Inter-Process Communication (IPC) Demo Documentation

This program demonstrates inter-process communication using named pipes (FIFOs) between a parent process, two child processes, and a daemon process. The system calculates the larger of two numbers provided as command-line arguments and communicates the result through the processes.

Overview

The system consists of: 1. A **main parent process** that coordinates the workflow 2. **Child Process 1** that compares two numbers 3. **Child Process 2** that displays the result 4. A **daemon process** that logs system activity

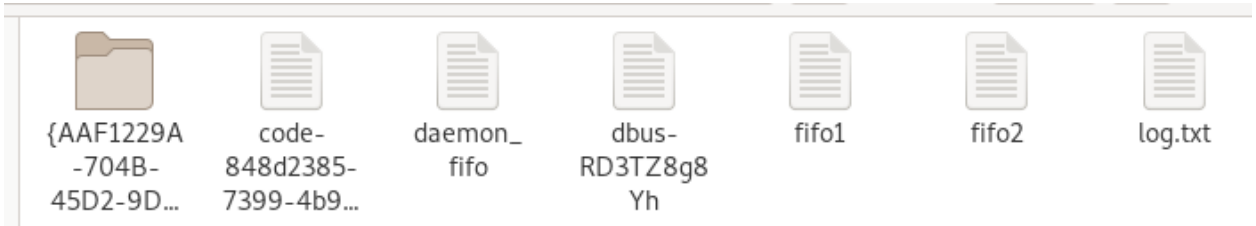
Key Components

FIFOs (Named Pipes)

- `/tmp/fifo1`: Communication between parent and Child Process 1
- `/tmp/fifo2`: Communication between Child Process 1 and Child Process 2
- `/tmp/daemon_fifo`: Communication channel for logging messages to the daemon

Log File

- `/tmp/log.txt`: Stores all logged messages from the daemon



Process Flow

1. **Initialization**:
   - Creates three named pipes if they don't exist
   - Launches the daemon process
   - Forks two child processes
2. **Data Flow**:
   - Parent writes two numbers to FIFO1
   - Child Process 1 reads numbers, finds the larger one, writes to FIFO2
   - Child Process 2 reads the result from FIFO2 and displays it
3. **Logging**:
   - All processes send status messages to the daemon via the daemon FIFO
   - Daemon writes messages to the log file with timestamps

4. **Cleanup**:
   - Processes terminate in sequence
   - FIFOs are unlinked when no longer needed

Functions

Main Functions

- `main()`: Entry point, validates arguments, creates pipes, launches processes
- `child1()`: Compares two numbers and finds the larger one
- `child2()`: Receives and displays the result
- `daemon_procces()`: Creates a logging daemon

Helper Functions

- `write_msg()`: Formats and writes messages to the daemon FIFO
- `close_all_fifo()`: Cleans up FIFO resources
- `deamon_handle_signal()`: Handles signals sent to the daemon
- `SIGCHLD_handler()`: Handles child process termination (not currently used in main flow)

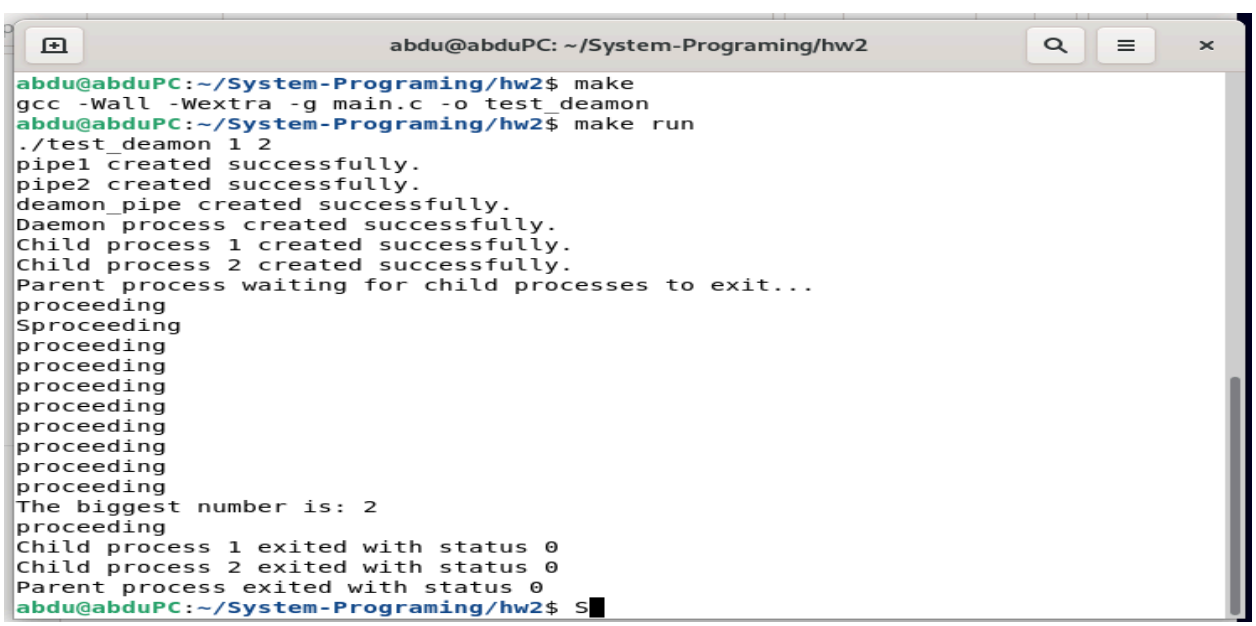
Usage

Compile and run the program with two integer arguments:

```
./test_deamon num1 num2
```

Example:

```
./test_deamon 1 2
```



```
abdu@abduPC:~/System-Programing/hw2$ cat /tmp/log.txt
[Tue Apr  8 18:10:12 2025] Main process ID: 3452
[Tue Apr  8 18:10:12 2025] Child process 1 ID: 3458
[Tue Apr  8 18:10:12 2025] Child process 2 ID: 3459
[Tue Apr  8 18:10:22 2025] Child process 2 is running...
[Tue Apr  8 18:10:22 2025] Child process 1 is running...
[Tue Apr  8 18:10:22 2025] Child process 2 finished successfully.
[Tue Apr  8 18:10:23 2025] Parent process finished successfully.
exitabdu@abduPC:~/System-Programing/hw2$
```

Error Handling

The program includes comprehensive error handling for: - Invalid arguments - FIFO creation/access failures - Process creation failures - Signal handling errors - Resource cleanup on failure

```
Parent process exited with status 0
abdu@abduPC:~/System-Programing/hw2$ ./test_deamon 1 aa
Error, Invalid argument.
abdu@abduPC:~/System-Programing/hw2$ S
```

```
pipe2 created successfully.
Note: deamon_pipe already exists, continuing...
deamon pipe created successfully.
```

Security Considerations

- Daemon process changes working directory to root
- File permissions are reset with umask(0)
- All unnecessary file descriptors are closed
- Standard I/O is redirected to /dev/null or log file

Limitations

- Fixed buffer sizes for messages
- Sleep-based synchronization between processes
- Limited error recovery capabilities

This documentation provides a high-level overview of the IPC system. For implementation details, refer to the inline comments in the source code.