CSC415 – Homework #2

Aleksandr Kibis

9/16/2014

The goal for this assignment was to create a basic version of the “cp” or “copy” program. While generally an easy undertaking, the necessity for using system calls made it tougher. As with our last assignment, two versions of the program were made; POSIX and Win32. The FILE I/O family of functions were used. This included open, read, write, and close for POSIX, plus CreateFile, ReadFile, and WriteFile for Win32. I have begun seeing a trend after programming for both platforms for 2 projects now. While POSIX has possibly less detailed information, it is also way easier to program for and looks way better in the end than Win32. The Win32 version just looks a bit clunky but it is more readable. There are also inconsistencies between the GCC and CL compilers. CL is very picky with variable initialization which caused me a bunch of headaches. One thing I did notice is that Win32 did not require OpenFile to be used, this helped with consolidation of code. This was offset by the need for an entire function being written just to display error messages.

Windows

Code

```c
/*
@File: main.c
@Author: Aleksandr Kibis
@Date: 9/15/2014

This is a windows implementation of a copy operator. Given 1 or two text documents, this
program copies the first document under the name of the second. The number of bytes read is returned.
*/

#include <stdio.h>
#include <Windows.h>
#include <strsafe.h>

#pragma comment(lib,"user32.lib");

#define EXIT_SUCCESS 0
#define EXIT_FAILURE 1
#define BUF_SIZE 256

void getError(LPTSTR lpszFunction);

int main(int argc, char* argv[]) {

        // input and file handlers
        char* inputFile;
        char* copiedFile;
        char* buf;
        HANDLE input;
        HANDLE output;

        // read/write counters
        int nbytes;
        DWORD bytes_read;
```

```
	DWORD bytes_written;
	int readReturn;
	int writeReturn;
	int totalBytes = 0;

	if (argc == 3) {
		inputFile = argv[1];
		copiedFile = argv[2];

		//printf("inputFile: %s\n", inputFile);
		//printf("copiedFile: %s\n", copiedFile);

		// OPEN FILE TO BE COPIED
		input = CreateFile(inputFile, GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL,
NULL);
		//printf("input: %d\n", (int)input);

		//getError(TEXT("GetProcessId"));


		// OPEN DESTINATION FILE, CREATE IF DOESN'T EXIST
		output = CreateFile(copiedFile, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);
		buf = (char*)malloc(BUF_SIZE * sizeof (char));

		//getError(TEXT("GetProcessId"));

		// COPY FILE TEXT

		while (1) {
			nbytes = BUF_SIZE;
			readReturn = ReadFile(input, buf, nbytes, &bytes_read, NULL);
			//getError(TEXT("GetProcessId"));
			totalBytes += bytes_read;
			//printf("Check: %d\n", bytes_read);
			if (bytes_read < BUF_SIZE) {
				writeReturn = WriteFile(output, buf, bytes_read, &bytes_written, NULL);
				break;
			}
			//printf("chunkSize: %d\n", bytes_read);
			writeReturn = WriteFile(output, buf, nbytes, &bytes_written, NULL);
			//printf("%s", buf);

		}
		// PRINT TOTAL BYTES COPIED
		printf("copied %d bytes\n", totalBytes);

		// COMPARE THE TWO FILES


		// FREE RESOURCES
		memset(buf, 0, sizeof(char));
		free(buf);
		CloseHandle(input);
		CloseHandle(output);
		fflush(stdin);
		fflush(stdout);

		exit(EXIT_SUCCESS);
	} else {
		fprintf(stderr, "Please enter two arguments.\n");
		exit(EXIT_FAILURE);
	}


	//return (EXIT_SUCCESS);
}

// implementation of a getError function. Code borrowed from:
```

```c
// http://msdn.microsoft.com/en-us/library/windows/desktop/ms680582(v=vs.85).aspx
void getError(LPTSTR lpszFunction) {
        DWORD error = GetLastError();
        LPVOID lpMsgBuf;
        LPVOID lpDisplayBuf;

        FormatMessage(
                FORMAT_MESSAGE_ALLOCATE_BUFFER |
                FORMAT_MESSAGE_FROM_SYSTEM |
                FORMAT_MESSAGE_IGNORE_INSERTS,
                NULL,
                error,
                MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                (LPTSTR) &lpMsgBuf,
                0, NULL );

        lpDisplayBuf = (LPVOID)LocalAlloc(LMEM_ZEROINIT,
                (lstrlen((LPCTSTR)lpMsgBuf) + lstrlen((LPCTSTR)lpszFunction) + 40) * sizeof(TCHAR));
        StringCchPrintf((LPTSTR)lpDisplayBuf,
                LocalSize(lpDisplayBuf) / sizeof(TCHAR),
                TEXT("%s failed with error %d: %s"),
                lpszFunction, error, lpMsgBuf);
        MessageBox(NULL, (LPCTSTR)lpDisplayBuf, TEXT("Error"), MB_OK);

        LocalFree(lpMsgBuf);
        LocalFree(lpDisplayBuf);
}
```

