1. teacher端展示最多10个在线学生信息，可以自由选择并输入code码：

   对应代码修改：

   - **master端**：统计在线学生人数，发送至多10个学生的信息和对应code码给teacher端

```
void *teacher_work(void *arg) {
    DBG("Teacher on.\n");
    int ind = *(int *)arg;
    if (send(teacher_fd[ind], (void *)&student_cnt, sizeof(int), 0) <= 0) {
        DBG("Number of online students send failed.\n");
        return NULL;
    }
    int cnt = student_cnt;
    const int basenum = 10;
    cnt = cnt > basenum ? basenum : cnt;
    for (int i = 0; i < size && cnt > 0; i++) {
        if (student[i].flag == true) {
            if(send_studentinfo(ind, i) < 0) {
                DBG("Send Student's Information Error.\n");
                close(teacher_fd[ind]);
                return NULL;
            }
            //发送学生收到的code码，可以和姓名一起同teacher端联系起来
            if (send(teacher_fd[ind], (void *)&i, sizeof(int), 0) <= 0) {
                DBG("Code of online students send failed.\n");
                return NULL;
            }
            cnt--;
        }
    }
```

   - **teacher端**： 去掉程序启动时需要输入code码信息

```
 int main(int argc, char **argv) {
-    if (argc != 2) {
-        DBG("Useage:dohelp Help-Code\n");
-        exit(1);
-    }
```

   添加接收最多10个在线学生信息，运行过程中选择code码的逻辑

```
//Recv for Student's Information.
int online_cnt = 0;
if (recv(sockfd, (void *)&online_cnt, sizeof(int), 0) <= 0) {
    DBG("Number of online students recv failed.\n");
    perror("recv");
    return -1;
}
printf("Total number of online students : %d\n", online_cnt);
int num = 10;
num = num > online_cnt ? online_cnt : num;
struct Msg_t students[10];

while(num--) {
    struct Msg_t msg_t;
    if (recv(sockfd, (void *)&msg_t, sizeof(msg_t), 0) <= 0) {
        DBG("Recv Student's Information Error.\n");
        close(sockfd);
        exit(1);
    }
    int code = 0;
    if (recv(sockfd, (void *)&code, sizeof(int), 0) <= 0) {
        DBG("Code of online students recv failed.\n");
        perror("recv");
        return -1;
    }
    DBG("code : %d\n",code);
    students[code] = msg_t;
    printf("name : %s\nreal_name : %s\npath : %s\nport : %d\n", msg_t.name,msg_t.
    real_name,msg_t.path,msg_t.port);
    printf("=====================\n");
}
printf("Please choose one student and input his or her code\n");
```

2. 解决多线程引发master端给两个不同客户端发送同一个code码的问题

   对应代码修改:

   - master端:

     添加flag_using标记，在check_online之后置为false。避免master端分配给两个student端同一个i作为code码发送

```
for (int i = 0; i < size; i++) {
    if (student[i].flag == false && student[i].flag_using == false) {
        student[i].flag_using = true;
        sub = i;
        break;
    }
}
```

3. student端： 接收文件大小有时会多接收不知道哪里来的字节(知识盲区)

   对应代码修改:

   get_file函数中：直到收到预期大小的字节数

```
int test_size = 0;
while(test_size < (int)sizeof(uint64_t)) {
    test_size = recv(sockfd, (void *)&filesize, sizeof(uint64_t), 0);
    DBG("test_size : %d\n",test_size);
    DBG("filesize received : %d\n",filesize);
}
```

4.杂项修改:

- 4.1 student端：文件名字修改.id_rsa=>id_rsa
  否则open函数无法创建，报段错误

```
-    sprintf(key_file, "%s/.id_rsa", msg.path);
+    sprintf(key_file, "%s/id_rsa", msg.path);
```

- 4.2 master端：ctrl+c信号处理函数do_exit添加关掉tmux的逻辑

```
 void do_exit(int x) {
+    char test_str[100] = {0};
+        char cmd[100] = {0};
+        sprintf(test_str, "helper-haizei%d", code.code);
+        //退出时同时把tmux关掉
+        sprintf(cmd, "tmux kill-session -t %s", test_str);
+        printf("%s\n", cmd);
+        system(cmd);
     printf("SSH-Tunnel closed.\n");
     printf("Bye.\n");
     close(sockfd);
```

- 4.3 student端： 因为是同一个云主机模拟不同student，所以把tmux打开的session name由固定的改成了不固定的:helper-haize + code码的形式（因为不影响，所以没改回来）

```
+        char test_str[100] = {0};
+        char cmd[100] = {0};
+        sprintf(test_str, "helper-haizei%d", code.code);
```

- 4.4 因为同一个云主机测试模拟不同student，所以名字也改了（上传的代码中加了注释）后续同一个云主机测试可以去掉注释

  注释第一处:

  master端：

```
int send_studentinfo(int ind, int help_code) {
    struct Msg_t msg_t;
    //同一个云主机测试，student端发给我的是假名字，这里改成真名
    //strcpy(student[help_code].name, "wanglu");
    strcpy(msg_t.name, student[help_code].name);
```

  注释第两处:

  student端：

```
    //同一个云主机测试，改了个假名字
    // int fake_name = getpid();
    // sprintf(msg.name,"%d",fake_name);
    // printf("name : %s\n",msg.name);
    strcpy(msg.name, name);
```

- 4.5 student端：.install.sh

  sudo权限执行，whoami一定会得到root,导致脚本退出无法继续执行

```
-username=`whoami`
+username=`echo $(logname)`
```