基金项目论文

Linux 系统教学中关于套接字文件的解析

赵 宏*，朱忠政，孔东一

（兰州理工大学，兰州 730050）

摘 要： 针对 Linux 系统相关内容教学中对于套接字文件讲述不够详细，导致学生对套接字文件认识模糊的问题，基于套接字通信原理，利用实例对比主机之间和进程之间利用套接字通信的差异，说明 Linux 系统中套接字文件的作用，帮助学生对套接字文件的深入理解。

关键词： Linux 系统；套接字文件；Socket 对象；Python

中图分类号： TP301 **文献标识码：** A **DOI：** 10.3969/j.issn.1003-6970.2020.09.009

本文著录格式： 赵宏，朱忠政，孔东一. Linux 系统教学中关于套接字文件的解析[J]. 软件，2020，41（09）：33-35

Analysis of Socket File in Linux Teaching

ZHAO Hong*, ZHU Zhong-zheng, KONG Dong-yi

(*Lanzhou University of Technology, Lanzhou* 730050)

【Abstract】: Most university students have the vague knowledge of socket file in Linux learning because the detailed explain about this file is absent in Linux teaching files. Based on the principle of socket communication, the differences between hosts and between processes are compared using examples, and the function of socket file in Linux is explained. Therefore, students will gain an in-depth understanding in socket file.

【Key words】: Linux system; Socket file; Socket object; Python

0 引言

Linux 系统作为开放源代码和自由软件的代表，广泛应用在各行各业，运行在各种机型和硬件平台上[1-2]。Linux 系统符合 POSIX（Portable Operating System Interface）标准，功能强大，效率高，配置灵活，安全性高，且具有丰富的工具软件和应用软件，其相关内容在大多数高校信息类专业中作为专业基础课开设[3-4]，例如《Linux 操作系统》、《Linux 系统内核分析》、《Linux 系统程序设计》等。

在 Linux 系统相关内容教学中，Linux 系统中的文件类型是基本内容，大多数教科书列举了 Linux 中的文件类型，包括普通文件（-）、目录文件（d）、字符设备文件（c）、块设备文件（b）、符号链接文件（l）、命名管道文件（p）和套接字文件（s）等七种文件类型[1,4]，对于前五种文件，一般都进行详细讲解，并用实例加以说明。但对于后两种文件，只是进行简单的描述，没有实例的说明，导致学生在学习中，对于命名管道文件和套接字文件的认识很模糊，不利于对 Linux 系统的深刻理解。

文献[5]详细介绍了命名管道文件的功能和实际应用实例，本文首先介绍主机间通过套接字通信的机制和实例，然后讨论进程间通过套接字通信的方式，并通过实例进行详细说明，加深学生对套接字文件的认识。

1 套接字介绍

网络上的主机之间通过 IP 地址与端口号进行通信，称为套接字（Socket）通信[6]。TCP/IP 协议簇中应用层的 HTTP、FTP、DNS 等都是通过套接字通信实现的。套接字通信中，提供服务的一端称为套接字服务端，调用套接字服务的一端称为套接字客户端。套接字服务端首先用自己的 IP 地址、指定端口号和连接方式创建服务并启动服务，监听来自客户端的连接请求；套接字客户端向服务端发起连接请求，连接请求被服务端接受后，双方就可以进行通信。

主机之间通过套接字进行通信时，无论是服务端还是客户端，都需要创建 socket 对象，并设置 family 参数和 type 参数。利用 Python 语言创建 socket 对象的语句格式如下。

s = socket.socket(family 参数, type 参数)

其中，s 表示创建的 socket 对象；socket.socket() 表示调用 socket 模块的 socket()函数；family 参数如表 1 所示，表示主机之间的网络连接方式；type 参数如表 2 所示，表示主机之间通信时所使用的传输协议。

2 主机之间通过套接字通信实例

本实例中，服务端将来自客户端的字符串中的字母转换为大写的服务。

基金项目： 兰州理工大学高教研究项目(GJ2019B-51)，教育部产学合作育人项目(201901197003)

作者简介： 赵宏(1971–)，男，博士，研究方向：深度学习，并行与分布式处理；朱忠政(1999–)，男，本科，研究方向：并行与分布式处理；孔东一(1996–)，男，硕士，研究方向：深度学习，自然语言处理。

表 1 socket()函数的 family 参数
Tab.1 Family parameters of function socket()

family 参数	作用
socket.AF_UNIX	用于 Unix 系统进程之间通信
socket.AF_INET	使用 IPv4 地址通信
socket.AF_INET6	使用 IPv6 地址通信
socket.PF_PACKET	链路层套接字
socket.AF_PACKET	链路层套接字

表 2 socket()函数的 type 参数
Tab.2 Type parameters of function socket()

type 参数	作用
socket.SOCK_STREAM	流式 socket，使用 TCP 传输协议
socket.SOCK_DGRAM	数据报式 socket，使用 UDP 传输协议
socket.SOCK_RAW	原始套接字，可处理 ICMP、IGMP 等网络报文和一些特殊的报文。

