Systems Software

Week 6: IPC and Pipes (Continued)

Notes By: Jonathan McCarthy

Overview

- Inter Process Communication (IPC)
- Message Queues
- Intro to Shared Memory

Message Queues

- A message queue operates as a linked list of messages
- The messages are stored in the system kernel
- Each queue has a unique identifier (queue ID or name)
- Implementations of message queues can vary for different types of environments.

Different Implementations

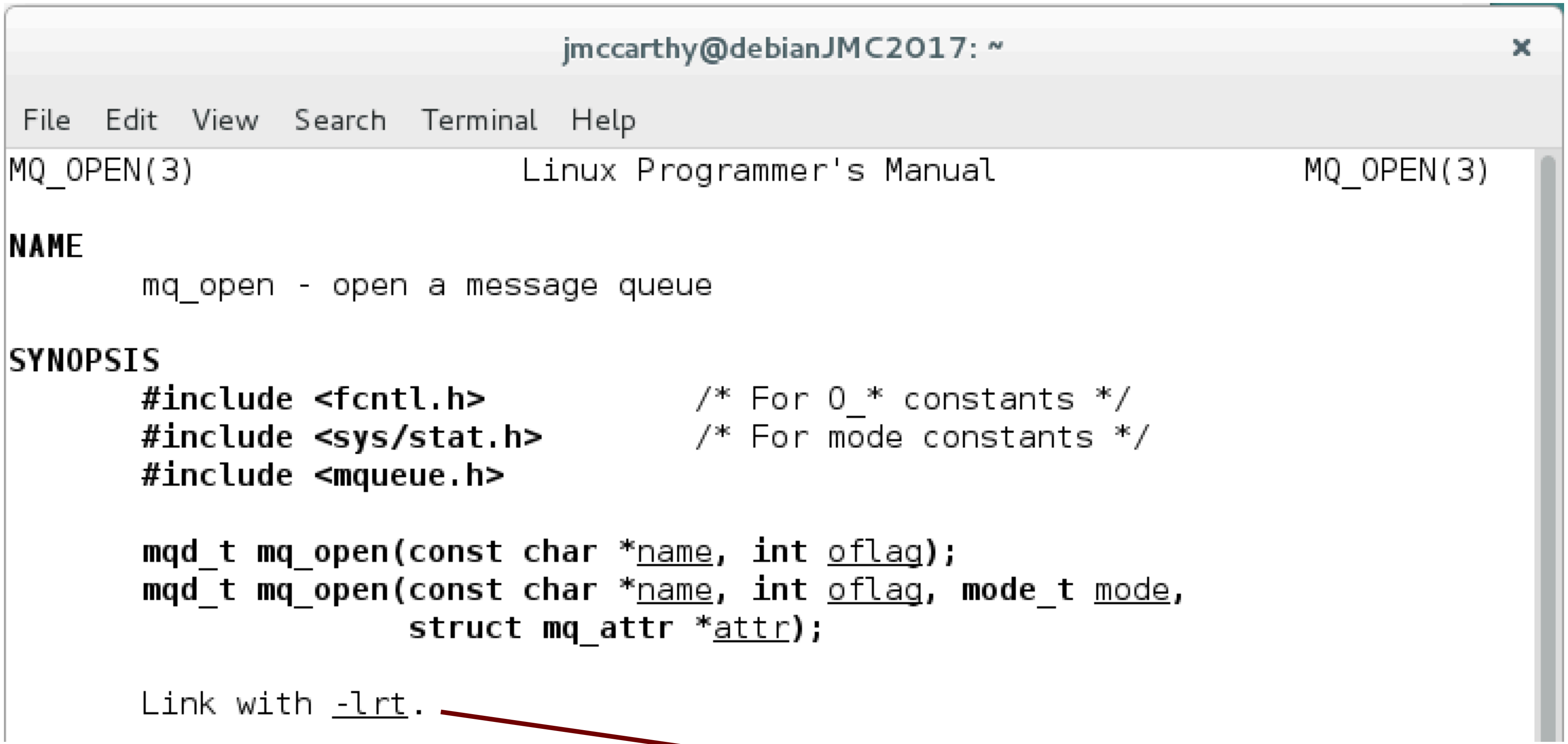
- POSIX message queues allow processes to exchange data in the form of messages.
- This API is distinct from that provided by System V message queues (msgget(2), msgsnd(2), msgrcv(2), etc.), but provides similar functionality.

Source: Linux Man Pages

Creating a Queue

- A Queue is created using mq_open()
- The return from creating the queue is a file descriptor.
- This needs to be stored as type mqd_t, this will be used to access the queue for all subsequent calls.
- Each queue will be given a unique name.
- This is provided when creating the queue.
- See **man mq_open** or **man mq_overview** for more details

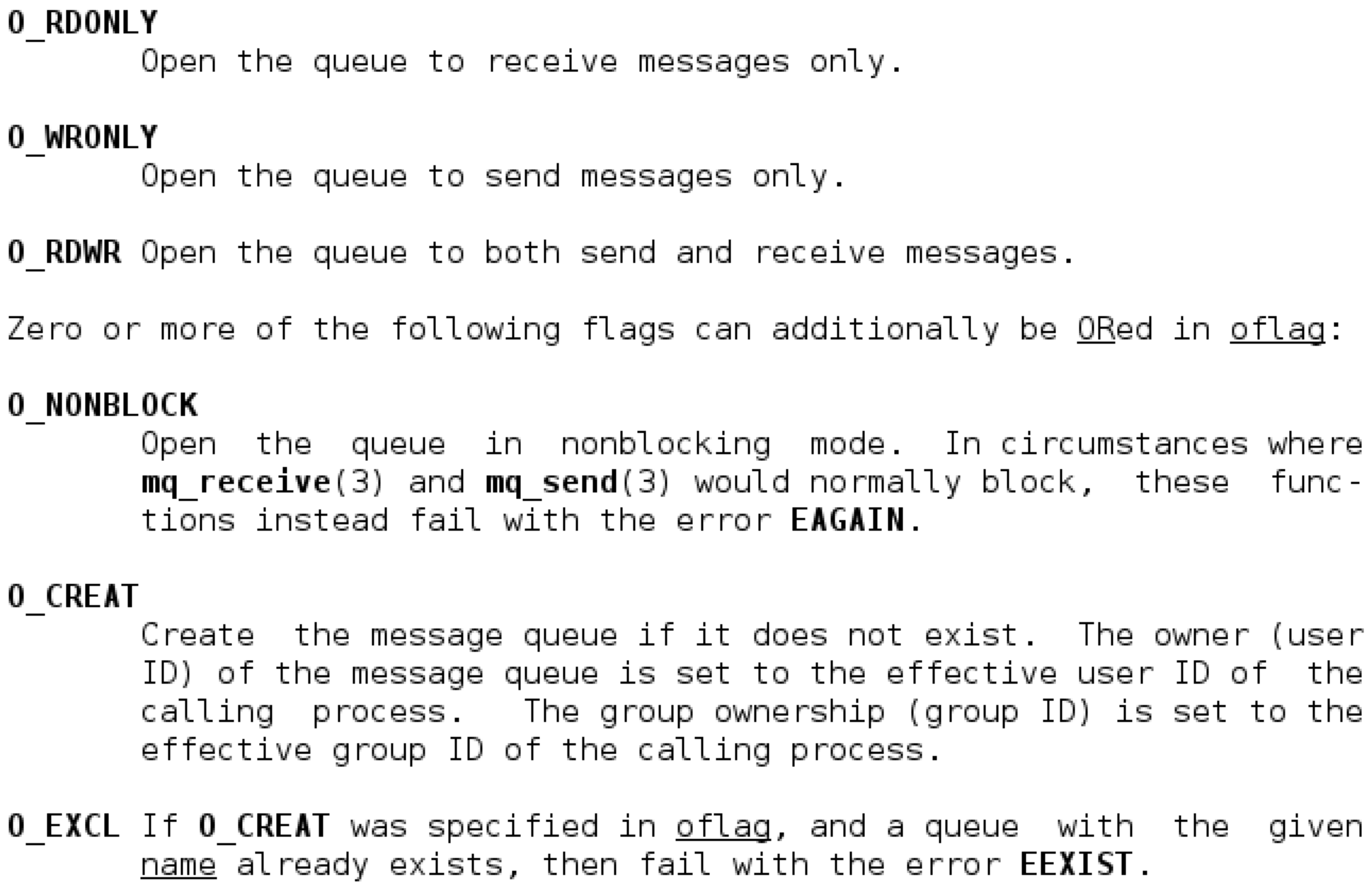
Mq_open()


