Lab 1

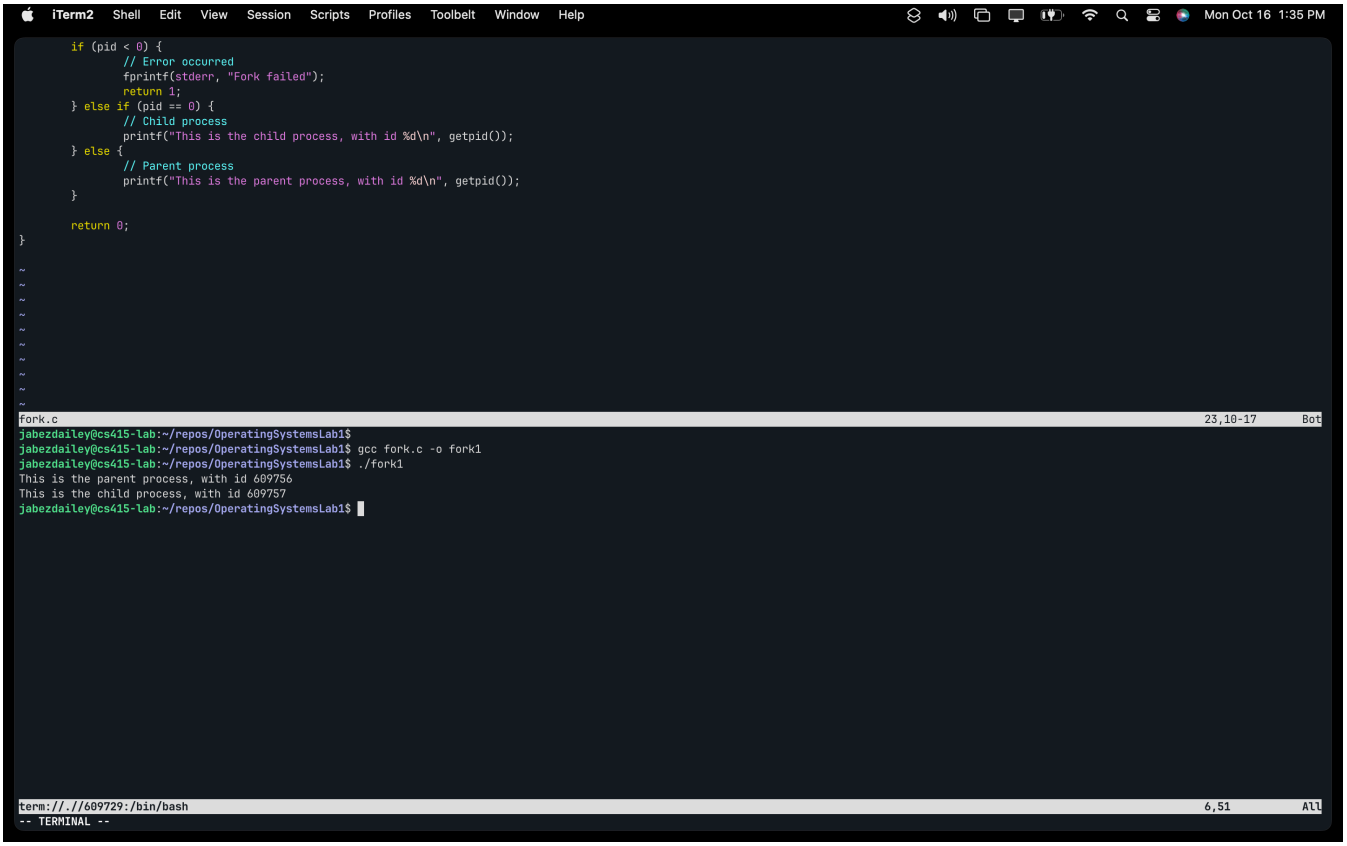
- Jabez Dailey

Questions

1. Give an example of using fork in C and explain your code and the output. (Need screenshots)
2. Exaplain how sleep() function can impact using fork. and exaplain your code and the output. (Need screenshots).
3. Demonstrate the difference between sleep and wait functions when used with fork. and explain your code and the output. (Need screenshots)

