Sockets

Advanced Embedded Software Development

with **Dan Walkes**

University of Colorado **Boulder**

Learning objectives:

Understand Linux Sockets

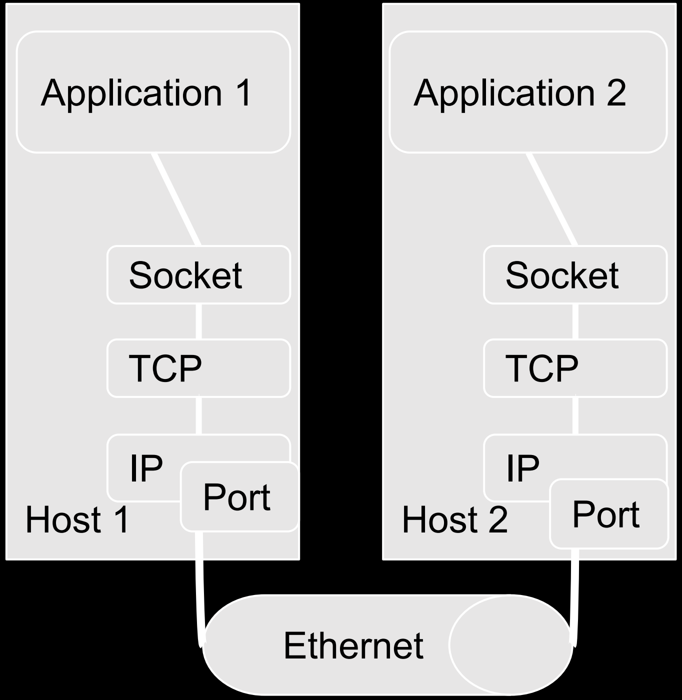
Understand How to Use Sockets in your programs

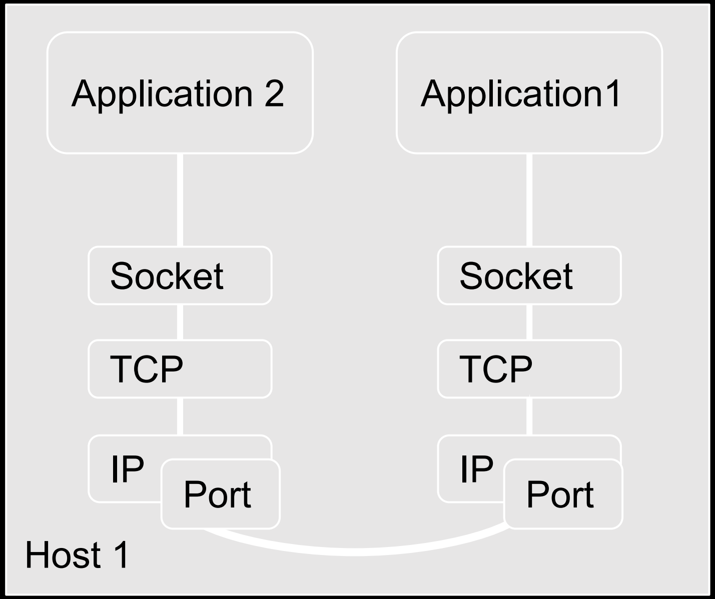
Sockets

- One of several forms of Interprocess Communication (IPC)
  - Can communicate across different systems over TCP/IP
- Also known as BSD or Berkeley Sockets
- Supported by all major operating systems

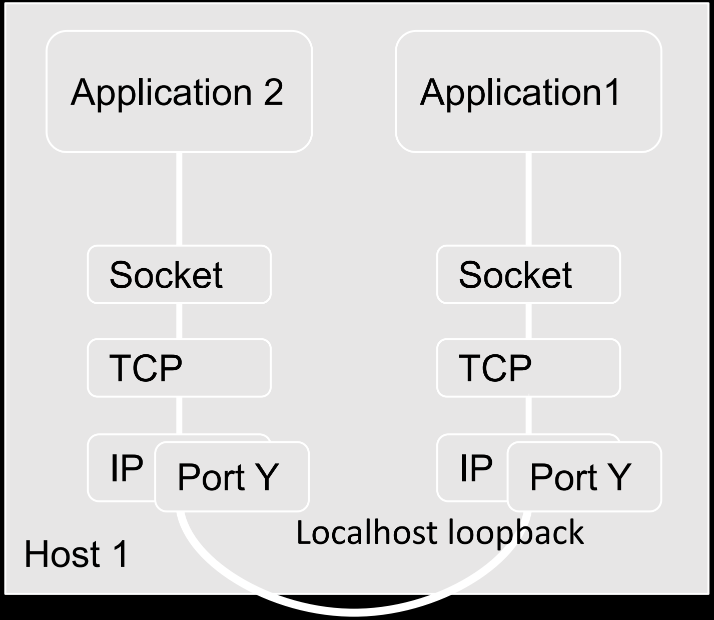
https://en.wikipedia.org/wiki/Berkeley_sockets

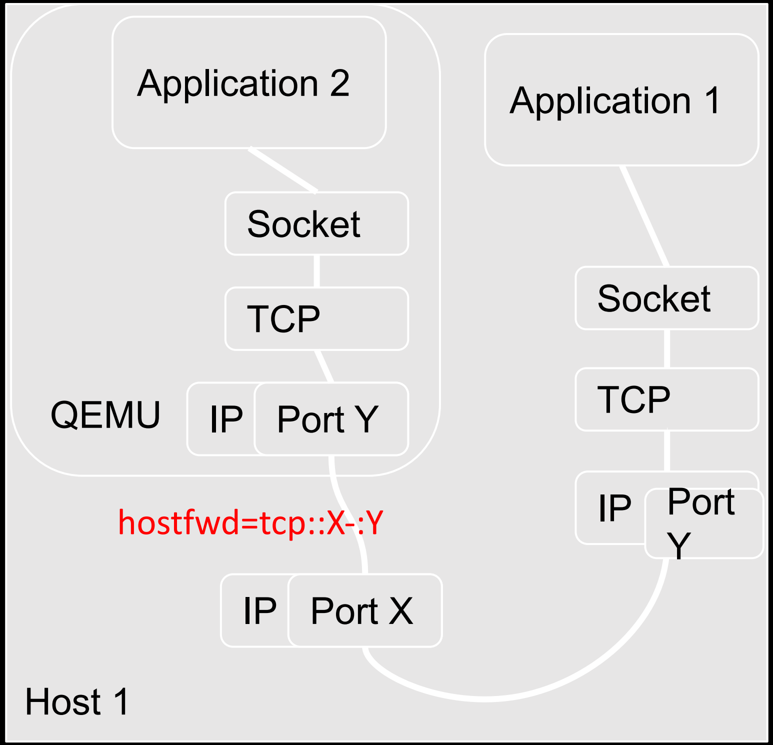
Connection Oriented Sockets





Port Forwarding





- Allows us to route port to VM or emulator

TCP

- Transmission Control Protocol
- Connection oriented protocol
  - Connection is established and maintained while programs are exchanging messages
- Accepts packets
- Manages flow control
- Handles retransmission of dropped packets
- Handles acknowledgement of packets

