CS-3411 Program II : Kernel and Block I/O

Fall 2023

In this assignment, you will develop a program which uses kernel I/O routines and learn about input output.

You are asked to develop a program that will copy the contents of the file given by the program's first argument to a new file given by the second argument. Doing so, the program should read the data in blocks of a given *blocksize*, compute a 32-bit checksum of the block, print this number as a hexadecimal number and write the block to the output file. The checksum is a running XOR of the block data where four consecutive bytes are treated as an unsigned integer (i.e. cast the data to an unsigned int) and xor'ed to the sum. The program should read and write the files one-block at a time. Each printed checksum should be written using the format %08x and a single space should separate them. No other text should prepend the checksum. Example: "80182946 73625183 16481094 36458174 47291098 ..."

Requirements:

1. Usage:

   ```
   copy <infile> <outfile> <blocksize>
   ```

   where the blocksize is the buffer size used in I/O operations, given in bytes. If the argument is skipped, assume a block size of 1024 bytes. The blocksize must be a multiple of four. If not, issue a warning message to stderr (i.e., descriptor 2), and round it up to the next four-byte boundary.

2. In this project you do not need to verify if the given file names are legitimate. However, your program should work correctly when correct inputs are provided. Minimally, your program should check for the number of arguments and inform the user when a kernel call fails with an appropriate error message. Do not attempt to copy the permissions of the input file to the output file. Instead, use a 0600 mask.

3. Your program should not contain any standard I/O calls other than sprintf. You can use sprintf to format output messages into a buffer and then use a write kernel call to write it to stdout (i.e., descriptor 1) for the checksum and stderr (i.e, descriptor 2) for error/warning message(s). Read the man page for more on sprintf.

4. If there is a short read (i.e., less than 4 bytes) at the end of the file, you must zero pad the remaining bytes when calculating the checksum, but the files should be identical in the end. The zero padding should be to the right of the data. Ex: Given 2 bytes 0x4872, you would pad to make 0x48720000.

5. If any error is detected, your program should print an appropriate error message and exit.

6. Do not have any GCC warnings when compiling with -Wall and -Wextra.

Submission:

1. Your submission should include the source code written in C called copy.c, and a Makefile. When the Makefile is invoked, it should generate a binary called copy in the current directory.

2. Submit your two files through Canvas in a .zip file (e.g. "zip prog2.zip copy.c Makefile").

Recommendations:

1. Read the input file one block at a time into an unsigned integer array.
2. Always attempt to read the input file in multiples of four bytes.
3. Always check the result of kernel calls for success/failure.
4. Debug as you write your code. Test in small increments.
5. Test the program with a variety of files, including files which contain text data, binary data (such as its own a.out), as well as large files and an empty file.