

南 京 理 工 大 学

2020 课程设计报告

姓 名: 蒋旭钊 学 号: 918106840727

学院(系): 计算机科学与工程学院

专 业: 计算机科学与技术

课 程: UNIX 操作系统基础

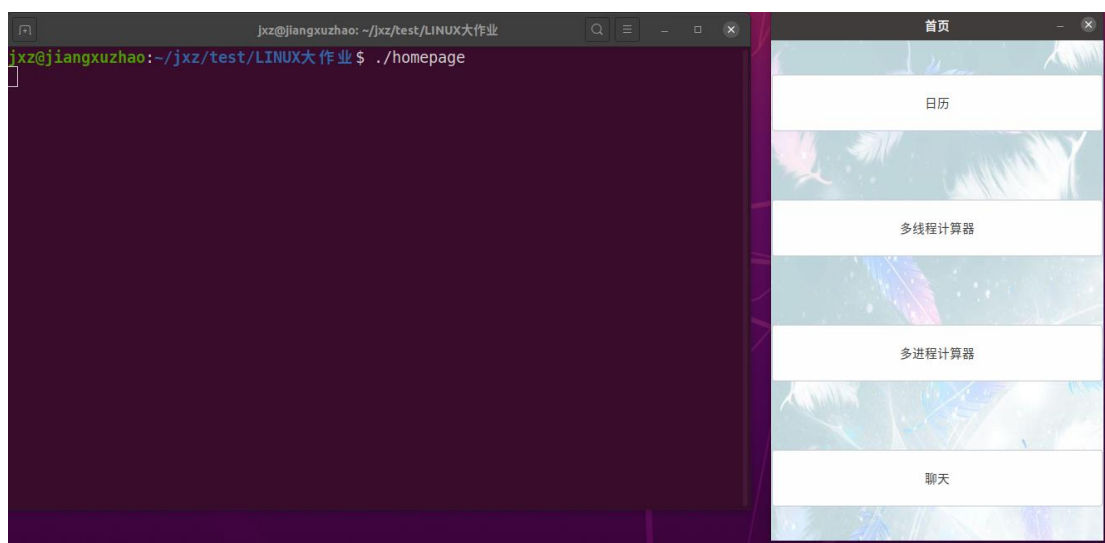
2020 年 12 月

基于 LINUX 的“多功能学生助手”开发

一、课程设计内容简介

基于网络编程 `socket` 的 C/S 模型，使用多线程和多进程分别开发了简单计算器；同时开发了多线程的聊天室功能，综合运用文件 I/O 操作，实现了私聊和群聊；运用 LINUX 下的 GTK 图形界面编程实现了日历查看功能；将这几者通过 GTK 图形界面编程整合实现了简单的学生助手功能。在助手开发的最后，由于自己对进程间通信的兴趣，又另外对进程间的四种通信方式进行了原理探讨和实例代码对比。

助手开发首页效果图如下：

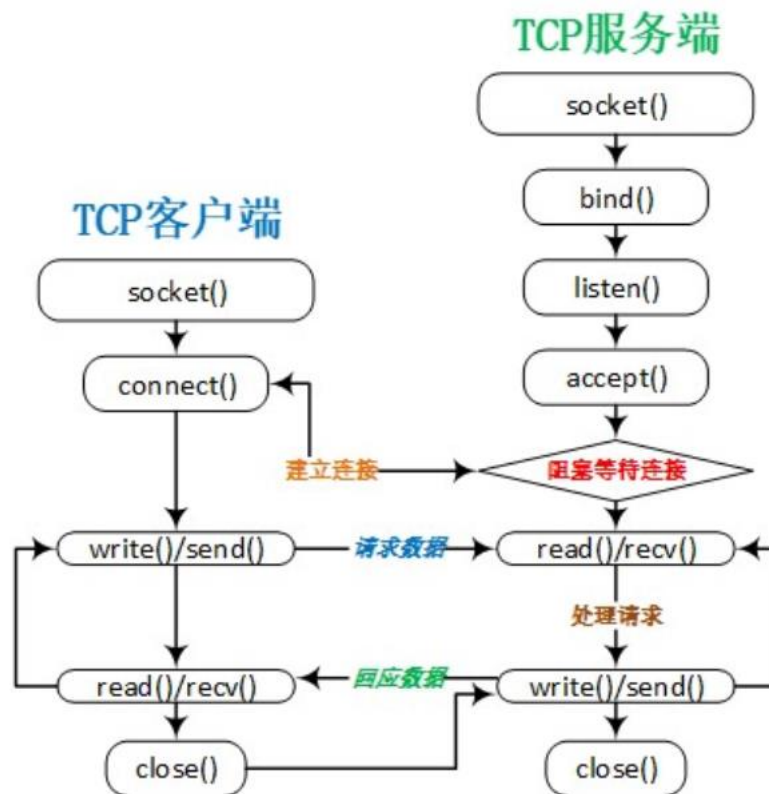


二、知识储备

1.网络编程：

网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个 `socket`。建立网络通信连接至少需要一对端口

