Operating System Project #4

2017011885 김현기 2017011903 남윤재 2019040519 김태형

1. 스레드풀 알고리즘 구현

(1) pthread_pool_init

먼저 스레드 풀을 초기화 하기 위해 들어온 인자를 확인한다. bee_size가 POOL_MAXBSIZE 보다 크거나, bee_size가 0보다 작거나 같거나, queue_size가 POOL_MAXQSIZE보다 크거나, queue_size가 0보다 작거나 같으면 즉시 POOL_FAIL를 리턴한다. 문제 조건에 맞게 bee_size가 queue_size보다 작은 값이 들어온 경우에는 queue_size를 bee_size와 동일하게 초기화 해준다.

pool 구조체의 변수 running, q_front, q_len, bee_size 또한 각각 true, 0, bee_size로 초기화해주고, 대기열 q와 일꾼 스레드 bee를 동적할당 해준다. 마찬가지로 뮤텍스락과 조건변수 2개도 각각 초기화 해준다.

이후, 일꾼 스레드 worker를 bee_size만큼 생성해준다. 각 일꾼 스레드들에게는 구조체 pool을 인자로 전달한다. 스레드 생성 과정 중에 오류가 발생하면, pthead_pool_shutdown 함수를 실행해 만들어놓은 스레드 풀을 shutdown 시키고, POOL_FAIL을 리턴한다. 오류가 없는 경우, POOL_SUCCESS를 리턴한다.

(2) worker

worker함수는 pool 내부에 있는 lock과 task_t에 접근해야 하기 때문에 인자를 pool로 받아야한다. running 변수가 true일 동안 While 루프를 돌며 대기열에서 작업을 하나씩 가져와 실행한다.

먼저 mutex를 획득하고 대기열이 비어있을 경우, while문을 이용해 조건 변수 full안에서 새 작업이 들어올 때까지 기다린다. 대기가 종료되었다면 pool의 running이 FALSE로 바뀌었을 경우를 탐지해 루프를 종료해야 교착상태나 포인터 오류를 방지할 수 있으므로 running 조건을 재검사한다.

이후, 대기열에서 task_t 하나를 빼서 fnc에 실행할 함수와 인자를 저장하고, 대기열 실행 위치 q_front를 1칸 밀어주고, q_len을 1 감소시킨다. 이후, mutex를 해제한 뒤 대기열에 빈자리가 있음을 signal을 보내 알리고, 저장한 함수를 실행한다.

(3) pthread_pool_submit

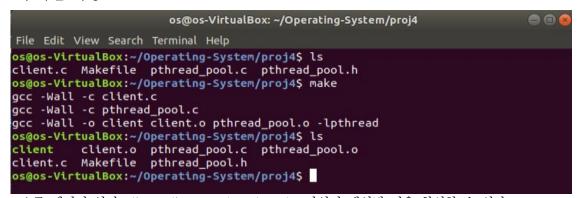
먼저 뮤텍스락을 획득하고, 대기열의 상태를 확인한다. 대기열이 가득 차있지 않다면 새 작업을 원형 대기큐에 넣고, 대기열의 길이를 하나 증가시킨다. 대기열이 가득 차 있고 flag값이 POOL_NOWAIT이라면, 뮤텍스락을 해제하고 즉시 POOL_FULL을 리턴한다. 대기열이 가득 차있고, flag값이 POOL_WAIT라면 while문을 사용해 대기열에 빈자리가 생길때까지 조건변수 empty에서 대기한다. 대기가 종료되고, running의 값이 바뀌어 false가 되었다면 스레드를 종료시킨다. 문제가 없다면 새 작업을 대기큐에 넣어준다. 이후 뮤텍스락을 풀고, 조건변수 full에 signal을 보내주고 함수를 종료한다.

