CSC-415 Operating Systems

File System Project

Group Submission

Team:

Diligence

GitHub:
https://github.com/CSC415-2022-Fall/csc415-filesystem-mkim797

Name	Student ID
Mark Kim	918204214
Peter Truong	915780793
Zeel Diyora	920838201
Chengkai Yang	921572896

Table of Contents

Description

This project is a file system using FAT for free space and volume management. The portion of code that our team completed is split into 5 sections (not including the fsshell.c. file and mfs header). These sections are fsInit, fsFree, fsDir, fsHelpers, and b_io, which handle file system initialization, free space management, directory management, misc helper functions, and buffered I/O for reading and writing to the file system.

Since we were using FAT to track the free space and the volume, our free space map occupies much more space than a bitmap would have. However, the use of FAT allowed us to dynamically manage non-contiguous allocation, reallocation, and deallocation of free space as we read and write files and directories. Since each block of the free space map only knew its own location and the next location (a linked list), handling the free space only required us to iterate through each file/directory's maps and the free space map and link up the ends to the beginnings as those requests are made. One disadvantage of this, however, is that one cannot jump directly to a particular block of the file system and to reach any block, we have to iterate through the entire list until we reach the block that we are looking for. From a design perspective, it simplifies some of the code, but also required us to think about how one would traverse a file by the block.

fsInit.c

This portion of the program simply initializes the file system by creating/loading the volume control block, initializing the free space, and setting the current working directory array and path. The initialization is fairly short as the bulk of the work is contained in the free space management (fsFree.c). This file also contains the logic for freeing any malloc'd memory.

fsFree.c

This section manages the free space map. We have an init_free() function that initializes the free space map by filling the FAT with the references to the next block and flagging the final block to 0xFFFFFFFE. All the other functions manage the allocation of free space or retrieving the location of a block on the volume.

fsDir.c

This file contains the functions that manage directories and directory entries. The init_dir() function initializes directories. parsePath() is likely one of the most important functions of the file system; many functions depend on this function. It takes a path string, tokenizes it, then navigates through the directory tree using those tokens. If it encounters an invalid entry, it simply returns an error. Otherwise, it returns the directory array associated with the final entry of the path (but does nothing with the final entry). It is the caller's responsibility to do something with the directory array returned. In addition to the functions mentioned above, this file also includes many other functions meant to validate, navigate or mutate directories.

FsHelpers.c

Any helper functions that are not really associated with free space or directory management are contained here. We have functions that do such things as setting the current working path, retrieving the final token in a path string, calculating the number of blocks a byte count would occupy, and writing to disk.

Issues Encountered

This file system was certainly difficult. Although we put in a countless amount of time planning and trying to figure out the structure of the system, the plans were often missing small details that would have giant unforeseen consequences which would require us to either refactor large portions of code, or just complete scrap entire sections.

One such small detail was how we would manage the path for the current working directory. We initially approached it by trying to mutate the path string as directory changes occurred. After many hours of frustration and failure on this, we ended up just scrapping all our work on it in favor of just navigating the entire directory tree backwards down to the root directory and constructing the path that way for every change.

We ran into many issues regarding jumbled data results from our b_write and b_read functions. This occurred because it was not readily apparent how to separate the cause of write versus read. We initially just copied files over to the file system then copied them back to Linux, but we weren't sure if it was being caused by write, read, or both. This took a lot of work to pour over the hexdump, which was extremely time-consuming and tedious. It wasn't until we realized that we could just read the SampleVolume sans hexdump to compare what was written on the file system with the payload. This was a massive time-saver and would've been a useful bit of knowledge that wasn't readily apparent.

On a similar note, we got the file system working with a buffer size of 200, but once we started switching to buffer sizes greater than the block size, we once again ran into issues. These issues were caused by "part2" of the read and write. At first, we could not figure out what was causing it, but it should've been obvious to us that because we were using FAT, we could not simply write consecutive blocks in one big chunk. We switched to writing one block at a time and iterating through the free space incrementally. I am sure we could implement optimized code that would be able to write contiguous blocks, but since this method works, we simply said it is "good enough."

Another point to note is that we are aware of memory leaks in the system. Although we could hunt those memory leaks down, we again decided that what we have is “good enough.”