jmccarthy@debianJMC2017: ~
File Edit View Search Terminal Help

```
MQ_OPEN(3)                 Linux Programmer's Manual                MQ_OPEN(3)

NAME
       mq_open - open a message queue

SYNOPSIS
       #include <fcntl.h>           /* For O_* constants */
       #include <sys/stat.h>        /* For mode constants */
       #include <mqueue.h>

       mqd_t mq_open(const char *name, int oflag);
       mqd_t mq_open(const char *name, int oflag, mode_t mode,
                     struct mq_attr *attr);

       Link with -lrt.
```

Compile with the –lrt flag

Mq_open – Options



```
O_RDONLY
       Open the queue to receive messages only.

O_WRONLY
       Open the queue to send messages only.

O_RDWR Open the queue to both send and receive messages.

Zero or more of the following flags can additionally be ORed in oflag:

O_NONBLOCK
       Open  the  queue  in  nonblocking  mode.  In circumstances where
       mq_receive(3) and mq_send(3) would normally block,  these  func-
       tions instead fail with the error EAGAIN.

O_CREAT
       Create  the message queue if it does not exist.  The owner (user
       ID) of the message queue is set to the effective user ID of  the
       calling  process.   The group ownership (group ID) is set to the
       effective group ID of the calling process.

O_EXCL If O_CREAT was specified in oflag, and a queue  with  the  given
       name already exists, then fail with the error EEXIST.
```

Sending and Receiving Messages with the Queue

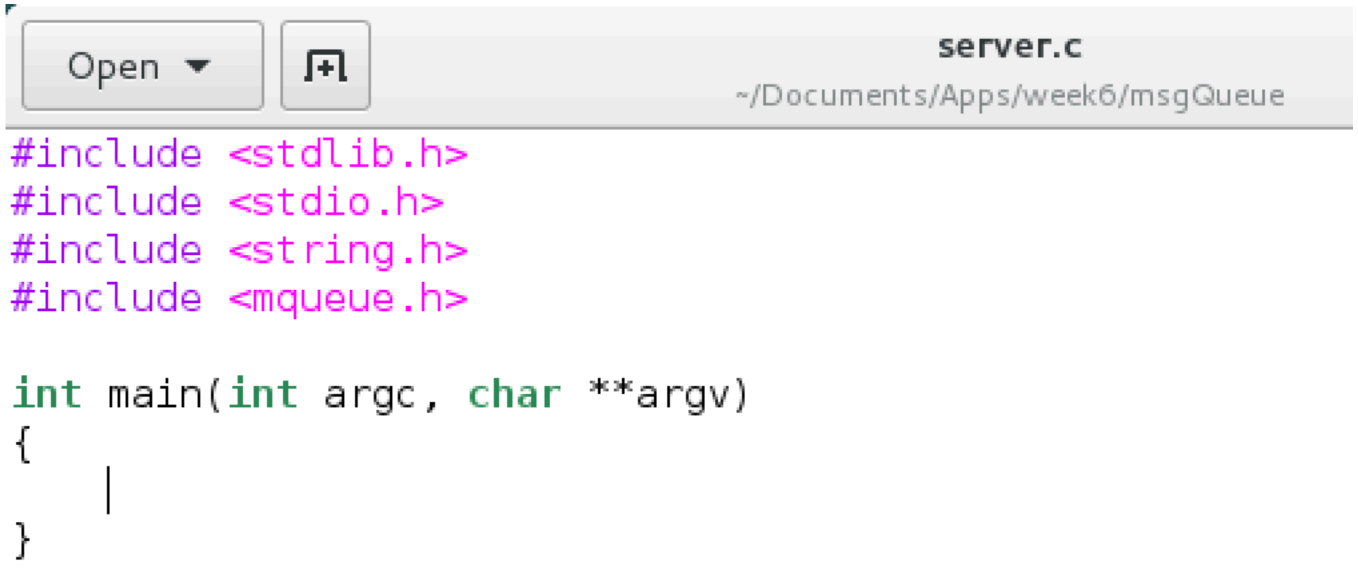
- With the queue setup:
  - messages can be sent with mq_send
  - messages can be received with mp_receive

Example

- For the Message Queue we will be creating a server program to manage the operation of the queue.
  - Server.c
- A client program will be created to show the operation of the queue!!
  - Client.c

