بسم الله الرحمن الرحيم

An-Najah National University

Faculty of Engineering and Information Technology

Computer Engineering Department

جامعة النجاح الوطنية

كلية الهندسة وتكنولوجيا المعلومات

قسم هندسة الحاسوب

Operating Systems

Assignment #2: 2nd semester 2025

Date: 20th /March/2025

Due date: 29th/March/2025

Assignment weight: 5 %

Assignment Details: The goal is to demonstrate the performance of various Inter Process Communication (IPC) techniques and understand the difference between multi-process and multi-thread programming.

We have two large arrays of **N** items each, called `packet1` and `packet2`, of type `double`. The arrays are declared and initialized in the parent process as following:

- Initialize `packet1` with random double values between **0** and **N**:

```
packet1 = (double)rand()/(double)(RAND_MAX/N);
```

- Initialize `packet2` with random double values between **1** and **10**:

```
packet2 = 1.0 + (double)rand()/(double)(RAND_MAX/9);
```

Part 1 (10%) — Serial computation

- Compute the following serially and store the results in `result_packet1`:

```
for (int i = 0; i < N; i++)
    result_packet1[i] = pow(packet1[i], packet2[i]);
```

- result_packet1 is in the parent process and is not shared.
- Compile and run your program:

gcc serial_compute.c -o serial_compute.out
./serial_compute.out

- Measure the performance of this technique: take a timestamp at the start of the loop and another timestamp at the end of the loop and subtract the two. See (Page 4) for more details.

Part 2 (30%) — Parallel computation using fork() and binary files

- Create **M** processes using `fork()`, where:
  - **N** = your student ID number (use the last 6 digits)
  - **M** = (sum of student ID digits) mod 6 + 4
- Each child process computes `(N/M)` items and writes them into a separate binary file named using its PID.
- The parent process collects all partial results by reading those files and storing them in `result_packet2`.
- Verify correctness: `result_packet1` and `result_packet2` must be identical.
- Measure and display performance.
- Note that it would be faster and more accurate to open the file in binary mode and write 8 bytes for each double precision number. The file would not be readable in a text editor, but you should be able to read it by the parent process. Use fread and fwrite for binary read/write to file and NOT of fprintf/fscanf for reading/writing ASCII.

Part 3 (30%) — Parallel computation using fork() and shared memory

- Repeat the same procedure but using shared memory (POSIX `shm_open`) instead of file IO.
- Each child process should write its portion of the result directly to the shared memory array.
- The parent process waits for all children to finish using `wait()`.
- Verify correctness by comparing with `result_packet1`.
- Measure performance.

Part 4 (30%) — Parallel computation using pthreads

- Use **M** threads to perform the same calculation.
- Each thread writes directly to the result array `result_packet3`.

- After threads join, verify correctness by comparing with `result_packet1`.
- Measure performance.

Results and explanation

At the end of execution, print:

- The measured times for:
  - Serial computation
  - Fork with file IO
  - Fork with shared memory
  - Pthreads
- Provide a short explanation of your observations:
  - Why one approach is faster or slower.
  - Effects of memory access and IPC choice.
  - Impact of your choice of **N** and **M**.

Submission rules:

- All code must be in **c**.
- Submit working `.c` file(s) and a **1-2 page PDF report** explaining your work.
- Deadline is 29th /March/2025. Late submissions are allowed for up to 5 days with a 5% penalty per day.
- You may work individually or in teams of two. For teams, use `(StudentID1 + StudentID2)/2` to compute **N** and **M**.
- Optional: try to make your parallel version as fast as possible:
  - You are free to change the number of processes (M) as you see necessary.
  - Increase number of cores? What happens if you exceed the physical number of cores on your machine.
  - Reduce number of processes (M) to match number of cores?
  - Change how the packets are split between processes? Check the cache dimensions in the CPU.
- If you manage to improve the performance, then you should explain how.
- If you change your code/parameters to make the parallel version faster, then you must submit two versions: Original as specified in the assignment AND updated/optimized version.
- Assignment weight is 5% from the course's total mark.
- You can work in teams of ***two maximum***: Use (StudenID1+StudentID2)/2 as your source for M and N.

How to measure time taken in c?

- using the `clock()` function from the `time.h` library
- `clock()` returns the processor time consumed by the program.
- `CLOCKS_PER_SEC` is a constant that represents the number of clock ticks per second

```
cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
```