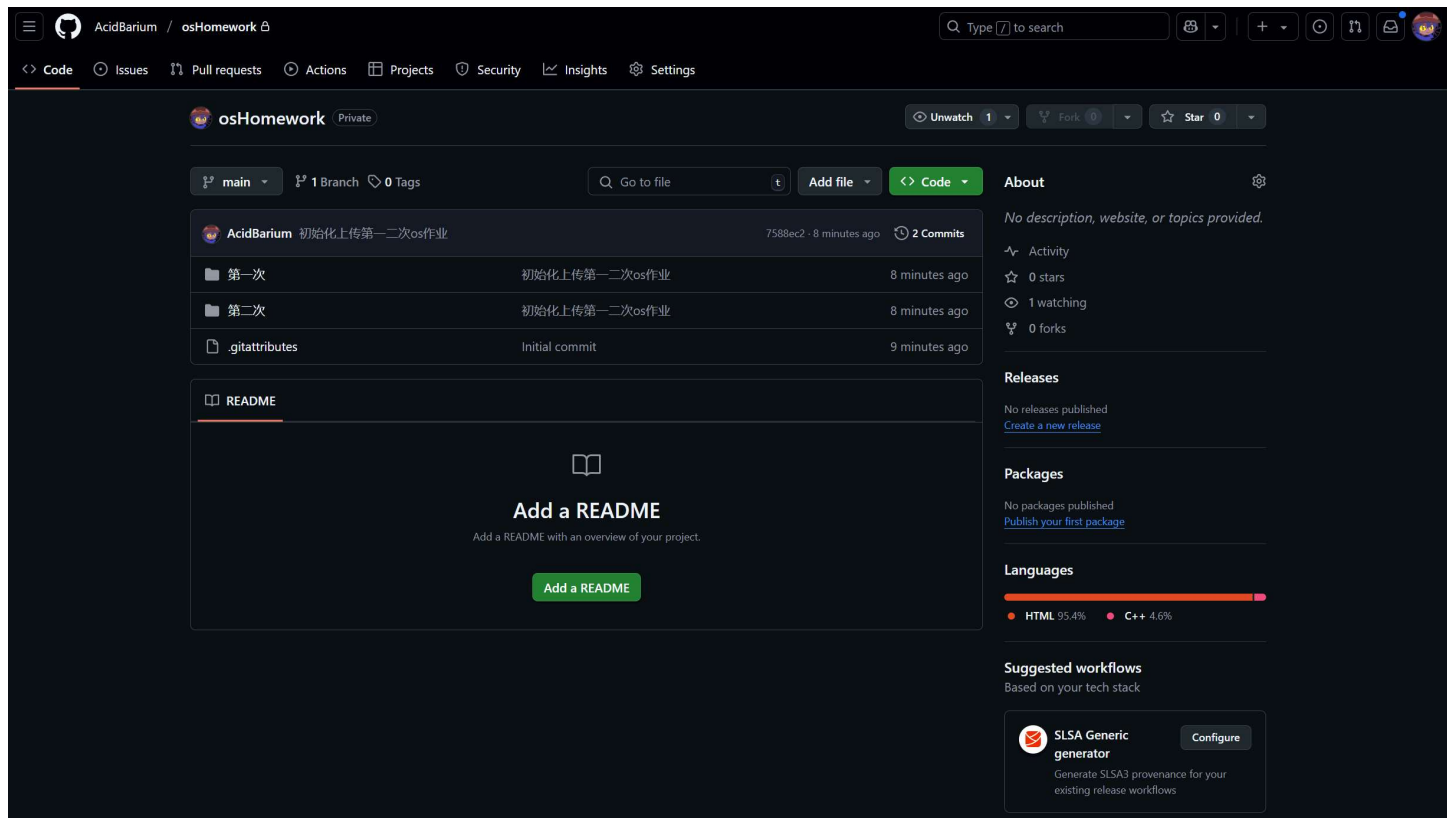


Git 使用与多线程编程实验报告

1. GitHub 仓库创建与管理

为了实现本地与虚拟机之间的协同开发，本实验首先创建了一个 GitHub 仓库，并将实验程序和记录上传至该仓库，便于在不同环境中同步和管理代码版本。

如图所示：



2. SSH 密钥配置与 GitHub 免密登录

使用下面的命令来生成ssh密钥，并将邮箱 acidbarium@163.com 作为密钥的注释，方便GitHub免密登录。

```
ssh-keygen -t ed25519 -C "acidbarium@163.com"
```

