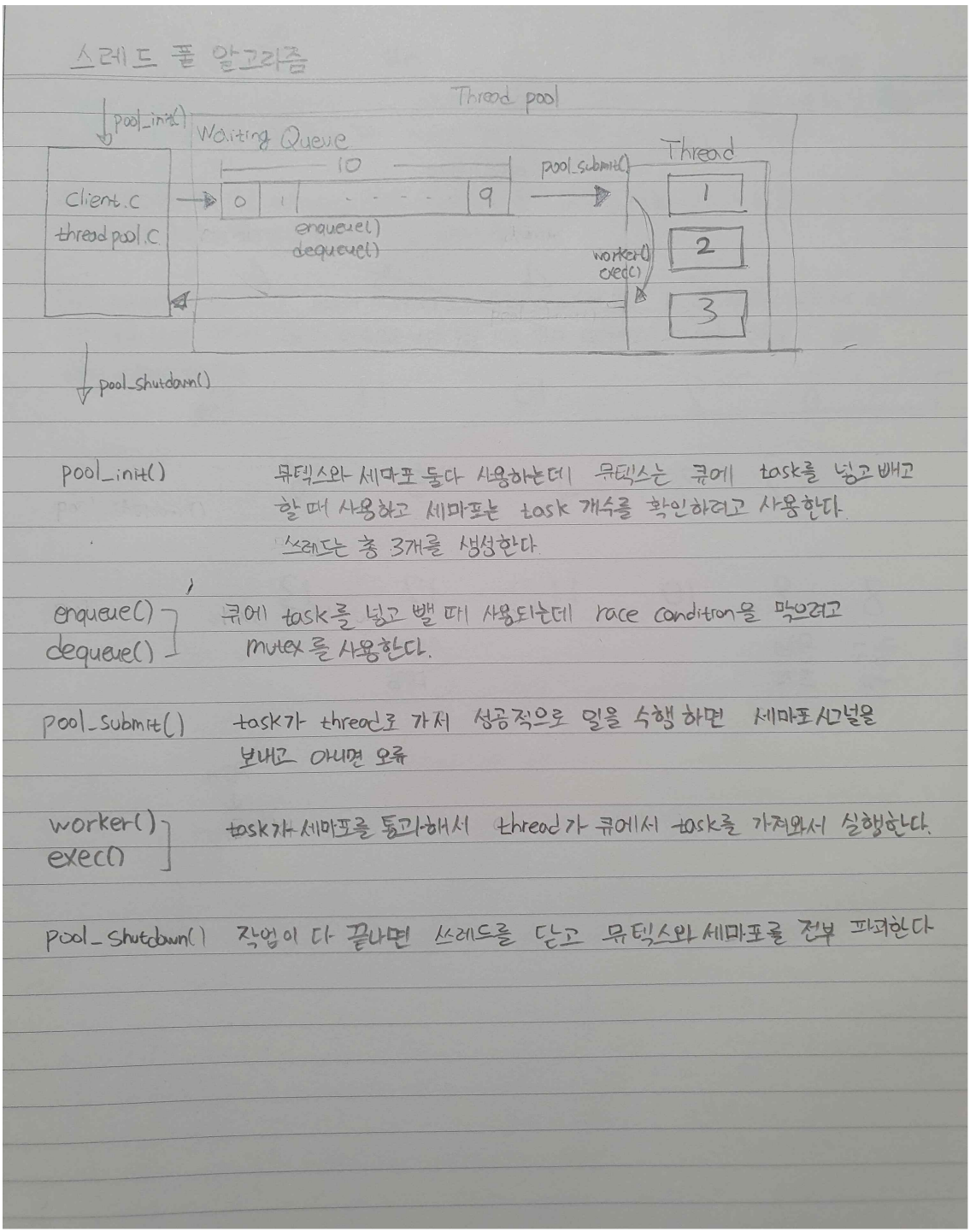
운영체제 프로젝트

2016003618 송현수

1. 알고리즘



2. 프로그램 소스파일

```c
/**
 * Implementation of thread pool.
 */
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
#include <semaphore.h>
#include "threadpool.h"
#define QUEUE_SIZE 10
#define NUMBER_OF_THREADS 3
#define TRUE 1
void exec(void (*somefunction)(void *p), void *p);
// this represents work that has to be
// completed by a thread in the pool
typedef struct
{
    void (*function)(void *p);
    void *data;
}
task;
// the worker bee
pthread_t bee[NUMBER_OF_THREADS];
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
sem_t tcount;
task list[QUEUE_SIZE + 1];
size_t liststart = 0, listend = 0;
// insert a task into the queue
// returns 0 if successful or 1 otherwise,
int enqueue(task t)
{
    pthread_mutex_lock(&mutex);
    if((listend + 1) % (QUEUE_SIZE + 1) == liststart)
    {
        pthread_mutex_unlock(&mutex);
        return 1;
    }
    list[listend] = t;
    listend = (listend + 1) % (QUEUE_SIZE + 1);
    pthread_mutex_unlock(&mutex);
    return 0;
}
// remove a task from the queue
task dequeue()
{
    pthread_mutex_lock(&mutex);
    task remove = list[liststart];
    liststart = (liststart + 1) % (QUEUE_SIZE + 1);
    pthread_mutex_unlock(&mutex);
    return remove;
}
// the worker thread in the thread pool
void *worker(void *param)
{
    // execute the task
```

```
    task worktodo;
    while(TRUE)
    {
        sem_wait(&tcount);
        worktodo = dequeue();
        exec(worktodo.function, worktodo.data);
    }
    pthread_exit(0);
}
void exec(void (*somefunction)(void *p), void *p)
{
    (*somefunction)(p);
}
/**
 * Submits work to the pool.
 */
int pool_submit(void (*somefunction)(void *p), void *p)