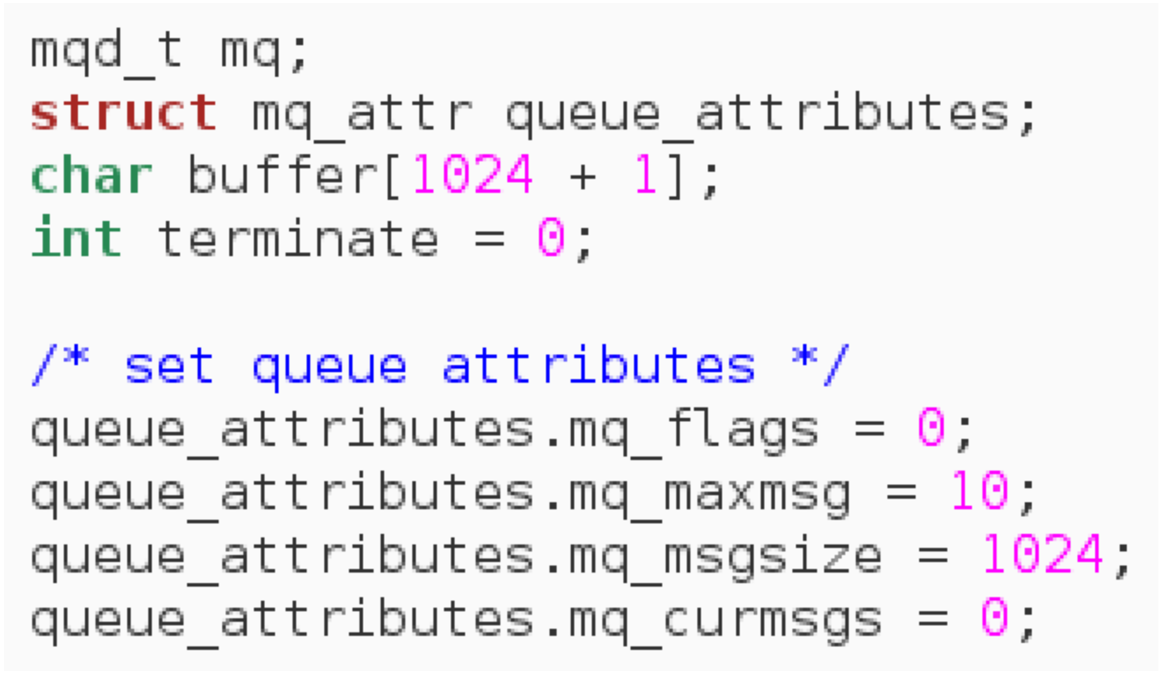
Setup the Server

- Create a file named server.c



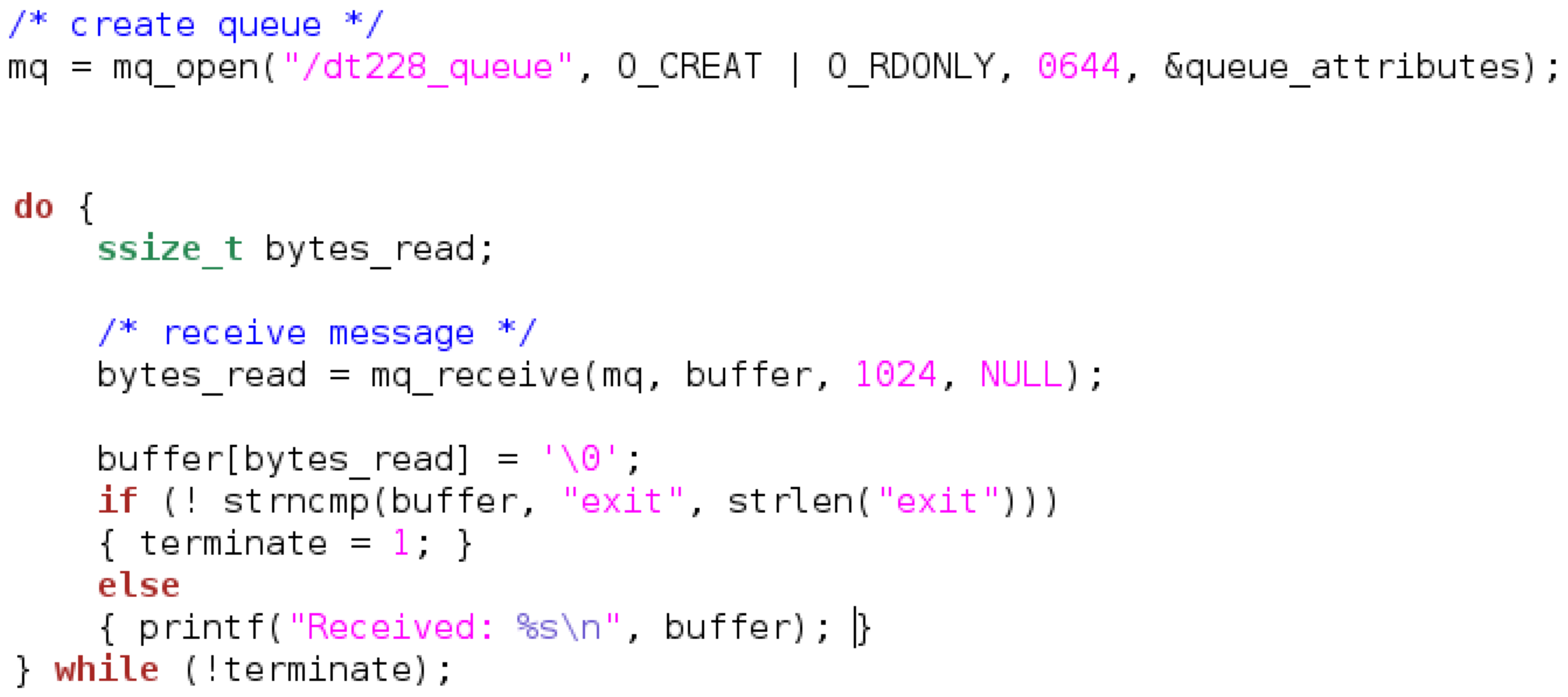
```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <mqueue.h>

int main(int argc, char **argv)
{
    
}
```

Setup the Server



```c
mqd_t mq;
struct mq_attr queue_attributes;
char buffer[1024 + 1];
int terminate = 0;

/* set queue attributes */
queue_attributes.mq_flags = 0;
queue_attributes.mq_maxmsg = 10;
queue_attributes.mq_msgsize = 1024;
queue_attributes.mq_curmsgs = 0;
```

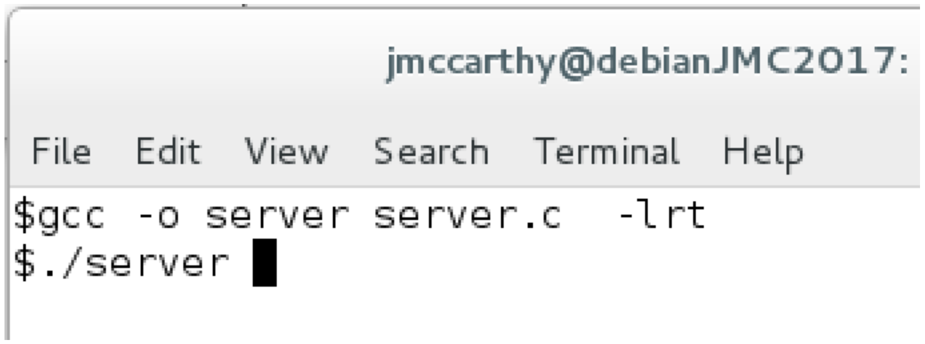
Setup the Server



```
/* create queue */
mq = mq_open("/dt228_queue", O_CREAT | O_RDONLY, 0644, &queue_attributes);


do {
    ssize_t bytes_read;

    /* receive message */
    bytes_read = mq_receive(mq, buffer, 1024, NULL);

    buffer[bytes_read] = '\0';
    if (! strncmp(buffer, "exit", strlen("exit")))
    { terminate = 1; }
    else
    { printf("Received: %s\n", buffer); }
} while (!terminate);
```

