**Relatório Técnico: Implementação de Comunicação Cliente-Servidor em C com FIFO e Chamadas ao Sistema**

**Introdução**

O presente relatório visa descrever detalhadamente a implementação de um sistema de comunicação entre processos cliente-servidor utilizando FIFO (First In First Out) e estruturas dinâmicas de dados na linguagem C. Ao longo do desenvolvimento, enfrentaram-se vários desafios técnicos relacionados com a biblioteca GLib, a gestão de processos, e a sincronização de dados entre processos pai e filho. Este documento expõe o percurso completo, desde a ideia inicial à resolução final, com foco na aprendizagem adquirida e soluções encontradas.

**Estrutura Inicial do Projeto**

A arquitetura inicial previa que o servidor criasse um FIFO principal para enviar a informação do cliente para o servidor e vice-versa. Cada cliente, ao iniciar comunicação, enviava uma mensagem fazendo um request ao servidor. O servidor, ao receber esta informação, cria um FIFO único para a partilha de informação servidor-cliente (este em específico), lia do FIFO a informação que o utilizador pretendia, processava o pedido e enviava a resposta através do mesmo FIFO.

Cada novo pedido por parte de um cliente implicava a criação de um processo filho por parte do servidor, sendo este responsável por indexar os dados recebidos numa estrutura chamada Livro. Esta estrutura era inicialmente guardada numa GHashTable, parte integrante da biblioteca GLib, permitindo um acesso rápido à informação indexada por chaves (keys).

**Objetivo Funcional:**

1. FIFO principal faz a conexão inicial entre o cliente e o servidor.
2. Cria-se processo filho para cada pedido.
3. Filho processa dados e armazena em GHashTable.
4. Resposta enviada ao cliente pelo FIFO único desta comunicação.

**Problemas Identificados**

Durante a implementação, surgiram vários entraves, que comprometeram a funcionalidade pretendida:

**Problemas com a GLib:**

* Incapacidade de armazenar estruturas Livro corretamente.
* Dificuldade em gerar chaves únicas apropriadas.
* Valores associados perdiam consistência após inserção.

A gravidade destes problemas conduziu à decisão de abandonar o uso da GLib temporariamente, optando pela implementação manual de uma estrutura baseada em Linked Lists.

**Problemas de comunicação entre processos pai e filho:**

* As informações processadas pelo filho não estavam a ser devidamente comunicadas ao processo pai.
* Ausência de sincronização provocava perda de dados ou incoerência de estado.

**Soluções Implementadas**

**Substituição da GHashTable por Linked Lists**

Foi implementada uma estrutura de lista ligada, com os seguintes atributos:

* Cada nodo representa um Livro com os dados relevantes.
* Ponteiro para o próximo nodo.
* Funções de inserção, remoção, pesquisa e liberação de memória.

Essa solução permitiu controlar melhor o conteúdo armazenado, mesmo que sacrificando a eficiência de pesquisa direta que a GHashTable oferecia.

**Serialização de Dados via Pipes Anônimos**

Para resolver a comunicação entre pai e filho, foi criada uma função de serialização condicional:

* Quando chamada pelo pai: leitura de dados vindos do pipe.
* Quando chamada pelo filho: escrita dos dados processados para o pipe.

Esta abordagem garantiu a troca unidirecional de dados e a sincronização dos estados internos entre processos.

**Testes e Complicações Finais**

Com o programa aparentemente funcional, foi executado o script de validação disponibilizado via BlackBoard. Contudo, o resultado foi dececionante:

* Apenas 100 a 200 anexações processadas entre cerca de 1600 esperadas.

**Tentativas de Resolução:**

1. Limitar o número de processos simultâneos a 20, controlando via contador.
2. Utilizar flag -f para esperar apenas no fim por todos os filhos.

Ambas as abordagens falharam em escalabilidade e estabilidade. Foi então decidido abandonar o modelo concorrente e adotar um modelo sequencial, tratando um processo de cada vez.

**Versão Final do Funcionamento**

A solução final consistiu na seguinte arquitetura de comunicação:

**Cliente:**

1. Cria FIFO exclusivo com o nome client\_response%d, onde %d é o seu PID.
2. Envia o seu PID através do FIFO principal.
3. Envia o pedido através do seu FIFO exclusivo.

**Servidor:**

1. Lê o PID do cliente através do FIFO principal.
2. Lê o pedido através do FIFO exclusivo do cliente.
3. Processa o pedido.
4. Devolve a resposta pelo mesmo FIFO.
5. Indexa a informação em lista ligada.

**Encerramento:**

O servidor termina a sua execução e liberta recursos quando recebe o comando especial -f vindo de um cliente, encerrando:

* FIFO principal.
* Memória alocada dinâmicamente.
* Todas as estruturas auxiliares.

**Considerações Finais**

O projeto enfrentou diversos desafios, desde problemas técnicos com bibliotecas até dificuldades no modelo de comunicação entre processos. As soluções encontradas, embora não ideais em termos de performance, permitiram a estabilidade e funcionalidade do sistema.

A experiência adquirida nesta implementação permitiu um melhor entendimento sobre:

* Troca de informação entre processos usando pipes e FIFO.
* Importância do controlo de concorrência em ambientes multi-processo.

Este trabalho, embora com soluções simples, atingiu o objetivo principal de permitir comunicação eficiente e organizada entre cliente e servidor, com capacidade de armazenar e reutilizar dados através de estruturas dinâmicas adequadas.

**Trabalho realizado por:**

João Durães, 109065, LCC

João Neiva, 108579, LCC

Miguel Paço, 108574, LCC