如图所示

```
[acidbarium@localhost ~]$ ssh-keygen -t ed25519 -C "acidbarium@163.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/acidbarium/.ssh/id_ed25519):
Created directory '/home/acidbarium/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/acidbarium/.ssh/id_ed25519
Your public key has been saved in /home/acidbarium/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:wsBVbP/GpTg7HtHYERtN0+KNLISyYr203MfXhzvfdRM "acidbarium@163.com"
The key's randomart image is:
```

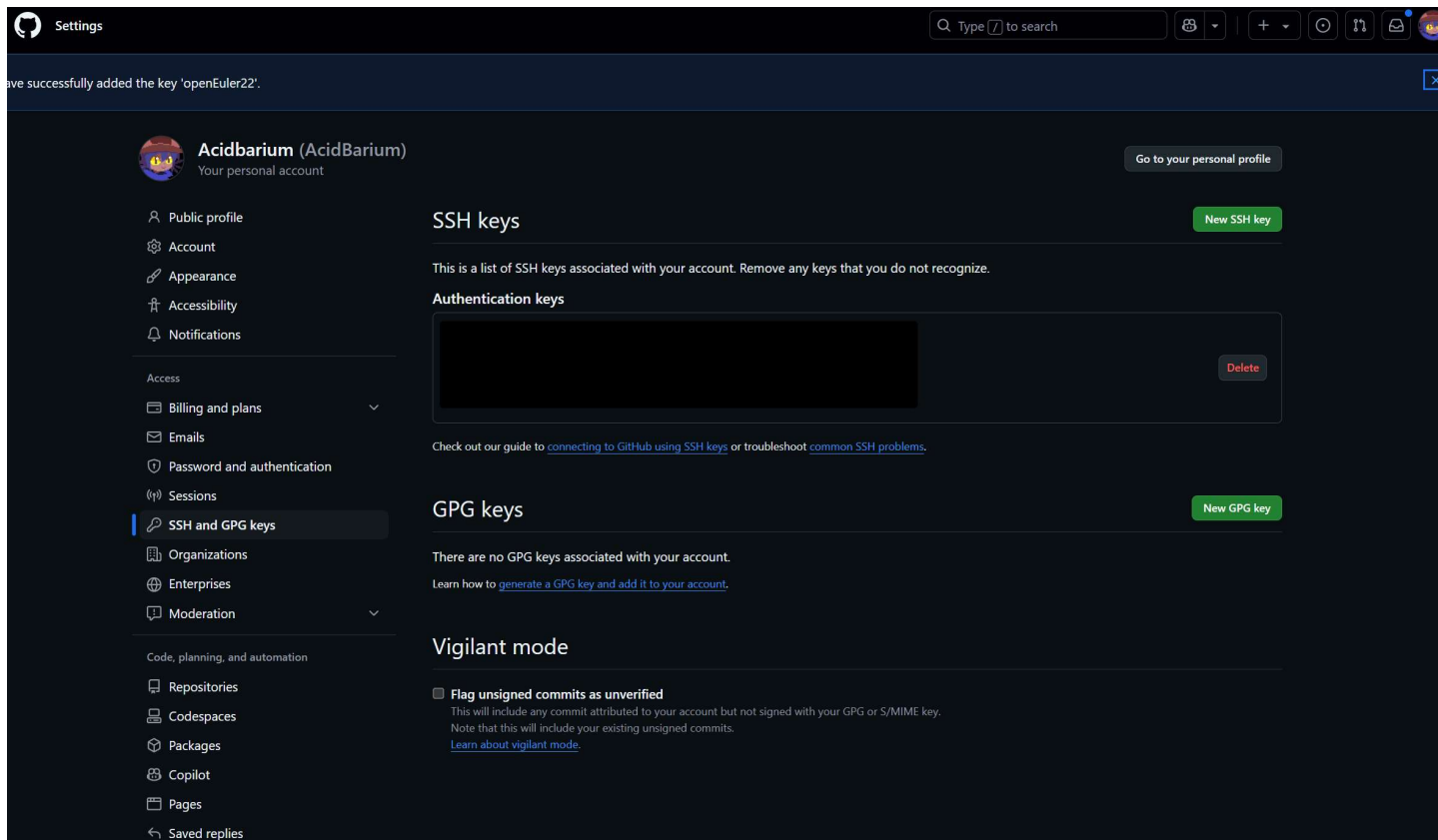
```
[acidbarium@localhost ~]$
```



使用下面的命令来查看SSH公钥

```
cat ~/.ssh/id_ed25519.pub
```

将该公钥添加至 GitHub 中的 SSH 密钥设置，如图所示：

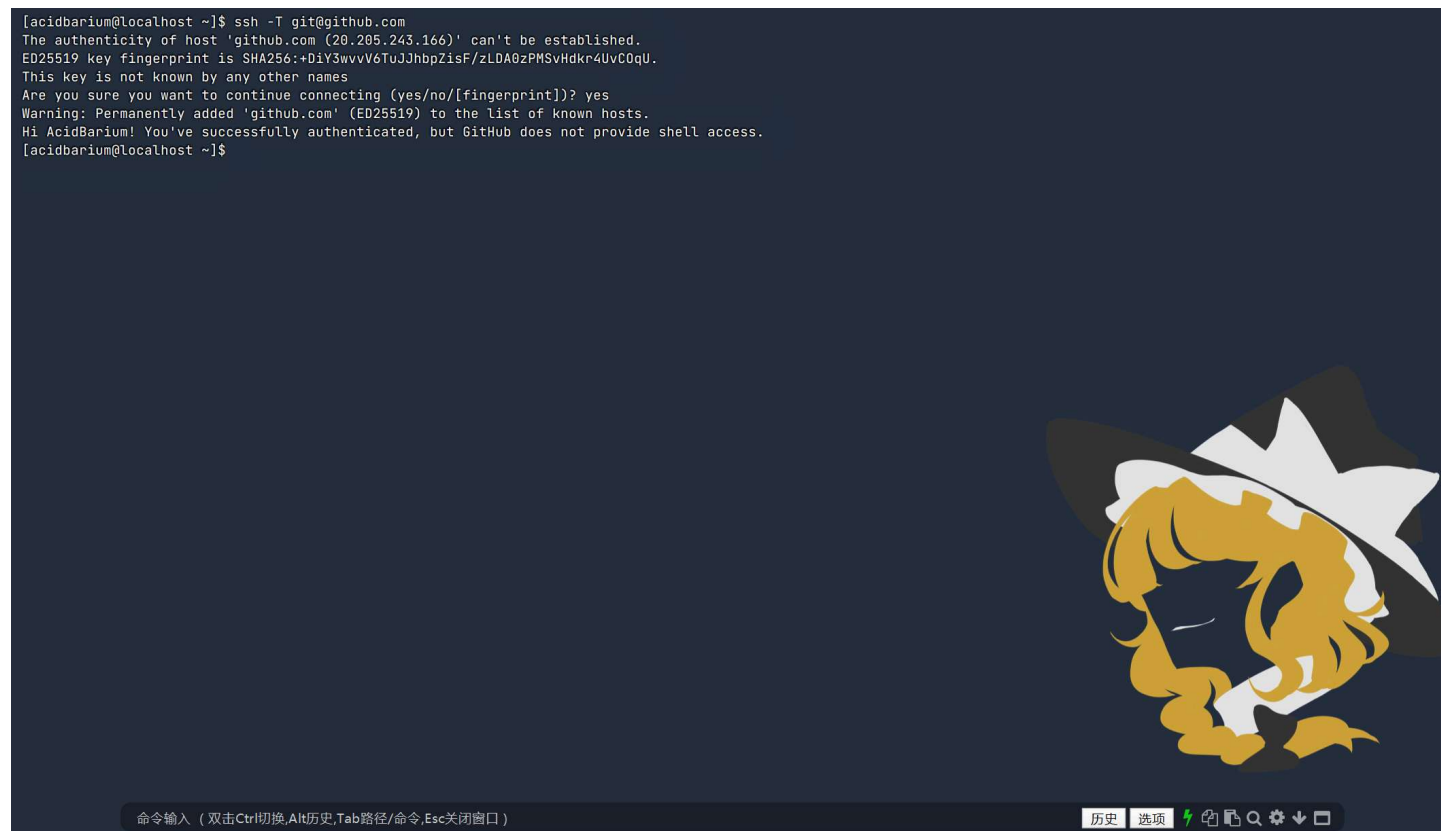


然后，使用以下命令来验证 SSH 密钥是否已正确配置并能成功连接 GitHub：

```
ssh -T git@github.com
```

如图所示，配置正确，显示成功消息：

```
[acidbarium@localhost ~]$ ssh -T git@github.com
