BAHIRDAR INSTITUTE OF TECHNOLOGY (BIT)

DEPARTMENT OF SOFTWARE ENGINEERING

OPERATING SYSTEM AND SYSTEM PROGRAMMING

INDIVIDUAL ASSIGNMENT

The Message queue open System call(mq_open)

Name: Tsegaye Talegngeta

Full ID: BDU1307046

Submitted To: Mr. Wendimu B.(Msc)

Submission date: 17/11/2014 E.C

Introduction

In this document I have tried to describe about a message queue open system call, what a system call(specifically the mq_open) is, how the system call works, and also why do we need it, following that I have also tried to list and explain about its parameters and also the flags that it contains and finally there is the implementation of the system call, I hope this bring a knowledge about the open message queue system calls.

The Message Queue Open System Call

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

What is mq_open system call?

- The mq_open() system call is one of the message queue system calls that establishes the connection between a process and a message queue with a message queue descriptor(mqd_t). The *mq_open system* call is used to, as the name suggests, open a message queue.

NAME

mq_open - open a message queue,

SYNOPSIS

#include <mqueue.h>

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

Why do we use the mq_open?

mq_open creates an open message queue description that refers to the message queue, and a message queue descriptor that refers to that open message queue description. The message queue descriptor(mqd_t) is used by other functions to refer to that message queue. it creates a new POSIX message queue or opens an existing queue. The queue is identified by *name*.

How do we use the mq_open?

We can use the mq_open system call by first including the header file **mqueue.h** and then by filling the place of the parameters and also the place of the flags after than we can use it to open the message queue easily.

Parameters:

name

- ✓ The name of the message queue that you want to open.

oflag

- ✓ This argument specifies flags that control the operation of the call.(Definitions of the flags values can be obtained by including <fcntl.h>).

mode_t mode

- ✓ The file permissions for the new queue. For more information. If you set any bits other than file permission bits, they're ignored. Read and write permissions are analogous to receive and send permissions; execute permissions are ignored.

struct mq_attr *mq_attr

- ✓ NULL, or a pointer to an mq_attr structure that contains the attributes that you want to use for the new queue.

Flags:

Exactly one of the following must be specified in *oflag*:

O_RDONLY
Open the message queue for receiving messages. The process can use the returned message queue descriptor with mq_receive(), but not mq_send(). A message queue may be open multiple times in the same or different processes for receiving messages.

O_WRONLY
Open the queue for sending messages. The process can use the returned message queue descriptor with mq_send() but not mq_receive(). A message queue may be open multiple times in the same or different processes for sending messages.

O_RDWR
Open the queue for both receiving and sending messages. The process can use any of the functions allowed for O_RDONLY and O_WRONLY. A message queue may be open multiple times in the same or different processes for sending messages.

O_CREAT Create a message queue. If the pathname name has already been used to create a message queue that still exists, then this flag has no effect, except as noted under O_EXCL. Otherwise, a message queue will be created without any messages in it. The user ID of the message queue will be set to the effective user ID of the process, and the group ID of the message queue will be set to the effective group ID of the process. The permission bits of the message queue will be set to the value of the mode argument, except those set in the file mode creation mask of the process. When bits in mode other than the file permission bits are specified, the effect is unspecified. If attr is NULL, the message queue is created with implementation defined default message queue attributes. If attr is non-NULL and the calling process has the appropriate privilege on name, the message queue mq_maxmsg and mq_msgsize attributes will be set to the values of the corresponding members in the mq_attr structure referred to by attr. If attr is non-NULL, but the calling process does not have the appropriate privilege on name, the mq_open() function will fail and return an error without creating the message queue.

O_EXCL If O_EXCL and O_CREAT are set, mq_open() will fail if the message queue name exists.

O_NONBLOCK Open the queue in nonblocking mode. In circumstances where mq_recive(3) and mq_send(3) would normally block, these functions instead fail with the error **EAGAIN**.

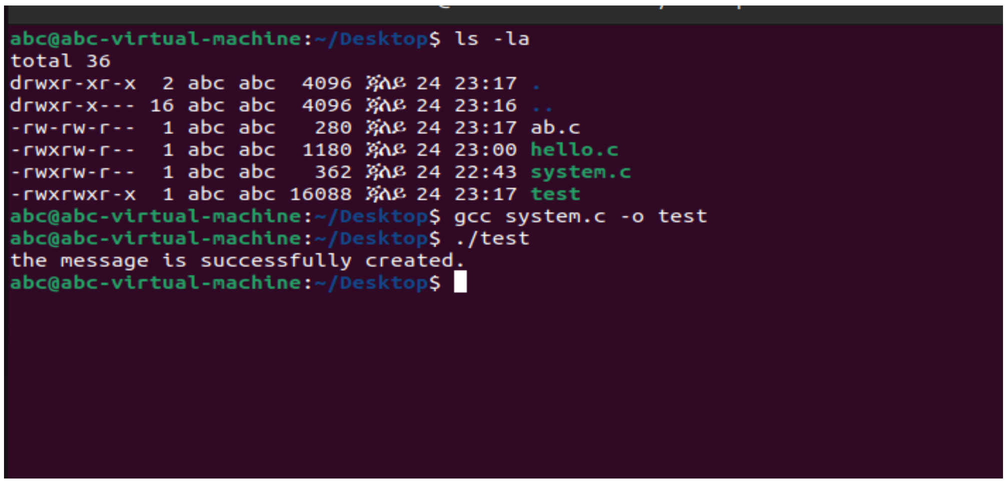
To summarize :