Answers

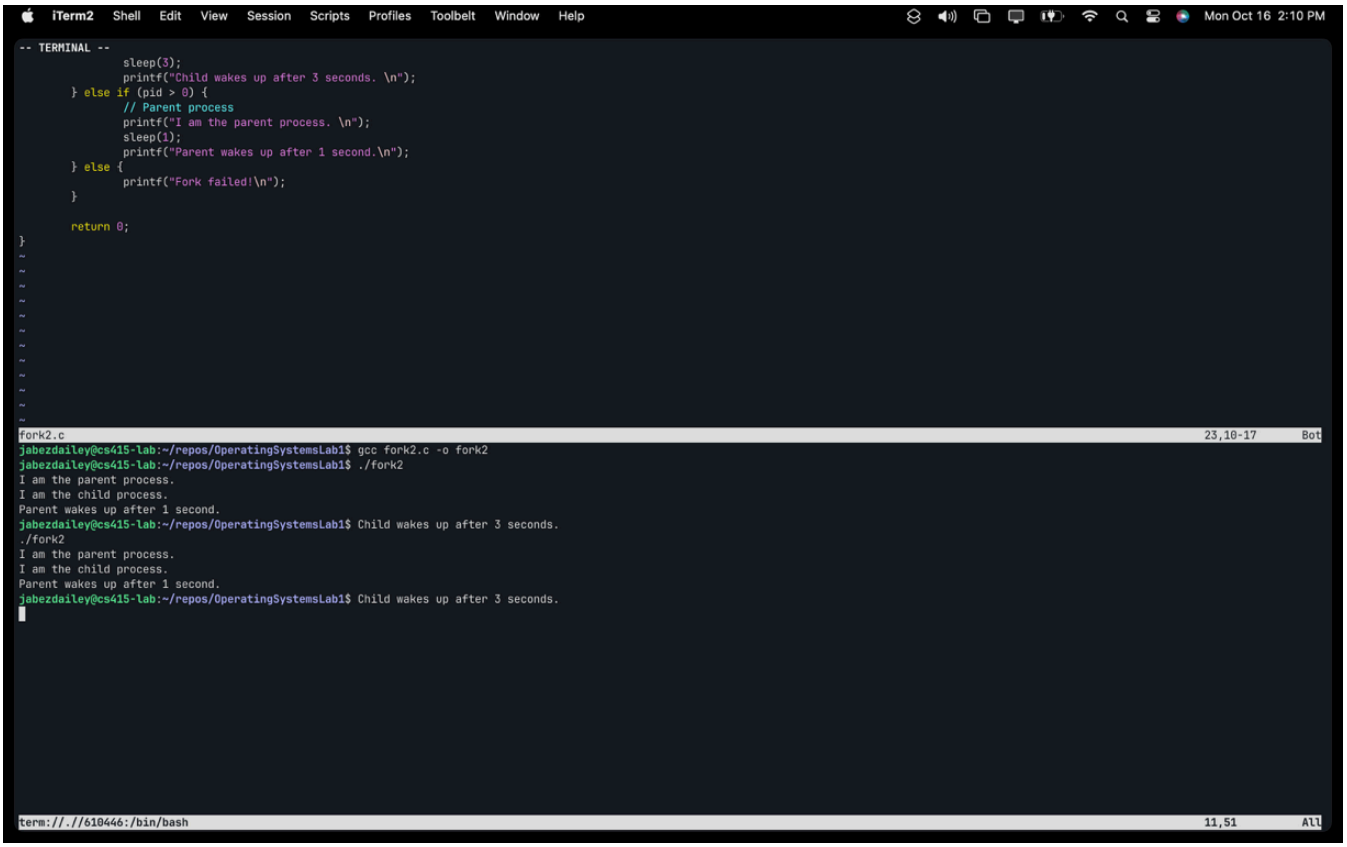
Question 1



Explanation

1. We first include necessary headers. `sys/types.h` , and `unistd.h` .
2. Inside the `main` function, we declare a variable `pid` of the type `pid_t` which will hold the value returned by the `fork()` function.
3. The `fork()` function is then called, which creates a new child process that is a copy of the current process.
4. There are three possible return values from the `fork()` function:
   - A negative value indicates that the fork failed.
   - Zero indicates that we are in the child process.
   - A positive value indicates that we are in the parent process and the returned value is the process ID of the child.
5. Based on these return values, we print messages:
   - If `pid` is less than 0, and error message is printed.
   - If `pid` is 0, we are in the child process, so we print a message with the child process ID.
   - If `pid` is positive, we are in the parent process, so we print a message with the parent process ID

