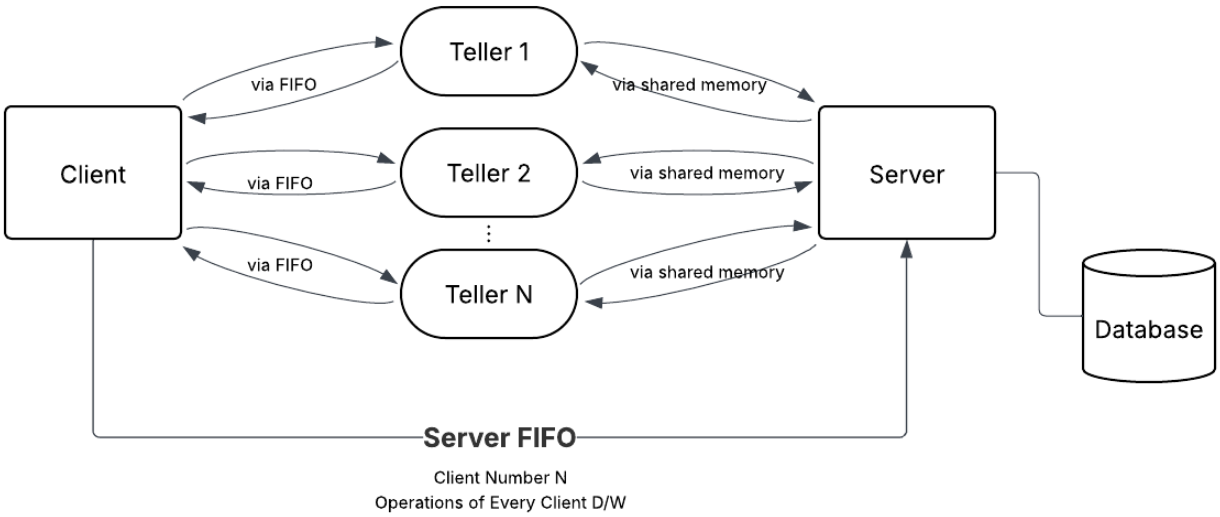
CSE344 – Midterm Project

Muhammet Talha Memişoğlu

210104004009

Introduction

The objective of this project is to implement a basic **banking server system** that manages multiple client operations using interprocess communication (IPC) mechanisms such as **FIFOs (named pipes)**, **shared memory**, and **semaphores**. The server handles multiple teller processes dynamically created for each client request, processes transactions like deposits and withdrawals, and maintains a persistent log of all bank accounts. The program was developed in C and tested on Debian 11 and 12. Additionally, the program is tested for memory leaks and the result was zero memory leak.



Server

Program Initialization

- The server program expects two command-line arguments: the bank name and the server FIFO name. If not provided, it exits with a usage message.

```
"Usage: %s <bank_name> <server_fifo_name>\n", argv[0]
```

Server Setup

- A server FIFO is created for client-server communication via initialize_server() function.

```
void initialize_server(const char* bank_name, const char* server_fifo_name);
```

Database Setup

- The bank database is initialized from a log file if it exists; otherwise, a new database is created.
- A semaphore (log_sem_id) is initialized to synchronize access to the log file during updates.

```
void initialize_database();
```

Signal Handling

- Program configures signal handlers for graceful shutdown(handling SIGINT and SIGTERM).
- If any teller isn't terminated yet, program waits with waitTeller() to shut down without any memory leaks.

Processing Client Connections

- The server enters a loop in process_client_connections() where it continuously listens for new client requests using select() on the server FIFO with a timeout until 'running' variable is false .
- When a client request is received, containing multiple sub-clients and their intended operations, the server spawns teller processes for each sub-client via create_teller()

```
for(int i = 0; i<client_req.client_count && i < MAX_CLIENTS; i++){
    create_teller(client_req.client_pid , (i + 1), client_req.operation[i]);
}
```

- The function initialize process_teller_requests() in order to process every request that comes from clients, via shared memory

```
//Process any teller requests(via shared memory)
process_teller_requests();
```

- In the end of the function, function waits for every teller.

```
int result = waitTeller(teller_pids[i], &status);
```

Creating Teller

- Each teller process is associated with:
  - A **shared memory** segment to exchange requests and responses with the server.
  - A **semaphore** to synchronize shared memory access.
- Shared memory and semaphore that is associated with teller is initialized in this function
- The teller is created based on operations. Teller takes operation that must do as input and process that operation.

```
pid_t teller_pid;
if(client_operation == 'D'){
    teller_pid = Teller((void*)deposit, args);
} else if(client_operation == 'W'){
    teller_pid = Teller((void *)withdraw, args);
} else {
```

Transaction Processing

- Supported operations:
  - **Deposit (D):** Adds funds to an existing or newly created bank account.
  - **Withdraw (W):** Removes funds from an existing account if sufficient balance is available.
- Transactions are read from FIFO that is created by Client.
- Transactions are validated for bank ID correctness, valid amounts, and account existence. If something is wrong, request_ready is set to -1 to inform the server.
- Transactions are copied to **shared memory** in order server to process via shared memory. This operation is protected by **semaphores**.
- It waits for a response that comes from the server and sends this response to the client.
- In the end of the function, shared memory and semaphore is cleaned up.

