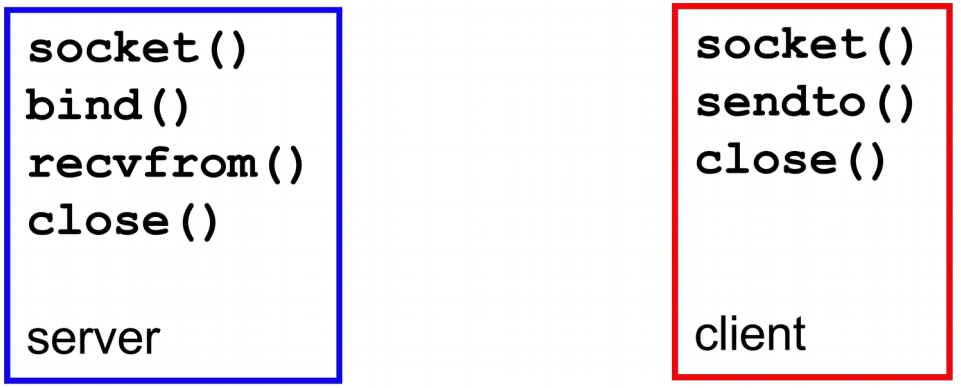
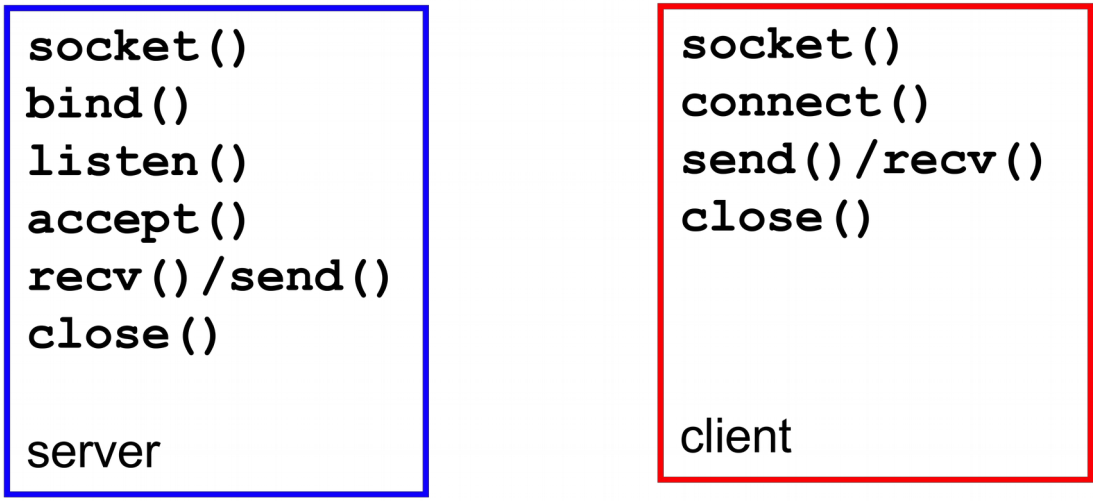
API Usage

- UDP



- TCP



C Code Examples

- socket: common Unix interface to the network
  - end point of network communication in source code
  - Berkeley Software Distribution socket API
    - 4.2 BSD UNIX
    - most OSes provide a BSD socket interface
  - Sockets are ints
    - file descriptor
  - Linux header files

```
#include <sys/types.h>  // data types
#include <arpa/inet.h>  // htonX, etc
#include <netinet/ip.h> // Internet interface
#include <sys/socket.h> // also included by netinet/ip.h
```

Other functions

- **inet_pton()**, inet_ntop()
  - convert IPv4/IPv6 address from/to text/binary
  - p: printable text
  - n: network (binary)

- getnameinfo()
  - get name info from address
- **getaddrinfo()**
  - get address info from name
  - query DNS

TCP Sockets

```
int result;

int theSocket = socket(AF_INET, SOCK_STREAM, 0);

// AF_INET
// SOCK_STREAM
// 0

if ( -1 == theSocket )
{
  perror("socket failed:");
  return -1;
}
```

TCP Sockets

```
char const * const szHost = "64.59.233.197";

struct sockaddr_in sAddr; // IPv4

sAddr.sin_family = AF_INET;
sAddr.sin_port = htons( 25 );

result = inet_pton(sAddr.sin_family, szHost, &sAddr.sin_addr);


result = connect(theSocket,  (struct sockaddr*) &sAddr,
sizeof(struct sockaddr_in));
```

TCP Sockets

```
int count = send(theSocket, szSendBuffer, strlen(szSendBuffer), 0);

// does this code send the \0 ?

// should we?



count = recv(theSocket, szRecvBuffer, BUF_SIZE, 0);



close(theSocket);
```

UDP

- UDP over IP
  - domain (protocol family): AF_INET
  - type: SOCK_DGRAM
  - protocol: 0 (IP)

UDP

- ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen)

- flags tells the function how to behave

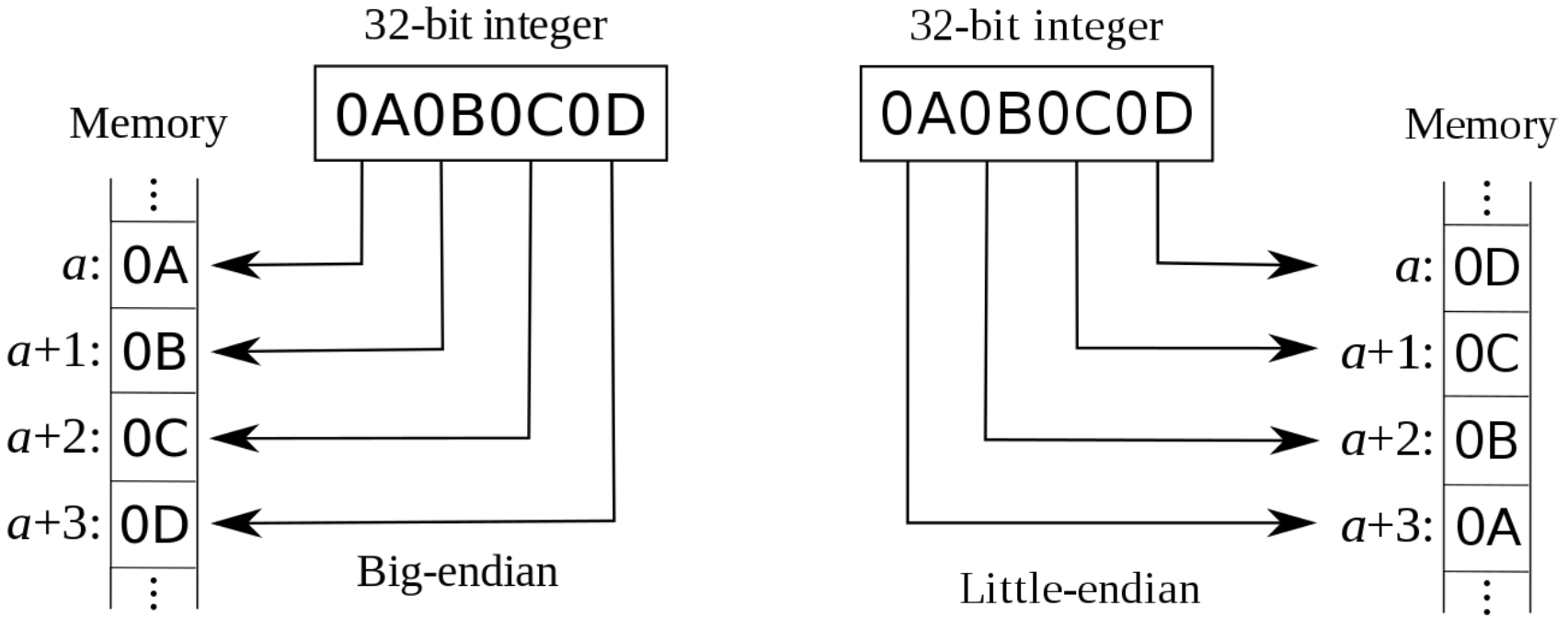
- sendto() is the complement of recvfrom
  - man sendto
- close(int sockfd)
  - just like closing a file

Network Byte Order

- What order are bytes stored in?
  - big endian
    - network byte order (on the wire!)
    - Sun SPARC
    - Motorola 68K
    - IBM z/Architecture
  - little endian
    - intel/AMD

Endianness

C-type	*name*	*initial value of variable*	**Big-Endian** *memory at offset*				**Little-Endian** *memory at offset*			
			+0	+1	+2	+3	+0	+1	+2	+3
`int32_t`	longVar	= 0x0a0b0c0d;	$0A_h$	$0B_h$	$0C_h$	$0D_h$	$0D_h$	$0C_h$	$0B_h$	$0A_h$
`int16_t`	shortVar	= 0x0c0d;	$0C_h$	$0D_h$			$0D_h$	$0C_h$		



https://en.wikipedia.org/wiki/Endianness

Conversion

- Even intel to intel connections must be big endian on the wire

C Datatypes

- #include <inttypes.h>
- #include <stdint.h>
- uint8_t
- uint16_t
- uint32_t
- uint64_t
- int32_t

http://en.cppreference.com/w/c/types/integer
https://en.wikipedia.org/wiki/C_data_types#Fixed-width_integer_types

Practice

- p 169
  - R3, R4, R5, R8, R10, R11, R12, R13, R15, R16, R23, R24, R26, R27