Assignment 1

Operating Systems | Anthony Nguyen

a1_script.txt = sample run | **systcalls.txt** = system calls | **copyfile.c** = source code

Summary of Implementation:

The implementation was done in C. Below is a quick summary of the implementation.
Note: Can view the source code attached if you want to see in further detail.

1. Grabs file name from user input (stdin) and stores into variable.
2. Opens input file for reading using file pointer. If file cannot be opened or does not exist, then exit the program.
3. Checks if destination file exists and if it does then exit the program.
4. Create destination file and open it for writing (set file pointer for writing).
5. Read a single character from input file and put it into target file using the two file pointers attained above. This process is looped until the file pointer that is reading the input file reaches the end of file.

Implementation Details:

```
15  int main()
16  {
17    // Variable Declaration
18    char src_file[SIZE], dest_file[SIZE];
19    char c;
20    FILE *fp_src, *fp_des;
21
22    // Get souce file name
23    printf("Enter input file name: ");
24    fgets(src_file, SIZE, stdin); // fgets() I
25    RemoveNL(src_file);           // Removes t
26
27    // Get destination file name
28    printf("Enter destination file name: ");
29    fgets(dest_file, SIZE, stdin);
30    RemoveNL(dest_file);
```

(1)

Grabbed user input (from stdin) for source and destination file names and stored it into a char array variable (string) with SIZE = 100.
Note: RemoveNL is my own defined function (explained below).

```
73  // Removes trailing newline caused by fgets
74  void RemoveNL(char *string)
75  {
76    char *ptr;
77    if ((ptr = strchr(string, '\n')) != NULL)
78      *ptr = '\0';        // dereferencing ptr
79  }
```

(2)

RemoveNL() is a function I defined that removes the newline ('\n') from the end of the char array that stores users input (Files names don't have '\n')

```
33  if ( (fp_src = fopen(src_file, "r")) )
34  {
35    printf("\n<%s> succesfully opened\n", src_file);
```

(3a)

fopen() takes the user inputed value "src_file" and tries to open the file for reading. If successful it returns the file pointer and stores it into "fp_src" and the if statement passes which implies that the file exist. However if the file cannot be found because it does not exist or is not able to be opened, then fopen() returns NULL and the if statement fails.

```
64  else
65  {
66    printf("\n<%s> was not found... exiting\n", src_file);
67    exit(EXIT_FAILURE);
68  }
```

(3b)

If the if statement (3a) fails then this else statement is ran.
It prints out to the screen stating that the user inputed file name for the file source does not exist and the program exits with a EXIT_FAILURE.

```
37  // Checks if destination file exists
38  if ( access(dest_file, F_OK) == 0 )
39  {
40    // abort
41    printf("\n<%s> already exists... exiting\n", dest_file);
42    fclose(fp_src);
43    exit(EXIT_FAILURE);
44  }
```

(4)

If the source file is successfully opened in (3a) for reading then the next part is to check the destination file. This code checks if the user input for the destination file exists using access(). If it exists, then the file pointer used to read the file source is closed and the program is exited with EXIT_FAILURE.

```
46      // creates destination file for writing..
47      if ( (fp_des = fopen(dest_file, "w")) )
48      {
49        printf("<%s> succesfully created\n", dest_file);
50
51        // src -> dest character by character.
52        while((c = fgetc(fp_src)) != EOF)
53          fputc(c, fp_des);
54        printf("\n<%s> contents succesfully copied into <%s>\n", src_file, dest_file);
55      }
```

(5a)

If the destination file does not exist (4) then the destination file name given by user input will be created with fopen() for writing. If the file is successfully created for writing than there is a while loop on line 52 that reads a single character from the file souce using fgetc() and stores it into a variable "c". The character stored in "c" then is written into the destination folder in line 53 using fputc(). Line 52 and 53 repeats until fgetc() returns EOF which signifies the end of the source file (fp_src)

```
56      else
57      {
58        printf("Could not access or create <%s>\nexited...\n", dest_file);
59        fclose(fp_src);
60        exit(EXIT_FAILURE);
61      }
```

(5b)

From part (5a) if the if statement fails to create the destination file with the given name then a message it printed, the file pointer to source file is close, and the program exits with EXIT_FAILURE.

Note: only the file pointer to the source file needs to be closed before exiting since file pointer to the destination file failed (making it NULL)

```
62      fclose(fp_des);
63      fclose(fp_src);
```

(6)

Assuming the program has not exited from section (3b), (4), or (5b).

This section will run after the program successfully copies the files over and all that it does is close the file pointers to the source/destination files.

