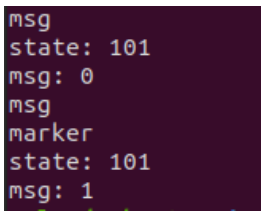
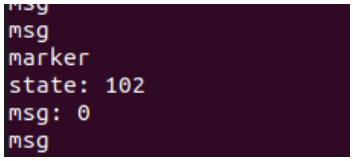
1. 实现全局状态的快照算法，并监控下列程序：两个进程 P 和 Q 用两个通道连成一个环，它们不断地轮转消息 m。在任何一个时刻，系统中仅有一份 m 的拷贝。每个进程的状态是指由它接收到 m 的次数。P 首先发送 m。在某一点，P 得到消息且它的状态是 101。在发送 m 之后，P 启动快照算法，要求记录由快照算法报告的全局状态。

   P 进程首先通过 socket 向 Q 发送 msg 消息，Q 收到后回发给 P 进程。在 P 收到第 101 个 msg 时，再发送一个 msg，启动快照。此时记录下状态为 101，通道上为 0。向 Q 发送一个 marker 消息。
   Q 首先收到 P 在 marker 之前的 msg 消息，此时状态由 101 改为 102，回复 msg。收到 marker 消息后，记录当前状态为 102，通道上为 0。向 P 发送 marker。
   P 首先收到 msg，将通道上由空改为一个 msg 消息。快照终止。
   运行效果如下：
   P:




   Q:




   最终状态为
   P<101>  Pc<1msg>
   Q<102>  Qc<NULL>

   P:

```
import socket                    # 导入 socket 模块
import time
s = socket.socket()              # 创建 socket 对象
host = socket.gethostname() # 获取本地主机名
port = 12346                     # 设置端口
s.bind((host, port))             # 绑定端口
s.listen(5)                      # 等待客户端连接
c,addr = s.accept()        # 建立客户端连接
c.send('msg'.encode())
time.sleep(2)
start_count = 101
recv_count = 0
msg_count = 0
keep_count = 0
```

```
while True:
    msg = c.recv(1024).decode()
    print(msg)
    if(msg == 'msg'):
        recv_count +=1
        msg_count += 1
        c.send('msg'.encode())
    if(msg == 'marker'):
        print('state:',keep_count)
        print('msg:',msg_count)
        exit()
    if recv_count == start_count:
        c.send('marker'.encode())
        print('state:',recv_count)
        keep_count = recv_count
        msg_count = 0
        print('msg:',msg_count)
    time.sleep(0.1)
c.close()                    # 关闭连接
```

Q:

```
import socket                    # 导入 socket 模块
import time
s = socket.socket()              # 创建 socket 对象
host = socket.gethostname() # 获取本地主机名
port = 12346                     # 设置端口号
s.connect((host, port))
recv_count = 0
msg_count = 0
while True:
    msg = s.recv(1024).decode()
    print(msg)
    if(msg == 'msg'):
        recv_count +=1
        msg_count += 1
        s.send('msg'.encode())
    if(msg == 'marker'):
        msg_count = 0
        print('state:',recv_count)
        print('msg:',msg_count)
        s.send('marker'.encode())
    time.sleep(0.1)
s.close()
```