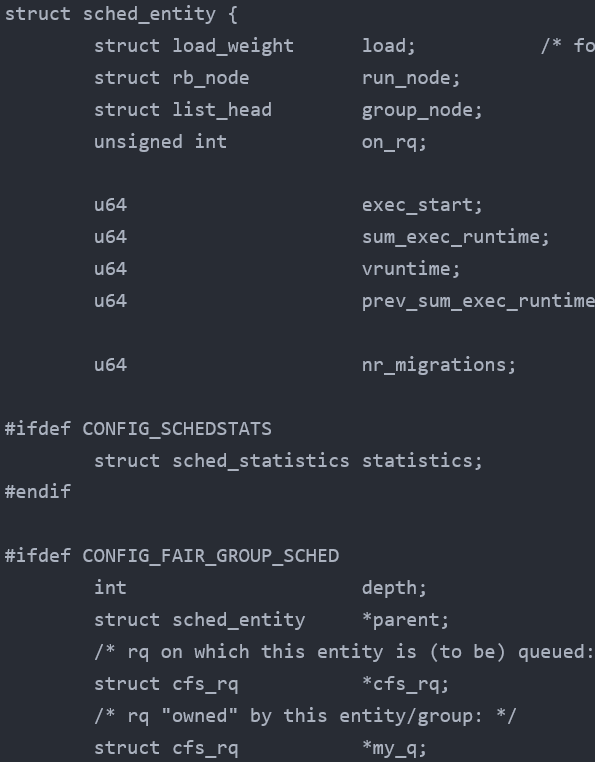
1. **sched\_entity调度实体**

**Load即为权重**

**Rb\_node即红黑树节点，红黑树储存各进程。**

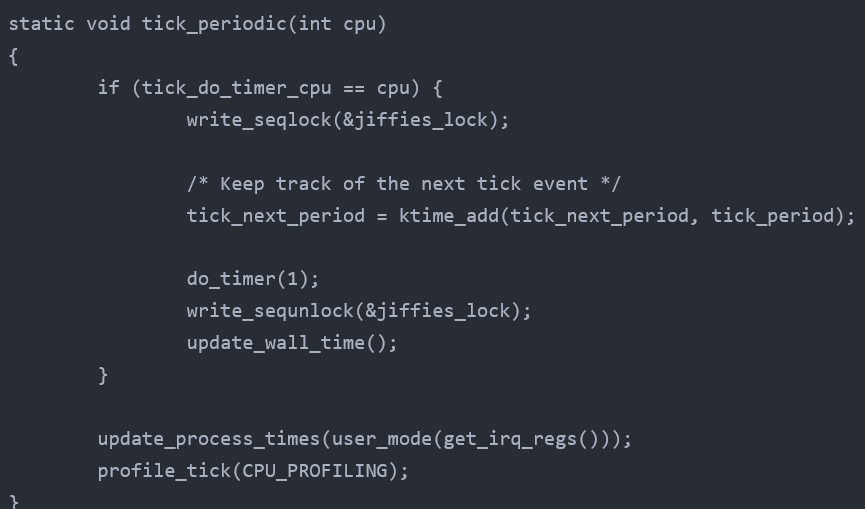
**Vruntime 为u64 类型，表示虚拟调度时间**

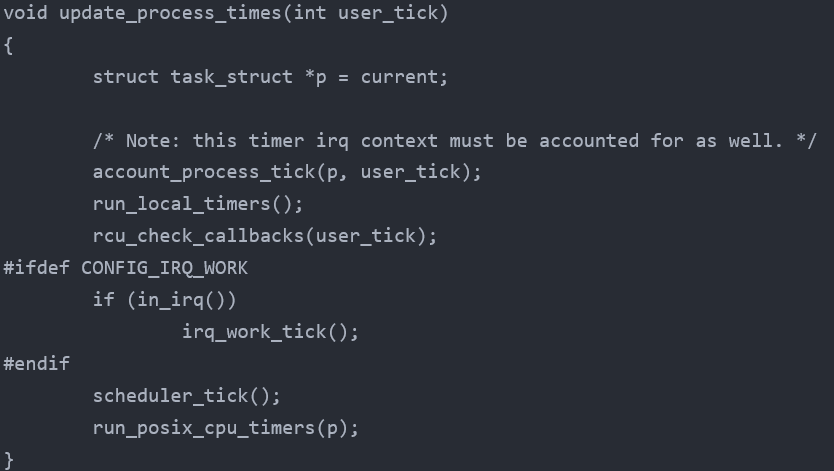
**Struct cfs\_rq \*my\_q为调度队列**

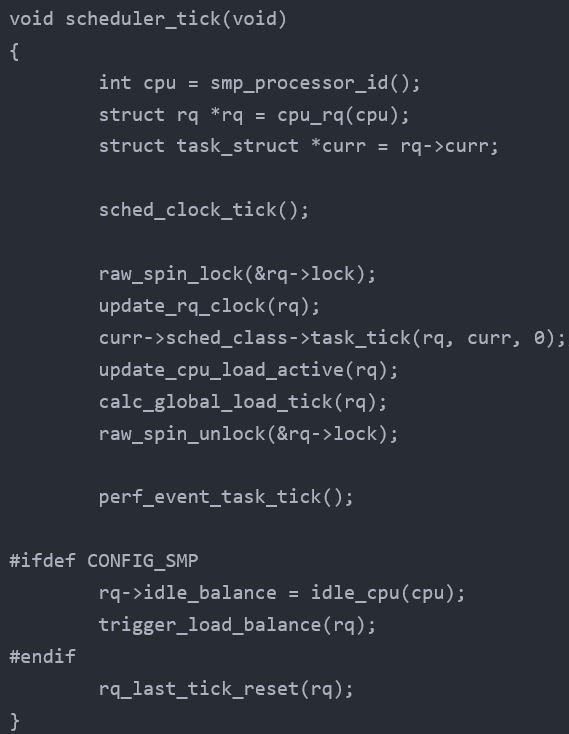
****

1. **vruntime更新函数**

tick\_periodic()->update\_process\_times()->scheduler\_tick()

