	**Universidade Federal da Bahia**
	Instituto de Matemática e Estatística
	Departamento de Ciência da Computação
	Fundamentos de Sistemas Distribuídos (2019.1)
	Professor: Alirio Sá

Prática 01: Serviço de arquivos distribuídos (versão 1)

Sumário

	Universidade Federal da Bahia
	Instituto de Matemática e Estatística
	Departamento de Ciência da Computação
	Fundamentos de Sistemas Distribuídos (2019.1)
	Professor: Alirio Sá

Prática 01: Serviço de arquivos distribuídos (versão 1)

1 Objetivo

Assimilar os aspectos básicos da comunicação entre pares de processos distribuídos, a partir de um projeto prático envolvendo comunicação direta usando API da Internet e invocação de procedimentos remotos.

2 Descrição

Nesta atividade prática, a implementação consiste em um sistema de arquivos distribuídos que permite que clientes realizem operações sobre um sistema de arquivos de um servidor remoto, seguindo os seguintes passos (ver Figura 1):

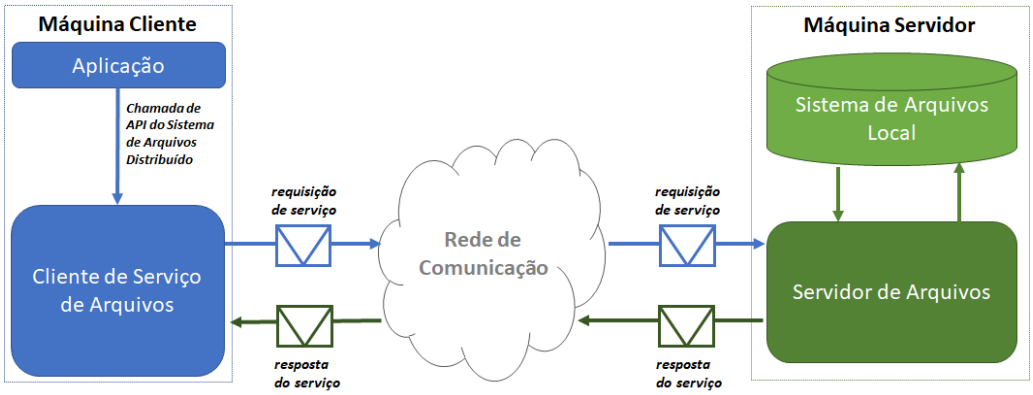

Figura 1 - Execução de uma chamada a API do sistema de arquivos distribuídos

1. Uma aplicação usuária usa a API disponibilizada pelo cliente do serviço de arquivos para solicitar uma operação sobre arquivos remotos.
2. O cliente de serviço de arquivos transforma a chamada de operação em uma mensagem de requisição de serviço que é enviada pela rede ao servidor de arquivos remoto.
3. O servidor de arquivos interpreta a requisição recebida e realiza a operação correspondente no seu sistema de arquivos local.
4. O servidor transforma o resultado da operação executada em uma mensagem de resposta de requisição de serviço que é enviada pela rede ao cliente de serviço.
5. O cliente de serviço completa a operação devolvendo o retorno a chamada da API para a aplicação usuária.

Quando uma aplicação usuária faz uma chamada a uma operação disponibilizada pela API do cliente do serviço de arquivos, a mesma fica bloqueada até que a operação seja concluída. É permitido que uma aplicação usuária *multithread* tenha diferentes threads realizando chamadas a diferentes operações da API do cliente do serviço.

Apesar de cada operação de sua API ser bloqueante, o cliente do serviço de arquivos deve ser capaz de realizar operações em paralelo, atendendo simultaneamente às chamadas individuas de múltiplas threads de uma mesma aplicação usuária. Para cada operação chamada, uma mensagem de requisição de serviço, como as informações da operação, é construída e enviada pela rede ao servidor de arquivos remoto. Uma chamada de operação da API do cliente só concluir, retornando o resultado para o chamador da operação, quando o cliente receber uma mensagem de resposta do servidor de arquivos com os resultados da operação executada remotamente.

O servidor de arquivos deve ser capaz de atender simultaneamente a requisições de múltiplos clientes distintos. Por conta disto, o servidor deve realizar o devido controle de concorrência entre as operações para evitar que haja inconsistências nas operações concorrentes realizadas sobre seus arquivos (e.g., duas operações distintas, envolvendo modificação de estado, acessando o mesmo arquivo).

Prática 01: Serviço de arquivos distribuídos (versão 1)

3 API do cliente de serviço de arquivos distribuídos

Para realizar operações remotas sobre os arquivos do servidor, a API do cliente de serviço de arquivos distribuídos fornece as operações $ropen$, $rread$, $reof$, $rwrite$, $rremove$, $rseek$, $rgetpos$ e $rclose$ como versões de execução remota das operações $fopen$, $fread$, $feof$, $fwrite$, $remove$, $fseek$, $fgetpos$ e $fclose$ do padrão POSIX que serão executadas no servidor. Por exemplo, quando a aplicação usuária chama $ropen$ no lado do cliente do serviço, as interações entre cliente e servidor serão realizadas para que a operação $fopen$ seja realizada no lado do servidor (ver Figura 2).

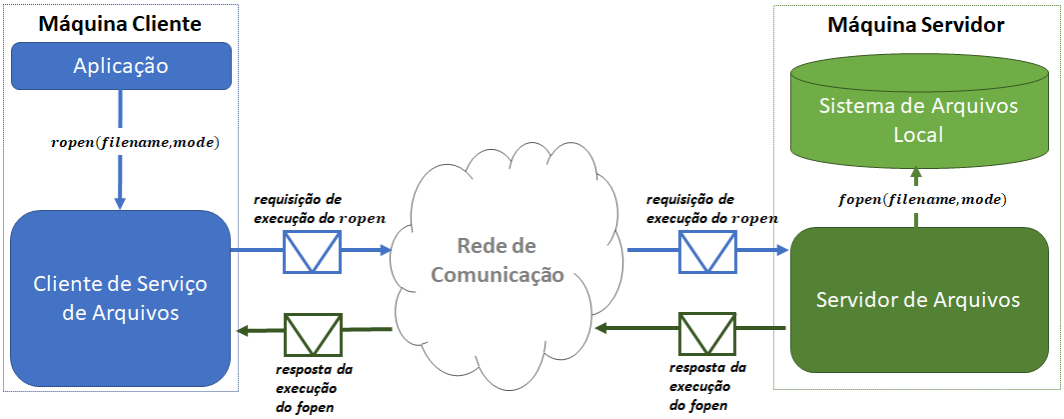

Figura 2 - Exemplo da execução da operação ropen

3.1 Operação $ropen$

Usada para a abertura de arquivos remotos. Oferece para a aplicação cliente a seguinte assinatura:

$$rid = ropen(filename, mode)$$

A função $ropen$ abre, usando o modo de acesso indicado por $mode$, o arquivo indicado por $filename$ e retorna um identificador (rid) para acesso ao fluxo de bytes com o arquivo aberto.

3.1.1 Detalhamento dos parâmetros

- $filename$ – Cadeia de caracteres que define o nome do arquivo
- $mode$ – Cadeia de caracteres que especifica o modo de acesso, podendo ser usado:
  - "r", para abertura de arquivo existente em modo somente leitura (a partir do início)
  - "w", para a criação de um arquivo para a escrita, sobrescrevendo caso o mesmo já exista.
  - "a", para abertura de um arquivo existente para escrita (a partir do final do arquivo)
  - "r+", para abertura de um arquivo para leitura e escrita (a partir do início do arquivo), retornando um erro caso o arquivo não existe.
  - "w+", para criação de um arquivo para leitura e escrita, sobrescrevendo caso o mesmo já exista.
  - "a+", para abertura de um arquivo para leitura e escrita, iniciando do final do arquivo e criando um arquivo novo caso o arquivo não exista.

3.1.2 Detalhamento do retorno

- Se a execução for bem-sucedida, retornará um inteiro positivo (rid) que poderá ser usado em futuras evocações das demais operações da API do cliente.
- Caso a execução for malsucedida, retornará zero indicando que a operação não pode ser realizada.

3.2 Operação $rread$

Usada para a leitura do conteúdo de arquivo remotos. Oferece para a aplicação cliente a seguinte assinatura:

$$sizeout = rread(buf, sizebuf, count, rid)$$

A função $rread$ lê bytes de um arquivo, identificado por rid, para o $buffer$ em memória.

3.2.1 Detalhamento dos parâmetros

- $buffer$ – Define em que região da memória do cliente os bytes lidos devem ser armazenados.
- $sizebuf$ – Especifica o tamanho da região de memória aponta por $buffer$.
- $count$ – Define o número de bytes a serem lidos.
- rid – Especifica o identificador do arquivo sobre o qual a operação deve ser realizada.

