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;

}



A close up of a screen

Description automatically generated

1. Basics of XV6A screenshot of a cell phone

   Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of text

Description automatically generated

2. Basics of XV6 Continued(gdb and disassemble)

A screenshot of a cell phone

Description automatically generated

1. Debugging

Debugging of xv6 following Part 3 of the lab.

A close up of text on a black background

Description automatically generated

A screenshot of a cell phone

Description automatically generated

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\

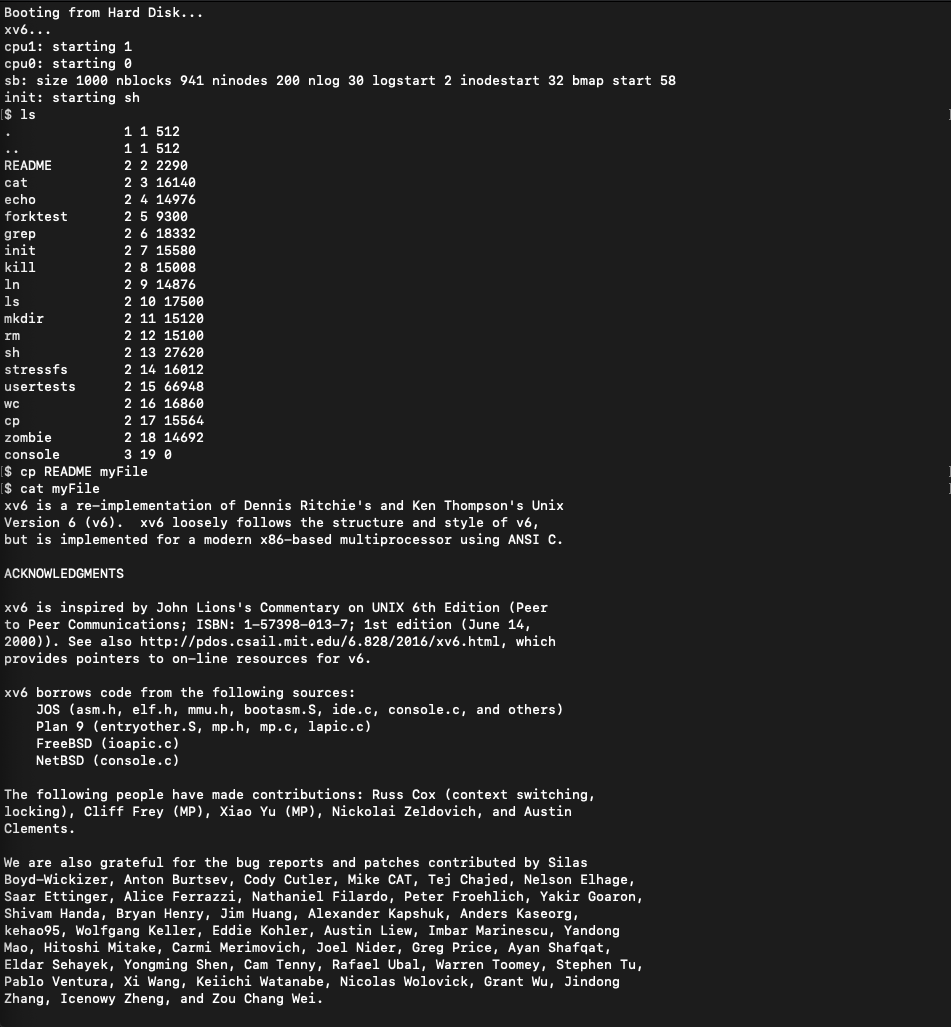
\_stressfs\

\_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.