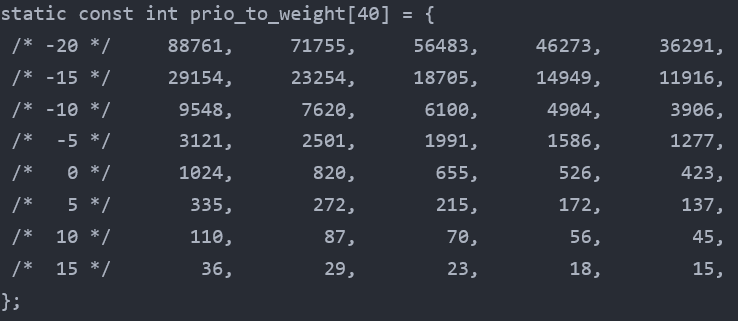
****

****

****

1. **数组**prio\_to\_weight

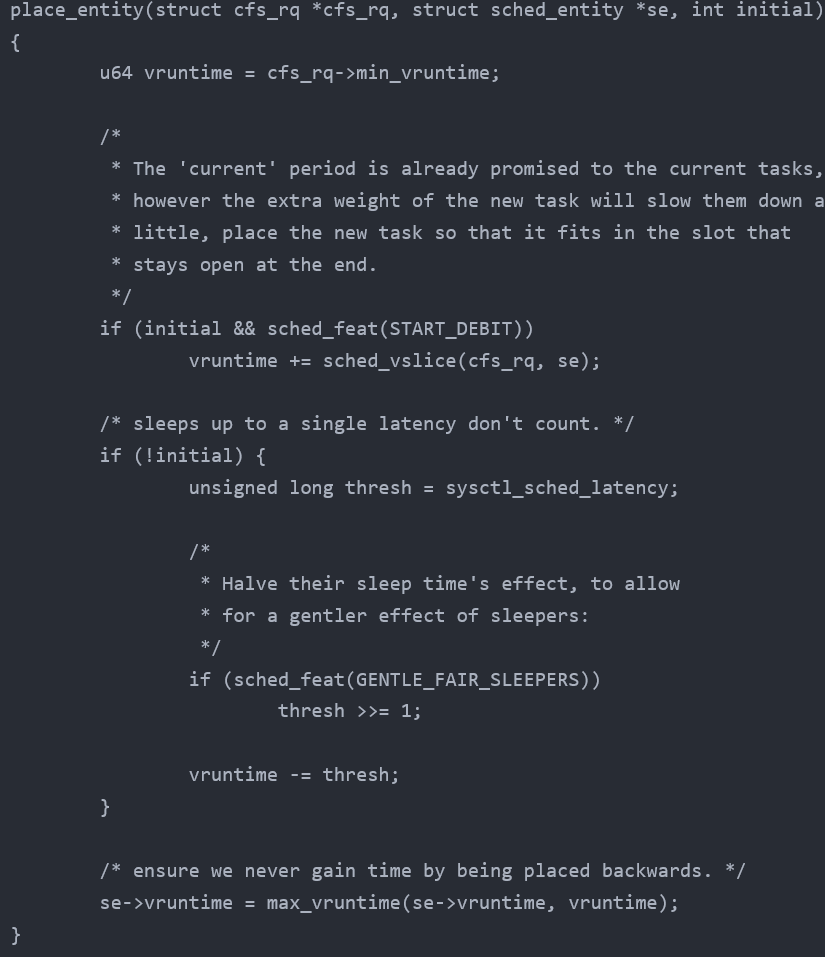
可以看出每一级cpu占用时间差百分之十左右。



1. 休眠进程的唤醒与新进程的加入

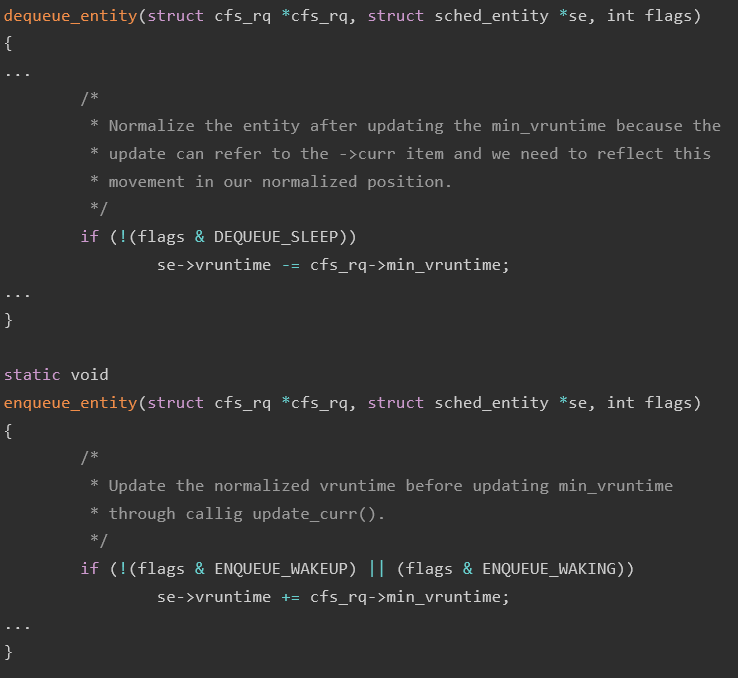
START\_DEBIT决定了新进程相对min\_vruntime的增加程度

重新设置vruntime,防止太占优势



1. 进程在cpu之间的迁移

减去原来的min\_vruntime,加上新cpu的min\_vruntime,以保证公平



Questions

1. 进程优先级、nice值和权重之间的关系?

