Terça, 24/02/2009

Programação

Tutorial de Sockets - Parte III

Por: Frederico Perim

Como antes, isso é muito para absorver de uma vez, então segue um fragmento de código:

```
    // a porta que usuários estarão se
conectando

    //quantas conexões pendentes serão
suportadas

    main()
    {

        int sockfd, novo_fd; //ouve em sockfd, novas conexões
em novo_fd

        struct sockaddr_in meu_end; //informação do meu
endereço

        struct sockaddr_in outro_end; //informação do
endereço do usuário
        int sin_size;

        sockfd = socket(AF_INET, SOCK_STREAM, 0); //checagem de
erro

        meu_end.sin_family = AF_INET; //host byte order
        meu_end.sin_port = htons(3490 ); // converte para short,
network byte order
        meu_end.sin_addr.s_addr = INADDR_ANY; //preenche com meu IP
        memset(&(meu_end.sin_zero), '\0', 8); //zera o resto da
estrutura

        //não esquecer de fazer checagem de erros para essas
chamadas

        bind (sockfd, (struct sockaddr *)&meu_end,
sizeof(struct sockaddr));

        listen(sockfd, 10 );

        sin_size = sizeof(struct sockaddr_in);

        novo_fd = accept(sockfd, &outro_end, &sin_size);
```

Novamente note que nós usaremos outro_end para todas as chamadas send() e recv().Se você não vai servir novas conexões , você pode fechar (close()) sockfd para cancelar outras conexões na mesma porta, se assim desejar.

send() e recv() - Fale comigo!

Estas duas funções servem para se comunicar através de "SOCK_STREAM". Se você deseja usar datagram sockets ("SOCK_DGRAM") , então você vai usar sendto() e recvfrom(), abaixo.

A chamada send():

```
int send(int sockfd, const void *msg, int len, int flags);
```

sockfd é nosso velho conhecido descritor de socket para onde você envia os dados (seja um retornado por socket() ou por accept()). msg é um ponteiro para os dados que você deseja enviar, e len é o tamanho dos dados em bytes. Ajuste flags como 0. (Consulte o manual de send() para mais informações a respeito de flags).

Um código ilustrativo seria:

```
char *msg = "Oi tudo bem?";
int tamanho, bytes_enviados;
.
.
.
tamanho = strlen(msg);
bytes_enviados = send(sockfd, msg, tamanho, 0);
.
.
.
```

send() retorna o número de bytes realmente enviados - isso pode ser menos do que você mandou enviar. Algumas vezes você envia um monte de dados e send() não pode dar conta. Assim ela vai enviar o máximo possível , e confiar que você mande o resto mais tarde. Lembre-se, se o valor retornado por send não for igual ao valor de tamanho, depende de você mandar o resto da string. A boa notícia: se o pacote for pequeno (menor que 1K), a função provavelmente vai dar um jeito de enviar tudo de uma vez.

A função recv é similar em muitos aspectos:

```
int recv(int sockfd, void *buf, int len, unsigned int flags);
```

sockfd é o descritor de onde você lê os dados, buf é onde você lê as informações contidas em buffer, len é o tamanho máximo de buffer, e flags poder ser 0.

recv() retorna o número de bytes realmente lidos em buffer, ou -1 em caso de erro.

Espere! recv() pode retornar zero. Isso significa somente uma coisa: o host remoto fechou sua conexão! Um valor de retorno 0 é a forma que recv() utiliza para você saber que isso aconteceu.

Ufa! Foi fácil, não? Agora você pode receber e enviar dados através de "stream sockets".

«**Anterior** **Próximo**»

listen() - Tem alguém me chamando?
4.5. accept() - "Obrigado por chamar a porta 3490."
send() e recv() - Fale comigo!
sendto() e recvfrom() -- Fale comigo no estilo DGRAM

Enviar para um amigo | Imprimir | Índice de artigos