Prazo: 20/11

Introdução

Programação de soquete onde o servidor deve lidar com vários clientes é um desafio em que podem ser utilizadas duas estratégias. A primeira é o *Select()*, que é melhor em questões de desempenho computacional, principalmente quando se tem centenas de operações simultâneas. No entanto, este requer um cuidado maior na escrita do código, pois o estilo de codificação é muito diferente da programação tradicional de encadeamento único. Por exemplo, você precisa evitar chamar métodos de bloqueio, pois isso pode parar sua aplicação por completo. A segunda é a *Thread*, que é mais simples, pois a maioria do código se assemelha ao código de um único processo. A parte difícil é lidar com os poucos locais em que é necessária a comunicação entre *threads*, seja por meio de *mutexes* ou outros mecanismos de sincronização.

Tarefa

O objetivo deste último trabalho prático é utilizar programação soquete no qual o servidor consegue lidar com vários clientes. Para isso, vocês podem utilizar *threads* ou *select()*. A aplicação a ser desenvolvida deve ser um *chat* com as seguintes funcionalidades:

1. O cliente conecta em um servidor através de um nome específico (string de 15 caracteres, no máximo). O nome do usuário deve ser passado como parâmetro ao inicializar o cliente;
2. O cliente pode recebe uma lista atualizada de usuários conectados ao servidor; (ver comandos)
3. O cliente pode mandar uma mensagem para **todos** os usuários conectados ao servidor; (ver comandos)
4. O cliente também pode mandar mensagem para um usuário específico; (ver comandos)
5. Se um cliente desconecta do servidor, a lista de usuários deve ser atualizada no momento em que isso ocorre; (ver comandos)
6. As conexões devem ser realizadas através do protocolo TCP, exclusivamente; e

Comandos que o cliente pode enviar ao servidor:

- users – servidor responde com todos os usuários conectados.
- all;msg – msg do cliente é enviada para todos os usuários conectados ao servidor.
- uni;usuarioX;msg – mensagem unicast msg do cliente é enviada para o usuárioX.
- exit – cliente e servidor fecham a conexão. Servidor atualiza lista de usuários ativos.

Dicas

1. Para a escolha entre *threads* ou *select()*, pense (além das características de cada estratégia) na quantidade de clientes que vão se conectar ao servidor. Por exemplo, se você precisará apenas de um pequeno número de *threads*, usar um modelo de programação mais simples pode ser mais vantajoso.
2. Você pode usar os códigos de cliente/servidor TCP feitos por você no trabalho prático anterior.