Name: Yuxiang Chen

SID: 904224400

Subject: Advanced Operating Systems

22 Apr. 2022

COMP7500 Project 4
pWordCount: A Pipe-based WordCount Tool

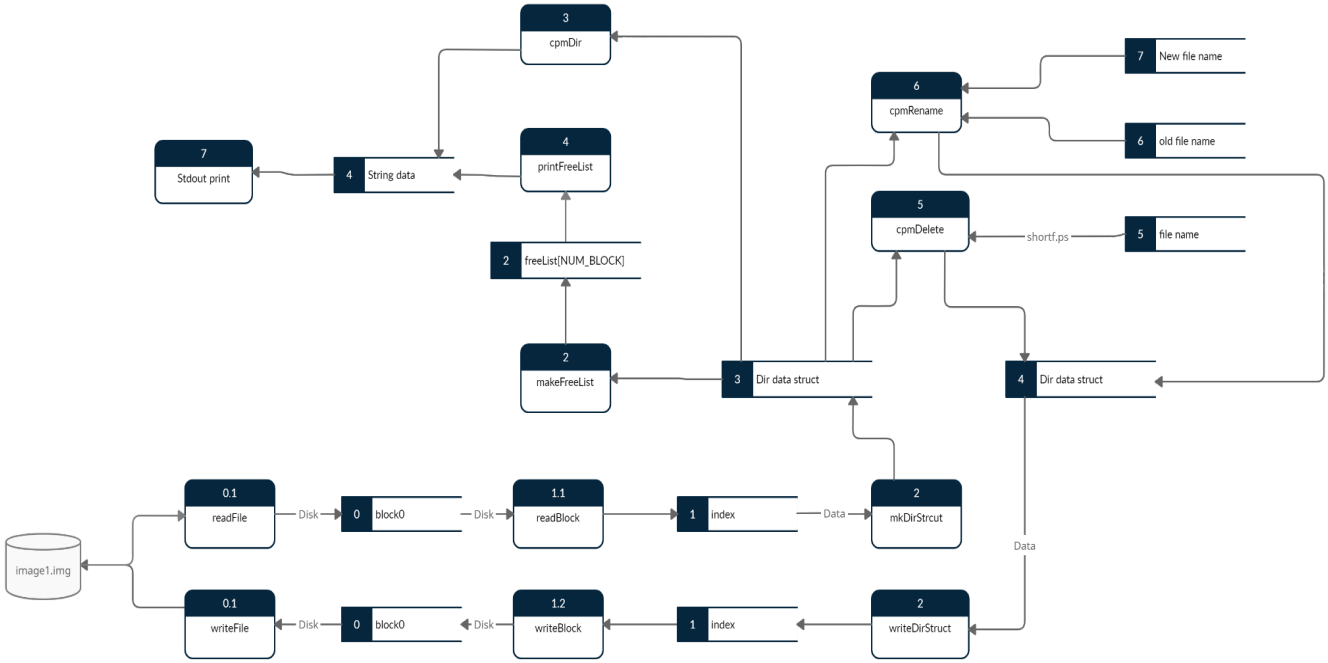
TASK OVERVIEW

The goal of this project is to design and implement a simple file system called cpmFS (i.e., CP/M file sytem). Through the late 1970s and into the mid-1980s, CP/M (Control Program for Microcomputers) – a disk-based operating system – had dominated its era as much as MS-DOS and later Windows dominated the IBM PC world. CP/M is clearly not the last word in advanced file systems, but it is simple, fast, and can be implemented by a competent programmer in less than a week.

Your simple file system allows users to list directory entries, rename files, copy files, delete files, as well as code to read/write/open/close files. We will use a version of the CP/M file system used on 5.25” and 8” floppy disks in the 1970’s (support for CP/M file systems is still included in Linux to this day). You will develop your code in C. You may use any computer with an ANSI C compiler (e,g. gcc, clang, etc.). You will not be modifying the linux kernel but developing a stand-alone program (i.e., a simulated file-system).

