Project 03 - Test Program

Implementing simple schedulers on xv6 and locking method

Due date

2024. 05. 19. 11:59 pm

기본 구조

- 다음의 코드들은 xv6 디렉토리와 Makefile에 추가하여 컴파일하고 실행합니다.

- 테스트시 실행할 프로그램 소스
  - thread_test.c
  - thread_exec.c
  - thread_exit.c
  - thread_kill.c
  - thread_locking.c
  - hello_thread.c (직접 실행할 필요는 없음)

- 출력은 꼬일 수 있고, 순서가 달라질 수도 있습니다. 하지만 실행 도중 에러가 나거나, 실패했다는 메시지가 발생하지 않게 해야 합니다.

thread_test.c - Test 1

- 스레드 API의 기본적인 기능(create, exit, join)과 스레드 사이에서 메모리가 잘 공유되는지를 평가합니다.
- 스레드 0은 곧바로 종료하고, 스레드 1은 2초 후 종료합니다. 두 스레드가 모두 종료된 후 메인 스레드에서 join이 종료되어야 합니다.

```
$ thread_test
Test 1: Basic test
Thread 0 start
Thread 0 end
Thread 1 start
Parent waiting for children...
Thread 1 end
Test 1 passed
```

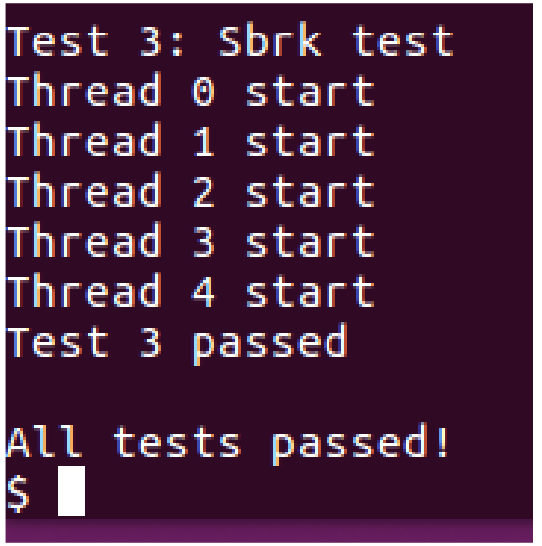
thread_test.c - Test 2

- 스레드 내에서 fork를 했을 때 올바르게 동작하는지 평가합니다.
- fork 이후 반드시 부모 프로세스는 기존 프로세스의 주소 공간에서 계속 동작해야 하고, 자식 프로세스는 분리된 주소 공간에서 동작해야 합니다.
- 즉, 처음 fork가 이루어졌을 때의 메모리 내용은 같게 시작하지만, 이후 자식 프로세스가 사용하는 메모리는 부모 프로세스에게는 아무런 영향을 미치지 않고, 그 반대로도 마찬가지입니다.
- 만일 자식 프로세스가 부모 프로세스와 주소 공간을 공유한다면 테스트 프로그램이 에러를 감지할 것입니다.

```
Test 2: Fork test
Thread 0Thread 1 start
Thread 2 start
Thread 3 start
Thread 4 start
 start
Child of thread 1 start
Child of thread 3 start
Child of thread 0 start
Child of Child of thread 4 start
thread 2 start
Child of thread 1 end
Child of thread 3 end
TThread 3 end
Child of thread 0 end
hread 1 end
Child of thread 2 end
Child of thThread 0 end
Thread 2 end
read 4 end
Thread 4 end
Test 2 passed
```

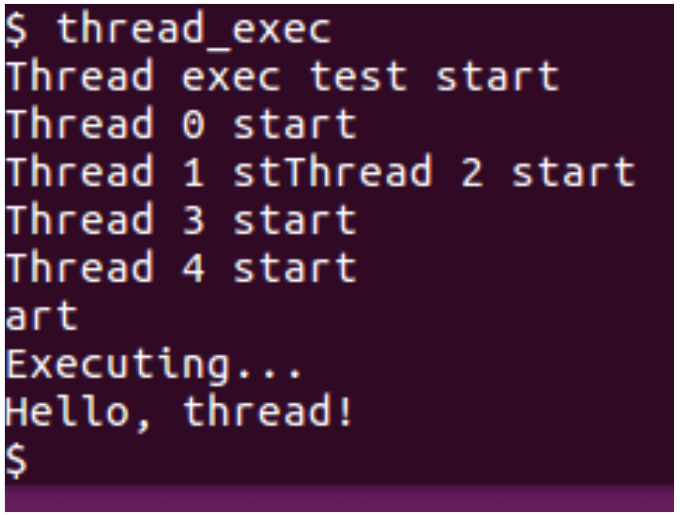
thread_test.c - Test 3

- 스레드에서 sbrk가 올바르게 동작하는지 평가합니다. malloc을 사용하면 내부적으로 sbrk가 호출됩니다.
- 한 스레드에서 메모리를 할당받은 뒤 다른 스레드들이 접근하는 데에 문제가 없는지 확인합니다.
- 여러 스레드들이 각자 메모리를 할당받아도 겹치는 주소에 중복 할당되지는 않았는지, 동시에 할당받고 해제해도 문제가 발생하지는 않는지를 확인합니다.



thread_exec.c

- 스레드에서 exec가 올바르게 동작하는지 평가합니다.
- 만들어지는 스레드 중 하나가 exec로 hello_thread 프로그램을 실행합니다.
- exec가 실행되는 순간 다른 스레드들은 모두 종료되어야 합니다. 종료되지 않고 남아있으면 에러 메시지가 출력됩니다.



thread_exit.c

- 스레드에서 exit이 올바르게 동작하는지 평가합니다.
- 하나의 스레드에서 exit이 호출되면 그 프로세스 내의 모든 스레드가 모두 종료되어야 합니다.
- Exiting... 출력 이후 바로 쉘로 빠져나가져야 합니다.

```
$ thread_exit
Thread exit test start
Thread 0 start
ThThread 2 start
Thread 3 start
Thread 4 start
read 1 start
Exiting...
$
```

thread_kill.c

- 프로세스가 kill 되었을 때 그 프로세스 내의 모든 스레드가 올바르게 종료되는지 확인합니다.
- 프로그램이 fork를 통해 두 개의 프로세스로 나누어진 뒤, 각각 5개씩의 스레드를 생성합니다. 그 중 부모 프로세스 쪽의 스레드 하나가 자식 프로세스를 kill 합니다.
- 부모 프로세스의 스레드는 kill에 영향을 받아서는 안되고, 자식 프로세스들의 스레드는 모두 즉시 종료되어야 합니다.
- 좀비 프로세스가 발생하면 안됩니다.

```
$ thread_kill
Thread kill test start
Killing process 4
This code should be executed 5 times.
This code should be executed 5 times.
This code should be executed 5 times.
This code should be executed 5 times.
This code should be executed 5 times.
Kill test finished
$
```

Thank you