号。



TCP 服务端的配置步骤:

Step1: 建立 Socket 对象 `Sk = socket.socket()`

Step2: 把一个地址族中的特定地址赋给 socket `Sk.bind((host,port))`

Step3: 开启监听进程等待客户端链接服务端 `Sk.listen(1)`

Step4: 接受客户端的申请，获取客户端的 IP 地址及其端口 PORT `clnt, addr = Sk.accept()`

Step5: 建立好连接之后，等待接受客户端的数据 `data = clnt.recv(1024)`

Step6: 应答客户机的请求，将对应的数据发回去 `clnt.sendall(data)`

Step7: 关闭 socket 对象 `Sk.close()`

TCP 客户端的配置步骤:

Step1: 建立 Socket 对象 `S = socket.socket()`

Step2: 把连接客户端到对应的服务端 `S.connect((host, port))`

Step3: 连接成功后向服务端发送数据 `S.sendall(cmd)`

Step4: 接等待服务端的回应数据 `data = S.recv(1024)`

Step5: 关闭 socket 对象 `Sk.close()`

三、各功能模块开发简介

1. 基于多线程的 socket 网络计算器开发:

在客户端, 基于 GTK 图形界面, 实现计算器基本图形页面的设计, 根据按钮绑定回调函数 `deal_num`。在 `deal_num` 函数中, 用户按下数字键则将数字字符存入发送缓冲区, 按下“C”则清空缓冲区, 如果按下“=”, 则在计算机客户端用 `socket()` 建立 socket 对象, 用 `bind()` 和指定服务器 IP 地址和端口 `port` 绑定, 并 `connect()` 连接到服务器, 将缓冲区中的算式内容通过 `write(client_socket, sendbuf, strlen(sendbuf))` 发送给服务端。

在服务器端, 首先定义服务器运作标志位 `g_run = 1`, 线程池运作标志位 `g_pool_run = 1`, 队列中任务数 `g_send = 0`, 并初始化互斥量 `g_mutex` 和条件变量 `g_cond`。

用 `socket()` 建立 socket 对象, 用 `bind()` 和任意客户端的 IP 地址和端口 `port` 绑定, 开启监听进程 `listen()` 等待客户端链接服务端。然后设定线程池大小, 并用 `pthread_create()` 创建线程池, 若创建

失败，则用 `pthread_join()` 收尸函数将已经创建的线程回收。

在线程创建的入口函数 `thr_routine()` 中，在 `while(1)` 循环中定义一个 `while` 循环，循环条件是 `g_pool_run == 1 && g_send == 0`，执行语句是 `pthread_cond_wait(&g_cond, &g_mutex)`，线程池中的所有线程都阻塞在这里。定义任务分发函数 `task_dispatch()`，当接收到客户端连接 `accept()` 时，调用 `task_dispatch()`，在互斥锁的队列中的任务数 `g_send+1`，并且广播 `g_cond` 信号，某个线程接收到后能跳出 `pthread_cond_wait` 阻塞，并且仍然持有锁，在锁中让 `g_cond-1` 使得其他线程仍然阻塞在 `pthread_cond_wait` 处，但是该线程本身去执行下面的计算器运算操作，执行完后通过 `write(connfd, sendbuf, strlen(sendbuf))` 将结果发回给客户端。

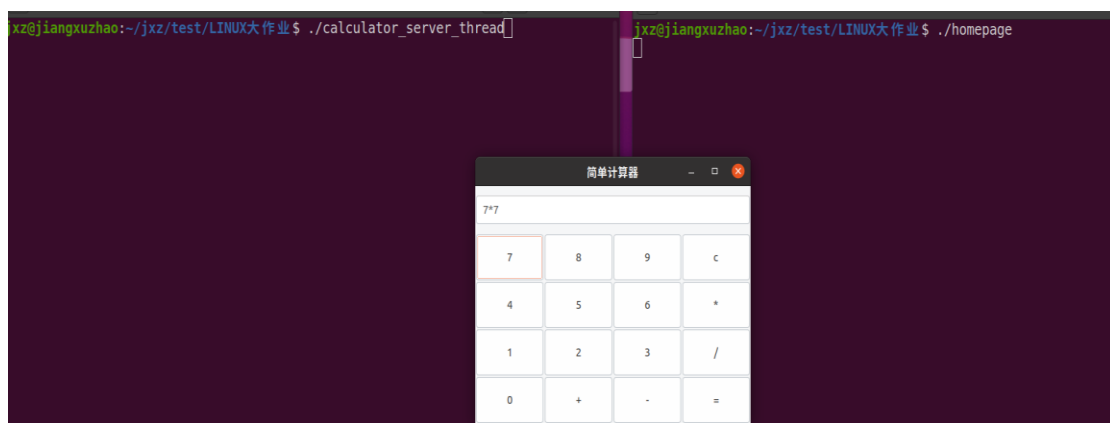
客户端通过 `read(client_socket, recvbuf, BUFFER_SIZE)` 接收到服务器处理后的数据，并将其显示在图形界面中。