Output

```
C:\Users\rusky\Dropbox\Fall 2014\415\415 - Homework 2 - CP Operator\415 - Homework 2 - CP Operator\Submission\Win32>cl -o test main.c

Microsoft (R) C/C++ Optimizing Compiler Version 17.00.61030 for x64

Copyright (C) Microsoft Corporation.  All rights reserved.



cl : Command line warning D9035 : option 'o' has been deprecated and will be rem

oved in a future release

main.c

main.c(14) : warning C4081: expected 'newline'; found ';'

Microsoft (R) Incremental Linker Version 11.00.61030.0

Copyright (C) Microsoft Corporation.  All rights reserved.



/out:main.exe

/out:test.exe

main.obj
```

TEXT INPUT

```
C:\Users\rusky\Dropbox\Fall 2014\415\415 - Homework 2 - CP Operator\415 - Homework 2 - CP Operator\Submission\Win32>test.exe
textInput.txt copiedText.txt

copied 446 bytes

C:\Users\rusky\Dropbox\Fall 2014\415\415 - Homework 2 - CP Operator\415 - Homework 2 - CP Operator\Submission\Win32>comp
textInput.txt copiedText.txt

Comparing textInput.txt and copiedText.txt...

Files compare OK

Compare more files (Y/N) ? n
```

BINARY INPUT

```
C:\Users\rusky\Dropbox\Fall 2014\415\415 - Homework 2 - CP Operator\415 - Homework 2 - CP Operator\Submission\Win32>test.exe
binaryInput.txt copiedBinary.txt

copied 178 bytes

C:\Users\rusky\Dropbox\Fall 2014\415\415 - Homework 2 - CP Operator\415 - Homework 2 - CP Operator\Submission\Win32>comp
binaryInput.txt copiedBinary.txt

Comparing binaryInput.txt and copiedBinary.txt...

Files compare OK

Compare more files (Y/N) ? n
```

Linux

Code

```c
/*
 * File:   main.c
 * Author: netdom
 *
 * Created on September 9, 2014, 5:30 PM
 */

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

#define EXIT_SUCCESS 0
#define EXIT_FAILURE 1
#define BUF_SIZE 128

/*
 * This program uses low level POSIX calls to copy a file.
 * OPEN, CLOSE, READ, WRITE
 */
int main(int argc, char** argv) {

    char* inputFile;
    char* copiedFile;
    char* buf;
    int input;
    int output;

    if (argc == 3) {
        mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH;
        inputFile = argv[1];
        copiedFile = argv[2];

        //printf("inputFile: %s\n", inputFile);
        //printf("copiedFile: %s\n", copiedFile);

        // OPEN FILE TO BE COPIED
        input = open(inputFile, O_RDONLY | O_CREAT);
        //printf("input: %d\n", input);

        // OPEN DESTINATION FILE, CREATE IF DOESN'T EXIST
        output = open(copiedFile, O_WRONLY | O_CREAT | O_TRUNC, mode);
        buf = malloc(BUF_SIZE * sizeof (char));

        size_t nbytes;
        ssize_t bytes_read;
        ssize_t bytes_written;

        int totalBytes;

        // COPY FILE TEXT
        while (1) {
            nbytes = BUF_SIZE;
            bytes_read = read(input, buf, nbytes);
            totalBytes += bytes_read;
            //printf("Check: %d\n", (int) bytes_read);
            if (bytes_read < BUF_SIZE) {
                bytes_written = write(output, buf, (size_t) bytes_read);
                break;
            }
            //printf("chunkSize: %d\n", (int)bytes_read);
            bytes_written = write(output, buf, nbytes);
            //printf("%s", buf);

        }
        // PRINT TOTAL BYTES COPIED
        printf("copied %d bytes\n", totalBytes);

        // FREE RESOURCES
        free(buf);
        close(input);
        close(output);

        exit(EXIT_SUCCESS);
    } else {
        fprintf(stderr, "Please enter two arguments.\n");
        exit(EXIT_FAILURE);
    }


    return (EXIT_SUCCESS);
}
```

Output

TEXT INPUT

```
netdom@netdom-virtual-machine ~/Desktop $ gcc -o test main.c

netdom@netdom-virtual-machine ~/Desktop $ ./test input copiedTest

copied 447 bytes
```

```
netdom@netdom-virtual-machine ~/Desktop $ diff input copiedTest

netdom@netdom-virtual-machine ~/Desktop $
```

BINARY INPUT

```
netdom@netdom-virtual-machine ~/Desktop $ ./test binaryInput binaryOutput

copied 179 bytes

netdom@netdom-virtual-machine ~/Desktop $ diff binary

binaryInput   binaryOutput

netdom@netdom-virtual-machine ~/Desktop $ diff binaryInput binaryOutput

netdom@netdom-virtual-machine ~/Desktop $
```