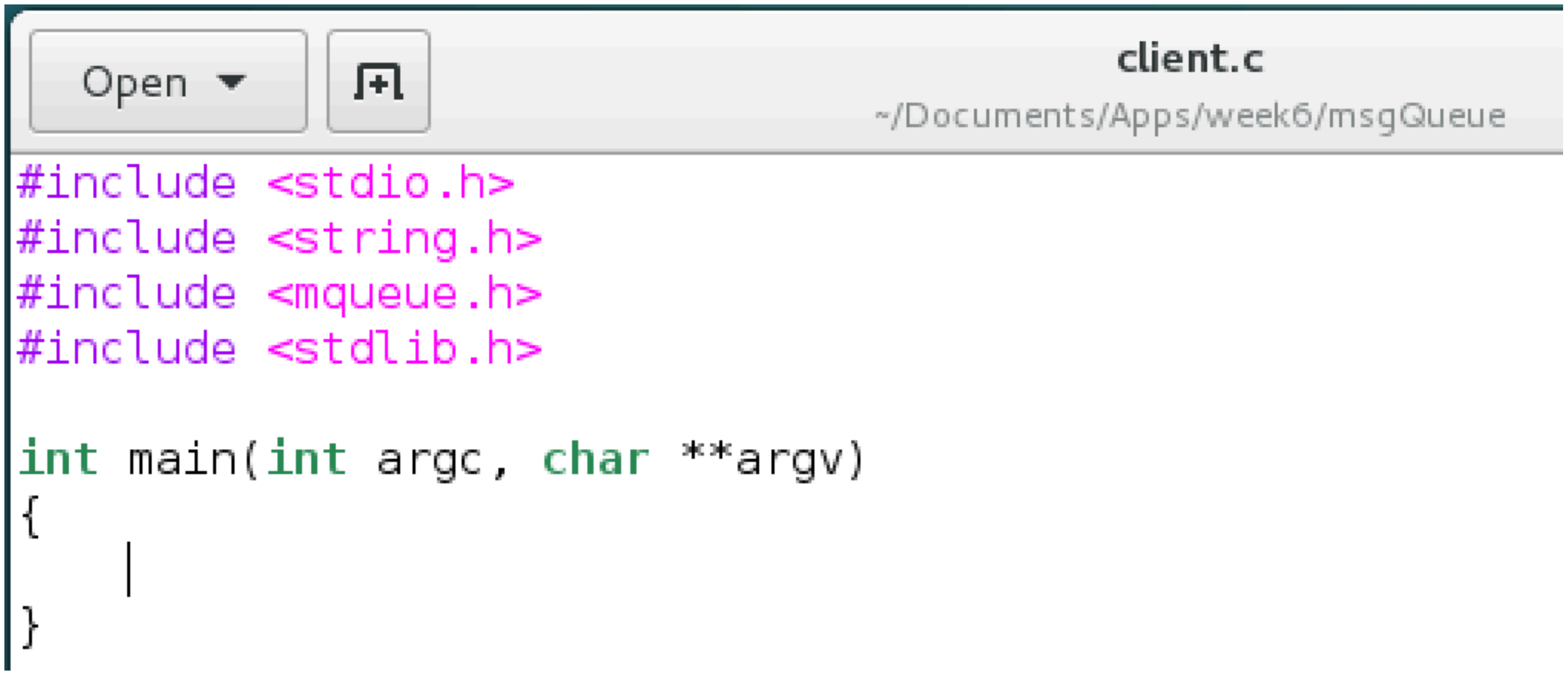
Setup the Server

```
mq_close(mq);
mq_unlink("/dt228_queue");
return 0;
```



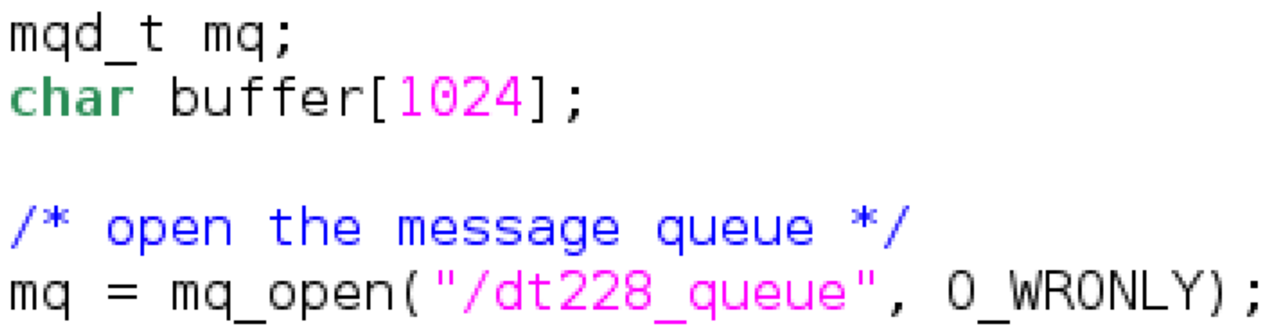
```
$gcc -o server server.c  -lrt
$./server
```

Setup the Client



```c
#include <stdio.h>
#include <string.h>
#include <mqueue.h>
#include <stdlib.h>

int main(int argc, char **argv)
{
    
}
```

Setup the Client



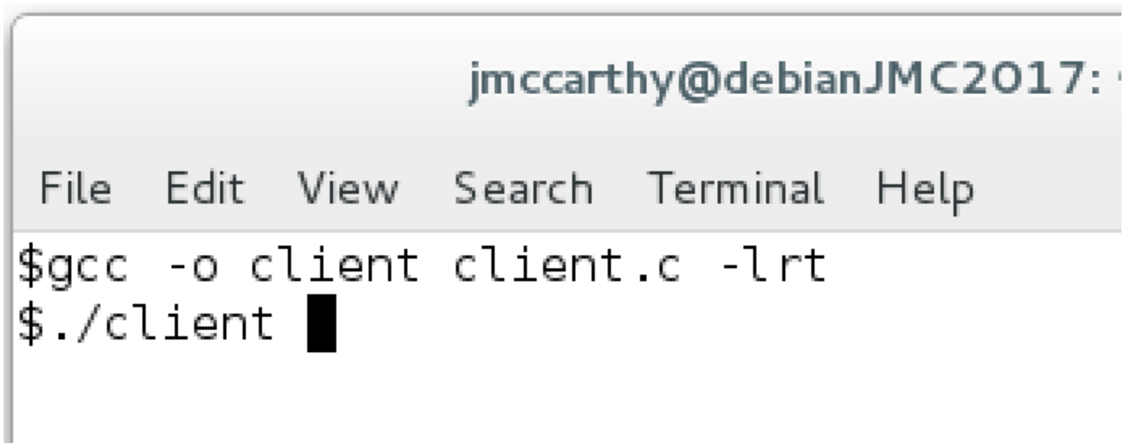
```
mqd_t mq;
char buffer[1024];

/* open the message queue */
mq = mq_open("/dt228_queue", O_WRONLY);
```

Setup the Client

```
printf("Send message to server (enter 'exit' to terminate):\n");

do {
    printf(">> ");
    fflush(stdout);

    memset(buffer, 0, 1024);
    fgets(buffer, 1024, stdin);
    mq_send(mq, buffer, 1024, 0);

} while (strncmp(buffer, "exit", strlen("exit")));
```

