2019. 10. 31.

**Operating System**

Assignment # 2

박 철 수 교수님

컴퓨터정보공학부

2015722087

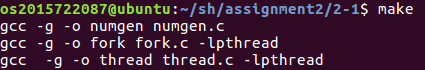
김 민 철

**Introduction**

이번 과제는fork 함수와 pthread 함수를 사용하여 파일에 기록된 정보를 읽어오는 동작을 처리하는 과정에서 fork 함수와 pthread 함수의 사용법을 익히고 반환하는 값의 처리와 동작 시간에서의 차이점을 분석해보는 것이 2-1 과제의 목표이며, Linux의 3가지 스케줄링 방법을 조사하고 모든 스케줄링 정책을 적용한 후에 시간 비교를 통해 스케줄링 정책을 이해하고 분석하는 것이 2-2과제의 목표이다.

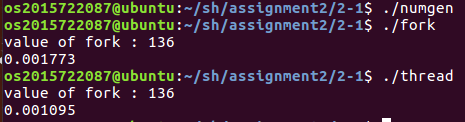
**Conclusion**

Assignment 2-1



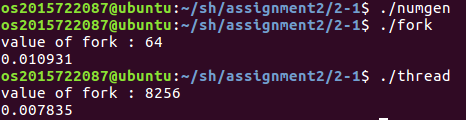
Makefile을 생성하여 한번의 make 명령어로 세개의 코드가 모두 컴파일 되도록 작성하였다.

–lpthread 옵션을 맨 앞에 작성했다가 semaphore와 pthread 헤더 파일을 컴파일하지 못하는 오류가 발생하여 시행착오 끝에 가장 뒤에 작성해야 한다는 것을 알게되었다. 컴파일 시 사용해야 하는 이유는 gcc 컴파일 스크립트에 pthread 라이브러리가 포함되지 않아서 인데, 이 옵션을 작성함으로써 pthread 라이브러리를 포함시켜 빌드하게 된다.



Fork() 함수를 이용하여 MAX 프로세스의 수를 8로 하여 1~16의 숫자를 모두 더하는 작업을 한 결과와, pthread를 이용하여 작업을 한 결과, 결과 값은 당연히 같은 값이 출력되었으나, Running Time이 차이나는 것을 확인할 수 있었다.

이유는 thread의 경우에는 부모와 공유하는 자원이 많은 데에 비해 fork는 공유하는 자원이 비교적 적어서 생성해야하는 것이 더 많기 때문에 수행 시간에서 차이가 발생하였다.



MAX 프로세스의 수를 64로 늘린 뒤, 같은 동작을 시행한 결과이다. 수행 시간은 fork() 함수를 사용하였을 때 더 오래 걸리는 것을 확인하였으나, 결과 값이 틀리게 나왔다.

이유는 종료 상태 값은 16비트의 값으로 반환하는데, 하위 8비트는 자식 프로세스가 비정상적으로 종료되었을 때, Signal 번호를 입력하게 되는 비트 자리이다. 이때는 상위 8비트가 모두 0으로 반환되며, 자식 프로세스가 정상적으로 종료되었을 때는 상위 8비트에 반환 값을 담아 반환하게 되며 하위 8비트가 모두 0으로 반환되기 때문에, 부모 프로세스에서 자식 프로세스의 반환 값을 받았을 때, 8번 shift right, 즉 256으로 나눠주어야 정상적인 값을 확인할 수 있으며, 이때 값은 8비트를 이용한 최대 값 만큼만 반환할 수 있으므로 최대 255 값 까지만 정상적으로 반환할 수 있다.

Assignment 2-2

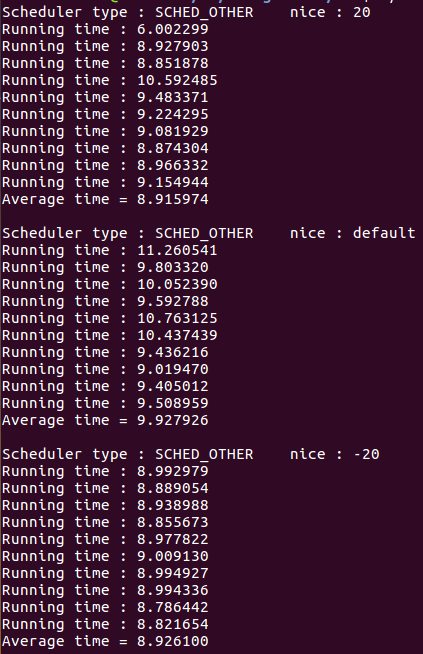


-lrt 옵션의 존재를 모르고 컴파일을 하다가 계속해서 오류가 발생하여 시행착오를 겪었다. 같은 과제를 진행하던 친구가 구글에서 알게 되어 해결하게 되었다.

-lrt 옵션에서 l은 library를 의미하여 rt library를 포함한다는 의미이다.



또 FIFO와 RR의 정책은 슈퍼 유저만 가능하다고 하여 SuperUser 권한으로 실행하였다.

