**CS 390**
**SPRING 2024**
**Mr. Rheinfurth**
**Programming Assignment 3**

## Introduction

The "filter" in Unix is implemented using a standard design. The compiled program is used to fit into a piped set of commands to implement a desired operation.

## Assignment Description

This assignment is to write a filter program which will take an input file name from stdin, modify the file, and send the output to stdout. Specifically, the filter will detect if the file is a text or a binary file. If the file is a text file and the optional argument "-u" has been entered the program will convert the file from a Windows to a Unix formatted file.

## Program Requirements

1. The student shall create a "C" program which will read stdin consisting of a stream of file names.
2. Each file shall be read as a binary file.
3. Each file shall be tested to determine if it is a text file.
4. One of 4 messages shall be displayed on stderr if there is an error reading the file.
   1) The file could not be opened
   2) It is a zero length file
   3) It could not fit into the buffer
   4) There was an error reading the file.
5. If the file is not a text file a message shall be displayed on stderr indicating that it is not text. If it is a text file the file name shall be display on stdout.
6. If 1) the user inputs a "-u", 2) the file is a text file and 3) the file is a windows file, the file shall be converted to a Unix file.
7. If the file has been converted to a Unix file it shall be saved back to the file system.

## Submission Guidelines

1. The assignment will be uploaded to the instructor using the **CANVAS** website.
2. The script file will use the following naming convention:
   **StudentLastName_cs390program3.c**
   Example: If your last name is smith the filename would be:
   **smith_cs390program3.c**
3. The assignment will be due Tuesday, April 9, 2024.
4. The assignment may be turned in Thursday, April 11, 2024 for a mandatory loss of one letter grade. The assignment will not be accepted after this date.

```
#include <stdio.h>
#include <malloc.h>
#include <string.h>
#include <ctype.h>

unsigned char* buffer;
int buffersize=0,datasize=0;

#define ONE_MB 1000000

int ReadBinaryFile(const char* filename);
int IsItText      (unsigned char* buffer,int datasize);
int ToUnix        (unsigned char* buffer,int datasize);

int main(int argc,char* argv[])
{
    int         status;
    const char  filename[] = “abc.txt”;

    buffersize = 5 * ONE_MB;
    buffer     = (unsigned char*) malloc(buffersize);

    status = ReadBinaryFile(filename);
    if (status > 0)
    {
        if      (status == 1) fprintf(stderr,"COULD NOT OPEN FILE!!");
        else if (status == 2) fprintf(stderr,"FILE IS ZERO LENGTH!!");
        else if (status == 3) fprintf(stderr,"FILE TOO BIG FOR BUFFER!!");
        else if (status == 4) fprintf(stderr,"ERROR READING FILE!!");
    }
    else
    {
        if (!IsItText(buffer,datasize))
        {
            fprintf(stderr,"FILE %s IS *NOT* TEXT",filename);
        }
        else
        {
            if (strstr((char*) buffer,"\r"))
            {
                fprintf(stdout,"FILE IS WINDOWS");
                ToUnix(buffer,datasize);
            }
            else
            {
                fprintf(stdout,"FILE IS UNIX");
            }
        }
    }
    if (buffer) free(buffer);
    return 0;
}
```

```
int ReadBinaryFile(const char* filename)
{
    int    status = 0;
    int    bytesInFile = 0;
    FILE*  ifs = 0;

    ifs = fopen(filename, "rb");
    if (!ifs)
    {
        status = 1;
    }
    else
    {
        /* make sure file will fit into buffer */
        fseek(ifs,0,SEEK_END);
        bytesInFile = (int) ftell(ifs);
        fseek(ifs,0,SEEK_SET);

        if (bytesInFile == 0)
        {
            status = 2;
        }
        else
        {
            if (bytesInFile >= (buffersize - 1))
            {
                status = 3;
            }
            else
            {
                datasize = (int) fread(buffer,1,bytesInFile,ifs);
                if (datasize != bytesInFile)
                {
                    status = 4;
                }
                else
                {
                    /* if we read a text file then this */
                    /* turns buffer into a "C" string   */
                    buffer[datasize] = '\0';
                }
            }
        }
    }
    if (ifs) fclose(ifs);
    return status;
}
int WriteBinaryFile(const char* filename)
{
    int   status = 0;
    FILE* ofs;

    ofs = fopen(filename,"wb");
    if (ofs)
    {
        fwrite(buffer,1,datasize,ofs);
        fclose(ofs);
        status = 1;
    }
    return status;
}
```

```
int charIsText(char c)
{
    int status = 0;
    if (isprint(c) != 0)
    {
        status = 1;
    }
    else
    {
        switch(c)
        {
        case '\f' :
        case '\t' :
        case '\r' :
        case '\n' :
        case '\a' :
        case '\b' :
        case '\v' : status = 1;
                    break;
        }
    }
    return status;
}
int IsItText(unsigned char* buffer,int datasize)
{
    int i,count;

    count = datasize < 100 ? datasize : 100;

    return 1;
}
int ToUnix(unsigned char* buffer,int datasize)
{
    int i,j,count;

    return 1;
}
```