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhp2isF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi AcidBarium! You've successfully authenticated, but GitHub does not provide shell access.
[acidbarium@localhost ~]$
```



3. Linux开发环境配置

为进行代码开发，首先在Linux上创建一个新的文件夹，如图所示

```
[acidbarium@localhost ~]$ pwd
/home/acidbarium
[acidbarium@localhost ~]$ ls
secondHomework test
[acidbarium@localhost ~]$ mkdir osHomework
[acidbarium@localhost ~]$ cd osHomework/
[acidbarium@localhost osHomework]$
```

通过下面的命令下载git

```
sudo yum install -y git
```

```
[acidbarium@localhost osHomework]$ sudo yum install -y git
```

通过 Git 克隆仓库到Linux:

```
git clone git@github.com:AcidBarium/osHomework.git
```

如图所示

```
[acidbarium@localhost osHomework]$ ls
[acidbarium@localhost osHomework]$ pwd
/home/acidbarium/osHomework
[acidbarium@localhost osHomework]$ git clone git@github.com:AcidBarium/osHomework.git
```

```
[acidbarium@localhost osHomework]$ ls
[acidbarium@localhost osHomework]$ pwd
/home/acidbarium/osHomework
[acidbarium@localhost osHomework]$ git clone git@github.com:AcidBarium/osHomework.git
正克隆到 'osHomework'...
remote: Enumerating objects: 97, done.
remote: Counting objects: 100% (97/97), done.
remote: Compressing objects: 100% (89/89), done.
remote: Total 97 (delta 9), reused 95 (delta 7), pack-reused 0 (from 0)
接收对象中: 100% (97/97), 15.07 MiB | 2.00 MiB/s, 完成.
处理 delta 中: 100% (9/9), 完成.
[acidbarium@localhost osHomework]$ ls
osHomework
[acidbarium@localhost osHomework]$ cd os
-bash: cd: os: No such file or directory
[acidbarium@localhost osHomework]$ cd osHomework/
[acidbarium@localhost osHomework]$ ls
第二次 第三次 第一次
[acidbarium@localhost osHomework]$
```



在仓库管理过程中, 使用以下命令同步更新代码并提交更改:

```
git pull origin main
git add .
git commit -m "提交信息"
git push origin main
```

