Week 4

Neutrino first uses priority level and then length of wait to arbitrate between threads

Neutrino priority levels

- the priority range is 0 (low) to 255 (high)
- the programmer can assign a thread a priority between 1 and 63. there is a special thread called the "idle thread" which has a priority 0
- if more than one thread has the same priority level, then Neutrino will consider length of wait

when the kernel decides that a thread should be given time to use the CPU is must switch contexts
– save the currently running thread's context information

- including register values, flags, stack information, etc.

– load the new thread's context into the CPU

– Pre-emption

- if a thread is currently using the CPU and a second thread of greater priority requires CPU time, the first thread is preempted

– when all the threads of higher priority are finished with the CPU,
the thread that was preempted is then given CPU time again. This is called resumption

FIFO – First In First Out

- a thread is allocated CPU time for as long as it needs it
- CPU will not be free for other threads until current thread is finished or until a thread of higher priority requires the CPU
- suppose the thread using the CPU contained an infinite loop

– no thread of lower priority would ever get CPU time
– no thread of the same priority would ever get CPU time

Round Robin

- The default scheduling algorithm for new threads.
- a thread is allocated CPU time for a predefined time-slice
- when the thread finishes with the CPU before the time-slice runs out, then the kernel will assign CPU time to another thread based on priority and policy
- if the time-slice runs out before the current thread completes its usage of the CPU, then the

kernel checks if any other threads of the same priority are waiting for CPU time. If so, the context will be switched that the first thread will then have to wait for the CPU

Sporadic
▪ Sporadic allows a thread to run at a high priority for a set period of time (budget).
▪ Once the budget has been exhausted the thread will drop to a lower priority and continue execution there.
▪ Eventually a “replenishment operation” triggers which bumps the thread back to its full priority and it begins to consume its budget again.

Some states:
– RUNNING - thread is currently using the CPU
– READY - in a queque, ready and waiting for the CPU
– BLOCKED - the thread is waiting for something to happen. There are several blocked states, depending on what the thread is waiting
for.

Two blocked states that we have already encountered are:
- MUTEX - blocked waiting for a mutex to unlock
- SEM - blocked waiting for a semaphore value to increase above 0 we’ll

see other blocked states as we go through the course

processes are your "building blocks"

Threads run code, processes own resources

programs deal with the kernel by using special library routines, called “kernel calls”, that execute code in the kernel

The kernel is the core of your system:

The forms of IPC provided by the kernel:
– Messages - exchanging information between proceses
– Pulses - delivering notification to a process
– Signals - interrupting a process and making it do something different (usually termination)

procnto is QNX
–proc for the process manager
–nto for the Neutrino microkernel

process creation and termination
• spawn / exec / fork

Threads have two basic states:
blocked
• REPLY blocked is waiting for a IPC reply
• MUTEX blocked is waiting for a mutex
• RECEIVE blocked is waiting to get a message
runnable
• RUNNING actually using the CPU
• READY waiting while someone else is running..

Shared Objects:
– are libraries loaded and linked at run time
– one copy used (shared) by all programs using
library
– also sometimes called DLLs
• shared objects and DLLs use the same
architecture to solve different problems