Operating Systems – Exercise 1

System Calls, Basic I/O

Submission & General Guidelines

- Submission deadline is **12/4/2021, 23:55 (April 12th)** 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

  Your zip file should NOT contain any folders, just the two files
- 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 appends content from a file to another. The application should receive the buffer size (in bytes), source file path and target file path as command line arguments. By default, the application does not create a target file if such does not exist, 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_APPEND, 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] BUFFER_SIZE SOURCE DEST

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

$ ./ex1 4096 /etc/passwd /tmp/passwd
Unable to open destination file for writing: No such file or directory

$ ./ex1 -f 4096 /etc/passwd /tmp/passwd
Content from file /etc/passwd was successfully appended to /tmp/passwd

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

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

Output Requirements & Testing:

In order to apply a uniform testing procedure to your submissions, your program must output one of the following types of messages (precisely and case-sensitive):

- Unable to open source file for reading
- Unable to open destination file for writing
- Unable to append to destination file
- Unable to append buffer content to destination file
- Unable to read source file
- Unable to close source file
- Unable to close destination file
- Content from file <source_file> was successfully appended to <destination file>

Or one of the various arguments parsing errors, as described in the examples above.

Note that all console outputs are expected to end with a newline as shown in the examples above.

Please notice that the output of your program will be checked, as well as the contents of the file that was created during the append process, if such append takes place.

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.

Using Docker for Smoke Testing

- The VM is installed with Docker software (more on that later in the semester).
- When being graded, your solution will be tested in a container identical to the one that will be built using the supplied Dockerfile.
- There are two smoke tests (sanity checks) to this exercise. You can explore the 'EX1/check_submission/do_not_change' folder for more details
    - Test1 – tests a standard scenario: source file and destination file exist and not empty
    - Test2 - tests a scenario in which the destination file is missing (and there is no '-f' flag)
- Usage:
    - You're given EX1 folder, containing:
        - A 'check_submission' folder, which contains two tests (Test0, Test1) and a Python script to execute it (check_submission.pyc).
        - A skeleton file 'ex1.c'
    - Do not move ANY of the files inside check_submission, or you won't be able to use our tests.
    - You should edit 'ex1.c' with your solution.
    - **Each time** you want to test your final submission file (zip file):
        - Open a terminal and go the EX1/check_submission.
        - copy your submission file (ex1-YOUR_ID.zip) to your machine (anywhere you want)

- Execute 'python check_submission.pyc <full path to your submission file>'. for example:

```
student@student-VirtualBox:~/ex1/check_submission$ python check_submission.pyc /home/stude
nt/ex1/check_submission/ex1-123456789.zip
```

- If the two tests pass, you'll get:

```
student@student-VirtualBox:~/ex1/check_submission$ python check_submission.pyc /home/stude
nt/ex1/check_submission/ex1-123456789.zip
A docker is being built. It might take a minute or two if this is the first time..
Done. Smoke tests are running now..
Test0:Pass
Test1:Pass
```

