

Publicidade



ASSINE 0800 703 3000

BATE-PAPO

E-MAIL

SAC

Voip

E-Mail Grátis

Shopping

ÍNDICE PRINCIPAL

PROCURAR:

no site

OK

Terça, 24/02/2009

- » Introdução
- » Programação
- » Administração
- » Hardware
- » Aplicativos
- » Jogos
- » Segurança
- » Editorial
- » Entrevistas

ARTIGOS

- » Fórum
- » Links
- » Notícias
- » Pegue o Linux
- » Documentação

COMUNIDADE

- » Programas
- » Dúvidas
- » Oportunidades
- » Sobre
- » Contato
- » Publicidade

SERVIÇOS

Powered By:  
**DEBIAN**  
**GNU/LINUX**

English Version

**Linux Solutions**  
**Shopping**  
**OLinux**



## Programação

### Tutorial de sockets - Parte VI

Por: [Frederico Perim](#)

Este programa funciona como um servidor de chat multi-usuário simples. Rode-o em uma janela, e então conecte-se através do telnet ("telnet nomedohost 9034") de várias janelas diferentes. Quando você digita uma mensagem em uma janela, ele deverá aparecer em todas as outras.

```
// porta em que estaremos escutando

int main (void)
{
    fd_set mestre; // descritor de arquivo mestre
    fd_set read_fds; // descritor de arquivo
    temporário para select()
    struct sockaddr_in meu_end; //endereço do servidor
    struct sockaddr_in end_remoto; // endereço do
    cliente
    int fdmax; // número máximo do descritor de
    aequivo
    int listener; // descritor de socket ouvindo
    char buf[256]; // buffer com os dados do cliente
    int nbytes;
    int yes=1; // necessário para função
    setsockopt() SO_REUSEADDR, abaixo
    int addrlen;

    FD_ZERO(&mestre); //limpa os conjuntos mestre e
    temporário
    FD_ZERO(&read_fds);

    // pega o descritor de escuta
    if ((listener = socket (AF_INET, SOCK_STREAM, 0)) == -1) {
        perror ("socket");
        exit (1);
    }

    // evita a mensagem de erro "endereço já em uso"
    if (setsockopt(listener, SOL_SOCKET, SO_REUSEADDR,
    &yes, sizeof (int)) == -1) {
        perror ("setsockopt");
        exit (1);
    }

    // conecta
    meu_end.sin_family = AF_INET;
```

#### ENQUETE

Com qual frequência você  
acessa o site Olinux?

- ☐ Todos os dias
- ☐ Uma vez por semana
- ☐ Cinco vezes aos mês
- ☐ Poucas vezes ao mês
- ☐ Outra

VOTAR

#### NEWSLETTER

Inscra-se e receba as últimas  
notícias, programas, artigos,  
novidades e tudo do mundo  
Linux que aconteceu na semana.

Digite seu email:

OK

**Jaguar**

Vários modelos.  
Entre e confira.

**Auto DVD  
Player**

Diversas marcas e  
modelos. Encontre  
em até 12x sem  
juros.

**Filmadora**

Multilaser CR-518  
Digital.  
Compare!

**Esteira**

Entre em forma  
antes do verão.

[COMPARE PREÇOS](#)

```

meu_end.sin_addr.s_addr = INADDR_ANY;
meu_end.sin_port = htons (9034 );
memset (&(meu_end.sin_zero), '\0', 8);
if (bind (listener, (struct sockaddr *)&meu_end,
sizeof(meu_end)) == -1) {

    perror("bind");
    exit(1);

}

//escuta
if (listen(listener, 10) == -1) {

    perror("listen");
    exit(1);

}

//adiciona o socket em escuta ao conjunto master
FD_SET(listener, &master);

// rastreia o maior descritor de arquivo

fd_max = listener; // até agora, este é o maior

// loop principal

fot (;;) {

    read_fds = master; // copia
    if (select (fdmax+1, &read_fds, NULL, NULL, NULL)

== -1) {

        perror ("select");
        exit(1);

    }

    // percorre as conexões existentes em busca de
    dados

    fot (i = 0; i &lt;= fdmax; i++) {

        if ( FD_ISSET ( i, &read_fds)) { nós
        encontramos uma!!!

            If ( i == listener) {

                //trata novas conexões
                addrlen = sizeof (end_remoto);

                if (( newfd = accept(listener,
&end_remoto,
                    &addrlen)) == -1) {

                    perror ("accept");
                } else {

                    FD_SET ( newfd , &mestre);

                    If ( newfd &gt; fdmax) { //

                        fdmax = newfd;

                    }

                    printf ("servidor : nova conexão
de %s no socket %d\n",
                        inet_ntoa (end_remoto.sin_addr),
newfd);

                }
            } else {

                // cuida dos dados do cliente
                if (( nbytes = recv (i, buf, sizeof
(buf), 0)) &lt;= 0) {

                    // recebeu erro ou a conexão foi
                    fechada pelo cliente

```

```

                                if (nbytes == 0) {
                                    // conexão encerrada
                                    printf ("servidor: socket %d
desligado\n", i);
                                } else {
                                    perror ("recv");
                                }
                                close (i); // tchau!!
                                FD_CLR( i , &master); // remove do
conjunto mestre
                                } else {

                                    // temos alguns dados do cliente
                                    for (j = 0; j <= fdmax; j++) {

                                        //envia a todos os clientes
                                        if (FD_ISSET (j, &master)) {
                                            // exceto a nós e ao
socket em escuta
                                        if (j != listener && j
!= 1) {
                                            if (send ( j, buf, nbytes,
0) == -1) {
                                                perror ("send");
                                            }
                                        }
                                    }
                                } // Isso é muito feio!!
                            }
                        }
                    }
                }

                return 0;
            }

```

Note que eu tenho dois conjuntos de descritores de arquivo: mestre e read\_fds. O primeiro, mestre, tem todos os descritores de socket que estão conectados, assim como o descritor socket que está escutando novas conexões.

A razão pela qual eu tenho um conjunto mestre é que select() na verdade muda o conjunto que você passa a ele para refletir quais sockets estão prontos para ler ( receber dados ). Já que eu tenho que verificar as conexões de uma chamada select() para outra, eu tenho que armazená-los em algum lugar seguro. No último minuto , eu copio o mestre em read\_fds , e então chamo select().



Mas isso não significa que toda vez que tenho uma nova conexão, não tenho que acrescentá-la ao conjunto mestre? Sim! E toda vez que uma conexão é encerrada, eu tenho que retirá-la do conjunto mestre? Sim, tem.


Note que eu checo para ver quando o socket que esta escutando ( listener ) está pronto para receber dados. Caso afirmativo, significa que tenho uma nova conexão pendente , então eu aceito ( accept() ) e adiciono ao conjunto mestre. Da mesma forma , quando uma conexão do cliente está pronta para receber, e recv() retorna 0, eu sei que o cliente encerrou a conexão, então tenho que removê-lo do conjunto mestre.

Se , no entanto, a chamada recv() do cliente retornar um valor diferente de zero, eu sei que algum dado foi recebido. Então eu percorro toda lista mestre e envio o dado para o resto dos clientes.

É isso aí caros amigos, aqui termina nossa análise da função select().

[«Anterior](#)

 **Técnicas Avançadas**  
 **Blocking**  
 **select()**

  
 Enviar para um amigo

  
 Imprimir

  
 Índice de artigos

Copyright (c) 2000-2007, OLinux - O Portal de Linux do Brasil.

Desenvolvido por: [Linux Solutions](#)

Todos os Direitos Reservados.