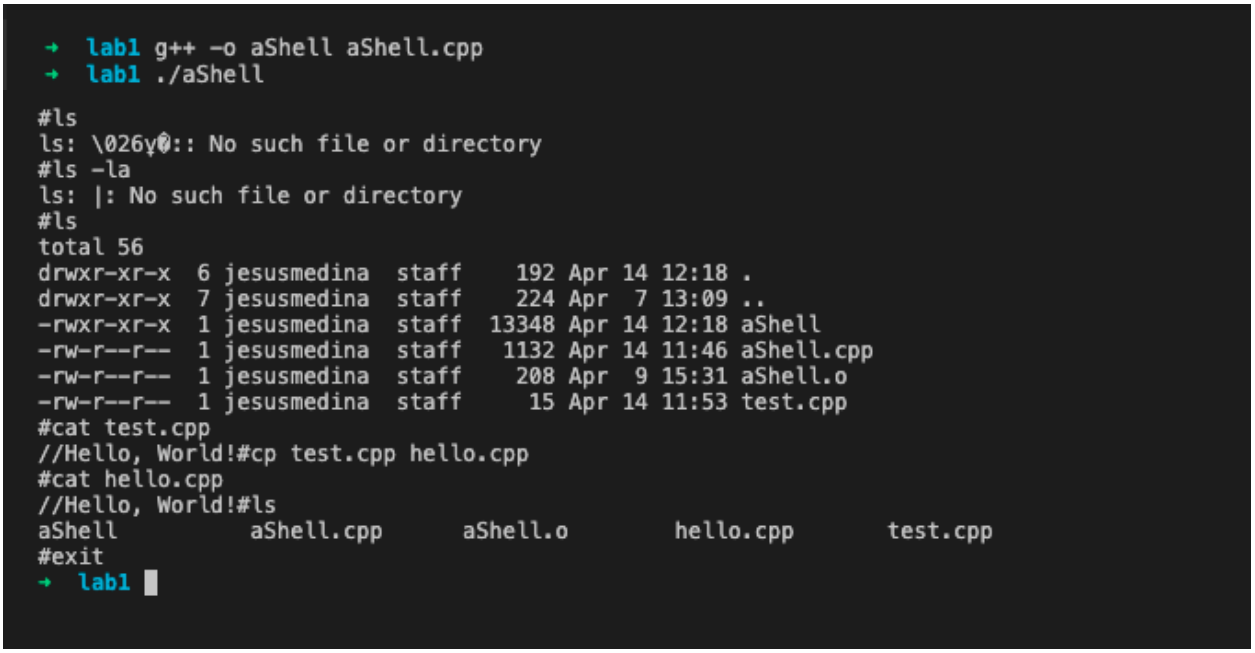
LAB 1 Report
Jesus Medina
Luis Altamirano

Writing a Simple Shell

aShell.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>
#include <string.h>

void read_command(char cmd[], char *par[])
{
        char line[1024];
        int count = 0, i = 0, j = 0;
        char *array[100], *pch;

        for (;;) {
                int c = fgetc(stdin);
                line[count++] = (char) c;
                if (c == '\n') break;
        }
        if (count == 1) return;
        pch = strtok(line, " \n");

        while (pch != NULL) {
                array[i++] = strdup(pch);
                pch = strtok(NULL, " \n");
        }

        strcpy(cmd, array[0]);

        for (int j = 0; j < i; j++) {
                par[j] = array[j];
        }
        par[i] = NULL;
}

void type_prompt()
{
        static int first_time = 1;
        if (first_time) {
                const char* CLEAR_SCREEN_ANSI = " \e[1;1H\e[2J";
                write(STDOUT_FILENO, CLEAR_SCREEN_ANSI, 12);
                first_time = 0;
        }
        printf("#");
}
```

