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

<u>Synopsis Report (2019-2020)</u>

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 many different file classification and coding systems, and there are no hard and fast rules for choosing a system. Choosing the right filing system will depend on a number of factors, such as

- The size and complexity of the organization
- The range of its business
- The quantity of files and other records
- The presence of case files
- The rate of creation of new files and records
- The cost of installing and maintaining the system
- The ease or difficulty with which the files and records can be organized into mutually exclusive categories reflecting specific functions and activities
- The training required to operate and sustain the system
- The skills level of the records staff.

File classification and coding systems must be designed to match the requirements of the organization they will serve. A file classification system should support business or organizational requirements. It should suit the organization it serves and support decision making and the activities of the organization.

- o It should matches users' needs.
- o It should provide the best, easiest and simplest solution.
- o It should be cost effective.
- o It should match resources, with adequate equipment, funds or staff.
- o It should not be dependent on outside resources for operational requirements.

A file classification system should be easy to understand, use and maintain. It should be based on logic or common sense. It should be understood by records staff and users. It should be independent of human memory.
It should use simple processes. It should inspire confidence in operators and users. A file classification system should be precise. It should minimize doubt about where to file papers. It should allow the quick identification and retrieval of files. A file classification system should be complete and comprehensive. It should cover all the files that need to be included. It should be capable of including files that may be created in future.

It should be flexible and allow for expansion, contraction or reorganization. A file classification system should be backed up by a procedures manual and training materials. It should be clearly and comprehensively documented. All procedures should be explained in easy-to-follow steps. It should provide master copies of all forms, with completed examples. It should be supported by training programs. It should be supported by professional advice or guidance. A file classification system should be easily automated. It should be capable of some form of useful automation, regardless of whether automation is planned, such as for word processing, computerized indexing, database management or a computerized record-keeping system. Filing systems have evolved over the years from filing paperwork in boxes to sophisticated software programs that store files electronically out of sight. Although you can choose a variety today, all filing systems share one main goal: effective records management.

Types of filing systems:-

Alphabetic Filing:- Alphabetic filing is the most common filing system for less than 5,000 records. Filing by alphabetic order is a system where you arrange files by names of individuals, businesses, institutions, agencies, subjects, topics or geographic locations according to dictionary order. Related topics are not kept together in this system. Usually this type of system is best when small amounts of information are involved. This type of filing and classification system is sometimes known as a "dictionary" system. When personal names are being filed, last names are used as the primary sorter, with first names used only in the case of identical last names.

Numeric Filing:- In setting up a numeric filing system, arrange files in sequential order using the numbers directly from the record or an assigned number. Most systems use an index to retrieve the files. Numeric filing and classification systems are very simple to use, since they generally start at the number one and label each file with the subsequent number. However, the use of this type of system is limited, as it often requires an index to help users find the files they seek, and high-activity files can become congested around the same numeric area. This type of filing system can handle large amounts of data. The different sets of numbers can correspond to major categories and sub-categories, paralleling the encyclopedia system of filing and classification.

Alpha-numeric Filing:- Alpha-numeric filing uses a combination of names and numbers. You commonly use this type of filing system with subject names and numbers. Arrange files according to alphabetic divisions or subject heading, then by number category. In alphanumeric filing systems, information is classified by category in an encyclopedic system, but using both letters and numbers to denote categories. The use of both letters and numbers allows for a much greater field of categories than does the use of numbers alone. Thus the Library of Congress filing and classification system, which is alphanumeric, allows for a greater array of categories than does the Dewey Decimal system, which is limited to ten major categories.

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.

OBJECTIVE

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.

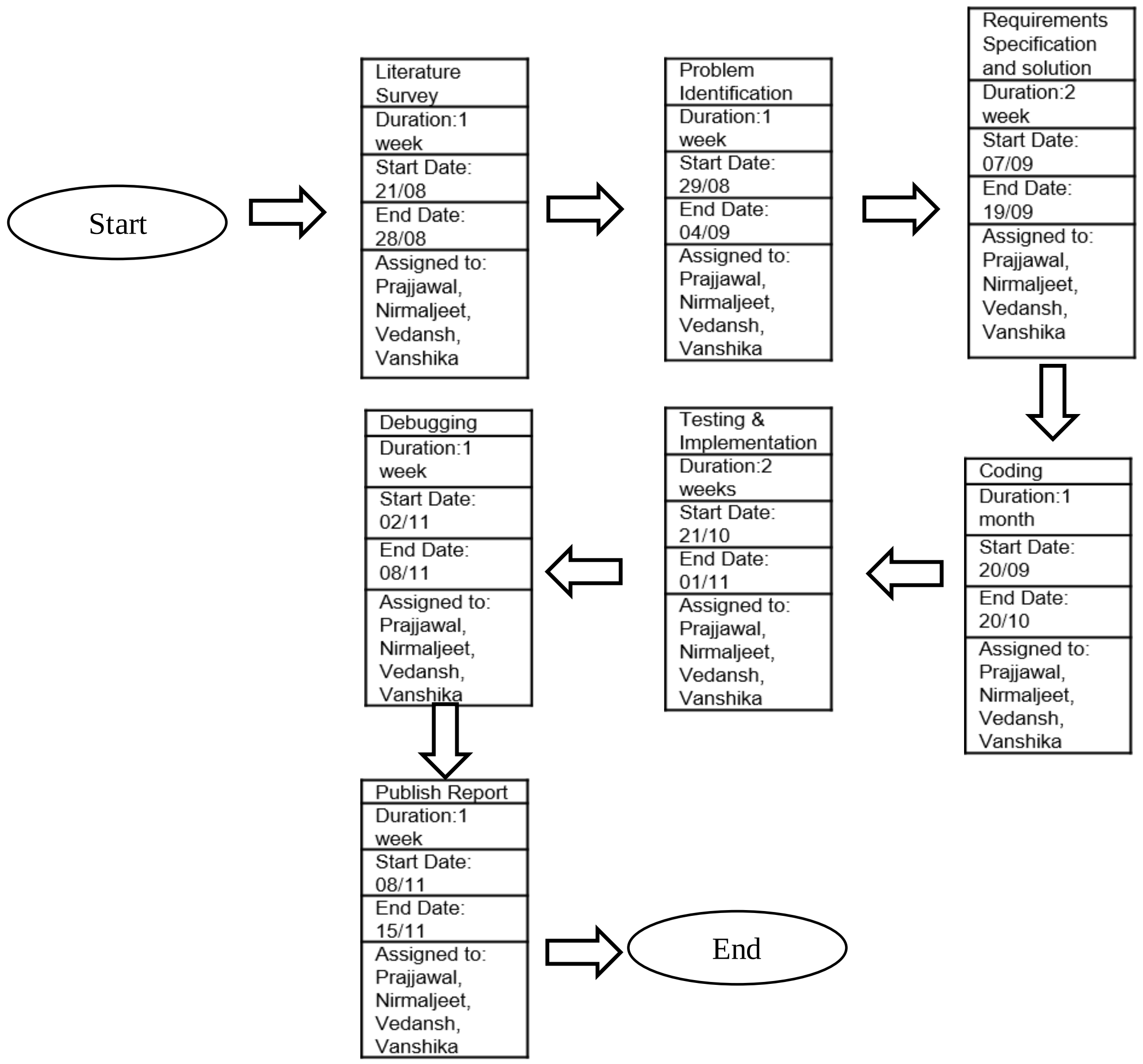
Operating Systems:-

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

Compilers:-

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

SCHEDULE (PERT CHART)



REFERENCES

- https://pubs.opengroup.org/onlinepubs/7908799/xsh/dirent.h.html
- 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