Christian Mesina
Linxuan Liu
CSE 461

Lab 1
Writing a Simple Shell and Study of XV6

1. Writing a Simple Shell

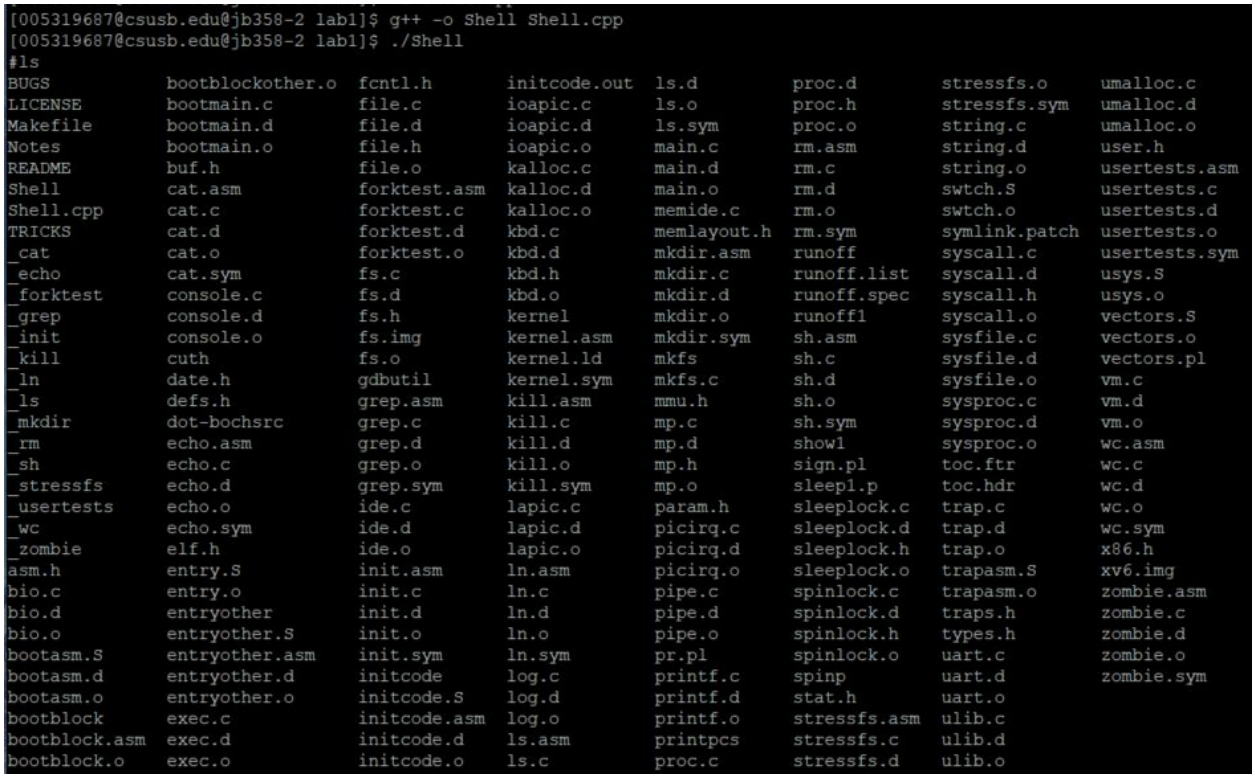
Source Code:

```
// Shell.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");
```

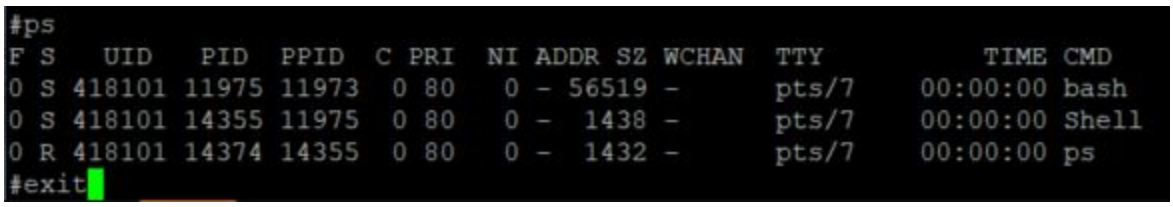
```
    }

    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;
}
```