Processing Teller Request

- If any request is ready in the shared memory, transaction is processed via process_transaction().
- It sends responses to tellers based on return of process_transaction().

```
//Process the request
response->status = process_transaction(response);
```

- In process_transaction(), transaction is handled, and log file is updated after every successful transaction.

Updating Log File

- All accounts and their transactions are written to log file.
- If account balance is zero, is added to beginning of the line in order to close account.

Program Termination

- log_sem_id semaphore is deleted.
- Server FIFO is closed and deleted at the end of the program to prevent memory leaks.

Client

Program Initialization

- The client program expects two command-line arguments: the transaction file and the server FIFO name. If missing, it exits with an error message.
- Initializes global variables and sets up signal handlers for proper cleanup during interruptions.

```
"Usage: %s <client_file> <server_fifo_name>\n", argv[0])
```

Signal Handling

- Program configures signal handlers for graceful shutdown(handling SIGINT and SIGTERM).
- If any client process is running, it is shut down and waited

Reading Client File

- The program reads a transaction file containing operations for multiple clients.

- Each line of the file specifies:
  - Account ID
  - Operation (either deposit or withdraw)
  - Amount
- It checks whether the format is appropriate.
- Transactions are parsed and stored into a client's array.

Connecting to Server

- After reading all client transactions, the client connects to the server FIFO (server_fifo).
- It sends a connection request containing:
  - The client process's PID
  - The number of client sub-processes (N)
  - The operation types for each client (D/W)

Client Process Creation and Transaction Handling

- For each transaction, a child process is forked.
- Each child process independently:
  - Creates a unique FIFO (/tmp/clientX_fifo where X is client number)
  - Sends its transaction request to the server
  - Waits for a response from the server
  - Prints the result (success or error message)
  - Cleans up its FIFO before exiting

```
//Send transaction request to teller
write(client_fifo_fd, &client->transaction, sizeof(client->transaction));
```

Program Termination

- The parent process waits for all child client processes to complete.
- After all clients finish, the program releases allocated resources and exits gracefully.

Server-Client Communication Flow

- **Client → Server:** Sends number of client and their operation type through the server FIFO.
- **Client → Teller:** Sends transaction requests to teller through the client FIFO.
- **Teller → Server:**
  - Shared memory: Teller places the client's request into shared memory.
  - Semaphore: Signals the server to read and process the request.
- **Server → Teller (response):**
  - Shared memory: Server writes the response.
  - Semaphore: Signals the teller that the response is ready.
- **Teller → Client:** Sends the final transaction result back via the client FIFO.

Inputs and Outputs of Program

Clients Files

Midterm > Clients01.txt

```
1  BankID_None deposit 50
2  hjashfjashf withdraw 20
3  N deposit -20
4  N deposit 20
5  BankID_01 deposit 250
6  BankID_01 withdraw 150
7  N deposit 1000
8  BankID_02 deposit 2000
9  BankID_02 withdraw 200
10 BankID_03 withdraw 1000
11 N deposit 1000
12 N withdraw 1000
13 N deposit 500
14 BankID_03 deposit 1000
15 BankID_03 withdraw 2000
```

Midterm > Clients02.txt

```
1  N deposit 200
2  N deposit 5000
3  BankID_01 deposit 200
4  BankID_02 withdraw 500
5  N deposit 5000
6  N deposit 2000
7  BankID_05 deposit 400
```

