## MINOR-1 PROJECT

**SYNOPSIS**

**ON**

**FILE CLASSIFIER TOOL USING C-LANGUAGE**

### Submitted By:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Roll No** | **Branch** | **SAP ID** |
| Nirmaljeet Singh | R171217039 | CSE-DevOps | 500060993 |
| Prajjawal Banati | R171217044 | CSE-DevOps | 500060722 |
| Vanshika Garg | R171217059 | CSE-DevOps | 500061511 |
| Vedansh Singhal | R171217060 | CSE-DevOps | 500061440 |

**Under the guidance of** Dr. Hitesh Kumar Sharma Assistant Professor (SG) Department of Cybernetics



### SCHOOL OF COMPUTER SCIENCE

**UNIVERSITY OF PETROLEUM & ENERGY STUDIES**

#### Dehradun – 248007, 2019-2020.

**Approved By**

(Dr. Hitesh Kumar Sharma) (Dr. Monit Kapoor)

#### Project Guide Department Head



**School of Computer Science**

**University of Petroleum & Energy Studies, Dehradun**

# Synopsis Report (2019-2020)

## Minor

I

**PROJECT TITLE:** FILE CLASSIFIER TOOL USING C Language.

#### ABSTRACT:

A file classifier tool is a program whose aim is to classify or differentiate files on the basis of different parameters. The most common parameter to categorize files can be the type of a file. When we talk about a system we can say that it contains huge amount of files which are needed to be accessed regularly. So there is a need of a utility which could manage our files.

**Keywords:** recommendations, collaborative filtering, movies.

# INTRODUCTION

A file classification scheme (also known as a file plan) is a tool that allows for classifying, titling, accessing and retrieving records. It is presented as a hierarchical structure of classification levels and is based on the business activities that generate records in a specific organizational business setting. Developing a file classification scheme is the process of identifying the category or categories of business activities and the records they generate and grouping them, if applicable, to facilitate retrieval, description, control, links and also for determining their disposition and access status. The development of a file classification scheme is based on an analysis of what are the functions and activities undertaken by an organization, so in other words on an analysis of what the organization does. Hence, file classifier will help to group same types of files under same folder. A variation of this basic algorithm is used in all systems dealing with large catalogues, like Apple and Windows in which all the files can be grouped helping many users to aggregate files.

# LITERATURE REVIEW

# There are a wide range of record order and coding frameworks, and there are no firm runs for picking a framework. Picking the correct documenting framework will rely upon various elements, for example,

# • The size and multifaceted nature of the association

# • The scope of its business

# • The amount of documents and different records

# • The nearness of case records

# • The pace of production of new documents and records

# • The cost of introducing and keeping up the framework

# • The simplicity or trouble with which the documents and records can be sorted out into totally unrelated classes reflecting explicit capacities and exercises

# • The preparing required to work and support the framework

# • The aptitudes level of the records staff.

# Record characterization and coding frameworks must be intended to coordinate the necessities of the association they will serve. A record characterization framework should bolster business or hierarchical necessities. It should suit the association it serves and bolster basic leadership and the exercises of the association.

# It should matches clients' needs.

# It ought to give the best, most straightforward and least difficult arrangement.

# It ought to be financially savvy.

# It should coordinate assets, with satisfactory gear, assets or staff.

# It ought not be subject to outside assets for operational prerequisites.

# A document grouping framework ought to be straightforward, use and keep up. It ought to be founded on rationale or sound judgment. It ought to be comprehended by records staff and clients. It ought to be free of human memory.

# It should utilize basic procedures. It ought to motivate trust in administrators and clients. A record characterization framework ought to be exact. It ought to limit question about where to document papers. It ought to permit the snappy recognizable proof and recovery of records. A record characterization framework ought to be finished and exhaustive. It should cover every one of the documents that should be incorporated. It ought to be equipped for including records that might be made in future.

# It ought to be adaptable and take into account development, constriction or revamping. A record order framework ought to be sponsored up by a systems manual and preparing materials. It ought to be plainly and thoroughly reported. All methods ought to be disclosed in simple to-pursue steps. It ought to give ace duplicates everything being equal, with finished models. It ought to be bolstered via preparing programs. It ought to be bolstered by expert exhortation or direction. A document arrangement framework ought to be effectively mechanized. It ought to be prepared to do some type of helpful computerization, paying little respect to whether robotization is arranged, for example, for word handling, mechanized ordering, database the executives or a modernized record-keeping framework. Documenting frameworks have advanced throughout the years from recording administrative work in boxes to complex programming programs that store documents electronically far out. In spite of the fact that you can pick an assortment today, all documenting frameworks share one primary objective: powerful records the board.

# Kinds of documenting frameworks:-

# Alphabetic Filing:- Alphabetic documenting is the most well-known documenting framework for under 5,000 records. Documenting by alphabetic request is where you organize records by names of people, organizations, foundations, offices, subjects, points or geographic areas as indicated by lexicon request. Related themes are not kept together in this framework. Typically this sort of framework is best when modest quantities of data are included. This sort of documenting and order framework is once in a while known as a "word reference" framework. At the point when individual names are being documented, last names are utilized as the essential sorter, with first names utilized uniquely on account of indistinguishable last names.

