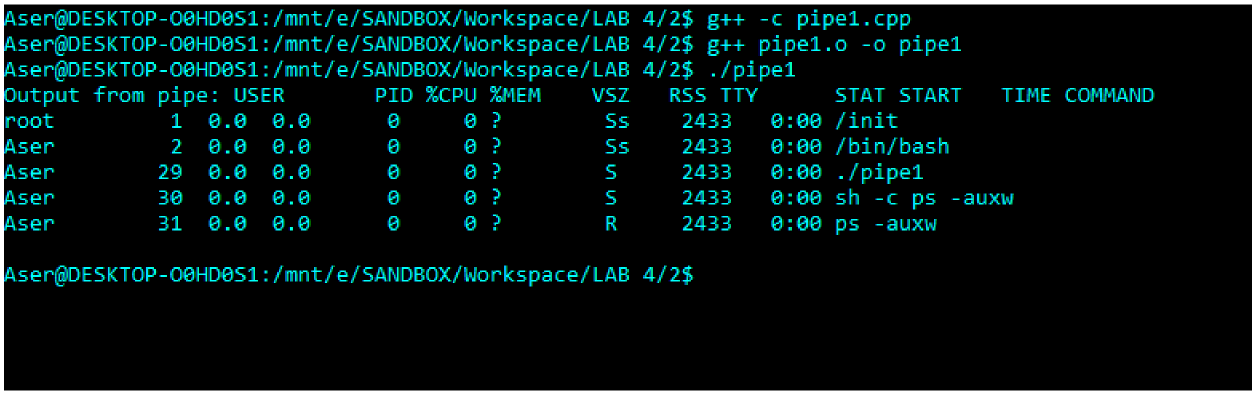
2:Process pipes

pipe1.cpp

It gives the following output:



```
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ g++ -c pipe1.cpp
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ g++ pipe1.o -o pipe1
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ ./pipe1
Output from pipe: USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0      0     0 ?        Ss    2433   0:00 /init
Aser         2  0.0  0.0      0     0 ?        Ss    2433   0:00 /bin/bash
Aser        29  0.0  0.0      0     0 ?        S     2433   0:00 ./pipe1
Aser        30  0.0  0.0      0     0 ?        S     2433   0:00 sh -c ps -auxw
Aser        31  0.0  0.0      0     0 ?        R     2433   0:00 ps -auxw

Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$
```

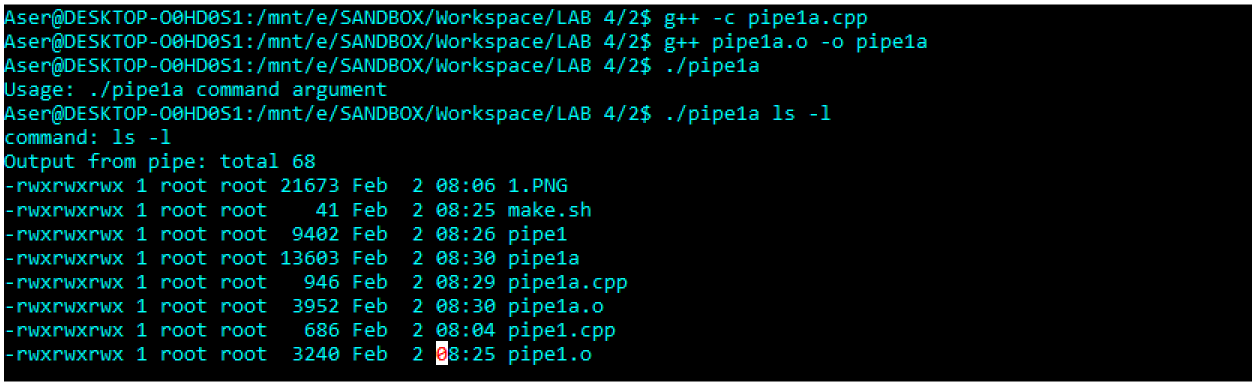
The program creates a FILE variable to read from the pipe. `popen(“ps -auxw”, “r”)` creates a pipe, forks and creates a shell and runs command `ps –auxw` with ‘`r`’ which means that the pipe will read the shell output. So the program reads the shell output and stores them in the charcter array and prints it on the terminal.