nice意为友好，nice值（NI）代表对其他进程的友好程度，nice值越高，优先级（PTR）越低，权重越小。（可以通过全局数组prio\_to\_weight来转换）

1. CFS调度器中的vruntime的基本思想是什么？是如何计算的？何时得到更新？其中的min\_vruntime有什么作用？

（1）vruntime即virtual runtime,是进程的虚拟时钟，进程执行时，其vruntime增长，而优先级越高的进程vruntime增长得越慢。因为运行时间短的进程会被优先执行，故通过vruntime可使优先级高的进程先运行，而不至于产生饥饿。

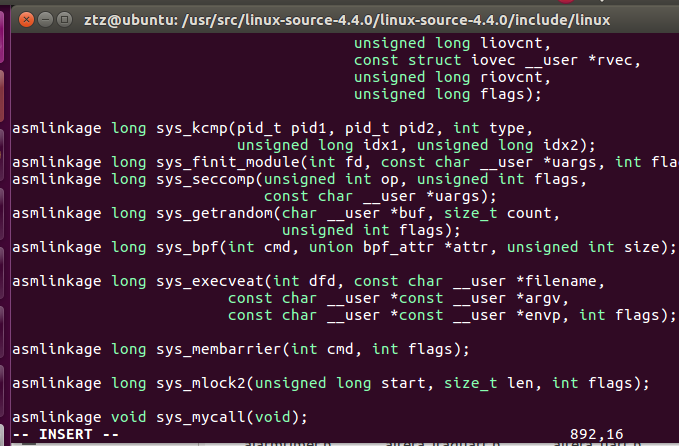
（2）Vruntime=(调度周期\*进程权重/所有系统总权重)