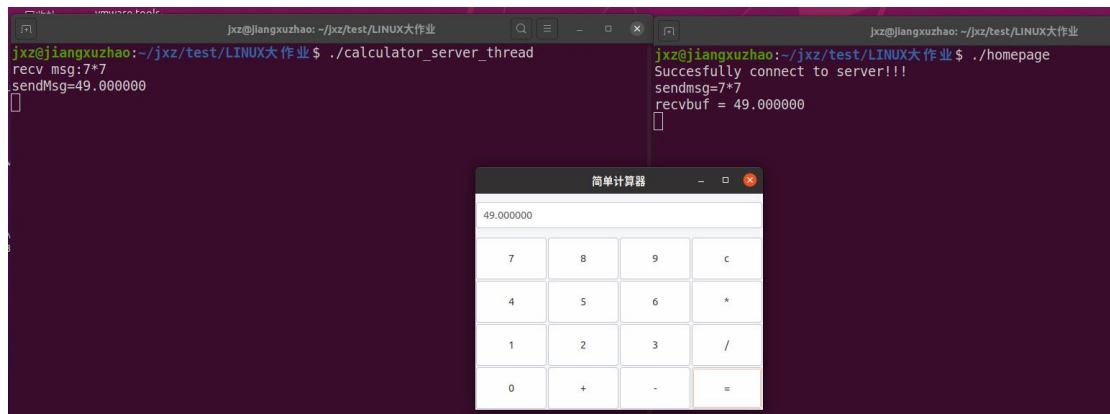
执行代码：

```
./calculator_server_thread
```

```
./homepage
```

效果图如下：





2. 基于多进程的 socket 网络计算器开发:

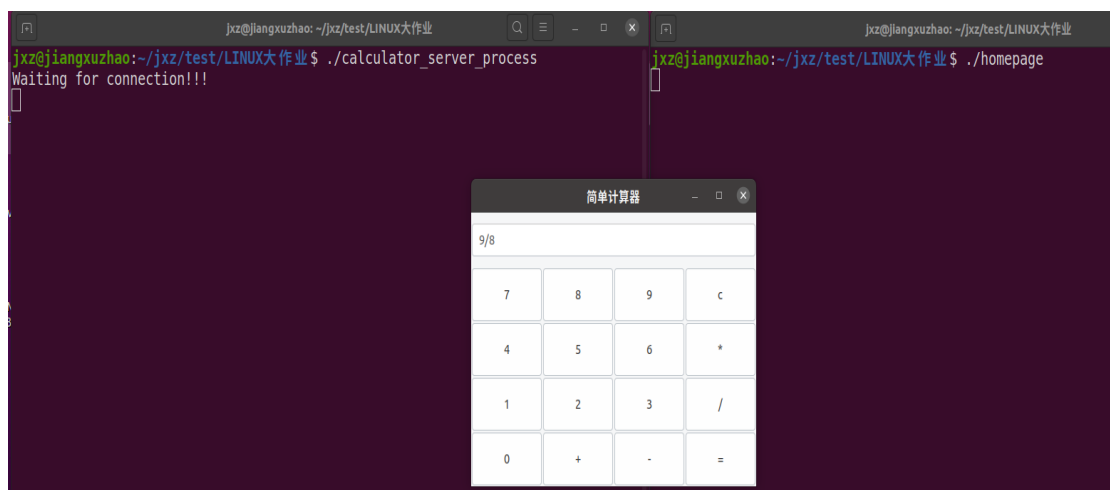
多进程的计算器客户端与多线程并无两异，在服务器的网络编程框架上也是一致的，只是在 `accept()` 执行并返回 `connfd` 文件操作符后，服务器会通过 `fork()` 创建一个子进程，父进程负责监听，处理时关闭父进程的 `connfd`；子进程负责处理，关闭子进程的监听描述符 `close(server_sockfd)`，然后进行计算器运算处理并将处理后的结果通过 `write(connfd, sendbuf, strlen(sendbuf))` 发回给客户端显示。

