多线程代码

任务1：用部署好的openeuler环境，在命令行下编辑并编译

1.编写多线程示例代码(以五个线程为例)

```
mkdir -p src/pthread_example
vim src/pthread_example/thread_demo.c
```

代码内容：

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

#define NUM_THREADS 5  // 线程数量

/* 线程工作函数 */
void *thread_worker(void *arg) {
    int thread_id = *((int *)arg);
    printf("Thread %d: start (PID: %d)\n", thread_id, getpid());

    // 模拟工作（随机延时0-3秒）
    sleep(rand() % 4);

    printf("Thread %d: complete\n", thread_id);
    return NULL;
}

int main() {
    pthread_t threads[NUM_THREADS];
    int thread_args[NUM_THREADS];  // 每个线程的参数

    srand(time(NULL));  // 初始化随机数种子

    printf("Main Thread[PID: %d] is creating %d Sub Threads...\n", getpid(), NUM_THREADS);

    /* 创建多个线程 */
    for (int i = 0; i < NUM_THREADS; i++) {
        thread_args[i] = i + 1;  // 线程编号从1开始

        int ret = pthread_create(&threads[i], NULL,
                                 thread_worker, &thread_args[i]);
        if (ret != 0) {
            perror("Threads Creation Failed!");
            exit(EXIT_FAILURE);
        }
    }

    /* 等待所有线程完成 */
    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }
```

```
    printf("All Threads Completed! \n");
    return EXIT_SUCCESS;
}
```

代码截图：

2. 编译运行多线程程序

```
gcc src/pthread_example/thread_demo.c -o src/pthread_example/thread_demo -lpthread
./src/pthread_example/thread_demo
```

3. 运行结果截图

```
int main(){
        pthread_t threads[NUM_THREADS];
        int thread_args[NUM_THREADS];

        srand(time(NULL));

        printf("Main Thread[PID: %d] is creating %d Sub Threads...\n",getpid(),NUM_THREADS);

        for(int i=0;i<NUM_THREADS;i++){
                thread_args[i]=i+1;
                int ret=pthread_create(&threads[i],NULL,thread_worker,&thread_args[i]);
                if(ret != 0){
                        perror("Thread Creation Failed!");
                        exit(EXIT_FAILURE);
                }
        }