Driver Program Operation

The driver program fsshell.c is largely untouched from the original provided. We implemented randomization for the buffer size just to rigorously test our file system. We also implemented the cmd_mv() function, but the actual logic of move is contained in the fsDir.c file. Finally, we added a fair number of extra functions to the shell to help us debug the file system. Those extra functions mostly call lower level functions and print out some result from the function call. This helped us tremendously as it provided us with really important data on what was occurring in our file system as we ran specific functions. These test functions include:

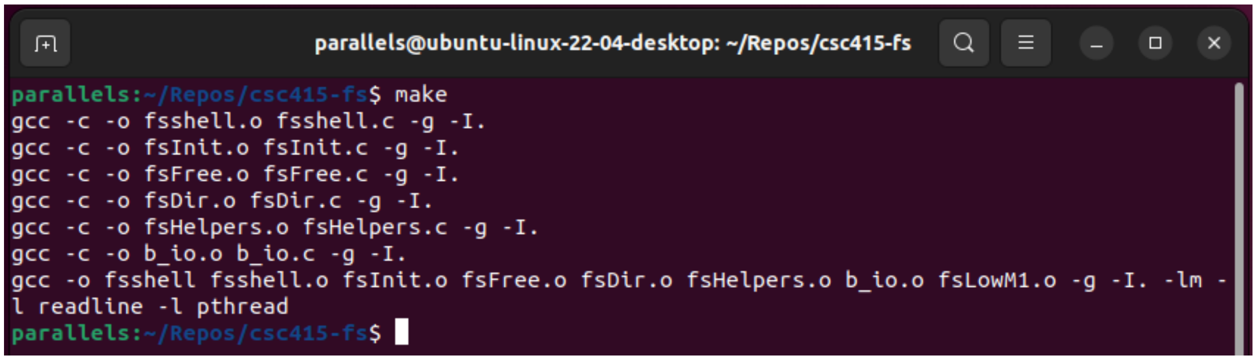
1. pp: parsePath() – print out the entire directory that parsePath returns
