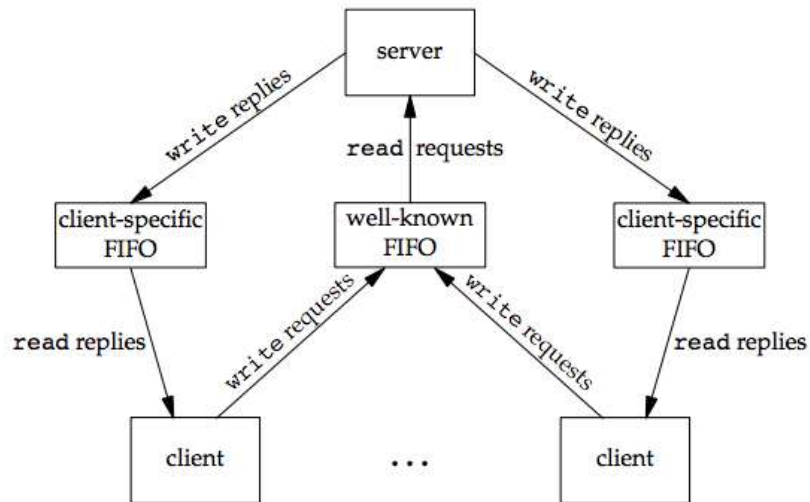


Tuba TOPRAK 161044116

System Programming Midterm Project



Title: Implementation of a Multi-Client Server using Forks, FIFOs, and Semaphores

Introduction:

The purpose of this report is to describe the implementation of a multi-client server using forks, FIFOs, and semaphores. The server allows multiple clients to connect simultaneously and perform various operations such as file reading, file writing, and listing files. The server handles client requests concurrently using a combination of forks for process creation, FIFOs for interprocess communication, and semaphores for synchronization.

Server Architecture:

The server architecture consists of the following components:

a. Parent Process:

The parent process acts as the main server process.

It creates a main FIFO for client-server communication.

It listens for client connections by continuously reading from the main FIFO.

Upon receiving a client request, it forks a child process to handle the request.

b. Child Process:

Each child process is responsible for handling a specific client's request.

It communicates with the client using a result FIFO for sending the response.

It performs the requested operation (e.g., file reading, file writing) and sends the result back to the client through the result FIFO.

After completing the request, the child process terminates.

c. FIFOs:

The main FIFO is used for client-server communication.

Each client is assigned a unique result FIFO for receiving the response from the server.

d. Semaphores:

Semaphores are used to synchronize access to shared resources.

A mutex semaphore is used to ensure exclusive access to critical sections of code, such as reading from/writing to files and updating shared data structures.

Additional semaphores can be used to manage resource allocation and concurrency control.

Client-Server Communication:

The client-server communication is achieved through FIFOs. The main FIFO is created by the server and is used by clients to send their requests to the server. Each client is assigned a unique result FIFO, which is created by the client and used by the server to send the response back to the client. The server listens for client connections by continuously reading from the main FIFO. Upon receiving a client request, it forks a child process and communicates with the client through the result FIFO.

Process Creation and Termination:

The server uses forks to create a child process for each connected client. Forking allows concurrent processing of client requests, as each child process operates independently. After completing the requested operation, the child process terminates, freeing up system resources.

Synchronization and Concurrency Control:

Semaphores are used for synchronization and concurrency control in the server. The mutex semaphore ensures exclusive access to critical sections of code, preventing race conditions and ensuring data integrity. For example, when a child process performs file operations or