pipe1a.cpp

Code:

```
//pipe1a.cpp
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
  FILE *fpi;                          //for reading a pipe
  if(argc != 3){
    cout<<"Usage: "<<argv[0]<<" command argument"<<endl;
    return 0;
  }
  char command[15];
  strcpy(command,argv[1]);
  strcat(command," ");
  strcat(command,argv[2]);
  cout<<"command: "<<command<<endl;
  char buffer[BUFSIZ+1];              //BUFSIZ defined in <stdio.h>
  int chars_read;
  memset ( buffer, 0,sizeof(buffer)); //clear buffer
  fpi = popen ( command, "r" );  //pipe to command "ps -auxw"
  if ( fpi != NULL ) {
    //read data from pipe into buffer
    chars_read = fread(buffer, sizeof(char), BUFSIZ, fpi );
    if ( chars_read > 0 )
     cout << "Output from pipe: " << buffer << endl;
    pclose ( fpi );              //close the pipe
    return 0;
  }
  return 1;
}
```

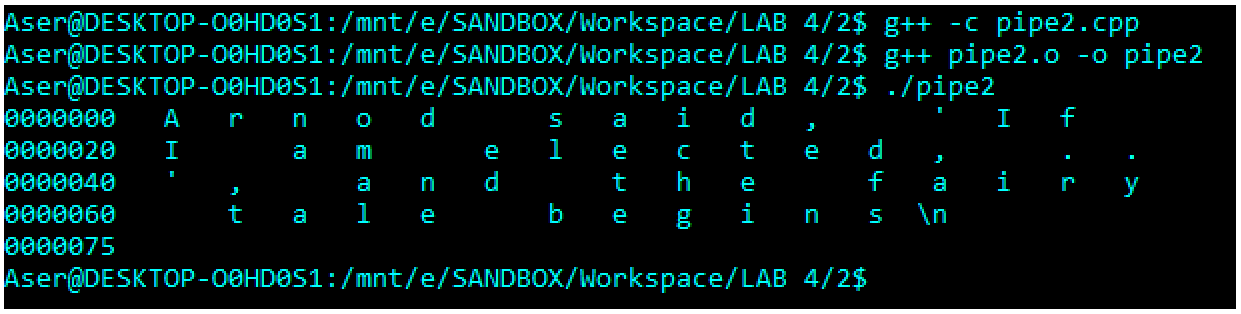
Output:



```
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ g++ -c pipe1a.cpp
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ g++ pipe1a.o -o pipe1a
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ ./pipe1a
Usage: ./pipe1a command argument
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ ./pipe1a ls -l
command: ls -l
Output from pipe: total 68
-rwxrwxrwx 1 root root 21673 Feb  2 08:06 1.PNG
-rwxrwxrwx 1 root root    41 Feb  2 08:25 make.sh
-rwxrwxrwx 1 root root  9402 Feb  2 08:26 pipe1
-rwxrwxrwx 1 root root 13603 Feb  2 08:30 pipe1a
-rwxrwxrwx 1 root root   946 Feb  2 08:29 pipe1a.cpp
-rwxrwxrwx 1 root root  3952 Feb  2 08:30 pipe1a.o
-rwxrwxrwx 1 root root   686 Feb  2 08:04 pipe1.cpp
-rwxrwxrwx 1 root root  3240 Feb  2 08:25 pipe1.o
```