Outputs:



```
[005319687@csusb.edu@jb358-2 lab1]$ g++ -o Shell Shell.cpp
[005319687@csusb.edu@jb358-2 lab1]$ ./Shell
#ls
BUGS            bootblockother.o  fcntl.h       initcode.out  ls.d          proc.d        stressfs.o     umalloc.c
LICENSE         bootmain.c        file.c        ioapic.c      ls.o          proc.h        stressfs.sym   umalloc.d
Makefile        bootmain.d        file.d        ioapic.d      ls.sym        proc.o        string.c       umalloc.o
Notes           bootmain.o        file.h        ioapic.o      main.c        rm.asm        string.d       user.h
README          buf.h             file.o        kalloc.c      main.d        rm.c          string.o       usertests.asm
Shell           cat.asm           forktest.asm  kalloc.d      main.o        rm.d          swtch.S        usertests.c
Shell.cpp       cat.c             forktest.c    kalloc.o      memide.c      rm.o          swtch.o        usertests.d
TRICKS          cat.d             forktest.d    kbd.c         memlayout.h   rm.sym        symlink.patch  usertests.o
_cat            cat.o             forktest.o    kbd.d         mkdir.asm     runoff        syscall.c      usertests.sym
_echo           cat.sym           fs.c          kbd.h         mkdir.c       runoff.list   syscall.d      usys.S
_forktest       console.c         fs.d          kbd.o         mkdir.d       runoff.spec   syscall.h      usys.o
_grep           console.d         fs.h          kernel        mkdir.o       runoff1       syscall.o      vectors.S
_init           console.o         fs.img        kernel.asm    mkdir.sym     sh.asm        sysfile.c      vectors.o
_kill           cuth              fs.o          kernel.ld     mkfs          sh.c          sysfile.d      vectors.pl
_ln             date.h            gdbutil       kernel.sym    mkfs.c        sh.d          sysfile.o      vm.c
_ls             defs.h            grep.asm      kill.asm      mmu.h         sh.o          sysproc.c      vm.d
_mkdir          dot-bochsrc       grep.c        kill.c        mp.c          sh.sym        sysproc.d      vm.o
_rm             echo.asm          grep.d        kill.d        mp.d          show1         sysproc.o      wc.asm
_sh             echo.c            grep.o        kill.o        mp.h          sign.pl       toc.ftr        wc.c
_stressfs       echo.d            grep.sym      kill.sym      mp.o          sleep1.p      toc.hdr        wc.d
_usertests      echo.o            ide.c         lapic.c       param.h       sleeplock.c   trap.c         wc.o
_wc             echo.sym          ide.d         lapic.d       picirq.c      sleeplock.d   trap.d         wc.sym
_zombie         elf.h             ide.o         lapic.o       picirq.d      sleeplock.h   trap.o         x86.h
asm.h           entry.S           init.asm      ln.asm        picirq.o      sleeplock.o   trapasm.S      xv6.img
bio.c           entry.o           init.c        ln.c          pipe.c        spinlock.c    trapasm.o      zombie.asm
bio.d           entryother        init.d        ln.d          pipe.d        spinlock.d    traps.h        zombie.c
bio.o           entryother.S      init.o        ln.o          pipe.o        spinlock.h    types.h        zombie.d
bootasm.S       entryother.asm    init.sym      ln.sym        pr.pl         spinlock.o    uart.c         zombie.o
bootasm.d       entryother.d      initcode      log.c         printf.c      spinp         uart.d         zombie.sym
bootasm.o       entryother.o      initcode.S    log.d         printf.d      stat.h        uart.o
bootblock       exec.c            initcode.asm  log.o         printf.o      stressfs.asm  ulib.c
bootblock.asm   exec.d            initcode.d    ls.asm        printpcs      stressfs.c    ulib.d
bootblock.o     exec.o            initcode.o    ls.c          proc.c        stressfs.d    ulib.o
```



```
#ps
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S 418101 11975 11973  0 80   0 - 56519 -      pts/7    00:00:00 bash
0 S 418101 14355 11975  0 80   0 -  1438 -      pts/7    00:00:00 Shell
0 R 418101 14374 14355  0 80   0 -  1432 -      pts/7    00:00:00 ps
#exit
```

