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

Prática 01: Serviço de arquivos distribuídos (versão 1)

Salvador - Bahia
17 de Abril de 2019

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) 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.

3) Introdução

3.1) Cliente - Servidor

Um computador que executa um programa que faz uma solicitação de serviços é chamado de cliente. Um computador que executa um programa que oferece serviços solicitados de um ou mais clientes é chamado servidor. A mídia para comunicação pode ser rede com ou sem fio.
Geralmente, programas em execução em máquinas clientes fazem solicitações a um programa (geralmente chamado de programa do servidor) em execução em uma máquina servidora. Elas envolvem serviços de rede fornecidos pela camada de transporte, que faz parte da pilha de software da Internet, geralmente chamada TCP / IP (protocolo de controle de transporte / protocolo da Internet). A camada de transporte compreende dois tipos de protocolos, o TCP (Transport Control Protocol) e o UDP (User Datagram Protocol) . As interfaces de programação mais utilizadas para esses protocolos são sockets.

4) Processo

Servidor:
1- Cria um IP End Point (endereço IP e Porta) e um objeto Socket e então vincula o objeto socket com o IP End Point e envia-o para o modo listen para aguardar a conexão do cliente.

Cliente:
2- Cria um IP End Point (endereço IP e Porta) e um objeto socket.
3- Prepara os dados do arquivo (em bytes) a ser enviado.
4- Tentar se conectar com o servidor usando socket cliente com ajuda do IP End Point.

Servidor:
5- Recebe a solicitação do cliente e aceita. Uma vez que a conexão esteja estabelecida, o servidor cria um outro objeto soquete que irá lidar com este cliente até que todos os pedidos de conexão terminem. O Cliente será informado internamente pelo TCP sobre o sucesso de conexão.

5) Passo a passo detalhado

1) Clientes fazem requisições para acessar o arquivo remoto; O arquivo permanece no servidor.
2) O arquivo é transferido do servidor para o cliente;
3) Acessos são realizados no cliente;
4) Ao término, o arquivo é retornado ao servidor.

6) Cliente

O programa cliente primeiro cria um *socket* através da função *socket()*. Em seguida ele se conecta ao servidor através da função *connect()* e inicia um *loop* (laço) que fica fazendo *send()* (envio) e *recv()* (recebimento) com as mensagens específicas da aplicação. É no par *send, recv* que temos a comunicação lógica. Quando alguma mensagem da aplicação diz que é o momento de terminar a conexão, o programa chama a função *close()* para finalizar o *socket*.

7) Servidor

O programa servidor também utiliza a mesma API de *sockets.* Ou seja, inicialmente ele também cria um *socket.* No entanto, diferentemente do cliente, o servidor precisa fazer um *bind(),* que associa o *socket* a uma porta do sistema operacional, e depois utilizar o *listen()* para escutar novas conexões de clientes nessa porta.

Quando um novo cliente faz uma nova conexão, a chamada *accept()* é utilizada para começar a se comunicar. Da mesma forma que no cliente, o servidor fica em um *loop* (laço) recebendo e enviando mensagens através do par de funções *send()* e *recv()*. Quando a comunicação com o cliente termina, o servidor volta a aguardar novas conexões de clientes.