Kernel I/O

CS3411 Fall 2018
Program Two
Due: Tuesday, Oct 9, 2018, AoE

In this assignment, you will develop a program which uses kernel I/O routines.

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, your program should compute a 32-bit checksum of the given file and **output this checksum as a hexadecimal number on the standard output.** The checksum is a running XOR of the input where four consecutive bytes are treated as an unsigned integer and xor'ed to the checksum. The program should read and write the files one-block at a time.

Requirements:

1. Usage:

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

   where blocksize is the buffer size used in I/O operations, given as bytes. If skipped, assume a block size of 64Kbytes. The blocksize must be a multiple of four. If not, issue a warning message and round it up to the next four-byte boundary.

2. Due to relatively short time assigned to the 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. Instead, use a 0600 mask.

3. Your program should not contain any standard i/o calls, except sprintf. You can use sprintf to format output messages into a buffer, but use write kernel call to write it.

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.

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

6. Write your checksum to stdout as a hexadecimal number

Submission:

1. You must submit an archive named prog2.tgz containing all of your project files. You can create the archive by executing the command: gtar czvf prog1.tgz * Note: this will archive everything in the current directory so please make sure the current directory only contains your project files. 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**. I will provide you with a script to verify that your submission will be graded.

2. Your submission should include a *statistics* file, called stats.txt. In this file, present the timing results for copying a file of 1KB, 1MB and 1GB for buffer sizes of 128, 256, 512, 1024, 2048, 4096, 65536 by comparing it with the Unix command *dd* with the same arguments. You can use the command *time* to determine how much time the copy operation took. You should review the manual pages for *time* and *dd*. Your stats file must follow the format:
   <file size> : <block size> : <copy time> : <dd time>

<file size> : <block size> : <copy time> : <dd time>
without the <>. The block size and time results should be values printed in ascii. Each row/line in stats.txt corresponds to a different test. There should be 1 line for every block size/file size tested.

Recommendations:

1. Read the input file one block at a time into an unsigned integer/character array.
2. Always attempt to read a multiple of four bytes.
3. Always check the result of kernel calls for success/failure.
4. Debug as you write your code.
5. Test the program with a variety of files, including files which are large, binary (such as a.out) as well as an empty file.
6. After copying the file using your program, use the Unix command *diff* to see if the source file and the copied file are identical.

Use Canvas to submit the archive prog2.tgz.