2. isfile: fs_isFile()
3. isdir: fs_isDir()
4. opendir: fs_opendir()
5. getcwd: fs_getcwd()
6. readdir: fs_readdir()
7. getde: print directory entry to console

Screenshots

Commands shown in screenshots

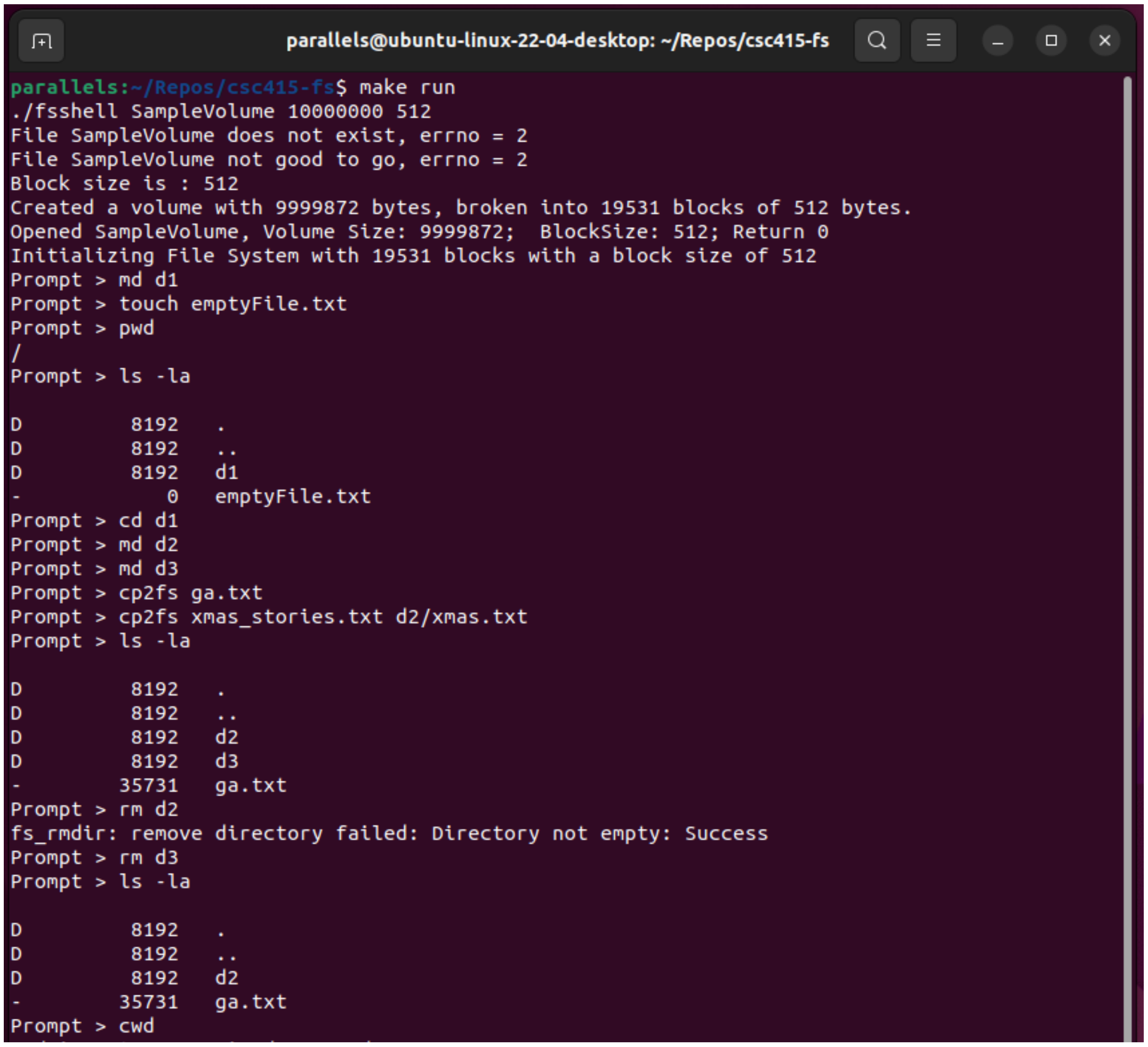
```
ls – Lists the file in a directory
cp – Copies a file – source [dest]
mv – Moves a file – source dest
md – Make a new directory
rm – Removes a file or directory
touch – creates a file
cat – (limited functionality) displays the contents of a file
cp2l – Copies a file from the test file system to the linux file system
cp2fs – Copies a file from the Linux file system to the test file system
cd – Changes directory
pwd – Prints the working directory
history – Prints out the history
help – Prints out help
```