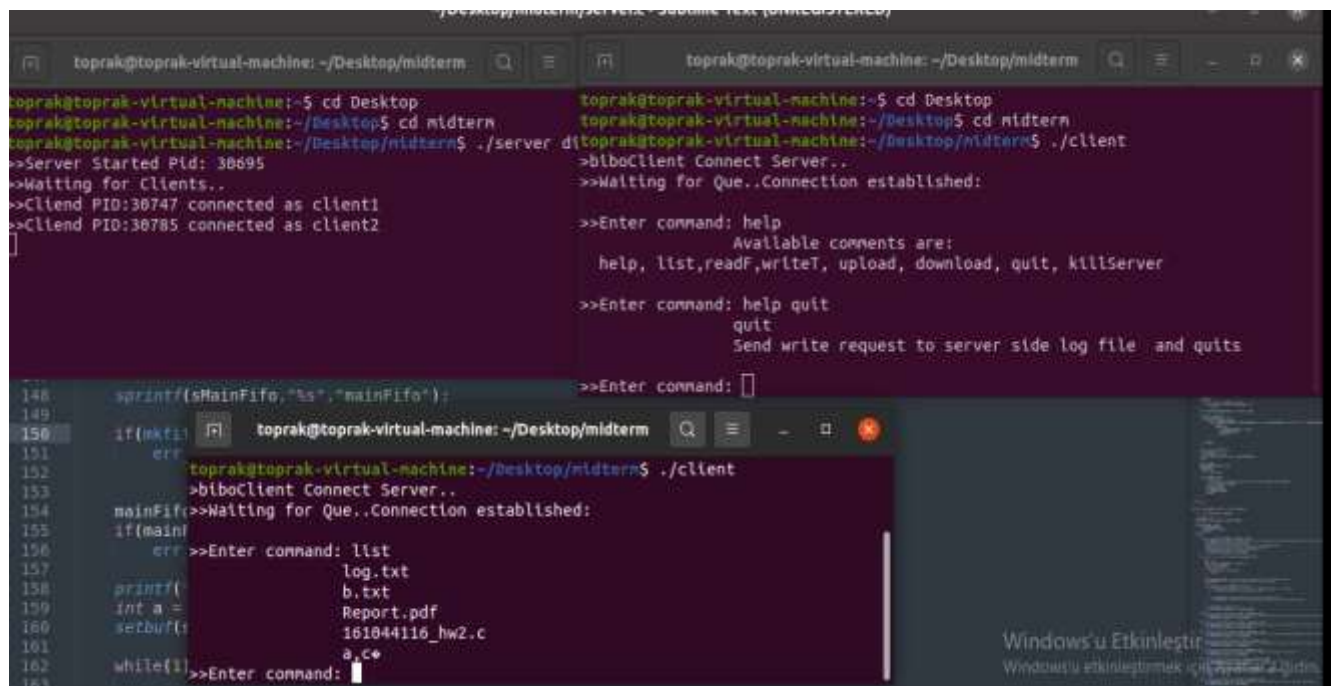
updates shared data structures, it acquires the mutex semaphore to prevent other processes from accessing the same resources concurrently.

Error Handling:

Proper error handling is implemented throughout the code to handle potential failures during process creation, FIFO creation, file operations, and interprocess communication. Error messages are displayed, and appropriate actions are taken, such as exiting the program or closing file descriptors.

Conclusion:

The implementation of a multi-client server using forks, FIFOs, and semaphores provides an efficient and scalable solution for handling concurrent client requests. The server architecture allows multiple clients to connect simultaneously and perform various operations concurrently. By using forks, FIFOs, and semaphores, the server achieves process isolation, interprocess communication, and synchronization of shared resources. The error handling mechanisms ensure robustness and reliability of the server.



```
toprak@toprak-virtual-machine: ~/Desktop/midterm
toprak@toprak-virtual-machine:~$ cd Desktop
toprak@toprak-virtual-machine:~/Desktop$ cd midterm
toprak@toprak-virtual-machine:~/Desktop/midterm$ ./server
>>Server Started Pid: 30695
>>Waiting for Clients..
>>Client PID:30747 connected as client1
>>Client PID:30785 connected as client2

toprak@toprak-virtual-machine:~/Desktop/midterm$ ./client
>>biboClient Connect Server..
>>Waiting for Que..Connection established:

>>Enter command: help
Available comments are:
help, list,readF,writeF, upload, download, quit, killServer

>>Enter command: help quit
quit
Send write request to server side log file and quits

>>Enter command:

148. sprintf(sMainFifo,"%s","mainFifo");
149.
150. if(!mkfifo(sMainFifo))
151.     err
152.     toprak@toprak-virtual-machine:~/Desktop/midterm$ ./client
153.     >>biboClient Connect Server..
154.     mainFifo>>Waiting for Que..Connection established:
155.     if(mainFifo)
156.         err >>Enter command: list
157.         log.txt
158.         printf("b.txt\n");
159.         int a = 161044116_hw2.c
160.         setbuf(stdout, NULL);
161.         a.c
162.         while(1)
163.             >>Enter command:

Windows'u etkinleştirin
Windows'u etkinleştirmek için yazılımınızın lisansını kontrol edin.
```