```
int main()
{
        char cmd[100], command[100], *parameters[20];

        char *envp[] = { (char *) "PATH=/bin", 0 };
        while (1) {
                type_prompt();
                read_command (command, parameters);
                if (fork() != 0)
                        wait(NULL);
                else {
                        strcpy(cmd, "/bin/");
                        strcat(cmd, command);
                        execve(cmd, parameters, envp);
                }
                if (strcmp(command, "exit") == 0) {
                        break;
                }
        }
        return 0;
}
```

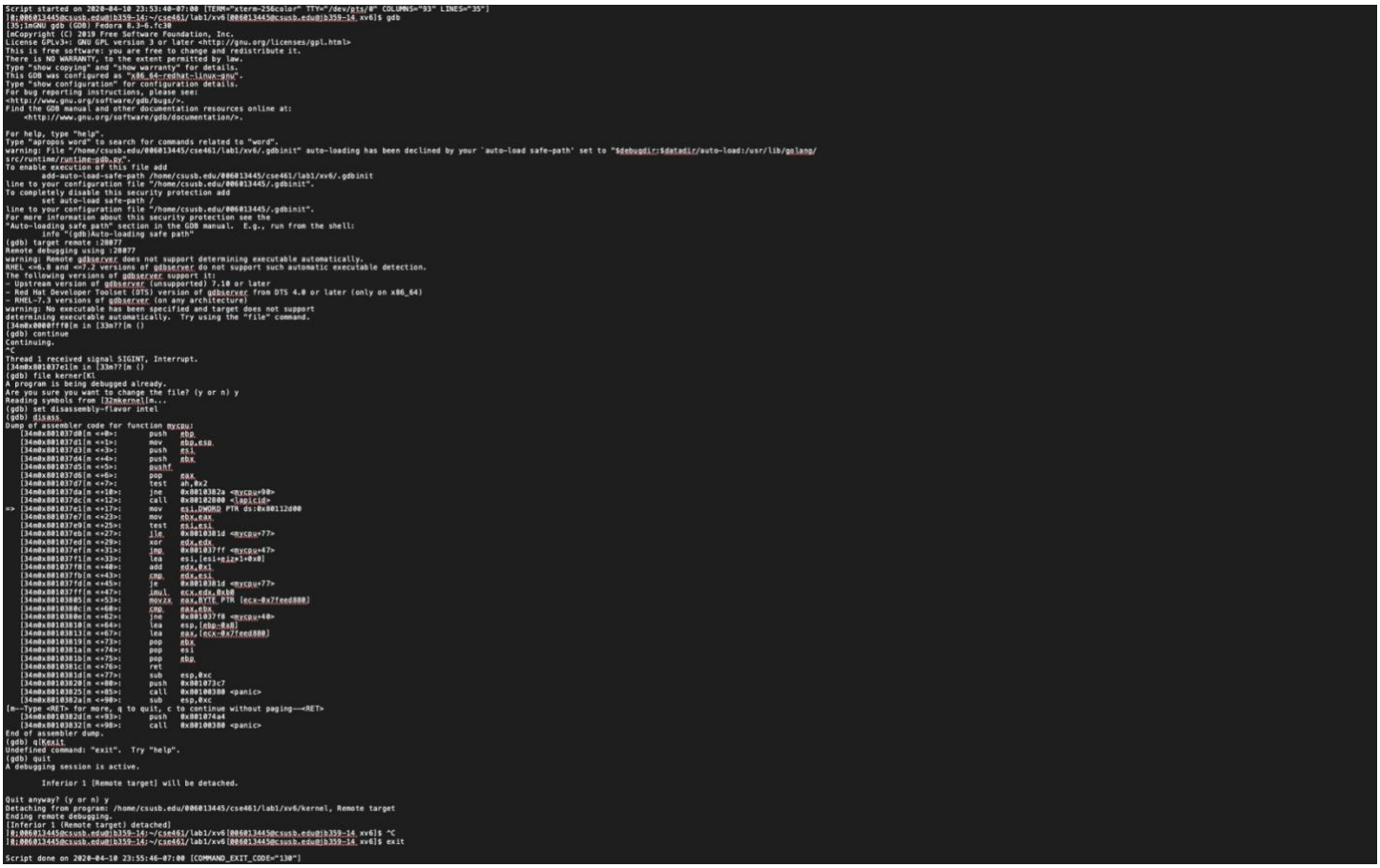


1. Basics of XV6

