Keawa Rozet

916796184

CSC 415

Project 2 - Basic Programming Skills Review

Command to build the project:

=============================

Method 1:

make -f makefile

Method 2:

gcc -o fc ./filecopy.c

Command to run the project:

===========================

./fc

Follow the prompts.

Be sure to add any extra files to project location for testing.

What my code does:

==================

My code is organized in the manner so that main() makes calls to other methods to do all the heavy work. First, prompts to the user are sent to STDOUT using write(). Next the file locations are stored in arrays passed by pointers to the setUserInput() method. In setUserInput() the read() method is used to store the user's input into an array.

copyFromFileToFile() takes the points to the file paths provided by the user and calls the open() method to access the files. Before 'from' and 'to' can be accessed though, the strings need to have the trailing '\n' removed by the correctFilePath() method.Then 'from' is passed into open() and fd\_input catches the fd to the now open file.

First 'to' is checked if the file already exists in the path or not. If it already exists, then the copyFromFileToFile() returns -1, which gets passed into printSuccess() to signal open() failed. If the file doesn't already exist, then open() is called again but this time creates the 'to' file and the fd is stored in fd\_output.

Next the copying occurs in the while loop. This loop runs for as long as read() still has bytes to read. If read() returns -1 then there are no more bytes to be read from the file in fd\_input and the loop ends.

I used a while loop because my buffer only holds 27 bytes, which is not enough to store the entirety of my test files mediumtest.txt and largetest.txt. The while loop reads the fd\_input until the buffer is full. Then the write() function is called. write() takes the buffer and writes as many bytes to fd\_output as fd\_input read, which was stored in bytes\_input.

A counter is incremented by bytes\_output to keep track of how many bytes were written in total. The while loop ends, fd\_input and fd\_output close, and the counter is returned. printSuccess() in main takes this value and outputs it to the user using write() for the final message.

Example of debugging text:

==========================

Welcome to the File Copy Program by Keawa Rozet!

Enter the name of the file to copy from:

mediumtest.txt

Debug read log ---> mediumtest.txt

Enter the name of the file to copy to:

med.txt

Debug read log ---> med.txt

Debug read log ---> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus condimentum sagittis lacus, laoreet luctus ligula laoreet ut. Vestibulum ullamcorper accumsan velit vel vehicula. Proin tempor lacus arcu. Nunc at elit

Debug write log ---> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus condimentum sagittis lacus, laoreet luctus ligula laoreet ut. Vestibulum ullamcorper accumsan velit vel vehicula. Proin tempor lacus arcu. Nunc at elit

Debug read log ---> condimentum, semper nisi et, condimentum mi. In venenatis blandit nibh at sollicitudin. Vestibulum dapibus mauris at orci maximus pellentesque. Nullam id elementum ipsum. Suspendisse cursus lobortis viverra. Proin e

Debug write log ---> condimentum, semper nisi et, condimentum mi. In venenatis blandit nibh at sollicitudin. Vestibulum dapibus mauris at orci maximus pellentesque. Nullam id elementum ipsum. Suspendisse cursus lobortis viverra. Proin e

Debug read log ---> t erat at mauris tincidunt porttitor vitae ac dui. Donec vulputate l

Debug write log ---> t erat at mauris tincidunt porttitor vitae ac dui. Donec vulputate l

Debug synopsis log ---> fd\_input: 3 fd\_output: 4 total bytes output: 501

File Copy Successful, 501 bytes copied

strace -c terminal:

===================

> strace -o trace2.txt -c ./fc

Welcome to the File Copy Program by Keawa Rozet!

Enter the name of the file to copy from:

smalltest.txt

Enter the name of the file to copy to:

sm.txt

File Copy Successful, 25 bytes copied

trace2.txt output annotated (annotations denoted by "<---"):

============================================================

% time seconds usecs/call calls errors syscall

------ ----------- ----------- --------- --------- ----------------

0.00 0.000000 0 5 read <--- 2 calls by setUserInput(), 1 call by copyFromFileToFile(), 2 calls made before my written code

0.00 0.000000 0 5 write <--- 3 calls by printWriteToScreen(), 1 call by copyFromFileToFile(), 1 call from printSuccess()

0.00 0.000000 0 4 close <--- 2 calls by copyFromFileToFile(), 2 calls made before my written code

0.00 0.000000 0 2 fstat <--- 2 calls made before my written code

0.00 0.000000 0 5 mmap <--- 5 calls made before my written code

0.00 0.000000 0 4 mprotect <--- 4 calls made before my written code

0.00 0.000000 0 1 munmap <--- 1 call made before my written code

0.00 0.000000 0 1 brk <--- 1 call made before my written code

0.00 0.000000 0 1 1 access <--- 1 call made before my written code

0.00 0.000000 0 1 execve <--- 1 call made before my written code

0.00 0.000000 0 1 arch\_prctl <--- 1 call made before my written code

0.00 0.000000 0 5 1 openat <--- 3 calls made in copyFromFileToFile(), 1 error in copyFromFileToFile() to check if file exists, 2 calls made before my written code

------ ----------- ----------- --------- --------- ----------------

100.00 0.000000 35 2 total

strace terminal:

===============

> strace -o trace.txt ./fc

Welcome to the File Copy Program by Keawa Rozet!

