Introdução

Um soquete de rede é um terminal em um fluxo de comunicação entre dois programas em execução em uma rede. Os soquetes são criados e usados com um conjunto de solicitações de programação (isto é, chamadas de função). Atualmente, a maioria das comunicações entre computadores é baseada no protocolo da *Internet*. Por esse fato, a maioria dos soquetes de rede são soquetes da *Internet*. No entanto, soquetes também podem ser usados para comunicação entre processos dentro de um mesmo computador. O soquete é, principalmente, um conceito usado na camada de Transporte.

Camada de Transporte

A camada de Transporte se localiza entre as camadas de Aplicação e de *Internet* na pilha TCP/IP e corresponde à camada de Transporte no modelo OSI. As principais funções da camada de Transporte são: comunicação entre processos; controle de fluxo; controle de erros; controle de congestionamento de rede; estabelecer e gerenciar conexões; dentre outras. Em síntese, a camada de Transporte tem o papel de fornecer funções que permitam a comunicação entre processos de aplicações (isto é, *softwares*) entre computadores diferentes. Assim, essa camada fornece um mecanismo pelo qual diversas aplicações distintas podem enviar e receber dados usando a mesma implementação de protocolos das camadas mais baixas.

Comunicação entre processos

Um processo é um *software* em execução (ou seja, uma entidade da camada de Aplicação) o qual irá utilizar os serviços fornecidos pela camada de Transporte. Para que seja possível entregar a mensagem ao processo correto, a camada de Transporte utiliza identificadores denominados números de portas.

Números de Portas

A comunicação entre processos em uma rede faz uso do paradigma cliente-servidor. Nesse paradigma, um processo em um dos *hosts* (cliente) requer serviços de um processo em outro *host* remoto (servidor). Em ambos os casos, os processos possuem o mesmo nome. Por exemplo, podemos realizar a conexão remota a um *host* utilizando uma aplicação (processo) cliente chamado *ssh* no *host* local, que irá se comunicar a um processo servidor de nome *ssh*, executado no *host* remoto. Para definir os *hosts* que irão se comunicar, é usado o endereçamento IP, realizado na camada de *Internet* do TCP/IP (camada de Rede do modelo OSI).

Soquetes

Um soquete é a combinação de um endereço IP com um número de porta. Tal combinação possibilita a definição de um processo de forma única, tanto no cliente quanto no servidor. Portanto, para usar os serviços de rede da camada de Transporte, é necessário um par de endereço de soquete: um para o cliente e outro para o servidor. Um soquete possui dados tanto da camada de *Internet* (isto é, endereço IP) quanto da camada de Transporte (número de porta).

Tarefa

O objetivo dessa tarefa é implementar um cliente e servidor TCP e um cliente e servidor UDP. Note: são quatro programas diferentes ao todo. Você deve usar a linguagem de programação C. Seu cliente e seu servidor (TCP ou UDP) se comunicará pela rede e trocará dados. O servidor iniciará no modo passivo, ouvindo uma transmissão do cliente. O cliente iniciará e contatará o servidor (em um determinado endereço IP e número de porta). O cliente passará ao servidor uma *string* (por exemplo: "trabalho de soquete") com, no máximo, 80 caracteres. Ao receber uma sequência de caracteres de um cliente, o servidor deve:

i. reverter todos os caracteres; e
ii. reverter a capitalização da sequência.

No exemplo, "trabalho de soquete" se tornará "ETEUQOS ED OHLABART". O servidor deve enviar a *string* de volta ao cliente. Por fim, o cliente exibirá a sequência de caracteres recebida e sairá. Sendo assim, sua tarefa é implementar os programas cliente e servidor em C usando UDP e TCP. Você deve entregar 4 programas diferentes:

1. Cliente em C usando UDP (nome do arquivo a ser entregue: **cliente_udp.c**);
2. Servidor em C usando UDP (nome do arquivo a ser entregue: **servidor_udp.c**);
3. Cliente em C usando TCP (nome do arquivo a ser entregue: **cliente_tcp.c**); e
4. Servidor em C usando TCP (nome do arquivo a ser entregue: **servidor_tcp.c**).

Exemplo

Iniciando o servidor. Suponha que você iniciou um servidor na máquina **192.168.56.10**, ouvindo a porta número **32000**. A sintaxe deve se parecer com o seguinte:

```
maquina1 > servidor 32000
```

Note que “servidor” deve ser substituído por um dos IPs especificados anteriormente. Iniciando o cliente:

```
maquina2 > cliente 192.168.56.10 32000
```

Note que “cliente” deve ser substituído por um dos IPs especificados anteriormente. Em seguida, o texto a ser inserido deve ser solicitado: “Programando Soquete em C”. Resposta do servidor: “c ME ETEUQOs ODNAMARGORp”.

```
maquina2 >
```

Neste ponto, depois de receber uma *string* revertida, o cliente deve sair. O servidor ficará ativo esperando uma nova requisição. O código deve funcionar para uma nova requisição.

Compilando seu código

Para compilar seu código C, use os seguintes comandos como guia:

```
gcc -g -o clienteUDP cliente_udp.c -lsocket –lnsl
```

Esse comando deve gerar um executável denominado “clienteUDP” para seu programa “cliente_udp.c”. Similarmente, faça isso para o cliente TCP. O mesmo ocorre para o comando de servidor:

```
gcc -g -o servidorUDP servidor_udp.c -lsocket
```

Diretrizes

1. Somente os 4 arquivos especificados, além do Makefile, precisam ser submetidos. Nenhum arquivo de cabeçalho ou arquivo de saída deve ser enviado.
2. Seu servidor não deve produzir nada no terminal e seu cliente deve ter exatamente a mesma interface que a do exemplo acima.
3. Seu código cliente (UDP/TCP) será testado com um código servidor (UDP/TCP) de outro aluno. O mesmo vale para seu código servidor. O código teste será selecionado aleatoriamente. O primeiro código selecionado será testado isoladamente e, caso esteja executando corretamente, servirá como teste para o seguinte. Se o seguinte estiver 100% correto, servirá como teste do próximo. Caso contrário, o código teste atual permanece.
4. Um bom código não implica em inúmeras linhas de codificação. O objetivo é ser eficiente, elegante e sucinto.
5. Você pode usar trechos de código da *Internet* para ajudá-lo a realizar a tarefa, desde que sejam devidamente referenciados (isto é, comentários no código e *link* de acesso). Trechos utilizados sem referência serão considerados plágio. Utilizar mais de 25% de códigos de terceiros não é permitido.
6. Por onde começar? Consulte o livro disponibilizado no *Moodle*: ***TCP/IP Sockets in C: Practical Guide for Programmers***.