Midterm > Clients03.txt

```
1  N deposit 200
2  N deposit 2000
3  BankID_01 deposit 200
4  BankID_02 withdraw 500
5  N deposit 1000
6  N deposit 2000
7
```

```
talhamem@Talha:~/projects/Midterm$ make
rm -f bankServer
gcc -Wall -Wextra -g -o bankServer bankServer.c
gcc -Wall -Wextra -g -o bankClient bankClient.c
./bankServer AdaBank ServerFIFO_Adabank0178
AdaBank is active...
No previous logs, creating the bank database...
Waiting for clients @ServerFIFO_Adabank0178...
Waiting for clients @ServerFIFO_Adabank0178...
- Received 15 clients from PIDClient114819..
-- Teller PID114932 is active serving Client01...
-- Teller PID114933 is active serving Client02...
Client01 deposited 50 credits. Operation not permitted.
-- Teller PID114934 is active serving Client03...
-- Teller PID114935 is active serving Client04...
Client02 withdraws 20 credits. Operation not permitted.
-- Teller PID114936 is active serving Client05...
Invalid amount in request from PID_Client114819_3 FIFO
-- Teller PID114937 is active serving Client06...
-- Teller PID114938 is active serving Client07...
-- Teller PID114939 is active serving Client08...
-- Teller PID114940 is active serving Client09...
-- Teller PID114941 is active serving Client10...
-- Teller PID114942 is active serving Client11...
-- Teller PID114943 is active serving Client12...
-- Teller PID114944 is active serving Client13...
-- Teller PID114945 is active serving Client14...
-- Teller PID114946 is active serving Client15...
Client04 deposited 20 credits to account BankID_01.
Client05 deposited 250 credits to account BankID_01.
Client06 withdrew 150 credits from account BankID_01.
Client07 deposited 1000 credits to account BankID_02.
Client08 deposited 2000 credits to account BankID_02.
Client09 withdrew 200 credits from account BankID_02.
Client10 withdraw failed for account BankID_03.
Client11 deposited 1000 credits to account BankID_03.
Client12 withdraw failed for account N.
Client13 deposited 500 credits to account BankID_04.
Client14 deposited 1000 credits to account BankID_03.
Client15 withdrew 2000 credits from account BankID_03.
Waiting for clients @ServerFIFO_Adabank0178...
Waiting for clients @ServerFIFO_Adabank0178...
```

```
talhamem@Talha:~/projects/Midterm$ ./bankClient Clients01.txt ServerFIFO_Adabank0178
Client is active...
Reading Clients01.txt...
15 clients to connect...
Creating client processes...
Connected to ServerFIFO_Adabank0178...
Client01 connected.. BankID_None depositing 50 credits
Client01 served. Invalid Bank ID!
Client02 connected.. hjashfjashf withdrawing 20 credits
Client03 connected.. N depositing -20 credits
Client02 served. Invalid Bank ID!
Client04 connected.. N depositing 20 credits
Client05 connected.. BankID_01 depositing 250 credits
Client03 served. Invalid amount request!
Client06 connected.. BankID_01 withdrawing 150 credits
Client07 connected.. N depositing 1000 credits
Client08 connected.. BankID_02 depositing 2000 credits
Client09 connected.. BankID_02 withdrawing 200 credits
Client10 connected.. BankID_03 withdrawing 1000 credits
Client11 connected.. N depositing 1000 credits
Client12 connected.. N withdrawing 1000 credits
Client13 connected.. N depositing 500 credits
Client14 connected.. BankID_03 depositing 1000 credits
Client15 connected.. BankID_03 withdrawing 2000 credits
Client05 served. Deposit successful
Client04 served. BankID_01
Client06 served. BankID_01
Client07 served. BankID_02
Client08 served. Deposit successful
Client09 served. BankID_02
Client10 served. Invalid account ID
Client11 served. BankID_03
Client12 served. Invalid account ID
Client13 served. BankID_04
Client14 served. Deposit successful
Client15 served. BankID_03
exiting..
```

```
Waiting for clients @ServerFIFO_Adabank0178...
Waiting for clients @ServerFIFO_Adabank0178...
- Received 7 clients from PIDClient115009..
-- Teller PID115083 is active serving Client01...
-- Teller PID115084 is active serving Client02...
-- Teller PID115085 is active serving Client03...
-- Teller PID115086 is active serving Client04...
-- Teller PID115087 is active serving Client05...
-- Teller PID115088 is active serving Client06...
-- Teller PID115089 is active serving Client07...
Client01 deposited 200 credits to account BankID_05.
Client02 deposited 5000 credits to account BankID_06.
Client03 deposited 200 credits to account BankID_01.
Client04 withdrew 500 credits from account BankID_02.
Client05 deposited 5000 credits to account BankID_07.
Client06 deposited 2000 credits to account BankID_08.
Client07 deposited 400 credits to account BankID_05.
Waiting for clients @ServerFIFO_Adabank0178...
Waiting for clients @ServerFIFO_Adabank0178...
```

```
talhamem@Talha:~/projects/Midterm$ ./bankClient Clients02.txt ServerFIFO_Adabank0178
Client is active...
Reading Clients02.txt...
7 clients to connect...
Creating client processes...
Connected to ServerFIFO_Adabank0178...
Client01 connected.. N depositing 200 credits
Client02 connected.. N depositing 5000 credits
Client03 connected.. BankID_01 depositing 200 credits
Client04 connected.. BankID_02 withdrawing 500 credits
Client05 connected.. N depositing 5000 credits
Client06 connected.. N depositing 2000 credits
Client07 connected.. BankID_05 depositing 400 credits
Client01 served. BankID_05
Client02 served. BankID_06
Client03 served. Deposit successful
Client04 served. BankID_02
Client05 served. BankID_07
Client06 served. BankID_08
Client07 served. Deposit successful
exiting..
```

```
Waiting for clients @ServerFIFO_Adabank0178...
^C
Signal received, closing...
Cleaning up server FIFO...
AdaBank says "Bye"...
```

```
talhamem@Talha:~/projects/Midterm$ ./bankClient Clients03.txt ServerFIFO_Adabank0178
Client is active...
Reading Clients03.txt...
6 clients to connect...
Creating client processes...
Error opening server FIFO: No such file or directory
Error connecting to ServerFIFO_Adabank0178
exiting..
```

Log File

Midterm > AdaBank.bankLog

```
1   # AdaBank Log file updated @13:27 April 27 2025
2   BankID_01 D 20 D 250 W 150 D 200 320
3   BankID_02 D 1000 D 2000 W 200 W 500 2300
4   #BankID_03 D 1000 D 1000 W 2000 0
5   BankID_04 D 500 500
6   BankID_05 D 200 D 400 600
7   BankID_06 D 5000 5000
8   BankID_07 D 5000 5000
9   BankID_08 D 2000 2000
10  ## end of log.
```

Testing with Existing Database

```
talhamem@Talha:~/projects/Midterm$ make
rm -f bankServer
rm -f bankClient
gcc -Wall -Wextra -g -o bankServer bankServer.c
gcc -Wall -Wextra -g -o bankClient bankClient.c
./bankServer AdaBank ServerFIFO_Adabank0178
AdaBank is active...
Loading bank database from log...
Waiting for clients @ServerFIFO_Adabank0178...
Waiting for clients @ServerFIFO_Adabank0178...
Waiting for clients @ServerFIFO_Adabank0178...
- Received 6 clients from PIDClient119206..
-- Teller PID119270 is active serving Client01...
-- Teller PID119271 is active serving Client02...
-- Teller PID119272 is active serving Client03...
-- Teller PID119273 is active serving Client04...
-- Teller PID119274 is active serving Client05...
-- Teller PID119275 is active serving Client06...
Client01 deposited 200 credits to account BankID_09.
Client02 deposited 2000 credits to account BankID_10.
Client03 deposited 200 credits to account BankID_01.
Client04 withdrew 500 credits from account BankID_02.
Client05 deposited 1000 credits to account BankID_11.
Client06 deposited 2000 credits to account BankID_12.
Waiting for clients @ServerFIFO_Adabank0178...
Waiting for clients @ServerFIFO_Adabank0178...
```

```
talhamem@Talha:~/projects/Midterm$ ./bankClient Clients03.txt ServerFIFO_Adabank0178
Client is active...
Reading Clients03.txt...
6 clients to connect...
Creating client processes...
Connected to ServerFIFO_Adabank0178...
Client01 connected.. N depositing 200 credits
Client02 connected.. N depositing 2000 credits
Client03 connected.. BankID_01 depositing 200 credits
Client04 connected.. BankID_02 withdrawing 500 credits
Client05 connected.. N depositing 1000 credits
Client06 connected.. N depositing 2000 credits
Client01 served. BankID_09
Client02 served. BankID_10
Client03 served. Deposit successful
Client04 served. BankID_02
Client05 served. BankID_11
Client06 served. BankID_12
exiting..
```

Log File

Midterm > AdaBank.bankLog

```
1  # AdaBank Log file updated @13:39 April 27 2025
2  BankID_01 D 20 D 250 W 150 D 200 D 200 520
3  BankID_02 D 1000 D 2000 W 200 W 500 W 500 1800
4  BankID_04 D 500 500
5  BankID_05 D 200 D 400 600
6  BankID_06 D 5000 5000
7  BankID_07 D 5000 5000
8  BankID_08 D 2000 2000
9  BankID_09 D 200 200
10 BankID_10 D 2000 2000
11 BankID_11 D 1000 1000
12 BankID_12 D 2000 2000
13 ## end of log.
```

Conclusion

In this project, one of the main challenges was ensuring safe and correct synchronization between multiple server-teller-client interactions using shared memory and semaphores. Designing the correct waiting and signaling mechanisms between tellers and the server was critical to avoid deadlocks and race conditions. Another challenge was debugging, which was solved because of long testing steps. Finally, I successfully completed the project, meeting all requirements. **Therefore, I expect to get full marks.**