# Numeric Filing:- In setting up a numeric documenting framework, orchestrate documents in consecutive request utilizing the numbers legitimately from the record or an appointed number. Most frameworks utilize a list to recover the records. Numeric recording and order frameworks are easy to use, since they for the most part start at the main and mark each document with the consequent number. Be that as it may, the utilization of this sort of framework is constrained, as it regularly requires a file to enable clients to discover the records they look for, and high-action documents can end up clogged around the equivalent numeric zone. This sort of recording framework can deal with a lot of information. The various arrangements of numbers can compare to significant classifications and sub-classifications, paralleling the reference book arrangement of recording and grouping.

# Alpha-numeric Filing:- Alpha-numeric recording utilizes a blend of names and numbers. You ordinarily utilize this kind of recording framework with subject names and numbers. Organize records as indicated by alphabetic divisions or subject heading, at that point by number classification. In alphanumeric documenting frameworks, data is characterized by classification in an all-encompassing framework, however utilizing the two letters and numbers to mean classifications. The utilization of the two letters and numbers takes into consideration an a lot more prominent field of classes than does the utilization of numbers alone. Consequently the Library of Congress recording and characterization framework, which is alphanumeric, takes into account a more prominent exhibit of classifications than does the Dewey Decimal framework, which is constrained to ten noteworthy classifications.

# PROBLEM STATEMENT

In today’s world everyone lacks time in his/her everyday life. To search a file amongst hundreds of file is a tedious task. To manage these hundreds of files user has to create different folders for different extensions of files. But the problem is how a user can dedicate time to create hundred folders for hundred different files.

So with the help of C language we are going to create a program which creates different folders for every file with a unique file extension. In this way we are going to organize all the files with unique file extension together in a common folder, so that a user can directly access that folder which will save his/her time.

# OBJECTIVES

The path of the directory would be taken as input from the user. The files in the directory will be listed with their extensions. Extension would be stored and uniquely identified such that number of directories made will be equal to no. of unique extensions. Will use Low-Level File I/O to copy the files from the parent destination to its extension directory. Low-Level File I/O first open the file in the form of bytes and every byte of data is stored in a buffer from where it is copied to another file.

# METHODOLOGY

**Feasibility study:-**

Understanding the basic concept of classification of file systems, how it helps in managing files, need of efficient method for classification.

**Implementation of Managing directory:-**

In C “dirent.h” library is used to interact with the local repositories of the system. It contains many predefined functions like closedir, opendir, readdir.

The <dirent.h> header defines the following data type through typedef:

DIR:-A type representing a directory stream.

It also defines the structure dirent which includes the following members:

ino\_t d\_ino file serial number

char d\_name[] name of entry

The array d\_name is of unspecified size, but shall contain a filename of at most {NAME\_MAX} bytes followed by a terminating null byte. The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

**Functions:-**

int closedir(DIR \*);

DIR \*opendir(const char \*);

struct dirent \*readdir(DIR \*);

int readdir\_r(DIR \*restrict, struct dirent \*restrict, struct dirent \*\*restrict);

**Implementation of Low-Level File I/O:-**

In low-level File I/O, data cannot be written as individual characters, or as strings or as formatted data. One way data can be written or read in low-level file I/O functions-as a buffer full of bytes.

Since low-level file I/O functions parallel the methods that the OS uses to write to the disk, they are more efficient than the high-level file I/O functions.

Since there are few layers of routines to go through, low-level I/O functions operate faster than their high-level counterparts. Following functions are listed following the file I/O.

**FUNCTIONS:-**

**To open a file:- open(source, permissions):**-

In low-level file I/O open() returns an integer value called ‘file handle’. This is a number assigned to a particular file, which is used thereafter to refer to a file. It returns -1 for unsuccessful opening.

**To read a file:- read( inhandle, buffer, sizeofbuffer):-**

In low-level file I/O read() function returns the number of bytes actually read. This is an important number since it may be very well be less than the buffer size (512 bytes), and we will need to know just how full buffer is before we can do anything with its contents.

**To write into a file:- write( outhandle, buffer, bytes):-**

In low-level file I/O write() function inserts the data read by the buffer into the file which is opened by the user. It writes the data from the buffer array to the file in the form of bytes.

# SYSTEM REQUIREMENTS

Hardware:

* 64 bits processor architecture supported by windows.
* Minimum RAM requirement for proper functioning is 1 GB.
* 9-58 GB free hard disk space depending on edition and configuration, including space required for files.

Operating Systems:-

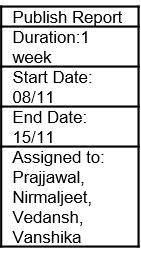
* -Windows(XP and above)
* -Ubuntu(4.04 and above)

Compilers:-

* GCC, Turbo C++, Code Blocks, Atom etc.

# SCHEDULE (PERT CHART)

# 



**REFERENCES**

* Agrawal, N., Bolosky, W.J., Douceur, J.R., Lorch, J.R.: A Five-Year Study of File-System Metadata. ACM Trans. Storage 3(3), 9 (2007)
* Ames, A., Maltzahn, C., Bobb, N., Miller, E.L., Brandt, S.A., Neeman, A., Hiatt, A., Tuteja, D.: Richer File System Metadata Using Links and Attributes. In: Proc. IEEE MSST, pp. 49–60 (2005)
* Gifford, D.K., Jouvelot, P., Sheldoon, M.A., Toole Jr., J.W.O.: Semantic File Systems. In: Proc. ACM SOSP, pp. 16–25 (1991)
* Let Us C by Yashwant Kanetkar
* https://pubs.opengroup.org/onlinepubs/7908799/xsh/dirent.h.html