DESIGN

Data Flow Diagram



The architecture of the program

(1) User–input handling;

(2) Disk-Buffer-Memory;

(3) Data Structure;

(4) Filename String handling;

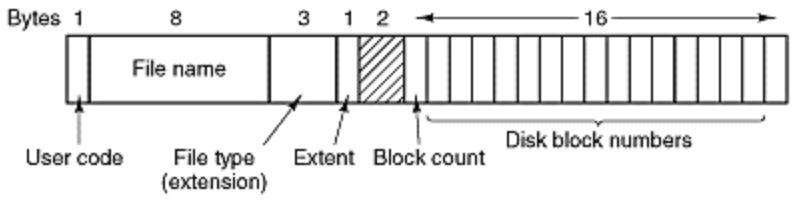
DATA STRUCTURE

Function prototype:

```
//function to allocate memory for a DirStructType and populate it
```

```
//DirStructType value to return.

DirStructType *mkDirStruct(int index,uint8_t *e);

void writeDirStruct(DirStructType *d, uint8_t index, uint8_t *e);

void makeFreeList();

void printFreeList();

int findExtentWithName(char *name, uint8_t *block0);

bool checkLegalName(char *name);

void cpmDir();

int cpmRename(char *oldName, char * newName);

int  cpmDelete(char * name);
```

DATA STRUCTURE



EXECUTION

Makefile:

```
cpmRun: diskSimulator.o  cpmfsys.o fsysdriver.o
    gcc -o cpmRun diskSimulator.o cpmfsys.o fsysdriver.o

diskSimulator.o: diskSimulator.c diskSimulator.h
    gcc -c diskSimulator.c
```

```
cpmfsys.o: cpmfsys.h cpmfsys.c
    gcc -c cpmfsys.c

fsysdriver.o: fsysdriver.c
    gcc -c fsysdriver.c

all:
    cpmRun

clean:
    rm *.o
```

Usage:

1. *Make*
2. */cpmRun*

After execution:

3. *make clean*

RESULT & TEST

Since the main() function is already defined by the package, this part of the test gives the same result as sampleOutput.txt

```
DIRECTORY LISTING
mytestf1.txt 15874
holefile.txt 1152
shortf.ps 1032
mytestf. 1026
FREE BLOCK LIST: (* means in-use)
  0: * . * . * . * . * . * . * . * .
 10: * . . . . . . . . . . . . . . .
 20: * . . . * . . . . . . . . . . .
 30: * * . . . . . . . . . . . . . .
 40: . . . . . . . . . . . . . . . .
 50: . . . . . . . . . . . . . . . .
 60: . . . . . . . . . . . . . . . .
 70: . . . . . . . . . . . . . . . .
 80: . . . . . . . . . . . . . . . .
 90: . . . . . . . . . . . . . . . .
 a0: * . . . * . . . . . . . . . . .
 b0: . . . . . . . . . . . . . . . .
 c0: . . . . . . . . . . . . . . . .
 d0: . . . . . . . . . . . . . . . .
 e0: . . . . . . . . . . . . . . . .
 f0: . * . * . * . * . * . * . * . *
DIRECTORY LISTING
mytestf1.txt 15874
holefile.txt 1152
mytestf. 1026
cpmRename return code = 0,
DIRECTORY LISTING
mytest2.tx 15874
holefile.txt 1152
mytestv2.x 1026
FREE BLOCK LIST: (* means in-use)
```

```
 0: * . * . * . * . * . * . * . * .
10: * . . . . . . . . . . . . . . .
20: * . . . * . . . . . . . . . . .
30: . . . . . . . . . . . . . . . .
40: . . . . . . . . . . . . . . . .
50: . . . . . . . . . . . . . . . .
60: . . . . . . . . . . . . . . . .
70: . . . . . . . . . . . . . . . .
80: . . . . . . . . . . . . . . . .
90: . . . . . . . . . . . . . . . .
a0: * . . . * . . . . . . . . . . .
b0: . . . . . . . . . . . . . . . .
c0: . . . . . . . . . . . . . . . .
d0: . . . . . . . . . . . . . . . .
e0: . . . . . . . . . . . . . . . .
f0: . * . * . * . * . * . * . * . *
```

Result: Successful!

CONCLUSION

I will successfully complete Assignment Project 3 cpm-A simple file system. During this assignment, I used the technique. I have more understanding about the string library <string.h>, I tried to use **strtok()** for splitting a string to get a filename and its extension. **strcpy()** for copying a string to another, sometimes when we are trying to edit a string, like strtok(), we need to store the string from user input before we edit it. strcpy() will help. **strcmp()** to check whether a file is existing in the file system.

For operating system knowledge, in this project, I am familiar with the cpm file system. And file system level. Disk drive- Disk simulator - File system.