AMERICAN UNIVERSITY OF BEIRUT
Faculty of Arts and Sciences
Department of Computer Science
CMPS 240 – Operating Systems
Fall 2022 – 2023

Assignment #3
M. J. Khabbaz

Due Date: October 28, 2022 at 23:59 PM

A. Objectives:

The main objective of this assignment is to implement a direct inter-process communication mechanism using shared memory. The assignment focuses mainly on multi-process programming using `fork()`, `wait()` and so forth, as well as on data sharing among processes through address space modification using `shmget()`, `shmat()`, and other functions.

B. Worker/Reducer IPC Model:

You are to develop a new programming model called worker/reducer, which is a simplified version of MapReduce[i]. Your model will be used to concurrently count the number of occurrences of the strings "`CMPS`", "`CCE`" and "`ECE`" in a large file called input.txt consisting of 320,000 lines. Your model needs to be developed according to the following guidelines:

1. The main process takes a command line argument denoting the number of splits N. For example, if the number of splits is equal to 4, the file has to be split into 4 parts each consisting of 80,000 lines. If the executable of your program is stored in a file `WorkerReducer.o`, then to run your program from the CLI prompt and instruct it to create 4 splits, you will need to type `./WorkerReducer 4`. Then, the main process has to spawn a new process that does the split. It will then need to wait for the termination of this new child. In order to do the split, you may program your child process to execute the following command:

```
split -l 80000 -a 1 -d input.txt output
# split the file into several parts with 80000 lines each.
# i.e., if input.txt contains 320,000 lines it will be
# split into 4 parts being:
# output0, output1, output2, and output3.
```

2. The main process creates N shared memories SM_0, SM_1, ⋯, SM_{n-1} of size equal to $4 \times$ `sizeof(long)`.
3. The main process creates N worker processes and three reducer processes.

4. Worker process *i* counts the occurrences of "CMPS", "CCE", and "ECE" in split *i* (*i.e.*, output<i>). When done, it writes 999 (which is a special flag denoting the termination of the worker) and the occurrences of "CMPS", "CCE" and "ECE" in shared memory SM_i, respectively.
5. First (respectively second and third) reducer process sums the occurrence of "CMPS" (respectively "CCE" and "ECE") generated by all the worker processes. For this, it is required to keep checking if the first (respectively second and third) flag is equal to 999 before reading the corresponding occurrence value "CMPS" (respectively "CCE" and "ECE").

Note that it is absolutely required to avoid any code redundancy! It is possible to create a modular program for the worker that takes as CLI arguments the index of the split and the identifier of the shared memory SM_i. Consequently, when the main process spawns the worker processes, it modifies their code to execute the modular program of the worker by passing the corresponding parameters (*i.e.*, index of the split and the identifier of the shared memory). Similarly, code redundancy must also be avoided for reducers.

C. Submission:

Submit via Moodle a compressed file called cmps-240.a2.USER-ID.tar.bz2 (or .zip), where USER-ID is your user-id. The compressed file is an archive that contains:

1. An IPC folder that contains:
    a. A Makefile with the appropriate targets.
    b. The source and the header files (if any).
2. A document containing:
    a. A description of difficulties encountered throughout this assignment.
    b. In case of group work, a time-table describing the work done by each of the students. (Note that a group can only be formed of a maximum of two students).

[i] https://en.wikipedia.org/wiki/MapReduce