Group 11 - Report 3

Introduction to Operating System (IOS)

01FB15ECS213 Praveen C Naik

01FB15ECS202 Parashara Ramesh

01FB15ECS209 Prajwal B

01FB15ECS224 Rahul Pillai

Minix Version : 3.2.1

http://www.minix3.org

Mailbox Implementation

What is a mailbox in operating systems:

- Mailboxes are usually used in operating systems as a form of Inter Process Communication.
- It is an asynchronous way of doing IPC.
- This is similar to “message passing” concept in IPC.

Why mailbox in minix?:

- When we want to implement new system calls in minix3, user processes are not allowed to send messages to each other so this implementation of mailbox allows it.

Design considerations:

- In our implementation we have considered that each process has its own mailbox and each mailbox can have multiple publishers and can store a list of messages(implemented as a linked list) which comprises of messages from each publisher.
- Messages are from any publisher are always appended to the end of this “message linked list”.Therefore we can say that this message list is actually a queue which the owner can read on a first come first served basis.
- Each mailbox can have a maximum of 256 publishers.
- What we did not do was give priority to each message so that messages of higher priority are stored at the beginning of this message list.Also we were thinking of making the owner read messages in a “round robin” way where each publisher’s message gets read in a fair manner but we did not get the time to test this out.

Different functions implemented in our mailbox:

We mainly have 3 files for our implementation.

A. mailboxtest.c (this is the test program present in the user level ie root directory)
B. mailbox.h (this file contains functions which are wrappers to our actual system calls this is present in the /usr/include directory)
C. mailbox.c (this file contains all the implementations of the various system calls we are making for this implementation present in the /usr/src/servers/pm directory)

1.CREATING A MAILBOX:

```
struct mailbox{
  int nbOfMes;
  int owner;
  struct publisher * publishers;
  struct messageList * mList;
  struct mailbox * next;
};

struct mailbox * defaultBox=NULL;

//the structure of our mailbox
```

mailboxtest.c file is the driver function for the implementation of the mailbox. This driver function provides 5 test functions among which this is one of the functions.

```
void func_createMailbox(int publisher[256]);
```

The test function calls the wrapper function createMailbox that calls the actual system call for creating a mailbox. For creating the mailbox, following _syscall of **do_createmailbox** made:

```
int x=_syscall(PM_PROC_NR,CREATEMAILBOX,&m);
```

This maps to “do_createMailbox” present in the mailbox.c file present in the /servers/pm directory.

The do_createMailbox system call does the following actions:

- Makes sure that each process can have only one mailbox else returns an error.
- Initialises each member of the mailbox structure and sets the list of publishers given from the user
- We attach this “new” mailbox node as the head of the mailbox linked list “defaultBox”.
- returns the mailbox id i.e. “who_e” if successful else returns -1.

2.REMOVING MAILBOX:

In mailboxtest.c when we try to remove a mailbox for the current process we call the removeMailbox() function present in the mailbox.h file.

in removeMailbox( ) we have a message ‘m’ initialized in it and passed to do_removeMailbox syscall (in mailbox.c). The initialized message ‘m’ is just a dummy passed to call the system call not actually containing anything.

In the do_removeMailbox() system call we do the following actions:

- If the process calling this function is the current owner i.e. who_e(which also happens to be the mailbox id) it removes that mailbox from the head of the mailbox linked list i.e., “defaultBox”.
- Else, if the head of this mailbox linked list “defaultBox” has another owner id which is not that of the process currently calling it then traverse through this mailbox linked list until you find the current owner and delete that owner’s mailbox.
- Returns 1 if successful else returns -1.

3.DEPOSITING A MESSAGE INTO THE MAILBOX:

In mailboxtest.c we have a function **void func_deposit(int receiver[256], char * message);**

- Which stores the message we want to deposit in the m1_p1 member of the message structure.
- We can also tell specify all the mailbox ids of the receivers of this message in the receivers array(max size 256).

This test function then calls the wrapper function “int deposit(char *mes,int *receivers,int n,int l)” present in the mailbox.h file.

Inside this function present in the mailbox.h file we do the following actions:

- We have a local variable of message type and we initialize various members of this message and pass this message as a pointer to the actual system call.
- This message has the following members.

```
m.m1_i1=l;          //Message length
m.m1_i2=n;          //Number of receivers
m.m1_p1=mes;        //the actual message which is to be put into
the mailbox
m.m1_p2=(char*)receivers;
//an array containing the list of all the receiver mailbox ids.
```

Inside the do_deposit() system call present in mailbox.c we do the following actions:

- It is deposited to all the receiver mailbox ids. Each receiver could either be a blocked receiver and non-blocked receiver.(the concept of blocked comes when we try to retrieve a

message from an empty mailbox ,at that time that process trying to retrieve ie the owner of the mailbox is blocked).