Compile



```
parallels:~/Repos/csc415-fs$ make
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsInit.o fsInit.c -g -I.
gcc -c -o fsFree.o fsFree.c -g -I.
gcc -c -o fsDir.o fsDir.c -g -I.
gcc -c -o fsHelpers.o fsHelpers.c -g -I.
gcc -c -o b_io.o b_io.c -g -I.
gcc -o fsshell fsshell.o fsInit.o fsFree.o fsDir.o fsHelpers.o b_io.o fsLowM1.o -g -I. -lm -
l readline -l pthread
parallels:~/Repos/csc415-fs$
```

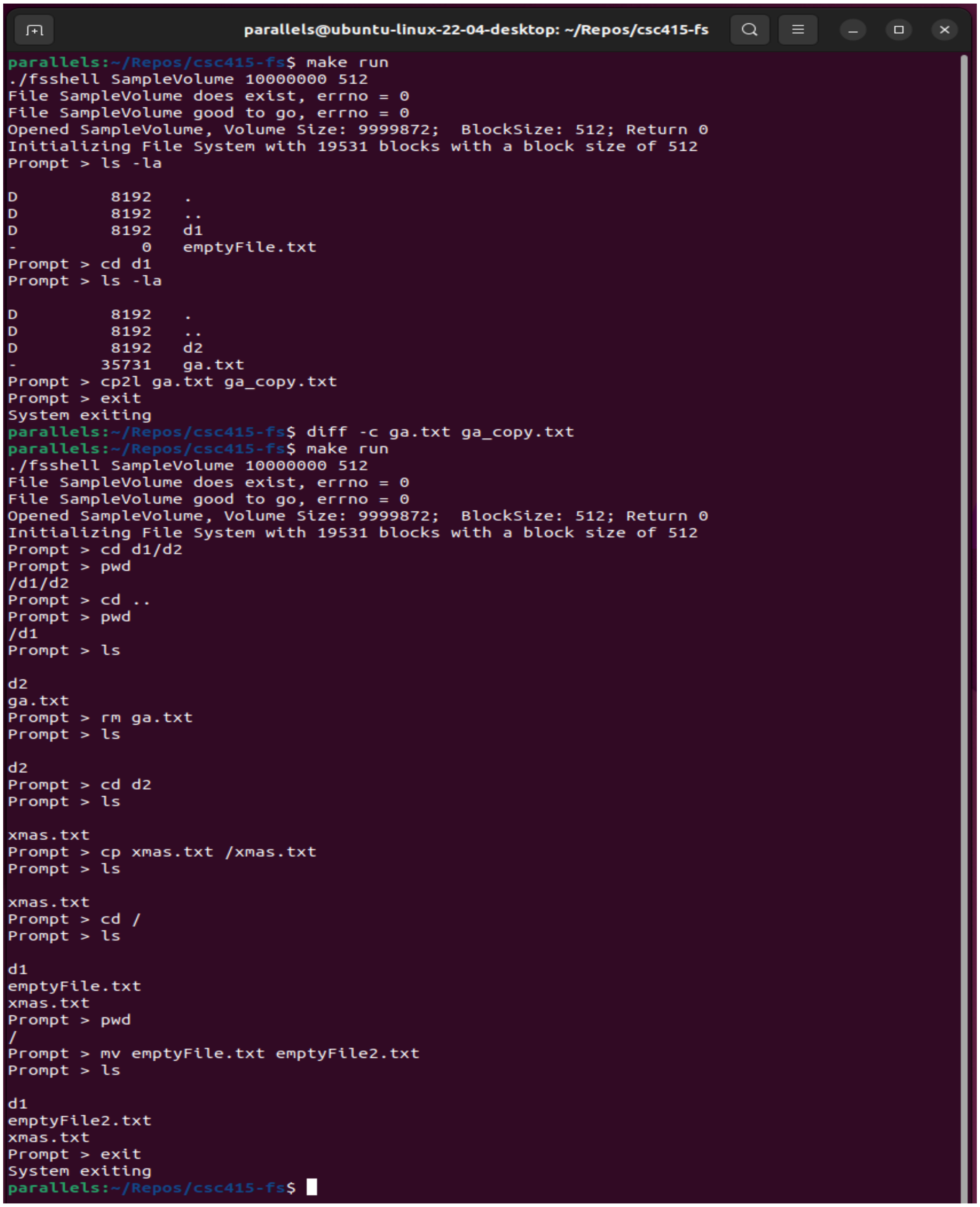
Initialize New Volume; md, touch, pwd, ls -la, cd, cp2fs, rm (directory)



```
parallels:~/Repos/csc415-fs$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does not exist, errno = 2
File SampleVolume not good to go, errno = 2
Block size is : 512
Created a volume with 9999872 bytes, broken into 19531 blocks of 512 bytes.
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Prompt > md d1
Prompt > touch emptyFile.txt
Prompt > pwd
/
Prompt > ls -la

D         8192   .
D         8192   ..
D         8192   d1
-            0   emptyFile.txt
Prompt > cd d1
Prompt > md d2
Prompt > md d3
Prompt > cp2fs ga.txt
Prompt > cp2fs xmas_stories.txt d2/xmas.txt
Prompt > ls -la

D         8192   .
D         8192   ..
D         8192   d2
D         8192   d3
-        35731   ga.txt
Prompt > rm d2
fs_rmdir: remove directory failed: Directory not empty: Success
Prompt > rm d3
Prompt > ls -la

D         8192   .
D         8192   ..
D         8192   d2
-        35731   ga.txt
Prompt > cwd
```

