This process will be maintaining its own temperature and will send that value along to the central process after recomputing what it should be given how that quantity changes according to the value received from central.

The initial value for the temp is provided as command line argument #1, and the second argument is the value for which outer process this is to the central process. Temps will continue to be updated until the stable field is received as a one, and then this outer process will halt.

All msg system call return values are examined for errors ...

```c
#include <stdio.h>
#include <sys/errno.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "message.h"

int main(int argc, char *argv[])
{
  struct {
    long priority;
    int temp;
    int pid;
    int stable;
  } msgp;

  int count, ct, pid, flag, temp;
  int msqid, serverid;
  int status;
  long mtyp;
  long key;

  if (argc < 3) {
    printf("error - need two args\n");
    return 1;
  }

  pid = atoi(argv[2]);
  temp = atoi(argv[1]);

  msgp.priority = 2;
  key = 20 + pid;

  if ((msqid = msgget(key, 0666 | IPC_CREAT)) < 0) {
    perror("Multiplier failed to open its message queue.\n");
    return 1;
  }

  printf("msgid = %d\n", msqid);

  if ((serverid = msgget(20, 0666 | IPC_CREAT)) < 0) {
    perror("Multiplier failed to open the queue of the central process.\n");
    return 1;
  }

  printf("msgid = %d\n", msqid);
```

do {
    msgp.pid = pid;
    msgp.temp = temp;
    msgp.priority= 2;/* all messages will be "priority" 2*/
    msgp.stable = 0;

    // send my temperature to the central mailbox
    if ((status = msgsnd(serverid,&msgp,sizeof(msgp)-sizeof(long),0)) < 0)
        printf("process %d - error in send \n",key-20);

    // now obtain the temperature of the system
    if ((status = msgrcv(msqid,&msgp,sizeof(msgp)-sizeof(long),2,0)) < 0)
        printf("process %d - error in receive \n",key-20);

    // calculate the new temperature
    temp = (3 * temp + 2 * msgp.temp) / 5;
} while (msgp.stable != 1);

printf("Final temp for %d is %d.\n", pid, temp);

// we're finished - now remove the message queue
if (msgctl(msqid, IPC_RMID, &buffer) < 0){
    perror("Multiplier failed to remove its message queue.\n");
    return 1;
}

return 0;