4. 多线程编程实验

4.1 基本多线程程序设计

首先编写一个简单的多线程程序，该程序使用 `pthread` 库创建一个线程并输出 "Ciallo world!". 以下是代码示例：

```
#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;

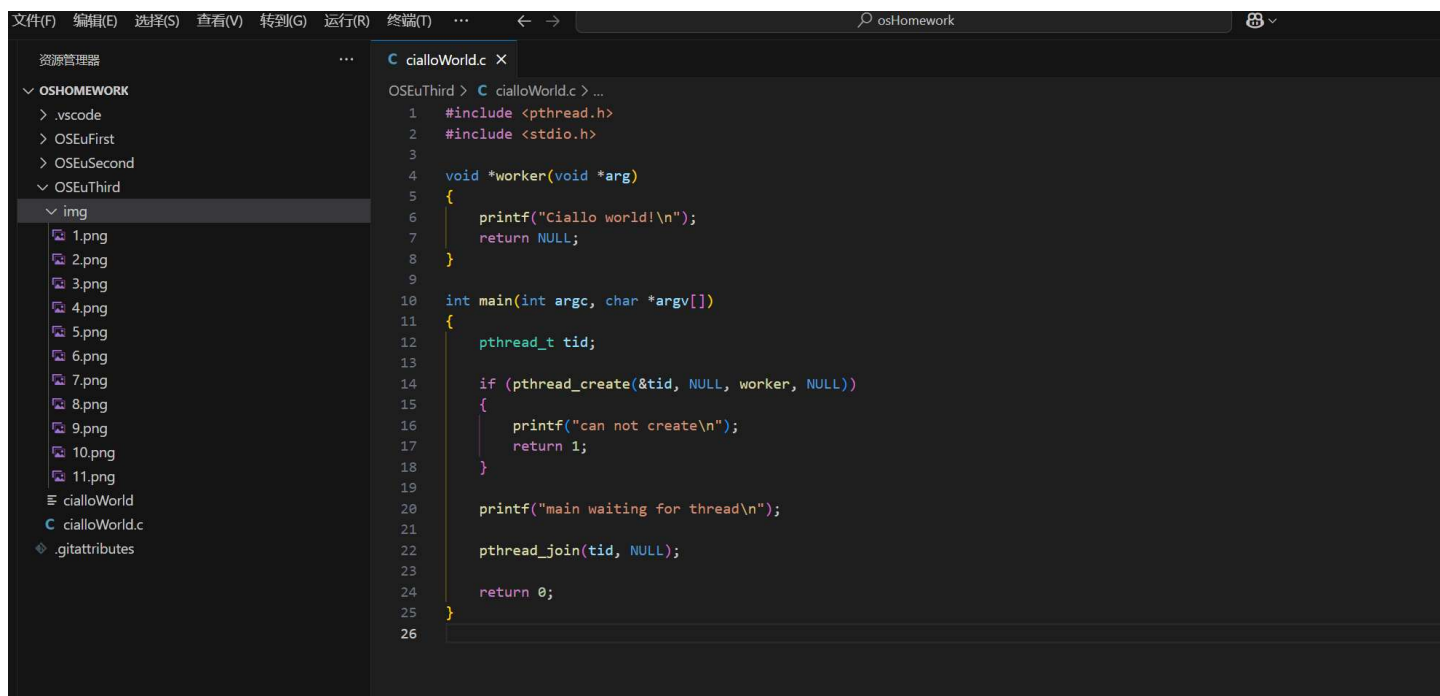
    if (pthread_create(&tid, NULL, worker, NULL))
    {
        printf("can not create\n");
        exit(1);
    }

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

    pthread_join(tid, NULL);

    exit(0);
}
```

如下图所示



```
文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) ... < -> osHomework
资源管理器
OSHOMEWORK
> .vscode
> OSEuFirst
> OSEuSecond
v OSEuThird
  img
    1.png
    2.png
    3.png
    4.png
    5.png
    6.png
    7.png
    8.png
    9.png
    10.png
    11.png
    cialloWorld
    cialloWorld.c
    .gitattributes
C cialloWorld.c X
OSEuThird > C cialloWorld.c > ...
1  #include <pthread.h>
2  #include <stdio.h>
3
4  void *worker(void *arg)
5  {
6      printf("Ciallo world!\n");
7      return NULL;
8  }
9
10 int main(int argc, char *argv[])
11 {
12     pthread_t tid;
13
14     if (pthread_create(&tid, NULL, worker, NULL))
15     {
16         printf("can not create\n");
17         return 1;
18     }
19
20     printf("main waiting for thread\n");
21
22     pthread_join(tid, NULL);
23
24     return 0;
25 }
26
```

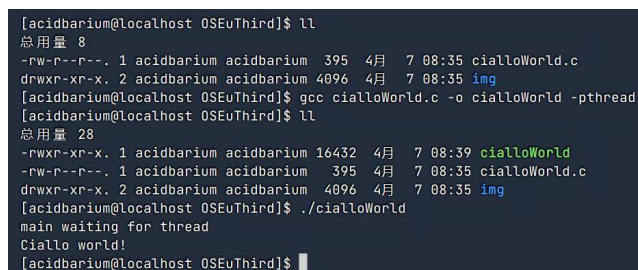
通过下面的指令编译

```
gcc cialloWorld.c -o cialloWorld -pthread
```

