BAHIR DAR UNIVERSITY INSTITUTE
OF TECHNOLOGY
FACULUTY OF COMPUTING
DEPARTMENT OF SOFTWARE ENGINEERING
OPERATING SYSTEM AND
SYSTEM PROHRAMMING
INDIVIDUAL ASSIGNMENT

Prepared by : Ezira Tigab ID : 1308251

...
...

Submitted to : instructor Wondimu Bayu.
Submission date: 17/11/2014 E.C

1. What ,how and is msgctl system call?

- ✓ The msgctl function is used to perform one of several control operations on an UNIX System Services message queue.
- ✓ msgctl() one of system call that performs the control operation specified by cmd on the System message queue with identifier msqid.

Return Value

0	**If msgctl() is** successful it return 0.
-1	**If not it retuns -1** . The *errno* variable is set to indicate the error.

Error Conditions

If **msgctl()** is not successful, *errno* usually indicates one of the following errors. Under some conditions, *errno* could indicate an error other than those listed here.

[EACCES]

Permission denied.

- ✓ An attempt was made to access an object in a way forbidden by its object access permissions.
- ✓ The thread does not have access to the specified file, directory, component, or path.
- ✓ The *cmd* parameter is IPC_STAT and the calling thread does not have read permission.

[EDAMAGE]

- ✓ If damaged object is encountered a referenced object is damaged.
- ✓ The object cannot be used.
- ✓ The message queue has been damaged by a previous message queue operation.

[EFAULT]

The error occured by the following :

- ✓ If the address used for an argument is not correct.
- ✓ In attempting to use an argument in a call, the system detected an address that is not valid.
- ✓ While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]

- ✓ If the value specified for the argument is not correct.
- ✓ A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.
- ✓ If argument value is not valid, out of range, or NULL.

One of the following has occurred:

- The *msqid* parameter is not a valid message queue identifier.
- The *cmd* parameter is not a valid command.
- The *cmd* parameter is IPC_SET and the value of msg_qbytes exceeds the system limit.

[EPERM}

✓ If an Operation is not permitted.

We must have appropriate privileges or be the owner of the object or other resource to do the requested operation.

The *cmd* parameter is equal to IPC_RMID or IPC_SET and both of the following are true:

- the caller does not have the appropriate privileges.
- the effective user ID of the caller is not equal to the user ID of the owner or the user ID of the creator of the message queue.

The *cmd* parameter is IPC_SET and an attempt is being made to increase the maximum number of bytes for the message queue, but the the caller does not have appropriate privileges.

[EUNKNOWN]

✓ It occured by unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

2. whta are it's parameter?

Parameters

This system call have three system call .

msqid

(Input) Message queue identifier, a positive integer. It is returned by the msgget() function and used to identify the message queue on which to perform the control operation.

cmd

(Input) Command, the control operation to perform on the message queue. Valid values are listed above.

buf

(I/O) Pointer to the message queue data structure to be used to get or set message queue information.

The members of the msqid_ds structure in buf parameter are as follows:

tag	Description
msgnum_t msg_qnum	The number of messages currently on the message queue.
msglen_t msg_qbytes	The maximum number of bytes allowed on the message queue
pid_t msg_lspid	The process ID of the last job to send a message using msgsnd
pid_t msg_lrpid	The process ID of the last job to receive a message using msgr

time_t msg_stime	The time the last job sent a message to the message queue usi
time_t msg_rtime	The time the last job received a message from the message qu
time_t msg_ctime	The time the last job changed the message queue using msgct
struct ipc_perm msg_perm	

The members of the ipc_perm structure are as follows:

tag	Description
uid_t uid	The user ID of the owner of the message queue.
gid_t gid	The group ID of the owner of the message queue.
uid_t cuid	The user ID of the creator of the message queue.
gid_t cgid	The group ID of the creator of the message queue.
mode_t mode	The permissions for the message queue.

Flags of cmd parameter

IPC_RMID (0x00000000)

- ✓ To remove the message queue identifier msqid from the system and destroy any messages on the message queue.
- ✓ Any threads that are waiting in msgsnd() or msgrcv() are woken up and msgsnd() or msgrcv() returns with a return value of -1 and errno set to EIDRM.
- ✓ The IPC_RMID command can be run only by a thread with appropriate privileges or one that has an effective user ID equal to the user ID of the owner or the user ID of the creator of the message queue.
- ✓ The structure pointed to by *buf is ignored and can be NULL.

IPC_SET (0x00000001)

This value used To Set

- ✓ the user ID of the owner,
- ✓ the group ID of the owner,
- ✓ the permissions,
- ✓ and the maximum number of bytes for the message queue to the values in the msg_perm.uid, msg_perm.gid, msg_perm.mode, and msg_qbytes members of the msqid_ds data structure pointed to by *buf.