Load Existing Volume; cd, pwd, cp, mv

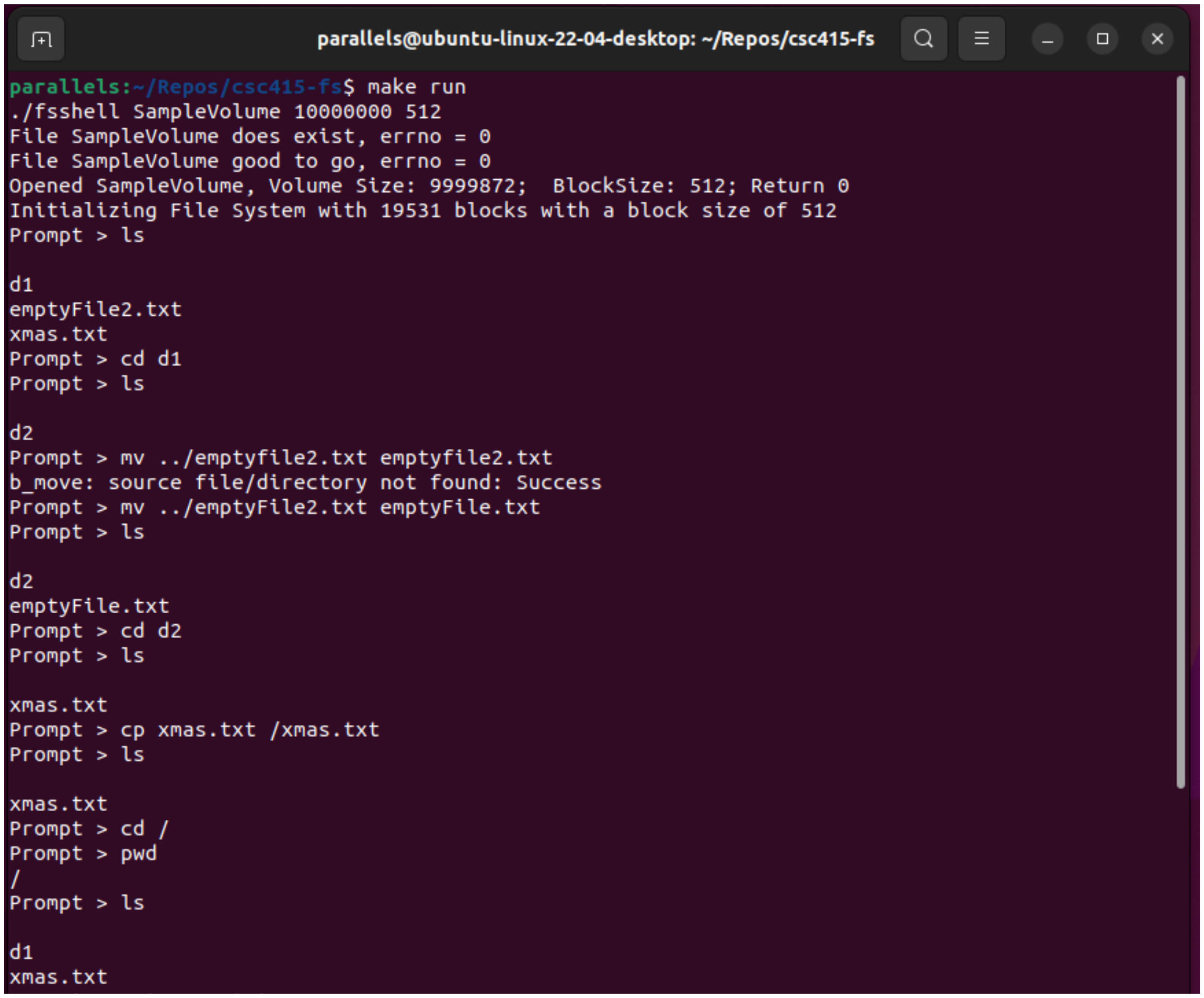
We also did a diff comparison between files after copying it to and back from the file system



parallels@ubuntu-linux-22-04-desktop: ~/Repos/csc415-fs

```
parallels:~/Repos/csc415-fs$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Prompt > ls -la

D          8192   .
D          8192   ..
D          8192   d1
-             0   emptyFile.txt
Prompt > cd d1
Prompt > ls -la

D          8192   .
D          8192   ..
D          8192   d2
-         35731   ga.txt
Prompt > cp2l ga.txt ga_copy.txt
Prompt > exit
System exiting
parallels:~/Repos/csc415-fs$ diff -c ga.txt ga_copy.txt
parallels:~/Repos/csc415-fs$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Prompt > cd d1/d2
Prompt > pwd
/d1/d2
Prompt > cd ..
Prompt > pwd
/d1
Prompt > ls

d2
ga.txt
Prompt > rm ga.txt
Prompt > ls

d2
Prompt > cd d2
Prompt > ls

xmas.txt
Prompt > cp xmas.txt /xmas.txt
Prompt > ls

xmas.txt
Prompt > cd /
Prompt > ls

d1
emptyFile.txt
xmas.txt
Prompt > pwd
/
Prompt > mv emptyFile.txt emptyFile2.txt
Prompt > ls

d1
emptyFile2.txt
xmas.txt
Prompt > exit
System exiting
parallels:~/Repos/csc415-fs$
```