```
[005172852@csusb.edu@jb358-2 xv6]$ make
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.579085 s, 8.8 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.00180471 s, 284 kB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
416+1 records in
416+1 records out
213108 bytes (213 kB, 208 KiB) copied, 0.00716937 s, 29.7 MB/s
[005172852@csusb.edu@jb358-2 xv6]$ make qemu-nox



SeaBIOS (version 1.12.0-2.fc30)


iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF91280+1FED1280 C980



Booting from Hard Disk...
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2290
cat            2 3 16140
echo           2 4 14976
forktest       2 5 9300
grep           2 6 18332
init           2 7 15580
kill           2 8 15008
ln             2 9 14876
ls             2 10 17500
mkdir          2 11 15120
rm             2 12 15100
sh             2 13 27620
stressfs       2 14 16012
usertests      2 15 66948
wc             2 16 16860
cp             2 17 15564
zombie         2 18 14692
console        3 19 0
$ echo cse 461
cse 461
$ cat README
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, Anton Burtsev, Cody Cutler, Mike CAT, Tej Chajed, Nelson Elhage,
Saar Ettinger, Alice Ferrazzi, Nathaniel Filardo, Peter Froehlich, Yakir Goaron,
Shivam Handa, Bryan Henry, Jim Huang, Alexander Kapshuk, Anders Kaseorg,
```

```
kehao95, Wolfgang Keller, Eddie Kohler, Austin Liew, 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, Grant Wu, Jindong
Zhang, Icenowy Zheng, 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

Please send errors and suggestions to Frans Kaashoek and Robert Morris
(kaashoek,rtm@mit.edu). The main purpose of xv6 is as a teaching
operating system for MIT's 6.828, so we are more interested in
simplifications and clarifications than new features.

BUILDING AND RUNNING XV6

To build xv6 on an x86 ELF machine (like Linux or FreeBSD), run
"make". On non-x86 or non-ELF machines (like OS X, even on x86), you
will need to install a cross-compiler gcc suite capable of producing
x86 ELF binaries. See http://pdos.csail.mit.edu/6.828/2016/tools.html.
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC
simulator and run "make qemu".
$ grep os README
Version 6 (v6).  xv6 loosely follows the structure and style of v6,
2000)). See also http://pdos.csail.mit.edu/6.828/2016/xv6.html, which
Mao, Hitoshi Mitake, Carmi Merimovich, Joel Nider, Greg Price, Ayan Shafqat,
(kaashoek,rtm@mit.edu). The main purpose of xv6 is as a teaching
will need to install a cross-compiler gcc suite capable of producing
x86 ELF binaries. See http://pdos.csail.mit.edu/6.828/2016/tools.html.
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC
$ cat README | grep os | wc
7 63 486
$ echo cse 460 lab report > myFile
$ cat myFile
cse 460 lab report
$
```

2. Basics of XV6 Continued(gdb and disassemble)



2. Debugging

Debugging of xv6 following Part 3 of the lab.

