Programação Utilizando Sockets

Dr. Osmar Marchi dos Santos

Socket?

- Define uma interface, abstracão, entre a aplicação e a rede

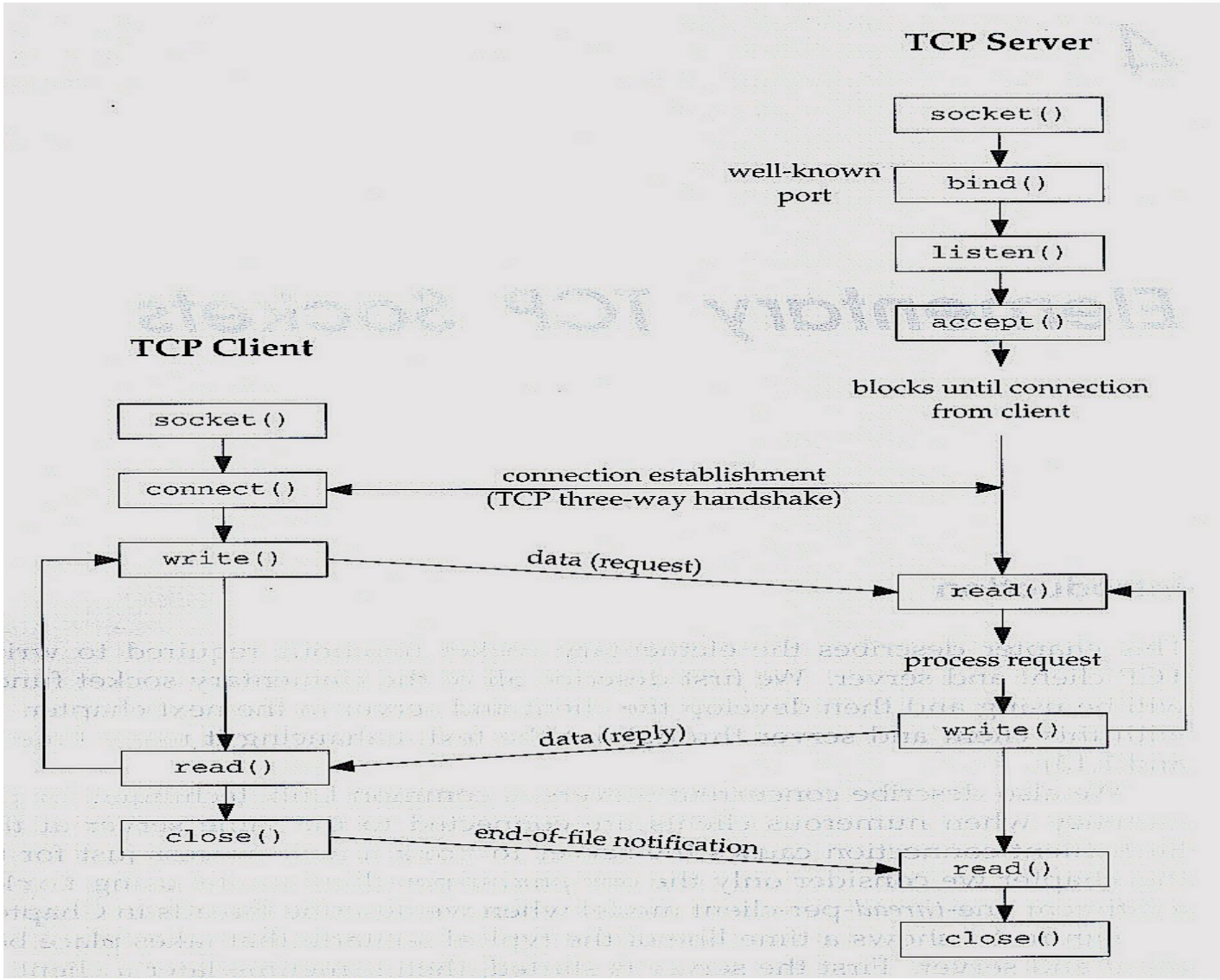
- Uma aplicação cria um socket por onde passam os dados a serem trafegados pela rede
  - O tipo do socket especifica a comunicação a ser utilizada, por exemplo (considerando o protocolo TCP/IP):
    - Sem conexão (UDP)
    - Orientada à conexão (TCP)
  - Após a configuração do socket é possível receber e enviar dados pela rede para diferentes máquinas (*hosts*)

