

中国矿业大学计算机学院

2019 级本科生课程作业

课程名称	Linux 操作系统
作业次数	作业三（结课大作业）
作业时间	2022 年 5 月 6 日
姓 名	胡钧耀
学 号	06192081
专 业	计算机科学与技术
任课教师	姜秀柱

目 录

一、 基本命令 (20'=2×10')	1
1.1 第一小问	1
1.2 第二小问	5
二、 shell 编程 (30'=10'+5×4')	8
2.1 问题分析与流程图绘制	8
2.2 程序源代码	9
2.3 运行截图	11
三、 socket 编程 (30'=18'+12')	15
3.1 TCP 运算服务	15
3.2 UDP 聊天室	18
四、 进程 (20')	24
4.1 问题分析	24
4.2 程序源代码	24
4.3 运行结果截图	27
五、 附加 (20')	31
5.1 问题分析	31
5.2 程序源代码	31
5.3 运行结果截图	32
六、 感悟	33
参考文献	33

一、基本命令 (20'=2×10')

题目：给出完成以下功能的 Linux 基本命令及每条命令的执行结果截图。

(1) 查看当前目录，在当前目录下创建一个新目录，然后进入这个新目录，在这个新目录下创建一个空文件，分别查看该文件的简单列表，文件类型和详细属性以及该文件所占空间，接下来将该文件的所有者改为 root，赋给所有者读写执行完全权限，并将该文件的有效时间更新为 2027 年 8 月 1 日 24 时，最后回到当前目录。

(2) 在当前目录下，用屏幕输出命令将"hello, world!"写入操作(1)建立的文件中，并用接受键盘输入在屏幕显示的命令向该文件添加三组姓名-学号对（姓名用拼音）的内容，然后计算该文件的单词数。接下来对该文件内容按行排序，输出最后一行，再将文件中的学号提取出来输出到一个新建文件中，比较这两个文件，比较结果输出到第三个文件中。最后将该目录下的三个文件拷贝到其父目录中，再将该目录删除。

1.1 第一小问

该问题为基本命令的操作，根据问题描述，写出每一条要求对应命令的命令即可，分析问题可知，该小问的可以被拆解为如下步骤。

1.1.1 查看当前目录

使用 `pwd` 命令直接查询即可，本次作业的文件夹是 Linux 课程的结课实验的第一题，即 `~/LinuxCourse/final_lab/t1`，因此输入该命令后应该在命令行得到同样结果。

```
pwd
```

命令输入后结果如下，得到当前目录的路径 `~/LinuxCourse/final_lab/t1`。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ pwd
/home/hujunyao/LinuxCourse/final_lab/t1
```

图 1.1-1 查看当前目录

1.1.2 在当前目录下创建一个新目录

使用 `mkdir` 命令可以创建新目录，因此创建名为 `newfolder` 的文件夹。为了验证 `newfolder` 是否创建成功。

```
mkdir newfolder
```

命令输入后结果如下，首先当前文件夹下为空，创建命令结束后，`ls` 命令只列出了一个名为 `newfolder` 文件夹，说明创建成功。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ ls
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ mkdir newfolder
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ ls
newfolder
```

图 1.1-2 创建目录

1.1.3 然后进入这个新目录

进入目录使用 `cd` 命令，直接进入 `newfolder` 文件夹即可。

```
cd newfolder
```

命令输入后，观察到命令行提示已经进入了 *newfolder* 文件夹内，即当前地址为 *~/LinuxCourse/final_lab/t1/newfolder*。或者使用 *pwd* 验证当前目录也可以。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ cd newfolder
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ pwd
/home/hujunyao/LinuxCourse/final_lab/t1/newfolder
```

图 1.1-3 进入目录

1.1.4 在这个新目录下创建一个空文件

创建空文件使用 *touch* 命令，直接创建 *newfile.txt* 即可。可使用 *ls* 命令进行验证，查看当前文件夹内是否有名为 *newfile.txt* 的文件。

```
touch newfile.txt
```

命令输入后结果如下，首先当前文件夹下为空，创建命令结束后，*ls* 命令只列出了一个名为 *newfile.txt* 的文件，说明创建成功。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ ls
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ touch newfile.txt
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ ls
newfile.txt
```

图 1.1-4 创建空文件

1.1.5 查看该文件的简单列表

这一小问其实不是太懂具体问的什么，是指“该文件夹下的文件简单列表”还是说“该文件的简单信息列表”，但看命令的话，对于前者应该是使用 *ls* 命令，对于后者好像并没有简单信息，只有详细信息（下文讲提到）。

```
ls
```

命令输入后，观察到命令行提示当前文件夹只有 *newfile.txt* 一个文件。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ ls
newfile.txt
```

图 1.1-5 查看文件列表

1.1.6 查看该文件的文件类型

这里使用 *file* 命令即可查看该文件的类型。

```
file newfile.txt
```

命令输入后，提示 *newfile.txt* 是一个空文件，而不是文本文件。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ file newfile.txt
newfile.txt: empty
```

图 1.1-6 查看文件类型

1.1.7 查看该文件的详细属性

这里使用 *stat* 命令，其用于显示文件或文件系统的详细信息。

```
stat newfile.txt
```

命令输入后, *newfile.txt* 文件的名称、文件大小、inode 结点信息、链接信息、访问时间、更改时间等详细信息。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ stat newfile.txt
 文件: newfile.txt
 大小: 0                块: 0                IO 块: 4096    普通空文件
设备: 805h/2053d        Inode: 1572944      硬链接: 1
权限: (0664/-rw-rw-r--) Uid: ( 1000/hujunyao)  Gid: ( 1000/hujunyao)
最近访问: 2022-05-01 11:15:45.570815115 +0800
最近更改: 2022-05-01 11:15:45.570815115 +0800
最近改动: 2022-05-01 11:15:45.570815115 +0800
创建时间: -
```

图 1.1-7 查看文件详细属性

1.1.8 查看该文件的所占空间

这里使用 `du` 命令, 其用于显示文件或文件系统的详细信息。

```
du newfile.txt
```

命令输入后, *newfile.txt* 文件的大小将被显示出来, 因为是空文件, 只有 0 字节。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ du newfile.txt
0      newfile.txt
```

图 1.1-8 查看文件详细属性

1.1.9 修改该文件的所有者为 root

这里使用 `chown` 命令, 修改 *newfile.txt* 的所有者为 `root`, 因为涉及到管理员, 因此需要加上 `sudo` 前缀, 扩大命令权限。

```
sudo chown root newfile.txt
```

命令输入前后, 使用 `ls -l` 命令可以查看 *newfile.txt* 的所有者是否被更改, 可知已经从 `hujunyao` 变成了 `root`, 说明所有者已经修改成功。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ ls -l
总用量 0
-rw-rw-r-- 1 hujunyao hujunyao 0 5月  1 11:15 newfile.txt
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ chown root newfile.txt
chown: 正在更改 'newfile.txt' 的所有者: 不允许的操作
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ sudo chown root newfile.txt
[sudo] hujunyao 的密码:
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ ls -l
总用量 0
-rw-rw-r-- 1 root hujunyao 0 5月  1 11:15 newfile.txt
```

图 1.1-9 修改文件所有者

1.1.10 修改该文件所有者读写执行完全权限

这里使用 `chmod` 命令, 修改 *newfile.txt* 的所有者 `user` 的读写修改权限为完全权限, 也就是 `rwx`, 同时需要加上 `sudo` 前缀, 扩大命令权限。

```
sudo chmod u=rwx newfile.txt
```

命令输入前后, 使用 `ls -l` 命令可以查看 *newfile.txt* 的所有者是否被更改, 可知该文件的权限位已经从已经 `-rw-rw-r--` 变成了 `-rwxrwx-r--`, 说明已经修改成功。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ ls -l
总用量 0
-rw-rw-r-- 1 root hujunyao 0 5月  1 11:15 newfile.txt
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ chmod u=rwx newfile.txt
chmod: 正在更改 'newfile.txt' 的权限: 不允许的操作
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ sudo chmod u=rwx newfile.txt
[sudo] hujunyao 的密码:
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ ls -l
总用量 0
-rwxrwx-r-- 1 root hujunyao 0 5月  1 11:15 newfile.txt
```

图 1.1-10 修改文件读写执行权限

1.1.11 修改该文件的有效时间为 2027 年 8 月 1 日 24 时

这一小问也其实不是太懂具体问的什么，是指“修改用户密码的有效时间”，还是“修改该文件的一种时间（访问、更改、改动时间）”，个人理解是修改后者的更改时间，这个可以使用 `stat` 命令对文件时间进行修改，但是在题目中“24 时”是不存在的，而且通常在生活中使用 24 小时制的时间也是从 00:00-23:59，不存在“24 时”的说法，就像进制一样，应该逢 24 进 1，因此认为和题目对应的表达也就是“2027 年 8 月 2 日 0 时”。使用该命令时，同时需要加上 `sudo` 前缀，扩大命令权限。`m` 代表修改更改时间（modified time），`t` 代表时间戳（timestamp），其语法是 `[[CC]YY]MMDDhhmm[.ss]`。

```
sudo touch -mt 202708020000.00 newfile.txt
```