In *mq_open* we pass the parameters — message queue name *(/mymq,* which needs to begin with a *"/")*; the flags O_CREAT (which creates the message queue if it doesn't already exist) and O_RDWR to read and write from and to the message queue; the access permission and, finally, the queue attributes. If the *mq_open* call returns successfully with a valid queue descriptor value, we print the message: "Message Q is successfully created ….."; otherwise, we print "Message Q is not created…." to indicate that the message queue creation has failed. The oflag argument requests the desired receive and/or send access to the message queue. The requested access permission to receive messages or send messages would be granted if the calling process would be granted read or write access, respectively, to an equivalently protected file. The value of oflag is the bitwise-inclusive OR of values from the following list. The mq_open() system call does not add or remove messages from the queue.

The Code Implementation Of The O_RDWR Flag

```
#include<stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <mqueue.h>
#include <fcntl.h>
int main (){
    int md;
/* Create message queue */
    md = mq_open ("my_queue", O_CREAT|O_RDWR, NON_BLOCK, 100|5);
   if(md){
   printf("the message is successfully created");
   }
   else {
   printf("the message is not created");
   }
return 0;
}
```

OUTPUT

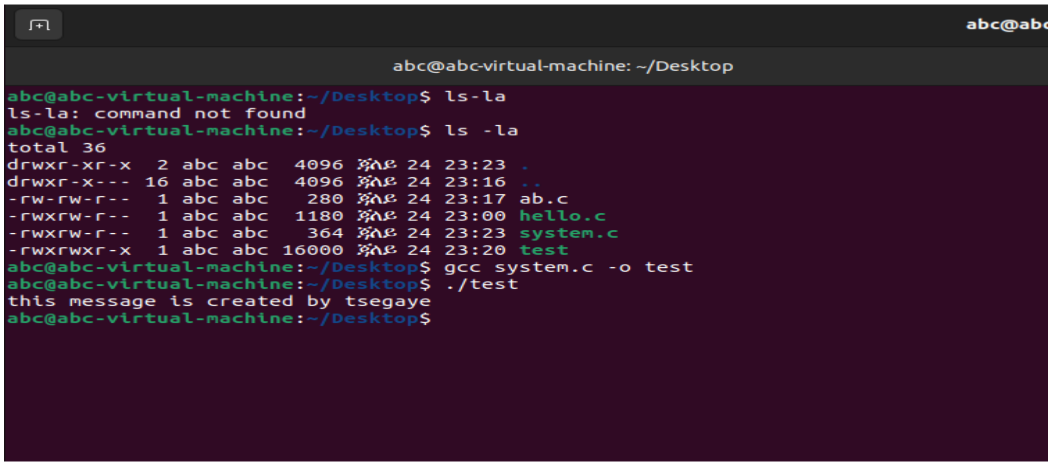


```
abc@abc-virtual-machine:~/Desktop$ ls -la
total 36
drwxr-xr-x  2 abc abc  4096 ጁላይ 24 23:17 .
drwxr-x--- 16 abc abc  4096 ጁላይ 24 23:16 ..
-rw-rw-r--  1 abc abc   280 ጁላይ 24 23:17 ab.c
-rwxrw-r--  1 abc abc  1180 ጁላይ 24 23:00 hello.c
-rwxrw-r--  1 abc abc   362 ጁላይ 24 22:43 system.c
-rwxrwxr-x  1 abc abc 16088 ጁላይ 24 23:17 test
abc@abc-virtual-machine:~/Desktop$ gcc system.c -o test
abc@abc-virtual-machine:~/Desktop$ ./test
the message is successfully created.
abc@abc-virtual-machine:~/Desktop$ 
```

The Code Implementation Of The O_WRONLY Flag

```
#include<stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <mqueue.h>

#include <fcntl.h>

int main (){

    int md;

/* Create message queue */

    md = mq_open ("my_queue", O_CREAT|O_WRONLY ,NON_BLOCK, 100|5);

   if(md){

   printf("this message is created by tsegaye");

   }

   else {

   printf("the message is not created");

   }

return 0;

}
```

