Operating Systems – Exercise 1

System Calls, Basic I/O

Submission & General Guidelines

- Submission deadline is **27/3/2019, 23:55** Moodle server time
- This exercise must run properly on the provided virtual machine
- Submit your answers, in the course website only, as a single file named ex1-YOUR_ID.zip (e.g. ex1-012345678.zip), containing only:
  - ex1.pdf
  - ex1.c
- Place your name and ID at the top of every source file, as well as in the PDF with the answers.
- No late submission will be accepted.
- Do not submit handwritten work.
- Please provide concise answers, but make sure to explain them.
- Write clean code (readable, documented, consistent, etc...)

Part 1 (36 points)

In this question we will implement a C application that copies a file. The application should receive the source file path, target file path and buffer size (in bytes) as command line arguments. By default, the application does not overwrite an existing file, unless the -f option was specified.

1. Read the manual page for the following System Calls i.e. execute: man 2 read
   a. READ(2)
   b. WRITE(2)
   c. OPEN(2), and Carefully read about the following option flags: O_RDONLY, O_WRONLY, O_CREAT, O_TRUNC, O_EXCL.
   d. CLOSE(2)
2. Read the manual page for the following C Library Calls
   a. PERROR(3)
   b. PRINTF(3)
   c. EXIT(3)
3. Complete the application in the provided ex1.c file (missing parts are marked with //TODO).

Execution output examples:

$ ./ex1
Invalid number of arguments
Usage:
ex1 [-f] SOURCE DEST BUFFER_SIZE

$ ./ex1 /etc/passwd /tmp/passwd
Invalid number of arguments
Usage:
ex1 [-f] SOURCE DEST BUFFER_SIZE

$ ./ex1 /etc/passwd /tmp/passwd 4096
File /etc/passwd was copied to /tmp/passwd

$ ./ex1 /etc/passwd /tmp/passwd 4096
Unable to open destination file for writing: File exists

$ ./ex1 -f /etc/passwd /tmp/passwd 4096
File /etc/passwd was copied to /tmp/passwd

$ ./ex1 -f /tmp/passwd /etc/passwd 4096
Unable to open destination file for writing: Permission denied

$ ./ex1 /etc/password /tmp/passwd 4096
Unable to open source file for reading: No such file or directory

Guidelines

- Use the manuals. Chapters 2 & 3 are your friends
- Make sure to always close the files you are using
- Always check system calls return value for errors. ALWAYS.
- You are not allowed to use any external / C-Library code that copies files, if you are not sure if you are allowed to use something, ask in the forum.
- Your program must finish executing with EXIT_SUCCESS only when there were no errors (otherwise it must finish executing with EXIT_FAILURE after printing a helpful message).
- Do not change function signatures, if provided.
- **Hint:** the 4 system calls and 3 libc calls mentioned above are all **you** need to implement this part of the exercise successfully.

Part 2 (24 points)

We will now examine the performance of our program from part 1.

1. Create a 5MB file using the following command:

```
dd if=/dev/zero of=/tmp/test.5mb bs=1M count=5
```

2. Run your program on this file, preceded with the **time(1)** command, using the following buffer sizes: 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536

```
$ time ./ex1 -f /tmp/test.5mb /tmp/test.5mb_dest 65536
File /tmp/test.5mb was copied to /tmp/test.5mb_dest

Real 0m0.004s
User 0m0.000s
Sys 0m0.000s
```

3. Draw a graph (using Google Sheets or Microsoft Excel) with a single series. The graph should show the **real** time each run takes (in milliseconds) [Y axis], as a function of the buffer size [X axis]
4. Explain the graph. Why isn't it a straight line, parallel to the X axis?
5. Hypothetically, if we change the **copy_file** in part 1 to print out to the screen a message each time we read **buffer_size** bytes, will the running time change significantly?
    - **IMPORTANT**: Do not make this change. Submission with this change will result in 0 points for this question!

Part 3 (40 points)

For each of the statements below, state whether **it is true or false and explain concisely** (two lines at most):

1. Some instructions are not executable in user mode.
2. System calls are executed by the operating system and therefore their running time is faster.
3. The JVM let users to run the same Java code on different OS. One of the tasks of the JVM is to translate the Java code to the specific system calls of the operating system.
4. Notepad is a built-in OS(Windows) program which is used in order to write and read from files. Therefore, it is executed in Kernel mode.
5. The following line will cause a transition from User mode to Kernel mode:

```
throw new RuntimeException();
```

6. The following line will cause a transition from User mode to Kernel mode:

```
int a = 0;
int b = 5/a;
```

7. An application can tell to a specific hardware device to stop sending interrupts.
8. Interrupts are sent to the process that currently running.
9. A programmer can decide whether to run a program in Kernel mode or not.
10. When clicking on the mouse button, while working on a virtual machine. Who will get an interrupt first? The guest OS or the host OS? (Answer guest or host).