API de Sockets

José Pedro Oliveira
(jpo@di.uminho.pt)

Grupo de Sistemas Distribuídos
Departamento de Informática
Escola de Engenharia
Universidade do Minho

Sistemas Operativos I
2006-2007

Conteúdo

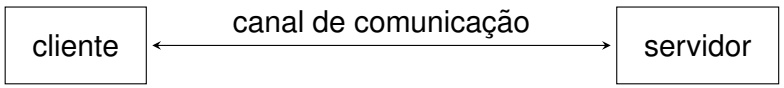
Modelo Cliente/Servidor (1/2)

Arquitectura cliente/servidor

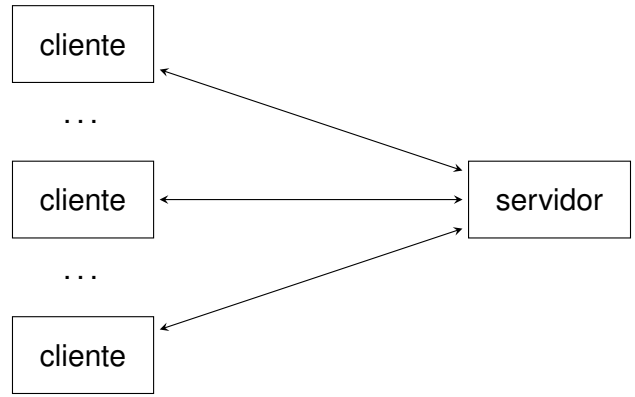
A troca de informação é baseada em mensagens.

servidor - Fornecedor de serviços. Tipicamente está à espera de ser contactado por clientes interessados nos seus serviços.

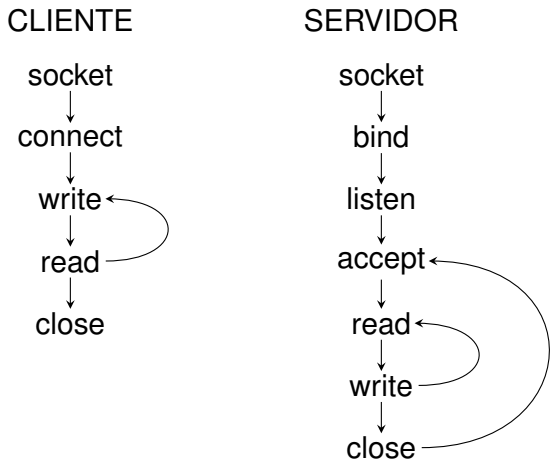
cliente - Consumidor de serviços. Toma a iniciativa de contactar um servidor para obter um serviço.



Modelo Cliente/Servidor (2/2)



Sequência de chamadas



Sockets . . .

Caracterização

Os seguintes cinco parâmetros permitem caracterizar um socket:

- endereço IP da origem
- porta da origem
- endereço IP do destino
- porta do destino
- protocolo

Protocolos - introdução sumária

Protocolos

- UDP - User Datagram Protocol
- TCP - Transmission Control Protocol

UDP - User Datagram Protocol

- análogo ao sistema postal
- permite comunição em grupo (1 para n)

TCP - Transmission Control Protocol

- análogo ao sistema telefónico
- permite apenas ligações ponto-a-ponto

TCP/IP: serviços standard

TCP/IP: serviços standard

Nome	TCP	UDP	RFC	Descrição
echo	7	7	862	O servidor retorna o que o cliente lhe envia.
discard	9	9	863	O servidor ignora o que o cliente lhe envia.
daytime	13	13	867	O servidor retorna a hora e a data em formato legível.
chargen	19	19	864	O servidor TCP envia continuamente uma *stream* de caracteres até o cliente se desligar. O servidor UDP envia um datagrama contendo um número de caracteres aleatórios cada vez que o cliente envia um datagrama.
time	37	37	868	O servidor envia o tempo como um número de 32-bits. Este número representa o número de segundos desde a meia-noite de 1 de Janeiro de 1900 UTC.

Network Information

Tipos de informação relacionada com a rede

Informação	Ficheiro	Estrutura	Funções
hosts	/etc/hosts	hostent	gethostbyaddr, gethostbyname
networks	/etc/networks	netent	getnetbyaddr, getnetbyname
protocols	/etc/protocols	protoent	getprotobyname, getprotobynumber
services	/etc/services	servent	getservbyname, getservbyport

TCP/IP: serviços standard

Ficheiro: **/etc/services**

```
...
# service-name  port/protocol  [aliases...]  [# comment]
...
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
...
```

Conteúdo

Função: **socket** (1/3)

Synopsis

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

Descrição

Cria um extremo da comunicação e retorna o descriptor a ele associado.