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: 100, 200, 400, 800, 1600, 3200, 6400, 12800, 25600, 51200

```
$ time ./ex1 -f 51200 /tmp/test.5mb /tmp/test.5mb_dest
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 3 series: Real, User, Sys. The series should show the time each run takes (in milliseconds) [Y axis], as a function of the buffer size [X axis]
4. Explain the graph:
   - What is the meaning of Real, User and Sys (use Google)?

     Real - הזמן שלקח מרגע הפעלת הקריאה עד לסיומה

     User - הזמן שהמעבד (cpu) מבזבז מחוץ ל Kernel אבל בתוך התהליך

     Sys - הזמן שהמעבד (cpu) מבזבז ב-Kernel שבתוך התהליך
   - Why don't Sys and User sum up to Real?

     כאשר אנו מריצים תהליך מרגע ההפעלה ועד הסיום קורות התהליך נמצא במצבים שונים שאינם כוללים את הפעולות והמצבים שמתקיימים ב- User mode & Kernel. היות ו- User, Sys מתארים את הפעולות הספציפיות שמתקיימות ב- User & Kernel אזי סכום זמנים אלו לא יכול להיות שווה לזמן שלוקח כל התהליך.
   - Why isn't Real a straight line, parallel to the X axis?

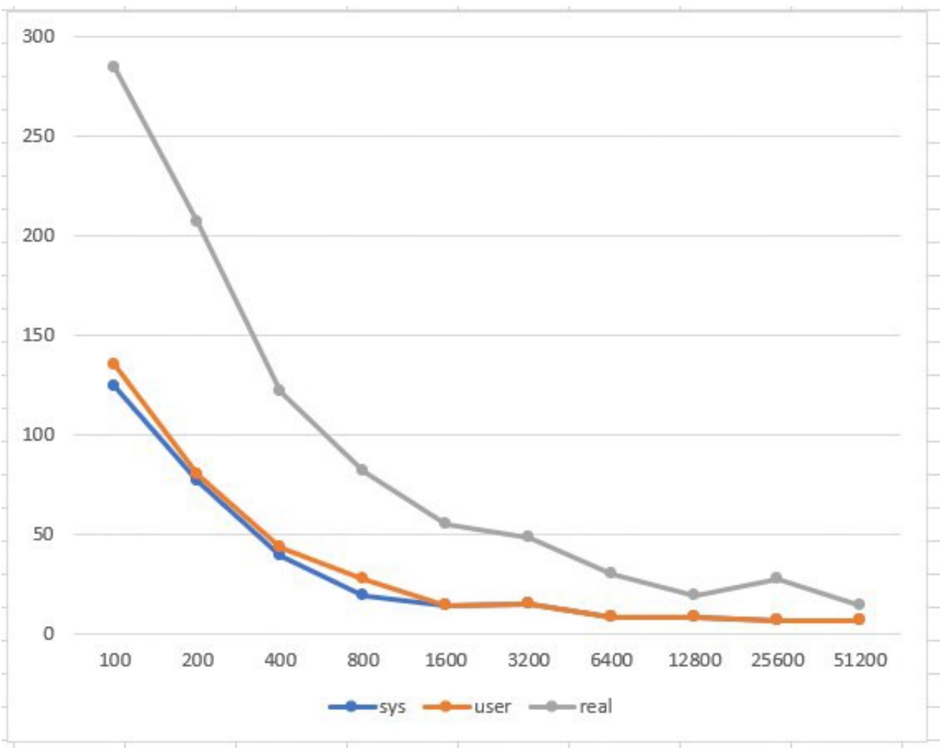
     מכיוון ש Real תלוי בהקשר אותו אנו מבצעים. יתכן ועבור קבצים מסויימים תהליך ההעתקה נעצר שתהליך מההפעלה למצב מספר שונה מצבים שאינו קשורים למעבד או לתהליך יתכן כי הפעולות שונות המעבד לתהליך דורשות זמן רב יותר עבור הקובץ לא מוכן.
5. Hypothetically, if we change the **append_file** in part 1 to print out to 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!

היות ואנו נדפיס למסך הדפסה של המסך, התהליך יצטרך להשתמש ב מערכות I/O. בדרך כלל שימוש ב-I/O דורש מהתהליך להיות במצב המתנה מכיוון לתהליכים שצריכים לקרות ב I/O ולכן אנו נשים לב לשינוי משמעותי בזמני ריצה שיביא להאטת הזמנים.

עידו מור - 208508259 - תרגיל 1 - מערכות הפעלה

שאלה 3:

sys	user	real	bufer
124	11	149	100
76	4	126	200
39	4	78	400
19	8	54	800
14	0	41	1600
15	0	33	3200
8	0	22	6400
8	0	11	12800
6	0	21	25600
6	0	8	51200



Part 3 (40 points)

Answer 'Homework 1 Quiz' in Moodle (10 T/F questions, 4 points each).