计算机网络实验-lab实验报告

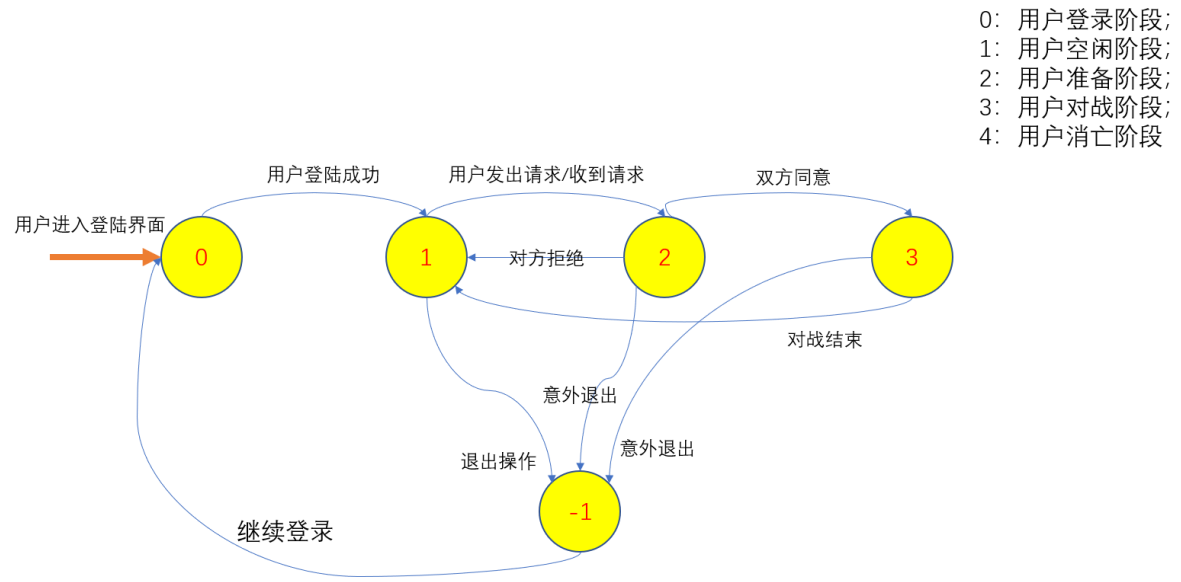
姓名：王少杰 学号：201220116

[代码对应的github]

一、用户端设计。

1、整体思路。

我是使用了有限状态机管理整个用户的行为。状态机表示如下：



2、设计框架

在设计客户端的代码的时候，我使用了三种技术，1、select多路复用技术；2、pthread多线程技术；3、线程锁同步技术。

大致的框架如下：

```
void * handle_write(void *)
{
    while(true)
    {
        try
        {
        write_to_client();//发送报文
        }
        catch()
        {
            //如果有什么异常，直接跳出while循环
            //如果不想等待，这里写continue
        }
        pthread_wait()//等待服务器传回的消息
    }
```

```
    pthread_exit(NULL);
}
int main //主要是读取服务器的消息
{
    int sockfd=socket();
    //绑定服务器的IP和端口号
    while(true)
    {
        //设置监听描述符
        select()
        if(FD_ISSET())//有描述符准备好了
        {
            try
            {
                read_from_cilent();//处理收到的信息
            }
            catch()
            {
                //处理错误消息
                //如果不想唤醒写线程，这里写continue
            }
            pthread_signal()//唤醒写线程
        }
    }
    pthread_join(tid,NULL);
    close(sockfd);
    return 0;
}
```

解释：

1、为了解决同时收发：

在主函数使用select函数被动等待服务器发来的报文，为了能够在接受报文的同时能够发送报文，所以我使用一个线程发报文，一个主线程收报文，所以使用了双线程。

2、为了解决一发一收问题：

而是用同步锁是因为我们协议设计的方式是一发一收的，就是客户端发出去一个报文，必须要等到客户端回传一个报文，我们才能继续发报文或者做其他的动作，所以如果上面的代码没有同步锁，就会导致一种现象，比如：客户端刚发一个查询哪个用户空闲的报文，这时正确的行为应是等待服务器回传相应的用户信息。但是while(true)会使得用重新进入写函数，用户仍然可以进行其他操作，比如发其他的报文等等，这样用户行为就会显得很乱。所以为了规范用户行为，形成一发一收的节奏，我们使用了同步锁。

当用户发完一个报文后，pthread_wait()使其等待被唤醒，等主线程收到报文并处理后，pthread_signal()唤醒写线程，使其能够继续其他行为。

3、为了解决有时不需要等待服务器报文的情况：

当你不需要等待服务器报文的时候（比如我使用了clear清屏命令），那就可以使用continue命令跳过pthread_wait()的执行。

二、服务器设计

结合本次实验的要求，服务器是不会主动给用户发送报文的，所以服务器是一个被动等待的状态。所以整体使用的是select函数。但为了同时位每一个用户提供服务，于是当每一次连接建立的时候，我们都为这一个用户建立一个thread帮其处理，直到用户退出连接，线程才消亡。

整体的设计框架如下：

```
void *handl_one_client()
{
    while(recv())
    {
        analyze_client_message();//分析对应的client报文
    }
    delete_client();//服务器删除对应用户记录
    broadcast_exit();//退出信息广播
    close()//关闭对应的socket
    pthread_exit(NULL);
}
int main()
{
    //创建sockfd、bind、listen等操作
    while(true)
    {
        select();
        if(FD_ISSET)
        {
            //accept获取新的套接字
            int newsockfd=accept();
            //创建线程为其处理
            thread tid;
            thread_create(tid,handl_one_client）;//起一个线程为其处理
            thread_detach(&tid);
        }
    }
    close()//关闭服务器套接字
    return 0;
}
```

analyze_client_message();函数就是按照协议手册里里面的操作码来执行相应的操作，操作全在server_show.cpp里面。