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 333: sys_my_clock()**
  - Get the data by getnstimeofday()
- **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():Implementing in queue.When process is ready,it will be pushed into the queue. In each time slice 500 UNIT_TIME or the running process **i** finish in time slice,the next running process will be select by the header of ready queue and then if process **i** is not finished yet,push it back to the 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

Error Rate: round off to the 4nd decimal place

Start time, End time: theoretical time (calculate by math)

Expect exec time:the time the process should finish

My start time, My end time:the time I get from my program (Initialize by minimun My start time)

My exec time:the time the process finish in my program

Error rate: the error between Expect exec time & My exec time

Unit Time

Unit Time	0.001377

FIFO_1.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	0	500	500	0.00	489.92	489.92	2.02%
P2	500	1000	500	501.70	989.51	487.81	2.44%
P3	1000	1500	500	989.66	1471.77	482.11	3.58%
P4	1500	2000	500	1473.90	1944.95	471.06	5.79%
P5	2000	2500	500	1955.16	2440.11	484.95	3.01%

FIFO_2.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	0	80000	80000	0.00	81020.72	81020.72	1.28%
P2	80000	85000	5000	81480.00	86600.20	5120.19	2.40%
P3	85000	86000	1000	86615.81	87680.65	1064.84	6.48%
P4	86000	87000	1000	87686.84	88737.23	1050.39	5.04%

FIFO_3.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	0	8000	8000	0.00	8272.39	8272.39	3.40%
P2	8000	13000	5000	8331.03	13369.11	5038.09	0.76%
P3	13000	16000	3000	13616.25	16751.26	3135.01	4.50%
P4	16000	17000	1000	16779.69	17781.91	1002.22	0.22%
P5	17000	18000	1000	17852.79	18812.31	959.52	4.05%
P6	18000	19000	1000	18856.43	19819.16	962.74	3.73%
P7	19000	23000	4000	19845.95	23677.05	3831.10	4.22%

FIFO_4.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	0	2000	2000	0.00	1908.33	1908.33	4.58%
P2	2000	2500	500	1908.45	2393.82	485.37	2.93%
P3	2500	2700	200	2393.94	2580.48	186.54	6.73%
P4	2700	3200	500	2580.58	3071.49	490.91	1.82%

FIFO_5.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	0	8000	8000	0.00	7615.50	7615.50	4.81%
P2	8000	13000	5000	7684.69	12521.26	4836.57	3.27%
P3	13000	16000	3000	12521.39	15365.40	2844.01	5.20%
P4	16000	17000	1000	15374.26	16343.53	969.28	3.07%
P5	17000	18000	1000	16343.65	17308.94	965.29	3.47%
P6	18000	19000	1000	17329.71	18302.54	972.83	2.72%
P7	19000	23000	4000	18302.65	22178.37	3875.72	3.11%

PSJF_1.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P4	3000	6000	3000	2952.96	5772.22	2819.26	6.02%
P3	2000	10000	8000	1985.08	9565.21	7580.13	5.25%
P2	1000	16000	15000	999.11	15534.46	14535.35	3.10%
P1	0	25000	25000	0.00	24435.57	24435.57	2.26%

PSJF_2.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P2	1000	2000	1000	1025.34	2047.76	1022.42	2.24%
P1	0	4000	4000	0.00	4247.36	4247.36	6.18%
P4	5000	7000	2000	5300.43	7204.36	1903.93	4.80%
P5	7000	8000	1000	7246.28	8194.32	948.04	5.20%
P3	4000	11000	7000	4267.75	11281.43	7013.68	0.20%

PSJF_3.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P2	500	1000	500	527.20	1058.74	531.55	6.31%
P3	1000	1500	500	1058.93	1572.18	513.25	2.65%
P4	1500	2000	500	1591.81	2092.58	500.77	0.15%
P1	0	3500	3500	0.00	3587.39	3587.39	2.50%

PSJF_4.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P3	100	1100	1000	105.83	1097.84	992.01	0.80%
P2	0	3000	3000	0.00	3008.82	3008.82	0.29%
P4	3000	7000	4000	3008.94	7102.74	4093.80	2.35%
P1	7000	14000	7000	7102.88	14252.52	7149.64	2.14%

PSJF_5.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	100	200	100	0.00	97.86	97.86	2.14%
P3	200	400	200	98.11	288.63	190.51	4.74%
P2	400	4400	4000	288.74	4451.90	4163.16	4.08%
P4	4400	8400	4000	4452.02	8822.97	4370.95	9.27%
P5	8400	15400	7000	8823.09	15905.36	7082.27	1.18%

RR_1.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	0	500	500	0.00	476.97	476.97	4.61%
P2	500	1000	500	477.08	963.75	486.67	2.67%
P3	1000	1500	500	963.93	1448.80	484.87	3.03%
P4	1500	2000	500	1451.45	1931.59	480.14	3.97%
P5	2000	2500	500	1933.55	2413.40	479.85	4.03%

RR_2.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	600	8100	7500	0.00	7469.94	7469.94	0.40%
P2	1100	9600	8500	543.74	8740.55	8196.80	3.57%

RR_3.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P3	4200	18200	14000	2864.92	15990.97	13126.06	6.24%
P1	1200	19700	18500	0.00	18079.26	18079.26	4.85%
P2	2700	20200	17500	1422.31	18412.00	16989.68	5.61%
P6	8200	28200	20000	5658.99	25624.15	19965.16	4.93%
P5	6700	30200	23500	5202.43	27845.31	22642.87	3.65%
P4	6200	31200	25000	4744.18	28304.54	23560.37	5.76%

RR_4.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P4	1500	5500	4000	1414.06	5132.77	3718.71	7.03%
P5	2000	6000	4000	1881.51	5690.23	3808.72	4.78%
P6	2500	6500	4000	2353.02	6146.94	3793.92	5.15%
P3	1000	14500	13500	936.79	13741.42	12804.63	5.15%
P7	3500	18500	15000	2820.43	17203.33	14382.89	4.11%
P2	500	20000	19500	468.53	19318.20	18849.67	3.34%
P1	0	23000	23000	0.00	21573.35	21573.35	6.20%

RR_5.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P4	1500	5500	4000	1452.16	5294.98	3842.81	3.93%
P5	2000	6000	4000	1929.89	5743.69	3813.80	4.66%
P6	3000	7000	4000	2420.96	6250.48	3829.52	4.26%
P3	1000	14500	13500	967.05	13983.14	13016.09	3.58%
P7	3500	18500	15000	2896.11	17334.03	14437.92	3.75%
P2	500	20000	19500	484.09	19231.51	18747.42	3.86%
P1	0	23000	23000	0.00	22300.76	22300.76	3.04%

SJF_1.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P2	0	2000	2000	0.00	1924.81	1924.81	3.76%
P3	2000	3000	1000	1924.92	2868.64	943.72	5.63%
P4	3000	7000	4000	2881.88	6726.39	3844.50	3.89%
P1	7000	14000	7000	6726.53	13442.00	6715.47	4.06%

SJF_2.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	100	200	100	0.00	93.84	93.84	6.16%
P3	200	400	200	94.06	280.43	186.37	6.81%
P2	400	4400	4000	282.73	4076.89	3794.17	5.15%
P4	4400	8400	4000	4122.07	7888.90	3766.83	5.83%
P5	8400	15400	7000	7901.57	14839.23	6937.66	0.89%

SJF_3.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	100	3100	3000	0.00	2922.60	2922.60	2.58%
P4	3100	3110	10	2948.51	2958.10	9.59	4.11%
P5	3110	3120	10	2958.60	2968.18	9.58	4.24%
P6	3120	7120	4000	2968.26	6903.22	3934.96	1.63%
P7	7120	11120	4000	6923.27	10888.61	3965.34	0.87%
P2	11120	16120	5000	10888.77	15678.89	4790.12	4.20%
P3	16120	23120	7000	15703.92	22392.17	6688.25	4.45%
P8	23120	32120	9000	22469.48	31038.16	8568.68	4.79%

SJF_4.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	0	3000	3000	0.00	2956.21	2956.21	1.46%
P2	3000	4000	1000	2956.33	3949.70	993.37	0.66%
P3	4000	8000	4000	3949.83	7682.86	3733.03	6.67%
P5	8000	9000	1000	7840.16	8820.35	980.19	1.98%
P4	9000	11000	2000	8820.46	10685.87	1865.41	6.73%

SJF_5.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P1	0	2000	2000	0.00	1951.38	1951.38	2.43%
P2	2000	2500	500	1951.52	2404.55	453.04	9.39%
P3	2500	3000	500	2435.20	2924.41	489.21	2.16%
P4	3000	3500	500	2924.52	3382.44	457.93	8.41%

TIME_MEASUREMENT.txt

process	start time	end time	expect exec time	my start time	my end time	my exec time	error rate
P0	0	500	500	0.00	502.11	502.11	0.42%
P1	1000	1500	500	1026.94	1547.33	520.39	4.08%
P2	2000	2500	500	2072.89	2584.39	511.50	2.30%
P3	3000	3500	500	3097.03	3609.67	512.64	2.53%
P4	4000	4500	500	4106.02	4584.37	478.35	4.33%
P5	5000	5500	500	5094.30	5581.82	487.52	2.50%
P6	6000	6500	500	6076.74	6560.32	483.58	3.28%
P7	7000	7500	500	7073.64	7590.78	517.14	3.43%
P8	8000	8500	500	8116.06	8607.56	491.50	1.70%
P9	9000	9500	500	9112.25	9607.52	495.27	0.95%

Conclusion

- **Correctness**
  In my theoretical time assumption,there is no content switch and some hardware propagation delay, so each unit time is basicly the same.However,there might be some time error.But the error rate between theoretical execution time and real execution time aren still not smaller than 10%.The execution time is smaller than thereotical time because child process maybe run on the unavailable time before timer start. **My end time often be further from end time because previous process time error will affect the next one process,but the my exec time will not be affected by the previous one because if the process start earlier/later,it will end earlier/later.**
- **Work Loading**
  When CPU was busy or has other jobs,such like I/O input and output,the performance would be affected depending on the utilization of CPU.
- **Synchronization**
  Because we use two core to simulate parent and child process,maybe child will run on the unavailable time. So waitpid() is a key funtion in scheduler.c to make synchronization acheivable.