假设服务端 IP 地址为 192.168.3.13，在服务端创建 Python 程序文件，socket_s.py，代码如下。

```
1 #!/usr/bin/env python3
2 # coding: utf-8
3 import socket
4 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 s.bind(('192.168.3.13', 8088))
6 s.listen(1)
7 print('Wait for connecting...')
8 (conn,addr)=s.accept()
9 print('conn=',conn)
10 print('addr=',addr)
11 while True:
12     str1=conn.recv(1024)
13     str2=str(str1,encoding='utf-8')
14     print('I received a string is: ',str2)
15     str3=str2.upper()
16     conn.send(str3.encode('utf-8'))
17     if str2 =='.' :
18         break
19 conn.close()
20 s.close()
```

代码前的行号是为叙述方便而加，以#开头的代码为注释，不实际执行。

程序第 3 行引入 socket 模块。第 4 行构造 socket 对象 s，family 参数为 socket.AF_INET，表示主机之间使用 IPv4 地址通信，type 参数为 socket.SOCK_STREAM，表示使用 TCP 传输协议。第 5 行调用函数 bind()将对象 s 绑定到元组('192.168.3.13', 8088)表示的地址上，其中'192.168.3.13'为服务端 IP 地址，8088 为端口号。第 6 行调用函数 listen()开始监听来自客户端的连接，参数为 1 表示只接受 1 个连接。第 8 行调用函数 accept()接受一个来自客户端的连接，返回元组(conn,addr)，其中，conn 也是一个 socket 对象，用来与客户端通信，addr 为元组变量，保存客户端的 IP 地址和端口号。第 11 行至 18 行的循环使用 conn 通过函数 recv()和 send()与客户端通信，recv()函数使用参数 1024，表示 1 次最多接收 1024 字节数据。由于通信双方交换 bytes 字节流数据，因此，第 13 行利用 str()函数将 bytes 字节流数据转换为字符串。第 15 行调用函数 upper()将字符串中小写字母转换为大写字母。第 16 行调用函数 send()发送数据之前，利用函数 encode()将字符串转换为 bytes 字节流后进行发送。第 17 行判断接收到的来自客户端的字符串是否为结束标志“.”，若收到结束标志则利用 break 语句退出循环。第 19 行调用函数 close()断开连接，第 20 行调用函数 close()释放对象 s。

在客户端创建 Python 程序文件，socket_c.py，代码如下。

```
1 #!/usr/bin/env python3
2 # coding: utf-8
3 import socket
4 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 s.connect(('192.168.3.13',8088))
6 print('I am connecting the server!')
7 for xx in ['aBch','f 服务 d','h7Tq','.']:
8     s.send(xx.encode('utf-8'))
9     str1=s.recv(1024)
10    str2=str(str1,encoding='utf-8')
11    print('The original string is:',xx,'\tthe processed string is:',str2)
12 s.close()
```

程序第 3 行引入 socket 模块。第 4 行构造 socket 对象 s，family 参数和 type 参数与服务端相同。第 5 行调用函数 connect()连接服务端，服务端的 IP 地址和端口号用元组表示。第 6 行打印提示信息。第 7 行至第 11 行的循环向服务器发送要处理的数据和接收处理完毕的数据。与服务段程序类似，传输的数据格式为 bytes 字节流，因此，在数据发送前和接收数据后，需要对数据格式进行转换。第 12 行调用函数 close()断开与服务段的连接。

在服务端运行程序 socket_s.py，在客户端运行程序 socket_c.py，服务端和客户端主机将通过套接字进行通信，服务端程序运行结果如图 1 所示，客户端程序运行结果如图 2 所示。

```
Wait for connecting...
conn= <socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('192.168.3.13', 8088), raddr=('192.168.3.37', 45542)>
addr= ('192.168.3.37', 45542)
I received a string is:  aBch
I received a string is:  f 服务 d
I received a string is:  h7Tq
I received a string is:  .
```

图 1 服务端程序运行结果
Fig.1 Executive result of server program

```
I am connecting the server!
The original string is: aBch     the processed string is: ABCH
The original string is: f 服务 d     the processed string is: F 服务 D
The original string is: h7Tq     the processed string is: H7TQ
The original string is: .     the processed string is: .
```

图 2 客户端程序运行结果
Fig.2 Executive result of client program

从图 1 和图 2 可知，服务端为客户端提供字符转换服务，客户端 IP 地址为 192.168.3.37，端口号为 45542。

3　进程之间通过套接字通信实例

从上述实例可知，主机之间通过套接字通信时使用由 IP 地址和端口号组成的元组。如果要实现进程之间通过套接字通信，则需要使用套接字文件，并且，通信双方创建套接字对象时，family 参数设置为 socket.AF_UNIX。