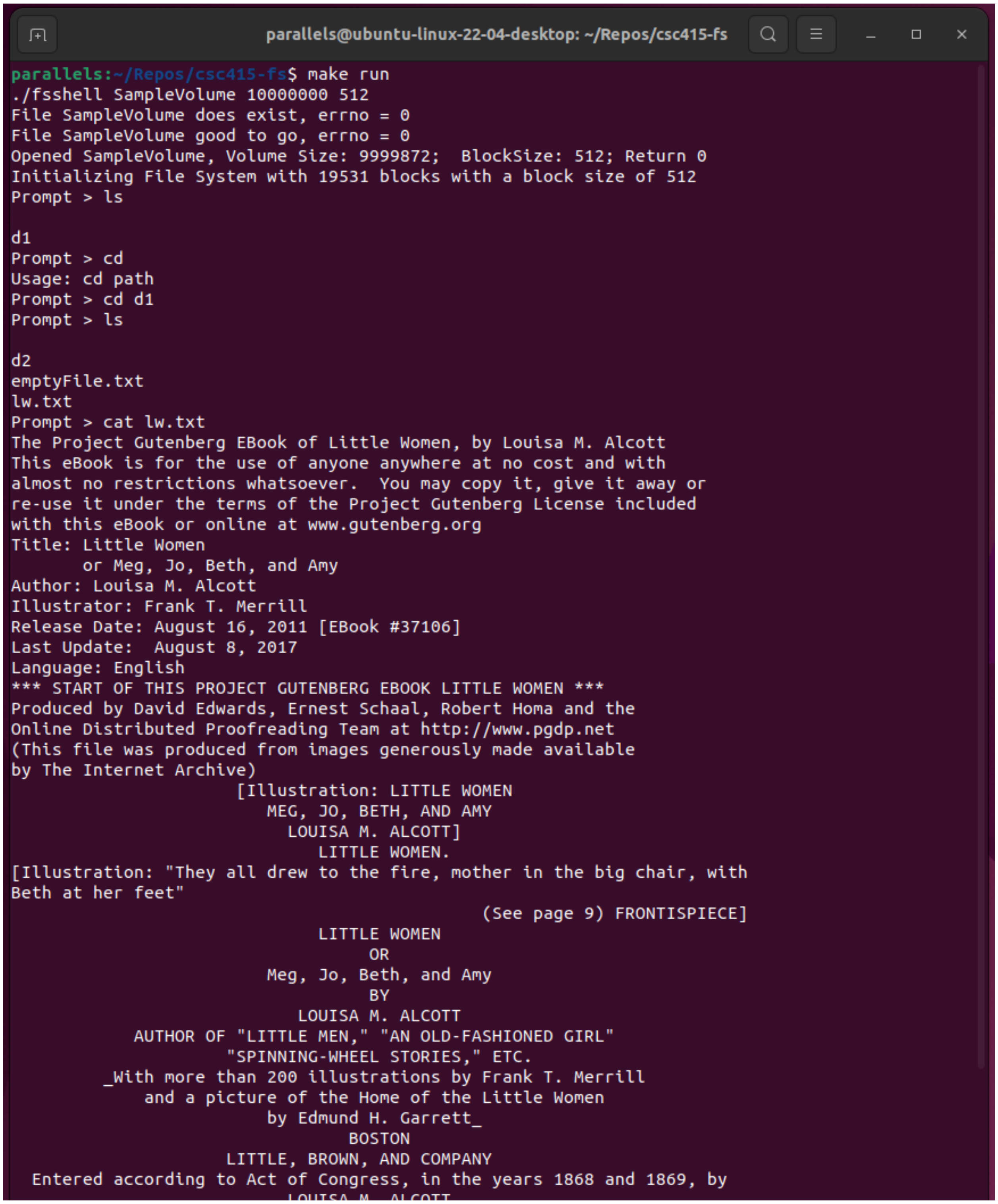
Load Existing Volume; ls, cd, mv, cp, pwd