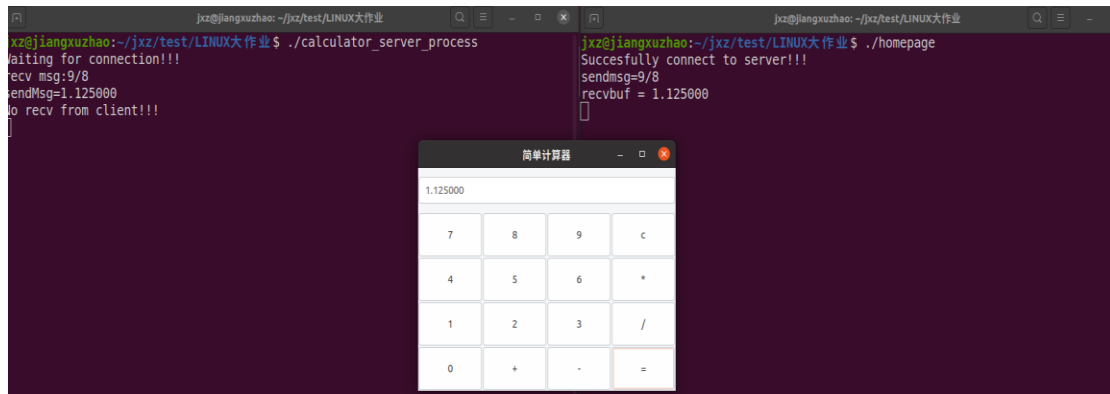
执行代码:

`./calculator_server_process`

`./homepage`

效果图如下:





3. 基于多线程和文件 I/O 的聊天室开发:

功能分为三部分, 客户端登录模块, 服务端转发客户端消息模块, 客户端私聊模块。

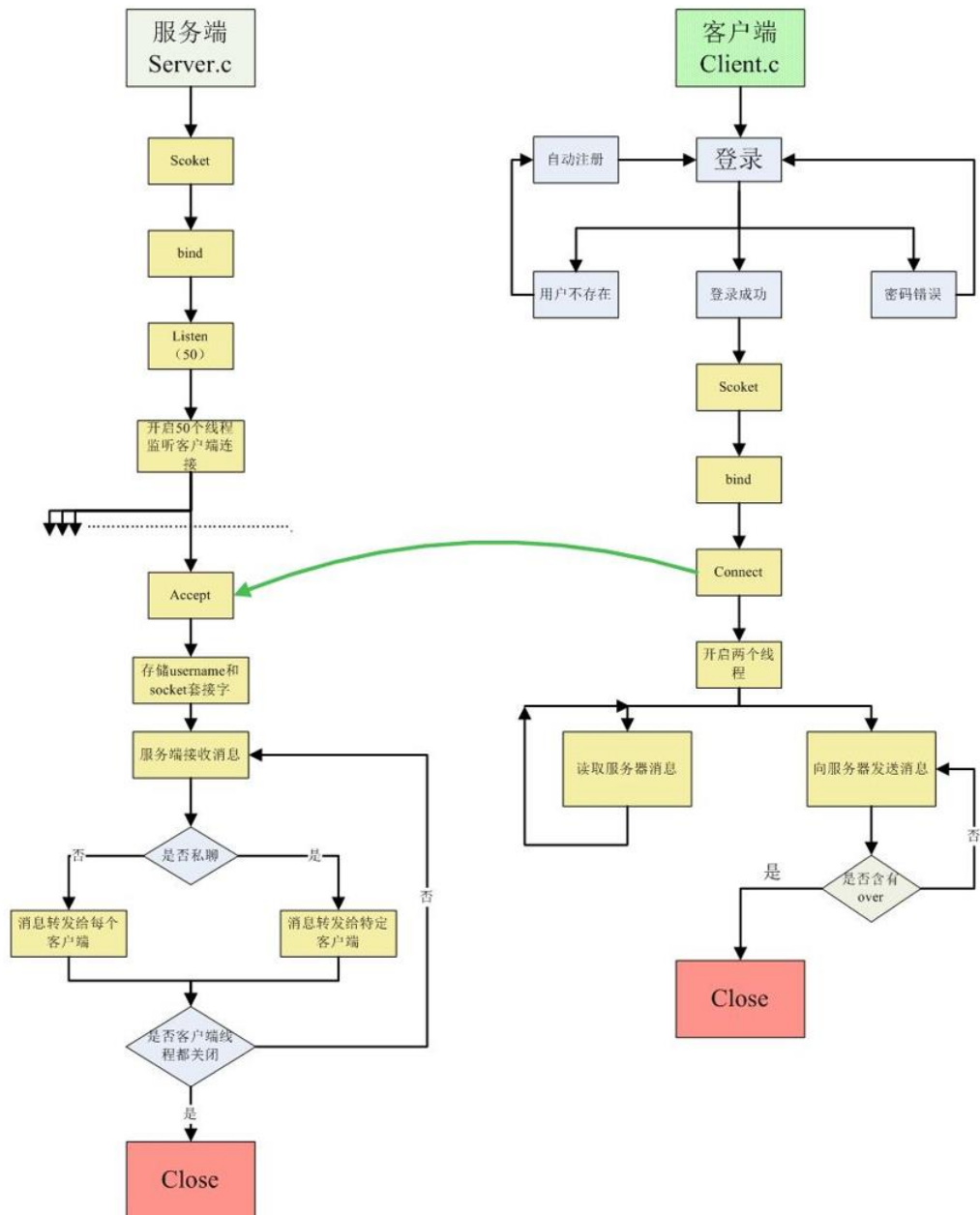
客户端登录模块: 调用 login.c 里的函数, 通过文件 I/O 操作, 将用户名和密码存储到文件 login.txt 中去, 如果用户名不存在则进行注册操作 regist, 用户名存在、密码错误则登录失败提示密码错误, 用户名存在、密码正确则登陆成功, 进行和服务端连接操作。客户端登录模块使用 open ()、read ()、write ()、strtok ()、malloc ()、strcpy ()、strcat ()、remove () 等函数。实现原理为: 将 username 和 password 两个参数进行字符串拼接成 username-password\n, 然后依次存入 login.txt 文件, 取出数据时, 通过 split () 方法对字符串进行两次字符分割, 分别是 “\n” 和 “-” 分割, 迭代确认 key 值, 返回 value 值, 从而达到通过 key 取 value 的功能。通过这两个功能, 实现了类似于 map 的 get 函数和 set 函数, 从而实现登录、注册, 验证身份的功能。

