Lab7实验报告

班级：计算机科学与技术2班

姓名：许金曼

学号：171491220

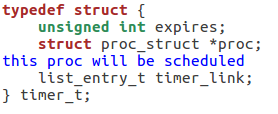
1.了解信号量和管程的实现机制

（1）同步互斥的底层支持是如何实现的？

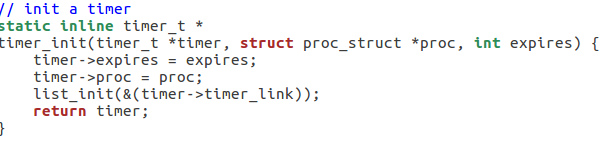
定时器：

定时器提供了基于时间事件的调度机制。基于此时间单位，操作系统得以向上提供基于时间点的事件，并实现基于时间长度的睡眠等待和唤醒机制。在每个时钟中断发生时，操作系统产生对应的时间事件。

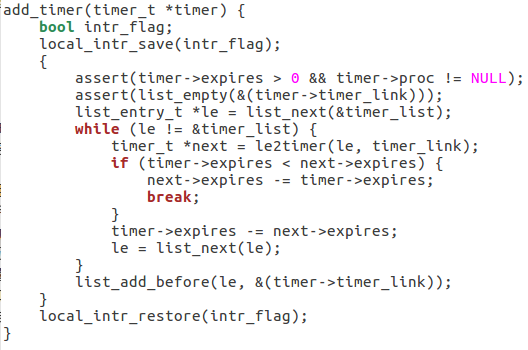
typedef struct{……}timer\_t定义一个定时器的基本结构



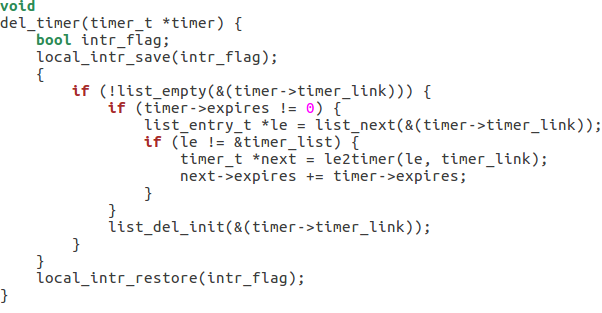
对定时器进行初始化，让它在expires时间片后唤醒proc进程



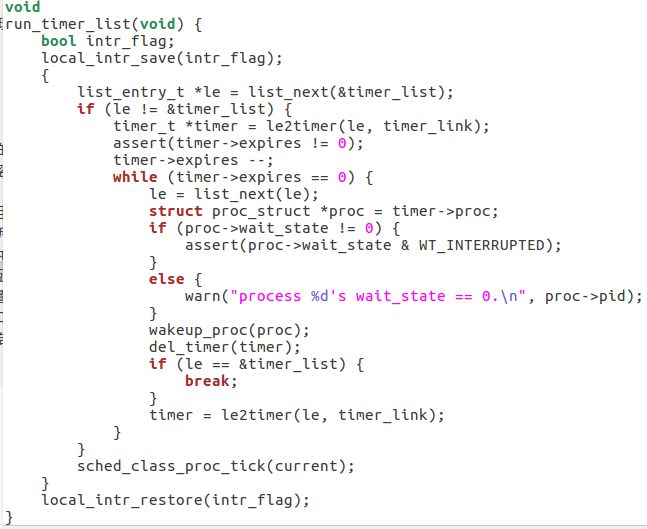
add\_timer(timer\_t \*timer)向系统添加某个初始化过的定时器，该定时器在指定时间后被激活，并将对应的进程唤醒至等待状态



del\_timer(timer\_t \*timer)在系统中取消一个定时器，取消后不会被系统激活并唤醒进程



run\_timer\_list更新当前系统时间点，遍历当前所有处在系统管理内的定时器，找出所有应该激活的计数器并激活它们。



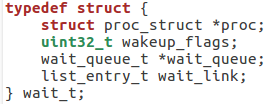
屏蔽/使能中断：

通过调用函数，最终调用机器指令cli和sti，实现了关（屏蔽）和开（使能）中断，即设置了eflags寄存器中与中断相关的位。通过关闭中断，可以防止对当前执行的控制流被其他中断事件处理打断，即实现了对临界区的互斥操作。

等待队列：

用户进程或内核线程可以转入等待状态以等待某个特定事件，需要等待事件的进程则插入到等待队列中。当事件发生后，内核遍历相应等待队列，唤醒休眠的用户进程或内核进程，并设置其状态为就绪状态，然后将该进程从等待队列中清除。