Tipos básicos de sockets

- SOCK_STREAM
  - Como o nome diz (*stream*), define sockets que utilizam o protocolo TCP
  - Com garantias de recepção
  - Com ordem
  - Orientado à conexão

- SOCK_DGRAM
  - Como o nome diz (*datagram*), define sockets que utilizam o protocolo UDP
  - Sem garantias de recepção
  - Sem ordem
  - Sem noção de conexão

Uso de sockets



Criação de socket em C

- Função socket:
  - **int sock = socket(domain, type, protocol);**
  - **sock**: descritor do socket criado pelo SO, contém um número inteiro que identifica o socket, atua como um descritor de arquivo
  - **domain**: especifica o domínio da comunicação, por exemplo AF_INET (significa o protocolo IPv4)
  - **type**: tipo da comunicação, como visto antes (SOCK_STREAM, SOCK_DGRAM)
  - **protocol**: protocolo a ser utilizado (IPPROTO_TCP, IPPROTO_UDP). Deve ser compatível com o **type**, descrito acima.

- Exemplo:
  - sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

Bind do socket em C

- Para um socket comunicar, é preciso especificar onde ele se encontra (como o código postal e número de casa em uma cidade):
  - **Endereço IP:** Endereço onde se encontra a aplicação a se conectar (127.0.0.1 ou localhost é especial para a mesma máquina)
  - **Porta:** Utilizar portas entre 1024 e 49151 (portas 1 a 1023 são reservadas e portas entre 49152 até 65535 são dinâmicas, usadas pelo SO e podem gerar conflito)
  - **Socket:** nesse contexto, o socket é a abstração para acessar corretamente o endereço da casa

- Exemplo:
  - sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

Bind do socket em C

- A função bind associa um socket com uma porta para uso:
  - **int status = bind(sock, addrport, size);**
  - **status**: retorna o status da função, 0 sucesso, -1 erro
  - **sock**: socket a ser associado (criado anteriormente)
  - **addrport**: estrutura contendo o endereço do socket
  - **size**: tamanho em bytes da estrutura addrport

- Addrport (estrutura do tipo sockaddr_in):
  - sin_family: família do endereço (AF_INET)
  - sin_port: número da porta
  - sin_addr: endereço IP
  - sin_zero: não-utilizado

- Exemplo:
  - sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

Bind do socket em C

- Exemplo:
  - // Declara a estrutura a ser usada, falta preenchê-la
  - struct sockaddr_in address;
  - // Associa a família do endereço
  - address.sin_family = AF_INET;
  - // Associa a porta (htons transforma o valor em network byte order, uso de big-endian – parte mais significante a frente)
  - address.sin_port = htons(1024);
  - // INADDR_ANY possibilita utilizar o endereço corrente sem saber o mesmo
  - address.sin_addr.s_addr = INADDR_ANY;
  - // Uso da função bind
  - if (bind(sock, (struct sockaddr *) &address, sizeof(address)) == -1) {
    - printf(“Erro!\n”);
    - return -1;
  - }

Configuração da conexão

- Uma conexão contêm duas entidades, como um modelo cliente-servidor
  - O servidor fica a espera (listen) de conexões
  - O cliente entra em contato com o servidor para estabelecer a conexão

- Uma vez estabelecida a conexão, ambos podem enviar e receber dados, assim como terminar as conexões

- Servidor: espera -> aceita -> comunica
- Cliente: conecta e estabelece conexão -> comunica