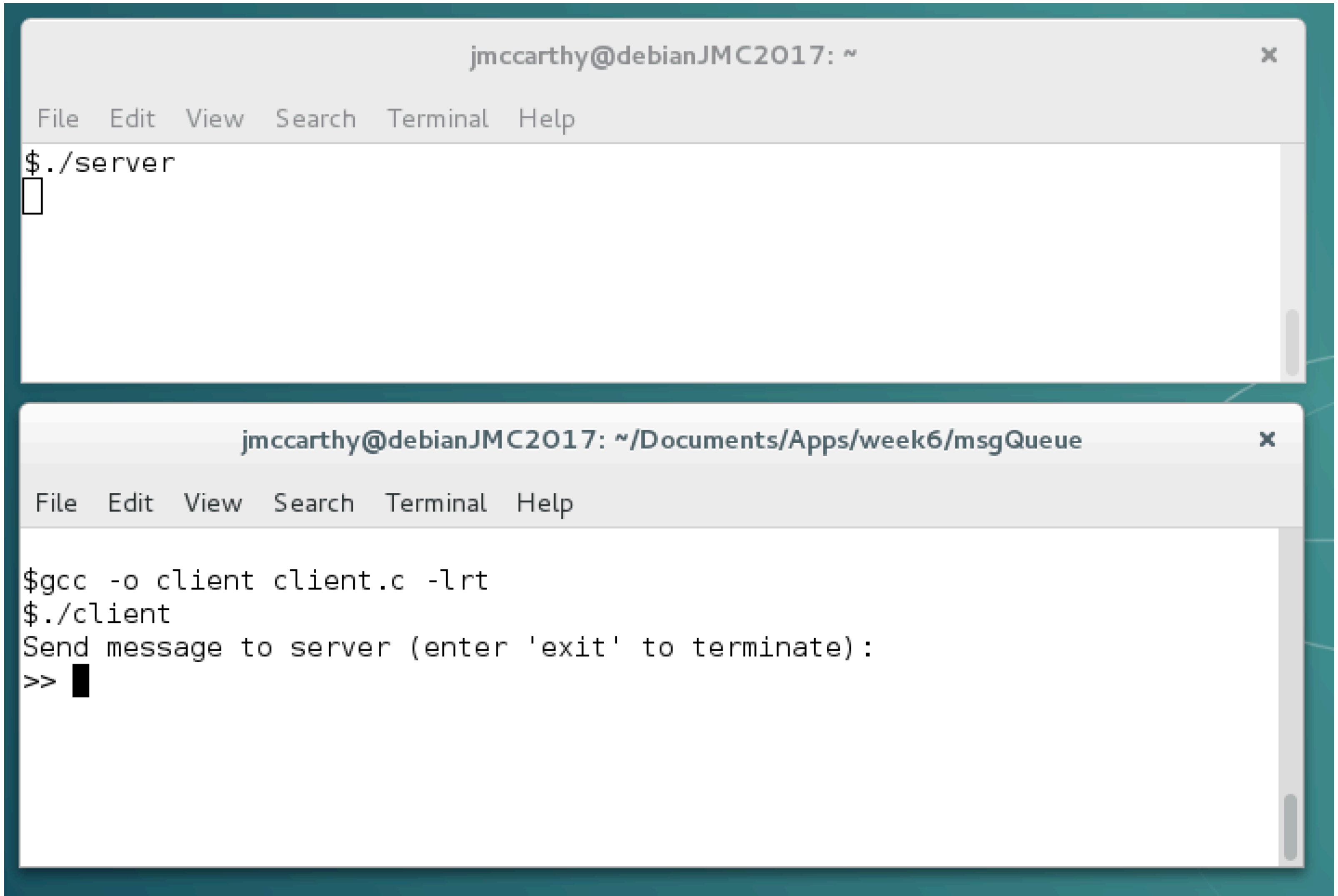
Setup the Client

```
mq_close(mq);
return 0;
```



```
$gcc -o client client.c -lrt
$./client
```

Run the Example

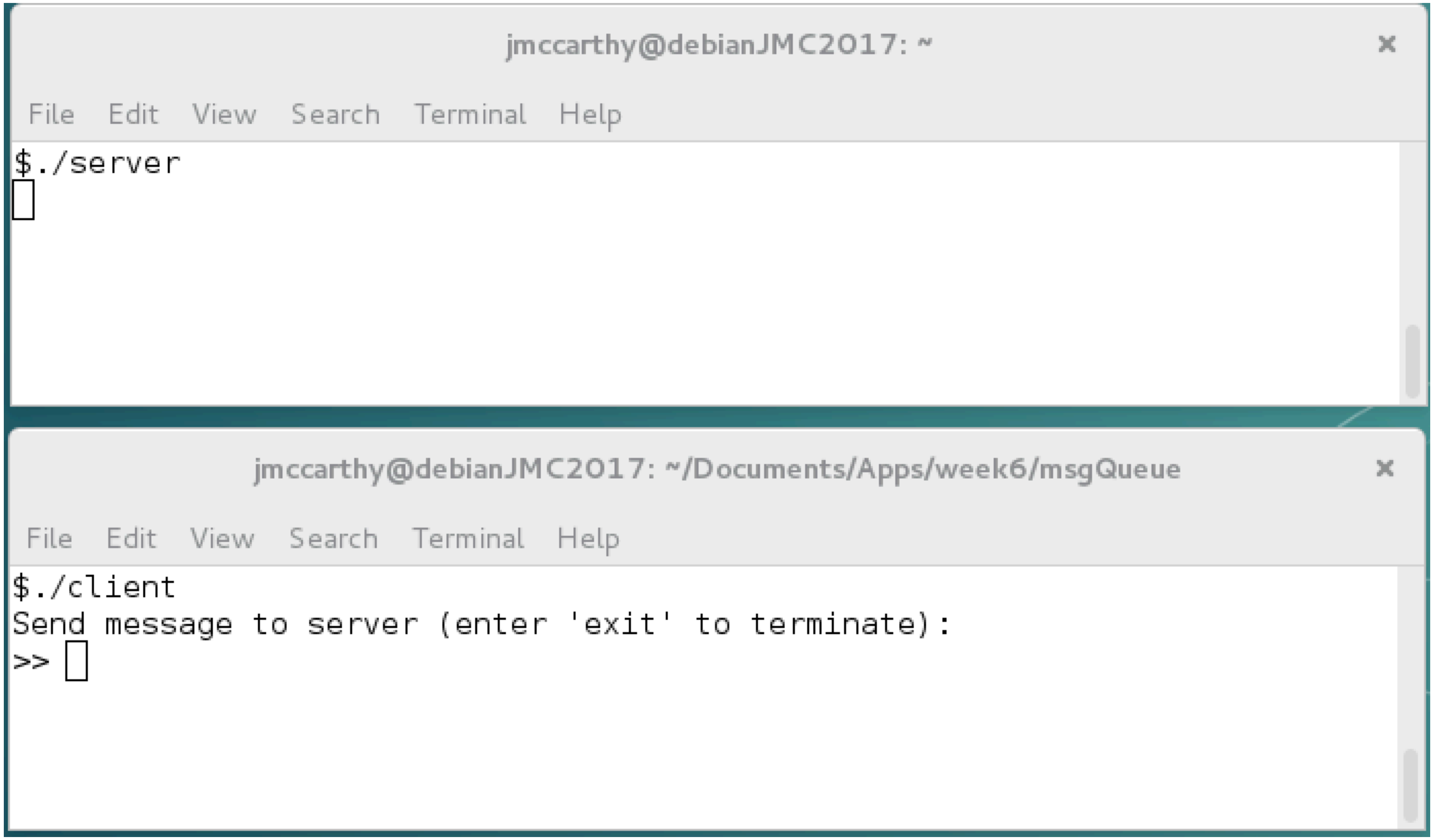

jmccarthy@debianJMC2017: ~
File Edit View Search Terminal Help

```
$./server
```

jmccarthy@debianJMC2017: ~/Documents/Apps/week6/msgQueue
File Edit View Search Terminal Help

```
$gcc -o client client.c -lrt
$./client
Send message to server (enter 'exit' to terminate):
>>
```