- For the list of non-blocked receivers the mailbox linked list is traversed. If any of the mailbox already has its capacity full ie 10 messages it will return "**mailbox full"**. If the current owner is one of the receivers and we can publish to that person with his no of messages < 10 then we traverse and appends it to the linked list and increment message count.
- For blocked receivers, traverse the blocked processes in linked list (even if it happens to be the receiver) sys_resume(bp->blocked) is made.
- If return state is -1 if it is unsuccessful and returns 1 then is successful.

4.RETRIEVING A MESSAGE FROM THE MAILBOX:

In mailboxtest.c we have a function "void func_receive(char * response);" where we do the following:

- We can either specify the mailbox id from which we want to retrieve the message or retrieve the topmost message in the messagelist.
- After retrieving from a mesage from the messagelist it gets deleted from the "messagelist" linked list.
- The message retrieved is stored in the variable "response" which is passed as a parameter to this function

```
void func_receive(char * response){
    int receiveFrom=0;
    printf("Receive from which process (0 mean most ancient
message):");
    scanf("%d",&receiveFrom);
    retrieve (response,receiveFrom);
    printf("Message: %s\n",response);
    printf("\nType enter to continue");
    getchar();
    getchar();
}
```

This calls the wrapper function "int retrieve (char * r, int source)" which is in mailbox.h which does the following:

- We pass the parameters of this function into a message structure.

- Now we try calling the system call until the time it is successful as there is a possibility that we could be retrieving from an empty mailbox.

This system call do_retrieve() is defined in mailbox.c which does the following:

- We have a variable called returnAddress which holds the response i.e. message.
- We start traversing the linked list of mailboxes “defaultBox” until we find the process’s mailbox which is done by checking if the owner is equal to “who_e”.
- Then in the messagelist we traverse this linked list and find the sender whose message we want.
- If the mailbox is empty then add this to the tail of the “blockedproc” list.
- Then we stop the present process’s mailbox by using the kernelcall “sys_stop(who_e);”
- We then copy the message from the sourceaddress to returnaddress i.e. into the response variable before mentioned.
- When retrieving the message from the messagelist we call the function deleteMessage() which deletes it from this linked list.
- If this operation fails we return -1.

5. ADDING A PUBLISHER TO THE MAILBOX:

In mailboxtest.c we have a function void func_addPublisher() that calls the function addPublisher() present in mailbox.h.

This function adds a publisher to the mailbox.

Each publisher has an id “m1_i1”. The function addPublisher() calls the system call do_addPublisher() present in mailbox.c. The already existing mailbox ids (of other processes) are assigned as publisher id for the current mailbox.

For the current mailbox id “who_e” the publisher is added to the publisher linked list.

- A new publisher is added to the end of the publisher linked list if publishers already exist.

```
p->next=malloc(sizeof(struct publisher));
p->next->id=m_in.m1_i1;
p->next->next=NULL;
```

- If no publishers are assigned to a mailbox. A new publisher is added as the first element (head) of the linked list.

```
p=malloc(sizeof(struct publisher));
p->id=m_in.m1_i1;
p->next=NULL;
```

6. REMOVING A PUBLISHER FROM THE MAILBOX:

In mailboxtest.c we have a function void func_removePublisher() that calls the function removePublisher() present in mailbox.h.

This function removes a publisher from the mailbox.

Each publisher has an id “m1_i1”. The removePublisher function calls the system call do_removePublisher() present in mailbox.c.

For the current mailbox id “who_e” the publisher is removed from the publisher linked list.

- Remove the head of the publisher linked list if the required publisher id is in the head.

```
if (p->id == m_in.m1_i1 ) {
       p= p->next;
       free(p);
}
```

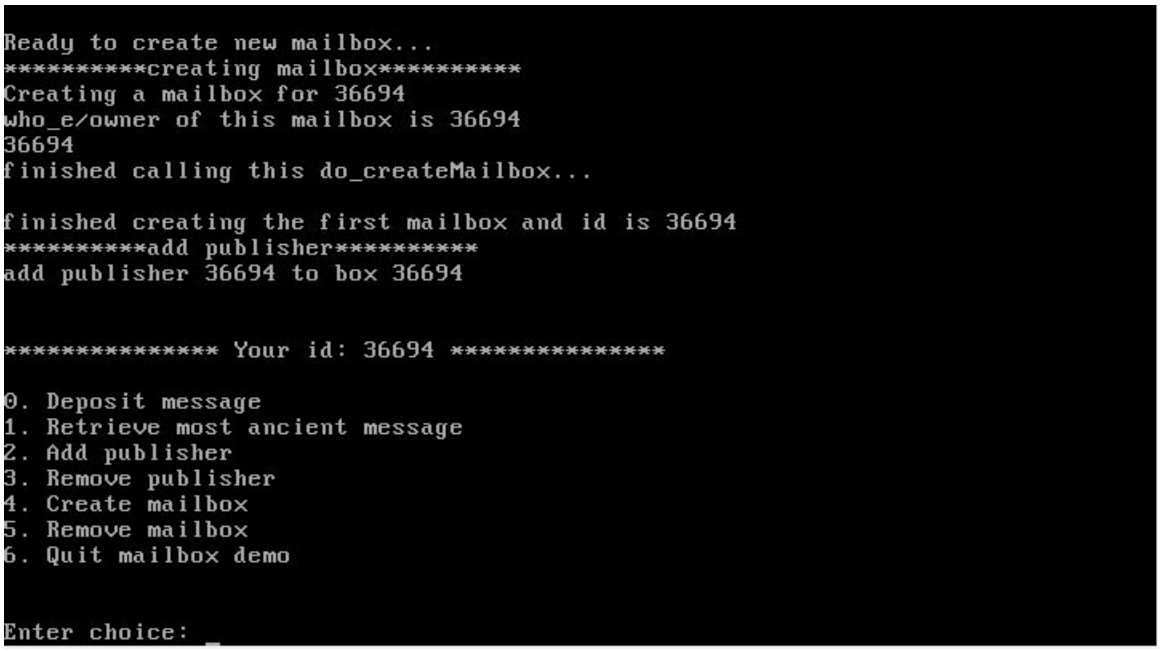
- Remove the node of the publisher linked list if the required publisher id is somewhere in the linked list other than the head.

```
if (p->next->id ==m_in.m1_i1 ) {
     p->next = p->next->next;
     free(p->next);
}
```

Trying to remove a publisher id that is not assigned to the mailbox results in popping up of “Not found” message.

Working demo of our mailbox simulation:

Creating mailbox for 1st process i.e. terminal 1(MAILBOX ID=36694)



Calling the test code in terminal 2 (MAILBOX ID=36699)to get the id of this process similarly in terminal 3(MAILBOX ID=36704)

```
# cc mailboxtest.c
# ./a.out
going to create  first mailbox 161
publishers is 161

Ready to create new mailbox...
36699
finished calling this do_createMailbox...

finished creating the first mailbox and id is 36699


**************** Your id: 36699 ****************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: _
```

In terminal 1 after initializing both mailboxes for terminal 2 and terminal 3

```
add publisher 36694 to box 36694


**************** Your id: 36694 ****************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: **********creating mailbox**********
Creating a mailbox for 36699
who_e/owner of this mailbox is 36699
**********add publisher**********
add publisher 36699 to box 36699
**********creating mailbox**********
Creating a mailbox for 36704
who_e/owner of this mailbox is 36704
**********add publisher**********
add publisher 36704 to box 36704
```

In terminal 1 after adding 1st publisher i.e. terminal 2's MAILBOX ID

```
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter publisher id:36699
**********add publisher**********
add publisher 36699 to box 36694


*************** Your id: 36694 ***************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: _
```

In terminal 1 after adding 2nd publisher i.e. terminal 3's MAILBOX ID

```
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter publisher id:36704
**********add publisher**********
add publisher 36704 to box 36694


*************** Your id: 36694 ***************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: _
```

In terminal 2 trying to deposit message to terminal 1's mailbox

```
0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: 0


**************** Your id: 36699 ****************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter message:i am 36699 talking to 36694
Enter id of receiver:36694_
```

```


0.Enter other receiver.
1.Send message


Enter choice:1_
```

In terminal 1 after finishing the 1st deposit from terminal 2

```
**********retrieve function**********
Mailbox found
Copy OK
Message: i am 36699 talking to 36694


Type enter to continue


*************** Your id: 36694 ***************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: **********deposit function**********
Copy OK
message: i am 36699 talking to 36694
```

