April 6th, 2024

Homework #2:

Create Two Different Processes and Implement IPC Communication

Implement a communication protocol between two processes using IPC. Follow these steps:

- Define an integer variable named “result”. Assign its value as zero and perform the following steps sequentially.

Parent Process:

- Create a program that takes an integer argument.
- Create two FIFOs (named pipes).
- Send an array of random numbers to the first FIFO and send a command (e.g., requesting a summation operation) to the second FIFO.
- Use the fork() system call to create two child processes and assign each to a FIFO.
- All child processes sleep for 10 seconds, execute their tasks, and then exit.
- Set a signal handler for SIGCHLD in the parent process to handle child process termination.
- Enter a loop, printing a message containing "proceeding" every two seconds.
- The signal handler should call waitpid() to reap the terminated child process, print out the process ID of the exited child, and increment a counter by two.
- When the counter reaches the number of children originally spawned, the program exits.

First Child Process (Child Process 1):

- Open the first FIFO and read random numbers to perform a summation operation.
- Write the result to the second FIFO.

Second Child Process (Child Process 2):

- Open the second FIFO and read the command to perform the specified operation (e.g., summation).
- Print the result to the screen.

Handle program errors and provide a message indicating whether the program has completed successfully or not using ***perror()*** methods.

Bonus Section:

1. ***Implement a zombie protection method to earn 15 points.***

2. ***Print the exit statuses of all processes for an additional 15 points.***

This assignment challenges you to use FIFOs for data communication between child processes and implement a complex IPC scenario.

Test Scenario:

Expected Results(In one scenario):

- The integer was correctly defined and assigned successfully.
- FIFOs were created successfully, and data/command transmission occurred without errors.
- The array was initialized with ten elements: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9].
- These values were read by the initial FIFO.
- In the second FIFO, the sum of all elements was calculated as 45.
- Child processes completed their tasks successfully and exited without errors.
- The parent process correctly managed the counter value and printed exit statuses for each child process.
- The printed value of the integer should be "45".

Error Scenarios:

- If FIFOs cannot be created or if there are errors in data/command transmission, appropriate error messages should be displayed.
- If child processes fail to complete their tasks successfully or encounter unexpected errors, error messages indicating the failure should be displayed.
- If the counter value is not managed correctly or if exit statuses are not printed for each child process, error messages should be displayed indicating the issue.

Grading Criteria:

1) No compilation: -100 points
2) Missing Makefile or Makefile without "make clean": -30 points
3) Each memory leak that we encounter: -20 points
4) No report submitted: -100 points (You must test and demonstrate every step.)
5) Failure in one of the tasks: -10 points
6) No error control block used or failure in throwing a message: -10 points
7) Creating processes without using the fork() system call: -20 points
8) If one of the FIFOs does not exist: -20 points

Late submissions will not be accepted.
Deadline: April 16th at 23:59
Good Luck!