Documentação

Indices:

1. Introdução:

O programa cliente-servidor com memória compartilhada é uma aplicação que permite a troca de mensagens e o compartilhamento de arquivos entre servidor e vários clientes. A utilização de memória compartilhado nesse caso é uma técnica eficiente para a comunicação entre processos, pois permite acesso direto à memória, sem a necessidade de comunicação por meio de sockets ou outros mecanismos IPC (Inter-Process Communication) .Neste relatório, analisaremos a execução do programa, as bibliotecas utilizadas e a estruturação do código.

2. Execução do Programa:

Para executar o programa siga os passos abaixo:

1. Compilação: Utilize um compilador C para compilar os arquivos "client.c" e "server.c" em binários executáveis. Por exemplo:
    a.

```
gcc server.c funcoes.o -o server
gcc client.c funcoes.o -o client
```

2. Execução do Servidor: Inicie o programa servidor executando o binário do servidor. Isso abrirá um terminal que ficará aguardando a conexão dos clientes. Por exemplo:

```
./server
```

3. Execução dos Clientes: Em diferentes instâncias do terminal, execute o programa cliente executando o binário do cliente. Isso permitirá a comunicação entre os clientes e o servidor. Por exemplo:

```
./client
```

4. Interagindo com o Programa: Ao executar o cliente, você será apresentado a um menu com opções para enviar mensagens, visualizar mensagens, enviar arquivos, visualizar arquivos e sair. Selecione a opção desejada digitando o número correspondente e siga as instruções adicionais.
5. Encerramento do Programa: Para encerrar o programa cliente ou servidor, selecione a opção "Sair" no menu.

É importante ressaltar que o programa cliente-servidor com memória compartilhada deve ser executado em um ambiente adequado, com as permissões necessárias e as bibliotecas requeridas. Caso contrário, erros podem ocorrer durante a execução.

4. Bibliotecas Utilizadas:

O programa faz uso de algumas bibliotecas importantes. A seguir, descreveremos as principais bibliotecas utilizadas e sua função no contexto do programa:

- `stdio.h`: Essa biblioteca é amplamente utilizada em programas C e fornece funções para entrada e saída de dados. No código, ela é usada para realizar a leitura e escrita de dados pelo usuário no cliente e para exibir mensagens e informações no servidor.
- `stdlib.h`: Essa biblioteca padrão em C fornece funções para alocação de memória, conversão de tipos, geração de números aleatórios e outras operações gerais. No

código, é usada para realizar a alocação de memória compartilhada, manipulação de ponteiros e finalização do programa.

- `string.h`: Essa biblioteca oferece várias funções para manipulação de strings em C. No programa cliente, é utilizada para manipular as mensagens digitadas pelo usuário, como copiar, comparar e remover caracteres indesejados. Também é utilizada no servidor para manipular os dados recebidos dos clientes.
- `sys/ipc.h` e `sys/shm.h`: Essas bibliotecas são específicas para o uso de memória compartilhada em sistemas Unix. Elas fornecem as estruturas e funções necessárias para a criação, obtenção e manipulação de segmentos de memória compartilhada. No código, são usadas para criar e acessar a memória compartilhada entre o cliente e o servidor.
- `unistd.h`: Essa biblioteca padrão em C fornece acesso a várias constantes, tipos e funções relacionadas ao sistema operacional. No código, é utilizada para obter o ID do processo, permitindo a identificação única de cada cliente.
- `stdbool.h`: Essa biblioteca define o tipo de dados `bool` em C, juntamente com os valores `true` e `false`. No código, é usada para realizar verificações lógicas e controlar a execução do programa com base em condições.

Cada biblioteca desempenha um papel importante para o correto funcionamento do programa, fornecendo funções e estruturas necessárias para realizar tarefas específicas.

5.Funções Utilizadas:

```
generateKey() // Retorna uma key com variavies padrões do projeto
ftok() // Retorna uma chave unica para o arquivo
shmget() // Retorna o id do segmento de memoria compartilhada
shmat() // Usada para associar o segmento de memoria compartilhada ao processo
shmdt() // Desassocia o segmento de memoria compartilhada do processo
shmctl() // Remove o segmento de memoria compartilhada

getpid() // Retorna o id do processo atual
fgets() // Le uma string do teclado
strcpy() // Copia uma string para outra
memset() // Preenche um bloco de memoria com um valor especifico
fopen() // Abre um arquivo
fseek() // posiciona o ponteiro do arquivo
ftell() // retorna o tamanho do arquivo
```

6. Estruturação do Código:

O programa é dividido em duas partes principais: o cliente e o servidor. Ambos os códigos estão contidos em arquivos separados para facilitar a compreensão e manutenção do código.

A estrutura de dados `Message` é definida no código tanto do cliente quanto do servidor. Essa estrutura contém campos para o ID do cliente, a mensagem, o nome e o conteúdo do arquivo, bem como o tamanho do arquivo. Essa estrutura é utilizada para armazenar e transmitir os dados entre o cliente e o servidor por meio da memória compartilhada.

O código é estruturado de forma a permitir a comunicação bidirecional entre o cliente e o servidor, com o uso da memória compartilhada como meio de troca de informações. O servidor aguarda a chegada de mensagens ou arquivos dos clientes e os exibe no terminal. O cliente permite que o usuário digite mensagens, envie arquivos e visualize as mensagens recebidas do servidor.

7. Considerações Finais:

O programa apresentado é uma aplicação prática e eficiente para a troca de mensagens e compartilhamento de arquivos entre processos. A utilização de memória compartilhada proporciona uma forma rápida e direta de comunicação, evitando a necessidade de comunicação por meio de sockets ou outros mecanismos mais complexos.

No presente relatório, abordamos a execução do programa, destacando as etapas para compilação, execução e interação com o programa. Além disso, foram apresentadas as principais bibliotecas utilizadas, ressaltando seu papel no contexto do programa.

Também discutimos a estruturação do código, dividido em partes distintas para o cliente e o servidor. Ambos os códigos utilizam a estrutura de dados `Message` para armazenar e transmitir os dados pela memória compartilhada.

Em resumo, o programa demonstra o uso prático da comunicação entre processos por meio de memória compartilhada, oferecendo uma solução eficiente e de baixa complexidade para a troca de mensagens e compartilhamento de arquivos.

Referências:

1. Biblioteca `stdio.h` - **Link para a documentação**

2. Biblioteca `stdlib.h` - **Link para a documentação**
3. Biblioteca `string.h` - **Link para a documentação**
4. Biblioteca `sys/ipc.h` - **Link para a documentação**
5. Biblioteca `sys/shm.h` - **Link para a documentação**
6. Biblioteca `unistd.h` - **Link para a documentação**
7. Biblioteca `stdbool.h` - **Link para a documentação**
8. `ftok` - Link para a documentação
9. `fopen` - Link para a documentação
10. https://stackoverflow.com/questions/5656530/how-to-use-shared-memory-with-linux-in-c
11. Inter process communication shared_memory
12. How to create a shared memory segment