parallels@ubuntu-linux-22-04-desktop: ~/Repos/csc415-fs

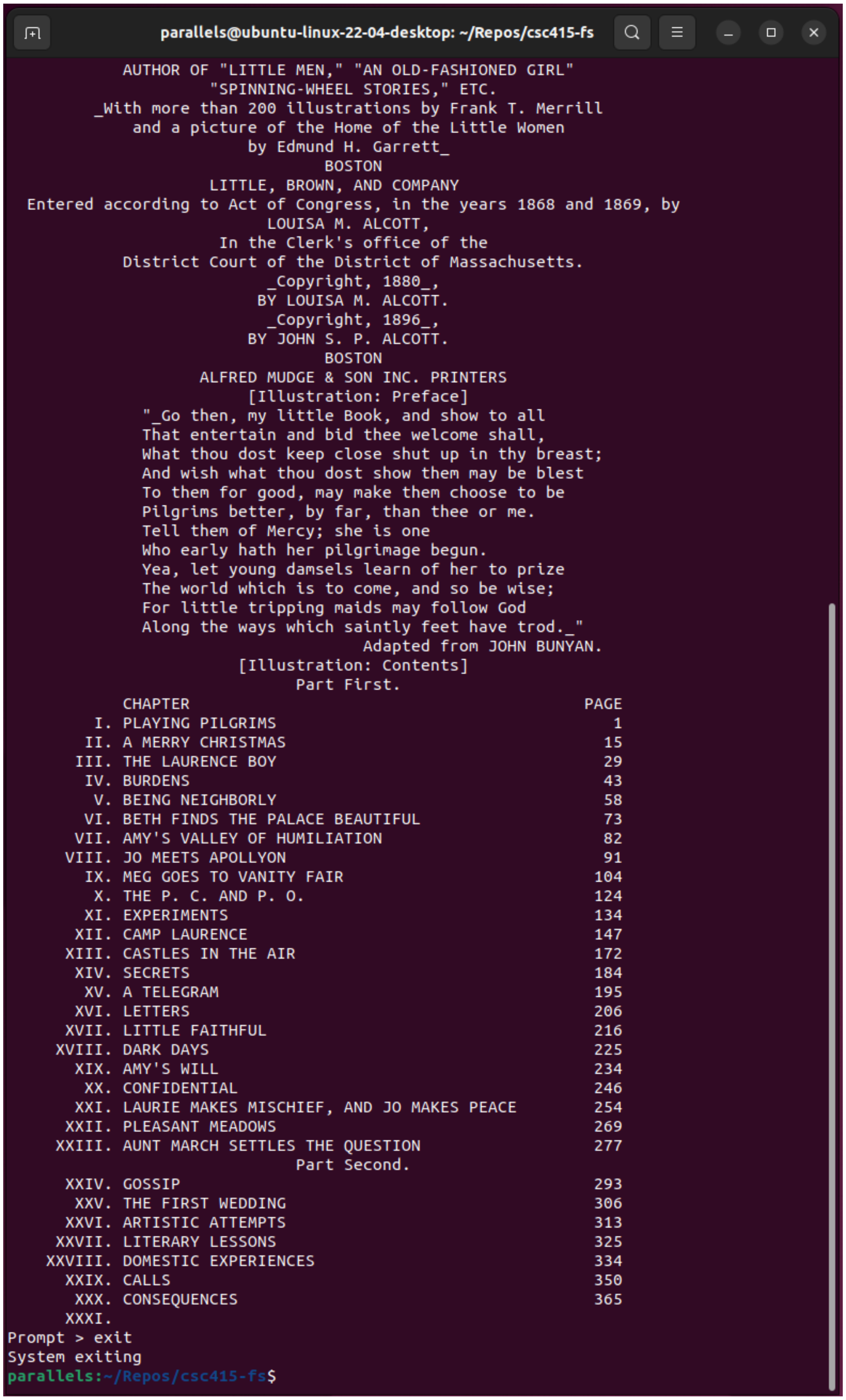
```
parallels:~/Repos/csc415-fs$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Prompt > ls

d1
emptyFile2.txt
xmas.txt
Prompt > cd d1
Prompt > ls

d2
Prompt > mv ../emptyfile2.txt emptyfile2.txt
b_move: source file/directory not found: Success
Prompt > mv ../emptyFile2.txt emptyFile.txt
Prompt > ls

d2
emptyFile.txt
Prompt > cd d2
Prompt > ls

xmas.txt
Prompt > cp xmas.txt /xmas.txt
Prompt > ls

xmas.txt
Prompt > cd /
Prompt > pwd
/
Prompt > ls

d1
xmas.txt
```