服务端转发客户端消息模块: 客户端和服务端进行连接, 服务端在 accep () 过程时开启 THREAD_POOL_SIZE 个线程, 监听

THREAD_POOL_SIZE 个客户端的连接请求。客户端连接成功后，将客户端 socket 套接字存入服务端事先准备好的 ClientSocket[THREAD_POOL_SIZE] 数组中，服务端接收到某一个客户端的消息后，通过遍历 ClientSocket[THREAD_POOL_SIZE]，将消息转发给其他客户端从而实现服务端转发客户端消息的功能。服务端转发客户端消息模块使用的函数有：socket ()、bind ()、listen ()、connect ()、accept ()、recv ()、send ()、pthread_create ()、memset ()、pthread_exit ()、close () 等函数。

客户端私聊模块：客户端在连接成功之后，将用户名发送给服务端，服务端将客户端用户名和相对应的 socket 套接字以 map 的形式（客户端登录模块使用的 get、set 方法）存入 login.txt 文件，其他客户端发消息进行了@用户名之后，服务端从 login.txt 获取用户名对应的 socket 套接字，然后发送消息，从而达到私聊功能，用户能够通过输入“over”退出聊天室。客户端私聊模块使用的函数有：open ()、read ()、write ()、strtok ()、malloc ()、strcpy ()、strcat ()、remove ()、strstr ()、strcmp ()、memset、strcat () 等函数。

这其中还运用了文件 I/O 操作的上锁来保存用户的消息记录到 logMessage.txt 文件中。课本上学到的 fcntl () 能够改变文件的属性，这里运用它的 fcntl (fd, F_GETLK, &lock) 来设置记录锁，防止在写入时受到其他线程的影响。