还是以服务端为客户端提供字符串转换服务的程序为例，说明进程之间通过套接字通信的过程。

创建服务端 Python 程序文件，socket_s_p.py，代码如下。

```
1 #!/usr/bin/env python3
2 # coding: utf-8
3 import socket
4 s = socket.socket(socket.AF_UNIX, socket.SOCK_ STREAM)
5 s.bind('a.socket')
6 s.listen(1)
7 print('Wait for connecting...')
8 (conn,addr)=s.accept()
9 print('conn=',conn)
10 print('addr=',addr)
11 while True:
12     str1=conn.recv(1024)
13     str2=str(str1,encoding='utf-8')
14     print('I received a string is: ',str2)
15     str3=str2.upper()
16     conn.send(str3.encode('utf-8'))
17     if str2 =='.' :
18         break
19 conn.close()
20 s.close()
```

程序第 4 行创建 socket 对象函数的 family 参数设置为 socket.AF_UNIX，表示该 socket 对象将用于进程之间的通信。第 5 行用文件名 a.socket 代替由 IP 地址和端口号组成的元组，表示进程之间将通过套接字文件 a.socket 进行通信。

创建客户端 Python 程序文件，socket_c_p.py，代码如下。

```
1 #!/usr/bin/env python3
2 # coding: utf-8
3 import socket
4 s = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
5 s.connect('a.socket')
6 print('I am connecting the server!')
7 for xx in ['aBch','f 服务 d','h7Tq','.']:
8     s.send(xx.encode('utf-8'))
9     str1=s.recv(1024)
10    str2=str(str1,encoding='utf-8')
11    print('The original string is:',xx,'\tthe processed string is:',str2)
12 s.close()
```

与服务端程序类似，程序第 4 行创建 socket 对象函数的 family 参数设置为 socket.AF_UNIX，表示该 socket 对象将用于进程之间的通信。第 5 行用文件名 a.socket 代替由 IP 地址和端口号组成的元组，表示进程之间将通过套接字文件 a.socket 进行通信。

在不同窗口分别运行服务端程序 socket_s_p.py 和客户端程序 socket_c_p.py，将分别创建服务端进程和客户端进程，这两个进程之间将通过套接字进行通信，服务端进程运行结果如图 3 所示，客户端进程运行结果如图 4 所示。

```
Wait for connecting...
conn= <socket.socket fd=4, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=a.socket>
addr=
I received a string is:   aBch
I received a string is:   f 服务 d
I received a string is:   h7Tq
I received a string is:   .
```

图 3　服务端进程运行结果
Fig.3　Executive result of server process

```
I am connecting the server!
The original string is: aBch     the processed string is: ABCH
The original string is: f 服务 d     the processed string is: F 服务 D
The original string is: h7Tq     the processed string is: H7TQ
The original string is: .     the processed string is: .
```

图 4　客户端进程运行结果
Fig.4　Executive result of client process

对比图 1 和图 3，图 2 和图 4 可知，图 1 和图 3 稍有差异，差异在于图 3 中用 a.socket 代替了图 1 中由服务端的 IP 地址和端口号组成的元组；图 3 中无 raddr 且 addr 值为空。图 2 和图 4 完全一样。说明进程之间仿照主机之间利用套接字进行通信，用套接字文件代替主机之间通信时所用的元组。

运行文件 socket_s_p.py，将在当前目录下创建套接字文件 a.socket，与命令管道文件类似，套接字文件的大小也为 0，也遵循 Linux 系统对文件的权限规定。

再次运行文件 socket_s_p.py，将给出错误提示“OSError: [Errno 98] Address already in use”，表示进程之间利用套接字通信时，每次都需要创建新的套接字文件，且不覆盖已经存在的同名套接字文件。只有先删除套接字文件 a.socket，socket_s_p.py 文件才可再次运行。

4　结束语

套接字文件是 Linux 系统的七种文件之一，也是进程之间通信的一种手段，在 Linux 系统中具有重要作用。本文通过对比主机之间和进程之间通过套接字通信的不同，说明 Linux 系统中套接字文件的作用，帮助学生深入理解套接字文件。

参考文献

[1] 鸟哥. 鸟哥的Linux私房菜基础学习篇(第四版)[M]. 北京: 人民邮电出版社, 2018, 10.

[2] Machtelt Garrels. Introduction to Linux[EB/OL]. (2010-05-12) [2019-09-27]. http://tille.garrels.be/training/tldp/.

[3] 燕彩蓉, 朱黎华, 刘瑜琪, 等. 新工科背景下Linux系统课程教学研究[J]. 计算机教育, 2019(6): 152-156.

[4] 吴淑泉. 高校“Linux操作系统”课程教学研究与探索[J]. 教育理论与实践, 2017, 37(33): 57-58.

[5] 赵宏, 朱忠政, 常兆斌. Linux系统教学中关于命名管道文件的解析[J]. 软件, 2020, 41(02): 108-110.

[6] 赵宏, 包广斌, 马栋林. Python网络编程(Linux)[M]. 北京: 清华大学出版社, 2018, 10.