Run the Example

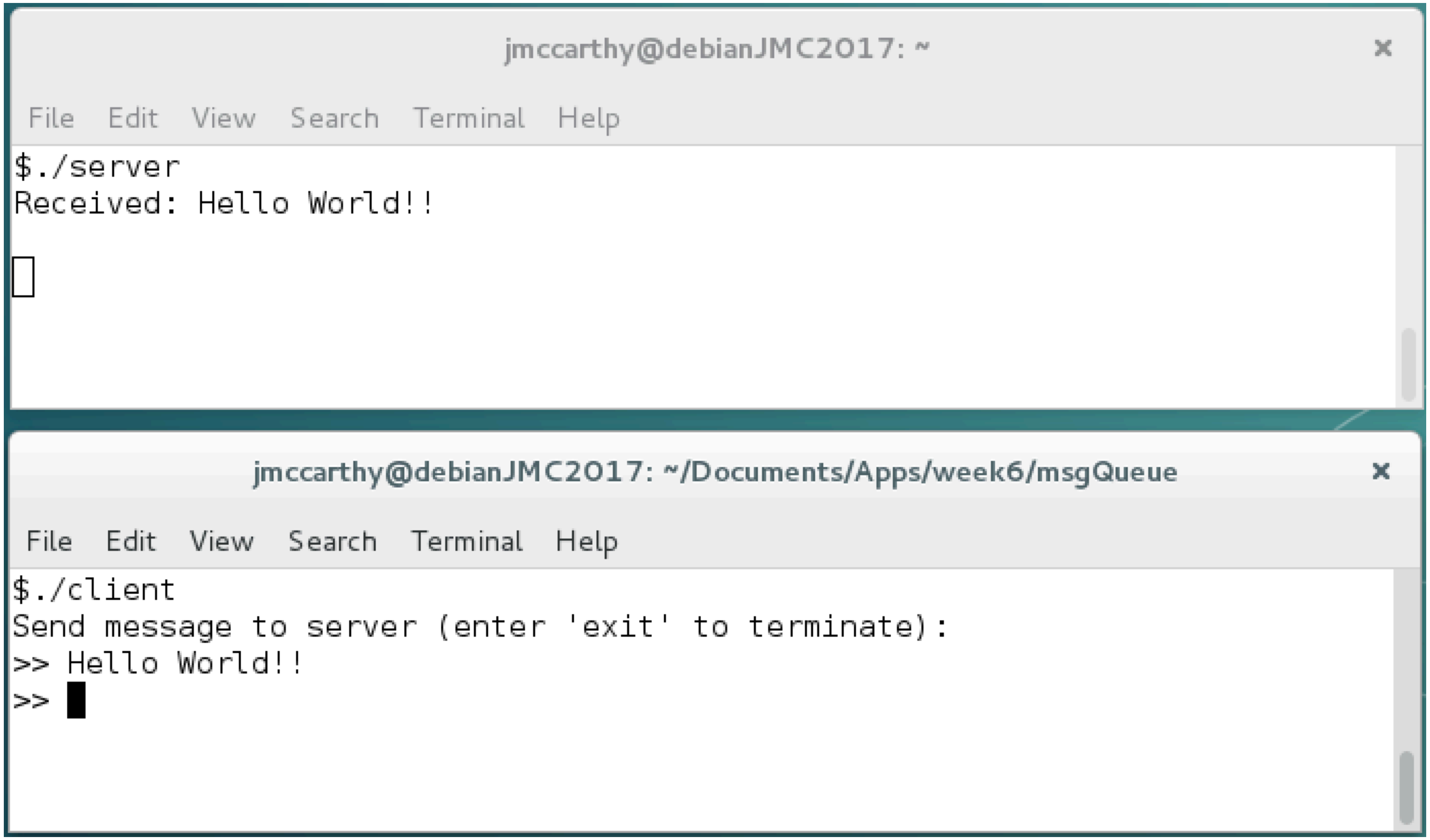

jmccarthy@debianJMC2017: ~
File Edit View Search Terminal Help

```
$./server
```

jmccarthy@debianJMC2017: ~/Documents/Apps/week6/msgQueue
File Edit View Search Terminal Help

```
$./client
Send message to server (enter 'exit' to terminate):
>>
```

Run the Example

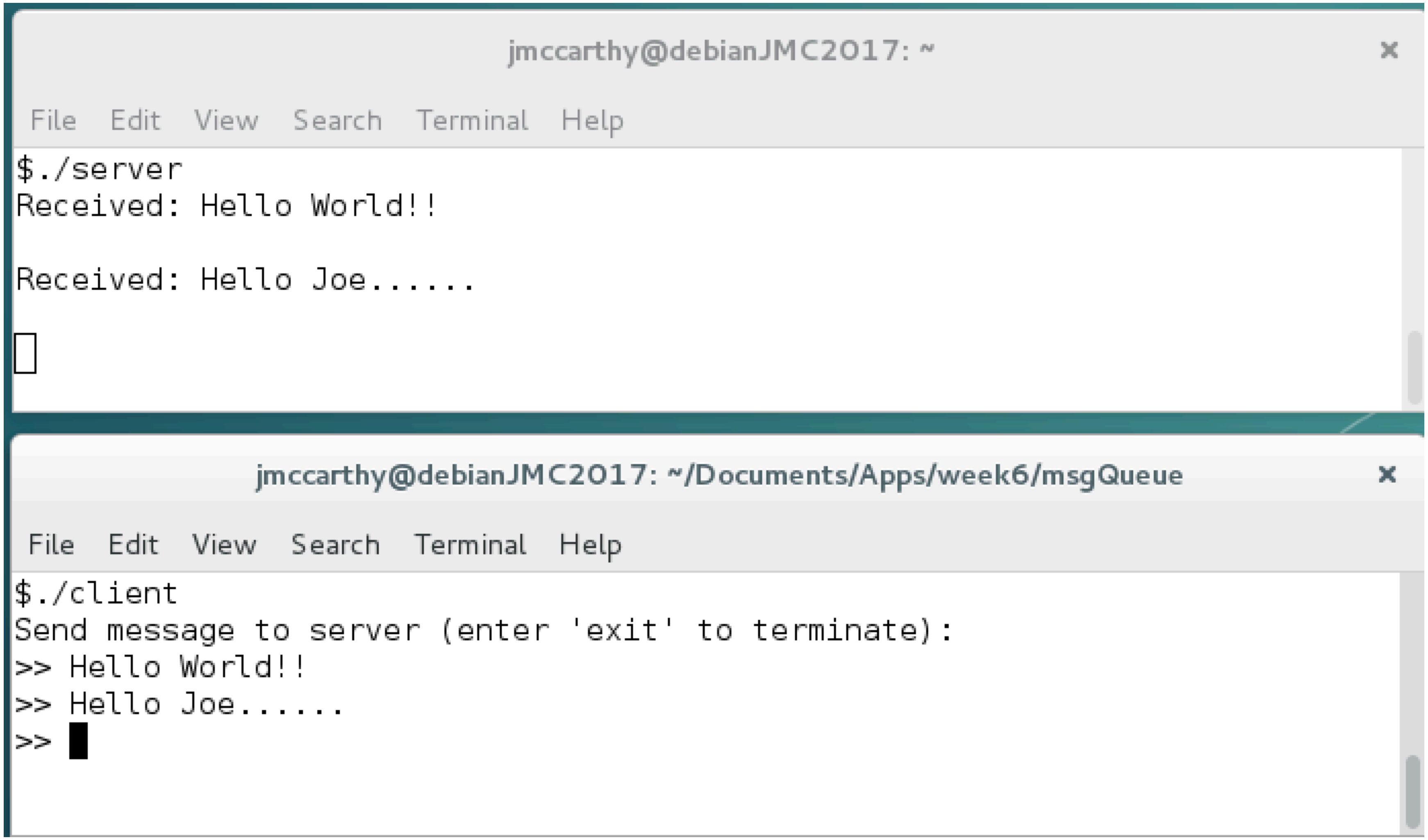

jmccarthy@debianJMC2017: ~
File Edit View Search Terminal Help

```
$./server
Received: Hello World!!
```