```
Script started on 2020-04-10 23:58:11-07:00 [TERM="xterm-256color" TTY="/dev/pts/0" COLUMNS="93" LINES="35"]
]0;006013445@csusb.edu@jb359-14:~/cse461/lab1/xv6[006013445@csusb.edu@jb359-14 xv6]$ gdb
[35;1mGNU gdb (GDB) Fedora 8.3-6.fc30
[mCopyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
warning: File "/home/csusb.edu/006013445/cse461/lab1/xv6/.gdbinit" auto-loading has been declined by your `auto-load safe-path' set to "$debugdir:$datadir/auto-load:/usr/lib/golang/
src/runtime/runtime-gdb.py".
To enable execution of this file add
        add-auto-load-safe-path /home/csusb.edu/006013445/cse461/lab1/xv6/.gdbinit
line to your configuration file "/home/csusb.edu/006013445/.gdbinit".
To completely disable this security protection add
        set auto-load safe-path /
line to your configuration file "/home/csusb.edu/006013445/.gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"
(gdb) target remote ;[K:28077
Remote debugging using :28077
warning: Remote gdbserver does not support determining executable automatically.
RHEL <=6.8 and <=7.2 versions of gdbserver do not support such automatic executable detection.
The following versions of gdbserver support it:
- Upstream version of gdbserver (unsupported) 7.10 or later
- Red Hat Developer Toolset (DTS) version of gdbserver from DTS 4.0 or later (only on x86_64)
- RHEL-7.3 versions of gdbserver (on any architecture)
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
[34m0x0000fff0[m in [33m??[m ()
(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 [32mkernel[m...
(gdb) gdb break switch
Undefined command: "gdb".  Try "help".
(gdb) break switch
Function "switch" not defined.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) break swtch
Breakpoint 1 at [34m0x801046eb[m: file [32mswtch.S[m, line 11.
(gdb) continue
Continuing.
[Switching to Thread 2]

Thread 2 hit Breakpoint 1, [33mswtch[m () at [32mswtch.S[m:11
11        movl [35m4[m[31m(%[m[32mesp[m[31m),[m [31m%[m[32meax[m
(gdb) step
12        movl [35m8[m[31m(%[m[32mesp[m[31m),[m [31m%[m[32medx[m
(gdb) step
15        pushl [31m%[m[32mebp[m
(gdb) step
[33mswtch[m () at [32mswtch.S[m:16
16        pushl [31m%[m[32mebx[m
(gdb) step
[33mswtch[m () at [32mswtch.S[m:17
17        pushl [31m%[m[32mesi[m
(gdb) step
[33mswtch[m () at [32mswtch.S[m:18
18        pushl [31m%[m[32medi[m
(gdb) step
[33mswtch[m () at [32mswtch.S[m:21
21        movl [31m%[m[32mesp[m[31m,[m [31m(%[m[32meax[m[31m)[m
(gdb) step
22        movl [31m%[m[32medx[m[31m,[m [31m%[m[32mesp[m
(gdb) step
[33mswtch[m () at [32mswtch.S[m:25
25        popl [31m%[m[32medi[m
(gdb) step
[33mswtch[m () at [32mswtch.S[m:26
26        popl [31m%[m[32mesi[m
(gdb) step
[33mswtch[m () at [32mswtch.S[m:27
27        popl [31m%[m[32mebx[m
(gdb) step
[33mswtch[m () at [32mswtch.S[m:28
28        popl [31m%[m[32mebp[m
(gdb) step
[33mswtch[m () at [32mswtch.S[m:29
29        [01;34mret[m
(gdb) step
[33mforkret[m () at [32mproc.c[m:401
401       [01mrelease[m[31m(&[mptable[31m.[mlock[31m);[m
(gdb) step
[33mrelease[m ([36mlk[m=0x80112d20 <ptable>) at [32mspinlock.c[m:49
49        [01;34mif[m[31m(![m[01mholding[m[31m([mlk[31m))[m
(gdb) step
[33mholding[m ([36mlock[m=0x80112d20 <ptable>) at [32mspinlock.c[m:92
92        [01;34mreturn[m lock[31m->[mlocked [31m&&[m lock[31m->[mcpu [31m==[m [01mmycpu[m[31m();[m
(gdb) continue
Continuing.