The IPC_SET command can be run only by a thread with appropriate privileges or one that has an effective user ID equal to the user ID of the owner or the user ID of the creator of the message queue.
In addition, only a thread with appropriate privileges can increase the maximum number of bytes for the message queue.

IPC_STAT (0x00000002)

Store the current value of each member of the msqid_ds data structure into the structure pointed to by *buf. The IPC_STAT command requires read permission to the message queue.

IPC_INFO (Linux-specific)

- ✓ Return information about system-wide message queue limits and parameters in the structure pointed to by buf.
- ✓ This structure is of type msginfo (thus, a cast is required), defined in <sys/msg.h> if the _GNU_SOURCE feature test macro is defined:

```
struct msginfo {
  int msgpool; /* Size in kibibytes of buffer pool
                  used to hold message data;
                  unused within kernel */
  int msgmap;  /* Maximum number of entries in message
                  map; unused within kernel */
  int msgmax;  /* Maximum number of bytes that can be
                  written in a single message */
  int msgmnb;  /* Maximum number of bytes that can be
                  written to queue; used to initialize
                  msg_qbytes during queue creation
                  (msgget(2)) */
  int msgmni;  /* Maximum number of message queues */
  int msgssz;  /* Message segment size;
                  unused within kernel */
  int msgtql;/* Maximum number of messages on all queues
                in system; unused within kernel */
  unsigned short msgseg;
                /* Maximum number of segments;
                  unused within kernel */
};
```

MSG_INFO (Linux-specific)

Return a msginfo structure containing the same information as for IPC_INFO, except that the following fields are returned with information about system resources consumed by message queues: the msgpool field returns the number of message queues that currently exist on the system; the msgmap field returns the total number of messages in all queues on the system; and the msgtql field returns the total number of bytes in all messages in all queues on the system.

MSG_STAT (Linux-specific)

Return a msqid_ds structure as for IPC_STAT. However, the msqid argument is not a queue identifier, but instead an index into the kernel's internal array that maintains information about all message queues on the system.

MSG_STAT_ANY (Linux-specific, since Linux 4.17)

Return a msqid_ds structure as for MSG_STAT. However,

msg_perm.mode is not checked for read access for msqid meaning that any user can employ this operation (just as any user may read /proc/sysvipc/msg to obtain the same information).

Code implementation of some flags

Before we see code implementation of thus flags first we see the return value of those flags

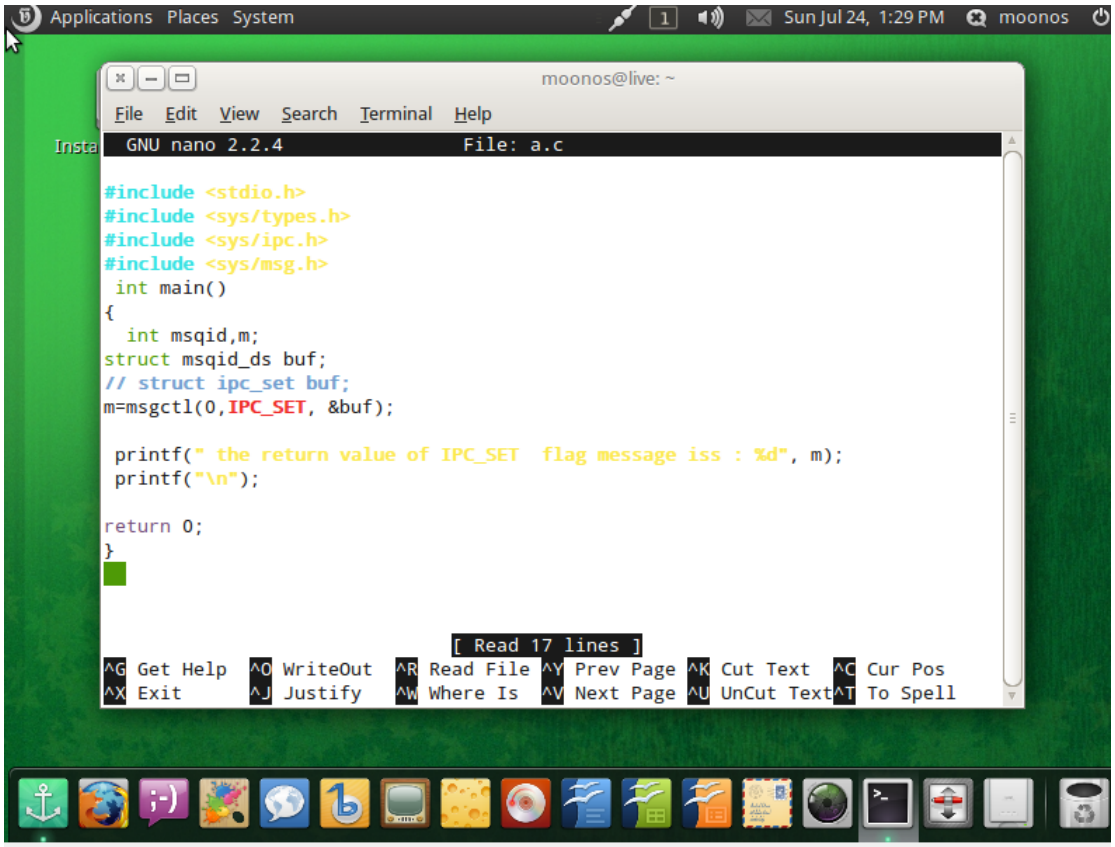
Flags	Return on succes	Return if fail
IPC_STAT,	0	-1
IPC_SET	0	-1
IPC_RMID	0	-1
IPC_INFO	Index of the highest index entry	-1
MSG_INFO	>>	-1
MSG_STAT	Identifier of index msqid queue	-1

Code implementation

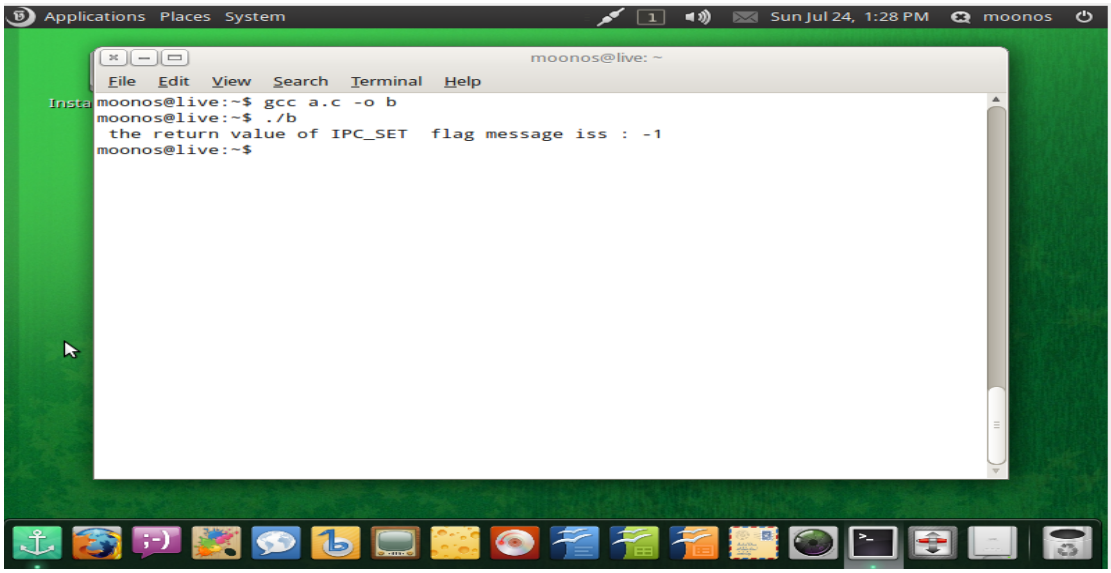
IPC_SET

```
 #include <stdio.h>
#include <sys/types.h>
#include <sys/msg.h>
#include <sys/ipc.h>
 int main ()
   {
     int msqid ,m;
     struct msqid buf;
     m= msgctl(0,IPC_SET, &buf);
     printf(" the return value of  IPC_SET  is : %d" , m);
return 0;
}
```

No


Applications Places System
Sun Jul 24, 1:29 PM
moonos
moonos@live: ~
File Edit View Search Terminal Help
GNU nano 2.2.4 File: a.c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int main()
{
int msqid,m;
struct msqid_ds buf;
// struct ipc_set buf;
m=msgctl(0,IPC_SET, &buf);
printf(" the return value of IPC_SET flag message iss : %d", m);
printf("\n");
return 0;
}
[Read 17 lines]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

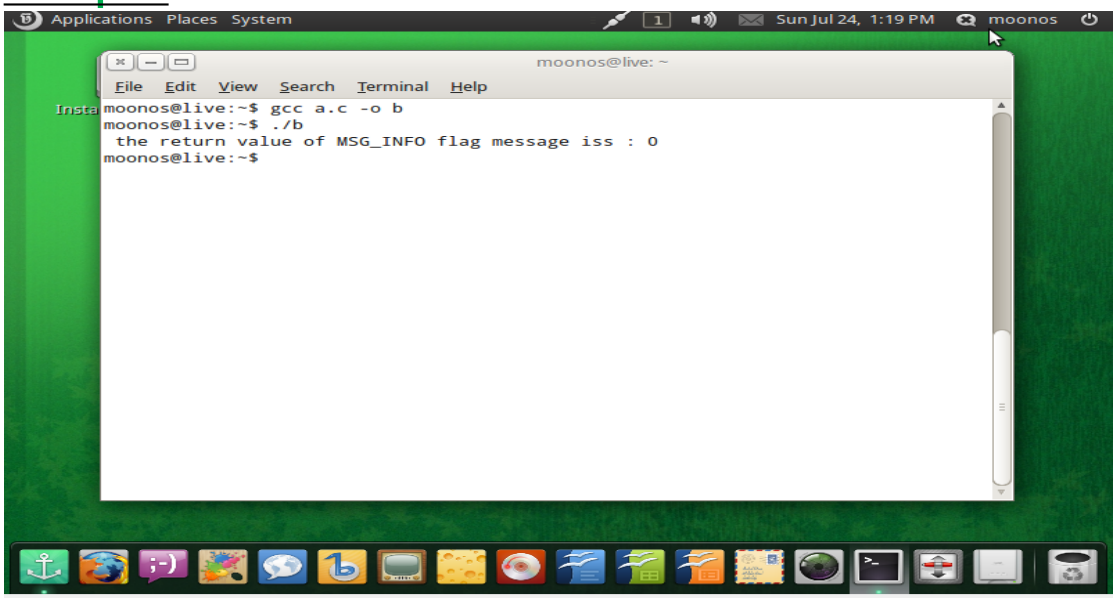
Out put


Applications Places System
Sun Jul 24, 1:28 PM
moonos
moonos@live: ~
File Edit View Search Terminal Help
moonos@live:~$ gcc a.c -o b
moonos@live:~$ ./b
the return value of IPC_SET flag message iss : -1
moonos@live:~$

MSG_INFO code implementation

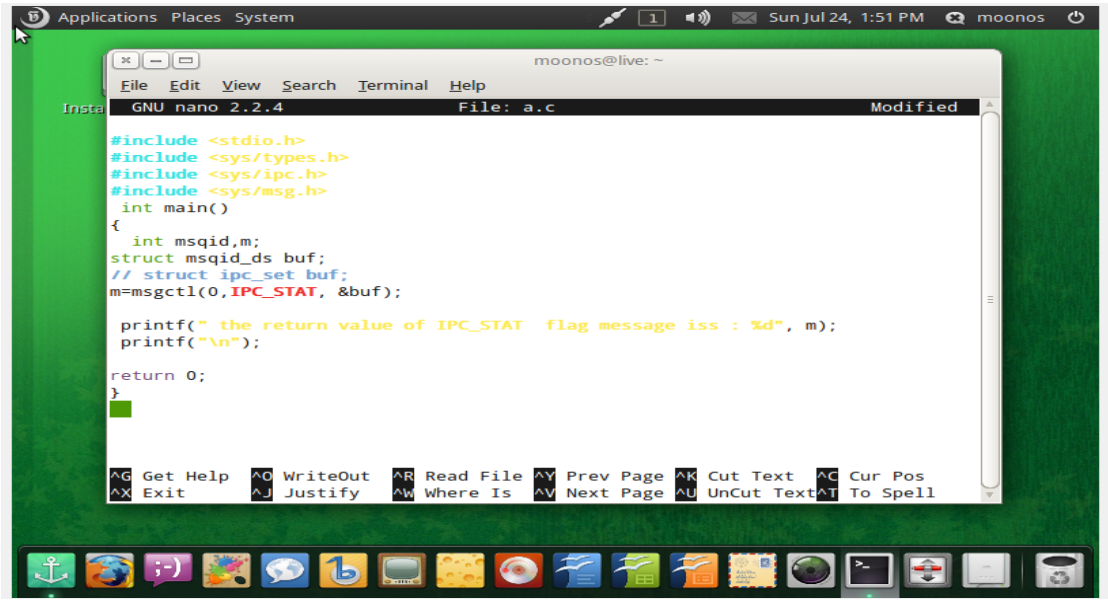
```
 #include <stdio.h>
#include <sys/types.h>
#include <sys/msg.h>
#include <sys/ipc.h>
 int main ()
   {
     int msqid ,m;
     struct msqid ;
     Struct msgid-ds;
   struct msginfo msginfo;
     m= msgctl(0,MSG_INFO, (struct msqid_ds *)&msginfo);
     printf(" the return value of  MSG_INFO  is : %d" , m);
return 0;
}
```

Out put



IPC_STAT code implementationn

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/msg.h>
#include <sys/ipc.h>
 int main ()
   {
     int msqid ,m;
     struct msqid buf;
     m= msgctl(0,IPC_STAT, &buf);
     printf(" the return value of IPC_STAT  is : %d" , m);
return 0;
}
```



Out put