OS 2020 project

– Project 1 –

B06902019 資工三 洪佳生

Kernel

- **Version**:
  Linux 4.14.25 -> https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.14.25.tar.xz
- **Added syscall 334: sys_my_printk()**
  - prints a string to dmesg

Design

main.c

- Read the input (number of process, scheduling policy, process name, ready time, execution time)
- Use scheduler() funtion to implement sheduling.

process.c

Define the followin function:

- TIME_UNIT(): define a basic unit of execution time.

- assign_proc_core(pid, core): decide which core the process “pid” will run on.

- proc_out(pid): use sched_setscheduler() to reduce the priority “pid” of process and then assign the child process back to the core 0 where the parent process is running.

- proc_wakeup(pid): assign the child process to the core where the core 1 and then use sched_setscheduler() to raise the priority “pid” of process.

- proc_exec(Process):

  (1) Scheduling process call fork() to simulate the process which is ready and stops child process by reduecing its priority until parent process wakes it up.Because parent and child process are in the same core. So due to the priority,child process won’t run if it shouldn’t run.But in case the child process run in the unavailable time and unfortunately start the timer, we set a while() loop to avoid the above problem and it will break the loop when child process priority is raised by parent process.

  (2) When the timer starts, child process will enter a while() loop for execution time of TIME_UNIT().

  (3) When the timer ends, use system call to output the message into dmesg.

scheduler.c

- First, we will assign a particular core 0 to scheduling process and raise its priority to the highest level to prevent potential preemptive problem between scheduling process and the child processes which are are generated by fork().

- Second,initializes child process by -1 to represent not ready process or already finished process.

- In while(1) loop, we will kepp doing tje following five steps until all processes are done.

  Step1:
  Check whether there are some process are already done in last UNIT_TIME.If so,label process’s pid into -1 and finished processes number += 1.If finished processes number is equal to total process number,break the while(1) loop and finish scheduling.

  Step2:
  Check whether there are some processes which are ready and if so,implement proc_exec().

  Step3:
  Use switch to choose the scheduling policy to find the next process to implement it.

  There are four policies:
  Assume there are some ready processes.
  FIFO():
  When the implementing process i finish,the next implement process will be i+1.
  SJF():
  When the implementing process i finish,the next implement process will be the shortest execution time process in the ready queue.
  RR():In each time slice 500 UNIT_TIME or the implementing process finish in time slice,the next implementing process will be the next in the ready queue.
  PSJF():
  NO matter the implementing process i is finished,the next implement process will be the shortest execution time process in the ready queue.

  Step4:
  If the next running process isn’t the same as now running process,then scheduling process will reduce the priority of now runnig process and raise the priority of next running process.
  Step5:
  run a TIME_UNIT in parent and now runnig child process simutaneously.

Result

Unit time	0.001377

FIFO_1.txt

Task	Expected time	Execution time	Error rate
P1	500	489.9160	2.0168%
P2	500	487.8090	2.4382%
P3	500	482.1090	3.5782%
P4	500	471.0580	5.7884%
P5	500	484.9500	3.0100%

FIFO_2.txt

Task	Expected time	Execution time	Error rate
P1	80000	81020.7230	1.2759%
P2	5000	5120.1950	2.4039%
P3	1000	1064.8440	6.4844%
P4	1000	1050.3880	5.0388%

FIFO_3.txt

Task	Expected time	Execution time	Error rate
P1	8000	8272.3870	3.4048%
P2	5000	5038.0870	0.7617%
P3	3000	3135.0080	4.5003%
P4	1000	1002.2240	0.2224%
P5	1000	959.5250	4.0475%
P6	1000	962.7360	3.7264%
P7	4000	3831.1020	4.2225%

FIFO_4.txt

Task	Expected time	Execution time	Error rate
P1	2000	1908.3270	4.5836%
P2	500	485.3650	2.9270%
P3	200	186.5370	6.7315%
P4	500	490.9060	1.8188%

FIFO_5.txt

Task	Expected time	Execution time	Error rate
P1	8000	7615.4980	4.8063%
P2	5000	4836.5720	3.2686%
P3	3000	2844.0080	5.1997%
P4	1000	969.2790	3.0721%
P5	1000	965.2920	3.4708%
P6	1000	972.8310	2.7169%
P7	4000	3875.7150	3.1071%

PSJF_1.txt

Task	Expected time	Execution time	Error rate
P4	3000	2819.2620	6.0246%
P3	8000	7580.1310	5.2484%
P2	15000	14535.3480	3.0977%
P1	25000	24435.5740	2.2577%

PSJF_2.txt

Task	Expected time	Execution time	Error rate
P2	1000	1022.4190	2.2419%
P1	4000	4247.3570	6.1839%
P4	2000	1903.9320	4.8034%
P5	1000	948.0400	5.1960%
P3	7000	7013.6780	0.1954%

PSJF_3.txt

Task	Expected time	Execution time	Error rate
P2	500	531.5470	6.3094%
P3	500	513.2470	2.6494%
P4	500	500.7710	0.1542%
P1	3500	3587.3940	2.4970%

PSJF_4.txt

Task	Expected time	Execution time	Error rate
P3	1000	992.0110	0.7989%
P2	3000	3008.8180	0.2939%
P4	4000	4093.8050	2.3451%
P1	7000	7149.6360	2.1377%

PSJF_5.txt

Task	Expected time	Execution time	Error rate
P1	100	97.8610	2.1390%
P3	200	190.5140	4.7430%
P2	4000	4163.1560	4.0789%
P4	4000	4370.9480	9.2737%
P5	7000	7082.2750	1.1754%

RR_1.txt

Task	Expected time	Execution time	Error rate
P1	500	476.9680	4.6064%
P2	500	486.6720	2.6656%
P3	500	484.8670	3.0266%
P4	500	480.1370	3.9726%
P5	500	479.8520	4.0296%

RR_2.txt

Task	Expected time	Execution time	Error rate
P1	7500	7469.9370	0.4008%
P2	8500	8196.8040	3.5670%

RR_3.txt

Task	Expected time	Execution time	Error rate
P3	14000	13126.0550	6.2425%
P1	19000	18079.2580	4.8460%
P2	18000	16989.6840	5.6129%
P6	21000	19965.1630	4.9278%
P5	23500	22642.8740	3.6473%
P4	25000	23560.3680	5.7585%

RR_4.txt

Task	Expected time	Execution time	Error rate
P4	4000	3718.7060	7.0323%
P5	4000	3808.7160	4.7821%
P6	4000	3793.9230	5.1519%
P3	13500	12804.6280	5.1509%
P7	15000	14382.8930	4.1140%
P2	19500	18849.6690	3.3350%
P1	23000	21573.3470	6.2028%

RR_5.txt

Task	Expected time	Execution time	Error rate
P4	4000	3842.8120	3.9297%
P5	4000	3813.7980	4.6551%
P6	4000	3829.5170	4.2621%
P3	13500	13016.0900	3.5845%
P7	15000	14437.9160	3.7472%
P2	19500	18747.4210	3.8594%
P1	23000	22300.7630	3.0402%

SJF_1.txt

Task	Expected time	Execution time	Error rate
P2	2000	1924.8090	3.7596%
P3	1000	943.7150	5.6285%
P4	4000	3844.5010	3.8875%
P1	7000	6715.4740	4.0647%

SJF_2.txt

Task	Expected time	Execution time	Error rate
P1	100	93.8440	6.1560%
P3	200	186.3710	6.8145%
P2	4000	3794.1680	5.1458%
P4	4000	3766.8300	5.8293%
P5	7000	6937.6590	0.8906%

SJF_3.txt

Task	Expected time	Execution time	Error rate
P1	3000	2922.5990	2.5800%
P4	10	9.5890	4.1100%
P5	10	9.5760	4.2400%
P6	4000	3934.9560	1.6261%
P7	4000	3965.3410	0.8665%
P2	5000	4790.1220	4.1976%
P3	7000	6688.2530	4.4535%
P8	9000	8568.6760	4.7925%

SJF_4.txt

Task	Expected time	Execution time	Error rate
P1	3000	2956.2080	1.4597%
P2	1000	993.3720	0.6628%
P3	4000	3733.0290	6.6743%
P5	1000	980.1890	1.9811%
P4	2000	1865.4100	6.7295%

SJF_5.txt

Task	Expected time	Execution time	Error rate
P1	2000	1951.3840	2.4308%
P2	500	453.0360	9.3928%
P3	500	489.2080	2.1584%
P4	500	457.9260	8.4148%

TIME_MEASUREMENT.txt

Task	Expected time	Execution time	Error rate
P0	500	502.1140	0.4228%
P1	500	520.3880	4.0776%
P2	500	511.5010	2.3002%
P3	500	512.6380	2.5276%
P4	500	478.3490	4.3302%
P5	500	487.5220	2.4956%
P6	500	483.5790	3.2842%
P7	500	517.1410	3.4282%
P8	500	491.5000	1.7000%
P9	500	495.2670	0.9466%

Conclusion

- **Correctness**
  The error rate between theoretical execution time and real execution time aren’ t bigger than 10%
- **Work Loading**
  When CPU was busy or has other jobs,the performance would be affected.