jmccarthy@debianJMC2017: ~/Documents/Apps/week6/msgQueue
File Edit View Search Terminal Help

```
$./client
Send message to server (enter 'exit' to terminate):
>> Hello World!!
>>
```

Run the Example

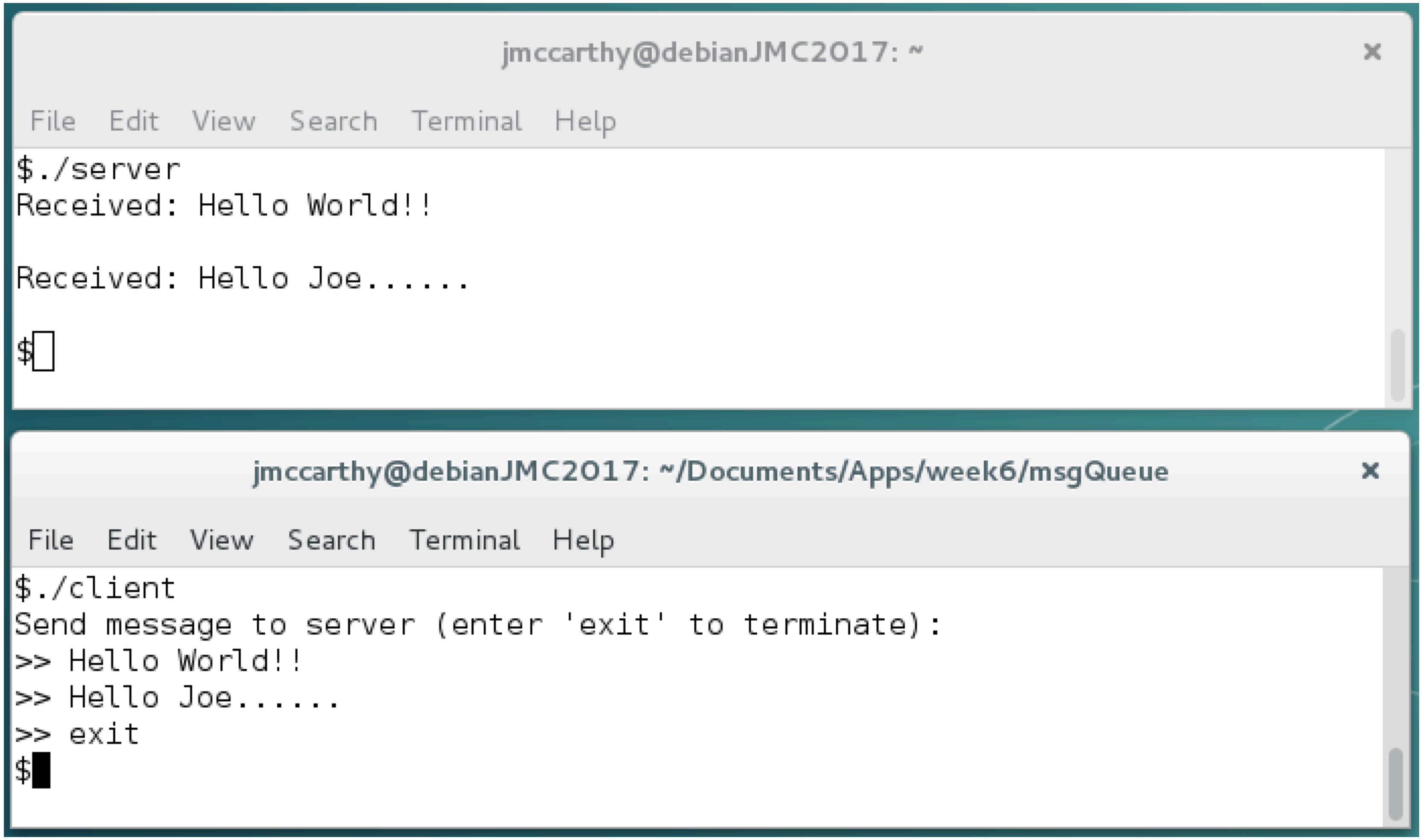

jmccarthy@debianJMC2017: ~
File Edit View Search Terminal Help

```
$./server
Received: Hello World!!

Received: Hello Joe......

```

jmccarthy@debianJMC2017: ~/Documents/Apps/week6/msgQueue
File Edit View Search Terminal Help

```
$./client
Send message to server (enter 'exit' to terminate):
>> Hello World!!
>> Hello Joe......
>>
```

Run the Example


jmccarthy@debianJMC2017: ~
File Edit View Search Terminal Help

```
$./server
Received: Hello World!!

Received: Hello Joe......

$
```

jmccarthy@debianJMC2017: ~/Documents/Apps/week6/msgQueue
File Edit View Search Terminal Help

```
$./client
Send message to server (enter 'exit' to terminate):
>> Hello World!!
>> Hello Joe......
>> exit
$
```