3.2.2 Detalhamento do retorno

- Se a execução for bem-sucedida, retornará o número de bytes lidos. Caso o final do arquivo seja alcançado, o número de bytes lidos será menor que o especificado pelo parâmetro $count$.
- Retornará zero, caso o cursor esteja apontando para o final do arquivo ou a operação seja malsucedida.

3.3 Operação $reof$

Usada para verificar se o curso de um arquivo remoto aponta para o final do arquivo. Oferece para a aplicação cliente a seguinte assinatura:

$$ret = reof(rid)$$

3.3.1 Detalhamento dos parâmetros

- rid – Especifica o identificador do arquivo para o qual se deseja checar se o cursor alcançou o final do arquivo.

3.3.2 Detalhamento do retorno

Retorna um valor diferente de zero se o final do arquivo foi alcançado, caso contrário retornará zero.

3.4 Operação $rwrite$

Usada para a escrita de conteúdo em arquivos remotos. Oferece para a aplicação cliente a seguinte assinatura:

$$sizeout = rwrite(buf, sizebuf, count, rid)$$

A função $rwrite$ escreve bytes do $buffer$ em memória para um arquivo remoto, identificado por rid.

3.4.1 Detalhamento dos parâmetros

- $buffer$ – Define a região da memória do cliente que contém os bytes a ser escritos.
- $sizebuf$ – Especifica o tamanho da região de memória aponta por $buffer$.
- $count$ – Define o número de bytes a serem escritos.
- rid – Especifica o identificador do arquivo sobre o qual a operação deve ser realizada.

3.4.2 Detalhamento do retorno

Retorna o número de bytes escritos com sucesso, o que pode ser menos que o especificado pelo parâmetro $count$, caso um erro ocorra.

3.5 Operação $rseek$

Usada definir uma posição no stream para o arquivo remoto aberto. Oferece para a aplicação cliente a seguinte assinatura:

$$rseek(rid, offset, origin);$$

A função $rseek$ define uma nova posição $offset$ para o stream rid, a partir da posição relativa ($origin$).

3.5.1 Detalhamento dos parâmetros

- rid – Especifica o identificador do arquivo sobre o qual a operação deve ser realizada.
- $offset$ – número de bytes do deslocamento da posição a partir de $origin$
- $origin$ – posição usada como referência para o deslocamento $offset$, podendo ser:
  - $SEEK\_SET$ – início do arquivo

- $SEEK\_CUR$ – posição atual
- $SEEK\_END$ – final do arquivo.

3.5.2 Detalhamento do retorno
Retorna zero se a operação foi bem-sucedida e, caso contrário, retornará um valor diferente de zero.

3.6 Operação $rgetpos$
Usada para capturar a posição atual do stream no arquivo remoto. Oferece para a aplicação cliente a seguinte assinatura:

$$rgetpos(rid, pos)$$

3.6.1 Detalhamento dos parâmetros

- rid – Especifica o identificador do arquivo sobre o qual a operação deve ser realizada.
- pos – após a operação deverá apontar para a posição atual no stream do arquivo.

3.6.2 Detalhamento do retorno
Retorna zero se a operação foi bem-sucedida e, caso contrário, retornará um valor diferente de zero.

3.7 Operação $rclose$
Usada para fechar um arquivo remoto. Oferece para a aplicação cliente a seguinte assinatura:

$$ret = rclose(rid)$$

3.7.1 Detalhamento dos parâmetros

- rid – Especifica o identificador do arquivo a ser fechado.

3.7.2 Detalhamento do retorno
Retorna zero se a operação foi bem-sucedida, caso contrário retornará EOF (*End of File*).

3.8 Operação $rremove$
Usada para apagar um arquivo remoto. Oferece para a aplicação cliente a seguinte assinatura:

$$ret = rremove(filename)$$

3.8.1 Detalhamento dos parâmetros

- $filename$ – Cadeia de caracteres que define o nome do arquivo a ser apagado.

3.8.2 Detalhamento do retorno
Retorna zero se a operação foi bem-sucedida, caso contrário retornará um valor diferente de zero.

4 Aspectos da implementação da interação Cliente/Servidor

Para esta prática, deve ser possível que o serviço de arquivo distribuído seja usando a partir de operações clientes implementadas usando sockets (TCP ou UDP) e invocação remotas (com RPC ou com RMI). Assim, a entrega do projeto considera a integração de duas implementações.

Na primeira implementação, a API do cliente do serviço é implementada usando sockets TCP ou UDP. Nessa implementação, quando uma operação da API cliente é chamada, uma mensagem de requisição deve ser criada e enviada ao servidor usando sockets. Da mesma forma, o servidor receberá a requisição via socket, interpretará a mensagem, realizará a operação em seu sistema de arquivos e devolverá a resposta via socket. A resposta recebida pelo cliente via socket será interpretada e o resultado da operação da API que foi chamada será retornado para a aplicação usuária.

