**Functions Required**

**int thread\_get\_nice(void)**

Returns the current thread’s *nice* value

**void thread\_set\_nice(int new\_nice)**

Sets the current thread’s nice value to new\_nice and recalculates the thread’s priority based on the new value (see Section B.2 [Calculating Priority], page 91). If the running thread no longer has the highest priority, yields.

**int thread\_get\_recent\_cpu(void)**

Returns 100 times the current thread’s recent\_cpu value, rounded to the nearest integer.

**int thread\_get\_load\_avg(void)**

Returns 100 times the current system load average, rounded to the nearest integer.

**Regarding Niceness**

* Nice (range -20 to 20)
* Starts with a value of 0. (set in initialize thread section)

**Calculating Priority**

* 64 priorities numbered 0 (PRI\_MIN) to 63 (PRI\_MAX)
* Calculated initially at thread initialization.
* Recalculated once every **fourth** clock tick, for every thread.
* Calculation:

priority = PRI\_MAX – (recent\_cpu / 4) – (nice \* 2)

**Calculating recent\_cpu**

* Initial value of recent\_cpu is 0 (set in initialize thread section)
* Calculation:

recent\_cpu = (2 \* load\_avg) / (2 \* load\_avg + 1) \* recent\_cpu + nice

* Recalculations of recent\_cpu be made exactly when system counter reaches a multiple of a second:

timer\_ticks() % TIMER\_FREQ == 0

* recent\_cpu can be negative
* \*\*\*Compute with coefficient of recent\_cpu before multiplying

**Calculating load\_avg**

* At system boot, it is initialized to 0.
* Updated once per second, with calculation:

load\_avg = (59/60) \* load\_avg + (1/60) \* ready\_threads

* load\_avg must be updated exactly when the system tick counter reaches a multiple of a second:

timer\_ticks() % TIMER\_FREQ == 0

**Regarding Fixed-Point Real Arithmetic**