Operating System HW2 **b03705002 林軒逸**

Q1.

Code:

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

int length;
int sum;
int minimum;
int maximum;
void *average(void *param) //find average(integer value)
{
    int*num= (int*)param;
    for(int i=1;i<=length;i++)
    {
        sum+=num[i];
    }
    pthread_exit(0);
    //Thread termination
}
void *find_minimum(void* param) //find minium
{
    int* num=(int*)param;
    minimum=num[1];
    for(int i=1;i<=length;i++)
    {
        if(num[i]<=minimum)
        {
            minimum=num[i];
        }
    }
    pthread_exit(0);
    //Thread termination
}
```

```c
void *find_maximum(void* param) //find max function
{
    int*num=(int*)param;
    maximum=num[1];
    for(int i=1;i<=length;i++)
    {
        if(num[i]>=maximum)
        {
            maximum=num[i];
        }
    }
    pthread_exit(0);
    //Thread termination
}
int main(int argc, char *argv[])
{
    if (argc < 2) //error handling
    {
        fprintf(stderr,"Please enter at least one number!\n");
        return -1;
    }
    length=argc-1;
    int numbers[argc];
    for(int i=1;i<=length;i++)
    {
        numbers[i]=atoi(argv[i]); //get the command line numbers
    }

    pthread_t tid; /* the thread identifier */
    pthread_attr_t attr; /* set of attributes for the thread */

    /* get the default attributes */
    pthread_attr_init(&attr);
    /* create three thread */
    //Use default thread attributes.
```

```
    pthread_create(&tid,&attr,average,numbers);
    pthread_create(&tid,&attr,find_minimum,numbers);
    pthread_create(&tid,&attr,find_maximum,numbers);
    //Create 3 separate threads, The parent thread waits for
child to complete.
    /* now wait for the thread to exit */
    pthread_join(tid,NULL);
    printf("Average:%d\n",sum/length);
    printf("Minimum:%d\n",minimum);
    printf("Maximum:%d\n",maximum);
}
```

Explain:

First, we define three functions: average, find_minimum, and find_maximum. The three functions are used to calculate the average, minimum, and maximum of the inputted array. One thing special to be mentioned is the parameter we passed into the function is “void”, and actually we want it as “int”. Thus, we set “numbers” to store the integer value of the parameter, and numbers[0] is undefined and we won’t use it. (since argv[0] isn’t what we want, we’ll explain it later.)

In the main function, we first do the error handling of the condition that the user doesn’t pass any numbers in the command line. After that, we read argv and turn it into an integer array(by atoi). Also, to be mentioned, we don't want argv[0] since it is not the number we want to calculate. Thus, our “numbers” array’s numbers[0] is undefined. Thus, my programs prevent accessing to the illegal space by starting every computation from numbers[1]. After changing argv to numbers, we create three threads, and do average, minimum, maximum independently. We wait the threads to compute, and output the result at last.

Execution Sample:

```
linxuanyideMacBook-Pro:OS_HW2_b03705002 StanleyLIn$ ./OS_HW2_Problem1 90 81 78 9
5 79 72 85
Average:82
Minimum:72
Maximum:95
linxuanyideMacBook-Pro:OS_HW2_b03705002 StanleyLIn$ gcc -o OS_HW2_Problem1 OS_HW
2_Problem1.c -lpthread
OS_HW2_Problem1.c:65:20: warning: implicit declaration of function 'atoi' is
      invalid in C99 [-Wimplicit-function-declaration]
        numbers[i]=atoi(argv[i]); //get the command line numbers
                   ^
1 warning generated.
linxuanyideMacBook-Pro:OS_HW2_b03705002 StanleyLIn$ ./OS_HW2_Problem1 90 81 78 9
5 79 72 85
The average value is 82
The minimum value is 72
The maximum value is 95
```

Q2.

Code:

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

int number;
int*fib_seq;
void *fib(void *param) {
    number=atoi(param);
    fib_seq= (int*)malloc(number*sizeof(int));
    fib_seq[0]=0;
    fib_seq[1]=1;
    for(int i=2;i<number;i++) //Generate fibonacci sequence
    {
        fib_seq[i]=fib_seq[i-1]+fib_seq[i-2];
    }
    pthread_exit(0);
    //Thread termination
}
int main(int argc, char *argv[])
{
    if (argc != 2) //error handling
    {
        fprintf(stderr,"Please enter only one number!\n");
```

```
        return -1;
    }

    pthread_t tid; /* the thread identifier */
    pthread_attr_t attr; /* set of attributes for the thread */

    /* get the default attributes */
    pthread_attr_init(&attr);
    /* create the thread */
    //Use default thread attributes.

    pthread_create(&tid,&attr,fib,argv[1]);
    //Create a separate thread The parent thread waits for
child to complete.
    /* now wait for the thread to exit */

    pthread_join(tid,NULL);
    for(int i=0;i<number;i++)
    {
        printf("%d ",fib_seq[i]); //print the fibonacci sequence
    }
}
```

Explain:

First, we crate a function fib to generate the Fibonacci sequence. One thing special to be mentioned is the parameter we passed into the function is “void”, and actually we want it as “int”. Thus, we set “number” to store the integer value of the parameter.

In the main function, we first do error handling of the condition that the user didn’t enter a number in the command line. After that, we crate a thread which does the Fibonacci generate business. Also, we pass argv[1] to the function as the length of Fibonacci sequence. After the thread finish executing, the parent thread output the Fibonacci sequence and terminates.

Execution sample:

```
linxuanyideMacBook-Pro:OS_HW2_b03705002 StanleyLIn$ gcc -o OS_HW2_Problem2 OS_HW
2_Problem2.c -lpthread
OS_HW2_Problem2.c:15:12: warning: implicit declaration of function 'atoi' is
      invalid in C99 [-Wimplicit-function-declaration]
    number=atoi(param);
           ^
OS_HW2_Problem2.c:16:20: warning: implicitly declaring library function 'malloc'
      with type 'void *(unsigned long)' [-Wimplicit-function-declaration]
    fib_seq= (int*)malloc(number*sizeof(int));
                   ^
OS_HW2_Problem2.c:16:20: note: include the header <stdlib.h> or explicitly
      provide a declaration for 'malloc'
2 warnings generated.
linxuanyideMacBook-Pro:OS_HW2_b03705002 StanleyLIn$ ./OS_HW2_Problem2 10
0 1 1 2 3 5 8 13 21 34 linxuanyideMacBook-Pro:OS_HW2_b03705002 StanleyLIn$
```