System Programming Lecture - Report 5

Student Number: 201520740

Name: LIN WEI

Exercise 1

Program

bank.h

```
#ifndef BANK_H
#define BANK_H

extern void* deposit(void *);
extern void* withdraw(void *);
extern int getBalance(void);

#endif
```

bank.c

```
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
#include "bank.h"

pthread_mutex_t balance_mutex = PTHREAD_MUTEX_INITIALIZER;

static int balance = 0;

/* This function adds a random number (1~100) of money to balance */
void* deposit (void *arg){
  pthread_mutex_lock(&balance_mutex);
  int n = rand() % 100 + 1;
  balance += n;
  printf("Have deposited %d in account.\n", n);
  pthread_mutex_unlock(&balance_mutex);
}

/* This function subtracts a random number (1~100) of money from balance */
void* withdraw (void *arg){
  pthread_mutex_lock(&balance_mutex);
  int n = rand() % 100 + 1;
  if(balance - n < 0)
    printf("Withdraw failed on %d...Insufficient balance.\n", n);
  else{
    balance -= n;
    printf("Have withdrawn %d from account.\n", n);
  }
  pthread_mutex_unlock(&balance_mutex);
}

/* This function returns balance of the account */
int getBalance (){
  int n;
  pthread_mutex_lock(&balance_mutex);
  n = balance;
  pthread_mutex_unlock(&balance_mutex);
  return n;
}
```

main.c

```
#include <stdio.h>
#include <pthread.h>
#include <time.h>
#include "bank.h"

#define THREAD 10
```

```
void init(void);

void main(){
  pthread_t save[THREAD], draw[THREAD];
  int i;

  init();

  for(i = 0; i < THREAD; i++){
    pthread_create(&save[i], NULL, deposit, NULL);
    pthread_create(&draw[i], NULL, withdraw, NULL);
  }

  for(i = 0; i < THREAD; i++){
    pthread_join(save[i], NULL);
    pthread_join(draw[i], NULL);
  }

  printf("Balance: %d\n", getBalance());

}

/* Initialize random number generator */
void init(){
  srand(time(NULL));
}
```

Result

```
lw@lw-VirtualBox:~/Documents/Report5/Exercise1$ gcc -pthread main.c bank.c
lw@lw-VirtualBox:~/Documents/Report5/Exercise1$ ./a.out
Have deposited 3 in account.
Withdraw failed on 92...Insufficient balance.
```

```
Have deposited 64 in account.
Withdraw failed on 72...Insufficient balance.
Have deposited 69 in account.
Have withdrawn 44 from account.
Have withdrawn 92 from account.
Have deposited 63 in account.
Have withdrawn 30 from account.
Have deposited 82 in account.
Have withdrawn 37 from account.
Have deposited 60 in account.
Have deposited 80 in account.
Have withdrawn 96 from account.
Have deposited 50 in account.
Have withdrawn 76 from account.
Have withdrawn 90 from account.
Have deposited 58 in account.
Have withdrawn 17 from account.
Have deposited 10 in account.
Balance: 57
lw@lw-VirtualBox:~/Documents/Report5/Exercise1$ ./a.out
Have deposited 20 in account.
Withdraw failed on 97...Insufficient balance.
Have deposited 46 in account.
Have withdrawn 46 from account.
Have deposited 16 in account.
Withdraw failed on 58...Insufficient balance.
Withdraw failed on 79...Insufficient balance.
Have deposited 86 in account.
Have withdrawn 24 from account.
Have deposited 24 in account.
Have withdrawn 29 from account.
Have deposited 6 in account.
Have deposited 15 in account.
Have withdrawn 4 from account.
Have deposited 91 in account.
Have withdrawn 7 from account.
Have withdrawn 61 from account.
Have deposited 69 in account.
Have withdrawn 33 from account.
Have deposited 72 in account.
Balance: 241
lw@lw-VirtualBox:~/Documents/Report5/Exercise1$ ./a.out
Have deposited 23 in account.
Have withdrawn 7 from account.
Have deposited 70 in account.
Have withdrawn 53 from account.
Have deposited 86 in account.
Have withdrawn 97 from account.
Withdraw failed on 84...Insufficient balance.
Have deposited 62 in account.
Withdraw failed on 95...Insufficient balance.
Have deposited 68 in account.
Have withdrawn 47 from account.
Have deposited 10 in account.
Have withdrawn 76 from account.
Have deposited 6 in account.
Withdraw failed on 100...Insufficient balance.
Have deposited 67 in account.
Have withdrawn 46 from account.
Have deposited 27 in account.
Withdraw failed on 95...Insufficient balance.
Have deposited 30 in account.
Balance: 123
```

Consideration on Exercise 1

- When I enlarge the number of threads, we can easily find that the deposit and withdraw thread are running not in sequence. But due to the mutex lock, the balance keeps the correct sum.
- The total times of depositing and withdrawing is the same of their thread number respectively.

