MP4 Design Documentation

Name: Shiyang Chen UIN: 623009273

Scheduler.H/.C

We use an array of 4 elements to implement the FIFO queue. Two pointers, front and rear, point to the head and the tail of the queue. Both pointers value are 0 at the beginning. When we say “point to”, we mean there is an actual element in that position. So each time when one element leaves the queue, we use statements “item=queue[front]; front=(front+1)%M;” where M is the capacity of the queue which is 4 in our case. When a new element is added into the queue, we use statements “queue[rear]=item; rear=(rear+1)%M;”.

In order to determine whether the queue is full or empty, another variable, tag, is introduced. Whenever an element leaves the queue, tag becomes “0”. Whenever a element is added into the queue, tag becomes “1”. Therefore, when “tag==1 AND front==rear”, the queue is full. When “tag==0 AND front==rear”, the queue is empty.

Thread.C

The major modification of thread.c occurs in thread_shutdown() function. When a thread terminates, it yields the control on CPU to the next thread in ready queue. Since we have defined a Scheduler * variable SYSTEM_SCHEDULER in kernel.c, in order to keep kernel.c intact we declare this variable again in thread.c by the statement “extern Scheduler * SYSTEM_SCHEDULER;” No more changes are maded to this file. More detail can be found in comments.

Kernel.C

Except for uncommenting “#define _TERMINATING_FUNCTIONS_” and “#define _USES_SCHEDULER_”, no more changes are maded.

Future progress

I am working on bonus option 1, which is to enable interrupt by modifying machine.c. I will upgrade the code in the next few days.