{
    int error = 0;
    task worktodo;
    worktodo.function = somefunction;
    worktodo.data = p;
    error = enqueue(worktodo);
    if(!error)
        sem_post(&tcount);
    return error;
}
// initialize the thread pool
void pool_init(void)
{
    int i = 0;
    sem_init(&tcount, 0, 0);
    while(i < NUMBER_OF_THREADS)
    {
        pthread_create(&bee[i], NULL, worker, NULL);
        i++;
    }
}
// shutdown the thread pool
void pool_shutdown(void)
{
    int k = 0;
    while(k < NUMBER_OF_THREADS)
        pthread_join(bee[k], NULL);
    sem_destroy(&tcount);
    pthread_mutex_destroy(&mutex);
}
```

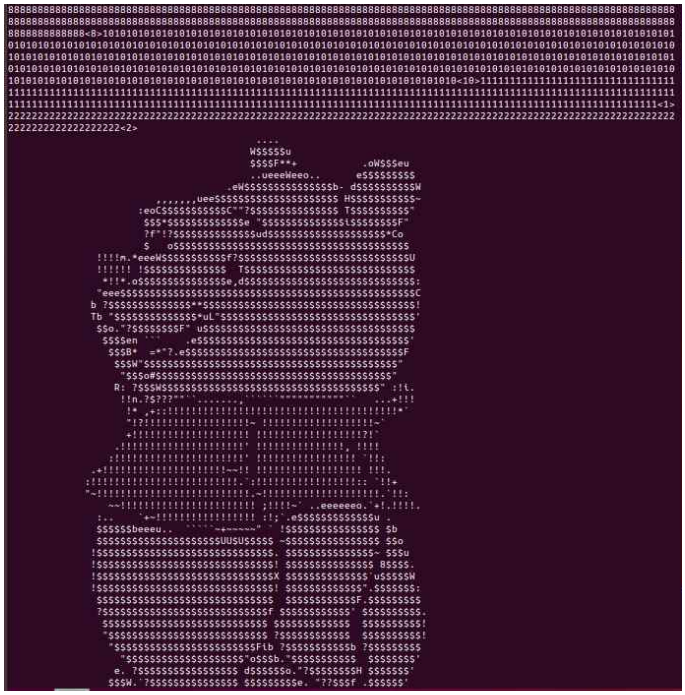
3. 컴파일

```
song@song-VirtualBox:~/Desktop/proj2$ make clean
rm -rf *.o
rm -rf example
song@song-VirtualBox:~/Desktop/proj2$ make
gcc -Wall -c client.c -lpthread
gcc -Wall -c threadpool.c -lpthread
gcc -Wall -o example client.o threadpool.o -lpthread
```

주어진 Makefile로 컴파일했다.

4. 실행 결과물의 주요 장면

```
song@song-VirtualBox:~/Desktop/proj2$ ./example
9: Queue is full.
11: Queue is full.
12: Queue is full.
13: Queue is full.
14: Queue is full.
15: Queue is full.
16: Queue is full.
17: Queue is full.
18: Queue is full.
19: Queue is full.
20: Queue is full.
21: Queue is full.
22: Queue is full.
23: Queue is full.
24: Queue is full.
25: Queue is full.
26: Queue is full.
27: Queue is full.
28: Queue is full.
29: Queue is full.
30: Queue is full.
31: Queue is full.
I add two values 5 and 10 result = 15
```



client.c에 있는 1st test, 2nd test, 3rd test 모두 잘 실행되었다.

Client.c 에는 총 3가지의 test case가 있다.

1. 구조체에 저장된 변수를 더하기
2. (NChar=)256 개의 문자를 출력한다. 총(NTASK=)32번 반복한다.
   쓰레드에 들어가지 못한 것은 에러 메세지 Queue is full 을 출력한다.
3. 곰돌이 푸 그림을 출력한다.

이 3가지 test case 모두 확인할 수 있다.