（3）运行后的进程vruntime会增加，休眠进程被唤醒，更换到另一个CPU和vruntime溢出超过unsigned long上限时vruntime也需要更新。

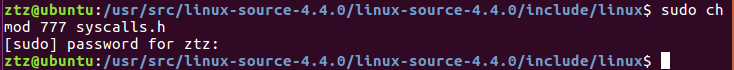
（4）min\_runtime是CFS红黑树中最小的虚拟时间。用于防止溢出和迅速选中Vruntime最小的进程。也用于给新进入的进程赋值。

M有call

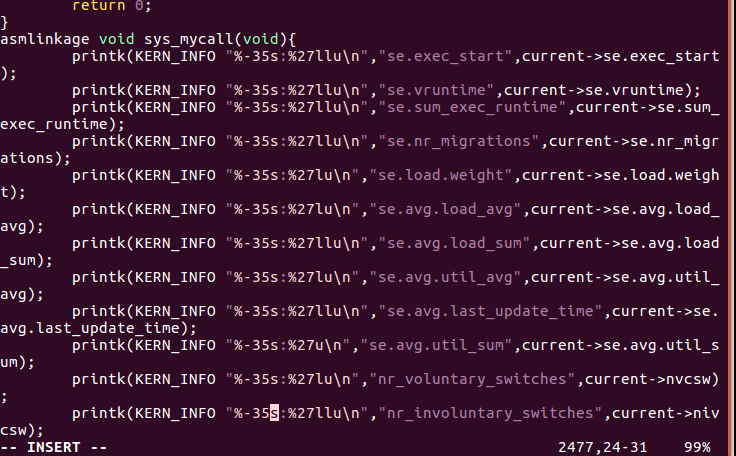
1修改syscalls.h,声明mycall

****

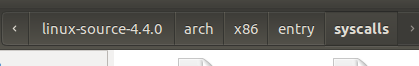
(都需要修改只读权限)



2在sys.c中添加具体定义：

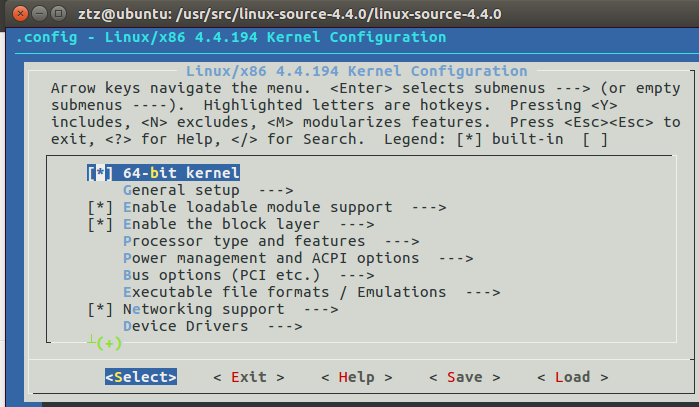


3 修改syscall\_64.tbl文件，增加调用mycall.