Exercise 2

Program

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define QSIZE 10  /* Size of buffer */
#define EN_TIMES 30  /* Times of enqueueing */
#define DE_TIMES 10  /* Times of dequeueing */
#define EN_NUM 2    /* Number of enqueueing threads */
#define DE_NUM 6    /*  Number of dequeueing threads*/

typedef struct{
  pthread_mutex_t buf_lock;
  int start;
  int num_full;
  pthread_cond_t notfull;
  pthread_cond_t notempty;
  void *data[QSIZE];
}circ_buf_t;

circ_buf_t *cbt;

/* Put data into the circle buffer */
void put_cb_data(circ_buf_t *cbp, void *data){
  pthread_mutex_lock(&cbp->buf_lock);
  while(cbp->num_full == QSIZE){
    printf("The current buffer is full, waiting for dequeueing...\n");
    pthread_cond_wait(&cbp->notfull, &cbp->buf_lock);
  }
  cbp->data[(cbp->start + cbp->num_full) % QSIZE] == data;
  cbp->num_full++;

  /* Status output */
  printf("Put \n");
  if(cbp->num_full == QSIZE)
    printf("Full \n");

  pthread_cond_signal(&cbp->notempty);
  pthread_mutex_unlock(&cbp->buf_lock);
}

/* Get data from the circle buffer */
void *get_cb_data(circ_buf_t *cbp){
  void *data;

  pthread_mutex_lock(&cbp->buf_lock);
  while(cbp->num_full == 0){
    printf("The current buffer is empty, waiting for enqueueing...\n");
    pthread_cond_wait(&cbp->notempty, &cbp->buf_lock);
  }
  data = cbp->data[cbp->start];
  cbp->start = (cbp->start + 1) % QSIZE;
  cbp->num_full--;

  /* Status output */
  printf("Get \n");
  if(cbp->num_full == 0)
```

```
    printf("Empty \n");

  pthread_cond_signal(&cbp->notfull);
  pthread_mutex_unlock(&cbp->buf_lock);

  return(data);
}

/* Thread of enqueue */
void* enqueue(void *arg){
  int i;
  for(i = 0; i < EN_TIMES; i++)
    put_cb_data(cbt, arg);
}

/*Thread of dequeue */
void* dequeue(void *arg){
  int i;
  for(i = 0; i < DE_TIMES; i++)
    get_cb_data(cbt);
}

void main(){
  pthread_t enq[EN_NUM], deq[DE_NUM];
  int i;

  /* Initialization of the buffer */
  cbt = (circ_buf_t *)malloc(sizeof(circ_buf_t));
  pthread_mutex_init(&cbt->buf_lock, NULL);
  cbt->start = 0;
  cbt->num_full = 0;
  pthread_cond_init(&cbt->notfull, NULL);
  pthread_cond_init(&cbt->notempty, NULL);

  for(i = 0; i < EN_NUM; i++)
    pthread_create(&enq[i], NULL, enqueue, NULL);

  for(i = 0; i < DE_NUM; i++)
    pthread_create(&deq[i], NULL, dequeue, NULL);

  for(i = 0; i < EN_NUM; i++)
    pthread_join(enq[i], NULL);

  for(i = 0; i < DE_NUM; i++)
    pthread_join(deq[i], NULL);

}
```

Result

```
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
Put
Put
Put
Put
Put
Put
Put
Put
Put
Put
```

```
Full
The current buffer is full, waiting for dequeueing...
Get
Get
Get
Get
Get
Get
Get
Get
Get
Get
Empty
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
Put
Put
Put
Put
Put
Put
Put
Put
Put
Put
Full
The current buffer is full, waiting for dequeueing...
Get
Get
```

```
Get
Get
Get
Get
Get
Get
Get
Get
Empty
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
Put
Put
Put
Put
Put
Put
Put
Put
Put
Put
Full
Get
Get
Get
Get
Get
Get
Get
Get
```

```
Get
Get
Empty
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
Put
Put
Put
Put
Put
Put
Put
Put
Put
Put
Full
The current buffer is full, waiting for dequeueing...
Get
Get
Get
Get
Get
Get
Get
Get
Get
Get
Empty
The current buffer is empty, waiting for enqueueing...
The current buffer is empty, waiting for enqueueing...
Put
```

```
Put
Put
Put
Put
Put
Put
Put
Put
Put
Full
The current buffer is full, waiting for dequeueing...
Get
Get
Get
Get
Get
Get
Get
Get
Get
Get
Empty
The current buffer is empty, waiting for enqueueing...
Put
Put
Put
Put
Put
Put
Put
Put
Put
```

```
Put
Full
Get
Get
Get
Get
Get
Get
Get
Get
Get
Get
Empty
```

Consideration on Exercise 2

- The enqueue and dequeue threads are running in a random order. But due to the mutex lock and condition variables, the buffer is protected from enqueueing data to a full buffer or dequeueing data from an empty buffer.
- The total number of Put and Get is the same of the thread number respectively, and finally the buffer restores to empty.

Review of this lecture

In this lecture, I learned the basic of POSIX threads, Mutex Lock and Condition Variable. I have rarely thought about data race problems in my previous programming work, so this lecture really help me a lot in understanding this concept. But there are still some confusion not clarified during the class, such as the “detached thread”. Maybe you can provide some examples so we can understand it more intuitively. Anyway, thank you for this lecture.