FILE AND DIRECTORY MANAGEMENT

A REPORT ON PACKAGE

SUBJECT: OPERATING SYSTEMS

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES, PSG COLLEGE OF TECHNOLOGY,COIMBATORE - 641004

SUBMITTED BY
SANGEETHA V D - 18PT31
SRINIDI V - 18PT37

TABLE OF CONTENTS

FILE AND DIRECTORY MANAGEMENT

Abstract

The package aims at understanding and implementing the file descriptors and directory descriptors in efficient ways. This package consists of five files, each having a different purpose for the above mentioned descriptors. Precisely, we can use this project to understand directory management and file manipulations and familiarize with operating system calls (especially, file system).

1.1 Introduction

The objective of this assignment is to provide an insight into the directory structure and properties and file properties that can be used to provide user friendly filters that help by providing only the required information about the files and directories to the user. Most of the inputs required for the programs are passed as command line arguments and C Programming is used for this task. The main function accepts two arguments, the second one is a character array which stores the file name at zeroth index followed by the command line arguments passed and the first one is an integer which stores the array size. Our task was to create the following five files.

1. Filter.c
2. Mysize.c
3. Dircount.c
4. Wordmatch.c
5. File.c

1.2 Description

- *Filter.c* provides different filters specific to a particular file which is also specified. We manipulate the contents of the file according to the specified filters and its output is displayed to the user and the contents of the file are also changed accordingly. The filters that can be applied are
    1. -M: converts all the lowercase letters into uppercase letters.
    2. -m: converts all the uppercase letters into lowercase letters.
    3. -T: converts all the uppercase letters into lowercase letters and all the lowercase letters into uppercase letters (toggle case).
    4. -C: gets an integer from the user and prints all the words of length value stored in the integer along with the number of occurrences of each word.
    5. -R: replaces all instances of a particular string to another string and both the strings are given as input by the user.
    6. -P: finds all the instances matching the pattern which is given as the input from the user and are highlighted.
    7. -D: finds and deletes all the instances of the word given as the input from the user.
    8. -nofilter: it does not apply any filter to the file, it just prints the whole file without any changes.

- *Mysize.c* provides different filters to deal with the files, based on their sizes, present in the current working directory of this program. If no filter is applied, it prints the size of all files in the current working directory. The filters that can be applied are
    1. -se: displaying file name, size of all files in the current directory with size equal to the specified one.
    2. -sg: displaying file name, size of all files in the current directory with size greater than or equal to the specified one.
    3. -sl: displaying file name, size of all files in the current directory with size lesser than or equal to the specified one.

4. -fe: displaying the file name, size of all files in the current directory with file name length equal to the specified one.
5. -fg: displaying file name, size of all files in the current directory with file name length greater than or equal to the specified one.
6. -fl: displaying file name, size of all files in the current directory with file name length lesser than or equal to the specified one.
7. -c: displaying all c files.
8. -t: displaying text files.

- *Dircount.c* displays the content of any directory specified and present and displays the content along with the count of the number of files and directories that it possesses. The arguments that can be applied are
    1. -i: displaying a detailed, clear structure of the entire system created hierarchically.
    2. -d: displaying only the directories by their hierarchy.

- *Wordmatch.c* deals with displaying all files in the directory mentioned by the user and the count of the given word in its content, so, simply stating, deals with extensive reading of the files and producing output accordingly based on specifications. Both the directory name and the word to be searched are given as command line arguments.

- *File.c* deals with the manipulations of the file names of all the files that were created and providing filters to produce the required output. The filters that can be applied are
    1. -sd: displaying all files with file name length equal to the specified size.
    2. -sdg: displaying all files with file name length greater than or equal to the specified size.
    3. -sdl: displaying all files with file name length lesser than or equal to the specified size.
    4. -c: displaying all files with the starting letter as provided.
    5. -e: displaying all files with the ending letter as provided.
    6. -x: displaying all files with the starting pattern in the file name as provided.