命令输入前后，使用 `touch` 命令可以查看 `newfile.txt` 的修改时间，用来验证文件的更改时间是否被修改。通过命令行的返回信息可知，修改时间已经从 2022-05-01 11:15:45 修改为了 2027-08-02 00:00:00，说明修改成功。同时根据返回信息也可知“202708012400”这种日期是不符合语法的，hh 的范围是 00-23。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ stat newfile.txt
 文件: newfile.txt
 大小: 0          块: 0          IO 块: 4096   普通空文件
设备: 0805h/2053d Inode: 1572944 硬链接: 1
权限: (0764/-rwxr--r--)  Uid: (   0/   root)   Gid: ( 1000/hujunyao)
最近访问: 2022-05-01 12:10:22.839640631 +0800
最近更改: 2022-05-01 11:15:45.570815115 +0800
最近改动: 2022-05-01 13:22:39.951431969 +0800
创建时间: -
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ touch -mt 202708020000.00 newfile.txt
touch: 正在设置 'newfile.txt' 的时间: 不允许的操作
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ sudo touch -mt 202708020000.00 newfile.txt
[sudo] hujunyao 的密码:
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ touch -t 202708012400.00 newfile.txt
touch: 日期格式 "202708012400.00" 无效
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ stat newfile.txt
 文件: newfile.txt
 大小: 0          块: 0          IO 块: 4096   普通空文件
设备: 0805h/2053d Inode: 1572944 硬链接: 1
权限: (0764/-rwxr--r--)  Uid: (   0/   root)   Gid: ( 1000/hujunyao)
最近访问: 2022-05-01 12:10:22.839640631 +0800
最近更改: 2027-08-02 00:00:00.000000000 +0800
最近改动: 2022-05-01 16:02:43.493553054 +0800
创建时间: -
```

图 1.1-11 修改文件日期

1.1.12 回到当前目录

返回到当前目录，也就是回到 `~/LinuxCourse/final_lab/t1` 路径，这是当前文件夹的父级目录，使用 `..` 即可返回到指定的目录。

```
cd ..
```

命令输入前后，使用 `pwd` 命令可以查看当前路径地址，可知最后已经回到了 `~/LinuxCourse/final_lab/t1`，验证成功。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ pwd
/home/hujunyao/LinuxCourse/final_lab/t1/newfolder
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1/newfolder$ cd ..
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ pwd
/home/hujunyao/LinuxCourse/final_lab/t1
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$
```

图 1.1-12 回到当前目录

1.2 第二小问

该问题同样是为基本命令的操作，根据问题描述，其中有一些步骤具有前者的输出是后者的输入的特征，因此不单单是写出每一条要求对应命令的命令即可，最好要能使用管道来优化命令，简化输入，分析问题可知，该小问可以被拆解为如下步骤。

1.2.1 用屏幕输出命令将"hello, world!"写入操作（1）建立的文件中

屏幕输出命令就是 `echo`，写入文件需要使用重定向，也就是将输出的文字使用覆盖写>输入到已经建立好的文件 `newfolder/newfile.txt` 中。

```
echo "hello, world!" > newfolder/newfile.txt
```

命令输入前后，使用 `cat` 命令可以查看 `newfolder/newfile.txt` 的内容是否经过了修改，未修改前，文件为空，输入命令后，输出了 "hello, world!"，说明已经写入文件。

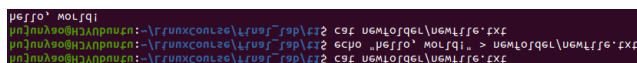


图 1.2-1 写入文件

1.2.2 并用接受键盘输入在屏幕显示的命令向该文件添加三组姓名-学号对（姓名用拼音）的内容，并计算该文件的单词数

既要写入文件，又要输出到屏幕上，此时光用 `echo` 不行，同时需要使用 `tee` 命令进行屏幕输出和管道输出，在添加姓名-学号对时。由于要输入三组，每组一行，为了能输入换行符 `\n`，需要对 `echo` 命令加上 `-e` 参数。由于要进行追加写操作，对 `tee` 命令加上 `-a` 参数。最后要计算文件单词数，这里不能使用管道，而是用单独的命令，因为前一个 `tee` 命令输出了只是三组姓名-学号对，不包括文件中的第一行 "hello, world!"，因此要用分号分隔开，单独使用 `wc` 命令，而不能使用 `|wc`，否则只统计三行数据。

```
echo -e "hujunyao-06192081\nzhangsan-11111111\nlisi-22222222" | tee -a newfolder/newfile.txt; wc newfolder/newfile.txt
```

输入命令后，命令上输出了三行姓名-学号对，同时统计出了文件的字符统计数量，单词数是 5 个 ("hello,", "world!" 和三组姓名-学号对)。在该命令输入前后，使用 `cat` 命令可以查看 `newfolder/newfile.txt` 的内容是否经过了修改，未修改前，文件只有一行，输入命令后，文件为四行，说明已经写入文件。

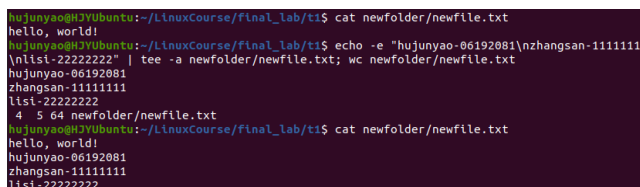


图 1.2-2 键盘输入与单词统计

1.2.3 对文件内容按行排序，输出最后一行

按照行排序使用 `sort` 命令，输出最后一行使用 `tail` 命令，并加上 `-1` 参数设置只输出最后一行，使用管道后 `tail` 无需指定文件名，而是将 `sort` 的输出结果作为输入。

```
sort newfolder/newfile.txt | tail -1
```

这个文件后四行，开头分别为 he, hu, z, l, 排序结果是 he, hu, l, z, 所以输出结尾最后一行的结果应该是 zhangsan-11111111, 结果验证正确。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ cat newfolder/newfile.txt
hello, world!
hujunyao-06192081
zhangsan-11111111
lisi-22222222
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ sort newfolder/newfile.txt | tail -1
zhangsan-11111111
```

图 1.2-3 排序与输出结尾

1.2.4 将文件中的学号提取出来输出到一个新建文件中

提取具有规则的信息可以使用正则表达式，也就是使用 `grep` 命令，学号在这里干扰项比较少，直接识别所有的连续数字即可，也就是 `[0-9]*`，同时使用了 `-o` 参数截取匹配的字符串，输出到新建文件可以使用管道，使用 `tee` 命令输出识别到的学号的同时也将其保存到文件 `newfolder/stunum.txt` 中。

```
grep -o "[0-9]*" newfolder/newfile.txt | tee newfolder/stunum.txt
```

输入该命令前后可以使用 `cat` 命令来查看是否将搜索到的学号存入新文件中，验证结果说明三个学号都已经存入 `newfolder/stunum.txt`，验证成功。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ cat newfolder/newfile.txt
hello, world!
hujunyao-06192081
zhangsan-11111111
lisi-22222222
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ grep -o "[0-9]*" newfolder/newfile.txt
tee newfolder/stunum.txt
06192081
11111111
22222222
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ cat newfolder/stunum.txt
06192081
11111111
22222222
```

图 1.2-4 提取信息并保存

1.2.5 比较这两个文件，比较结果输出到第三个文件中

比较文件使用 `diff` 命令，随后是两个需要比较的文件，也就是 `newfolder/newfile.txt` 和 `newfolder/stunum.txt` 进行比较，输出比较结果需要使用管道，使用 `tee` 命令输出识别到的学号的同时也将其保存到文件 `newfolder/diff.txt` 中。

```
diff newfolder/newfile.txt newfolder/stunum.txt |
tee newfolder/diff.txt
```

输入该命令后可以使用 `cat` 命令来查看是否将文件差异存入新文件中，验证成功。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ diff newfolder/newfile.txt
newfolder/stunum.txt | tee newfolder/diff.txt
1,4c1,3
< hello, world!
< hujunyao-06192081
< zhangsan-11111111
< lisi-22222222
---
> 06192081
> 11111111
> 22222222
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ cat newfolder/diff.txt
1,4c1,3
< hello, world!
< hujunyao-06192081
< zhangsan-11111111
< lisi-22222222
---
> 06192081
> 11111111
> 22222222
```

图 1.2-5 比较文件并保存

1.2.6 将该目录下的三个文件拷贝到其父目录中

拷贝文件使用 `cp` 命令，这里拷贝文件夹下的所有文件，且没有子文件夹，所以直接使用星号选择文件即可。

```
cp newfolder/* ..
```

输入该命令前后可以使用 `ll` 命令来查看是否将文件拷贝到父目录中，验证结果看到三个文件（`newfile.txt`，`stunum.txt`，`diff.txt`）都成功拷贝到 `~/LinuxCourse/final_lab`。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ ll ..
总用量 12
drwxrwxr-x 3 hujunyao hujunyao 4096 5月  1 19:58 ./
drwxrwxr-x 3 hujunyao hujunyao 4096 4月 30 19:39 ../
drwxrwxr-x 3 hujunyao hujunyao 4096 5月  1 19:55 t1/
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ cp newfolder/* ..
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t1$ ll ..
总用量 24
drwxrwxr-x 3 hujunyao hujunyao 4096 5月  1 19:58 ./
drwxrwxr-x 3 hujunyao hujunyao 4096 4月 30 19:39 ../
-rw-rw-r-- 1 hujunyao hujunyao 117 5月  1 19:58 diff.txt
-rwxr--r-- 1 hujunyao hujunyao  64 5月  1 19:58 newfile.txt*
-rw-rw-r-- 1 hujunyao hujunyao  27 5月  1 19:58 stunum.txt
drwxrwxr-x 3 hujunyao hujunyao 4096 5月  1 19:55 t1/
```