2. Debugging

a. **GDB Commands and Outputs**:

```
(gdb) target remote:28101
0x0000fff0 in ?? ()
(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...
(gdb) break swtch
Breakpoint 1 at 0x801046eb: file swtch.S, line 11.
(gdb) continue
Continuing.
[Switching to Thread 2]

Thread 2 hit Breakpoint 1, swtch () at swtch.S:11
11      movl 4(%esp), %eax
(gdb) step
12      movl 8(%esp), %edx
(gdb) step
15      pushl %ebp
(gdb) step
swtch () at swtch.S:16
16      pushl %ebx
(gdb) step
swtch () at swtch.S:17
17      pushl %esi
(gdb) step
swtch () at swtch.S:18
18      pushl %edi
(gdb) step
swtch () at swtch.S:21
21      movl %esp, (%eax)
(gdb) step
22      movl %edx, %esp
(gdb) step
swtch () at swtch.S:25
25      popl %edi
```

```
(gdb) step
swtch () at swtch.S:26
26      popl %esi
(gdb) step
swtch () at swtch.S:27
27      popl %ebx
(gdb) step
swtch () at swtch.S:28
28      popl %ebp
(gdb) step
swtch () at swtch.S:29
29      ret
(gdb) step
forkret () at proc.c:401
401      release(&ptable.lock);
(gdb) step
release (lk=0x80112d20 <ptable>) at spinlock.c:49
49      if(!holding(lk))
(gdb) step
holding (lock=0x80112d20 <ptable>) at spinlock.c:92
92      return lock->locked && lock->cpu == mycpu();
(gdb) continue
Continuing.

Thread 2 hit Breakpoint 1, swtch () at swtch.S:11
11      movl 4(%esp), %eax
(gdb) clear
Deleted breakpoint 1
(gdb) break exec
Breakpoint 2 at 0x80100a70: file exec.c, line 20.
(gdb) continue
Continuing.
[Switching to Thread 1]

Thread 1 hit Breakpoint 2, exec (path=0x1c "/init", argv=0x8dfffed0) at exec.c:20
20      struct proc *curproc = myproc();
(gdb) continue
Continuing.
[Switching to Thread 2]
```

```
Thread 2 hit Breakpoint 2, exec (path=0x836 "sh", argv=0x8dffeed0) at exec.c:20
20        struct proc *curproc = myproc();
(gdb) continue
Continuing.

Thread 2 hit Breakpoint 2, exec (path=0x18e0 "ls", argv=0x8dfbeed0) at
exec.c:20
20        struct proc *curproc = myproc();
(gdb) print argv[0]
$1 = 0x18e0 "ls"
(gdb) print argv[1]
$2 = 0x18e3 "-l"
(gdb) print argv[2]
$3 = 0x0
(gdb) backtrace
#0  exec (path=0x18e0 "ls", argv=0x8dfbeed0) at exec.c:20
#1  0x801053c0 in sys_exec () at sysfile.c:420
#2  0x801048a9 in syscall () at syscall.c:139
#3  0x80105911 in trap (tf=0x8dfbefb4) at trap.c:43
#4  0x8010565f in alltraps () at trapasm.S:20
#5  0x8dfbefb4 in ?? ()
Backtrace stopped: previous frame inner to this frame (corrupt stack?)
(gdb) up
#1  0x801053c0 in sys_exec () at sysfile.c:420
420       return exec(path, argv);
(gdb) list
415           break;
416         }
417         if(fetchstr(uarg, &argv[i]) < 0)
418           return -1;
419       }
420       return exec(path, argv);
421     }
422
423     int
424     sys_pipe(void)
```

- Based on the debugging process that we did, we observed that you need to have two windows/sessions of the terminal to be opened so that the qemu will be able to run along with the gdb. For this to happen, we had to look up the tcp number after running the qemu nox gdb. This is essential when running the gdb because it won't let you access the commands and files of the gdb. After setting everything up, we now have to learn how to use the debugging commands and implement it into a sample code that was given to us. Overall, the debugging process was straightforward and we figured everything out correctly by following the step-by-step process that was given to us.

b. **Outputs**:

```
0x0000fff0 in ?? ()
(gdb) continue
Continuing.
^C
Thread 1 received signal SIGINT, Interrupt.
0x801037e1 in ?? ()
(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...
(gdb) set disassembly-flavor intel
(gdb) disass
Dump of assembler code for function mycpu:
   0x801037d0 <+0>:     push   ebp
   0x801037d1 <+1>:     mov    ebp,esp
   0x801037d3 <+3>:     push   esi
   0x801037d4 <+4>:     push   ebx
   0x801037d5 <+5>:     pushf
   0x801037d6 <+6>:     pop    eax
   0x801037d7 <+7>:     test   ah,0x2
   0x801037da <+10>:    jne    0x8010382a <mycpu+90>
   0x801037dc <+12>:    call   0x80102800 <lapicid>
=> 0x801037e1 <+17>:    mov    esi,DWORD PTR ds:0x80112d00
   0x801037e7 <+23>:    mov    ebx,eax
   0x801037e9 <+25>:    test   esi,esi
   0x801037eb <+27>:    jle    0x8010381d <mycpu+77>
   0x801037ed <+29>:    xor    edx,edx
   0x801037ef <+31>:    jmp    0x801037ff <mycpu+47>
```

```
  0x801037f1 <+33>:    lea    esi,[esi+eiz*1+0x0]
  0x801037f8 <+40>:    add    edx,0x1
  0x801037fb <+43>:    cmp    edx,esi
  0x801037fd <+45>:    je     0x8010381d <mycpu+77>
  0x801037ff <+47>:    imul   ecx,edx,0xb0
  0x80103805 <+53>:    movzx  eax,BYTE PTR [ecx-0x7feed880]
  0x8010380c <+60>:    cmp    eax,ebx
  0x8010380e <+62>:    jne    0x801037f8 <mycpu+40>
  0x80103810 <+64>:    lea    esp,[ebp-0x8]
  0x80103813 <+67>:    lea    eax,[ecx-0x7feed880]
  0x80103819 <+73>:    pop    ebx
  0x8010381a <+74>:    pop    esi
  0x8010381b <+75>:    pop    ebp
  0x8010381c <+76>:    ret
  0x8010381d <+77>:    sub    esp,0xc
  0x80103820 <+80>:    push   0x801073c7
  0x80103825 <+85>:    call   0x80100380 <panic>
  0x8010382a <+90>:    sub    esp,0xc
  0x8010382d <+93>:    push   0x801074a4
  0x80103832 <+98>:    call   0x80100380 <panic>
End of assembler dump.
```

c. **Source Code**:

```
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"

char buf[512];

int
main(int argc, char *argv[])
{
 int fd0, fd1, fd2, n;


 if(argc <= 3){
  printf(1, "Need 3 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();
  }

  if((fd2 = open(argv[3], O_CREATE|O_RDWR)) < 0){
    printf(1, "cp: cannot open %s\n", argv[3]);
    exit();
  }

  while((n = read(fd0, buf, sizeof(buf))) > 0)
  {
    write(fd1, buf, n);
    write(fd2, buf, n);
  }

  close(fd0);
  close(fd1);
  close(fd2);
  exit();
}
```

Script Outputs:

```
Script started on 2020-04-14 11:48:25-07:00 [TERM="xterm" TTY="/dev/pts/0"
COLUMNS="93" LINES
="49"]
^[]0;006151141@csusb.edu@jb358-9:~/CSE461/lab1^G[006151141@csusb.edu
@jb358-9 lab1]$ make^M
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32
-Werror -fno-
omit-frame-pointer -fno-stack-protector   -c -o cp.o cp.c^M
```

```
ld -m    elf_i386 -N -e main -Ttext 0 -o _cp cp.o ulib.o usys.o printf.o
umalloc.o^M
objdump -S _cp > cp.asm^M
objdump -t _cp | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > cp.sym^M
./mkfs fs.img README _cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm
_sh _stressfs
_usertests _wc _cp _zombie ^M
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks
941 total 1000^
M
balloc: first 694 blocks have been allocated^M
balloc: write bitmap block at sector 58^M
dd if=/dev/zero of=xv6.img count=10000^M
10000+0 records in^M
10000+0 records out^M
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.389234 s, 13.2 MB/s^M
dd if=bootblock of=xv6.img conv=notrunc^M
1+0 records in^M
1+0 records out^M
512 bytes copied, 0.00198926 s, 257 kB/s^M
dd if=kernel of=xv6.img seek=1 conv=notrunc^M
416+1 records in^M
416+1 records out^M
213104 bytes (213 kB, 208 KiB) copied, 0.00706842 s, 30.1 MB/s^M
^[]0;006151141@csusb.edu@jb358-9:~/CSE461/lab1^G[006151141@csusb.edu
@jb358-9 lab1]$ make qemu
-nox^M
which: no qemu in
(/opt/anaconda3/bin:/opt/Xilinx/SDK/2018.2/bin:/opt/Xilinx/SDK/2018.2/gnu/m
icroblaze/lin/bin:/opt/Xilinx/SDK/2018.2/gnu/arm/lin/bin:/opt/Xilinx/SDK/2018.2
/gnu/microblaz
e/linux_toolchain/lin64_le/bin:/opt/Xilinx/SDK/2018.2/gnu/aarch32/lin/gcc-arm-l
inux-gnueabi/b
in:/opt/Xilinx/SDK/2018.2/gnu/aarch32/lin/gcc-arm-none-eabi/bin:/opt/Xilinx/SD
K/2018.2/gnu/aa
rch64/lin/aarch64-linux/bin:/opt/Xilinx/SDK/2018.2/gnu/aarch64/lin/aarch64-non
e/bin:/opt/Xili
nx/SDK/2018.2/gnu/armr5/lin/gcc-arm-none-eabi/bin:/opt/Xilinx/SDK/2018.2/tps
/lnx64/cmake-3.3.
```