In terminal 3 trying to deposit message to terminal 1's mailbox

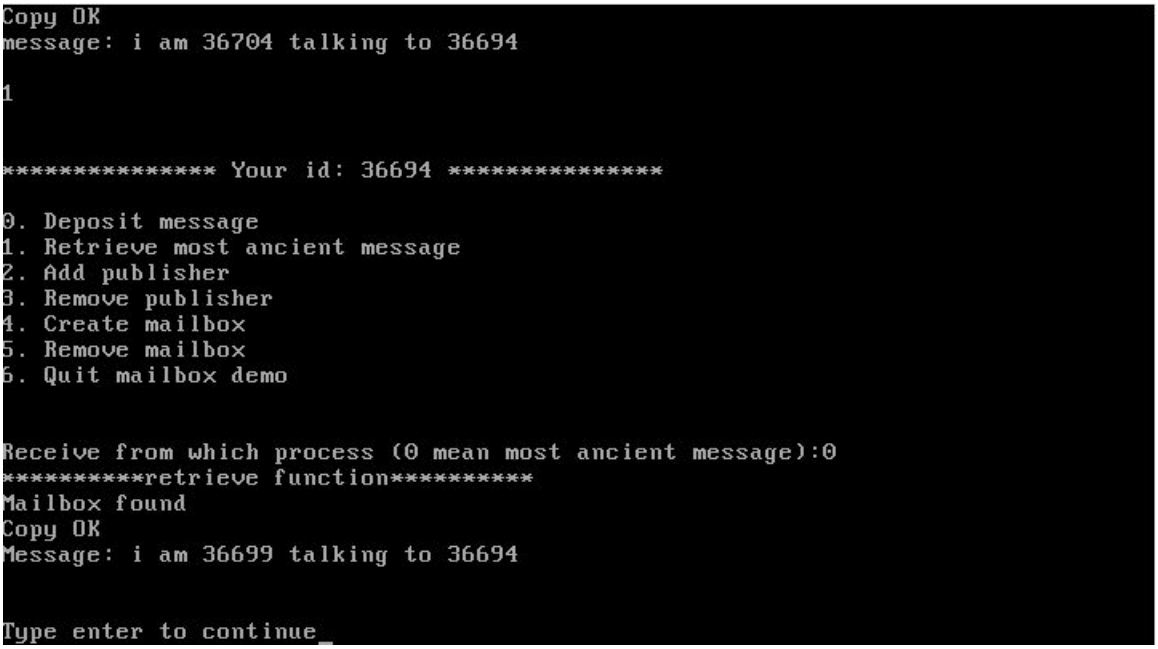
```
0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: 0


*************** Your id: 36704 ***************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter message:i am 36704 talking to 36694
Enter id of receiver:36694
```

```
0.Enter other receiver.
1.Send message

Enter choice:1_
```

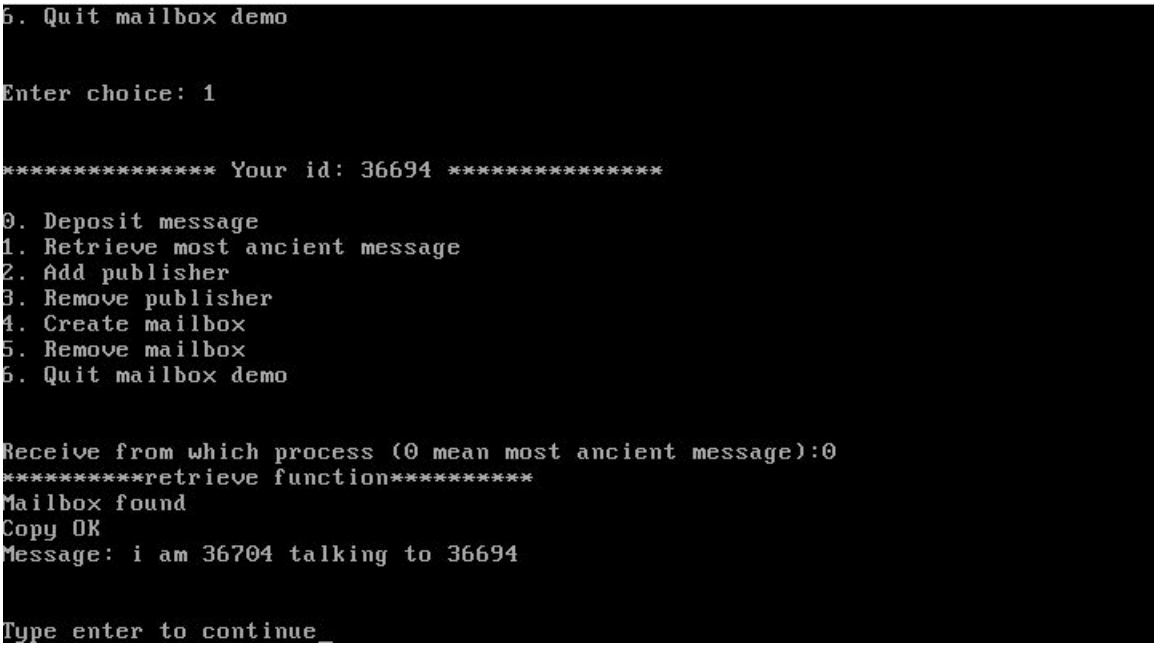
In terminal 1 after finishing the 2nd deposit from terminal 3

```
Type enter to continue


**************** Your id: 36694 ****************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: **********deposit function**********
Copy OK
message: i am 36699 talking to 36694

**********deposit function**********
Copy OK
message: i am 36704 talking to 36694

_
```

