Operating System Design MP2 Report

Pin-Xuan Lee (plee18)

Yu-Lin Chien (ychien8)

Shuo-Yang Wang (swang234)

Wei-Tze Tsai (wtsai10)

Contain files:

Makefile mp2.c mp2_given.h userapp.c userapp.h reinstall_and_test_module.sh mapping.csv calculate_mapping.c mapping.csv

How to run:

make mapping

./calculate_mapping 720 (generate "mapping.csv", which is already generated by the work station. You can adjust the number of 720 to whatever you want, bigger number will result in longer execution time. For 720, it takes approximately an hour)

make

sudo insmod mp2.ko

./userapp 1000 200 10 &

./userapp 500 100 40 &

./userapp 100 10 70 &

./userapp 150 15 60 &

./userapp 150 50 60 &

./userapp 1000 300 8 &

less log

Run in another way:

sudo su

./reinstall_and_test_module.sh (without regenerating “mapping.csv”)

Implementation and design decisions:

Init:

Create /proc/mp2 directory and create /proc/mp2/status file.

Initialize slab, spinlock, timer, and dispatching thread.

mp2_register_process():

Get pid, period, and computation for the process, create and insert mp2_task_struct into linked list if the process passes the admission control check successfully.

mp2_yield_process():

Update timer, wake up the scheduler, and sleep.

mp2_unregister_process():

Remove the process form the linked list, and release memory of the process.

mp2_write():

Get either register, yield, or unregister messages from userapp.

timer_callback():

The timer callback function (top half) will be intrigued each round at each process’s period time, and the scheduler (bottom half, dispatching thread) will be called.

Dispatching thread:

The dispatching thread will traverse the linked list, and use get_highest_priority_ready_task () to find the task (at READY state) with the highest priority, and then preempt the current task with this task (if there exists a running task. If no, just run the task). After that, the

scheduler will sleep again. In addition, if there does not exist a task in READY state, set the current running task to a lower priority.

exit():

Remove /proc/mp2/status file and /proc/mp2 directory, stop dispatching thread, and free memory.

userapp.c

Read the mapping.csv file to determine how much iteration should the do_job() function should execute. In addition, we output the start, end, and execution time of each do_job() function in each task to a single file called “log.”

calculate_mapping.c

Create a list which can map the iteration of do_job() to the execution time in millisecond.

Screen shot:

```
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
7698 500 100 1
```

```
7700    150    15     14.803955     156.384033     171.187988
7701    150    50     54.012939     176.245850     230.258789
7700    150    15     14.611084     308.362061     322.973145
7701    150    50     52.429932     325.303955     377.733887
7700    150    15     13.415039     460.283936     473.698975
7701    150    50     51.995850     478.346924     530.342773
7700    150    15     13.135986     633.412109     646.548096
7701    150    50     50.019043     648.923828     698.942871
7698    500    100    102.781982    530.036133     632.818115
7700    150    15     13.263184     763.573975     776.837158
7701    150    50     50.022949     779.890869     829.913818
7700    150    15     16.936035     915.605957     932.541992
7701    150    50     53.396973     935.660889     989.057861
7698    500    100    98.082031     1003.393066    1101.475098
7700    150    15     14.131104     1102.051025    1116.182129
7701    150    50     49.395020     1118.364990    1167.760010
7700    150    15     13.270996     1219.578125    1232.849121
7701    150    50     52.552002     1235.520996    1288.072998
7700    150    15     14.851807     1371.650146    1386.501953
7701    150    50     52.958008     1390.041992    1443.000000
7698    500    100    103.154053    1503.677979    1606.832031
7700    150    15     14.728027     1607.477051    1622.205078
7701    150    50     53.072021     1624.825928    1677.897949
7700    150    15     15.558105     1677.969971    1693.528076
7701    150    50     48.544922     1696.281982    1744.826904
7700    150    15     15.874023     1827.759033    1843.633057
7701    150    50     53.808105     1845.923828    1899.731934
7700    150    15     13.809082     1979.832031    1993.641113
7701    150    50     46.820068     1997.223877    2044.043945
7700    150    15     12.677979     2143.661133    2156.339111
7701    150    50     46.573975     2159.354980    2205.928955
7698    500    100    99.525146     2043.742920    2143.268066
7700    150    15     13.770996     2283.938965    2297.709961
7701    150    50     51.943848     2300.197998    2352.141846
7700    150    15     13.718018     2435.687012    2449.405029
7701    150    50     48.645020     2451.562988    2500.208008
```