Trabalho Prático 4: SOCKETS/PIPE/THREADS

Aluno em Graduação da Universidade Federal de Ouro Preto do curso Ciência da Computação:

Halliday Gauss Costa dos Santos.

Matrícula: 18.1.4093.

Área: Sistemas Operacionais.

Introdução:

Muitas aplicações em redes de computadores necessitam de comunicação entre duas ou mais máquinas distintas, ou seja, uma máquina deve ser capaz de se conectar a uma outra independente do SO. Para isso acontecer uma máquina precisa identificar a outra, então é usado o endereço de IP para fazer essas comunicações, já que o PID de cada máquina pode ser o mesmo impossibilitando a comunicação. Este documento apresenta a análise da implementação de Sockets em **C.**

Desenvolvimento:

No trabalho prático, foi criado um cliente e um servidor. O servidor é responsável por receber um determinado número do cliente, e, utilizando esse número, calcular a sequência de Fibonacci até o n-ésimo número passado como parâmetro. Por fim, o servidor deve retornar essa sequência para o cliente. Obs: o número passado para o servidor é o número utilizado ao executar o programa do cliente. Ex: executável 8, retornará para o cliente a sequência de Fibonacci até o oitavo número.

No servidor é criado um socket, com as seguintes características:

- Family = AF_INET -> protocolo IPv4.
- Porta -> 5000.
- IP -> 127.0.0.1.

Em seguida, é utilizada a função **bind** para conectar o socket ao endereço de IP e a Porta supracitada. O servidor fica esperando conexões de um cliente. Quando um cliente se conectar, o servidor espera um número ‘n’ do cliente (**recv**). Então é calculada a sequência de Fibonacci até o n-ésimo termo e essa mesma sequência é enviada para o cliente (**send**).

No cliente a implementação é mais simples:

- Primeiramente é criado um socket com as mesmas informações do socket do servidor (**IP, Porta** e **Family**.).
- Em seguida o cliente tenta se conectar ao servidor(**connect**).
- Após a conexão, o cliente envia um número ‘n’ para o servidor(**send)** e espera a sequência ser retornada para imprimi-la na tela (**recv).**

Diferente do uso de threads e PIPE, o uso de sockets não necessita que os processos estejam em uma mesma máquina para haver uma conexão entre eles, e como os computadores são autônomos e distintos não acontece os problemas que aparecem no uso de threads, pois a área de memória de cada programa é separada. A diferença está também no uso de PIPE e threads acontecerem em um mesmo sistema operacional, diferente do que ocorre com sockets que a comunicação entre os dispositivos independe do sistema operacional.

Conclusão:

A realização desse trabalho foi de suma importância para o melhor entendimento do funcionamento de sockets e sua implementação em uma linguagem de programação, e para entendimento da diferença entre sockets, PIPE e threads.