        for(int i=0;i<NUM_THREADS;i++){
        pthread_join(threads[i],NULL);
        }
        printf("ALL Thread Completed!");
        return EXIT_SUCCESS;
}
~
~
~
~
~
~
~
~
~
~
"src/pthread_example/thread_demo.c" 39L, 828B written
[root@vbox os-practice]# gcc src/pthread_example/thread_demo.c -o src/pthread_example/thread_demo -lpthread
[root@vbox os-practice]# ./src/pthread_example/thread_demo
Main Thread[PID: 1657] is creating 5 Sub Threads...
Thread 1 : start (PID:1657)
Thread 2 : start (PID:1657)
Thread 3 : start (PID:1657)
Thread 4 : start (PID:1657)
Thread 5 : start (PID:1657)
Thread 1: complete
Thread 3: complete
Thread 2: complete
Thread 4: complete
Thread 5: complete
ALL Thread Completed![root@vbox os-practice]#
```

任务2：用git创建并初始化自己的os实践项目，将每次HW实践和未来的上机都在项目目录下建立单独的子目录，并利用git工具管理好每一次的修改

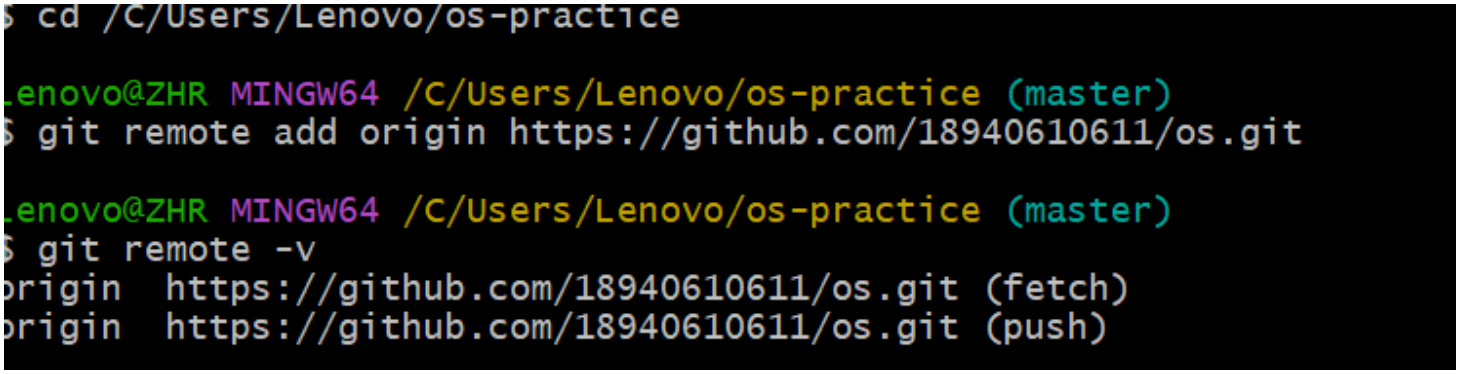
1. 创建项目目录结构

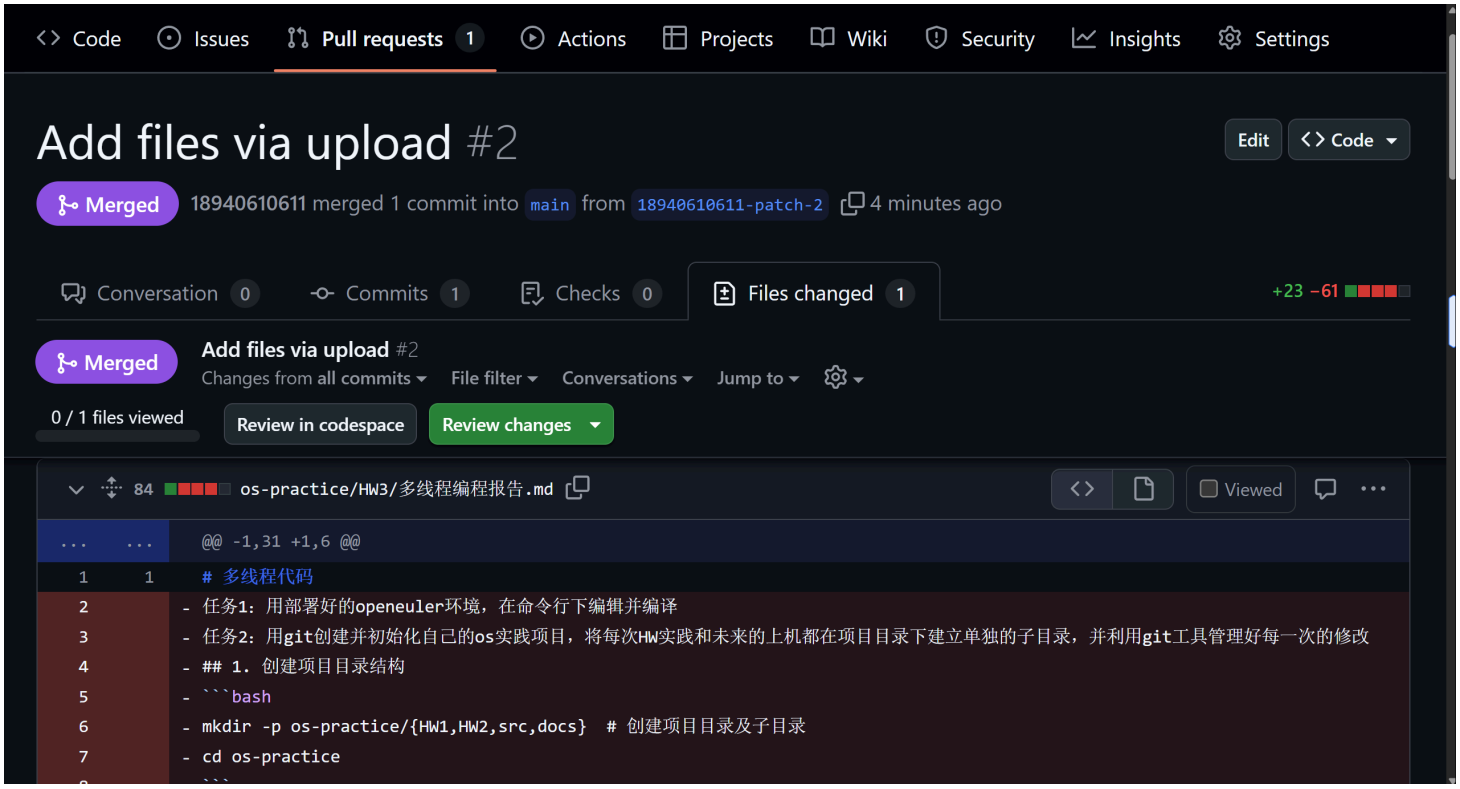
```
mkdir -p os-practice/{HW1,HW2,HW3}  # 创建项目目录及子目录
cd os-practice
```

2. 初始化Git仓库

```
git init
git add README.md
git commit -m "第一次提交"
git branch -M main
git remote add origin https://github.com/18940610611/os.git
 git push -u origin main
```

注：本人直接在github手动添加文件后在git bash上关联文件，HW3在github上创建补丁并拉取更新
附截图如下：





3.仓库网址:https://github.com/18940610611/os