图 1.2-6 拷贝文件

1.2.7 将该目录删除

该目录说的也就是 `~/LinuxCourse/final_lab/t1` 目录，首先使用 `cd` 命令退回到父级目录，然后使用 `rm` 命令删除该文件夹，同时加上 `-r` 递归参数。

```
cd ..
rm -r t1
```

输入该命令前后可以使用 `ls` 命令来查看是否目录是否已被删除，验证结果正确。

```
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab$ ls
diff.txt newfile.txt stunum.txt t1
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab$ rm t1
rm: 无法删除 't1': 是一个目录
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab$ rm -r t1
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab$ ls
diff.txt newfile.txt stunum.txt
```

图 1.2-7 删除目录

二、shell 编程（30'=10'+5×4'）

题目：该程序使用一个用户名为参数，如果指定参数的用户存在，就显示其存在，否则添加之，并设置初始密码为 123456，显示添加的用户的 id 等信息，当用户第一次登录时，会提示用户立即修改密码。

如果没有参数，则显示如下菜单：

- (1) 显示用户选择的目录内容；
- (2) 按照用户输入的目录切换路径；
- (3) 按照用户输入的文件名在/home 目录中创建文件；
- (4) 编辑用户输入的文件；
- (5) 删除用户选择的文件。

要求程序执行过程中如发生语法错误或命令错误能够自动提示，同时停止执行，并给出相应提示信息。

给出程序代码和全部执行结果截图。

2.1 问题分析与流程图绘制

根据分析，分析后的程序主要流程图如下所示。由题目可知，需要编写的程序需要接受两种合法类型的参数数量，一种是没有参数，一种是有参数的，还有一种大于一个参数的非法情况。问题提到了首次登录的问题，认为应该把判断用户是否首次登陆放在运行该程序最前面，就如同网站用户登录一样，只有先修改不安全的密码才能进行以后的后续操作。对于没有参数的情况，是进行一系列用户的操作，如进入目录、创建文件等，相当于复写了 Linux 系统的文件基本操作，对于一个参数的情况，输入的参数是新建的用户名称，这里规定了只有 root 管理员才能进行新用户的创建，避免混乱，之后，还需要判断将要创建的用户是否存在，只有不存在的时候才能继续进行用户的创建。

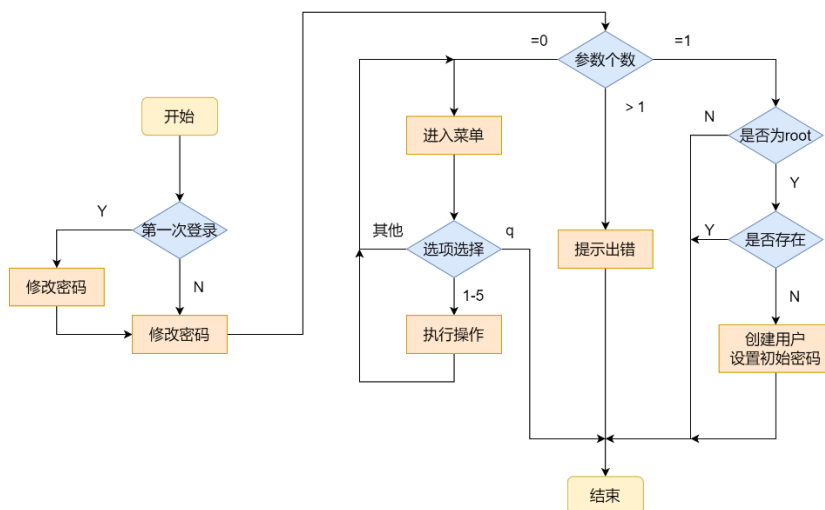


图 2.1-1 Shell 编程程序主要流程图

2.2 程序源代码

```
#!/bin/bash
# 检查用户是否首次登陆
if [ $(last | grep $(whoami) | wc -l) -le 1 ]
then
    echo "检测到首次登陆，您的密码过于简单，请重新设置"
    passwd $username
fi
# 判断输入参数个数
if [ $# -eq 0 ]
then
    echo "没有参数"
    echo ""
    while : true
    do
        echo ""
        echo "*****菜单*****"
        echo "请根据下列提示选择操作： "
        echo "1. 显示用户选择的目录内容"
        echo "2. 按照用户输入的目录切换路径"
        echo "3. 按照用户输入的文件名在/home 目录中创建文件"
        echo "4. 编辑用户输入的文件"
        echo "5. 删除用户选择的文件"
        echo "q. 退出"
        echo "*****"
        echo ""
        read -p "请选择 1-5: " option
        echo "你输入的是: " $option
        case $option in
            "1")
                echo "当前地址"
                pwd
                ;;
            "2")
                read -p "请输入需要切换的路径: " address;
                cd $address
                if [ $? -eq 0 ] #判断用户是否添加成功
                then
                    echo "已成功进入 $address"
                else
                    echo "未能进入 $address，请重试"
                fi
                ;;
            "3")
```

```
    read -p "请输入需要新建的文件: " filename;
    cd /home/${whoami}
    touch $filename
    if [ $? -eq 0 ] #判断文件是否添加成功
    then
        echo "已成功创建 $filename"
    else
        echo "未能创建 $filename, 请重试"
    fi
;;
"4")
    read -p "请输入需要编辑的文件: " filename
    vim $filename
    if [ $? -eq 0 ] #判断文件是否编辑成功
    then
        echo "已成功编辑 $filename"
    else
        echo "未能编辑 $filename, 请重试"
    fi
;;
"5")
    read -p "请输入需要删除的文件: " filename
    rm $filename
    if [ $? -eq 0 ] #判断文件是否删除成功
    then
        echo "已成功删除 $filename"
    else
        echo "未能删除 $filename, 请重试"
    fi
;;
"q")
    echo "即将退出..."
    exit
;;
*)
    echo "参数错误, 请重新输入..."
    continue
;;
esac
done
elif [ $# -eq 1 ]
then
    username=$1
    echo "输入的用户名是 $username"
```

```

if [ $UID -ne 0 ]
then
    echo "请切换到 root 用户进行新增用户！"
    exit 0
fi
if id -u $username >/dev/null 2>&1;
then
    echo "用户 $username 存在，请重新输入"
    exit 0
else
    echo "用户 $username 不存在，即将创建..."
    echo -e "123456\n123456\n\n\n\n\n\n\n" | adduser
$username
    echo -e "\033[32m 默认密码为 123456，请登陆后及时修改\n\033[0m"
    id $username
    if [ $? -eq 0 ] #判断用户是否添加成功
    then
        echo "已成功添加用户 $username "
    else
        echo "未能成功添加用户 $username "
    fi
fi
else
    echo "参数太多了，有问题啊，重开吧"
    echo ""
fi

```

2.3 运行截图

2.3.1 参数数量测试

以 hujunyao 用户运行 *t2.bash* 程序，参数为 0 个，进入菜单界面；参数为 1 个，进入创建用户功能，但因非 root 管理员用户而退出程序；参数为 1 个以上，因参数过多而退出程序。三种情况的运行截图如下所示。

```

hujunyao@H3YUbuntu:~/LinuxCourse/Final_lab/t2$ bash t2.sh
没有参数

*****菜单*****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在/home目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****
请选择 1-5:

hujunyao@H3YUbuntu:~/LinuxCourse/Final_lab/t2$ bash t2.sh alice
输入的用户名是 alice
请切换到root用户进行新增用户!
hujunyao@H3YUbuntu:~/LinuxCourse/Final_lab/t2$

hujunyao@H3YUbuntu:~/LinuxCourse/Final_lab/t2$ bash t2.sh alice bob
参数太多了，有问题啊，重开吧
hujunyao@H3YUbuntu:~/LinuxCourse/Final_lab/t2$

```

图 2.3-1 参数数量测试

2.3.2 用户菜单操作测试

以 `hujunyao` 用户以 0 个参数运行 `t2.bash` 程序。进入菜单界面。

输入数字 1，输出了当前所在的目录为 `/home/hujunyao/LinuxCourse/final_lab/t2`，然后再次循环运行到菜单界面。截图如下所示。

```
hujunyao@H3YUubuntu:~/LinuxCourse/final_lab/t2$ bash t2.sh
没有参数

*****菜单*****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在/home目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****

请选择 1-5: 1
你输入的是: 1
当前地址
/home/hujunyao/LinuxCourse/final_lab/t2

*****菜单*****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在/home目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****

请选择 1-5: █
```

图 2.3-2 显示用户当前目录

输入数字 2，程序要求输入需要跳转的路径，输入 `/home/hujunyao` 作为需要跳转的路径，然后程序将显示是否跳转成功，然后再次循环运行到菜单界面。截图如下所示。

```
请选择 1-5: 2
你输入的是: 2
请输入需要切换的路径: /home/hujunyao
已成功进入 /home/hujunyao

*****菜单*****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在/home目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****

请选择 1-5: █
```