运行结果如下

```
main waiting for thread
Ciallo world!
```

如下图所示



```
[acidbarium@localhost OSEuThird]$ ll
总用量 8
-rw-r--r--. 1 acidbarium acidbarium 395  4月  7 08:35 cialloWorld.c
drwxr-xr-x. 2 acidbarium acidbarium 4096  4月  7 08:35 img
[acidbarium@localhost OSEuThird]$ gcc cialloWorld.c -o cialloWorld -pthread
[acidbarium@localhost OSEuThird]$ ll
总用量 28
-rwxr-xr-x. 1 acidbarium acidbarium 16432  4月  7 08:39 cialloWorld
-rw-r--r--. 1 acidbarium acidbarium 395  4月  7 08:35 cialloWorld.c
drwxr-xr-x. 2 acidbarium acidbarium 4096  4月  7 08:35 img
[acidbarium@localhost OSEuThird]$ ./cialloWorld
main waiting for thread
Ciallo world!
[acidbarium@localhost OSEuThird]$
```



4.2 多线程性能测试

为了测试多线程在大规模计算中的性能提升，编写了一个多线程计算和单线程计算对比的程序。该程序计算从 1 到 MAX_NUM（10000000）的整数之和，并使用2个线程分配计算任务，从而提高计算效率。

编写代码如下

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

#define MAX_NUM 10000000
#define THREADS 2

// 线程参数结构体
typedef struct
{
    uint64_t start;
    uint64_t end;
    uint64_t result;
} ThreadArgs;

// 线程函数: 计算 [start, end] 的和
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;
}

// 单线程计算
uint64_t single_thread_sum()
{
    uint64_t sum = 0;
    for (uint64_t i = 1; i <= MAX_NUM; i++)
    {
        sum += i;
    }
    return sum;
}

// 多线程计算
uint64_t multi_thread_sum()
{
    pthread_t threads[THREADS];
    ThreadArgs args[THREADS];
    uint64_t total_sum = 0;

```



```
// 分配计算范围
uint64_t segment = MAX_NUM / THREADS;
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;

    // 单线程测试
    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;
}
```

运行结果如下图所示

```
[acidbarium@localhost OSEuThird]$ git pull origin main
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0 (from 0)
展开对象中: 100% (4/4), 1.01 KiB | 1.01 MiB/s, 完成.
来自 github.com:AcidBarium/osHomework
* branch      main      -> FETCH_HEAD
a6f4823..c98bc8f main    -> origin/main
更新 a6f4823..c98bc8f
Fast-forward
 OSEuThird/timeCompare.c | 89 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 89 insertions(+)
 create mode 100644 OSEuThird/timeCompare.c
[acidbarium@localhost OSEuThird]$ ll
总用量 32
-rwxr-xr-x. 1 acidbarium acidbarium 16432  4月  7 08:39 cialloWorld
-rw-r--r--. 1 acidbarium acidbarium   395  4月  7 08:35 cialloWorld.c
drwxr-xr-x. 2 acidbarium acidbarium  4096  4月  7 08:42 img
-rw-r--r--. 1 acidbarium acidbarium  1871  4月  7 08:52 timeCompare.c
[acidbarium@localhost OSEuThird]$ gcc timeCompare.c -o timeCompare -pthread
[acidbarium@localhost OSEuThird]$ ll
总用量 52
-rwxr-xr-x. 1 acidbarium acidbarium 16432  4月  7 08:39 cialloWorld
-rw-r--r--. 1 acidbarium acidbarium   395  4月  7 08:35 cialloWorld.c
drwxr-xr-x. 2 acidbarium acidbarium  4096  4月  7 08:42 img
-rwxr-xr-x. 1 acidbarium acidbarium 16520  4月  7 08:54 timeCompare
-rw-r--r--. 1 acidbarium acidbarium  1871  4月  7 08:52 timeCompare.c
[acidbarium@localhost OSEuThird]$ ./timeCompare
单线程结果: 50000005000000, 耗时: 0.03150 秒
多线程结果: 50000005000000, 耗时: 0.01784 秒
[acidbarium@localhost OSEuThird]$
```



运行该程序后，可以观察到多线程计算的时间明显低于单线程计算，在多线程的支持下，程序的计算速度大约提高了一倍，证明了多线程在并行计算中的优势。