EENG 348/CPSC338: Digital Systems

Kevin Truong & Rob Brunstad

April 30, 2018

1 Lab 5 Implementation

1.1 Part 1

For creating processes on the fly, we traverse the ready queue whenever there is a call to ready_queue and add it to the end of the queue. To free the memory associated with the process_t struct, we free the stack pointer and then the entire struct.

Algorithm 1 Memory

```
1: procedure FREEING MEMORY
2:     free(process_t→sp)
3:     free(process_t)
4: end procedure
```

1.2 Part 2

Part 2 was implemented by keeping a shorted queue at all times. Process_create_prio() was very similar to Procress_create and only differed by adding a new field called prio into process_t. It is also important to note the Process_create() was modified so that by default,

Algorithm 2 Create

```
1: procedure PROCESS_CREATE_PRIO
2:     Allocate space for the stack pointer and check if the stack can allocate that much
       memory
3:     Malloc space for a new process
4:     Set the fields and assign prio to be equal to the input priority
5: end procedure
```

ordinary processes have priority 128. The queue was sorting in decreasing order using the priority as the key.

Algorithm 3 Sorting

```
1: procedure SORT QUEUE IN DECREASING ORDER
2:     if the tail has greater or equal priority than the new process then
3:         Add the procress to the end
4:     else if the new procress has higher priority then
5:         Add to the beginning of the queue
6:     else
7:         Traverse queue until the new process has higher priority than some process in the
   queue
8:     end if
9: end procedure
```

1.3 Part 3

For the real-time scheduling queue, we assigned the real-time task a priority between 129 and 255, since normal tasks have a priority of 128. We kept track of the shortest deadline and the longest deadline and we updated the deadlines of all the real-time tasks if either the shortest or the longest deadlines change. This maintained the decreasing order in the queue. To keep track of wrong worst case execution estimated, we kept track of the duration of the long each process ran.

Algorithm 4 Real-Time

```
1: procedure REAL-TIME SCHEDULING
2:     if no real time tasks have been scheduled yet then
3:         new_process→prio = 255
4:         shortest deadline = deadline
5:         longest deadline = deadline
6:     else if there is a new process with a shorter or longer deadline then
7:         Recompute shortest and longest deadlines
8:         Recompute priority for all real-time jobs
9:     else
10:        Calculate priority with given shortest and longest deadlines
11:    end if
12: end procedure
```