2/bin:/opt/Xilinx/DocNav:/opt/Xilinx/Vivado/2018.2/bin:/usr/lib64/openmpi/bin:/opt/UCSF/Chime
ra64-1.12/bin:/usr/local/MATLAB/R2018a/bin:/share/bin:/usr/local/racket/bin:/opt/Xilinx/14.7/
ISE_DS/ISE/bin/lin64:/opt/Xilinx/14.7/ISE_DS/common/bin/lin64:/opt/Xilinx/Vivado/2017.2/bin:/
opt/Xilinx/Vivado_HLS/2017.2/bin:/opt/android-studio/bin:/opt/android-sdk-linux/tools:/opt/an
droid-sdk-linux/platform-tools:/usr/java/latest/bin:/u01/app/oracle/product/12.2.0/dbhome_1/b
in:/opt/anaconda3/bin:/opt/Xilinx/SDK/2018.2/bin:/opt/Xilinx/SDK/2018.2/gnu/microblaze/lin/bi
n:/opt/Xilinx/SDK/2018.2/gnu/arm/lin/bin:/opt/Xilinx/SDK/2018.2/gnu/microblaze/linux_toolchai
n/lin64_le/bin:/opt/Xilinx/SDK/2018.2/gnu/aarch32/lin/gcc-arm-linux-gnueabi/bin:/opt/Xilinx/S
DK/2018.2/gnu/aarch32/lin/gcc-arm-none-eabi/bin:/opt/Xilinx/SDK/2018.2/gnu/aarch64/lin/aarch6
4-linux/bin:/opt/Xilinx/SDK/2018.2/gnu/aarch64/lin/aarch64-none/bin:/opt/Xilinx/SDK/2018.2/gn
u/armr5/lin/gcc-arm-none-eabi/bin:/opt/Xilinx/SDK/2018.2/tps/lnx64/cmake-3.3.2/bin:/opt/Xilin
x/DocNav:/opt/Xilinx/Vivado/2018.2/bin:/usr/lib64/openmpi/bin:/opt/UCSF/Chimera64-1.12/bin:/u
sr/local/MATLAB/R2018a/bin:/share/bin:/usr/local/racket/bin:/opt/Xilinx/14.7/ISE_DS/ISE/bin/l
in64:/opt/Xilinx/14.7/ISE_DS/common/bin/lin64:/opt/Xilinx/Vivado/2017.2/bin:/opt/Xilinx/Vivad
o_HLS/2017.2/bin:/opt/android-studio/bin:/opt/android-sdk-linux/tools:/opt/android-sdk-linux/
platform-tools:/usr/java/latest/bin:/opt/anaconda3/bin:/opt/Xilinx/SDK/2018.2/bin:/opt/Xilinx
/SDK/2018.2/gnu/microblaze/lin/bin:/opt/Xilinx/SDK/2018.2/gnu/arm/lin/bin:/opt/Xilinx/SDK/201
8.2/gnu/microblaze/linux_toolchain/lin64_le/bin:/opt/Xilinx/SDK/2018.2/gnu/aarch32/lin/gcc-ar
m-linux-gnueabi/bin:/opt/Xilinx/SDK/2018.2/gnu/aarch32/lin/gcc-arm-none-eabi/bin:/opt/Xilinx/

```
SDK/2018.2/gnu/aarch64/lin/aarch64-linux/bin:/opt/Xilinx/SDK/2018.2/gnu/aarc
h64/lin/aarch64-n
one/bin:/opt/Xilinx/SDK/2018.2/gnu/armr5/lin/gcc-arm-none-eabi/bin:/opt/Xilinx
/SDK/2018.2/tps
/lnx64/cmake-3.3.2/bin:/opt/Xilinx/DocNav:/opt/Xilinx/Vivado/2018.2/bin:/usr/li
b64/openmpi/bi
n:/opt/UCSF/Chimera64-1.12/bin:/usr/local/MATLAB/R2018a/bin:/share/bin:/usr
/local/racket/bin:
/opt/Xilinx/14.7/ISE_DS/ISE/bin/lin64:/opt/Xilinx/14.7/ISE_DS/common/bin/lin
64:/opt/Xilinx/Vi
vado/2017.2/bin:/opt/Xilinx/Vivado_HLS/2017.2/bin:/opt/android-studio/bin:/opt/
android-sdk-li
nux/tools:/opt/android-sdk-linux/platform-tools:/usr/java/latest/bin:/usr/lib64/qt-3.
3/bin:/u
sr/share/Modules/bin:/usr/lib64/ccache:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/s
bin:/hom
e/csusb.edu/006151141/.local/bin:/home/csusb.edu/006151141/bin:.:/home/csusb.
edu/006151141/bi
n)^M
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw
-drive file=xv6.
img,index=0,media=disk,format=raw -smp 2 -m 512 ^M
^[c^[[?7l^[[2J^[[0mSeaBIOS (version 1.12.0-2.fc30)^M^M
^M
^M
iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF91280+1FED1280
C980^M^M
Press Ctrl-B to configure iPXE (PCI 00:03.0)...^M
                                ^M^M
^M
^M
Booting from Hard Disk..xv6...^M
cpu1: starting 1^M
cpu0: starting 0^M
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start
58^M
init: starting sh^M
$ ls^M
.               1 1 512^M
```

```
..              1 1 512^M
README          2 2 2290^M
cat             2 3 16136^M
echo            2 4 14972^M
forktest        2 5 9296^M
grep            2 6 18328^M
init            2 7 15576^M
kill            2 8 15004^M
ln              2 9 14872^M
ls              2 10 17496^M
mkdir           2 11 15116^M
rm              2 12 15096^M
sh              2 13 27616^M
stressfs        2 14 16008^M
usertests       2 15 66944^M
wc              2 16 16856^M
cp              2 17 15836^M
zombie          2 18 14688^M
console         3 19 0^M
$ cp README^M
Need 3 arguments!^M
$ cp RAD^H ^H^H ^HEADME myFile1 myFilw^H ^He2^M
$ cat myFile1^M
xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix^M
Version 6 (v6).  xv6 loosely follows the structure and style of v6,^M
but is implemented for a modern x86-based multiprocessor using ANSI C.^M
^M
ACKNOWLEDGMENTS^M
^M
xv6 is inspired by John Lions's Commentary on UNIX 6th Edition (Peer^M
to Peer Communications; ISBN: 1-57398-013-7; 1st edition (June 14,^M
2000)). See also http://pdos.csail.mit.edu/6.828/2016/xv6.html, which^M
provides pointers to on-line resources for v6.^M
^M
xv6 borrows code from the following sources:^M
    JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)^M
    Plan 9 (entryother.S, mp.h, mp.c, lapic.c)^M
    FreeBSD (ioapic.c)^M
    NetBSD (console.c)^M
```

^M
The following people have made contributions: Russ Cox (context switching,^M
locking), Cliff Frey (MP), Xiao Yu (MP), Nickolai Zeldovich, and Austin^M
Clements.^M
^M
We are also grateful for the bug reports and patches contributed by Silas^M
Boyd-Wickizer, Anton Burtsev, Cody Cutler, Mike CAT, Tej Chajed, Nelson
Elhage,^M
Saar Ettinger, Alice Ferrazzi, Nathaniel Filardo, Peter Froehlich, Yakir
Goaron,^M
Shivam Handa, Bryan Henry, Jim Huang, Alexander Kapshuk, Anders
Kaseorg,^M
kehao95, Wolfgang Keller, Eddie Kohler, Austin Liew, Imbar Marinescu,
Yandong^M
Mao, Hitoshi Mitake, Carmi Merimovich, Joel Nider, Greg Price, Ayan
Shafqat,^M
Eldar Sehayek, Yongming Shen, Cam Tenny, Rafael Ubal, Warren Toomey,
Stephen Tu,^M
Pablo Ventura, Xi Wang, Keiichi Watanabe, Nicolas Wolovick, Grant Wu,
Jindong^M
Zhang, Icenowy Zheng, and Zou Chang Wei.^M
^M
The code in the files that constitute xv6 is^M
Copyright 2006-2016 Frans Kaashoek, Robert Morris, and Russ Cox.^M
^M
ERROR REPORTS^M
^M
Please send errors and suggestions to Frans Kaashoek and Robert Morris^M
(kaashoek,rtm@mit.edu). The main purpose of xv6 is as a teaching^M
operating system for MIT's 6.828, so we are more interested in^M
simplifications and clarifications than new features.^M
^M
BUILDING AND RUNNING XV6^M
^M
To build xv6 on an x86 ELF machine (like Linux or FreeBSD), run^M
"make". On non-x86 or non-ELF machines (like OS X, even on x86), you^M
will need to install a cross-compiler gcc suite capable of producing^M
x86 ELF binaries. See http://pdos.csail.mit.edu/6.828/2016/tools.html.^M
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC^M

simulator and run "make qemu".^M
$ cat myFile2^M
xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix^M
Version 6 (v6). xv6 loosely follows the structure and style of v6,^M
but is implemented for a modern x86-based multiprocessor using ANSI C.^M
^M
ACKNOWLEDGMENTS^M
^M
xv6 is inspired by John Lions's Commentary on UNIX 6th Edition (Peer^M
to Peer Communications; ISBN: 1-57398-013-7; 1st edition (June 14,^M
2000)). See also http://pdos.csail.mit.edu/6.828/2016/xv6.html, which^M
provides pointers to on-line resources for v6.^M
^M
xv6 borrows code from the following sources:^M
    JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)^M
    Plan 9 (entryother.S, mp.h, mp.c, lapic.c)^M
    FreeBSD (ioapic.c)^M
    NetBSD (console.c)^M
^M
The following people have made contributions: Russ Cox (context switching,^M
locking), Cliff Frey (MP), Xiao Yu (MP), Nickolai Zeldovich, and Austin^M
Clements.^M
^M
We are also grateful for the bug reports and patches contributed by Silas^M
Boyd-Wickizer, Anton Burtsev, Cody Cutler, Mike CAT, Tej Chajed, Nelson
Elhage,^M
Saar Ettinger, Alice Ferrazzi, Nathaniel Filardo, Peter Froehlich, Yakir
Goaron,^M
Shivam Handa, Bryan Henry, Jim Huang, Alexander Kapshuk, Anders
Kaseorg,^M
kehao95, Wolfgang Keller, Eddie Kohler, Austin Liew, Imbar Marinescu,
Yandong^M
Mao, Hitoshi Mitake, Carmi Merimovich, Joel Nider, Greg Price, Ayan
Shafqat,^M
Eldar Sehayek, Yongming Shen, Cam Tenny, Rafael Ubal, Warren Toomey,
Stephen Tu,^M
Pablo Ventura, Xi Wang, Keiichi Watanabe, Nicolas Wolovick, Grant Wu,
Jindong^M
Zhang, Icenowy Zheng, and Zou Chang Wei.^M