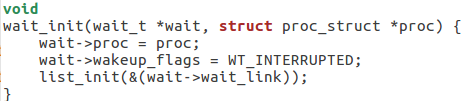
定义等待项的结构：



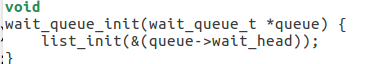
定义等待队列的结构：



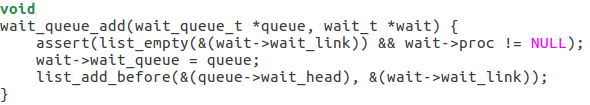
初始化wait结构：



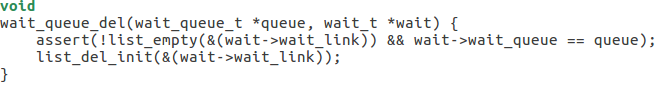
初始化wait\_queue结构：



把wait前插到wait queue中：



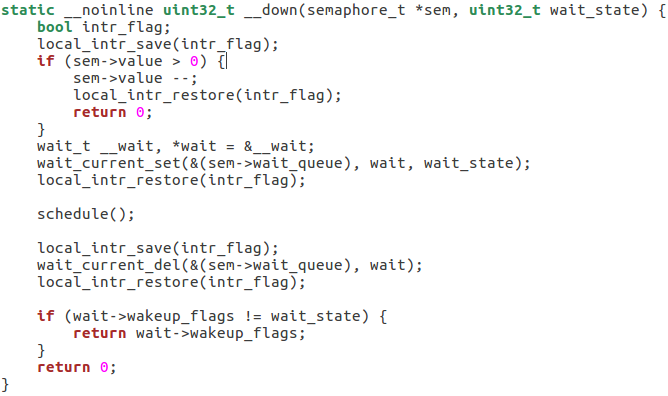
从wait queue中删除wait：



（2）对比原理课上学到的信号量和p，v操作，说明ucore中信号量机制的实现。

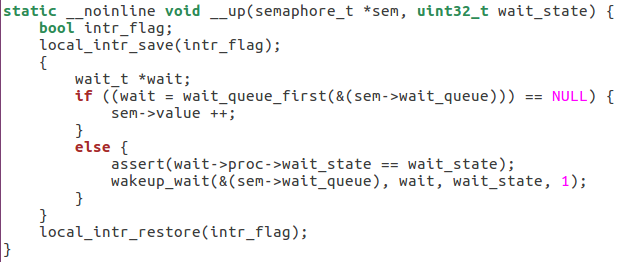
当多个进程可以进行互斥或同步合作时，一个进程会由于无法满足信号量设置的某条件而在某一位置停止，直到它接收到一个特定的信号。为了发信号，需要使用一个称作信号量的特殊变量。为通过信号量s传送信号，信号量的V操作采用进程可执行原语semSignal(s)；为通过信号量s接受信号，信号量的P操作采用进程可执行原语semWait（s）；如果相应的信号仍然没有发送，则进程被阻塞或睡眠，直到发送完为止。ucore中信号量建立在开关中断机制和wait\_queue的基础上进行实现。

实现P操作，请求一个信号量对应的资源：



查询value来判断是否存在多余的可分配资源，如果有，则取出资源，value-1；如果没有可用的资源，则将其状态改为SLEEPING状态，将其挂到对应信号量的等待队列中。如果资源得到满足，则被唤醒，将自身从等待队列中删除。

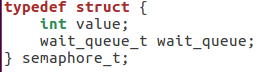
实现V操作，释放一个信号量资源：



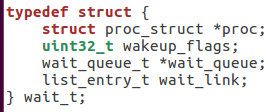
查询等待队列是否为空，空的话，value+1，如果等待队列不空，则取出一个进程唤醒。

（3）ucore中的信号量是基于信号量和条件变量实现的，请说明其中的数据结构和函数方法的设计

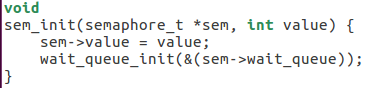
定义信号量的数据结构：



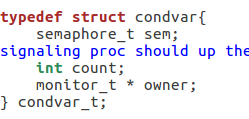
定义一个结构体，存放指向等待进程的指针，进程被放入等待队列的原因，等待队列和用于还原结构体的等待队列标志。



sem\_init函数对信号量进行初始化:

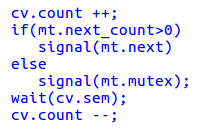


定义管程中的条件变量的数据结构：

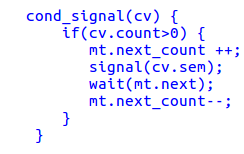


信号量sem用于让发出wait\_cv操作的等待某个条件Cond为真的进程睡眠，而让发出signal\_cv操作的进程通过这个sem来唤醒睡眠的进程。count表示等在这个条件变量上的睡眠进程的个数。owner表示此条件变量的宿主是哪个管程。

wait\_cv：



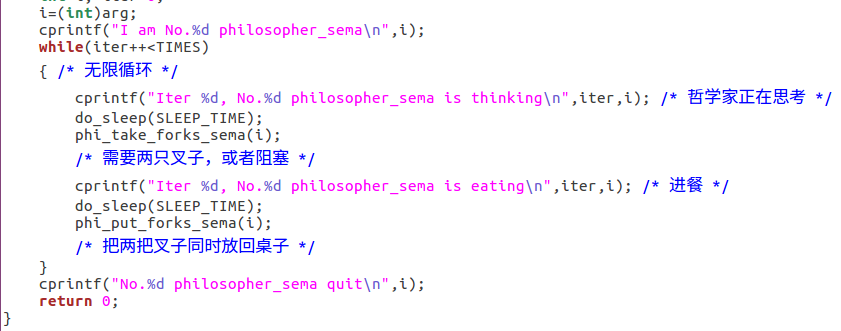
signal\_cv：



2.了解基于信号量和管程的哲学家就餐问题

（1）说明ucore中基于信号量的哲学家就餐问题的实现机制





共有N个哲学家，则设置一个有N个状态的数组，再设置一个互斥的信号量当某个哲学家同时P到两个信号量返回时可以用餐，否则阻塞等待；用餐结束后，同时V 两个信号量。

（2）说明ucore中基于管程的哲学家就餐问题的实现机制

对每个哲学家设置信号量，再设置一个函数，如果当前的哲学家处于饥饿状态并且两侧的哲学家不在吃饭状态，则当前哲学家进入吃饭状态；如果进入吃饭状态不成功，则该哲学家就在信号量阻塞等待。当一个哲学家进程调用放下筷子进程时，会通过函数测试它的邻居，如果邻居处于饥饿状态，且该邻居的邻居不在吃饭状态，则该邻居进入吃饭状态。