In terminal 1's mailbox trying to retrieve the 1st message from its list of messages(then removes it)



In terminal 1's mailbox trying to retrieve the 1st message from its list of messages(which was initially the second message in the mailbox but now becomes the first one after the retrieval of the first message)



Trying to remove publisher to terminal 1's mailbox

```
0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: 3


**************** Your id: 36694 ****************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter publisher id:36699_
```

Successfully removed the publisher

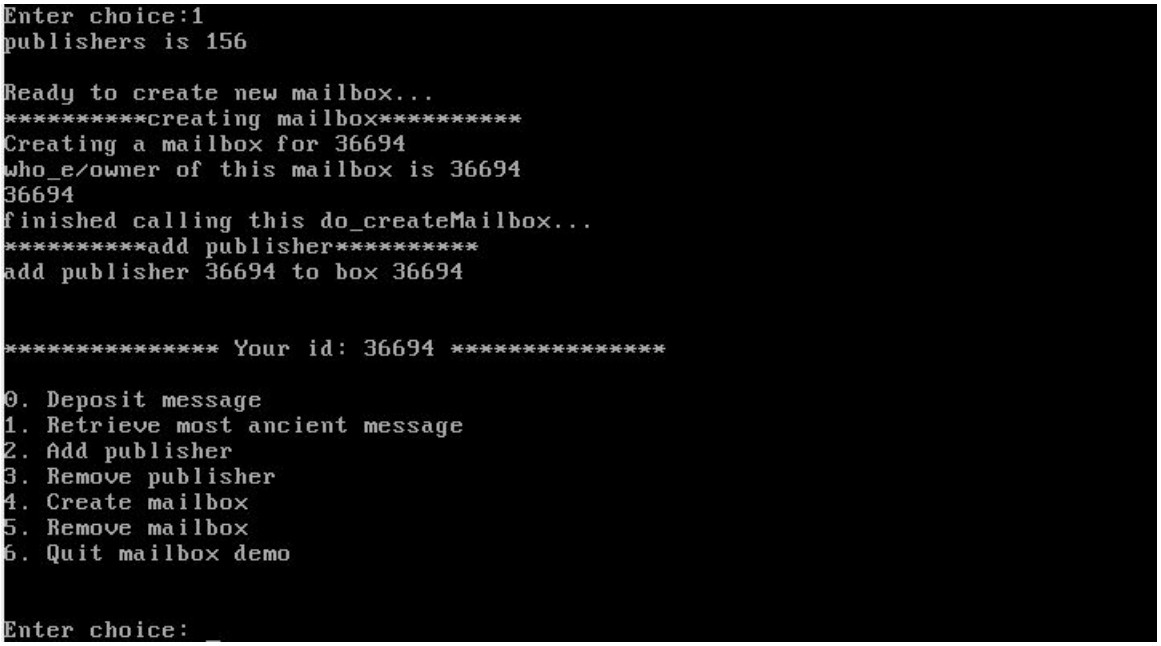
```
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter publisher id:36699
**********remove publisher**********
remove publisher 36699 from box 36694


**************** Your id: 36694 ****************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: _
```

Removing the mailbox of terminal 1

```
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: 5
**********remove function**********
Try to delete mailbox 36694
Mailbox found
sucussfully deleted this mailbox!


**************** Your id: 36694 ****************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter choice: _
```

Creating a new mailbox for terminal 1

```


Enter choice: 4


**************** Your id: 36694 ****************

0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo


Enter id of publisher:36699


0.Allow another publisher.
1.Create mailbox


Enter choice:0
Enter id of publisher:36704_
```


Enter choice:1
publishers is 156
Ready to create new mailbox...
**********creating mailbox**********
Creating a mailbox for 36694
who_e/owner of this mailbox is 36694
36694
finished calling this do_createMailbox...
**********add publisher**********
add publisher 36694 to box 36694
********** Your id: 36694 **********
0. Deposit message
1. Retrieve most ancient message
2. Add publisher
3. Remove publisher
4. Create mailbox
5. Remove mailbox
6. Quit mailbox demo
Enter choice: