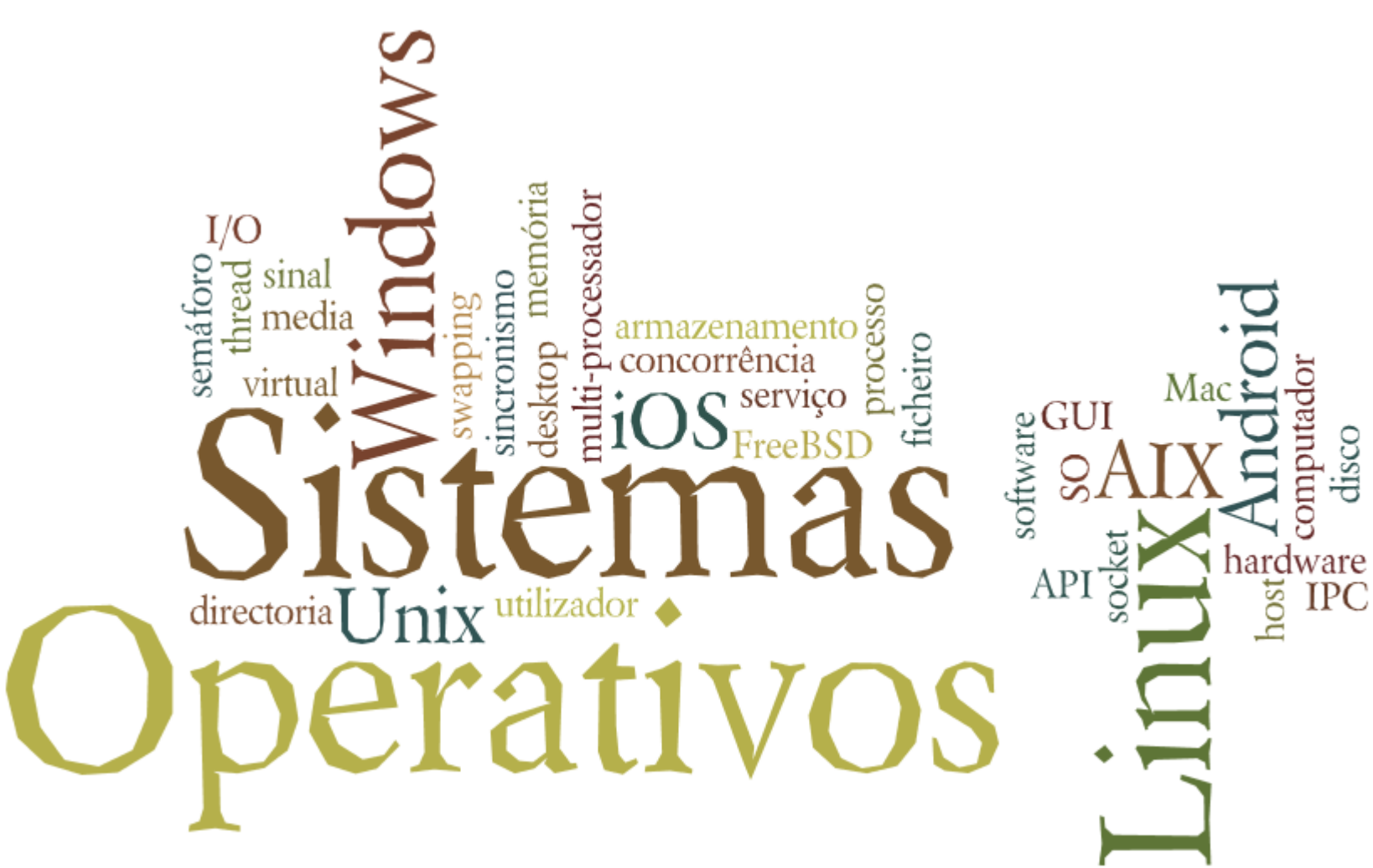


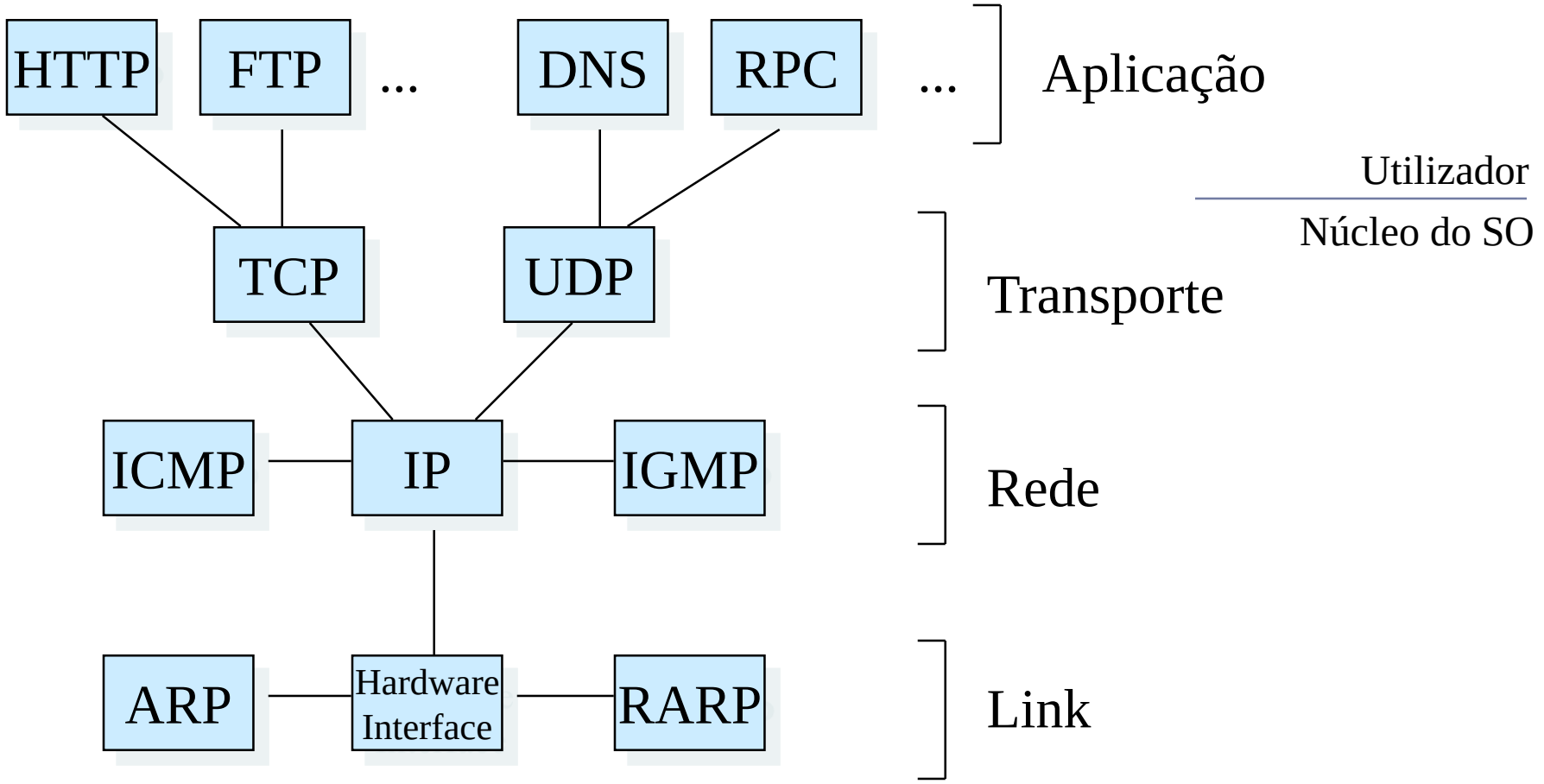

# Programação com Sockets

## Sistemas Operativos

2016/2017

Valentim Realinho

# Protocolos da Internet



# Unix BSD Sockets

- Interface padrão para comunicação entre processos nas redes TCP/IP
- Nasceu com o Unix de Berkeley
- Tentou-se usar ao máximo as chamadas de sistema do Unix
- Implementada hoje em vários SO
- Programar com sockets pode ser visto como **desenvolver um protocolo aplicacional**

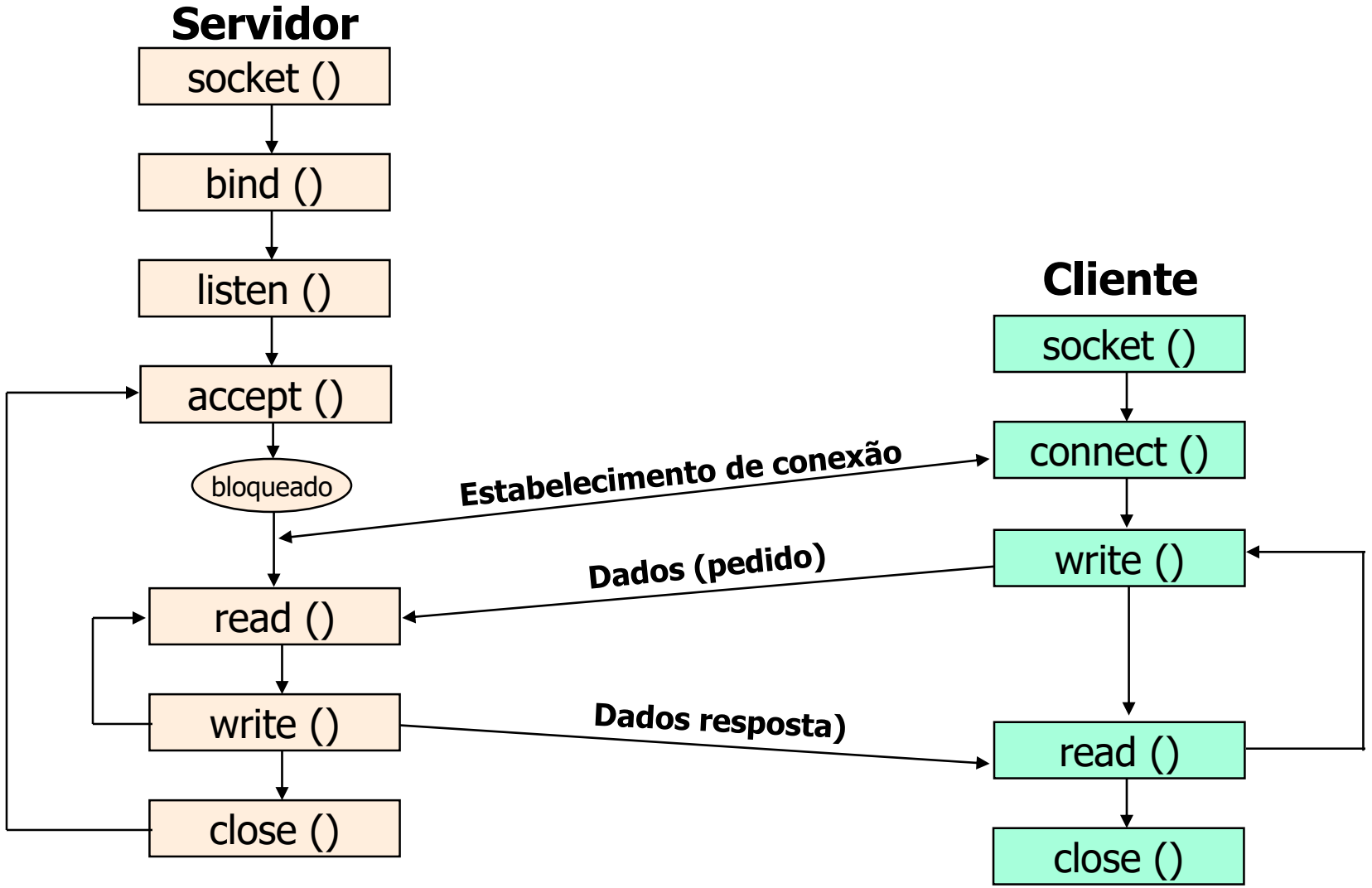
# Tipos de Sockets

- Serviço com conexão
  - Implementa um stream de dados (SOCK_STREAM)
  - Protocolo TCP
- Serviço sem conexão
  - Implementa um serviço de datagramas (SOCK_DGRAM)
  - Protocolo UDP
  - Acede directamente à camada de rede (SOCK_RAW)
- Serviço de baixo nível
  - Protocolo IP

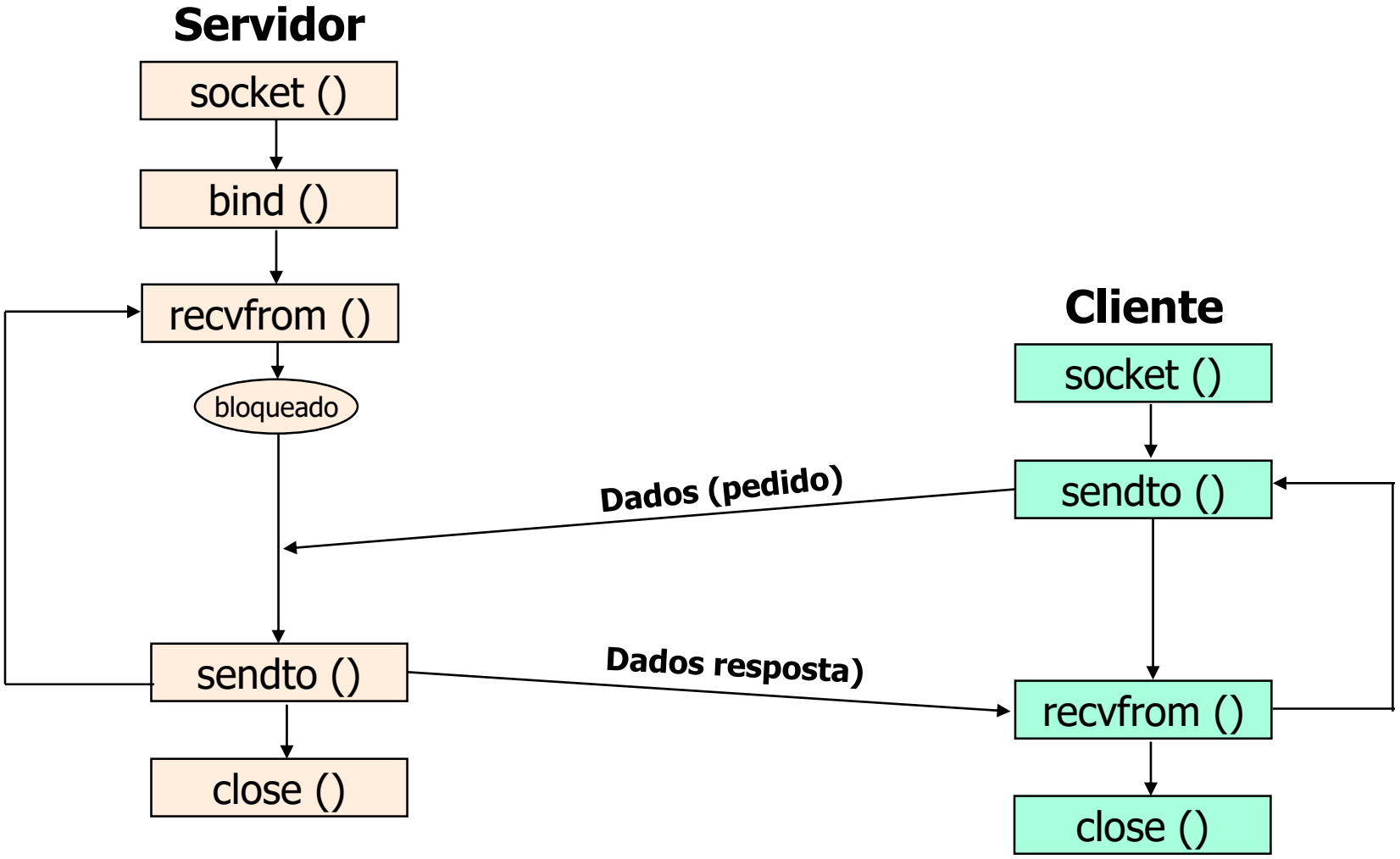
# Principais funções da API

| Função | Descrição |
|---|---|
| socket | Cria um novo descriptor para comunicação |
| connect | Iniciar uma ligação com o servidor |
| write | Escreve dados na ligação |
| read | Ler dados de um ligação |
| close | Fecha a ligação |
| bind | Atribui um endereço IP e um porto a um socket |
| listen | Coloca o socket em modo passivo, para escutar portos |
| accept | Bloqueia o servidor até chegada de pedido de ligação |
| recvfrom | Recebe um datagrama e guarda o endereço do emissor |
| sendto | Envia um datagrama especificando o endereço |

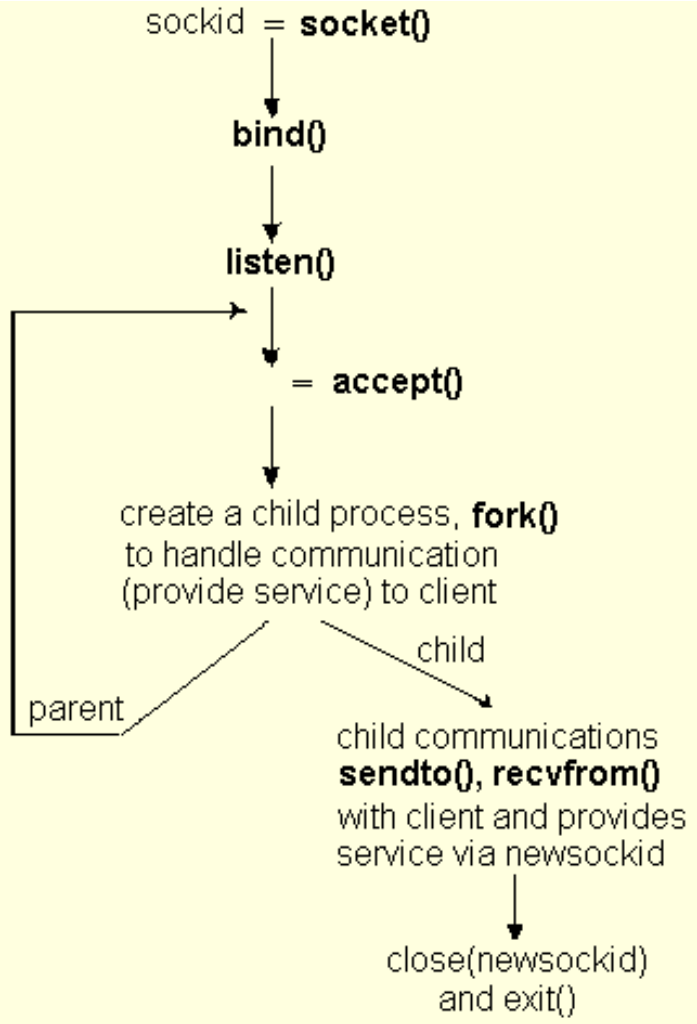
# Serviço com Conexão (TCP)



# Serviço sem Conexão (UDP)



# Estrutura Típica de um Servidor



# Portos

| | |
|---|---|
| 1-255 | Reservados para serviços padrão portos "bem conhecidos" |
| 256-1023 | Reservado para serviços Unix |
| 1-1023 | Só podem ser usados por utilizadores privilegiados (superusers) |
| 1024-4999 | Usados por processos de sistema e de utilizador |
| 5000- | Usados somente por processos de utilizador |

# Sockets em C/C++

- C é a linguagem "básica" para programação com sockets
- De maneira diferente de Java, programar com sockets em C/C++ envolve utilizar todas as chamadas da API

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

int main(int argc, char **argv)
{
   int s;
   struct sockaddr_in dest;
   char msg_write[100], msg_read[100];
   s = socket(AF_INET, SOCK_STREAM, 0));

   bzero(&dest, sizeof(dest));
   dest.sin_family = AF_INET;
   dest.sin_port = htons(9999);
   inet_aton("127.0.0.1", &dest.sin_addr.s_addr);

   connect(s, (struct sockaddr*)&dest, sizeof(dest));

   do {
        scanf("%s",msg_write);
        write (s, msg_write, strlen(msg_write)+1);
        read (s, msg_read, MAXBUF);
   } while (strcmp(msg_read,"bye"));

    close(s);

}
```

```
#include ...
#include <sys/socket.h>
int main(int argc, char **argv)
{  int s, client_s;
   struct sockaddr_in self, client;
   int addrlen = sizeof(client);
   char msg_write[100], msg_read[100];

   s = socket(AF_INET, SOCK_STREAM, 0);
   bzero(&self, sizeof(self));
   self.sin_family = AF_INET;
   self.sin_port = htons(9999);
   self.sin_addr.s_addr = INADDR_ANY;

   bind(s, (struct sockaddr*)&self, sizeof(self));
   listen(s, 5);
   while (1) {
        client_s = accept(s, (struct sockaddr*)&client, &addrlen);
        do {
               read (client_s, msg_read, MAXBUF);
               write (client_s, msg_read, strlen(msg_read)+1);
        } while (strcmp(msg_read,"bye"));
        close(client_s);
   }
}
```

# Sockets sem Conexão (C)

- Cliente:
  - `s = socket(AF_INET, SOCK_DGRAM, 0);`
  - `sendto(s, msg, length, flags, destaddr, addrlen);`
  - `recvfrom(s, msg, length, flags, fromaddr, addrlen);`
- Servidor:
  - `s = socket(AF_INET, SOCK_DGRAM, 0);`
  - `bind(s, dest, sizeof(dest));`
  - `recvfrom(s, msg, length, flags, fromaddr, addrlen);`
  - `sendto(s, msg, length, flags, destaddr, addrlen);`