Na segunda implementação, a API cliente do serviço é implementada usando chamada remota de procedimento, a partir de RPC ou RMI. Desse modo, o servidor receberá a requisição via invocação remota (RPC ou RMI) e retornará o resultado na conclusão da invocação remota (RPC ou RMI).

As duas implementações acima devem coexistir – isto é, o acesso aos serviços do sistema de arquivos distribuído poderá ser feito, de forma concorrente, tanto por requisições feitas via sockets quanto por requisições feitas via invocação remota.

5 Restrições de implementação

(1) Deve ser permitido que cada processo do sistema execute em um computador distinto e se comuniquem via rede de computadores;

(2) Caso sejam realizados testes em uma mesma máquina, a comunicação entre os processos deve ainda usar os mecanismos de comunicação disponíveis no sistema operacional para interação via rede.

(3) O programa pode ser implementado em linguagens para programação desktop em rede, como por exemplo Java, Python, C, C++, Pascal, Pascal Object, VB, VB.NET, C# etc.

(4) Não são permitidas linguagens voltadas para programação web executando scripts em servidores web, como por exemplo PHP, PERL, ASP, ASP.NET etc.

(5) Também não são permitidas linguagens voltadas para programação web executando scritps em clientes web, como por exemplo javascript, vbscript etc.

(6) Todo o programa deve estar devidamente comentado, de modo a facilitar o entendimento do código.

(7) Não devem ser usados middlewares específicos para comunicação entres os módulos cliente e servidor, salvo aqueles nativos de linguagens com Java, C, C++, Python etc.

6 Equipes

A prática deve ser realizada por equipes de até 03 (três) estudantes. Todos os estudantes devem estar completamente cientes de todos os aspectos do projeto e da implementação. Cabe lembrar que as notas serão dadas, não apenas de acordo com os entregáveis, mas também conforme o desempenho individual de cada estudante na apresentação dos mesmos – tendo estas apresentações o maior peso na nota final.

7 Detalhamento dos entregáveis

Etapa	Descrição	Prazo de Entrega
Projeto do Software	Documento com as definições dos módulos de software, suas responsabilidades e formas de interação para atender aos requisitos especificados. Além disso, deve ser definida quais componentes serão instanciados nas máquinas dos clientes e nas máquinas dos servidores.	17/04/2019
Sockets	Apresentar códigos fontes e documentação relacionada a implementação do serviço usando comunicação via sockets TCP ou UDP. A documentação deve ser suficiente para entendimento do código, dos formatos das mensagens, estruturas de dados etc., devendo indicar também os passos para a configuração e uso do serviço.	08/05/2019
Chamada Remota	Apresentar códigos fontes e documentação relacionada a implementação do serviço usando comunicação via chamada remota RPC ou RMI. A documentação deve ser suficiente para entendimento do código, das estruturas de dados etc., devendo indicar também os passos para a configuração e uso do serviço.	15/05/2019
Integração	Entrega de versão final da prática com as duas implementações integradas, acompanhado de documentação final da versão integrada, considerado prováveis ajustes realizados nos demais integráveis.	19/05/2019

8 Referências

8.1 Bibliografia básica

Tanebaum, A. S.; Steen, M. V. **Sistemas Distribuídos: Princípios e Paradigmas. 2ª Edição**, 2008, Editora Pearson.

Coulouris et. al. **Sistemas Distribuídos: Conceitos e Projeto. 5ª Edição**, 2013, Editora Bookman.

8.2 Bibliografia complementar

Stevens, W. R.; Fenner, B.; Rudoff, A. M. **Unix Network Programming. 3rd Edition**, Volume 1, 2003, Addison Wesly.

Tanebaum, A. S.; Steen, M. V. **Distributed System. 3rd Edition**, 2017, Editora Pearson. Disponível online em: https://www.distributed-systems.net/index.php/books/distributed-systems-3rd-edition-2017/

Veríssimo, P.; Rodrigues, L. **Distributed Systems for System Architects**. 2000, Kluwer Academic Publishers.

8.3 Links úteis

- https://en.cppreference.com/w/c/io/fclose
- https://en.cppreference.com/w/c/io/feof
- https://en.cppreference.com/w/c/io/fopen
- https://en.cppreference.com/w/c/io/fread
- https://en.cppreference.com/w/c/io/fwrite
- https://en.cppreference.com/w/c/io/remove