pipe2.cpp

It has the output



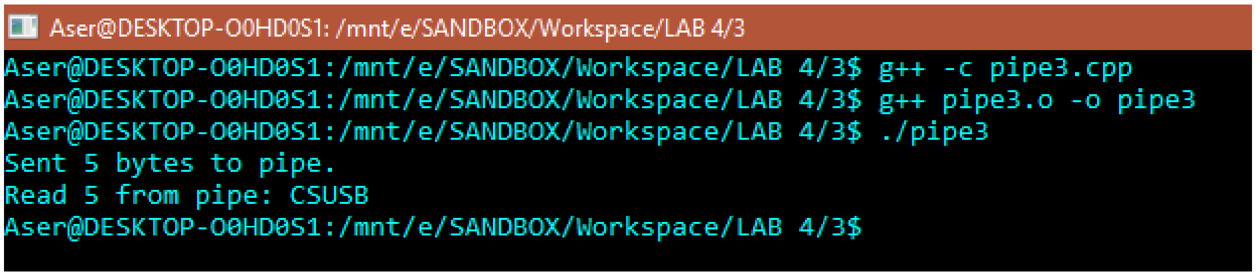
```
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ g++ -c pipe2.cpp
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ g++ pipe2.o -o pipe2
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$ ./pipe2
0000000   A   r   n   o   d       s   a   i   d   ,       '   I   f
0000020   I       a   m       e   l   e   c   t   e   d   ,       .   .
0000040   '   ,       a   n   d       t   h   e       f   a   i   r   y
0000060       t   a   l   e       b   e   g   i   n   s  \n
0000075
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/2$
```

The code sends the text to the pipe which runs the `od` command with `c` parameter. The final result is the printable characters in the text given to it.

3:

pipe3.cpp

The program displays the number of bytes sent to the pipe and displays the data read from the pipe. The program creates a pipe with `pipe()` function with `fd` as argument. After the pipe is created the `fd[0]` carries the read end of the pipe and `fd[1]` carries the write end of the pipe. With write function, the program writes to the pipe using `fd[1]`. Write return the number of bytes written in the pipe, which is diplayed in the first line. Then the program reads from the pipe using `read` with read end of the pipe i.e. `fd[0]` and displays the number of bytes read.



```
Aser@DESKTOP-O0HD0S1: /mnt/e/SANDBOX/Workspace/LAB 4/3
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/3$ g++ -c pipe3.cpp
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/3$ g++ pipe3.o -o pipe3
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/3$ ./pipe3
Sent 5 bytes to pipe.
Read 5 from pipe: CSUSB
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/3$
```

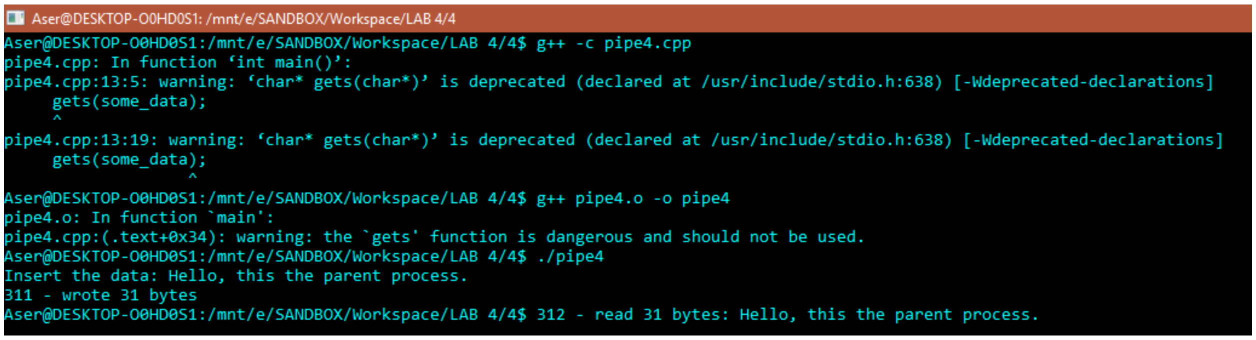
4:

```
char some_data[] = "123";
```

is changed to

```
char some_data[256];
printf("Insert the data to send: ");
gets(some_data);
```

OUTPUT:

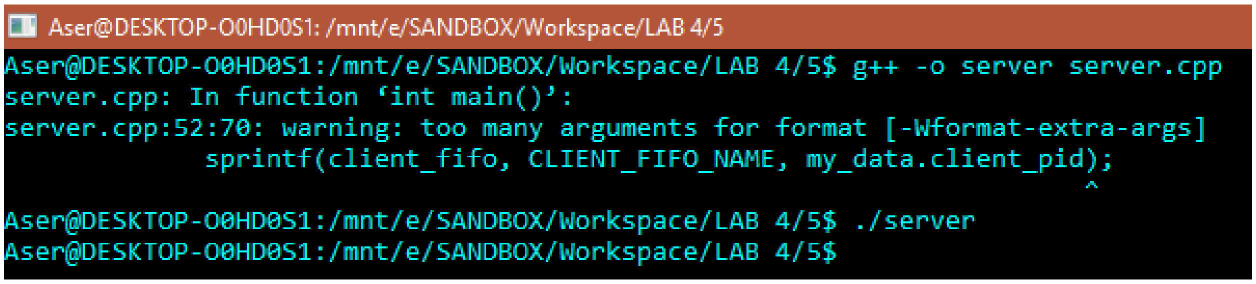


```
Aser@DESKTOP-O0HD0S1: /mnt/e/SANDBOX/Workspace/LAB 4/4
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/4$ g++ -c pipe4.cpp
pipe4.cpp: In function 'int main()':
pipe4.cpp:13:5: warning: 'char* gets(char*)' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
     gets(some_data);
     ^
pipe4.cpp:13:19: warning: 'char* gets(char*)' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
     gets(some_data);
                   ^
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/4$ g++ pipe4.o -o pipe4
pipe4.o: In function `main':
pipe4.cpp:(.text+0x34): warning: the `gets' function is dangerous and should not be used.
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/4$ ./pipe4
Insert the data: Hello, this the parent process.
311 - wrote 31 bytes
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/4$ 312 - read 31 bytes: Hello, this the parent process.
```

5:

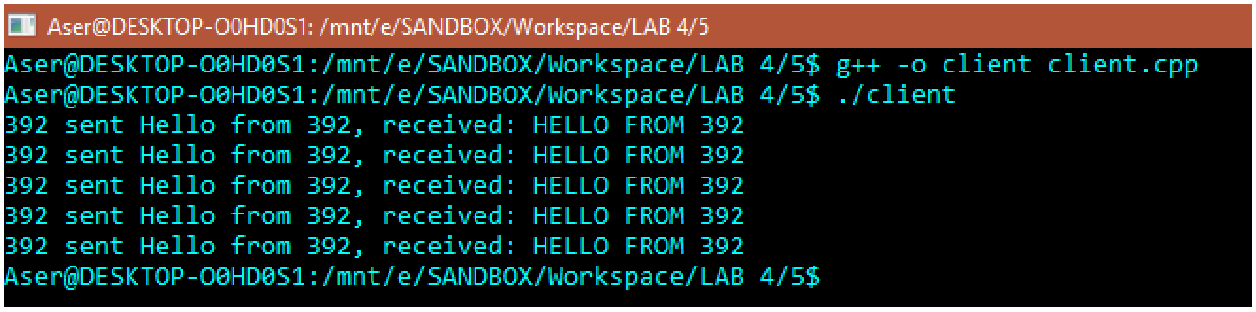
Output

Server



```
Aser@DESKTOP-O0HD0S1: /mnt/e/SANDBOX/Workspace/LAB 4/5
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$ g++ -o server server.cpp
server.cpp: In function 'int main()':
server.cpp:52:70: warning: too many arguments for format [-Wformat-extra-args]
             sprintf(client_fifo, CLIENT_FIFO_NAME, my_data.client_pid);
                                                                      ^
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$ ./server
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$
```