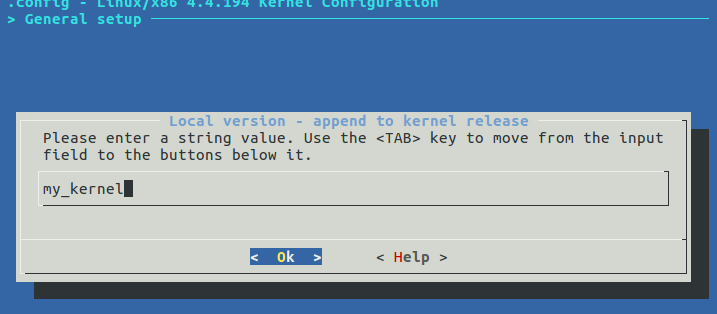




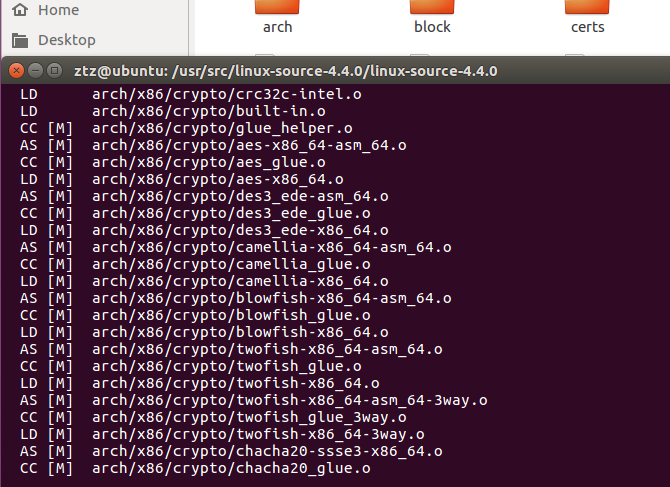
4 Make menuconfig:

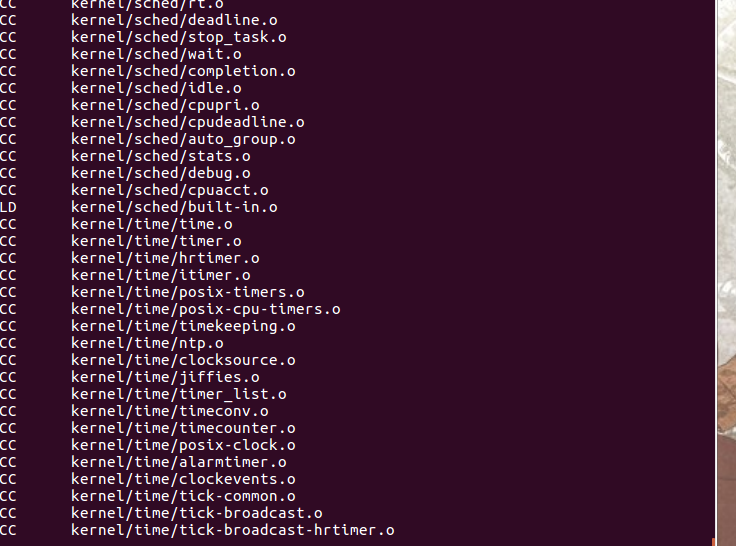


修改localversion名称：

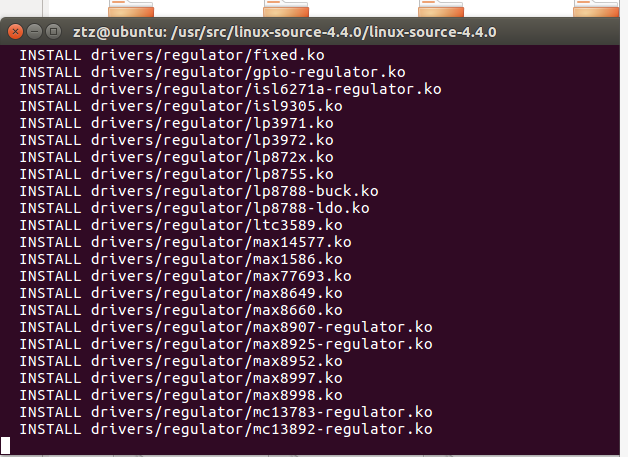


5 开始make:



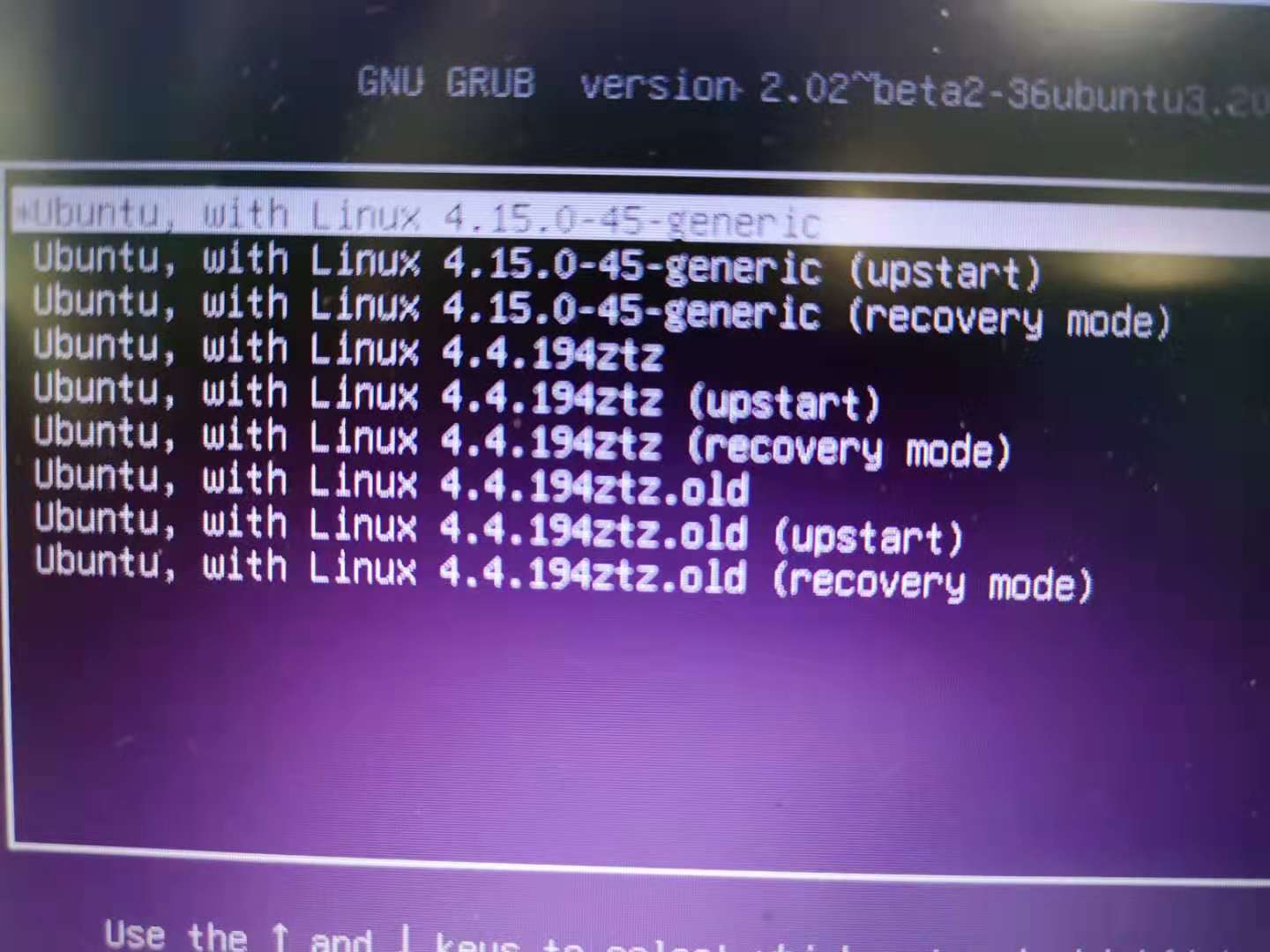


之后调用make modules\_install 安装模块，



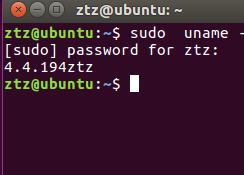
调用 make install 安装模块。

之后 gedit /etc/default/grub 修改grub文件使内核可以选择。

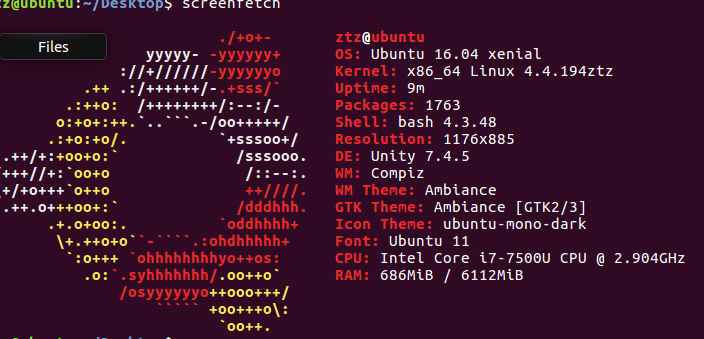


选择编译的4.4内核。

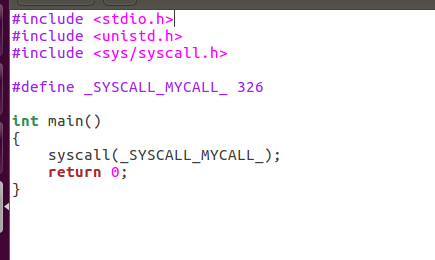
进去后 uname -r 确认无误。

、、

Screenfetch:



定义mycall.c,测试内核调用：



测试无误。