Enter the name of the file to copy from:

smalltest.txt

Enter the name of the file to copy to:

sm.txt

File Copy Successful, 25 bytes copied

trace.txt output annotated (annotations denoted by "<---"):

===========================================================

execve("./fc", ["./fc"], 0x7ffdee101710 /\* 32 vars \*/) = 0 <--- program executed successfully

brk(NULL) = 0x561561962000 <--- memory address of some pointer

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory) <--- checks if the file exists and has read access = -1

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3 <--- opens file at location with read only or close-on-exec and assigns fd = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=128333, ...}) = 0 <--- assigns the above file with permissions 644 and size 128.333KB

mmap(NULL, 128333, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fa0ec80c000 <--- allocates 128KB at location NULL, allows read and write, all at fd = 3

close(3) = 0 <--- closes fd = 3, and frees 3

openat(AT\_FDCWD, "/usr/lib/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3 <--- opens file at location with read only or close-on-exec and assigns fd = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`\20\2\0\0\0\0\0"..., 832) = 832 <--- read 832 bytes from fd = 3 and store in a buffer

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2065784, ...}) = 0 <--- fd = 3 is a regular file with permission 755 and size 2.065MB

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fa0ec80a000 <--- allocate 8.9KB at address(NULL) with fd = -1

mmap(NULL, 3893488, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fa0ec251000 <---allocate 3.89MB at address(NULL)

mprotect(0x7fa0ec3ff000, 2093056, PROT\_NONE) = 0 <--- set no protection to address provided

mmap(0x7fa0ec5fe000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ad000) = 0x7fa0ec5fe000 <--- allocate 24.6KB for fd = 3 with read and execute privileges

mmap(0x7fa0ec604000, 14576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fa0ec604000 <--- allocate 14.6KB for fd = -1 with read and execute privileges

close(3) = 0 <--- closes fd = 3, and frees 3

arch\_prctl(ARCH\_SET\_FS, 0x7fa0ec80b4c0) = 0 <--- set 64 bit base for FS at address provided

mprotect(0x7fa0ec5fe000, 16384, PROT\_READ) = 0 <--- set read only protection for 16KB of memory at address provided

mprotect(0x5615618de000, 4096, PROT\_READ) = 0 <--- set read only protection for 40KB of memory at address provided

mprotect(0x7fa0ec82c000, 4096, PROT\_READ) = 0 <--- set read only protection for 40KB of memory at address provided

munmap(0x7fa0ec80c000, 128333) = 0 <--- release 128KB of memory for address provided

write(1, "Welcome to the File Copy Program"..., 49) = 49 <--- writes 49 bytes from buffer to fd = 1

write(1, "Enter the name of the file to co"..., 41) = 41 <--- writes 41 bytes from buffer to fd = 1

read(1, "smalltest.txt\n", 256) = 14 <--- reads 14 bytes from buffer to fd = 1

write(1, "Enter the name of the file to co"..., 39) = 39 <--- writes 39 bytes from buffer to fd = 1

read(1, "sm.txt\n", 256) = 7 <--- reads 7 bytes from buffer to fd = 1

openat(AT\_FDCWD, "smalltest.txt", O\_RDONLY) = 3 <--- opens file at location with read only privileges at fd = 3

openat(AT\_FDCWD, "sm.txt", O\_ACCMODE) = -1 ENOENT (No such file or directory) <--- attempts to open file at location but failes with ENOENT and fd = -1

openat(AT\_FDCWD, "sm.txt", O\_WRONLY|O\_CREAT, 0644) = 4 <--- opens file at location with create and write only privileges at fd = 4

read(3, "TEST TEST TEST TEST TEST\n", 216) = 25 <--- reads 25 bytes from buffer at fd = 3

write(4, "TEST TEST TEST TEST TEST\n", 25) = 25 <--- writes 25 bytes from buffer at fd = 4

read(3, "", 216) = 0 <--- reads 0 bytes from buffer to fd = 3

close(3) = 0 <--- closes fd = 3, and frees 3

close(4) = 0 <--- closes fd = 4, and frees 4

write(1, "File Copy Successful, 25 bytes c"..., 38) = 38 <--- writes 38 bytes from buffer to fd = 1

exit\_group(0) = ? <--- exits all threads in a process

+++ exited with 0 +++