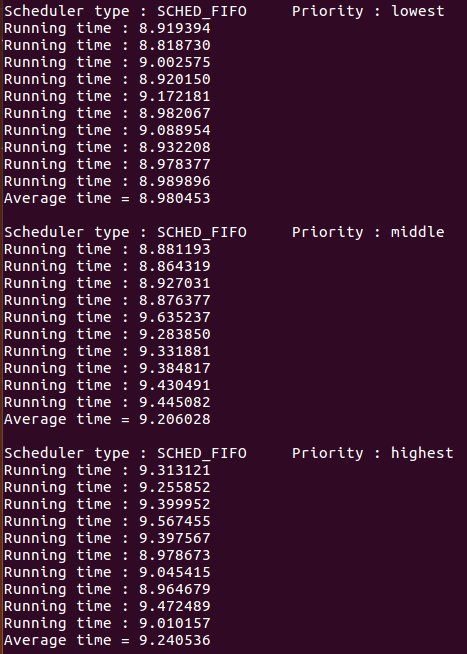


SCHED\_OTHER, the standard round-robin time-sharing policy 라는 Linux에서 지원하는 CPU 스케줄링 정책을 적용한 뒤, nice 값의 변화에 따른 수행 시간을 측정한 결과이다.

이 정책은 가장 일반적인 정책으로 타임 슬라이스와 커널에 의해 지속적으로 변경되는 동적 우선 순위를 사용한다. 이 정책에서 우선 순위 계산식은

우선 순위 = 남은 타임 슬라이스 + (20 - nice)의 비례 값

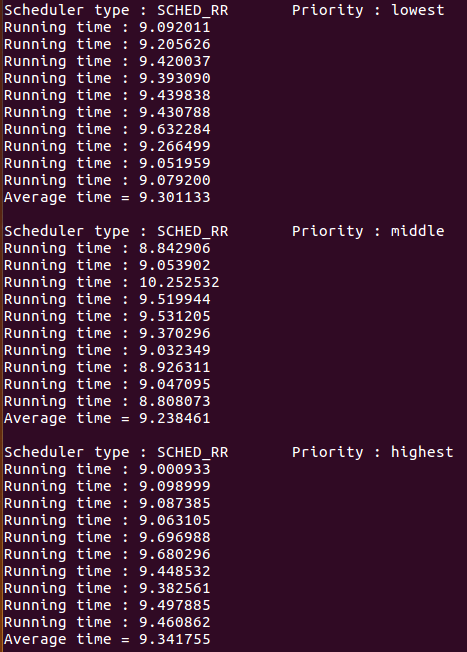
이다. 따라서 nice 값은 -20이 lowest이고 20이 highest이지만, 음수이면 우선순위가 highest이고, 양수 일 땐 lowest가 된다. 수행 시간에서의 차이는 발견하지 못했다.



SCHED\_FIFO, a first-in, first-out policy 라는 Linux에서 지원하는 CPU 스케줄링 정책을 적용한 뒤, 임의로 부여한 Priority의 변화에 따른 수행 시간을 측정한 결과이다.

이 정책은 실시간 프로세스에 사용되는 정책으로 SCHED\_OTHER 정책의 프로세스들보다 높은 우선 순위를 가지며 타임 슬라이스의 개념이 없어 스스로 CPU를 반환할 때만 반환이 된다.

우선 순위는 1부터 99까지 존재하므로 lowest로 1, highest로 99, middle로 50을 주어 시간을 비교해보았으나 시간에서의 차이가 발생하지 않았다.



SCHED\_RR, a round-robin policy 라는 Linux에서 지원하는 CPU 스케줄링 정책을 적용한 뒤, 임의로 부여한 Priority의 변화에 따른 수행 시간을 측정한 결과이다.

이 정책 역시 SCHED\_OTHER 정책의 프로세스들 보다 높은 우선 순위를 가지며, 같은 우선 순위에서는 타임 슬라이스에 의한 라운드 로빈 스케줄링 기법이 적용된다.

우선 순위는 1부터 99까지 존재하므로 lowest로 1, highest로 99, middle로 50을 주어 시간을 비교해보았으나 시간에서의 차이가 발생하지 않았다.

이번 과제에서는 fork() 함수와 pthread 함수를 사용하여 수행 시간을 비교해보는 과제를 진행하면서 이론으로만 배웠던 프로세스를 생성하는 것과 스레드를 생성하는 것의 차이를 직접 코드를 작성해보면서 이해할 수 있게 되었다. 공유하는 영역이 많으면 새로 생성해야 하는 것이 적고, 공유하는 것이 적으면 거의 새로운 프로세스를 생성하게 되므로 시간이 더 오래 걸린다는 것을 직접 경험해보니 더 이해가 잘 된 것 같다.

또 스케줄링 정책에 대해서도 더 공부하게 되었는데, 아쉽게도 수행 시간에서의 차이를 확인하지 못해 확실한 이해는 하지 못했지만, 슈퍼 유저 권한으로 실행하는 것과, 사용자가 임의로 스케줄링 정책을 변경하고 우선 순위를 변경하는 방법이 어렵지 않다는 것을 알게 되었다.