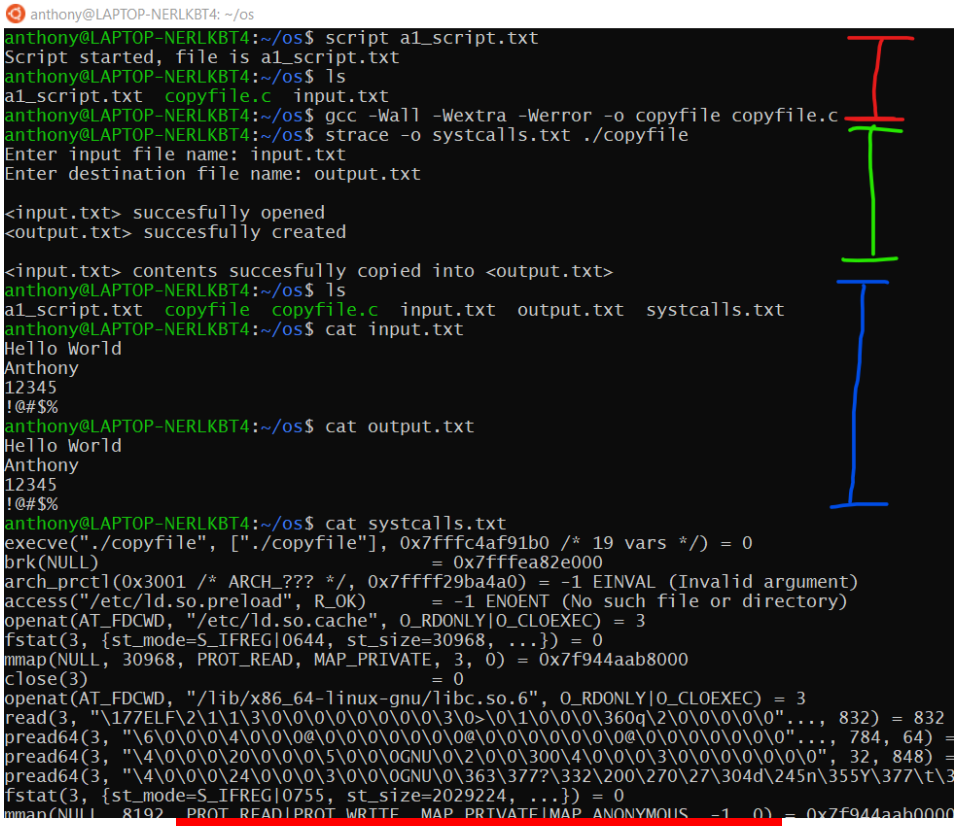
```
71      return 0;
```

(7)

If this line is reach (end of the program), then it signifies that the program was successfully completed. Operating systems normally take the exit code of "0" to be a successful termination.

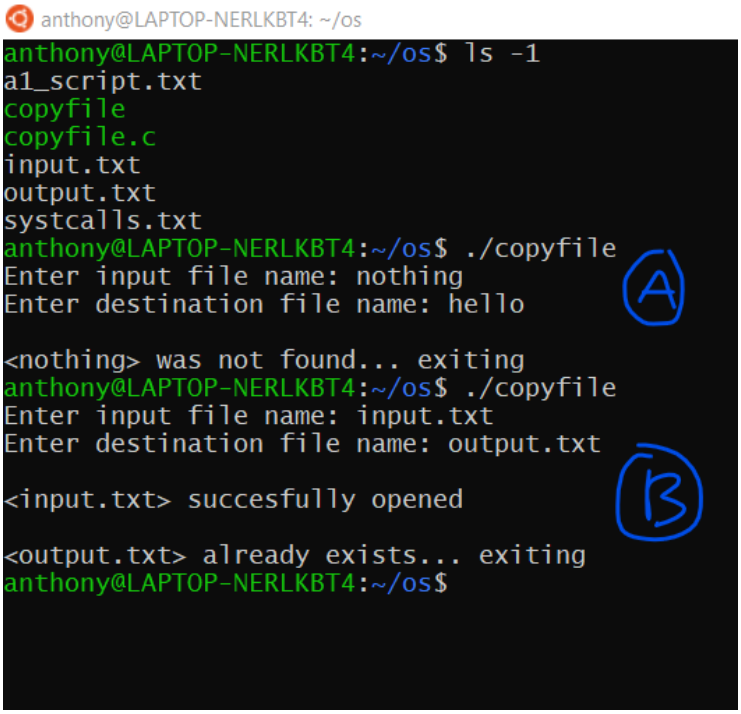
Sample Run (input/output):

"input.txt" is copied into "output.txt". Also, the system calls were saved into "systcalls.txt" file. Below is the sample run:



The rest is just printing out the contents of the system calls

Sample Run (with errors):



Just simple screenshot of showing
(A) - when no existing input file is used
(B) – when an existing output file is used

(System Calls on next page)

System Calls (strace):

```
execve("./copyfile", ["./copyfile"], 0x7fffc4af91b0 /* 19 vars */) = 0
brk(NULL)                               = 0x7fffea82e000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffff29ba4a0) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=30968, ...}) = 0
mmap(NULL, 30968, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f944aab8000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\0300\4\0\0\0\3\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f944aab0000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\0300\4\0\0\0\3\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f944a8b0000
mprotect(0x7f944a8d5000, 1847296, PROT_NONE) = 0
mmap(0x7f944a8d5000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) =
0x7f944a8d5000
mmap(0x7f944aa4d000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7f944aa4d000
mmap(0x7f944aa98000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) =
0x7f944aa98000
mmap(0x7f944aa9e000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f944aa9e000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f944aab1540) = 0
mprotect(0x7f944aa98000, 12288, PROT_READ) = 0
mprotect(0x7f944aaf3000, 4096, PROT_READ) = 0
mprotect(0x7f944aaed000, 4096, PROT_READ) = 0
munmap(0x7f944aab8000, 30968)           = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
brk(NULL)                               = 0x7fffea82e000
brk(0x7fffea84f000)                     = 0x7fffea84f000
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "Enter input file name: ", 23) = 23
read(0, "input.txt\n", 1024)            = 10
write(1, "Enter destination file name: ", 29) = 29
read(0, "output.txt\n", 1024)           = 11
openat(AT_FDCWD, "input.txt", O_RDONLY) = 3
write(1, "\n", 1)                       = 1
write(1, "<input.txt> succesfully opened\n", 31) = 31
access("output.txt", F_OK)              = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 4
write(1, "<output.txt> succesfully created"..., 33) = 33
fstat(3, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0
read(3, "Hello World\nAnthony\n12345\n!@#$%\n", 4096) = 32
fstat(4, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
read(3, "", 4096)                       = 0
write(1, "\n", 1)                       = 1
write(1, "<input.txt> contents succesfully"..., 58) = 58
write(4, "Hello World\nAnthony\n12345\n!@#$%\n", 32) = 32
close(4)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```