Configuração da conexão - listen

- A função listen prepara um socket para aceita conexões:
  - **int status = listen (sock, queuelen);**
  - **status**: retorna o status da função, 0 sucesso, -1 erro
  - **sock**: socket sendo usado
  - **queuelen**: número máximo de participantes aguardando na fila para serem atendidos por sock (a quantidade máxima de conexões ativas por porta depende do SO, por exemplo, número máximo de descritores abertos – ver a seguir *accept()*)

- Exemplo:
  - if (listen(sock, 1) == -1) {
    - printf(“Erro\n”);
    - return -1;
  - }

Configuração da conexão - accept

- Uma vez que foi realizado o bind e o listen, a função accept executa a espera (função bloqueante) de uma nova conexão:
  - **int new_sock = accept (sock, &cliaddr, &addrlen);**
  - **new_sock**: retorna um novo socket utilizado para a transferência de dados
  - **cliaddr**: estrutura de endereço do cliente que acabou de conectar
  - **addrlen**: tamanho em bytes da estrutura cliente

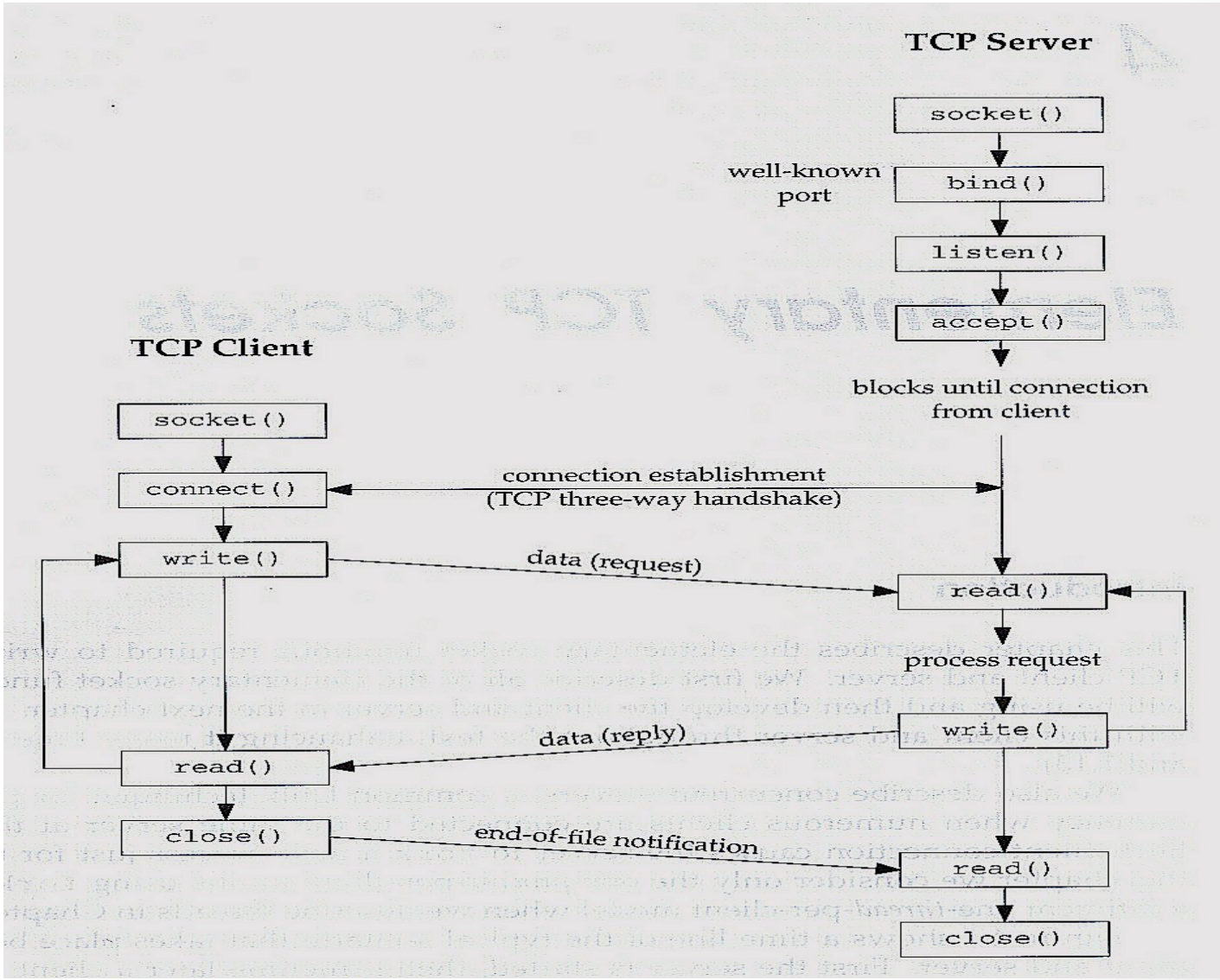
- Exemplo:
  - new_sock = accept(sock, (struct sockaddr *)&address, (socklen_t *) &addressSize);
  - // A partir daqui, o socket new_sock pode ser usado para transferência de dados

Cliente

- Do ponto de vista do cliente, os passos a serem realizados incluem: criação do socket (já visto) e conexão com a máquina remota (função connect):
  - **int status = connect(sock, &servaddr, adrlen);**
  - **status**: retorna o status da função, 0 sucesso, -1 erro
  - **sock**: socket sendo usado na conexão
  - **servaddr**: estrutura contendo o endereço do servidor
  - **addrlen**: tamanho em bytes da estrutura do servidor

- Exemplo:
  - if (connect(sock, (struct sockaddr *)&address, sizeof(address)) == -1) {
    - printf("Erro\n");
    - return -1;
  - }

Enviando e recebendo dados



Enviando e recebendo dados

- Uma vez que dois sockets (entre o cliente e servidor estão conectados), podemos utilizar funções para receber e enviar informações
- Para isso, o socket funciona como se fosse um descritor de arquivos, com funções de escrita e leitura

- int count = send(int sock, const void *msg, int len, unsigned int flags);
  - **count**: número de bytes transmitidos, -1 erro
  - **sock**: socket sendo usado na conexão
  - **msg**: dados a serem enviados
  - **len**: tamanho em bytes dos dados a serem enviados
  - **flags**: opções especiais, utilizaremos 0

- int count = recv(int sock, void *msg, int len, unsigned int flags);
  - **count**: número de bytes transmitidos, 0 não recebeu nada (outra ponta fechou a conexão)
  - **sock**: socket sendo usado na conexão
  - **msg**: buffer de dados recebidos
  - **len**: tamanho em bytes dos dados a serem enviados
  - **flags**: opções especiais, utilizaremos 0

Enviando e recebendo dados

- Exemplo:

- // envia dados (n bytes enviados, -1 erro)
- n = send(sock, buffer, strlen(bufffer), 0);

- // recebe dados (n bytes recebidos, 0 fechou a conexão)
- n = recv(sock, buffer, 255, 0);

Fechando conexões

- Após o uso da conexão, a última etapa do processo é fechar as conexões. Como um arquivo, utiliza-se a função close.

- int status = close(sock);

- Para utilizar sockets, precisamos utilizar algumas bibliotecas:
- #include <sys/types.h>
- #include <sys/socket.h>
- #include <netdb.h>
- #include <arpa/inet.h>
- #include <netinet/in.h>

Exercícios

- Criar um servidor de “echo”, onde o cliente envia uma frase e a mesma é enviada de volta e impressa na tela.

- Ver exemplo no Moodle e função gethostbyname(char *name, int nameLen).

- Link interessante:
- http://www.beej.us/guide/bgnet/output/html/multipage/index.html