Monday, May 27, 2024

Operating Systems: Lab Assignment 3

System call fork()

Objective:

Familiarize yourself with the `fork()` system call in the Linux operating system.

How to use `fork()`:

```
#include <unistd.h>
pid_t pid = fork();
```

Assignments:

1. **Basic fork() Usage:**
    - Write a program that calls `fork()`. Before calling `fork()`, declare an integer variable and assign it a value of 100. Answer the following questions:
        - What is the value of the variable `x` in the child process?
        - If both the parent and child processes modify the value of `x`, how does `x` change?

2. **File Descriptor Inheritance:**
    - Write a C program that uses the `open()` system call to open a file and then calls `fork()`. Answer the following questions:
        - Can both the parent and child processes access the file descriptor returned by `open()`?
        - What happens if both processes write to the file simultaneously? Do they write to the file concurrently?

3. **Standard Output Closure:**
    - Write a program that creates a child process. In the child process, close the standard output (STDOUT_FILENO). Then, test what happens when the child process calls `printf()` to print a string to the standard output.
    - **Hint:** To close the standard output, use `close(STDOUT_FILENO)` or `fclose(stdout)`. Note that `STDOUT_FILENO` is defined in `<unistd.h>` and represents unbuffered I/O, directly invoking system calls such as `write` and `close`. In contrast, `stdout` is standard I/O defined in `<stdio.h>`

4. ADVANCED PROCESS CONTROL:

- Extend the basic fork() program to create a chain of processes (parent, child, grandchild). Each process should print its process ID and its parent's process ID. Observe and document the order of execution and the relationship between the processes.
- **Hint:** Use `getpid()` and `getppid()` to retrieve the process IDs.

5. Pipes for Inter-process Communication:

- Write a program that creates a pipe using `pipe()`. The parent process should write a string to the pipe, and the child process should read this string from the pipe and print it.

- **Hint:** Use `pipe()`, `write()`, and `read()` system calls for pipe communication.