图 2.3-3 切换用户当前路径

输入数字 3，程序要求输入需要创建的文件名称，输入 `test.txt` 作为创建的文本文件，然后程序将显示是否创建成功，然后再次循环运行到菜单界面。可以使用 `ll` 和 `grep` 组合命令验证是否已经创建该文件，截图如下所示。

```
请选择 1-5: 3
你输入的是: 3
请输入需要新建的文件: test.txt
已成功创建 test.txt

*****菜单*****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在/home目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****

请选择 1-5: █

hujunyao@H3YUubuntu:~$ ll | grep test
-rw-rw-r-- 1 hujunyao hujunyao 0 5月 5 17:41 test.txt
hujunyao@H3YUubuntu:~$ █
```

图 2.3-4 创建文件

输入数字 4, 程序要求输入需要修改的文件名称, 输入 `test.txt` 作为修改的文本文件, 也就是之前创建的那一个文件, 然后命令行将进入 `vim` 编辑器模式, 可以进行文本的修改, 随后将文件进行保存, 然后再次循环运行到菜单界面。使用 `cat` 命令验证是否已经修改过该文件, 截图如下所示。

```
请选择 1-5: 4
你输入的是: 4
请输入需要编辑的文件: test.txt
已成功编辑 test.txt

*****菜单*****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在/home目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****

请选择 1-5: 
hujunyao
hello,world!
i love linux
~
hujunyao@H3YUubuntu:~$ cat test.txt
hujunyao
hello,world!
i love linux
hujunyao@H3YUubuntu:~$
```

图 2.3-5 修改文件

输入数字 5, 程序要求输入需要删除的文件名称, 输入 `test.txt` 作为将要被删除的文本文件, 也就是之前创建的那一个文件, 然后再次循环运行到菜单界面。可以使用 `ll` 和 `grep` 组合命令验证是否已经删除该文件, 截图如下所示。

```
请选择 1-5: 5
你输入的是: 5
请输入需要删除的文件: test.txt
已成功删除 test.txt

*****菜单*****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在/home目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****

请选择 1-5: 
hujunyao@H3YUubuntu:~$ ll | grep test
hujunyao@H3YUubuntu:~$
```

图 2.3-6 删除文件

输入 `q`, 程序直接退出。输入其他非法选项, 将提示重新输入, 然后再次循环运行到菜单界面。截图如下所示。

```
*****菜单*****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在/home目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****

请选择 1-5: q
你输入的是: q
即将退出...
hujunyao@H3YUubuntu:~/LinuxCourse/final_lab/t2$

请选择 1-5: 123456abc
你输入的是: 123456abc
参数错误, 请重新输入...

*****菜单*****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在/home目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****

请选择 1-5: 
```

图 2.3-7 退出程序与非法输入

2.3.3 创建用户操作测试

以 hujunyao 用户以 1 个参数 `usertest` 运行 `t2.bash` 程序。由于 hujunyao 并不是 root 用户，会直接提示需要 root 权限并退出程序。因此需要使用 `su` 命令将当前用户设置为 root 管理员，再以 1 个参数 `usertest` 运行 `t2.bash` 程序。此时提示成功创建 `usertest`，为了验证是否创建成功，同时测试程序是否能检测到参数代表的用户已经被创建，再在 root 管理员模式下以 1 个参数 `usertest` 运行 `t2.bash` 程序，此时程序将提示用户已存在，说明已经创建了 `usertest` 新用户，并且初始密码是 123456。

```
hujunyao@JYUbuntu:~/LinuxCourse/final_lab/t2$ bash t2.sh usertest
输入的用户名是 usertest
请切换到root用户进行新增用户!
hujunyao@JYUbuntu:~/LinuxCourse/final_lab/t2$ su
密码:
root@JYUbuntu: /home/hujunyao/LinuxCourse/final_lab/t2# bash t2.sh usertest
输入的用户名是 usertest
用户 usertest 不存在, 即将创建...
正在添加用户 "usertest"...
正在添加新组 "usertest" (1009)...
正在添加新用户 "usertest" (1009) 到组 "usertest"...
创建主目录 "/home/usertest"...
正在从 "/etc/skel" 复制文件...
新的 密码: 重新输入新的 密码: passwd: 已成功更新密码
正在改变 usertest 的用户信息
请输入新值, 或直接敲回车键以使用默认值
全名 []: 房间号码 []: 工作电话 []: 家庭电话 []: 其它 []:
Use of uninitialized value $answer in chop at /usr/sbin/adduser line 621.
Use of uninitialized value $answer in pattern match (n//) at /usr/sbin/adduser l
line 622.
这些信息是否正确? [Y/n] 默认密码为123456, 请登录后及时修改
用户 id=1009(usertest) 组id=1009(usertest) 组=1009(usertest)
已成功添加用户 usertest
root@JYUbuntu: /home/hujunyao/LinuxCourse/final_lab/t2# bash t2.sh usertest
输入的用户名是 usertest
用户 usertest 存在, 请重新输入
root@JYUbuntu: /home/hujunyao/LinuxCourse/final_lab/t2#
```

图 2.3-8 创建用户

2.3.4 登录新用户提示修改密码测试

在 root 管理员模式下使用 `login` 命令登录新的用户 `usertest`，使用初始密码 123456 可以成功登录，`usertest` 用户以 0 个参数运行 `t2.bash` 程序，程序提示首次登录修改密码。成功修改密码后，程序继续运行，也就是进入菜单界面，作为 `usertest` 用户同样可以进行如下操作，但不能以 1 个参数运行该程序，因为 `usertest` 并没有 root 权限。

```
root@JYUbuntu: /home/hujunyao/LinuxCourse/final_lab/t2# login
JYUbuntu 用户名: usertest
密码:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.13.0-40-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

270 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

usertest@JYUbuntu: $ bash /home/hujunyao/LinuxCourse/final_lab/t2/t2.sh
检测到首次登陆, 您的密码过于简单, 请重新设置
更改 usertest 的密码:
Current password:
新的 密码:
重新输入新的 密码:
passwd: 已成功更新密码
没有参数

***** 菜单 *****
请根据下列提示选择操作:
1. 显示用户选择的目录内容
2. 按照用户输入的目录切换路径
3. 按照用户输入的文件名在 /home 目录中创建文件
4. 编辑用户输入的文件
5. 删除用户选择的文件
q. 退出
*****

请选择 1-5:
```

图 2.3-9 修改新用户首次登陆密码

三、socket 编程 (30'=18'+12')

题目：分别编写基于 TCP 和 UDP 的 socket 程序：

- (1) 实现加减乘除和求幂（任选其一）的运算服务（TCP）；
- (2) 实现即时聊天（UDP）。

给出全部程序代码和执行结果截图。

3.1 TCP 运算服务

3.1.1 问题分析

本题选择实现加减乘除的运算服务，主要思路是利用 TCP 协议的相关方法，在客户端上输入需要计算的表达式，传给服务器，由服务器使用 `eval` 方法进行计算，如果能计算出结果，就将结果返回给客户端，如果计算出错，将出错结果返回给客户端，支持客户端连续发送不同的表达式进行持续计算。下图是 TCP 客户端和 TCP 服务器的运算服务流程图。

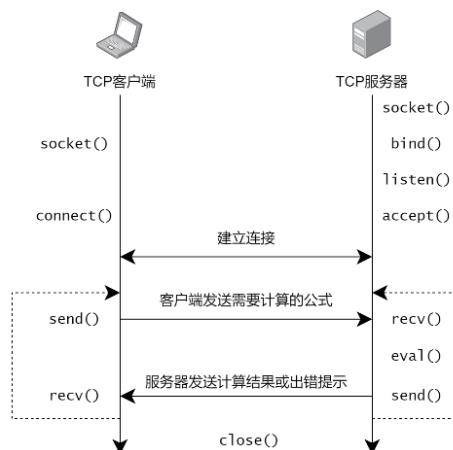


图 3.1-1 TCP 运算服务流程图

3.1.2 程序源代码

如下是客户端的源代码 `client.py`。

```

#!/usr/bin/env python3
import socket
import time
print("客户端已开启")
sk=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
ip_port=('192.168.175.128',5000)
while True:
    try:
        sk.connect(ip_port)
        break
    except ConnectionRefusedError:
        print("服务器拒绝连接，继续尝试连接...")
  
```

```

        time.sleep(2)

print("socket 已连接")
print(sk)
print("")

while True:
    message = input("请输入四则运算公式（输入 exit 退出）")
    sk.send(message.encode('utf-8'))
    if message=='exit':
        print("客户端主动退出...")
        break
    print("[send] [" + time.ctime(time.time()) + "] " + message)
    print("服务器正在计算中...")
    data=sk.recv(1024)
    print ("[recv] [" + time.ctime(time.time()) + "] " +
data.decode('utf-8'))
    print("计算结果为: " + data.decode('utf-8'))
    print("")

sk.close()
print ("退出")

```

如下是服务器的源代码 *server.py*。

```

#!/usr/bin/env python3
import socket
import time
print("服务器已开启")
sk=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
ip_port=('192.168.175.128',5000)
sk.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sk.bind(ip_port)
sk.listen(1)

print("socket 已连接")
a,b=sk.accept()
print(a)
print("")

while True:
    data=a.recv(1024)
    recv_msg = data.decode('utf-8')
    if recv_msg == "exit":
        print("服务器被动退出...")

