# 使用Git创建os实验项目与多线程编程

## git的使用

1.使用git绑定用户名与邮箱 alt text 2.生成ssh,便于github免密登录 alt text 3.github添加生成的ssh alt text 4.使用github创建os实验项目 alt text 5.克隆到本地并建立远程连接上传之前的项目
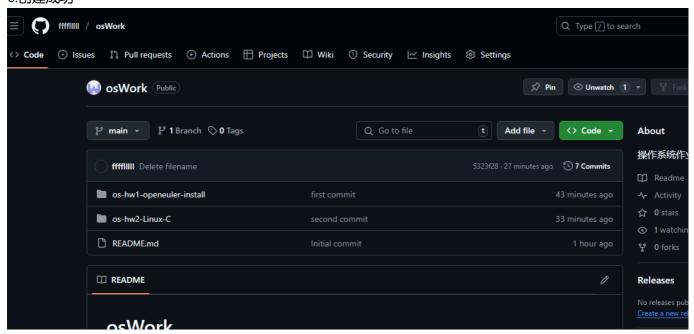
alt text

```
90840@flflflfl MINGW64 /e/HW_GIT/osWork (main)
$ git add .
warning: in the working copy of 'os-hw2-Linux-C/2.md', LF will be replaced by CRLF the next time Git touches it

90840@flflflfl MINGW64 /e/HW_GIT/osWork (main)
$ git commit
Aborting commit due to empty commit message.

90840@flflflfl MINGW64 /e/HW_GIT/osWork (main)
$ git push origin main
Everything up-to-date

90840@flflflfl MINGW64 /e/HW_GIT/osWork (main)
$ |
```

6.创建成功



## 多线程编程

1.基本多线程程序设计 编写如下代码使用 pthread 库创建一个线程并输出 "Ciallo world!"

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

void *worker(void *arg)
{
        printf("Ciallo world!\n");
        return NULL;
}

int main(int argc, char *argv[])
{
        pthread_t tid;

        if (pthread_create(&tid, NULL, worker, NULL))
        {
                printf("can not create\n");
                exit(1);
        }

        printf("main waiting for thread\n");

        pthread_join(tid, NULL);

        exit(0);
}
~
~
~
~
~
~
~
~
~
```

```
[root@localhost work3]# vim 1.c
[root@localhost work3]# ls
1.c
[root@localhost work3]# gcc 1.c -o 1 -pthread
[root@localhost work3]# ls
1  1.c
[root@localhost work3]# ./1
main waiting for thread
Ciallo world!
[root@localhost work3]# ~
```

## 2.多线程性能测试 编写如下代码进行多线程计算和单线程计算对比

```
[root@localhost work3]# cat 2.c
#include <stdio.h>
#include <pthread.h>
#include <time.h>
#include <stdint.h>

#define MAX_NUM 10000000
#define THREADS 2
typedef struct
{
        uint64_t start;
        uint64_t end;
        uint64_t result;
} ThreadArgs;

void *calculate_sum(void *arg)
{
        ThreadArgs *args = (ThreadArgs *)arg;
        uint64_t sum = 0;
        for (uint64_t i = args->start; i <= args->end; i++)
        {
                sum += i;
        }
        args->result = sum;
        return NULL;
}

uint64_t single_thread_sum()
{
        uint64_t sum = 0;
        for (uint64_t i = 1; i <= MAX_NUM; i++)
                {
                        sum += i;
                }
                return sum;
}

uint64_t multi_thread_sum()
{
```

```
        for (int i = 0; i < THREADS; i++)
        {
                args[i].start = i * segment + 1;
                args[i].end = (i == THREADS - 1) ? MAX_NUM : (i + 1) * segment;
                args[i].result = 0;
                pthread_create(&threads[i], NULL, calculate_sum, &args[i]);
        }

                                                        // 等待所有线程完成
        for (int i = 0; i < THREADS; i++)
        {
                pthread_join(threads[i], NULL);
                total_sum += args[i].result;
        }

        return total_sum;
}

int main()
{
        clock_t start, end;
        uint64_t sum;
        double time_used;
        start = clock();
        sum = single_thread_sum();
        end = clock();
        time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
        printf("单线程结果: %lu, 耗时: %.5f 秒\n", sum, time_used);

                // 多线程测试
        start = clock();
        sum = multi_thread_sum();
        end = clock();
        time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
        printf("多线程结果: %lu, 耗时: %.5f 秒\n", sum, time_used);

        return 0;
}
```

```
[root@localhost work3]# gcc 2.c -o 2 -pthread
[root@localhost work3]# ./ 2
-bash: ./: Is a directory
[root@localhost work3]# ./2
单线程结果: 50000005000000, 耗时: 0.00948 秒
多线程结果: 50000005000000, 耗时: 0.01112 秒
```

运行结果如下图