Thread 2 hit Breakpoint 1, [33mswtch[m () at [32mswtch.S[m:11
11        movl [35m4[m[31m(%[m[32mesp[m[31m),[m [31m%[m[32meax[m
(gdb) clear[K[Kar
Deleted breakpoint 1
(gdb) break exec
Breakpoint 2 at [34m0x80100a70[m: file [32mexec.c[m, line 20.
(gdb) continue
Continuing.
[Switching to Thread 1]

Thread 1 hit Breakpoint 2, [33mexec[m ([36mpath[m=0x1c "/init", [36margv[m=0x8dfffed0) at [32mexec.c[m:20
20        [01;34mstruct[m [32mproc[m [31m*[mcurproc [31m=[m [01mmyproc[m[31m();[m
(gdb) continue
Continuing.
[Switching to Thread 2]
```

```
Continuing.
[Switching to Thread 2]

Thread 2 hit Breakpoint 2, [33mexec[m ([36mpath[m=0x836 "sh", [36margv[m=0x8dffeed0) at [32mexec.c[m:20
20        [01;34mstruct[m [32mproc[m [31m*[mcurproc [31m=[m [01mmyproc[m[31m();[m
(gdb) continue
Continuing.
continue
[Switching to Thread 1]

Thread 1 hit Breakpoint 2, [33mexec[m ([36mpath[m=0x18e0 "ls", [36margv[m=0x8dfbeed0) at [32mexec.c[m:20
20        [01;34mstruct[m [32mproc[m [31m*[mcurproc [31m=[m [01mmyproc[m[31m();[m
(gdb) continue
Continuing.
[Switching to Thread 2]

Thread 2 hit Breakpoint 2, [33mexec[m ([36mpath[m=0x18e0 "ls", [36margv[m=0x8df23ed0) at [32mexec.c[m:20
20        [01;34mstruct[m [32mproc[m [31m*[mcurproc [31m=[m [01mmyproc[m[31m();[m
(gdb) q
A debugging session is active.

        Inferior 1 [Remote target] will be detached.

Quit anyway? (y or n) y
Detaching from program: /home/csusb.edu/006013445/cse461/lab1/xv6/kernel, Remote target
Ending remote debugging.
[Inferior 1 (Remote target) detached]
]0;006013445@csusb.edu@ib359-14:~/cse461/lab1/xv6[006013445@csusb.edu@ib359-14 xv6]$ exit