OUTPUT

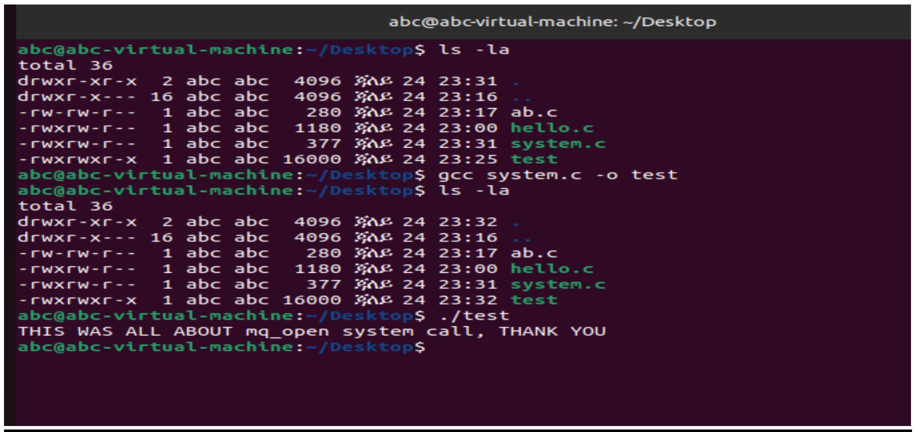


```
abc@abc-virtual-machine:~/Desktop$ ls-la
ls-la: command not found
abc@abc-virtual-machine:~/Desktop$ ls -la
total 36
drwxr-xr-x  2 abc abc  4096 ጁላይ 24 23:23 .
drwxr-x--- 16 abc abc  4096 ጁላይ 24 23:16 ..
-rw-rw-r--  1 abc abc   280 ጁላይ 24 23:17 ab.c
-rwxrw-r--  1 abc abc  1180 ጁላይ 24 23:00 hello.c
-rwxrw-r--  1 abc abc   364 ጁላይ 24 23:23 system.c
-rwxrwxr-x  1 abc abc 16000 ጁላይ 24 23:20 test
abc@abc-virtual-machine:~/Desktop$ gcc system.c -o test
abc@abc-virtual-machine:~/Desktop$ ./test
this message is created by tsegaye
abc@abc-virtual-machine:~/Desktop$
```

The Code Implementation Of The O_RDONLY Flag

```
#include<stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <mqueue.h>
#include <fcntl.h>
int main (){
    int md;
/* Create message queue */
    md = mq_open ("my_queue", O_CREAT|O_RDONLY ,NON_BLOCK, 100|5);
   if(md){
   printf("THIS WAS ALL ABOUT mq_open system call, THANK YOU");
   }
   else {
   printf("the message is not created");
   }
return 0;
}
```

OUTPUT



```
abc@abc-virtual-machine: ~/Desktop
abc@abc-virtual-machine:~/Desktop$ ls -la
total 36
drwxr-xr-x  2 abc abc  4096 ጃንዩ 24 23:31 .
drwxr-x--- 16 abc abc  4096 ጃንዩ 24 23:16 ..
-rw-rw-r--  1 abc abc   280 ጃንዩ 24 23:17 ab.c
-rwxrw-r--  1 abc abc  1180 ጃንዩ 24 23:00 hello.c
-rwxrw-r--  1 abc abc   377 ጃንዩ 24 23:31 system.c
-rwxrwxr-x  1 abc abc 16000 ጃንዩ 24 23:25 test
abc@abc-virtual-machine:~/Desktop$ gcc system.c -o test
abc@abc-virtual-machine:~/Desktop$ ls -la
total 36
drwxr-xr-x  2 abc abc  4096 ጃንዩ 24 23:32 .
drwxr-x--- 16 abc abc  4096 ጃንዩ 24 23:16 ..
-rw-rw-r--  1 abc abc   280 ጃንዩ 24 23:17 ab.c
-rwxrw-r--  1 abc abc  1180 ጃንዩ 24 23:00 hello.c
-rwxrw-r--  1 abc abc   377 ጃንዩ 24 23:31 system.c
-rwxrwxr-x  1 abc abc 16000 ጃንዩ 24 23:32 test
abc@abc-virtual-machine:~/Desktop$ ./test
THIS WAS ALL ABOUT mq_open system call, THANK YOU
abc@abc-virtual-machine:~/Desktop$
```

References

- https://w3.cs.jmu.edu/kirkpams/OpenCSF/Books/csf/html/MQueues.html
- https://www.math-linux.com/man/man3/mq_open.3.html
- https://www.daemon-systems.org/man/mq_open.3.html
- Linux manual page