7. -y: displaying all files with the ending pattern in the file name as provided.
8. -z: displaying all files with the pattern anywhere in the file name as provided.
9. -info: displaying all the information about any file which is specified.
10. -d: deleting the file or the directory passed

1.3 System Calls Used

Some system calls that were used include:

1. system() - to call various linux commands from within the c program
2. open() - to open the required files
3. read() - to read the contents of the required files
4. write() - to write into the required files
5. close() - to close files that were opened
6. opendir() - to open a directory whose contents are to be examined
7. readdir() - to read the content of the directory
8. lseek() - to change the position of the read/write pointer of a file descriptor
9. closedir() - to close a directory that was opened

1.4 Tools and Technologies

1. Ubuntu LTS 18.04LTS Application
2. GCC 7 compiler
3. Required header files were added

1.5 Workflow

For the operating system kernel, all files opened are identified using file descriptors. A file descriptor is a non-negative integer. When we open a file which already exists, the kernel returns a file descriptor to the process. When we want to read or write from/to a file, we identify the file with the file descriptor that was returned by the open call. Each open file has a current read/write position ("current file offset"). It is represented by a non-negative integer that measures the number of bytes from the beginning of the file. The read and write operations normally start at the current position and create an increment in that position equal to the number of bytes read or written. By default, this position is initialized to 0 when a file is opened, unless the option O_APPEND is specified. The current position (current_offset) of an opened file can be changed explicitly using the system call lseek.

To manipulate directories, we mainly used the system calls opendir, readdir and closedir. An open directory is identified with a directory descriptor, which is a pointer of type DIR (DIR*). When we open a directory with opendir, the kernel returns a directory descriptor from which the different entries to that directory can be read using the calls to the function readdir. The call readdir returns a directory entry in a pointer to a structure of type dirent (struct dirent*). Such structure will contain the fields corresponding to that entry such as the name of the entry, the type (if it is a normal file, another directory, symbolic links, etc.). Repeated calls to the function readdir will be returning the next entries in an open directory.

To run the five .c files,

1. For filter.c, use *./a.out <filtername> <inputfile>*

   Example: $cat temp.txt

   Operating Systems

   $./a.out -M temp.txt

   OPERATING SYSTEMS

   Note: The input file can be from any directory in the file system.

2. For mysize.c, use *./a.out <sizecondition> <size>*

   (i) Example: $touch simple.txt

   $./a.out -fl 5

   simple.txt

   … (displays all other files in the present working directory with file name length <=5)

   (ii) Example: $./a.out -c

   (displays all .c files)

3. For dircount.c, use *./a.out <directory name>*

   (i) Example: $mkdir sample

   $nano samp1.txt

   $./a.out sample

   .

   ..

   samp1.txt

   Files: 1

   Directories: 2

   (ii) Example: $./a.out -i

   (corresponding hierarchy of files and directories in your system is displayed)

4. For wordmatch.c, use *./a.out <directory name> <word-to-be-found>*

   Example: //assuming there is a directory called sample with a few files

   $./a.out sample life

   (displays all files in the directory with the word "life" with the count in each file)

5. For file.c, use *./a.out <corresponding filter> <size / start or end letter / pattern>*

   Example: $./a.out -c a

   (displays all files in the entire working space which have starting letter 'a')

1.6 Result and Discussion

On observing the purposes of each file, we can understand and get a wholesome idea of implementing the file and directory descriptors along with the usage of various string and commands manipulation.

Extensive use of header files like dirent.h, fcntl.h, sys/stat.h, time.h, string.h, and others like pwd.h, grp.h along with the regular ones can be observed.

1.7 Conclusion

We have successfully completed this assignment and have gained a deep insight into the usage of file descriptors and directory descriptors. We have also understood and demonstrated the various ways of acquiring the various file information and directory information using various operating system calls and presenting them in a user friendly way. Therefore, file manipulation and directory manipulation has been successfully demonstrated.

1.8 Bibliography

1.81 Books

- The UNIX System S.R. Bourne Addison-Wesley, 1983.
- Advanced UNIX Programming M.J. Rochkind Prentice-Hall, 1985.
- Programming Utilities and Libraries SUN Microsystems, 1990.
- Unix man pages (man function).

1.82 Websites

- https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm
- https://www.geeksforgeeks.org/introduction-of-system-call/
- https://www.sanfoundry.com/linux-program-read-file-using-file-descriptors/
- https://codeforwin.org/2018/03/c-program-to-list-all-files-in-a-directory-recursively.html
- http://man7.org/linux/man-pages/man3/
- http://web.theurbanpenguin.com/adding-color-to-your-output-from-c/
- http://codewiki.wikidot.com/c:system-calls:read
- https://www.geeksforgeeks.org/input-output-system-calls-c-create-open-close-read-write/
- https://linux-tips.com/t/hard-link-files/125/2
- https://c-for-dummies.com/blog/?p=3004
- https://cboard.cprogramming.com/c-programming/133346-determine-file-type-linux.html
- www.stackoverflow.com
- https://www.bottomupcs.com/file_descriptors.xhtml
- https://courses.engr.illinois.edu/cs241/sp2012/lectures/05-syscalls.pdf
- https://www.gnu.org/software/libc/manual/html_node/Testing-File-Type.html