Client



```
Aser@DESKTOP-O0HD0S1: /mnt/e/SANDBOX/Workspace/LAB 4/5
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$ g++ -o client client.cpp
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$ ./client
392 sent Hello from 392, received: HELLO FROM 392
392 sent Hello from 392, received: HELLO FROM 392
392 sent Hello from 392, received: HELLO FROM 392
392 sent Hello from 392, received: HELLO FROM 392
392 sent Hello from 392, received: HELLO FROM 392
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$
```

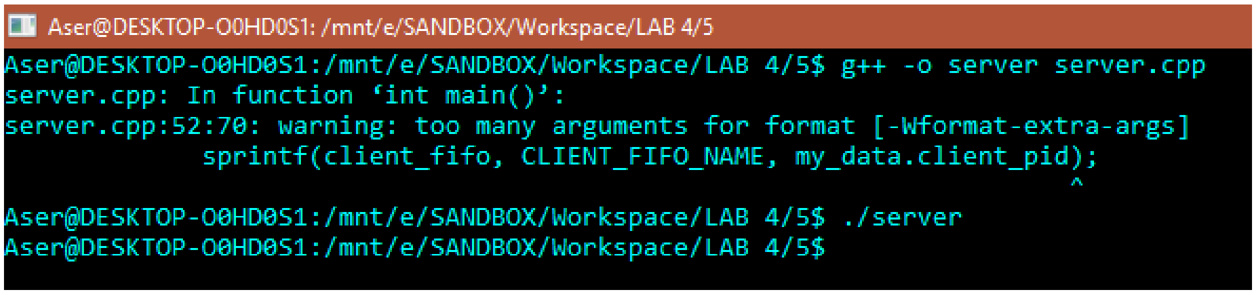
After Modification:

Code

Change line 49 in sever.cpp to:

```
*tmp_char_ptr = tolower(*tmp_char_ptr);
```

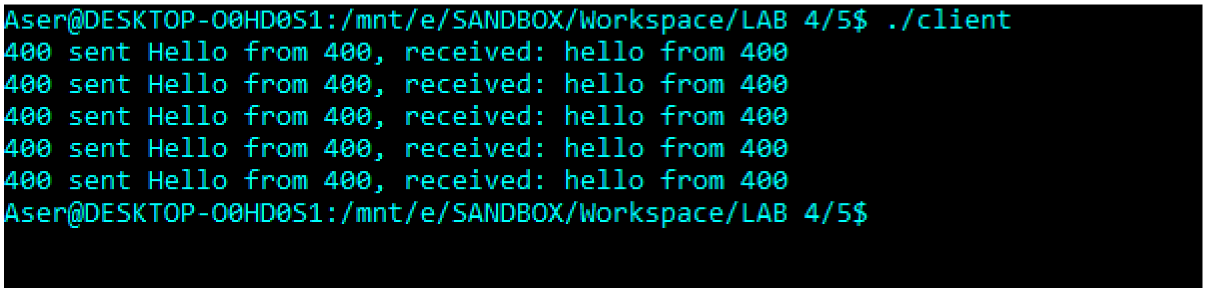
Server



```
Aser@DESKTOP-O0HD0S1: /mnt/e/SANDBOX/Workspace/LAB 4/5
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$ g++ -o server server.cpp
server.cpp: In function ‘int main()’:
server.cpp:52:70: warning: too many arguments for format [-Wformat-extra-args]
             sprintf(client_fifo, CLIENT_FIFO_NAME, my_data.client_pid);
                                                                      ^

Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$ ./server
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$
```

Client



```
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$ ./client
400 sent Hello from 400, received: hello from 400
400 sent Hello from 400, received: hello from 400
400 sent Hello from 400, received: hello from 400
400 sent Hello from 400, received: hello from 400
400 sent Hello from 400, received: hello from 400
Aser@DESKTOP-O0HD0S1:/mnt/e/SANDBOX/Workspace/LAB 4/5$
```

