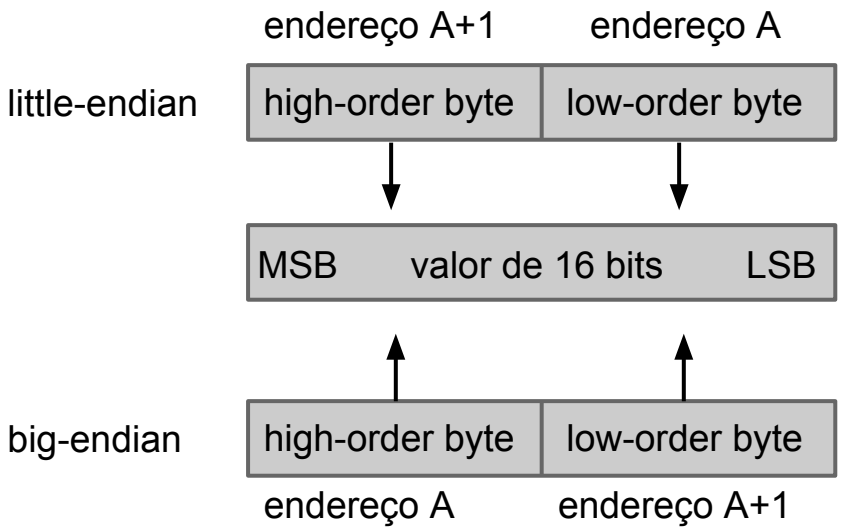
Ordem dos bytes e Sockets TCP

MC 833 – Programação em Redes de Computadores
Instituto de Computação – UNICAMP

Juliana Freitag Borin

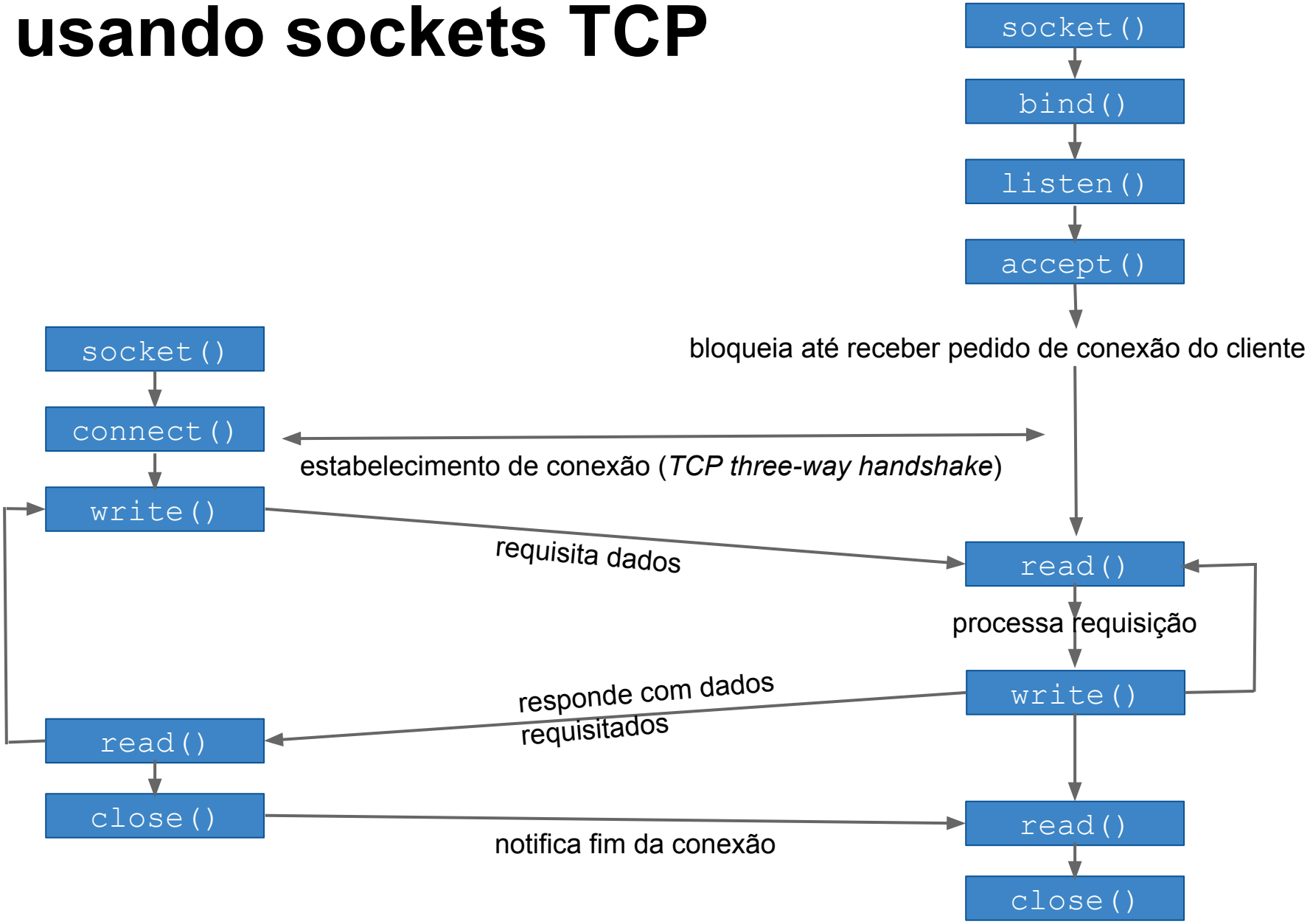
Ordem dos bytes




```
#include <netinet/in.h>

uint16_t htons (uint16_t host16bitvalue);
uint32_t htonl (uint32_t host32bitvalue);
uint16_t ntohs (uint16_t net16bitvalue);
uint32_t ntohl (uint32_t net32bitvalue);
```

Interação entre cliente e servidor usando sockets TCP



Sockets TCP - API

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

- Dado domínio/família do protocolo de rede + tipo do socket + protocolo de transporte retorna um descritor para o socket (sd).

domínio	**descrição**
AF_INET	protocolos IPv4
AF_INET6	protocolos IPv6
AF_LOCAL	protocolos Unix - cliente e servidor no mesmo nó
AF_ROUTE	sockets de roteamento
AF_KEY	sockets para interagir com mecanismos de segurança

tipo	**descrição**
SOCK_STREAM	fluxo de bytes
SOCK_DGRAM	datagramas
SOCK_SEQPACKET	datagramas via conexão
SOCK_RAW	envio de pacotes IP sem interferência da camada de transporte

Protocolo	**Descrição**
IPPROTO_TCP	Protocolo TCP
IPPROTO_UDP	Protocolo UDP
IPPROTO_SCTP	Protocolo SCTP

Sockets TCP - API

```
int bind(int sd, struct sockaddr *addr, u_int addr_len)
```

- Dado o descritor do socket (retornado pela função socket()) + endereço (formado por end. IP + num. da porta) + tamanho da estrutura de endereço, atribui um endereço local ao socket.

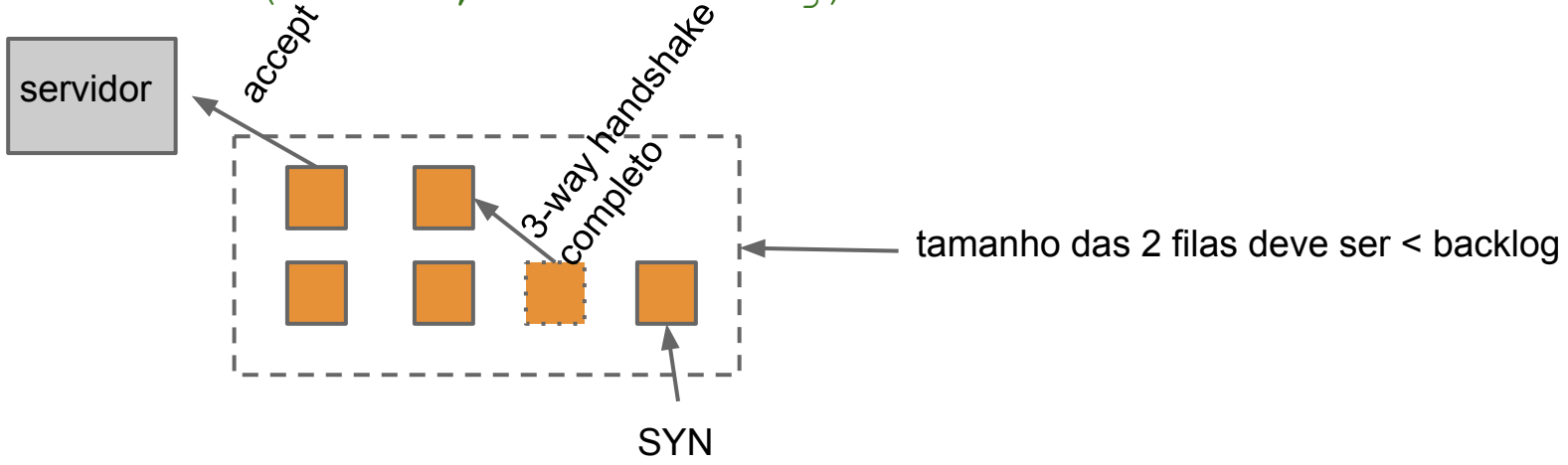
```
struct sockaddr { //socket generico
   u_short sa_len;
   u_short sa_family; /* AF_xxx */
   char sa_data[14]; /* endereço (pode ser IP+porta)*/
}
struct sockaddr_in { //socket IPv4
   u_short sin_len;
   short int sin_family; /* AF_xxx */
   u_short sin_port; /*numero da porta TCP ou UDP - 16
bits*/
   struct in_addr sin_addr; /*end. IPv4 - 32 bits*/
   char sin_zero[8]; /* não usado */
```

Sockets TCP - API

```
int connect (int sd, struct sockaddr *addr, u_int
addr_len)
```

- Conecta o socket local com a porta + endereço do destino

```
int listen(int sd, int backlog)
```




```
int accept (int sd, struct sockaddr *addr, u_int
*addr_len)
```

listening socket

endereço do cliente

novo descritor para a conexão estabelecida (servidor)

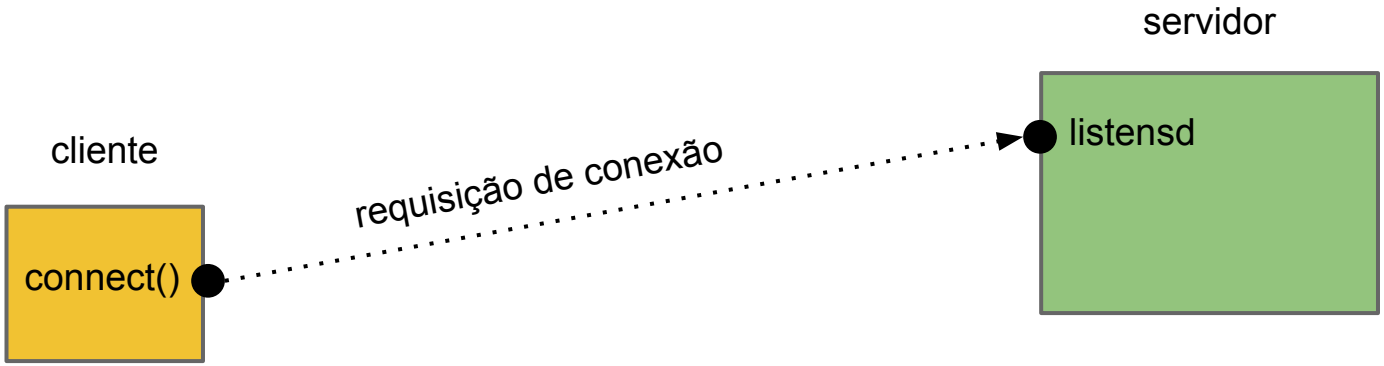
Sockets TCP - Servidor concorrente

Função *fork* (Unix): cria um novo processo (filho), que consiste em uma cópia do processo (pai) que chamou a função

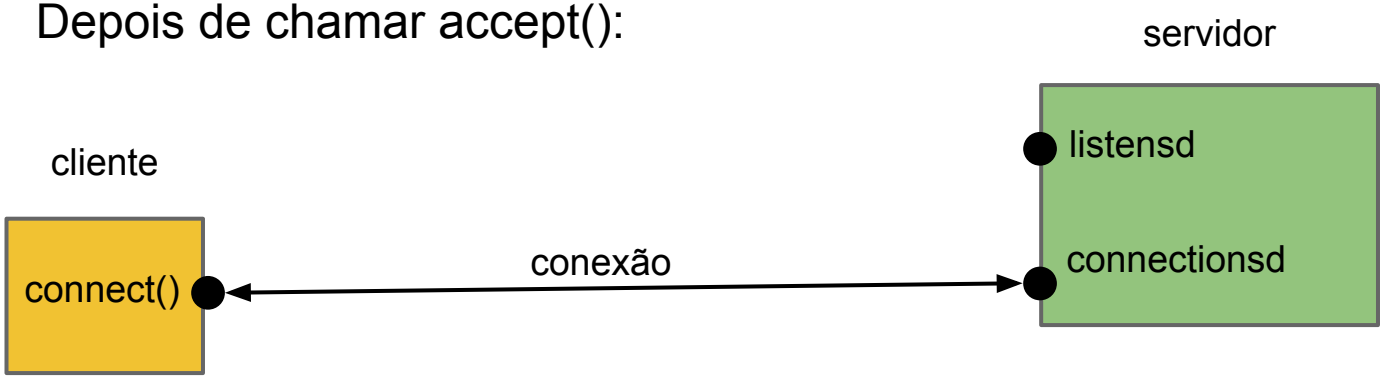
```
#include <unistd.h>
pid_t fork(void);
```

Sockets TCP - Servidor concorrente

Antes de chamar accept():

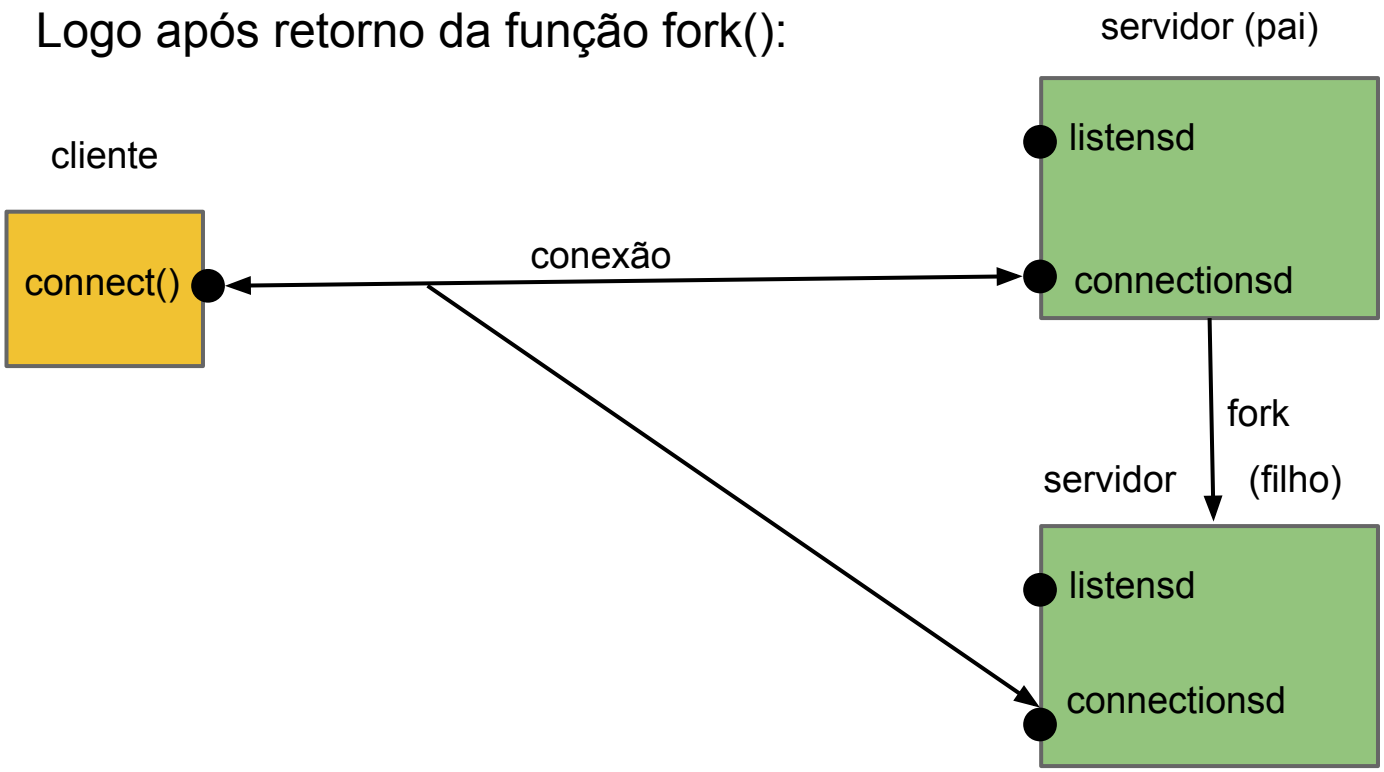


Depois de chamar accept():



Sockets TCP - Servidor concorrente

Logo após retorno da função fork():



Sockets TCP - Servidor concorrente

Depois de pai e filho finalizarem os sockets adequados: