



Operating System Report

Assignment 2

Professor	김태석 교수님
Department	Computer engineering
Student ID	2014722057
Name	김 진아
Class	화5 목6
Date	2016. 11. 4

A. Introduction

fork()와 POSIX thread를 사용해 여러 개의 process와 thread가 0부터 1E9까지의 합을 구하도록 구현한다. 이때 process와 thread의 생성 시간, 전체 프로그램 수행 시간을 측정하기 위해 clock_gettime()을 사용한다. 또한 실험 환경 맞춰 성능 차를 보여주기 위해 loop 횟수, process와 thread의 수를 임의로 조절한다. fork()를 이용한 process 생성은 fork.c에 구현하고 POSIX thread를 이용한 thread 생성은 thread.c에 구현한다.

B. Reference

clock_gettime 검색 = http://tewda.blogspot.kr/2015/02/clock_gettime.html

C. Conclusion

- fork.c

fork()를 통해 여러 개의 process를 구했을 때 다음과 같은 생성시간과 전체 프로그램 수행 시간이 나왔다. 이때 보다 확실한 성능 차를 보여주기 위해 아래와 같은 process의 수와 loop의 횟수를 정해줬다.

loop 횟수	process 수	process 생성시간	전체 프로그램 수행시간
1000	1	0.000114	0.000464
	2	0.000191	0.000708
	3	0.000345	0.001295
	4	0.000414	0.001376
	5	0.000492	0.001714
1000000	1	0.000112	0.005517
	2	0.000197	0.010955
	3	0.000281	0.017944
	4	0.000387	0.022558
	5	0.000513	0.030200

1000000000	1	0.000114	5.111830
	2	0.000208	10.101518
	3	0.000300	14.955075
	4	0.000400	20.357835
	5	0.000484	24.666654

위에 결과를 설명하면 다음과 같다. 일정한 loop의 개수에서 process의 수를 증가할수록 process 생성 시간과 전체 프로그램 수행 시간이 점점 증가하는 것을 볼 수 있었다. 또한 일정한 process의 수를 가지고 loop의 개수를 달리 했을 때 loop를 많이 돌수록 process 생성 시간에서의 많은 차이가 나지 않았지만 전체 프로그램 수행 시간이 확실한 차이를 볼 수 있었다. 이처럼 process의 수에도 시간에 대한 영향을 많이 받지만 확실히 얼마나 많은 loop를 도느냐에 따라 전체 프로그램 수행시간에 영향을 미친다는 사실을 확인할 수 있었다.

- thread.c

POSIX thread를 통해 여러 개의 thread를 생성했을 때 다음과 같은 생성시간과 전체 프로그램 수행 시간이 나왔다. 이때도 fork와 마찬가지로 process의 수와 loop의 횟수를 아래와 같이 조절해줬다.

loop 횟수	process 수	process 생성시간	전체 프로그램 수행시간
1000	1	0.000054	0.000096
	2	0.000073	0.000147
	3	0.000080	0.000179
	4	0.000087	0.000208
	5	0.000094	0.000285
1000000	1	0.000056	0.005378
	2	0.000080	0.010671
	3	0.000088	0.014930

	4	0.000110	0.021327
	5	0.000109	0.026226
1000000000	1	0.000054	4.911247
	2	0.000087	10.029019
	3	0.000108	18.046687
	4	0.000133	20.606413
	5	0.000179	25.635692

위에 결과를 설명하면 다음과 같다. POSIX thread도 fork 결과와 마찬가지로 일정한 loop의 개수에서 process의 수를 증가할수록 process 생성 시간과 전체 프로그램 수행 시간이 점점 증가하는 것을 볼 수 있었다. 또한 일정한 process의 수를 가지고 loop의 개수를 달리 했을 때 loop을 많이 돌수록 process 생성 시간에서의 많은 차이가 나지 않았지만 전체 프로그램 수행 시간이 확실한 차이를 볼 수 있었다. 이처럼 process의 수에도 시간에 대한 영향을 많이 받지만 확실히 얼마나 많은 loop를 도느냐에 따라 전체 프로그램 수행시간에 영향을 미친다는 사실을 확인할 수 있었다.

이때 process 생성 시간 관점에서 살펴보면 POSIX thread를 이용한 process 생성이 fork를 이용한 process 생성보다 시간이 짧게 걸렸다. 또한 전체 프로그램 시간 관점에서 살펴보면 loop의 개수가 많아질수록 POSIX thread를 이용한 process 생성이 fork를 이용한 process 생성보다 시간이 오래 걸렸다.

◆ 2-2

A. Introduction

CPU scheduling 정책이 다중 process의 동작에 어떠한 영향을 미치는지 분석하는 program을 구현한다. 특정 directory를 생성한 뒤 filegen.c에서 directory에 생성할 process의 수만큼 파일을 만들어 1000000000이상의 정수 아무거나 기록하도록 한다. schedtest.c에서 MAX_PROCESSES만큼의 process를 생성한 뒤 sched_setscheduler()을 사용해 CPU 스케줄링 정책을 변경한다. 이때 filegen.c로 생성된 파일의 정수 data을 읽도록 구현한다.

B. Reference

C. Conclusion

SCHED_RR, SCHED_FIFO, SCHED_IDLE, SCHED_NORMAL, SCHED_BATCH를 사용해 스케줄링을 했다. 이때 SCHED_RR와 SCHED_FIFO의 경우 root권한으로 해야 동작이 되므로 실행할 때 sudo를 붙여줘야 한다. SCHED_IDLE, SCHED_NORMAL, SCHED_BATCH를 실행할 때 priority가 0이어야 한다. 다음은 이를 토대로 스케줄링을 했을 때의 결과이다. 이때 MAX_PROCESSES의 수에 따라 process 생성시간과 전체 program 수행시간이 바뀌는지 살펴보기 MAX_PROCESSES의 수를 달리 해봤다.

MAX_PROCESSES	스케줄링 기법	process 생성 시간	전체 프로그램 수행 시간
10	SCHED_RR	0.001024	0.014850
	SCHED_FIFO	0.000979	0.020549
	SCHED_IDLE	0.000855	0.019471
	SCHED_BATCH	0.000894	0.034476
	SCHED_NORMAL	0.001037	0.021982
100	SCHED_RR	0.009241	0.461419
	SCHED_FIFO	0.008913	0.457954
	SCHED_IDLE	0.008495	0.570655
	SCHED_BATCH	0.008815	0.645987
	SCHED_NORMAL	0.023667	0.603184
1000	SCHED_RR	0.108423	44.370266
	SCHED_FIFO	0.102593	40.806681
	SCHED_IDLE	0.133144	49.516016
	SCHED_BATCH	0.138621	53.376146
	SCHED_NORMAL	0.156886	50.654037
	SCHED_RR	0.432255	732.889616

5000	SCHED_FIFO	0.446133	814.516779
	SCHED_IDLE	0.512114	943.325422
	SCHED_BATCH	0.529761	920.059708
	SCHED_NORMAL	0.582592	880.158031

MAX_PROCESSES가 10일 때의 process 생성시간과 전체 program 수행시간의 순위는 다음과 같다.

<process 생성시간>

SCHED_IDLE > SCHED_BATCH > SCHED_FIFO > SCHED_RR > SCHED_NORMAL

<전제 program 수행시간>

SCHED_RR > SCHED_IDLE > SCHED_FIFO > SCHED_NORMAL > SCHED_BATCH

MAX_PROCESSES가 100일 때의 process 생성시간과 전체 program 수행시간의 순위는 다음과 같다.

<process 생성시간>

SCHED_IDLE > SCHED_BATCH > SCHED_FIFO > SCHED_RR > SCHED_NORMAL

<전제 program 수행시간>

SCHED_FIFO > SCHED_RR > SCHED_IDLE > SCHED_NORMAL > SCHED_BATCH

MAX_PROCESSES가 1000일 때의 process 생성시간과 전체 program 수행시간의 순위는 다음과 같다.

<process 생성시간>

SCHED_FIFO > SCHED_IDLE > SCHED_BATCH > SCHED_RR > SCHED_NORMAL


<전제 program 수행시간>

SCHED_FIFO > SCHED_RR > SCHED_IDLE > SCHED_NORMAL > SCHED_BATCH

MAX_PROCESSES가 5000일 때의 process 생성시간과 전체 program 수행시간의 순위는 다음과 같다.

<process 생성시간>

SCHED_RR > SCHED_FIFO > SCHED_IDLE > SCHED_BATCH > SCHED_NORMAL



<전제 program 수행시간>

SCHED_RR > SCHED_FIFO > SCHED_NORMAL > SCHED_BATCH > SCHED_IDLE

이처럼 MAX_PROCESS의 수가 바뀔 때마다 스케줄링 정책에서의 process 생성시간과 전체 program 실행 시간이 바뀌는 것을 확인 할 수 있다.