# Simple Kernel 模型分析

## ProcessQueue

该类用来描述进程队列。

### INIT

初始化函数保证队列初始是空的。

### IsEmpty

通过检查elts是否为空，判断队列是否为空。

### Enqueue

通过把指定元素放在elts末尾，将指定元素加入队列。

### RemoveFirst

取出队列头部第一个元素，并将其从队列中删除。

### QueueFront

返回队列头部第一个元素。

### RemoveElement

删除指定元素。

## HardwareRegisters

该类用来描述寄存器状态。

## Semaphore

该类描述了信号量，提供了wait和signal操作。成员变量包括 \* waiters：记录等待进程的队列 \* scnt, initval：分别记录信号量的初值和当前值 \* ptab：存储ProcessTable \* sched：存储调度器 \* ctxt：存储Context信息 \* lck：用中断实现的锁，负责实现同步互斥

### INIT

初始化成员变量。

### Wait

使用lck取得锁，然后将scnt的值减一。如果scnt的值小于零，则把当前进程加入等待队列，并将其状态设为waiting。 同时换出Context，并在调度器中设置当前进程不可执行。最后通知调度器执行下一个进程。 如果scnt的值非负，则继续执行当前进。最后使用lck释放锁。

### Signal

使用lck取得锁，然后将scnt的值加一。如果scnt的值小于等于零，则从等待队列中取出第一个进程，将其状态设置为ready。 同时在调度器中将该进程的状态设置为ready。之后继续执行当前进程。最后使用lck释放锁。

## ProcessDescr

该类用来描述进程描述符。

### Priority & SetPriority

分别用来取得和设置进程优先级。

### ProcessStatus & SetProcessTo...

分别用来取得进程状态和设置进程状态。

### StoreSize

用来获得内存大小。

### StoreDescr & SetStoreDescr

分别用来获得内存描述符和设置内存描述符。

### FullContext & SetFullContext

分别用来获得进程当前全部的上下文信息和设置全部上下文信息。

## ProcessTable

构建从进程号到进程描述符的映射表。在初始化函数中主要实现的是将空闲进程加入到映射表中。

## Context

该类用来描述进程上下文信息。成员变量包括了进程表，调度器和硬件状态信息。

### SaveState

用来保存当前状态。通过调度器取得当前进程，再取得对应进程描述符。 然后从硬件状态中取得寄存器信息等与上下文有关的需要保存的信息。 最后通过SetFullContext将上下文信息保存到进程描述符中。

### RestoreState

用来恢复之前保存的状态。通过进程调度器取得当前进程，再取得对应的进程描述符。 然后通过FullContext函数取得进程描述符中保存的进程上下文信息，再将这些信息设置到硬件状态中。

### SwapOut

用来将当前进程的上下文换出。首先通过进程调度器取得当前进程，再取得对应的进程描述符。 然后将进程的状态设为waiting，并在调度器中设置进程状态为unready。 然后调用SaveState保存上下文信息。最后通知调度器调度下一个进程。

### SwapIn

用来恢复当前进程的上下文信息。首先通过进程调度器取得当前进程，再取得对应的进程描述符。 将当前进程的状态设为running，然后调用RestoreState恢复上下文信息。

### SwapContext

用来切换上下文。通过先后调用SwapOut和SwapIn来实现。

## MailBox

用来描述进程间通信的模型。每条信息由发送者和数据两部分组成。 MailBox中包含了一个消息的队列，用来存储信息。以及一个Lock，用来实现同步互斥。

### PostMessage

用来发送消息。 首先取得锁，然后将信息加入消息队列，然后释放锁。

### HaveMessages

用来查询队列中是否有消息。

### NextMessage

用来接收下一条消息。 首先取得锁，然后从消息队列中取出第一条信息，并返回。

## SemaphoreTable

该类用来描述信号量表。 成员变量包括信号量标识和信号量对象的映射表，和一个Lock对象。

### NewSemaphore

用来生成一个新的信号量对象。 首先获得锁，然后生成一个新的信号量对象，并分配一个id，然后将映射关系保存在表中。 最后释放锁。

### DelSemaphore

用来删除一个信号量对象。 首先取得锁，然后从映射表中删除待删除对象的映射，最后释放锁。

### GetSemaphore

用来取得一个信号量对象。 首先获得锁，然后根据id从表中取得指定的信号量对象，最后释放锁。

## UserLibrary

用来实现进程的创建与销毁。

### CreateProcess

根据输入的优先级，堆栈等信息，将进程状态设为new。 之后在进程表中加入新进程和其描述符的映射，并在调度器中将其加入准备队列。

### TerminateProcess

在调度器中将进程状态设为unready，然后在进程表中删除对应进程。

### Suspend

在调度器中挂起当前进程。