Question 2



Explanation

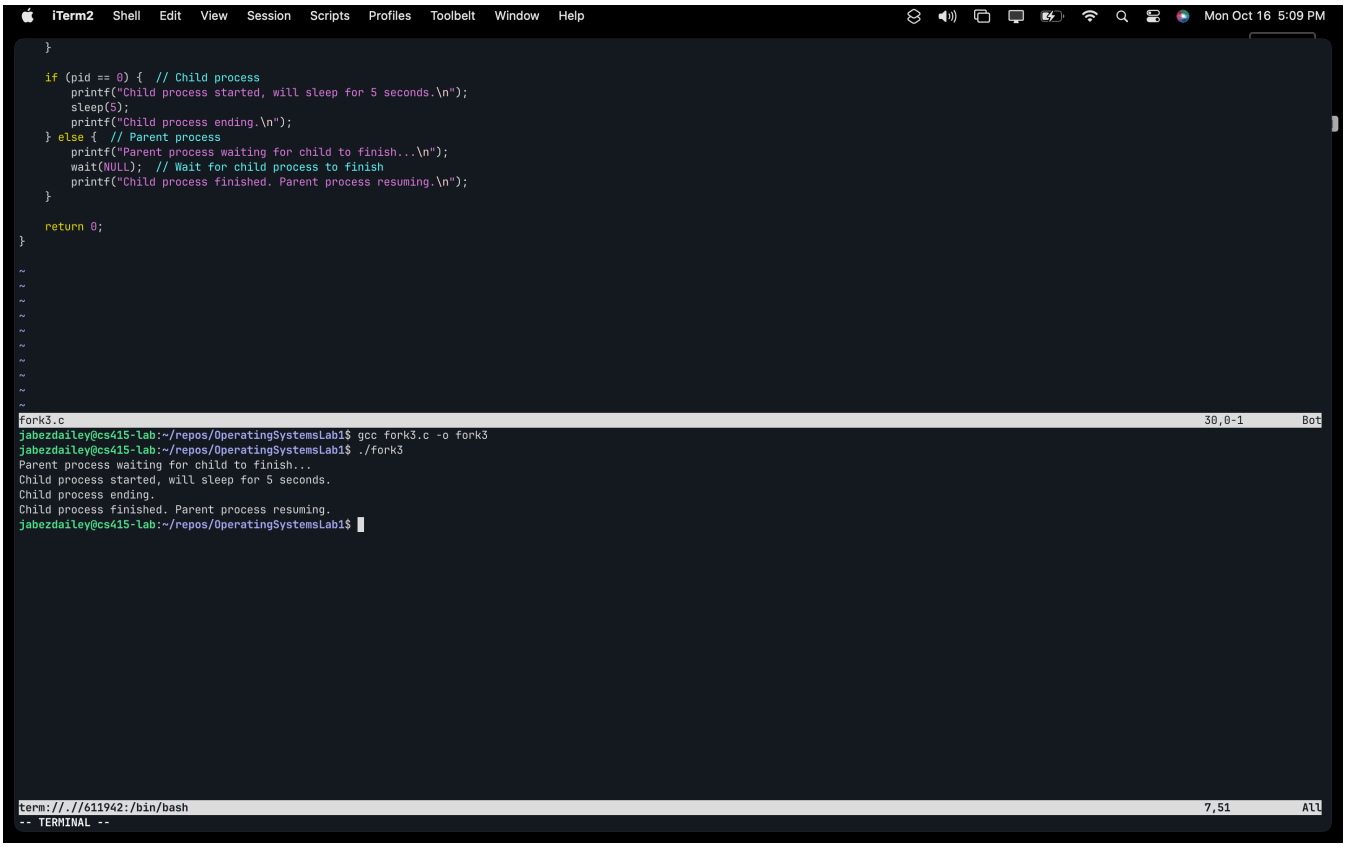
1. The main structure of this code is similar to the previous example, but we've added `sleep()` calls for both the child and the parent processes.
2. In the child process:
   - A message indicating the child process ID is printed.
   - The child process then sleeps for 3 seconds.
   - After waking up, a message is printed
3. In the parent process:
   - A message indicating the parent process ID is printed.
   - The parent process then sleeps for 1 seconds.
   - After waking up, a message is printed.

Impact of `sleep()`

- Without the `sleep()` function, both the parent and child processes would try to execute their respective print statements nearly simultaneously, leading to an unpredictable order of execution.

- By introducting the `sleep()` function:
  - We control the sequence of execution to an extent. The parent process sleeps for 1 seconds, so it will finish its execution before the child does (as the child sleeps for 3 seconds). This controlled delay ensures the “Parent process wakes up and finishes.” message is printed before the child’s wake-up message.

Question 3



Explanation

1. The program starts its execution in the parent process.
2. `fork()` is called, creating a child process.
3. The return value of `fork()` is stored in the `pid` variable.:
   - If `pid` is less than 0, an error occurred during the forking process, and the program will print “Fork failed.”
   - If `pid` is equal to 0, this code block is being executed in the child process. In this case:
     - If prints “Child process started, will sleep for 5 seconds.”
     - The child process then sleeps for 5 seconds.

        - After waking up, it prints “Child process ending.”
    - If `pid` is greater than 0, this code block is being executed in the parent process. In this case:
        - It prints “Parent process waiting for child to finish…”
        - The parent process then waits for the child process to finish using `wait(NULL)`.
        - Once the child process finishes, it prints “Child process finished. Parent process resuming.”