执行代码：登录注册, 登录三个用户连接聊天室

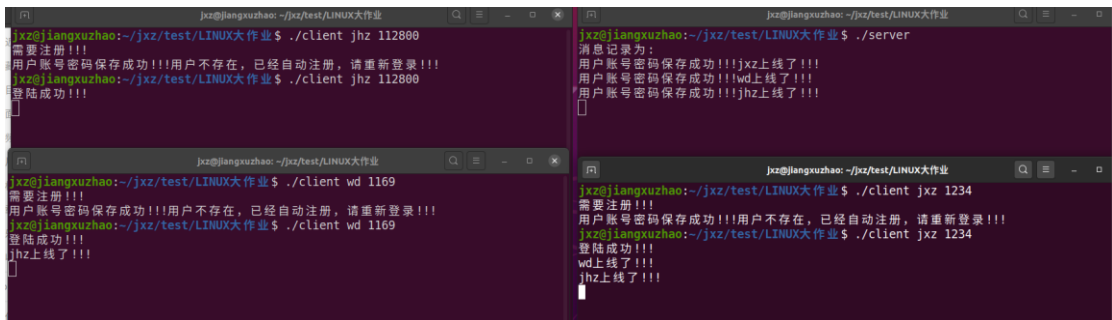
`./server`

`./client jxz 1234`

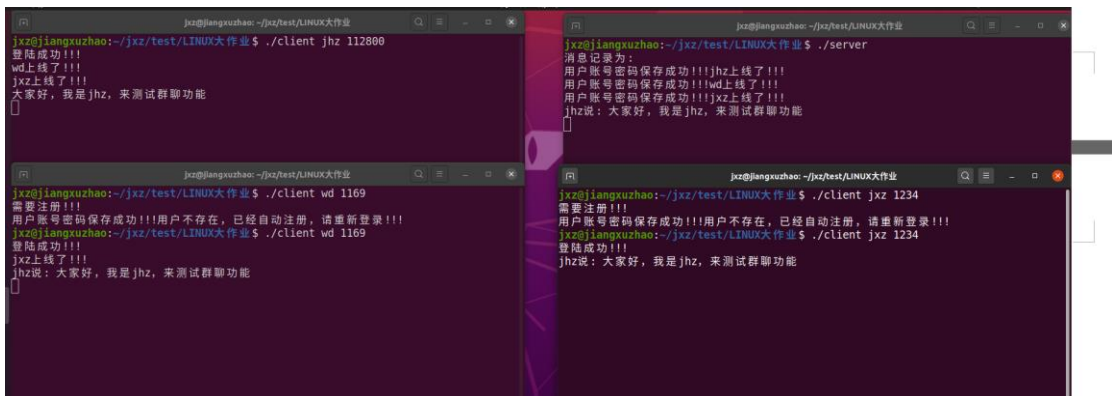
`./client wd 1169`

`./client jhz 112800`

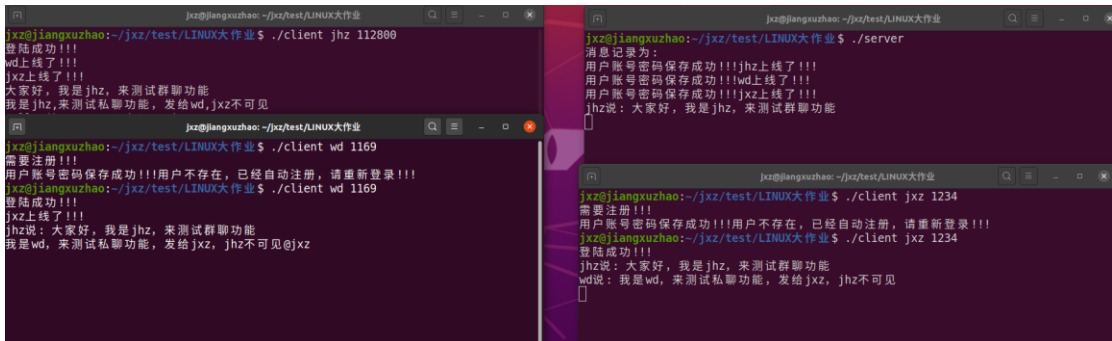
效果图如下：



群聊功能：



私聊功能：



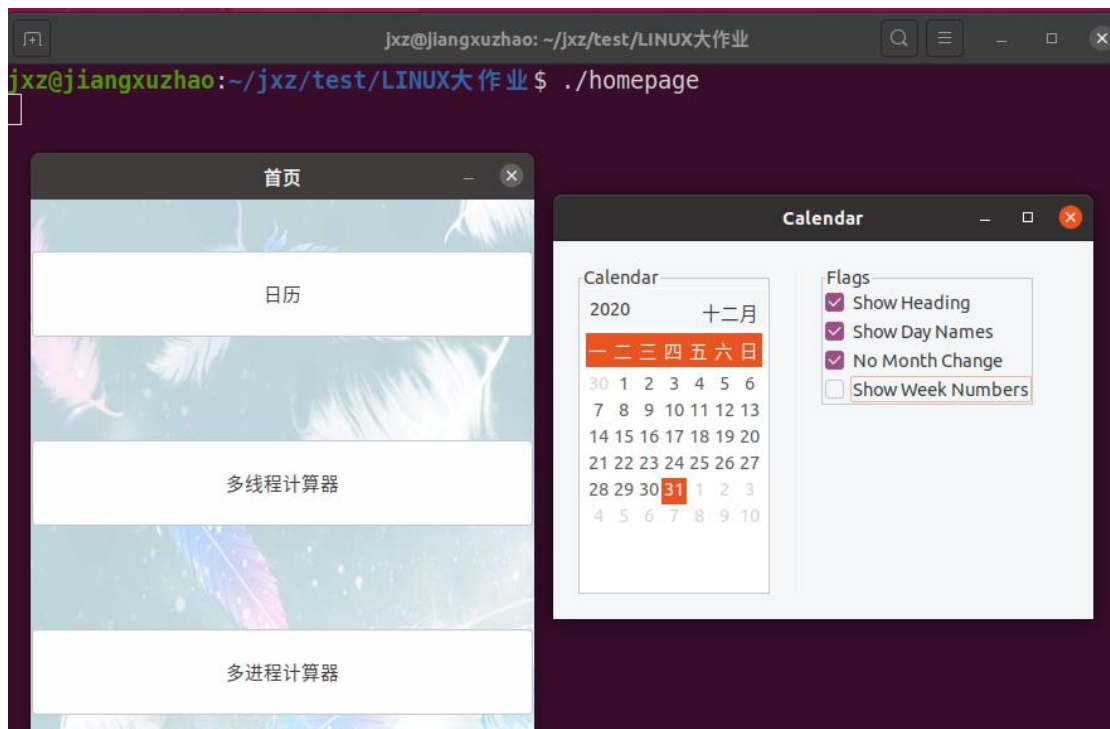
4. 基于 GTK 的日历开发：

在 linux 环境下，自主学习基本的 GTK 图形界面编程的知识，了解到了用 C 语言实现图形布局的难度，同时学习了 C 语言中回调函数的机制。

执行代码：

./homepage 并点击日历

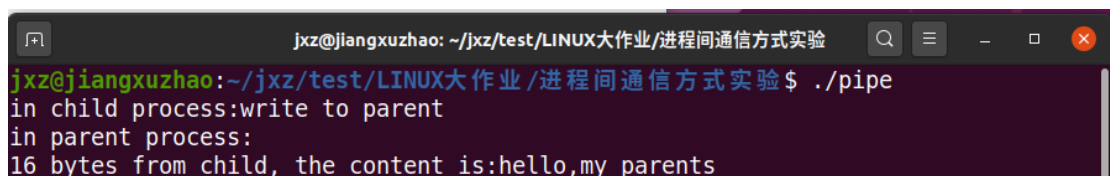
效果图如下：



5. 进程间四种通信方式的探究与试验

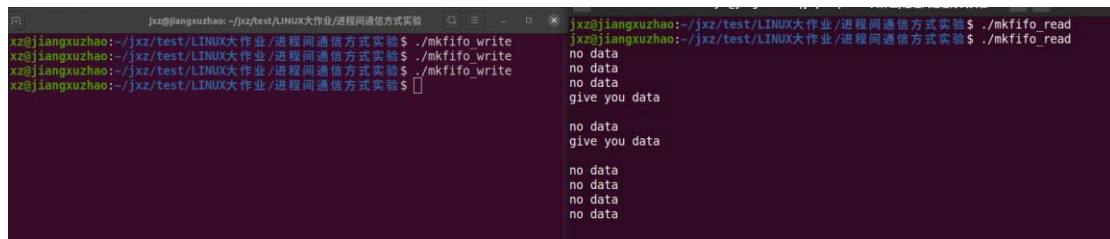
在课堂的最后，老师提及了进程间通信的几种方式，出于对此的兴趣，我运行了几个案例对相关操作进行了熟悉，同时进行了一定的比较。进程间通信的方式有四种：管道（无名管道和有名管道），消息队列，共享内存，信号量

无名管道：用于父子进程间通信。`pipe()` 函数创建无名管道，`fd[0]` 为读端，`fd[1]` 为写端。`pid==0` 子进程写，关闭子进程写端，`pid>0` 父进程读，关闭父进程读端。测试效果图如下：



命名管道：命名管道可以通过路径名来指出，并且在文件系统中是可见的。建立了有名管道之后，两个进程就可以把它当作普通文件一样进行读写操作，使用方便。数据严格遵循 FIFO 先进先出原则。测

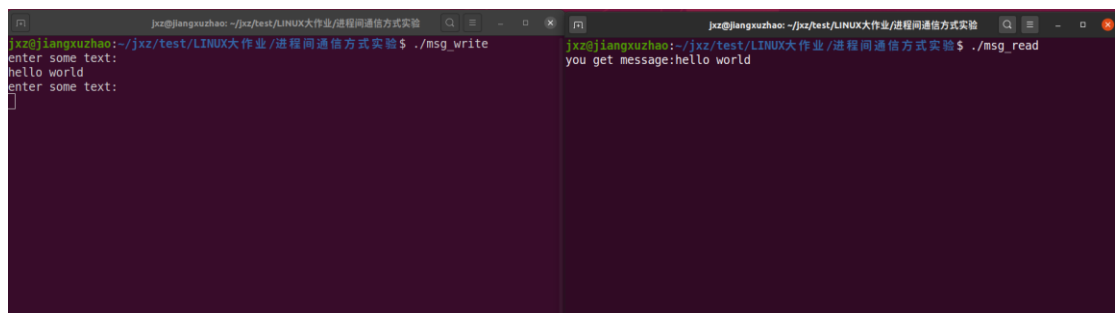
测试效果图如下：



```
jxz@jiangxuzhao: ~/jxz/test/LINUX大作业/进程间通信方式实验
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./mkfifo write
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./mkfifo write
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./mkfifo_write
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$

jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./mkfifo_read
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./mkfifo_read
no data
no data
no data
give you data
no data
give you data
no data
no data
no data
no data
```

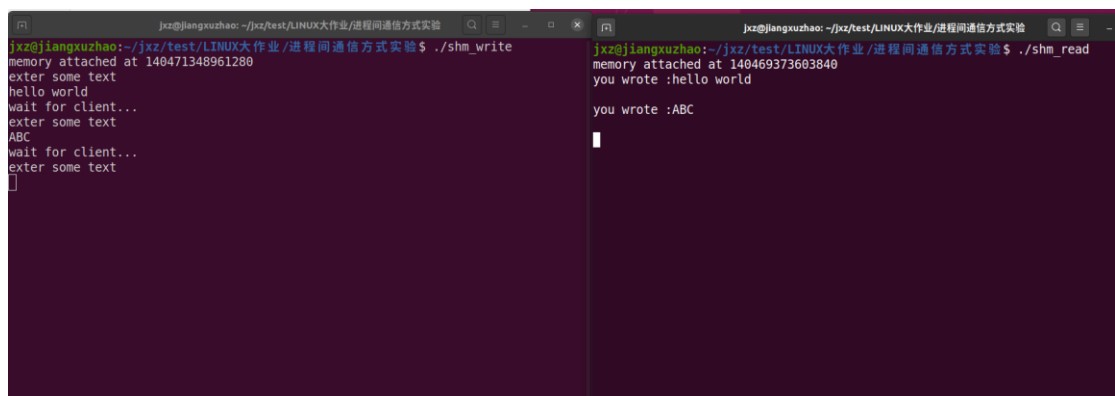
消息队列：用于运行于同一台机器上的进程通信，和管道很相似，是在一个系统的内核中用来保存消息的队列，它在系统内核中是以消息链表的形式出现的。可以同时发通过发送消息来避免命名管道的同步和阻塞问题，而不需要进程自己提供同步方法。测试效果图如下：



```
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./msg_write
enter some text:
hello world
enter some text:
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$

jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./msg_read
you get message:hello world
```

共享内存：允许两个不相关的进程访问同一个逻辑内存。数据不用进行传送，可以之间访问内存，加快了效率，但是没有提供同步机制。测试效果图如下：

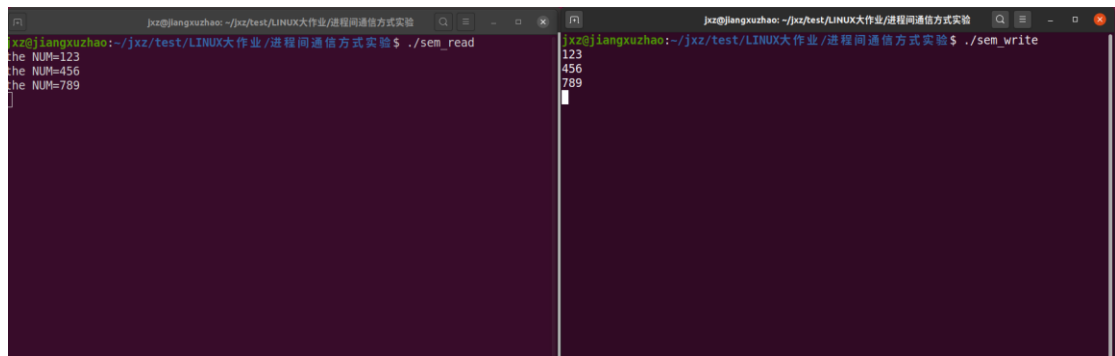


```
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./shm_write
memory attached at 140471348961280
enter some text
hello world
wait for client...
enter some text
ABC
wait for client...
enter some text
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$

jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./shm_read
memory attached at 140469373603840
you wrote :hello world
you wrote :ABC
```

信号量：共享内存是进程间最快的通信方式，但是共享内存的同步问题共享内存无法解决，可以采用信号量解决共享内存的同步问题。

信号负责共享内存的同步，而共享内存负责进程间的通信。将共享内存和信号量结合的方法非常好，因为共享内存是进程间通信最快的方式，而信号量又可以很好的解决共享内存的同步问题。测试效果图如下：



```
jxz@jiangxuzhao: ~/jxz/test/LINUX大作业/进程间通信方式实验
jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./sem_read
the NUM=123
the NUM=456
the NUM=789

jxz@jiangxuzhao:~/jxz/test/LINUX大作业/进程间通信方式实验$ ./sem_write
123
456
789
```

四、心得体会

在学习这门课之前我就多次听到 LINUX 操作系统，并被它的开源性深深吸引，因此选下了这门专选课。通过这个学期的学习，我逐渐窥其奥秘，按学习专题来分，我对 LINUX 的 I/O 操作，文件与目录属性、进程和线程、信号等等有了较深的了解，在大作业的设计中，我有意识地将其中的各个方面进行有效地结合，最后对进程间通信产生了比较浓厚地探索欲望，于是积极地搜寻相关资料，进行优劣性地对比。在学习 LINUX 的过程中，我真切感受到了“万物皆在掌握”的快感，相比于我们学生群体中十分普及的 windows 系统，LINUX 甚至给了我们深入底层的快捷机会。同时，“万物皆文件”的思想也让我大开眼界，小到一个文档，大到网络编程和系统文件，一切都和那个牛鼻子 fd 有关，高屋建瓴来看，一棵从 root 根延展的文件系统已经深

深深地刻在了我的脑海里。我觉得 LINUX 课程给我们的是一种缜密的操作思维，深入底层操作做探究，这应该是一个程序员的基本素养。同时 LINUX 操作系统也给了我们探索其他相关领域的机会，在云计算盛行的现在，在 LINUX 系统上部署分布式系统已经是一种很普遍的现象了。这次课程帮我打开了新世界的大门，希望可以用这笔财富继续武装我的头脑。