(3) pthread_pool_shutdown

뮤텍스락을 걸고, pool 구조체의 running 값을 false로 바꾸어 실행중인 일꾼 스레드들이 자연스럽게 루프를 빠져나오도록 한다. pthread_cond_broadcast로 조건변수 empty와 full에서 기다리고 있는 일꾼 스레드들을 깨우고, pthread_join 함수로 모든 일꾼 스레드들을 종료시킨다.

이후, 동적할당 시켜주었던 일꾼 스레드 bee와 대기열 q를 free 함수로 메모리 해제시키고, 뮤텍스락과 조건변수 2개도 각각 destroy 해준다. 이후, POOL_SUCCESS를 리턴한다.

2. 컴파일 과정



오류 메시지 없이, client, client.o, pthread_pool.o 파일이 생성된 것을 확인할 수 있다.

3. 실행결과물 설명

샘플 출력결과와 동일하게 일꾼 스레드 최대 수 초과, 대기열 최대 용량 초과 메시지를 차례로 출력한다. 이후, pthread_pool_init() 완료 메시지, pthread_pool_shutdown() 완료 메시지를 출력하고, 대기열이 찼다는 Queue is full 메시지를 출력한다. 이후, () {} 〈〉 안의 숫자들과 . 을 랜덤하게 출력하고,PASSED 메시지를 출력하는 것을 확인할 수 있다.

이후 샘플 출력 결과와 동일하게 얼굴이 중간에 잘리지 않게 출력하는 것을 확인할 수 있다.

얼굴이 중간에 잘리지 않게 설계하는 것이 좋다.....PASSED

사람 얼굴 출력을 마치고, () {} <> 안의 숫자들과 . 을 출력하는 것을 확인할 수 있다.

1023까지 숫자들을 전부 출력한 뒤에,PASSED 메시지를 출력하고 프로그램이 종료된다.

4. 과제를 수행하면서 경험한 문제점과 느낀점

2017011885 김현기: 저번 3번째 과제에 이어서 이번 과제까지 하면서 두 과제가 비슷하다는 생 각이 많이 들었다. 과제를 수행하기 전에 pdf를 보고 전반적인 흐름을 이해하는데 시간이 오래걸 렸고, 이번 과제에서는 submit을 맡았는데 pdf에 함수의 조건에 대해 상세하게 나와있어 따라서 구현하면서 로직에는 어려움이 없었으나 C언어 포인터 같은 부분에서 조금 시간이 많이 들어갔 다. 언어 사용 능력을 보완해야겠다는 생각이 들었다.

2017011903 남윤재: 이번 과제에선 worker 함수의 작성을 맡았는데 함수의 인자가 void* 로 되 어 있어서 어떤 인자를 주어야할지 고민을 많이 했다. 처음엔 실행할 함수의 인자를 넣는 방식으 로 작성했었는데, worker함수에서도 pool의 lock이나 조건변수를 사용해야하는데 이걸 전역변수 로 처리해야하나 고민을 많이했다. 그러나 구현도 잘 안 될뿐더러 오류도 많이 생겨서 그냥 pool 의 주소값을 넣어주는건 어떨까라고 생각해서 구현해 보았는데 pool의 주소를 넣어주자마자 신기 하게 고민했던 것들이 모두 해결되었다. 이번 과제에선 맡은 역할이 조금이라 쓰레드 풀에대한 이해도가 높지 않아서 기회가 된다면 이번 과제말고 다른 방식의 쓰레드 풀을 구현해서 문제를 해결해 보고 싶다.

2019040519 김태형: init 함수와 shutdown 함수를 맡아 구현하면서, 수많은 segment fault(core dump) 포인터 오류를 경험하였다. 디버깅을 통해 대부분 구조체의 동적할당 과정 때문에 일어났 다는 것을 알게 되었다. 실행중인 스레드를 강제로 철회하는 경우 client.c 파일의 스레드풀에게 부하를 주는 코드를 실행하는 과정에서 프로그램이 멈추는 것을 여러 차례 경험하면서, 이번 과 제는 지금까지 배운 내용을 전부 응용해야하는 쉽지 않은 과제였다는 생각이 들었다.