Assignment 2: Container Practice

105062539

1. Usage

Server:

```
cd server/
sudo runc run --pid-file /tmp/mnt_server.pid mnt_server
gcc -o server server.c
./server
```

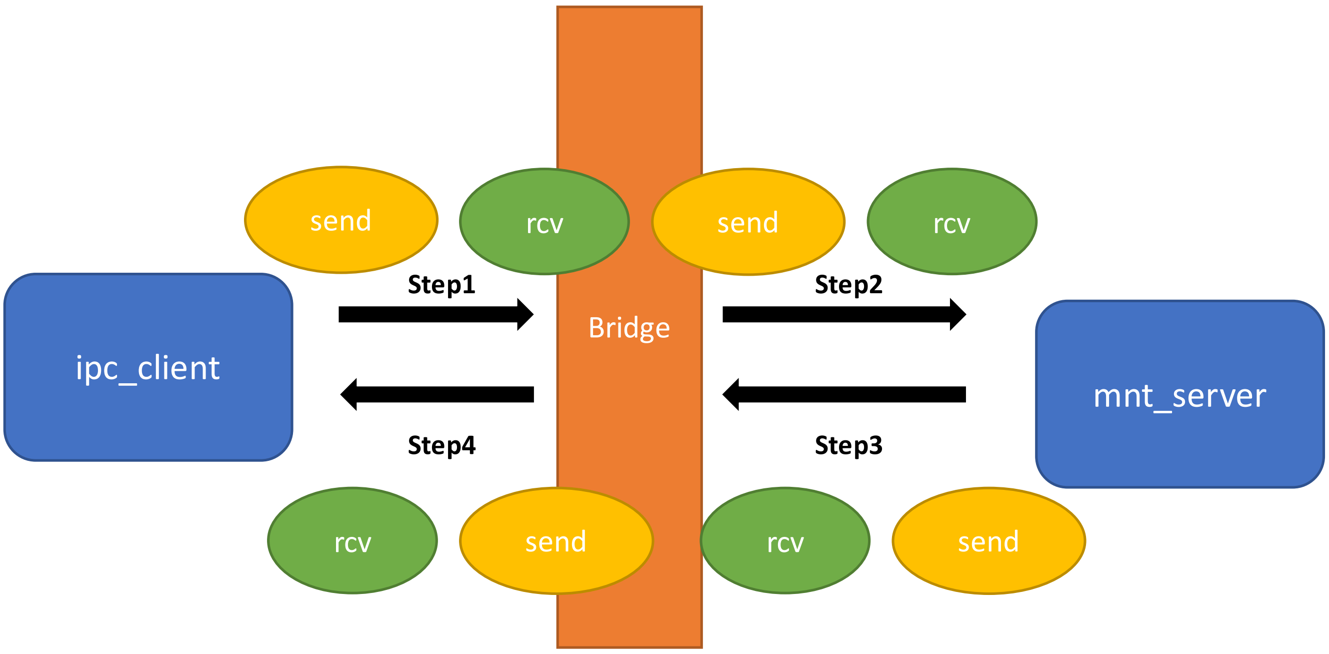
Client:

```
cd  client/
sudo runc run --pid-file /tmp/ipc_client.pid ipc_client
gcc -o clinet client.c
./client
```

Bridge:

```
make
sudo ./bridge
```

2. Work Flow



3. Implementation

   i. Namespace in Linux

      At first, I use setns() to set the process to the “ipc namespace” (obtain pid from */tmp/ipc_client.pid*)

      And, every time I need to communicate with server in “mnt namespace”, I’ll fork the process and use setns() to set the process to the “mnt namespace” (obtain pid from */tmp/mnt_server.pid*)

   ii. IPC Namespace (step 1, 4)

      - Step 1

        i. Both client and bridge build the message queue(key : 5566)

        ```
        msgqid = msgget(5566, MSGPERM|IPC_CREAT);
        ```

        ii. Client send the message to message queue with msg.type = 1

        ```
        rc = msgsnd(msgqid, &msg, sizeof(msg.mtext), 0);
        ```

        iii. Bridge receive the message from message queue and save it in the file (*/tmp/message*)

        ```
        rc = msgrcv(msgqid, &msg, sizeof(msg.mtext), 0, 0)
        ```

      - Step 4

        i. Both client and bridge build the message queue(key : 7788)

        ii. Bridge read the *output* file generated by server (in server/rootfs/output)

        iii. Bridge send the message to message queue with msg1.type = 2

        ```
        rc = msgsnd(msgqid, &msg, sizeof(msg1.mtext), 0);
        ```

        iv. Client receive the message from message queue and stdout

        ```
        rc = msgrcv(msgqid, &msg, sizeof(msg1.mtext), 0, 0)
        ```

   iii. MNT Namespace (step 2, 3)

      - Step 2

        i. Both server and bridge initial the inotify_event

        ```
        inotifyFd = inotify_init();
        ```

        ii. Bridge monitor the same folder in current directory with *IN_DELETE* state

        ```
        inotify_add_watch(inotifyFd, getcwd(NULL,0), IN_DELETE);
        ```

        iii. Bridge writes the sentence from */tmp/message* to *message* file and wait in while loop until the *message* to be received and deleted by server

iv. Server monitor the folder in current directory with *IN_CLOSE_WRITE* state

```
inotify_add_watch(inotifyFd, cwd, IN_CLOSE_WRITE);
```

v. When get the inotify_event in *IN_CLOSE_WRITE* with the event name “message”, server read the sentence recorded in message file

```
if((event->mask & IN_CLOSE_WRITE) && !strcmp(event-
>name, "message"))
```

vi. Server deletes the message file and bridge can go to step 3

```
system("rm -f message");
goto send;
```

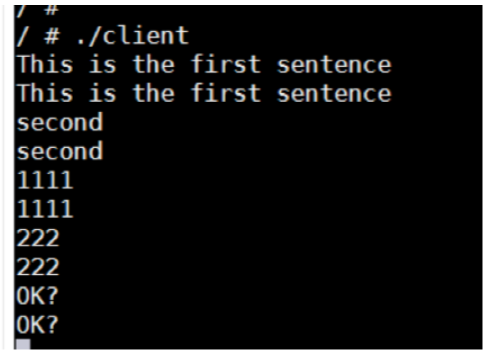
- Step 3

i. Server writes the sentence from to *message1* file and wait in while loop until the *message* to be received by bridge and output the file

ii. Server go to step 3 waiting for next sentence

```
if(!strcmp(event->name, "output")){
        goto rcv;
}
```

4. Result



```
/ #
/ # ./client
This is the first sentence
This is the first sentence
second
second
1111
1111
222
222
OK?
OK?
```