```

```

        break
    print("[recv] [" + time.ctime(time.time()) + "] " + recv_msg)
    print("服务器正在计算中...")
    try:
        result = str(eval(recv_msg))
    except ZeroDivisionError:
        result = "【错误：除数为0】"
    except NameError:
        result = "【错误：暂不支持变量名输入】"
    except SyntaxError:
        result = "【错误：公式输入有误】"
    print("计算结果为：" + result)
    a.send(result.encode('utf-8'))
    print("[send] [" + time.ctime(time.time()) + "] " + result)
    print("")
sk.close()
print("退出")

```

3.1.3 程序运行截图

首先配置好服务器的 IP 地址，然后开启两个命令行，先运行 *server.py*，后运行 *client.py* 程序，将在建立连接后，双方提示已经建立连接，服务器的端口为 5000，客户端的端口为 50658，且客户端提示可以输入四则运算公式，输入公式后，服务器将接收到客户端发送的公式并进行计算，然后把计算结果返回客户端。同时为了测试程序的鲁棒性，同时测试了除数为零、公式语法有误、输入了字母变量的三种非法情况，均能返回相应错误提示，验证效果较好，最后退出客户端程序，服务器也收到消息并停止服务。执行过程截图如下所示。

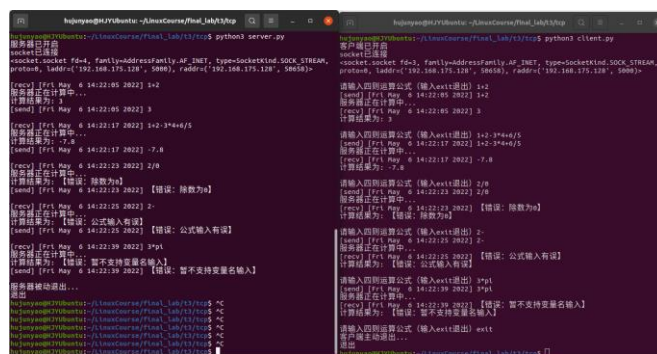


图 3.1-2 TCP 运算服务运行情况

若先运行 *client.py*，再运行 *server.py* 程序，客户端将无法连接到服务器，将不断尝试进行连接，直到服务器开始服务。运行状态如下所示。

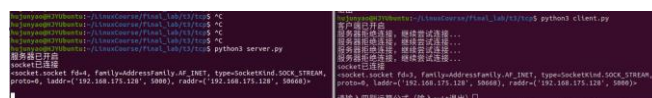


图 3.1-3 客户端等待服务器开始服务

3.2 UDP 聊天室

3.2.1 问题分析

本题要求实现用户聊天室服务，主要思路是利用 UDP 协议的相关方法，在客户端上输入需要发送的消息，传给服务器，由服务器对其发送的信息进行转发给别的和服务端连接的客户端，每个客户端能从服务器接收到来自其他客户端的消息，并能在聊天室展示是否有新的客户端上线，或者老的客户端下线，同时，服务器只会把消息发送给正在上线的客户端，已经下线的或者没有上线的客户端将不进行发送。为了实现同一用户可以连续发送信息，并且接收方可以即时收到消息，构建了发送消息和接收消息两个线程，也就避免了必须双方各说一句才能继续进行发言的问题。下图是 UDP 客户端和 UDP 服务器的聊天服务流程图。

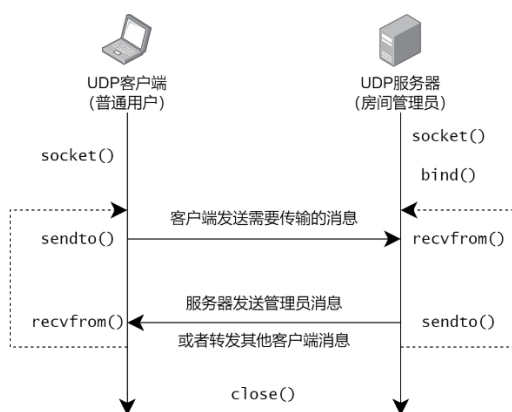


图 3.2-1 UDP 聊天服务流程图

聊天室支持多人进行聊天，而不仅仅是一个客户端和一个服务器进行信息互发，可以有多个客户端同时发送消息，其基本思想是将服务器作为信息收发中心，普通用户想要发送到聊天室的消息需要先发送到服务器，服务器再通过已经建立连接的客户端集合 `expected_ip_port` 依次进行转发，使当前除了发送信息的在线客户端用户自己和服务器本身，其他人都能收到消息。如果是管理员自行发布消息，将发送给所有在线的客户端用户。UDP 聊天室多人转发功能演示拓扑图如下所示。

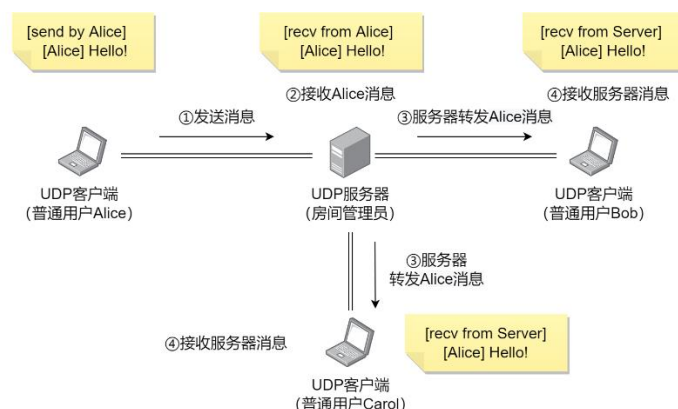


图 3.2-2 UDP 聊天室多人转发功能演示拓扑图

聊天室只会把消息传送给和服务端建立连接的在线用户，其实现思路主要是当某一客户端将要和服务端建立连接时，首先向服务器发送上线通知，这样服务器就能记录客户端的地址并进行保存，放进在线客户端集合 `expected_ip_port` 当中，后续如果有

客户端发送的消息或者管理员的消息，可以通过集合记录来发送消息。同理，如果有客户端用户需要下线，客户端最后将给服务器发送下线通知，服务器识别到下线通知，将把对应的客户端用户从在线客户端集合 `expected_ip_port` 当中移除，后续如果有客户端发送的消息或者管理员的消息，就不能够发送给该客户端了。如果服务器管理员需要关闭聊天室，直接退出聊天室，这样其他的客户端用户也就不能通过服务器的转发功能进行聊天了，发送的信息无法被接收，也无法接收到其他用户发送的消息。UDP 聊天室上线/下线功能与转发功能联动流程图如下所示。

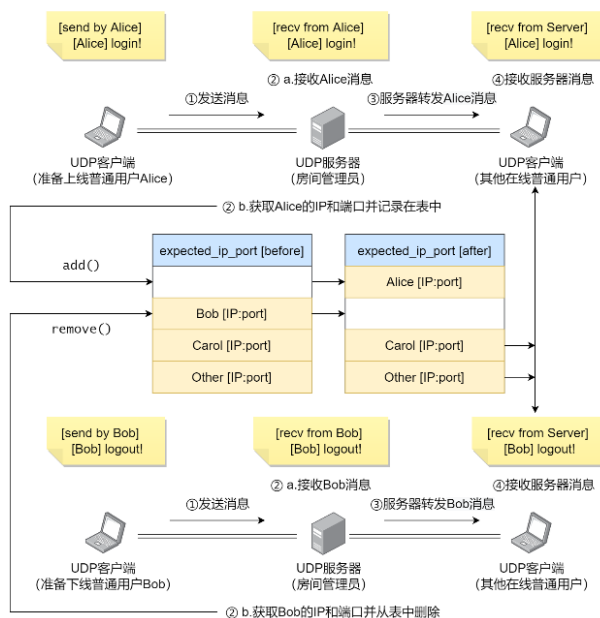


图 3.2-3 UDP 聊天室上线/下线功能与转发功能联动流程图

3.2.2 程序源代码

如下是客户端的源代码 `client.py`。

```
#!/usr/bin/env python3
import socket
import threading
import time
import os

def sendThread(name):
    while True:
        message = input()
        if message == 'exit':
            sk.sendto(("[" + myname + "] 退出了聊天室").encode('utf-8'), ip_port)
            print("退出聊天室")
            sk.close()
            os._exit(0)

        message = '[' + myname + ']' + message
```

```
sk.sendto(message.encode('utf-8'), ip_port)
print("[send] [" + time.ctime(time.time()) + "]")
print(message)
print("")

def recvThread(name):
    while True:
        data=sk.recvfrom(1024)#客户端发送的数据存储在 recv 里, 1024 指
        最大接受数据的量
        if data[0].decode('utf-8') == "exit":
            print("客户端被动关闭 socket")
            break

        print ("[recv] [" + time.ctime(time.time()) + "]")
        print(data[0].decode('utf-8'))
        print("")
    print ("recv 退出线程: " + name)

sk = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
ip = '192.168.175.128'
port = 10000
ip_port=(ip, port)
print(sk)
print("")

myname = input("输入你的聊天昵称: ")
sk.sendto(("[" + myname + "] 进入了聊天室").encode('utf-8'),
ip_port)

# 创建新线程
aRecvThread =
threading.Thread(target=recvThread ,args=("recv",))
aSendThread =
threading.Thread(target=sendThread ,args=("send",))
aRecvThread.start()
aSendThread.start()

print("")
aRecvThread.join()
aSendThread.join()
```

如下是服务器的源代码 *server.py*。

```
#!/usr/bin/env python3
import socket
import threading
import time
import os

def sendThread(name):
    while True:
        message = input()
        if message == 'exit':
            sk.sendto(("[" + myname + "] 关闭了聊天室").encode('utf-8'), ip_port)
            print("关闭聊天室")
            sk.close()
            os._exit(0)

        message = "[管理员]" + message
        for i in expected_ip_port:
            sk.sendto(message.encode('utf-8'), i)
        print("[send] [" + time.ctime(time.time()) + "]")
        print(message)
        print("")

def recvThread(name):
    global expected_ip_port

    while True:
        recv_data=sk.recvfrom(1024)#客户端发送的数据存储在 recv 里，1024 指最大接受数据的量

        if recv_data[1] not in expected_ip_port:
            for i in expected_ip_port:
                sk.sendto(recv_data[0], i)
            expected_ip_port.add(recv_data[1])
            print "[" + time.ctime(time.time()) + "]"
            print(recv_data[0].decode('utf-8'))
            print("")

        elif "]" 退出了聊天室" in recv_data[0].decode('utf-8'):
            expected_ip_port.remove(recv_data[1])
```

```
        for i in expected_ip_port:
            sk.sendto(recv_data[0], i)
        print "[" + time.ctime(time.time()) + "]"
        print(recv_data[0].decode('utf-8'))
        print("")

    else:
        for i in expected_ip_port:
            if i != recv_data[1]:
                sk.sendto(recv_data[0], i)
        print("[recv] [" + time.ctime(time.time()) + "]")
        print(recv_data[0].decode('utf-8'))
        print("")

sk.close()

sk = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
ip = '192.168.175.128'
port = 10000
ip_port=(ip, port)
expected_ip_port = set({})
sk.bind(ip_port)
print(sk)
print("")
myname = '管理员'
print("聊天室已开启")

# 创建新线程
aRecvThread =
threading.Thread(target=recvThread ,args=("recv",))
aSendThread =
threading.Thread(target=sendThread ,args=("send",))
aRecvThread.start()
aSendThread.start()

print("")
aRecvThread.join()
aSendThread.join()
```


3.2.3 程序运行截图

为了验证程序的可行性，在系统中开启了一个服务器和三个客户端（Alice，Bob，Carol），先打开服务器，三个客户端分别在不同时刻进入聊天室，在不同时刻退出聊天室。主要验证的几种情况如下所述。

1. 服务器开启时，客户端能够正常进行上线、下线聊天室；
2. 服务器开启服务时，客户端发送信息，服务器和其他的在线的客户端都能收到消息，未上线和已经下线的客户端不会接收消息；
3. 服务器开启服务时，客户端可以连续发送信息和即时连续接收信息；
4. 服务器关闭服务后，上线的客户端既不能发送信息，也不能接收信息。

主要验证过程截图如下所示。

```
h@huyang:~/Final_Lab/3.2.3$ python3 server.py
socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_DGRAM, proto=0, laddr=('192.168.175.128', 5000)
聊天室已开启
管理员自己说话，现在没人进聊天室
[send] [Fri May 6 20:09:34 2022]
[管理员]管理员现在没人进聊天室
[Fri May 6 20:09:39 2022]
[Alice] 进入了聊天室
[recv] [Fri May 6 20:09:34 2022]
[Alice] alice说了第一句话，你好
[管理员] 回复alice hello
[send] [Fri May 6 20:10:17 2022]
[管理员]管理员回复alice hello
[Fri May 6 20:10:23 2022]
[Bob] 进入了聊天室
[recv] [Fri May 6 20:10:17 2022]
[Bob] bob这个时候进入聊天室
[recv] [Fri May 6 20:10:22 2022]
[Alice] alice这个时候准备退出了
[Fri May 6 20:10:55 2022]
[Bob] bob发言
[send] [Fri May 6 20:11:30 2022]
[Carol] 进入了聊天室
[recv] [Fri May 6 20:11:40 2022]
[Carol] carol进入
[send] [Fri May 6 20:11:46 2022]
[管理员] 关闭了聊天室
exit
[Fri May 6 20:11:46 2022]
[管理员] 关闭了聊天室

h@huyang:~/Final_Lab/3.2.3$ python3 client.py
输入你的聊天昵称: Alice
alice说了第一句话，你好
[send] [Fri May 6 20:09:34 2022]
[Alice] alice说了第一句话，你好
[recv] [Fri May 6 20:10:17 2022]
[管理员]管理员回复alice hello
[recv] [Fri May 6 20:10:23 2022]
[Alice] 进入了聊天室
[recv] [Fri May 6 20:10:55 2022]
[Alice] 退出了聊天室
[send] [Fri May 6 20:11:30 2022]
[Bob] bob发言
[recv] [Fri May 6 20:11:40 2022]
[Carol] carol进入
[send] [Fri May 6 20:11:46 2022]
[管理员] 关闭了聊天室
exit
[Fri May 6 20:11:46 2022]
[管理员] 关闭了聊天室

h@huyang:~/Final_Lab/3.2.3$ python3 client.py
输入你的聊天昵称: Bob
bob这个时候进入聊天室
[send] [Fri May 6 20:10:17 2022]
[Bob] bob这个时候进入聊天室
[recv] [Fri May 6 20:10:22 2022]
[Alice] alice这个时候准备退出了
[recv] [Fri May 6 20:10:55 2022]
[Alice] 退出了聊天室
[send] [Fri May 6 20:11:30 2022]
[Bob] bob发言
[recv] [Fri May 6 20:11:40 2022]
[Carol] carol进入
[send] [Fri May 6 20:11:46 2022]
[管理员] 关闭了聊天室
exit
[Fri May 6 20:11:46 2022]
[管理员] 关闭了聊天室

h@huyang:~/Final_Lab/3.2.3$ python3 client.py
输入你的聊天昵称: Carol
carol进入
[send] [Fri May 6 20:11:40 2022]
[Carol] carol进入
[recv] [Fri May 6 20:11:46 2022]
[管理员] 关闭了聊天室
exit
[Fri May 6 20:11:46 2022]
[管理员] 关闭了聊天室
```

图 3.2-4 UDP 聊天室功能测试

四、进程（20'）

题目：编写一个显示"Hello, World!"的欢迎语程序，再编写一个分别采用 `fork()`、`exec()`、`exit()`、`wait()`调用其执行和适时等待和退出的程序，体会四个函数对进程创建、调度（阻塞、关闭）释放资源的过程和作用。

给出全部程序代码和执行结果截图。

4.1 问题分析

4.2 程序源代码

如下是被调用的欢迎语句源代码 *hello.py*。

```
#!/usr/bin/env python3
import sys
import os

def hello(name):
    print(f'\t\t\t\t\t{os.getppid()}:{os.getpid()} [调] Hello,
{name}!')

hello(sys.argv[1])
```

如下是测试四个函数作用的主程序源代码 *test.py*。

```
#!/usr/bin/env python3
import os
import time

def testForFork():
    def parent():
        for i in range(2):
            print(f'\t\t\t\t\tfork {i} ready , {os.getpid()}')
            newpid = os.fork()
            if newpid == 0:
                print(f'\t\t\t\t\t{os.getppid()}:{os.getpid()} [子{i}]
Hello')
            else:
                print(f'\t\t\t\t\t{os.getppid()}:{os.getpid()} [父{i}]
Hello')
            print(f'\t\t\t\t\tfork {i} finish , {os.getpid()}')
    print("\t\t\t\t\tparent() ready", os.getpid())
    parent()
    print("\t\t\t\t\tparent() finish", os.getpid())
```

```
def testForExec():
    def parent():
        for i in range(2):
            print(f'\t\tfork {i} ready , {os.getpid()}')
            pid = os.fork()
            if pid == 0:
                print(f'\t\t\t{os.getppid()}:{os.getpid()} [子{i}]')
                os.execle("hello.py", 'hello', 'hujunyao')
                print(f'\t\t\t{os.getppid()}:{os.getpid()} [子{i}]')
            else:
                print(f'\t\t\t{os.getppid()}:{os.getpid()} [父{i}]')
        print(f'\t\tfork {i} finish , {os.getpid()}')
    print("\t\t\t\tparent() ready", os.getpid())
    parent()
    print("\t\t\t\tparent() finish", os.getpid())

def testForExit():
    def parent():
        for i in range(2):
            print(f'\t\tfork {i} ready , {os.getpid()}')
            pid = os.fork()
            if pid == 0:
                print(f'\t\t\t{os.getppid()}:{os.getpid()} [子{i}]')
                os._exit(0)
                print(f'\t\t\t{os.getppid()}:{os.getpid()} [子{i}]')
            else:
                print(f'\t\t\t{os.getppid()}:{os.getpid()} [父{i}]')
        print(f'\t\tfork {i} finish , {os.getpid()}')
    print("\t\t\t\tparent() ready", os.getpid())
    parent()
```

```

    print("\tparent() finish", os.getpid())

def testForWait():
    def parent():
        for i in range(2):
            print(f'\t\tfork {i} ready , {os.getpid()}')
            newpid = os.fork()
            if newpid == 0:
                print(f'\t\t\t{os.getppid()}:{os.getpid()} [子{i}]
Hello')
            else:
                os.wait()
                print(f'\t\t\t{os.getppid()}:{os.getpid()} [父{i}]
Hello')

        print(f'\t\tfork {i} finish , {os.getpid()}')
    print("\tparent() ready", os.getpid())
    parent()
    print("\tparent() finish", os.getpid())

print("\n*****菜单*****")
print("1. fork")
print("2. exec")
print("3. exit")
print("4. wait")
print("*****\n")
option = eval(input("输入选项: "))

if option == 1:
    print(f'testForFork() ready', os.getpid())
    testForFork()
    print(f'testForFork() finish', os.getpid())
    #time.sleep(5)

elif option == 2:
    print(f'testForExec() ready', os.getpid())
    testForExec()
    print(f'testForExec() finish', os.getpid())
    #time.sleep(5)

elif option == 3:
    print(f'testForExit() ready', os.getpid())
    testForExit()

```

```

print(f'testForExit() finish', os.getpid())
#time.sleep(5)

elif option == 4:
    print(f'testForWait() ready', os.getpid())
    testForWait()
    print(f'testForWait() finish', os.getpid())
    #time.sleep(5)

else:
    print("没有该选项，请重试")
print("主程序 finish", os.getpid())
time.sleep(1)

```

4.3 运行结果截图

4.3.1 fork()

每一次 `os.fork()` 会分出一个子进程。在子进程中返回 0，在父进程中返回子进程的进程号。代码运行截图如下所示。

```

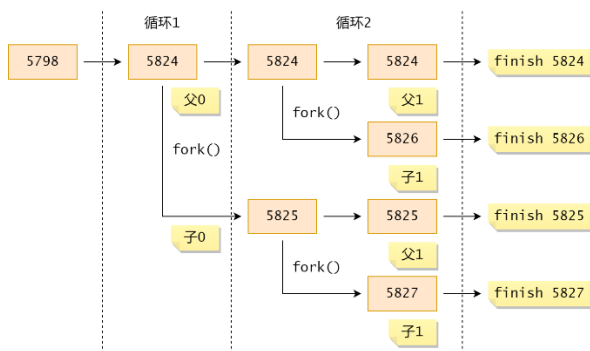
hujunyao@H3YUbuntu:~/LinuxCourse/final_lab/t4$ python3 test.py
*****菜单*****
1. fork
2. exec
3. exit
4. wait
*****

输入选项: 1
testForFork() ready 5824
    parent() ready 5824
        fork 0 ready , 5824
            5798:5824 [父0] Hello
        fork 0 finish , 5824
        fork 1 ready , 5824
            5824:5825 [子0] Hello
        fork 0 finish , 5825
        fork 1 ready , 5825
            5798:5824 [父1] Hello
        fork 1 finish , 5824
    parent() finish 5824
testForFork() finish 5824
主程序 finish 5824
    5824:5825 [父1] Hello
    fork 1 finish , 5825
    parent() finish 5825
testForFork() finish 5825
主程序 finish 5825
    5824:5826 [子1] Hello
    fork 1 finish , 5826
    parent() finish 5826
testForFork() finish 5826
主程序 finish 5826
    5825:5827 [子1] Hello
    fork 1 finish , 5827
    parent() finish 5827
testForFork() finish 5827
主程序 finish 5827
hujunyao@H3YUbuntu:~/LinuxCourse/final_lab/t4$

```

图 4.3-1 `os.fork()` 函数功能测试

通过绘制下图进行分析，这个测试文件进行了两次循环，第一次遇到分支函数，分成两个进程，两个进程分别进入第二次循环，遇到分支函数，一共得到四个进程，也就能看到主程序被结束了四次，第一次循环父子都输出了一次，而第二次循环，父子都输出了两次，这与分支函数的作用和预期一致，再观察进程和子进程的 id 号也同样能说明分支函数的作用。下图为 `os.fork()` 测试程序结果分析示意图。



4.3.2 exec()

exec*函数的 **l** 和 **v** 变体不同在于命令行参数的传递方式。如果在编码时固定了参数数量，则 **l** 变体可能是最方便的，各参数作为 **execl*()** 函数的附加参数传入即可。当参数数量可变时，**v** 变体更方便，参数以列表或元组的形式作为 **args** 参数传递。结尾包含 **p** 的变体将使用环境变量来查找程序 **file**，**path** 必须包含正确的绝对或相对路径。

```
hujunyaoy@H37YUBuntu:~/LinuxCourse/Final_lab7/c4$ python3 test.py
*****菜单*****
1. fork
2. exec
3. exit
4. wait
*****

输入选项: 2
testForExec() ready 5833
    parent() ready 5833
        fork 0 ready, 5833
            5798:5833 [父0] Hello
        fork 0 finish, 5833
        fork 1 ready, 5833
            5833:5834 [子0] Hello ready
            5798:5833 [父1] Hello
        fork 1 finish, 5833
    parent() finish 5833
testForExec() finish 5833
主程序 finish 5833

5833:5835 [子1] Hello ready
5833:5835 [调] Hello, hujunyaoy!
5833:5834 [调] Hello, hujunyaoy!
hujunyaoy@H37YUBuntu:~/LinuxCourse
```

通过绘制下图进行分析，这个测试文件进行了两次循环，第一次遇到分支函数，分成两个进程，父进程进入第二次循环，子进程遇到 `os.execl()`，跳出循环，进行调用。父进程又遇到分支函数，分成两个进程，子进程遇到 `os.execl()`，跳出循环，进行调用，所以，也就能看到主程序只被结束了一次，一共有三个线程。第一次循环父子都输出了一次，而第二次循环，父子都输出了两次，调用的两次也进行输出。这与 `os.fork()` 和 `os.execl()` 的作用和预期一致，再观察进程和子进程的 `id` 号也同样能说明 `os.execl()` 函数的作用。下图为 `os.execl()` 测试程序结果分析示意图。

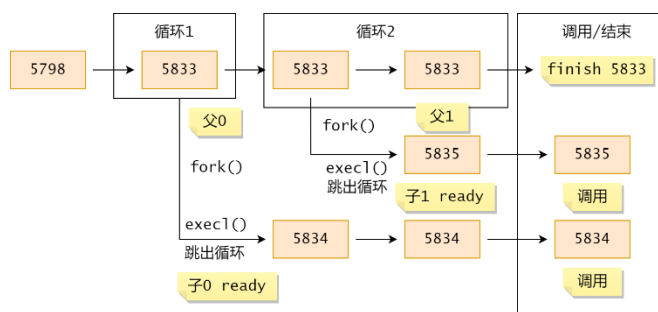


图 4.3-4 os.execl()测试程序结果分析示意图

4.3.3 exit()

os._exit(n)以状态码 n 退出进程，不会调用清理处理程序。退出的标准方法是使用 sys.exit(n)。而_exit()通常只应在 fork()出的子进程中使用。代码运行截图如下所示。

```

hujunyaogH3YUubuntu:~/LinuxCourse/final_lab/t4$ python3 test.py
*****菜单*****
1. fork
2. exec
3. exit
4. wait
*****
输入选项: 3
testForExit() ready 5841
parent() ready 5841
fork 0 ready , 5841
5798:5841 [父0] Hello
fork 0 finish , 5841
fork 1 ready , 5841
5841:5842 [子0] Hello
5841:5842 [子0] exit ready
5798:5841 [父1] Hello
fork 1 finish , 5841
parent() finish 5841
testForExit() finish 5841
主程序 finish 5841
5841:5843 [子1] Hello
5841:5843 [子1] exit ready
hujunyaogH3YUubuntu:~/LinuxCourse/final_lab/t4$

```

图 4.3-5 os._exit()函数功能测试

通过绘制下图进行分析，这个测试文件进行了两次循环，第一次遇到分支函数，分成两个进程，两个进程继续执行，子进程遇到 os._exit()，直接结束。父进程又遇到分支函数，分成两个进程，子进程遇到 os._exit()，直接结束。所以，也就能看到主程序只被结束了一次，一共有三个线程。第一次循环父子都输出了一次，而第二次循环，父子都输出了两次，子进程退出的两次也进行输出，主线程退出输出一次。这与 os.fork()和 os._exit()的作用和预期一致，再观察进程和子进程的 id 号也同样能说明 os._exit()函数的作用。下图为 os._exit()测试程序结果分析示意图。

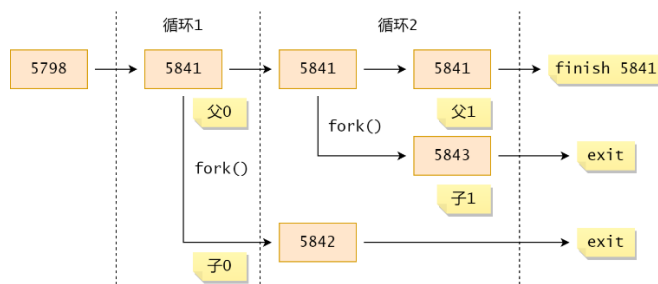


图 4.3-6 os._exit()测试程序结果分析示意图

4.3.4 wait()

