管道通信实验

程序编译命令

交叉编译命令

```
mknod FIFO1 p
mknod FIFO2 p
```

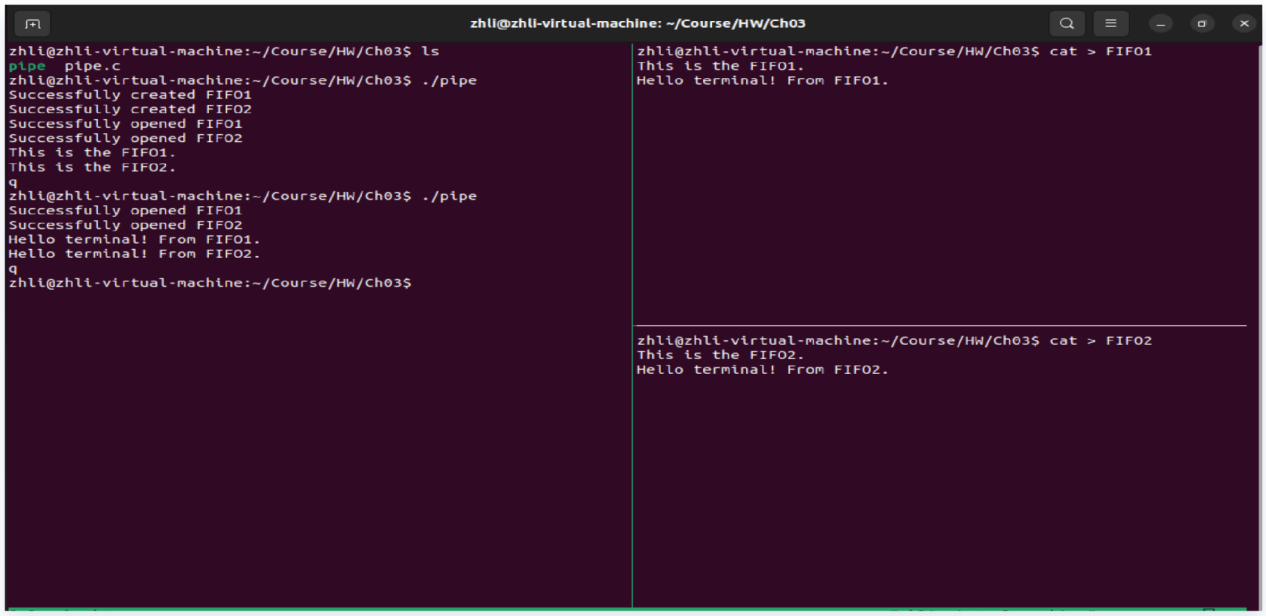
GCC 编译命令

```
gcc pipe.c -o pipe
```

程序运行

```
cat > FIFO1
This is the FIFO1.
Hello, terminal! From FIFO1.
cat > FIFO2
This is the FIFO2.
Hello, terminal! From FIFO2.
```

运行结果



完整源代码

pipe.c

```c
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>
#include <time.h>

#define FIFO1 "FIFO1"      // 管道1
#define FIFO2 "FIFO2"      // 管道2
#define BUFFER_SIZE 2048   // 缓冲区大小
#define INPUT_FILES 3      // 输入文件描述符个数
#define MAX_TIME 100        // 最大超时时间
#define max(a, b) ((a) > (b) ? (a) : (b))

int main(void)
{
    int fds[INPUT_FILES];             // 管道描述符
    char buffer[BUFFER_SIZE];

    fd_set Input_Set, tmp_inset;      // 文件描述符集
    struct timeval timer;             // 计时器

```

```
26        // ----------- Step 1 创建三个文件描述符 -----------
27        // 标准输入文件描述符
28        fds[0] = 0;
29
30        // 两个有名管道文件描述符
31        if (access(FIFO1, F_OK) == -1)
32        {
33            if ((mkfifo(FIFO1, 0666) < 0) && (errno != EEXIST))
34            {
35                printf("Error for creating FIFO1\n");
36                exit(1);
37            }else
38                printf("Successfully created FIFO1\n");
39        }
40        if (access(FIFO2, F_OK) == -1)
41        {
42            if ((mkfifo(FIFO2, 0666) < 0) && (errno != EEXIST))
43            {
44                printf("Error for creating FIFO2\n");
45                exit(1);
46            }else
47                printf("Successfully created FIFO2\n");
48        }
49
50        // 非阻塞打开管道文件
51        if ((fds[1] = open(FIFO1, O_RDONLY | O_NONBLOCK)) < 0)
52        {
53            printf("Error for opening FIFO1\n");
54            return 1;
55        }else
56            printf("Successfully opened FIFO1\n");
57
58        if ((fds[2] = open(FIFO2, O_RDONLY | O_NONBLOCK)) < 0)
59        {
60            printf("Error for opening FIFO2\n");
61            return 1;
62        }else
63            printf("Successfully opened FIFO2\n");
64
65
66        // 获取最大的文件描述符
67        int maxfd = max(max(fds[0], fds[1]), fds[2]);
68
69        // ----------- Step 2 初始化读文件描述符集合 -----------
70        FD_ZERO(&Input_Set);
71        for (int i = 0; i < INPUT_FILES; i++)
```

```
72          {
73              FD_SET(fds[i], &Input_Set);     // 添加到集合中
74          }
75
76          // ----------- Step 3 设置超时时间 -----------
77          timer.tv_sec = MAX_TIME;
78          timer.tv_usec = 0;
79
80          // 文件描述符是否准备就绪
81          while (FD_ISSET(fds[0], &Input_Set) || FD_ISSET(fds[1], &Input_Set)
    || FD_ISSET(fds[2], &Input_Set))
82          {
83              tmp_inset = Input_Set;     // 读文件描述符集合重置
84
85              // ----------- Step 4 select函数监视文件描述符集合的文件  -----------
86              int res = select(maxfd + 1, &tmp_inset, NULL, NULL, &timer);
87              switch (res)
88              {
89                  case -1:
90                  {
91                      printf("Error for selecting pipe\n");
92                      return 1;
93                  }
94                  break;
95                  case 0:
96                  {
97                      printf("Time out for selecting pipe\n");
98                      return 1;
99                  }
100                 break;
101                 default:
102                 {
103                     for (int i = 0; i < INPUT_FILES; i++)
104                     {
105                         if (FD_ISSET(fds[i], &tmp_inset))
106                         {
107                             memset(buffer, 0, BUFFER_SIZE);
108                             int real_read = read(fds[i], buffer,
    BUFFER_SIZE);
109                             if (real_read < 0)
110                             {
111                                 if (errno != EAGAIN)
112                                 {
113                                     return 1;
114                                 }
115                             }
```

```
                        else if (!real_read)
                        {
                            close(fds[i]);
                            FD_CLR(fds[i], &Input_Set);
                        }
                        else
                        {
                            // 标准输入
                            if (i == 0)
                            {
                                if ((buffer[0] == 'q') || (buffer[0] == 'Q'))
                                {
                                    return 1;
                                }
                            }
                            else
                            {
                                buffer[real_read] = '\0';
                                printf("%s", buffer);
                            }
                        }
                    }
                }
            }
            break;
        }
    }

    return 0;
}
```