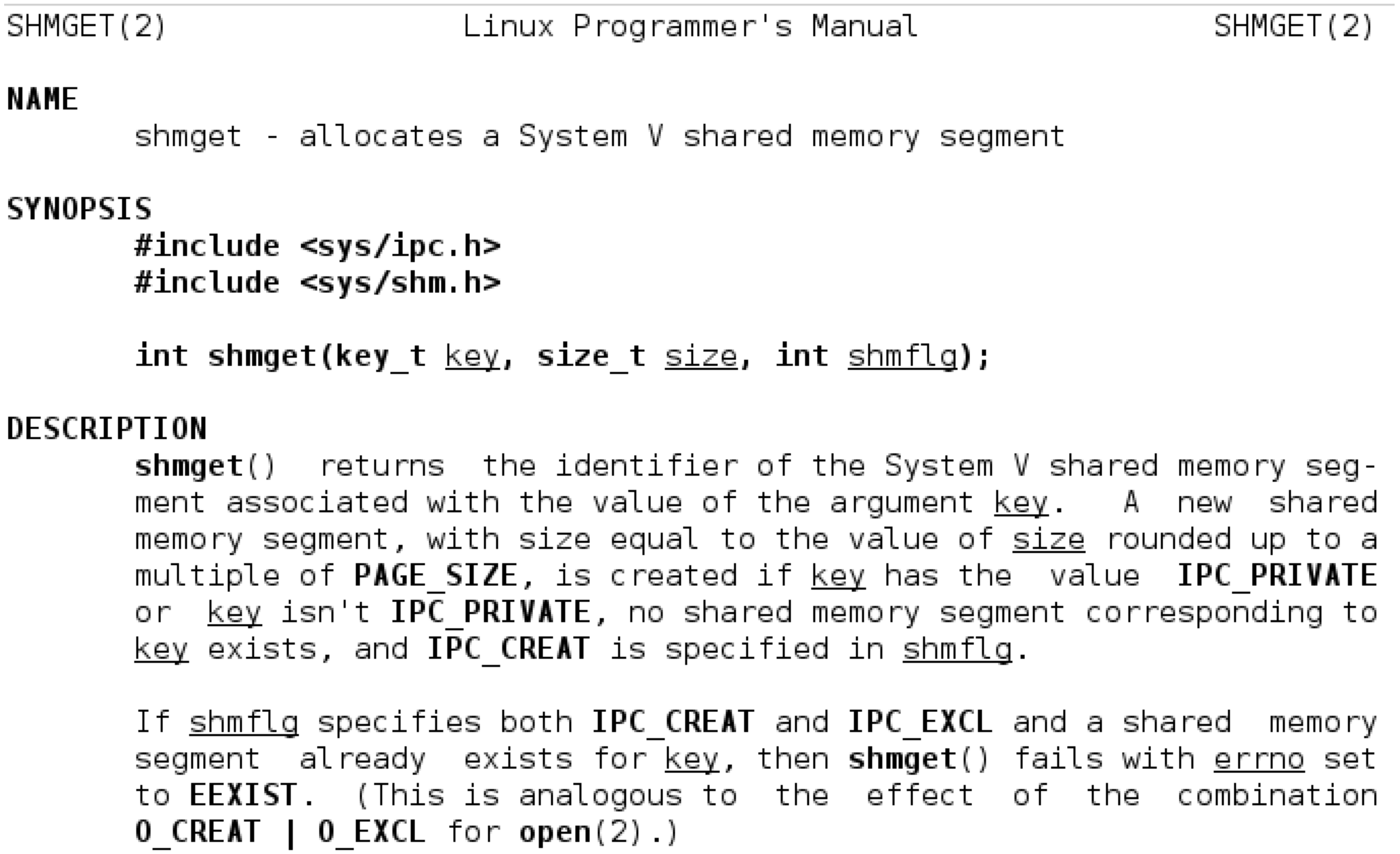
Shared Memory

- Shared memory can be used as a mechanism to pass data between different processes.
- One process will create the memory portion and other process can access the memory portion (if permitted).
- A process creates a shared memory segment using shmget()

Shared memory:: controlling access

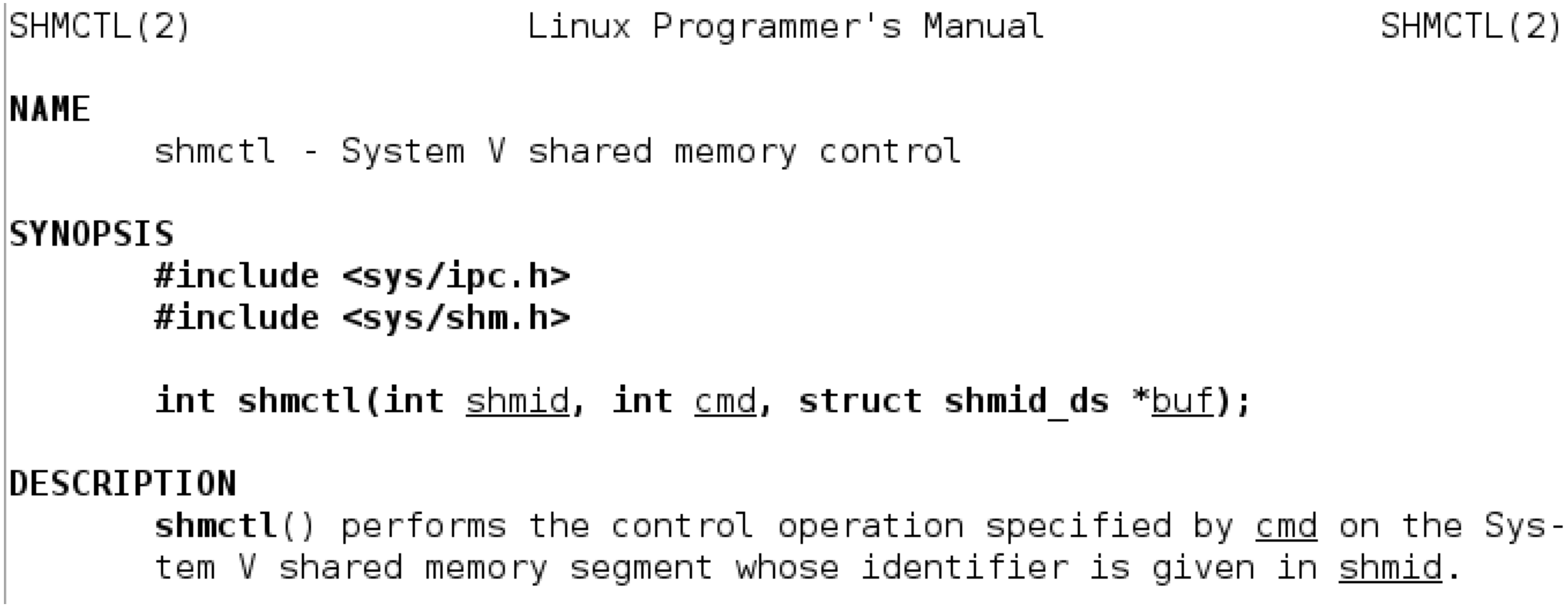
- The process that setups the shared memory segment can control what other processes can access it.
- Access can be granted using shmctl()
- Access can also be removed.

Shmget()



```
SHMGET(2)                  Linux Programmer's Manual                 SHMGET(2)

NAME
       shmget - allocates a System V shared memory segment

SYNOPSIS
       #include <sys/ipc.h>
       #include <sys/shm.h>

       int shmget(key_t key, size_t size, int shmflg);

DESCRIPTION
       shmget()  returns  the identifier of the System V shared memory seg-
       ment associated with the value of the argument key.   A  new  shared
       memory segment, with size equal to the value of size rounded up to a
       multiple of PAGE_SIZE, is created if key has the  value  IPC_PRIVATE
       or  key isn't IPC_PRIVATE, no shared memory segment corresponding to
       key exists, and IPC_CREAT is specified in shmflg.

       If shmflg specifies both IPC_CREAT and IPC_EXCL and a shared  memory
       segment  already  exists for key, then shmget() fails with errno set
       to EEXIST.  (This is analogous to  the  effect  of  the  combination
       O_CREAT | O_EXCL for open(2).)
```

Shmctl()



```
SHMCTL(2)                  Linux Programmer's Manual                 SHMCTL(2)

NAME
       shmctl - System V shared memory control

SYNOPSIS
       #include <sys/ipc.h>
       #include <sys/shm.h>

       int shmctl(int shmid, int cmd, struct shmid_ds *buf);

DESCRIPTION
       shmctl() performs the control operation specified by cmd on the Sys-
       tem V shared memory segment whose identifier is given in shmid.
```

Questions