`os.wait()`的作用是等待子进程执行完毕。代码运行截图如下所示。

```

hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t4$ python3 test.py
*****菜单*****
1. fork
2. exec
3. exit
4. wait
*****

输入选项: 4
testForWait() ready 5846
parent() ready 5846
fork 0 ready , 5846
5846:5847 [子0] Hello
fork 0 finish , 5847
fork 1 ready , 5847
5847:5848 [子1] Hello
fork 1 finish , 5848
parent() finish 5848
testForWait() finish 5848
主程序 finish 5848
5846:5847 [父1] Hello
fork 1 finish , 5847
parent() finish 5847
testForWait() finish 5847
主程序 finish 5847
5798:5846 [父0] Hello
fork 0 finish , 5846
fork 1 ready , 5846
5846:5849 [子1] Hello
fork 1 finish , 5849
parent() finish 5849
testForWait() finish 5849
主程序 finish 5849
5798:5846 [父1] Hello
fork 1 finish , 5846
parent() finish 5846
testForWait() finish 5846
主程序 finish 5846
hujunyao@HJYUbuntu:~/LinuxCourse/final_lab/t4$

```

图 4.3-7 `os.wait()`函数功能测试

通过绘制下图进行分析，这个测试文件进行了两次循环，第一次遇到分支函数，分成两个进程，两个进程继续执行，父进程遇到 `os.wait()`，需要等待子进程结束再继续运行，不能直接结束。第一层父子进程又遇到分支函数，分成两个进程，父进程遇到 `os.wait()`，需要等待子进程结束再继续运行，不能直接结束。所以，也就能看到主程序被结束了四次，一共有四个线程。第一次循环父子都输出了一次，而第二次循环，父子都输出了两次，一共四次，四个线程退出输出四次。线程结束的顺序是：层数高的先结束、同一层数子进程先结束，也就是主线程最后输出结束。这与 `os.fork()` 和 `os.wait()` 的作用和预期一致，再观察进程和子进程的 id 号也同样能说明 `os.wait()` 函数的作用。下图为 `os.wait()` 测试程序结果分析示意图。

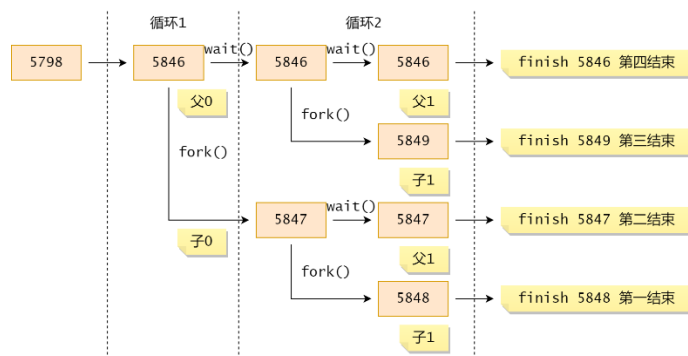


图 4.3-8 `os.wait()`测试程序结果分析示意图

五、附加（20'）

题目：编写一个定时监控某日志文件、监控内存使用、监控 CPU 使用或监控硬盘空间使用的 shell 运维程序（任选其一），要求设定处理门槛，实现及时止损和保护，同时立刻报警。

给出全部程序代码。

5.1 问题分析

本题选择监控 CPU 使用情况，获取 CPU 情况使用 `top` 命令，处理门槛设置为 CPU 占用大于 90%，这里使用 `if` 条件语句判断，止损和保护措施是直接杀死当前占用 CPU 最高的程序，使用 `top`、`tail`、`head` 组合命令获取该程序的进程 id，使用 `kill` 命令杀死进程，报警方式是给用户发送邮件进行提示，使用 `mutt` 命令发送给邮件，使用 `-a` 参数携带上日志附件。定时使用 `crontab -e` 命令，设置为每 15 分钟检测一次。

5.2 程序源代码

监控 CPU 代码 `test.sh` 如下。

```
#!/bin/bash
#提取当前 IP
IP=$(ifconfig ens33 | awk 'NR==2{print $2}')
ip_no_sep=$(echo $IP | sed 's/\./ /g')
#提取当前日期
today=$(date -d "0 day" +%Y 年%m 月%d 日)
today_no_sep=$(date -d "0 day" +%Y%m%d)

filename="log_${ip_no_sep}_${today_no_sep}.txt"

echo "IP 地址: $IP , 日期: $today " > $filename

echo "-----CPU-----" >> $filename
cpu_us=$(top -b -n 1 | grep Cpu | awk '{print $2}')
cpu_sy=$(top -b -n 1 | grep Cpu | awk '{print $4}')
cpu_id=$(top -b -n 1 | grep Cpu | awk '{print $8}')
cpu_wa=$(top -b -n 1 | grep Cpu | awk '{print $10}')
cpu_sum=$(echo 100 $cpu_id | awk '{printf "%.2f\n", $1 - $2}')
cpu_sum_int=$(echo $cpu_sum | awk '{printf "%d\n", $1}')
cpu_max_pid=$(top -b -n 1 | head -8 | tail -1 | awk '{print $1}')
cpu_max_name=$(top -b -n 1 | head -8 | tail -1 | awk '{print $12}')

echo -e "CPU_Sum: ${cpu_sum}% (CPU_Use: ${cpu_us}%, CPU_System: ${cpu_sy}%)"
echo -e "CPU_Idle: ${cpu_id}%"
echo -e "CPU_Wait: ${cpu_wa}%"
```

```

echo -e "CPU_max_pid: ${cpu_max_pid}"
echo -e "CPU_max_name: ${cpu_max_name}"

echo -e "CPU_Sum: ${cpu_sum}% (CPU_Use: ${cpu_us}%, CPU_System:
${cpu_sy}%)" >> $filename
echo -e "CPU_Idle: ${cpu_id}%" >> $filename
echo -e "CPU_Wait: ${cpu_wa}%" >> $filename

echo -e "CPU_max_pid: ${cpu_max_pid}" >> $filename
echo -e "CPU_max_name: ${cpu_max_name}" >> $filename

if [ $cpu_sum_int -ge 90 ];
then
    kill -9 ${cpu_max_pid}
    echo "CPU 占用到达 $cpu_sum_int %了, 请立刻检查!" | mutt
    "1078622540@qq.com" -s "[CPU 监控]${cpu_max_name}对 CPU 占用过高" -a
    $filename
fi

```

定时执行命令如下。

```

crontab -e
0,15,30,45 * * * * bash
/home/hujunyao/LinuxCourse/final_lab/t5/test.sh

```

5.3 运行结果截图

在 Ubuntu 系统上打开火狐浏览器, 连续打开多个页面, 运行程序, 提示当前 CPU 占用率达到 97%, 随后火狐浏览器自动关闭, QQ 邮箱收到带有详细信息的报警邮件。

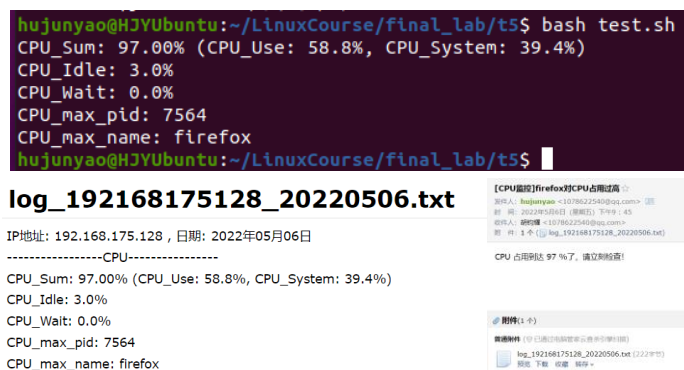


图 5.3-1 日志文件输出测试

将程序设置为每 15 分钟执行一次, 修改定时文件如下所示。

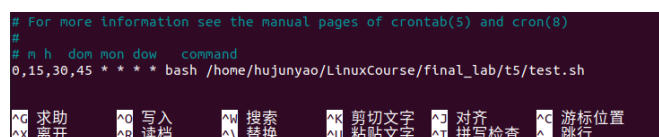


图 5.3-2 定时执行程序设置

六、感悟

这是 Linux 操作系统的最后一次作业，也是结课大作业，总体来说，每个题对应了不同的章节，但是难度也是从简单到复杂，如果光听课可能还是不够，这门课程的实践性质很强，所以还是需要多写代码，才能掌握得很熟练，同时也需要自行在网上查询更多的资料去了解，多和优秀的同学交流经验，可能会有更多收获，比如添加用户的时候，**adduser** 和 **useradd** 看起来差不多，但运行起来还是有一定差别的。

通过这次作业，我不仅巩固了对各种命令的操作和作用的理解，也加深了对操作系统、计算机组成原理等其他课程的一些认识。此外，在做题过程中，也使用了程序流程图、网络拓扑图等图形来描述程序设计的思路，也使用白盒测试的方法对做好的软件进行功能测试，将软件工程、软件测试等思想进行结合，体会到了计算机科学与技术作为一门学科的课程统一性。

参考文献

- [1] 鸟哥. 鸟哥的 Linux 私房菜（第四版）[M].人民邮电出版社,2019.
- [2] 刘遑. Linux 就该这么学[M]. 人民邮电出版社, 2017.
- [3] 菜鸟教程. Linux 教程[OL]. <https://www.runoob.com/linux/linux-tutorial.html>.