Load Existing Volume; cat



parallels@ubuntu-linux-22-04-desktop: ~/Repos/csc415-fs

```
parallels:~/Repos/csc415-fs$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Prompt > ls

d1
Prompt > cd
Usage: cd path
Prompt > cd d1
Prompt > ls

d2
emptyFile.txt
lw.txt
Prompt > cat lw.txt
The Project Gutenberg EBook of Little Women, by Louisa M. Alcott
This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.org
Title: Little Women
       or Meg, Jo, Beth, and Amy
Author: Louisa M. Alcott
Illustrator: Frank T. Merrill
Release Date: August 16, 2011 [EBook #37106]
Last Update:  August 8, 2017
Language: English
*** START OF THIS PROJECT GUTENBERG EBOOK LITTLE WOMEN ***
Produced by David Edwards, Ernest Schaal, Robert Homa and the
Online Distributed Proofreading Team at http://www.pgdp.net
(This file was produced from images generously made available
by The Internet Archive)
                      [Illustration: LITTLE WOMEN
                         MEG, JO, BETH, AND AMY
                           LOUISA M. ALCOTT]
                              LITTLE WOMEN.
[Illustration: "They all drew to the fire, mother in the big chair, with
Beth at her feet"
                                              (See page 9) FRONTISPIECE]
                              LITTLE WOMEN
                                   OR
                         Meg, Jo, Beth, and Amy
                                   BY
                            LOUISA M. ALCOTT
            AUTHOR OF "LITTLE MEN," "AN OLD-FASHIONED GIRL"
                     "SPINNING-WHEEL STORIES," ETC.
         _With more than 200 illustrations by Frank T. Merrill
             and a picture of the Home of the Little Women
                         by Edmund H. Garrett_
                                 BOSTON
                      LITTLE, BROWN, AND COMPANY
  Entered according to Act of Congress, in the years 1868 and 1869, by
```


parallels@ubuntu-linux-22-04-desktop: ~/Repos/csc415-fs

```
            AUTHOR OF "LITTLE MEN," "AN OLD-FASHIONED GIRL"
                     "SPINNING-WHEEL STORIES," ETC.
         _With more than 200 illustrations by Frank T. Merrill
             and a picture of the Home of the Little Women
                         by Edmund H. Garrett_
                                 BOSTON
                     LITTLE, BROWN, AND COMPANY
  Entered according to Act of Congress, in the years 1868 and 1869, by
                           LOUISA M. ALCOTT,
                      In the Clerk's office of the
            District Court of the District of Massachusetts.
                           _Copyright, 1880_,
                          BY LOUISA M. ALCOTT.
                           _Copyright, 1896_,
                         BY JOHN S. P. ALCOTT.
                                 BOSTON
                    ALFRED MUDGE & SON INC. PRINTERS
                         [Illustration: Preface]
              "_Go then, my little Book, and show to all
              That entertain and bid thee welcome shall,
              What thou dost keep close shut up in thy breast;
              And wish what thou dost show them may be blest
              To them for good, may make them choose to be
              Pilgrims better, by far, than thee or me.
              Tell them of Mercy; she is one
              Who early hath her pilgrimage begun.
              Yea, let young damsels learn of her to prize
              The world which is to come, and so be wise;
              For little tripping maids may follow God
              Along the ways which saintly feet have trod._"
                                     Adapted from JOHN BUNYAN.
                        [Illustration: Contents]
                              Part First.
            CHAPTER                                             PAGE
         I. PLAYING PILGRIMS                                       1
        II. A MERRY CHRISTMAS                                     15
       III. THE LAURENCE BOY                                      29
        IV. BURDENS                                               43
         V. BEING NEIGHBORLY                                      58
        VI. BETH FINDS THE PALACE BEAUTIFUL                       73
       VII. AMY'S VALLEY OF HUMILIATION                           82
      VIII. JO MEETS APOLLYON                                     91
        IX. MEG GOES TO VANITY FAIR                              104
         X. THE P. C. AND P. O.                                  124
        XI. EXPERIMENTS                                          134
       XII. CAMP LAURENCE                                        147
      XIII. CASTLES IN THE AIR                                   172
       XIV. SECRETS                                              184
        XV. A TELEGRAM                                           195
       XVI. LETTERS                                              206
      XVII. LITTLE FAITHFUL                                      216
     XVIII. DARK DAYS                                            225
       XIX. AMY'S WILL                                           234
        XX. CONFIDENTIAL                                         246
       XXI. LAURIE MAKES MISCHIEF, AND JO MAKES PEACE            254
      XXII. PLEASANT MEADOWS                                     269
     XXIII. AUNT MARCH SETTLES THE QUESTION                      277
                              Part Second.
      XXIV. GOSSIP                                               293
       XXV. THE FIRST WEDDING                                    306
      XXVI. ARTISTIC ATTEMPTS                                    313
     XXVII. LITERARY LESSONS                                     325
    XXVIII. DOMESTIC EXPERIENCES                                 334
      XXIX. CALLS                                                350
       XXX. CONSEQUENCES                                         365
      XXXI.
Prompt > exit
System exiting
parallels:~/Repos/csc415-fs$
```