^M
The code in the files that constitute xv6 is^M
Copyright 2006-2016 Frans Kaashoek, Robert Morris, and Russ Cox.^M
^M
ERROR REPORTS^M
^M
Please send errors and suggestions to Frans Kaashoek and Robert Morris^M
(kaashoek,rtm@mit.edu). The main purpose of xv6 is as a teaching^M
operating system for MIT's 6.828, so we are more interested in^M
simplifications and clarifications than new features.^M
^M
BUILDING AND RUNNING XV6^M
^M
To build xv6 on an x86 ELF machine (like Linux or FreeBSD), run^M
"make". On non-x86 or non-ELF machines (like OS X, even on x86), you^M
will need to install a cross-compiler gcc suite capable of producing^M
x86 ELF binaries. See http://pdos.csail.mit.edu/6.828/2016/tools.html.^M
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC^M
simulator and run "make qemu".^M
$ ls^M
. 1 1 512^M
.. 1 1 512^M
README 2 2 2290^M
cat 2 3 16136^M
echo 2 4 14972^M
forktest 2 5 9296^M
grep 2 6 18328^M
init 2 7 15576^M
kill 2 8 15004^M
ln 2 9 14872^M
ls 2 10 17496^M
mkdir 2 11 15116^M
rm 2 12 15096^M
sh 2 13 27616^M
stressfs 2 14 16008^M
usertests 2 15 66944^M
wc 2 16 16856^M
cp 2 17 15836^M
zombie 2 18 14688^M

```
console        3 19 0^M
myFile1        2 20 2290^M
myFile2        2 21 2290^M
$ QEMU 3.1.1 monitor - type 'help' for more information^M^M
(qemu) q^[[K^[[Dqu^[[K^[[D^[[Dqui^[[K^[[D^[[D^[[Dquit^[[K^M^M
^[]0;006151141@csusb.edu@jb358-9:~/CSE461/lab1^G[006151141@csusb.edu
@jb358-9 lab1]$ exit^M
exit^M

Script done on 2020-04-14 11:49:52-07:00 [COMMAND_EXIT_CODE="0"]
```

3. Report

While doing this lab, there are many new things we learnt and some difficulties we've encountered. In the first part, writing a simple shell, we follow the instruction and implement the code successfully and it works. The simple shell creates the new child in xv6 successfully and it can be shown on ps command. There aren't any difficulties we encountered there. Coming to the second part of this lab, the debugging process in the xv6, we learnt how to use debugging functions in xv6 and how to use some commands such as break, step, continue, clear, etc. We also disassembled the kernel in i386, and examined its code successfully. By following the instructions in the lab, we have finished the debugging part successfully. While during the gdb, we found there are several things we need to rethink about. The meaning of some parts in debugging are confused but we figured that out after trying some commands and searched on websites. The last part is to write the cp command in xv6. After the first time we complete the cp.c file, we found that the cp command only works for 2 arguments while the lab asks us to make it work under 3 arguments. We changed part of the code to make it work with 3 arguments and it can work finally. Furthermore, we have the ability to implement the command in xv6 and do some similar things like cp command. Overall, Lab 1 taught us how to write shell, debug xv6, and implement new commands in xv6. Therefore, we believe that we should get the full 20 points because we finished each part successfully.