https://searchnetworking.techtarget.com/definition/TCP

IP

- Addresses Packets
- Supports routing between sender and receiver
- IPv4 was first implementation - X.X.X.X - 4 byte (32 bit) format - ~4.3 billion addresses
- IPv6 supports 128 bits of address space - 340 billion billion billion billion addresses)
  - 4000 addresses for each person on earth
- Also uses a “port” for local addressing

https://rednectar.net/2012/05/24/just-how-many-ipv6-addresses-are-there-really/

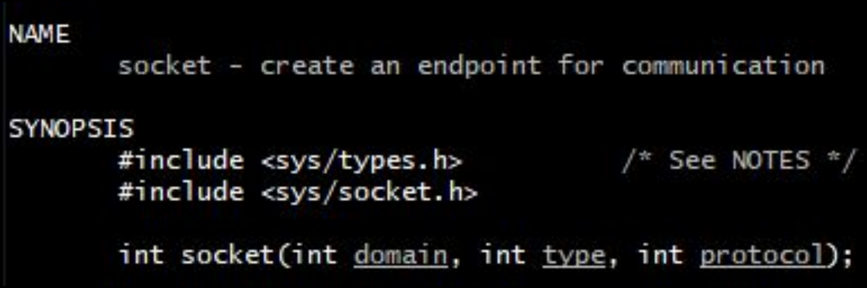
Types of Sockets

- SOCK_STREAM - Stream Sockets
  - Reliable two way connected (TCP) streams
  - Messages are delivered in order
  - Retried as necessary
- SOCK_DGRAM - Datagram Sockets
  - Connectionless sockets
  - Use UDP (User Datagram Protocol) instead of TCP

https://beej.us/guide/bgnet/html/single/bgnet.html

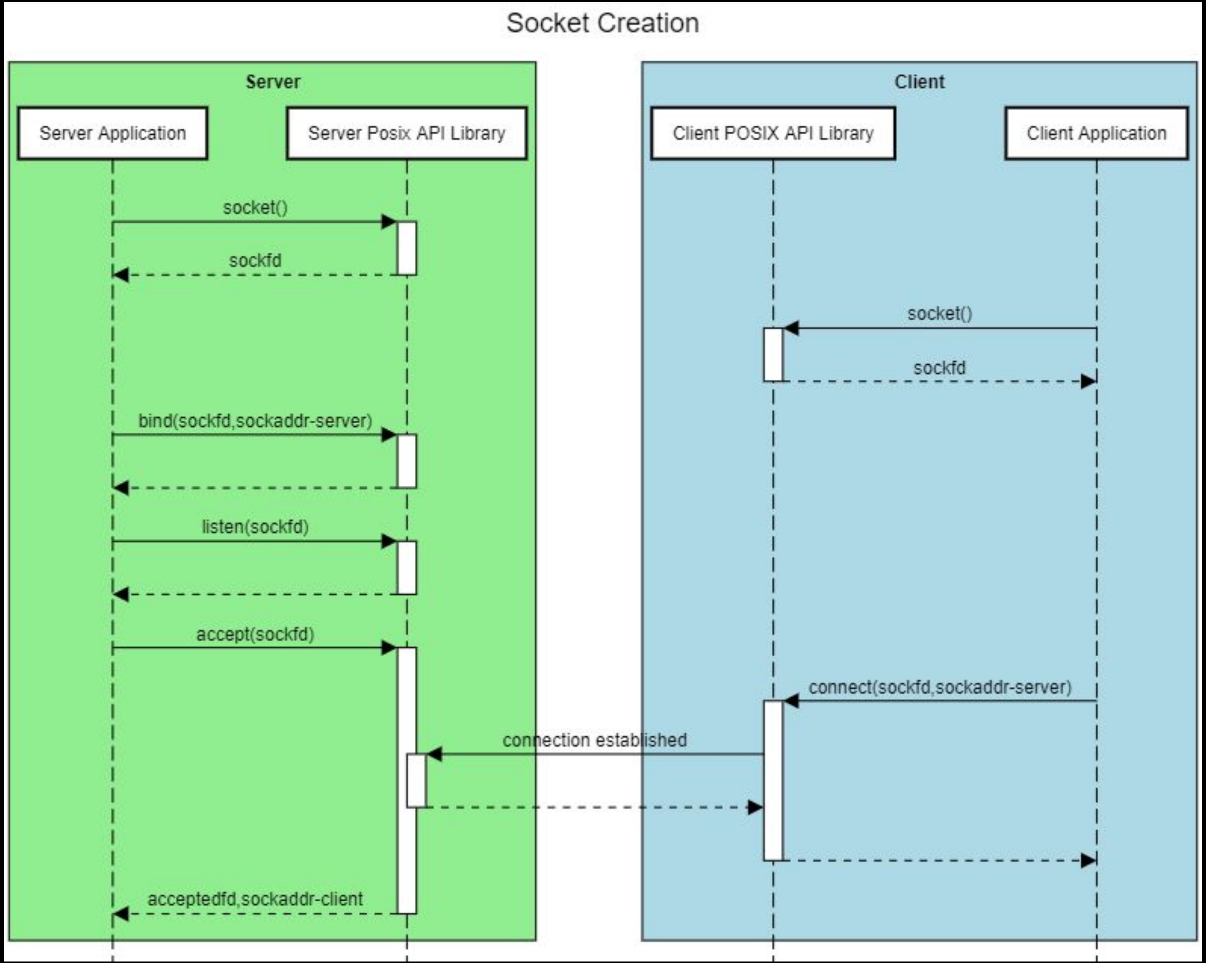
Accessing Sockets

- In Linux everything is a file
- How do we interact with sockets?
  - Using a socket file descriptor
- How do we obtain a socket file descriptor?
  - Use the socket() POSIX function



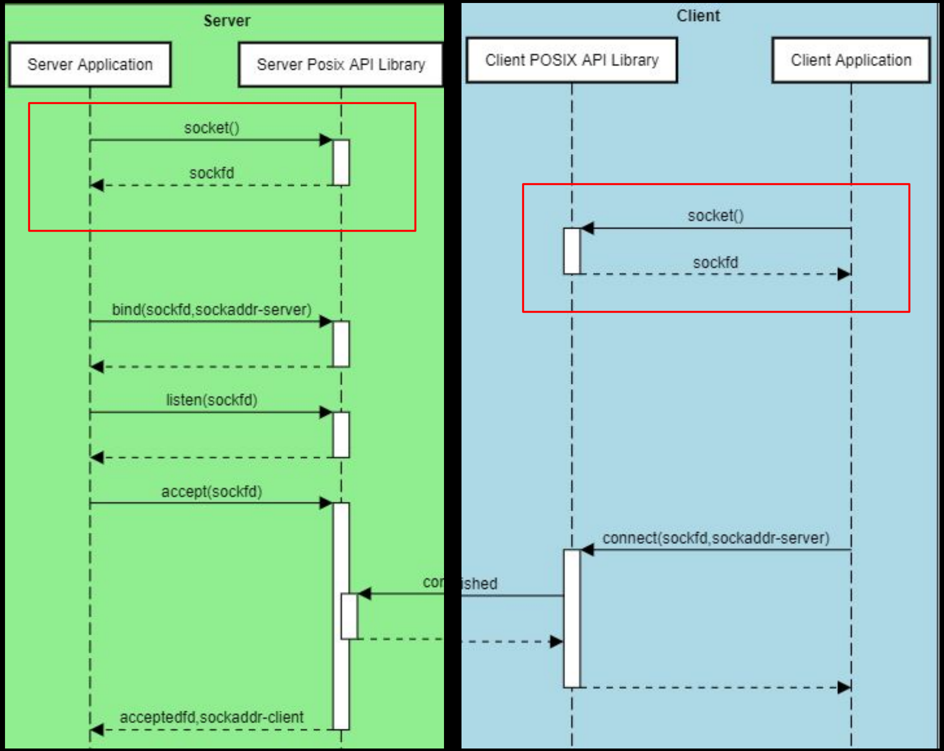
```
NAME
       socket - create an endpoint for communication

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

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

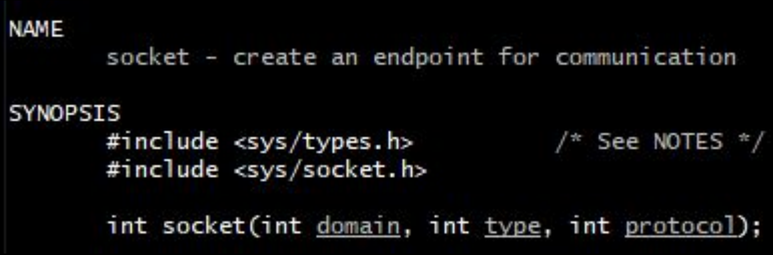
https://beej.us/guide/bgnet/html/single/bgnet.html



https://stackoverflow.com/questions/27014955/socket-connect-vs-bind

Socket



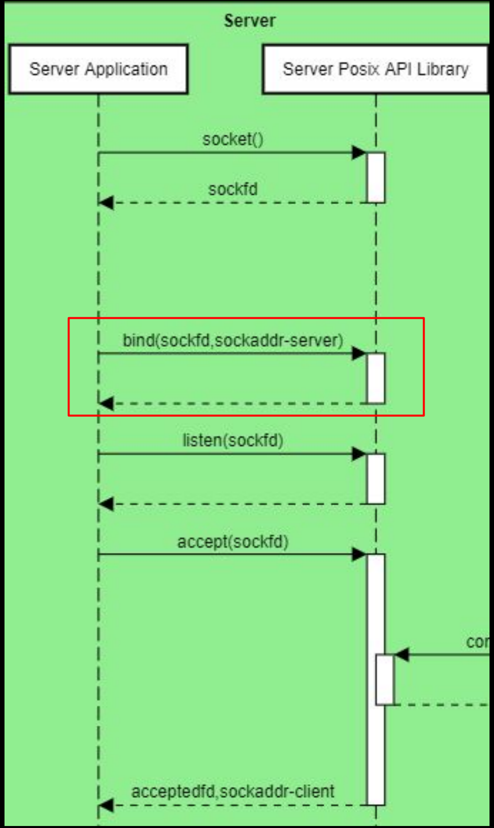


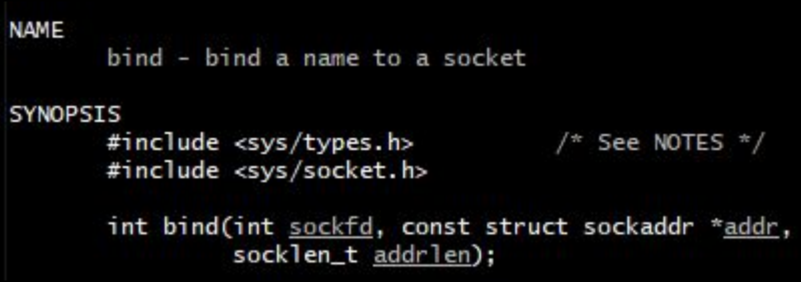
```
NAME
       socket - create an endpoint for communication

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

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

- domain - PF_INET or PF_INET6
- type SOCK_STREAM or SOCK_DGRAM
- protocol - 0 to choose proper for given type

https://beej.us/guide/bgnet/html/#getaddrinfoprepare-to-launch

bind



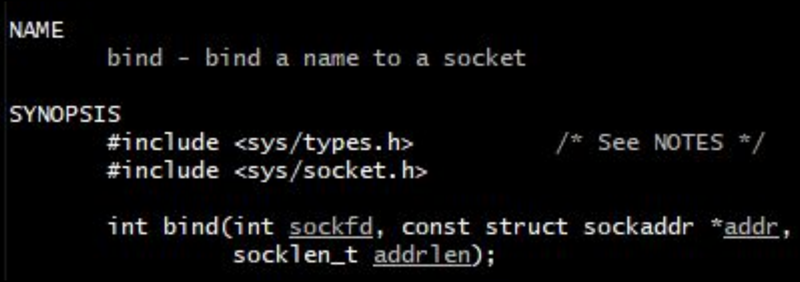


```
NAME
       bind - bind a name to a socket

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

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

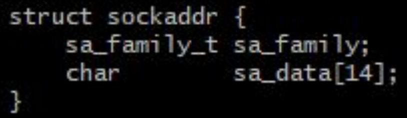
- Assigns an address to the socket
- sockfd is the fd from socket()
- sockaddr addr/addrlen describes the address to bind (optionally select a specific network adapter).

https://stackoverflow.com/questions/27014955/socket-connect-vs-bind
http://man7.org/linux/man-pages/man2/bind.2.html

bind - sockaddr



```
NAME
       bind - bind a name to a socket

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

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



```
struct sockaddr {
    sa_family_t sa_family;
    char        sa_data[14];
}
```

- sockaddr - maps to a server socket location
  - sa_family - AF_INET or AF_INET6
  - sa_data - destination address and port
- socketlen_t - unsigned integer type
  sizeof(struct sockaddr)

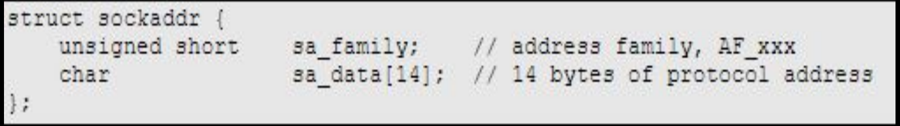
https://pubs.opengroup.org/onlinepubs/007908799/xns/syssocket.h.html
https://beej.us/guide/bgnet/html/#getaddrinfoprepare-to-launch

Setting up sockaddr

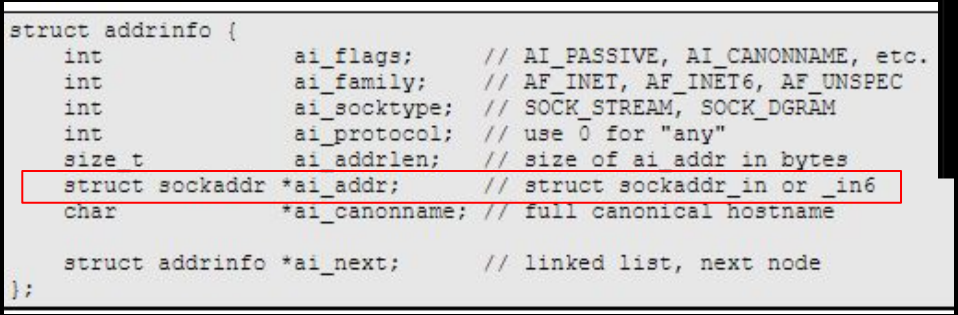
- sockaddr structure isn’t built directly, setup from other structures
- Two options discussed:
    - Setting up sockaddr_in and casting to sockaddr
    - Setting up with getaddrinfo()
    - getaddrinfo is newer and more flexible
    - Either is acceptable for the assignment

http://man7.org/linux/man-pages/man3/getaddrinfo.3.html
https://beej.us/guide/bgnet/html/

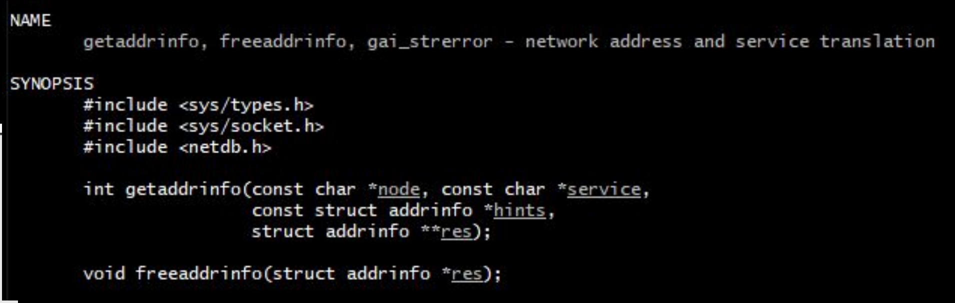
Setting up sockaddr



```
struct sockaddr {
    unsigned short    sa_family;    // address family, AF_xxx
    char              sa_data[14];  // 14 bytes of protocol address
};
```



```
struct addrinfo {
    int              ai_flags;      // AI_PASSIVE, AI_CANONNAME, etc.
    int              ai_family;     // AF_INET, AF_INET6, AF_UNSPEC
    int              ai_socktype;   // SOCK_STREAM, SOCK_DGRAM
    int              ai_protocol;   // use 0 for "any"
    size_t           ai_addrlen;    // size of ai_addr in bytes
    struct sockaddr *ai_addr;       // struct sockaddr_in or _in6
    char            *ai_canonname;  // full canonical hostname

    struct addrinfo *ai_next;       // linked list, next node
};
```

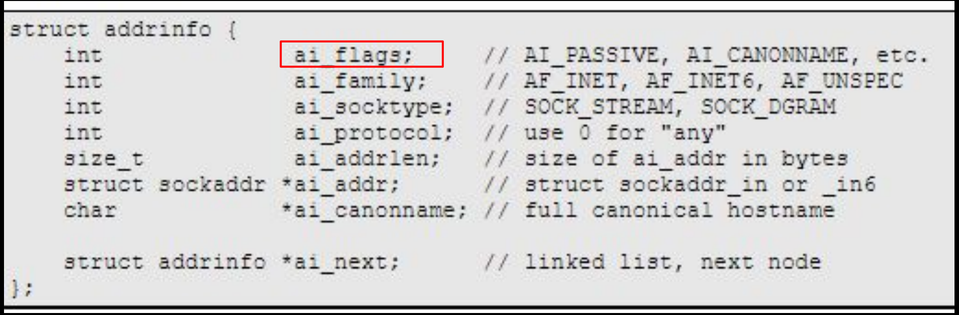


```
NAME
       getaddrinfo, freeaddrinfo, gai_strerror - network address and service translation

SYNOPSIS
       #include <sys/types.h>
       #include <sys/socket.h>
       #include <netdb.h>

       int getaddrinfo(const char *node, const char *service,
                       const struct addrinfo *hints,
                       struct addrinfo **res);

       void freeaddrinfo(struct addrinfo *res);
```

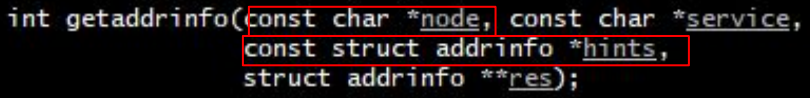
- getaddrinfo provides addrinfo (containing sockaddr) through addrinfo argume se

http://man7.org/linux/man-pages/man3/getaddrinfo.3.html
https://beej.us/guide/bgnet/html/

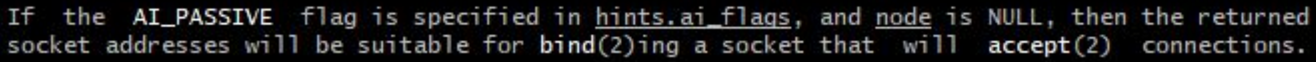
Setting addrinfo hints and node



```
struct addrinfo {
    int              ai_flags;     // AI_PASSIVE, AI_CANONNAME, etc.
    int              ai_family;    // AF_INET, AF_INET6, AF_UNSPEC
    int              ai_socktype;  // SOCK_STREAM, SOCK_DGRAM
    int              ai_protocol;  // use 0 for "any"
    size_t           ai_addrlen;   // size of ai_addr in bytes
    struct sockaddr *ai_addr;      // struct sockaddr_in or _in6
    char            *ai_canonname; // full canonical hostname

    struct addrinfo *ai_next;      // linked list, next node
};
```



```
int getaddrinfo(const char *node, const char *service,
                const struct addrinfo *hints,
                struct addrinfo **res);
```

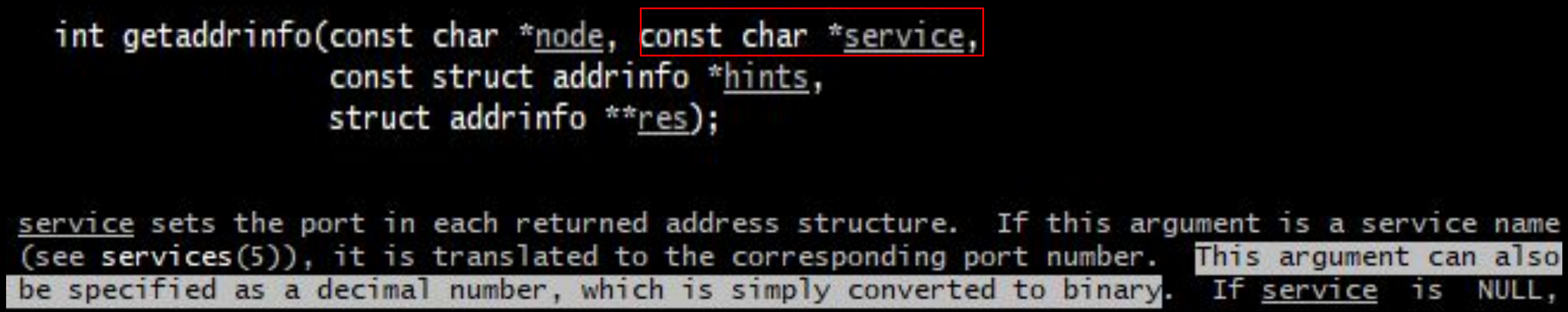


If the AI_PASSIVE flag is specified in hints.ai_flags, and node is NULL, then the returned socket addresses will be suitable for bind(2)ing a socket that will accept(2) connections.

- ai_flags in hints and node parameter sets up the socket address for bind()/accept()
  - hints.ai_flags = AI_PASSIVE, hints.ai_family = AF_INET, hints.ai_socktype = SOCK_STREAM
  - node = NULL

http://man7.org/linux/man-pages/man3/getaddrinfo.3.html
https://beej.us/guide/bgnet/html/

Setting getaddrinfo service



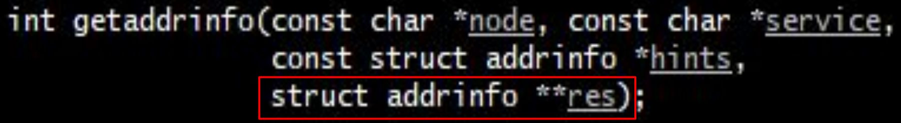
```
int getaddrinfo(const char *node, const char *service,
                const struct addrinfo *hints,
                struct addrinfo **res);
```

```
service sets the port in each returned address structure.  If this argument is a service name
(see services(5)), it is translated to the corresponding port number.  This argument can also
be specified as a decimal number, which is simply converted to binary.  If service  is  NULL,
```

- service parameter sets port for the connection
  - “1234” would setup for port 1234

http://man7.org/linux/man-pages/man3/getaddrinfo.3.html
https://beej.us/guide/bgnet/html/

Getting sockaddr

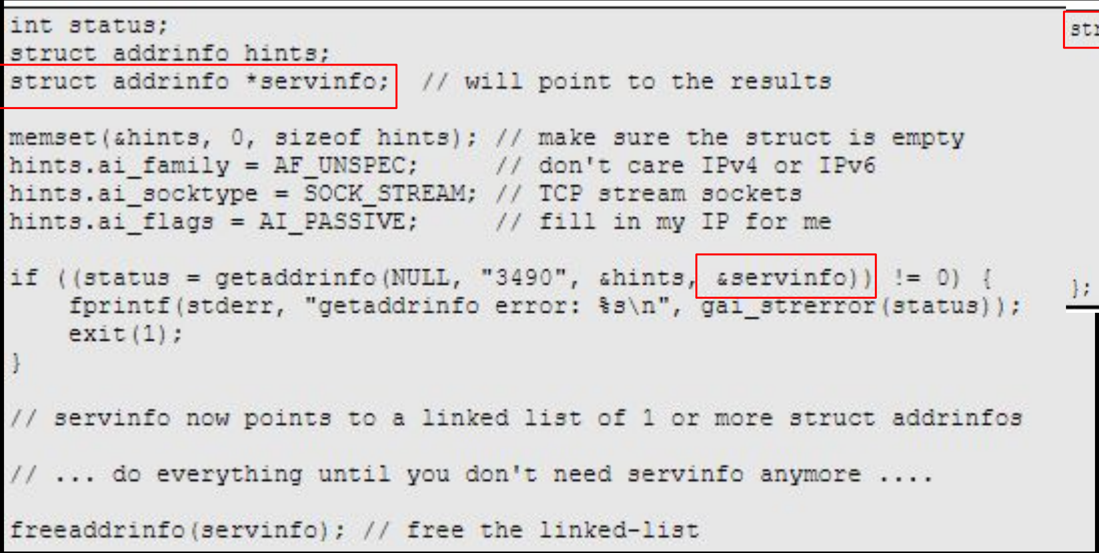


```
int getaddrinfo(const char *node, const char *service,
                const struct addrinfo *hints,
                struct addrinfo **res);
```

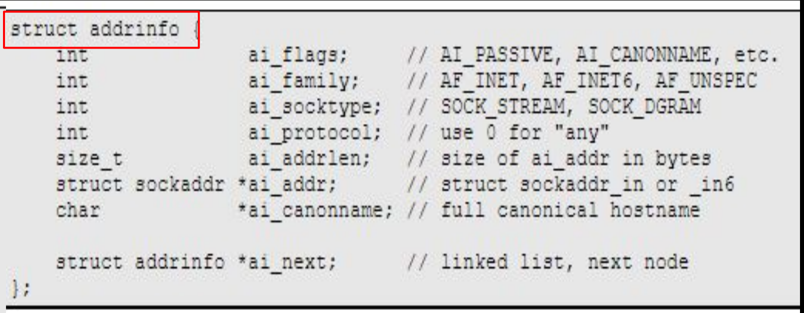
- Setup pointer res to store addrinfo returned from getaddrinfo
  - Pass address of pointer as res arg (pointer to pointer)

http://man7.org/linux/man-pages/man3/getaddrinfo.3.html
https://beej.us/guide/bgnet/html/

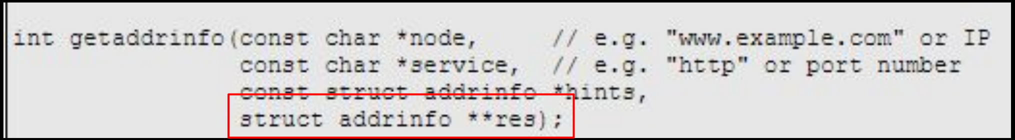
Getting sockaddr - pointer to pointer



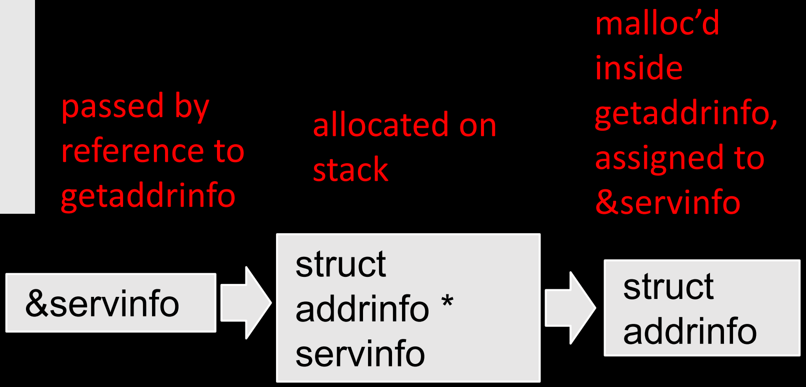
```
int status;
struct addrinfo hints;
struct addrinfo *servinfo;  // will point to the results

memset(&hints, 0, sizeof hints); // make sure the struct is empty
hints.ai_family = AF_UNSPEC;     // don't care IPv4 or IPv6
hints.ai_socktype = SOCK_STREAM; // TCP stream sockets
hints.ai_flags = AI_PASSIVE;     // fill in my IP for me

if ((status = getaddrinfo(NULL, "3490", &hints, &servinfo)) != 0) {
    fprintf(stderr, "getaddrinfo error: %s\n", gai_strerror(status));
    exit(1);
}

// servinfo now points to a linked list of 1 or more struct addrinfos

// ... do everything until you don't need servinfo anymore ....

freeaddrinfo(servinfo); // free the linked-list
```



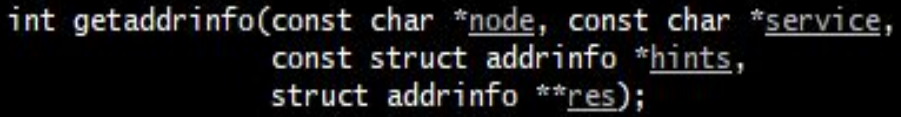
```
struct addrinfo {
    int              ai_flags;     // AI_PASSIVE, AI_CANONNAME, etc.
    int              ai_family;    // AF_INET, AF_INET6, AF_UNSPEC
    int              ai_socktype;  // SOCK_STREAM, SOCK_DGRAM
    int              ai_protocol;  // use 0 for "any"
    size_t           ai_addrlen;   // size of ai_addr in bytes
    struct sockaddr *ai_addr;      // struct sockaddr_in or _in6
    char            *ai_canonname; // full canonical hostname

    struct addrinfo *ai_next;      // linked list, next node
};
```



```
int getaddrinfo(const char *node,     // e.g. "www.example.com" or IP
                const char *service,  // e.g. "http" or port number
                const struct addrinfo *hints,
                struct addrinfo **res);
```


passed by reference to getaddrinfo
allocated on stack
malloc'd inside getaddrinfo, assigned to &servinfo
&servinfo
struct addrinfo * servinfo
struct addrinfo

Getting sockaddr

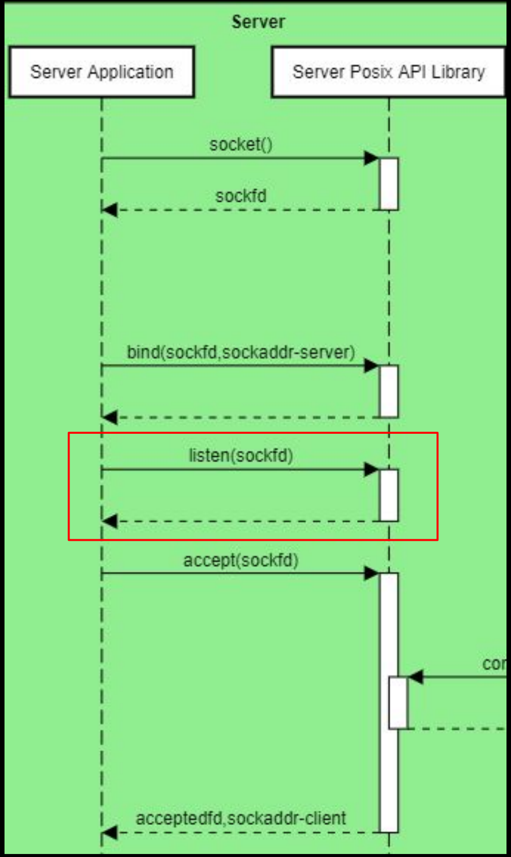


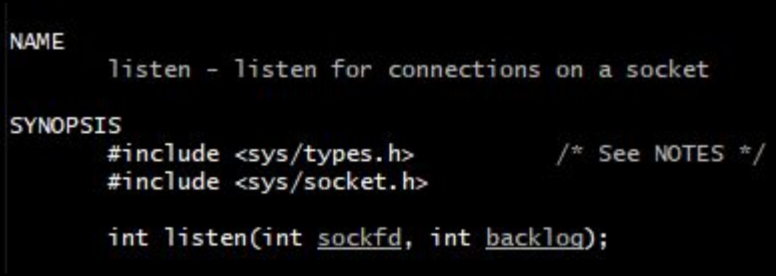
```
int getaddrinfo(const char *node, const char *service,
                const struct addrinfo *hints,
                struct addrinfo **res);
```

- Call getaddrinfo with hints, port string as service argument, and pointer to pointer in res
- Use res->ai_addr as sockaddr for bind()
- freeaddrinfo(res) when no longer needed
    - What if you forget to free?
        - Memory Leak

http://man7.org/linux/man-pages/man3/getaddrinfo.3.html
https://beej.us/guide/bgnet/html/

listen

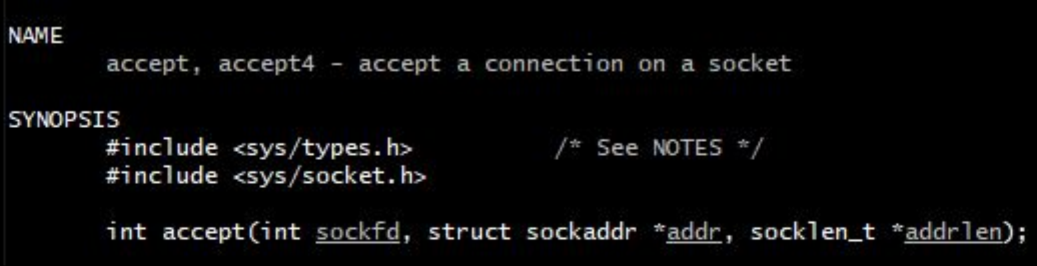



```
NAME
       listen - listen for connections on a socket

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

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

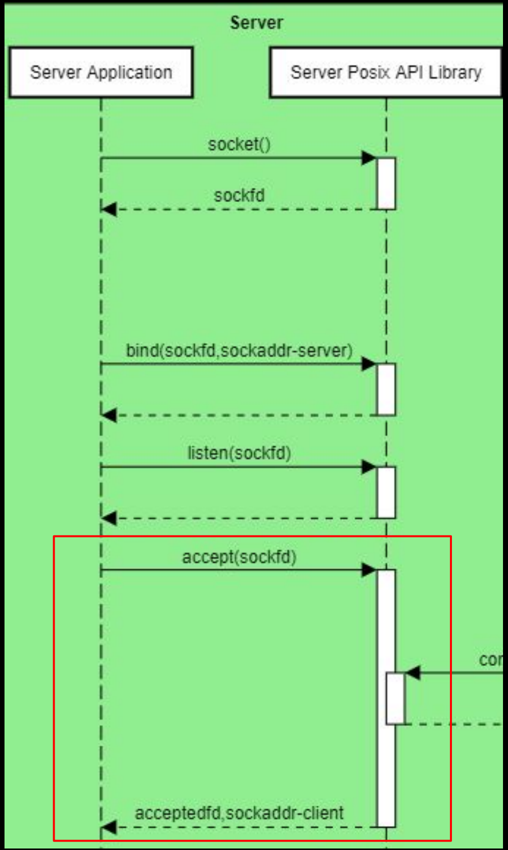
- Passed sockfd from socket()
- backlog specifies number of pending connections allowed before refusing

https://stackoverflow.com/questions/27014955/socket-connect-vs-bind
https://pubs.opengroup.org/onlinepubs/009695399/functions/listen.html

accept



```
NAME
       accept, accept4 - accept a connection on a socket

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

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

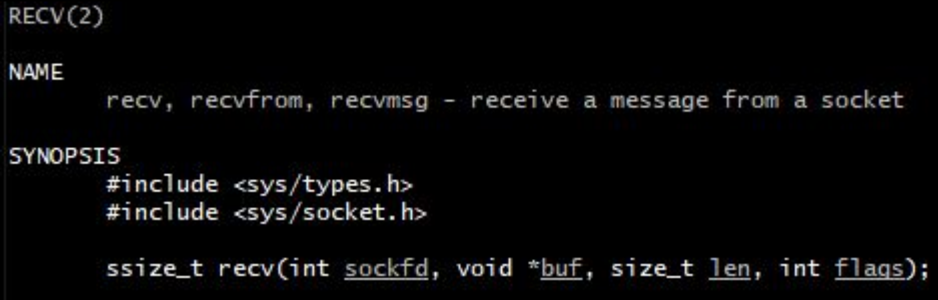


- sockfd - socket file descriptor from socket()
- addr - location to store the connecting address
- addrlen in/out - length of addr and location to store the length of result
- returns: fd for accepted connection

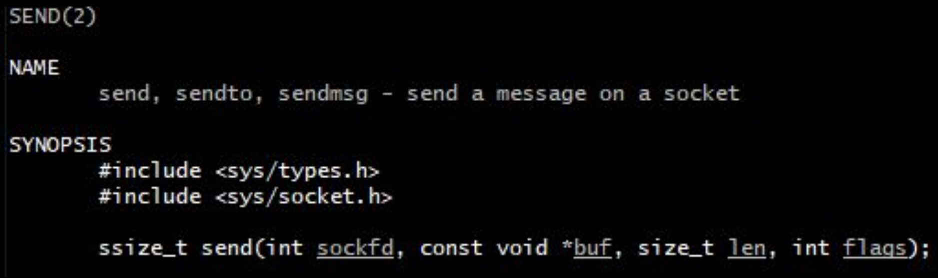
https://stackoverflow.com/questions/27014955/socket-connect-vs-bind
http://man7.org/linux/man-pages/man2/accept.2.html

recv/send

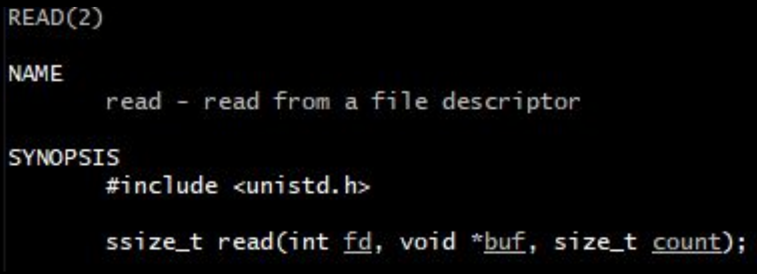
- Similar to read/write file descriptor based commands we’ve discussed in early lectures



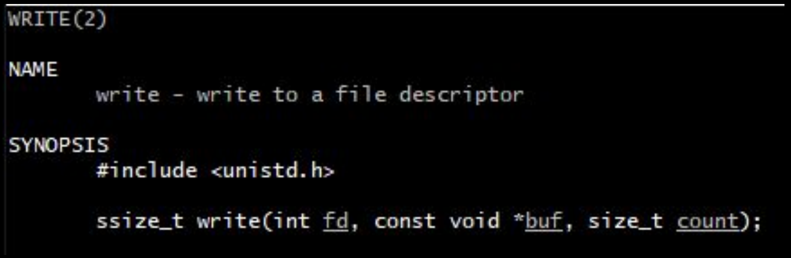
```
RECV(2)

NAME
       recv, recvfrom, recvmsg - receive a message from a socket

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

       ssize_t recv(int sockfd, void *buf, size_t len, int flags);
```



```
SEND(2)

NAME
       send, sendto, sendmsg - send a message on a socket

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

       ssize_t send(int sockfd, const void *buf, size_t len, int flags);
```



```
READ(2)

NAME
       read - read from a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t read(int fd, void *buf, size_t count);
```

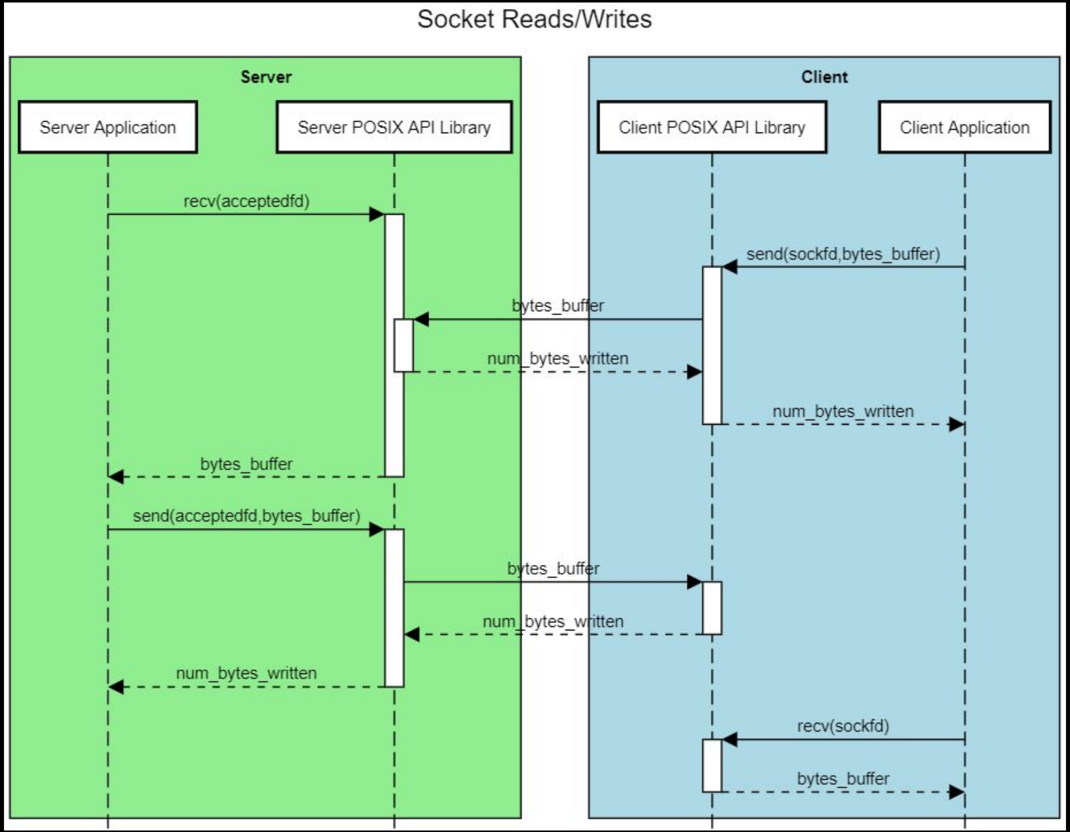


```
WRITE(2)

NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);
```

https://stackoverflow.com/questions/27014955/socket-connect-vs-bind
http://man7.org/linux/man-pages/man2/read.2.html
http://man7.org/linux/man-pages/man2/write.2.html

recv/send

- Use acceptedfd (from accept() return value) for server application
- Use blocking or non-blocking reads based on flags argument to recv/send



https://stackoverflow.com/questions/27014955/socket-connect-vs-bind
http://man7.org/linux/man-pages/man2/read.2.html
http://man7.org/linux/man-pages/man2/write.2.html