

```

// Vahe Karamian - www.karamian.com
// Filename: cp.c

// Basic cp file copy program

// Usage: cp file1 file2 - copy file1 to file2

#include <stdio.h>
#include <errno.h>

#define BUF_SIZE 256

int main( int argc, char *argv[] )
{
    /*
     * Open file objects are identified by pointers to FILE structures. NULL
     * indicates an invalid value. The pointers are, in effect, a form of
     * "handle" to the open file object.
     */
    FILE *inFile, *outFile;

    char rec[ BUF_SIZE ];

    size_t bytesIn, bytesOut;

    if(argc != 3)
    {
        printf( "Usage: cp file1 file2\n" );
        return( 1 );
    }

    /*
     * The call to fopen specifies whether the file is to be treated as a text
     * file or a binary file. Text files contain system-specific character
     * sequences to indicate situations such as an end of line. On may systems
     * I/O operations on a text file convert between the end-of-line character
     * sequence and the null character that C interprets as the end of string.
     * Here both files are opened in binary mode.
     */
    inFile = fopen( argv[1], "rb" );
    if( inFile == NULL )
    {
        /*
         * Error are diagnosed with perror, which, in turn, accesses the global
         * variable errno to obtain information about the function call failure.
         * Alternatively, theferror function will return an error code that is
         * associated with the FILE rather than the system.
         */
        perror( argv[1] );
        return( 2 );
    }

    outFile = fopen( argv[2], "wb" );
    if( outFile == NULL )
    {
        perror( argv[2] );
        return( 3 );
    }

    /*
     * The fread and fwrite functions directly return the number of bytes
     * processed, rather than return the value in an argument, and this
     * arrangement is essential to the program logic. A successful read is
     * indicated by a non-negative value, and 0 indicates an end of file.
     */

    // Process the input file a record at a time
    while( ( bytesIn = fread( rec, 1, BUF_SIZE, inFile) ) > 0 )
    {
        bytesOut = fwrite( rec, 1, bytesIn, outFile );
        if( bytesOut != bytesIn )
        {
            perror( "Fatal write error." );
            return( 4 );
        }
    }

    fclose( inFile );
    fclose( outFile );
}

```

```
return( 0 );  
}
```