Script done on 2020-04-11 00:04:15-07:00 [COMMAND_EXIT_CODE="0"]
```

3. Debugging of the xv6 continued…

XV6 CP Command

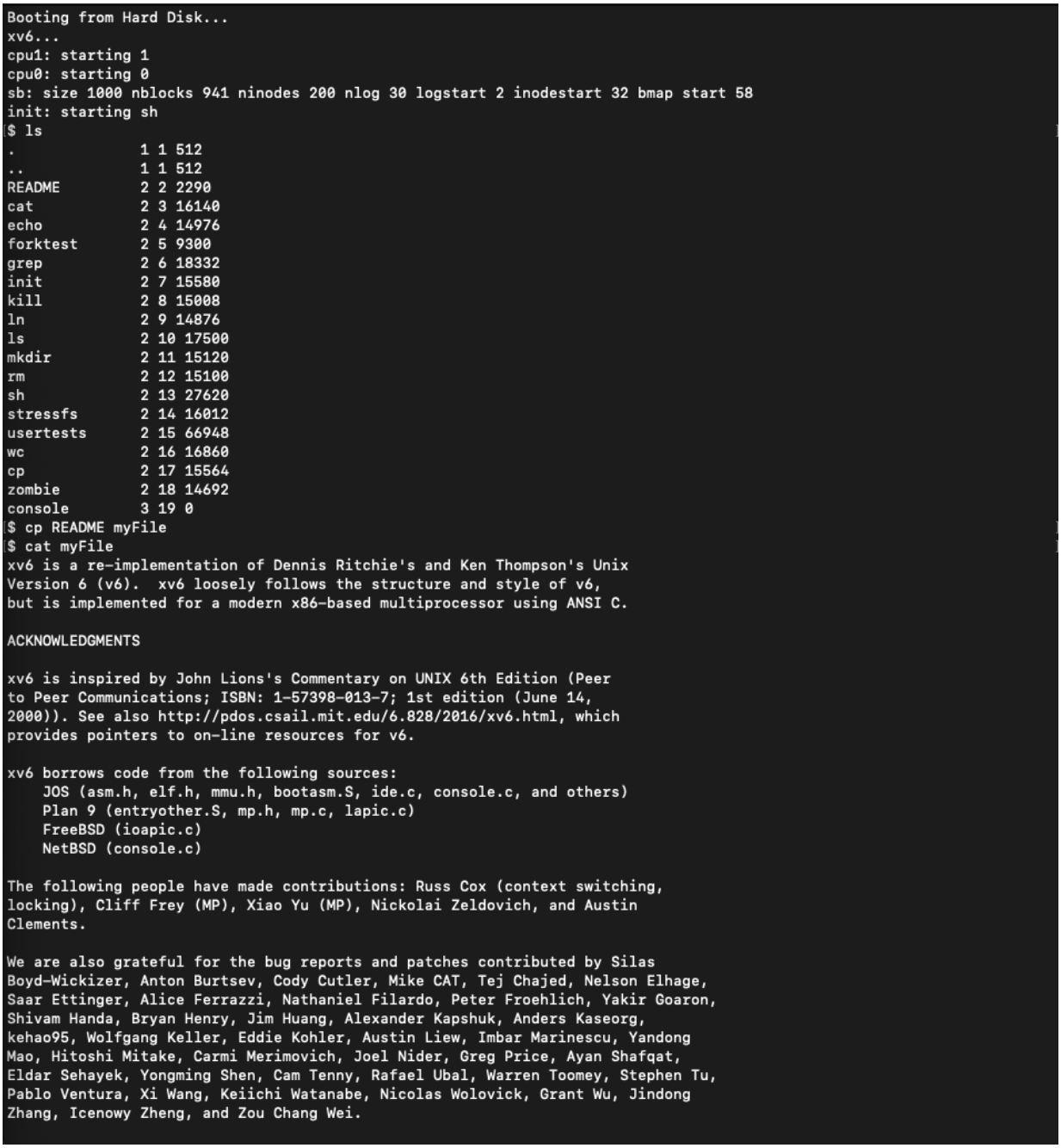
cp.c

```
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"
char buf[512];
int
main(int argc, char *argv[])
{
  int fd0, fd1, n;


  if(argc <= 2){
    printf(1, "Need 2 Arguments!\n");
    exit();
  }
    if((fd0 = open(argv[1], O_RDONLY)) < 0){
      printf(1, "cp: cannot open %s\n", argv[1]);
      exit();
    }
    if ((fd1 = open(argv[2], O_CREATE|O_RDWR)) < 0) {
      printf(1, "cp: cannot open %s\n", argv[2]);
      exit();

    }
```

```
    while (( n = read (fd0, buf, sizeof(buf))) > 0 ){
      write ( fd1, buf, n);
      }
    close(fd0);
    close(fd1);
  exit();
}
```

Makefile

```
EXTRA=\
mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c cp.c zombie.c\
printf.c umalloc.c\
README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
.gdbinit.tmpl gdbutil\

UPROGS=\
    _cat\
    _echo\
    _forktest\
    _grep\
    _init\
    _kill\
    _ln\
    _ls\
    _mkdir\
    _rm\
    _sh\
    _stressfs\
    _usertests\
    _wc\
    _cp\
     _zombie\
```



Conclusion

My partner and I successfully completed each part of the lab but we did encounter some difficulties along the way. We had trouble setting the assembly language to i386 because we could not locate the port number but eventually did. As we completed the lab, we became more familiar with the xv6 and it helped refresh from what we learned about it in CSE 460.