Função: **socket** (2/3)

Tipo

Tipo	Descrição
SOCK_STREAM	stream socket
SOCK_DGRAM	datagram socket
SOCK_RAW	raw socket

Função: **socket** (3/3)

Tipo

Família	Descrição
AF_INET	protocolos IPv4
AF_INET6	protocolos IPv6
AF_LOCAL	protocolos do domínio Unix
AF_ROUTE	sockets de routing
AF_KEY	sockets key

Função: **connect**

Synopsis

```
#include <sys/types.h>
#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr
*serv_addr, socklen_t addrlen);
```

Descrição

Se o socket for do tipo **SOCK_DGRAM** a estrutura `serv_addr` representa o endereço para o qual os datagramas são enviados por omissão e o endereço do qual aceita datagramas. Se o socket for do tipo **SOCK_STREAM**, esta função tenta estabelecer uma ligação com o socket que está associado ao endereço especificado por `serv_addr`.

Função: **bind**

Synopsis

```
#include <sys/types.h>
#include <sys/socket.h>

int bind(int sockfd, struct sockaddr *my_addr,
socklen_t addrlen);
```

Descrição

Associa um nome ao socket.

Função: **listen**

Synopsis

```
#include <sys/socket.h>

int listen(int s, int backlog);
```

Descrição

Fica à espera de ligações ao socket.

Função: **accept**

Synopsis

```
#include <sys/types.h>
#include <sys/socket.h>

int accept(int s, struct sockaddr *addr, socklen_t
*addrlen);
```

Descrição

Aceita ligações ao socket.

Função: **shutdown**

Synopsis

```
#include <sys/socket.h>

int shutdown(int s, int how);
```

Descrição

Permite fazer shutdown parcial de uma ligação full-duplex.

Função: **close**

Synopsis

```
#include <unistd.h>

int close(int fd);
```

Descrição

Fecha um socket (descriptor de ficheiro).

Função: **getsockname**

Synopsis

```
#include <sys/socket.h>

int getsockname(int s,
    struct sockaddr *name,
    socklen_t *namelen);
```

Descrição

Permite obter o nome do socket (endereço e porta).

Função: **getpeername**

Synopsis

```
#include <sys/socket.h>

int getpeername(int s,
    struct sockaddr *name,
    socklen_t *namelen);
```

Descrição

Permite obter o nome do socket remoto ligado (peer).

Funções: **htons**, **htonl**, **ntohs**, **ntohl**

Synopsis

```
#include <arpa/inet.h>

uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

Descrição

A transmissão de informação binária deverá ser efectuada em **Network Byte Order**. Para tal existe um comjunto de funções que permitem efectuar a conversão de **Host Byte Order** para **Network Byte Order** (htonx) e vice-versa (ntohx).

Conteúdo

Serviço tipo daytime

Exercício

Implementação de um cliente e de um servidor de um serviço equivalente ao daytime, ou seja, em que a uníca accção que desencadeia a resposta do servidor é o estabelecimento de uma ligação.

Cliente TCP (1/2)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>            /* memset */
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/socket.h>
7  #include <netinet/in.h>        /* struct sockaddr_in */
8  #include <arpa/inet.h>         /* inet_addr */
9
10 int main(int argc, char *argv[])
11 {
12     int cli_sock;
13     struct sockaddr_in srv_addr;
14     char ch;
15
16     if (argc != 3) {
17         printf("uso: tcpcli <endereco_ip> <porta>\n");
18         exit(EXIT_FAILURE);
19     }
```

Cliente TCP (2/2)

```
20     cli_sock = socket(PF_INET, SOCK_STREAM, 0);
21
22     /* Endereco e porto do servidor */
23     memset(&srv_addr, 0, sizeof(struct sockaddr_in));
24     srv_addr.sin_family = AF_INET;
25     srv_addr.sin_port = htons(atoi(argv[2]));
26     srv_addr.sin_addr.s_addr = inet_addr(argv[1]);
27
28     /* Estabelecer ligacao */
29     connect(cli_sock, (struct sockaddr *) &srv_addr,
30                       sizeof(srv_addr));
31
32     while ( read(cli_sock, &ch, 1) > 0 ) {
33         write(STDOUT_FILENO, &ch, 1);
34     }
35
36     close(cli_sock);
37     return 0;
38 }
```

Servidor TCP (1/3)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>            /* memset */
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/socket.h>
7  #include <netinet/in.h>        /* struct sockaddr_in */
8
9  int main(int argc, char *argv[])
10 {
11     int srv_sock, cli_sock;
12     struct sockaddr_in srv_addr, cli_addr;
13     socklen_t cli_addr_size;
14     char msg[] = "Sistemas Operativos\n";
15
16     if (argc != 2) {
17         printf("uso: tcpsrv <porta>\n");
18         exit(EXIT_FAILURE);
19     }
```

Servidor TCP (2/3)

```
20      srv_sock = socket(PF_INET, SOCK_STREAM, 0);
21
22      /* Enderecos e porta de escuta */
23      memset(&srv_addr, 0, sizeof(struct sockaddr_in));
24      srv_addr.sin_family = AF_INET;
25      srv_addr.sin_port = htons(atoi(argv[1]));
26      srv_addr.sin_addr.s_addr = INADDR_ANY;
27
28      bind(srv_sock, (struct sockaddr *) &srv_addr,
29                     sizeof(struct sockaddr_in));
30
31      listen(srv_sock, 5);
32
33      cli_addr_size = sizeof(struct sockaddr_in);
```

Servidor TCP (3/3)

```
34
35      while((cli_sock = accept(srv_sock,
36              (struct sockaddr *) &cli_addr, &cli_addr_size))) {
37
38          write(cli_sock, msg, strlen(msg));
39
40          shutdown(cli_sock, 2);
41          close(cli_sock);
42      }
43
44      close(srv_sock);
45
46      return 0;
47  }
```

Conteúdo

TCP/IP: serviços standard

TCP/IP: serviços standard

Nome	TCP	UDP	RFC	Descrição
echo	7	7	862	O servidor retorna o que o cliente lhe envia.
discard	9	9	863	O servidor ignora o que o cliente lhe envia.
daytime	13	13	867	O servidor retorna a hora e a data em formato legível.
chargen	19	19	864	O servidor TCP envia continuamente uma *stream* de caracteres até o cliente se desligar. O servidor UDP envia um datagrama contendo um número de caracteres aleatórios cada vez que o cliente envia um datagrama.
time	37	37	868	O servidor envia o tempo como um número de 32-bits. Este número representa o número de segundos desde a meia-noite de 1 de Janeiro de 1900 UTC.

xinetd

Serviço

```
service xinetd start
```

Configuração

Ficheiro: `/etc/xinetd.conf`
Directório: `/etc/xinetd.d/`

xinetd: activar serviço **daytime**

/etc/xinetd.d/daytime

```
1  # default: off
2  # description: An internal xinetd service which gets the current \
3  # system time then prints it out in a format like this: \
4  #    "Wed Nov 13 22:30:27 EST 2002". \
5  # This is the tcp version.
6
7  service daytime
8  {
9      type          = INTERNAL
10     id            = daytime-stream
11     socket_type   = stream
12     protocol      = tcp
13     user          = root
14     wait          = no
15     disable       = no
16 }
```

MacOSX: Simple TCP/IP Services

An Unofficial Xinetd Tutorial `http://www.macsecurity.org/resources/xinetd/tutorial.shtml`

Serviço **daytime**

```
service daytime start
```

Serviço **chargen**

```
service chargen start
```

Windows: Simple TCP/IP Services

Instalação

1. Control Panel
2. Add or Remove Programs
3. Windows Components
4. Networking services
5. Simple TCP/IP Services

Serviço

1. Administrative Tools
2. Services
3. Simple TCP/IP Services

Conteúdo

Referências

Bibliografia

- **UNIX Network Programming, Volume 1 (terceira edição) Networking APIs: Sockets and XTI**
  Autores: W. Richard Steven, Bill Fenner, Andy Rudoff
  `http://www.unpbook.com/`
- **TCP/IP Illustrated, Volume 2: The Implementation**
  Autores: Gary R. Wright, W. Richard Steven
  `http://www.kohala.com/start/tcpipiv2.html`
- **Internetworking With TCP/IP, Volume III: Client-Server Programming and Applications (BSD Socket Version)**
  Autores: Douglas E. Commer, David L. Stevens
  `http://www.cs.purdue.edu/homes/dec/netbooks.html`