6: **xv6**

Kernel Dissambley:

```
Program received signal SIGINT, Interrupt.
0x801041f8 in ?? ()
(gdb) file kernel
A program is being debugged already.
Are you sure you want to change the file? (y or n)
Please answer y or n.
A program is being debugged already.
Are you sure you want to change the file? (y or n) n
File not changed.
```

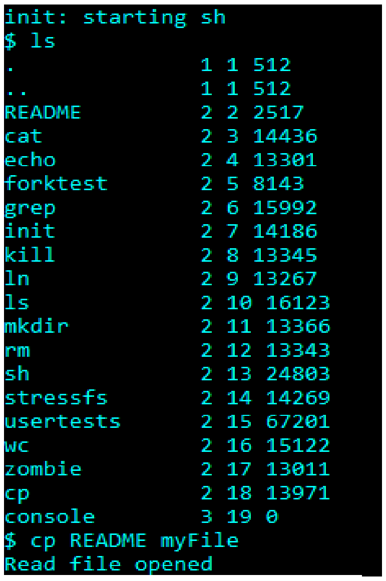
```
(gdb) file kernel
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Reading symbols from kernel...done.
(gdb) set disassembly-flavor intel
(gdb) disass
Dump of assembler code for function acquire:
   0x801041b0 <+0>:     push   ebp
   0x801041b1 <+1>:     mov    ebp,esp
   0x801041b3 <+3>:     sub    esp,0x18
   0x801041b6 <+6>:     pushf
   0x801041b7 <+7>:     pop    ecx
   0x801041b8 <+8>:     cli
   0x801041b9 <+9>:     mov    eax,gs:0x0
   0x801041bf <+15>:    mov    edx,DWORD PTR [eax+0xac]
   0x801041c5 <+21>:    test   edx,edx
   0x801041c7 <+23>:    jne    0x801041d5 <acquire+37>
   0x801041c9 <+25>:    and    ecx,0x200
   0x801041cf <+31>:    mov    DWORD PTR [eax+0xb0],ecx
   0x801041d5 <+37>:    add    edx,0x1
   0x801041d8 <+40>:    mov    DWORD PTR [eax+0xac],edx
   0x801041de <+46>:    mov    edx,DWORD PTR [ebp+0x8]
   0x801041e1 <+49>:    mov    ecx,DWORD PTR [edx]
   0x801041e3 <+51>:    test   ecx,ecx
   0x801041e5 <+53>:    je     0x801041ec <acquire+60>
   0x801041e7 <+55>:    cmp    eax,DWORD PTR [edx+0x8]
   0x801041ea <+58>:    je     0x8010422a <acquire+122>
   0x801041ec <+60>:    mov    ecx,0x1
   0x801041f1 <+65>:    jmp    0x801041fb <acquire+75>
   0x801041f3 <+67>:    nop
   0x801041f4 <+68>:    lea    esi,[esi+eiz*1+0x0]
=> 0x801041f8 <+72>:    mov    edx,DWORD PTR [ebp+0x8]
   0x801041fb <+75>:    mov    eax,ecx
   0x801041fd <+77>:    lock xchg DWORD PTR [edx],eax
---Type <return> to continue, or q <return> to quit---
```

2:

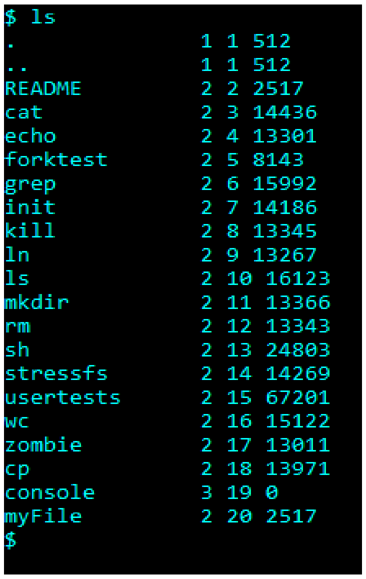
Code:

```
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"
#define O_RDWR    0x002
#define O_CREATE  0x200
#define BUF_SIZE 256
int main(int argc, char *argv[])
{
  if (argc !=3)
  {
    printf(1, "please input the command as [cp source
destination]\n");
    exit();
  }
  int fd0,fd1;
  char buf1[512];
  if((fd0 = open(argv[1],0)) < 0 ){
    printf(1, "cp: cannot open %s %d\n", argv[1] ,fd0);
    exit();
  }else{
    printf(1,"Read file opened\n" );
    if((fd1 = open(argv[2],O_CREATE | O_RDWR)) < 0 ){
       printf(1, "cp: cannot open %s %d\n", argv[2] ,fd1);
      exit();
    }
    int n;
    while((n = read(fd0, buf1, sizeof(buf1))) > 0){
      write(fd1, buf1, n);
    }
  }
exit();
return 0;
}
```

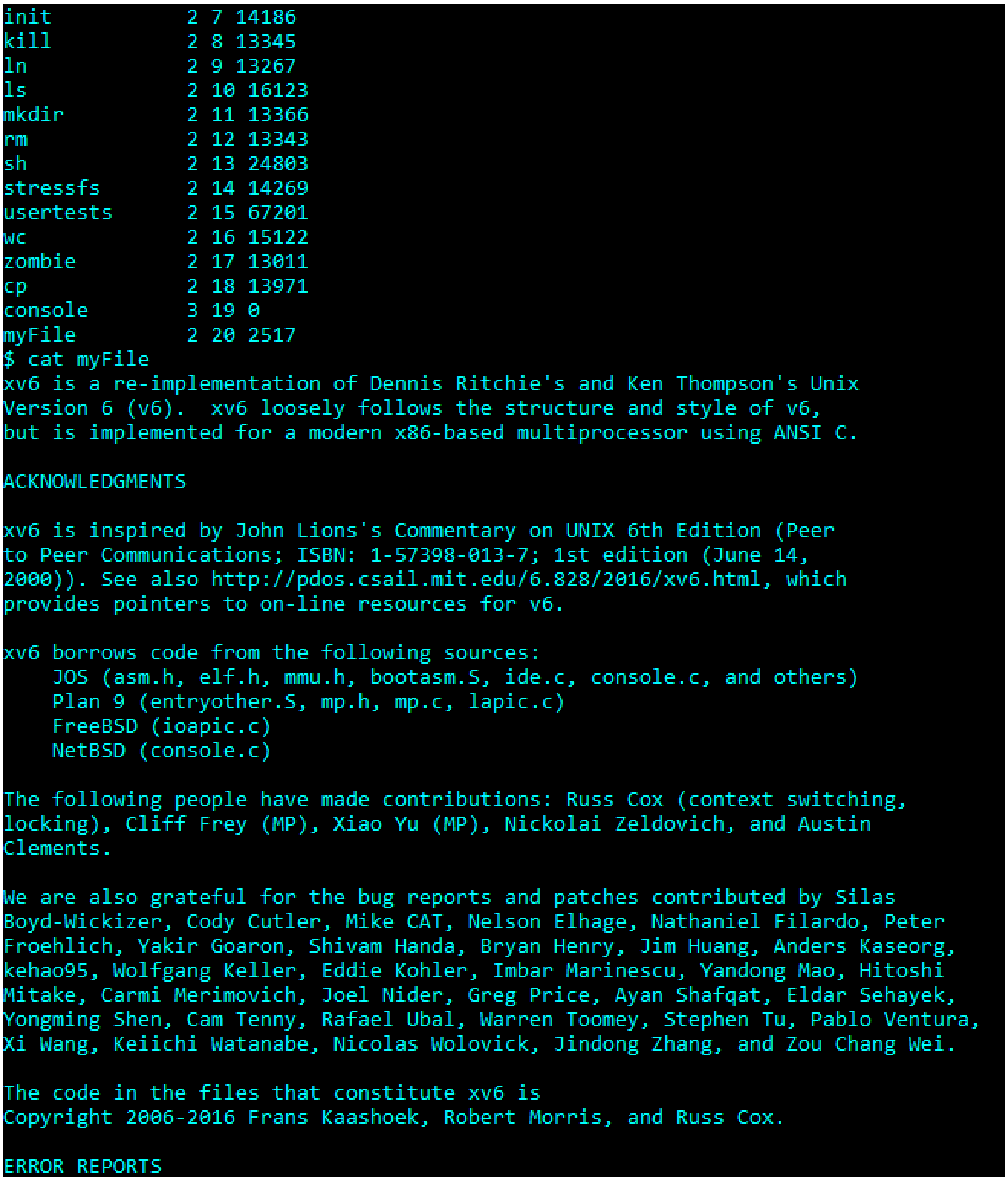
Output:



```
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2517
cat            2 3 14436
echo           2 4 13301
forktest       2 5 8143
grep           2 6 15992
init           2 7 14186
kill           2 8 13345
ln             2 9 13267
ls             2 10 16123
mkdir          2 11 13366
rm             2 12 13343
sh             2 13 24803
stressfs       2 14 14269
usertests      2 15 67201
wc             2 16 15122
zombie         2 17 13011
cp             2 18 13971
console        3 19 0
$ cp README myFile
Read file opened
```



```
$ ls
.              1 1 512
..             1 1 512
README         2 2 2517
cat            2 3 14436
echo           2 4 13301
forktest       2 5 8143
grep           2 6 15992
init           2 7 14186
kill           2 8 13345
ln             2 9 13267
ls             2 10 16123
mkdir          2 11 13366
rm             2 12 13343
sh             2 13 24803
stressfs       2 14 14269
usertests      2 15 67201
wc             2 16 15122
zombie         2 17 13011
cp             2 18 13971
console        3 19 0
myFile         2 20 2517
$
```



```
init           2 7 14186
kill           2 8 13345
ln             2 9 13267
ls             2 10 16123
mkdir          2 11 13366
rm             2 12 13343
sh             2 13 24803
stressfs       2 14 14269
usertests      2 15 67201
wc             2 16 15122
zombie         2 17 13011
cp             2 18 13971
console        3 19 0
myFile         2 20 2517
$ cat myFile
xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix
Version 6 (v6).  xv6 loosely follows the structure and style of v6,
but is implemented for a modern x86-based multiprocessor using ANSI C.

ACKNOWLEDGMENTS

xv6 is inspired by John Lions's Commentary on UNIX 6th Edition (Peer
to Peer Communications; ISBN: 1-57398-013-7; 1st edition (June 14,
2000)). See also http://pdos.csail.mit.edu/6.828/2016/xv6.html, which
provides pointers to on-line resources for v6.

xv6 borrows code from the following sources:
    JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)
    Plan 9 (entryother.S, mp.h, mp.c, lapic.c)
    FreeBSD (ioapic.c)
    NetBSD (console.c)

The following people have made contributions: Russ Cox (context switching,
locking), Cliff Frey (MP), Xiao Yu (MP), Nickolai Zeldovich, and Austin
Clements.

We are also grateful for the bug reports and patches contributed by Silas
Boyd-Wickizer, Cody Cutler, Mike CAT, Nelson Elhage, Nathaniel Filardo, Peter
Froehlich, Yakir Goaron, Shivam Handa, Bryan Henry, Jim Huang, Anders Kaseorg,
kehao95, Wolfgang Keller, Eddie Kohler, Imbar Marinescu, Yandong Mao, Hitoshi
Mitake, Carmi Merimovich, Joel Nider, Greg Price, Ayan Shafqat, Eldar Sehayek,
Yongming Shen, Cam Tenny, Rafael Ubal, Warren Toomey, Stephen Tu, Pablo Ventura,
Xi Wang, Keiichi Watanabe, Nicolas Wolovick, Jindong Zhang, and Zou Chang Wei.

The code in the files that constitute xv6 is
Copyright 2006-2016 Frans Kaashoek, Robert Morris, and Russ Cox.

ERROR REPORTS
```