523454

Computer Network Programming

LAB2: TCP Connections

Dr. Parin Sornlertlamvanich,

parin.s@sut.ac.th

Socket Address Structures

IPv4 Socket Address Structure

```
struct in_addr {
  in_addr_t      s_addr;        /* 32-bit IPv4 address */
                                /* network byte ordered */
};


struct sockaddr_in {
  uint8_t        sin_len;       /* length of structure (16) */
  sa_family_t    sin_family;    /* AF_INET */
  in_port_t      sin_port;      /* 16-bit TCP or UDP port number */
                                /* network byte ordered */
  struct in_addr sin_addr;      /* 32-bit IPv4 address */
                                /* network byte ordered */
  char           sin_zero[8];   /* unused */
};
```

Socket Address Structures

- Generic Socket Address Structure

```
struct sockaddr {
  uint8_t       sa_len;
  sa_family_t   sa_family;      /* address family: AF_xxx value */
  char          sa_data[14];    /* protocol-specific address */
};
```

```
int bind(int, struct sockaddr *, socklen_t);
struct sockaddr_in  serv;             /* IPv4 socket address structure */

/* fill in serv{} */

bind(sockfd, (struct sockaddr *) &serv, sizeof(serv));
```

Socket Address Structures

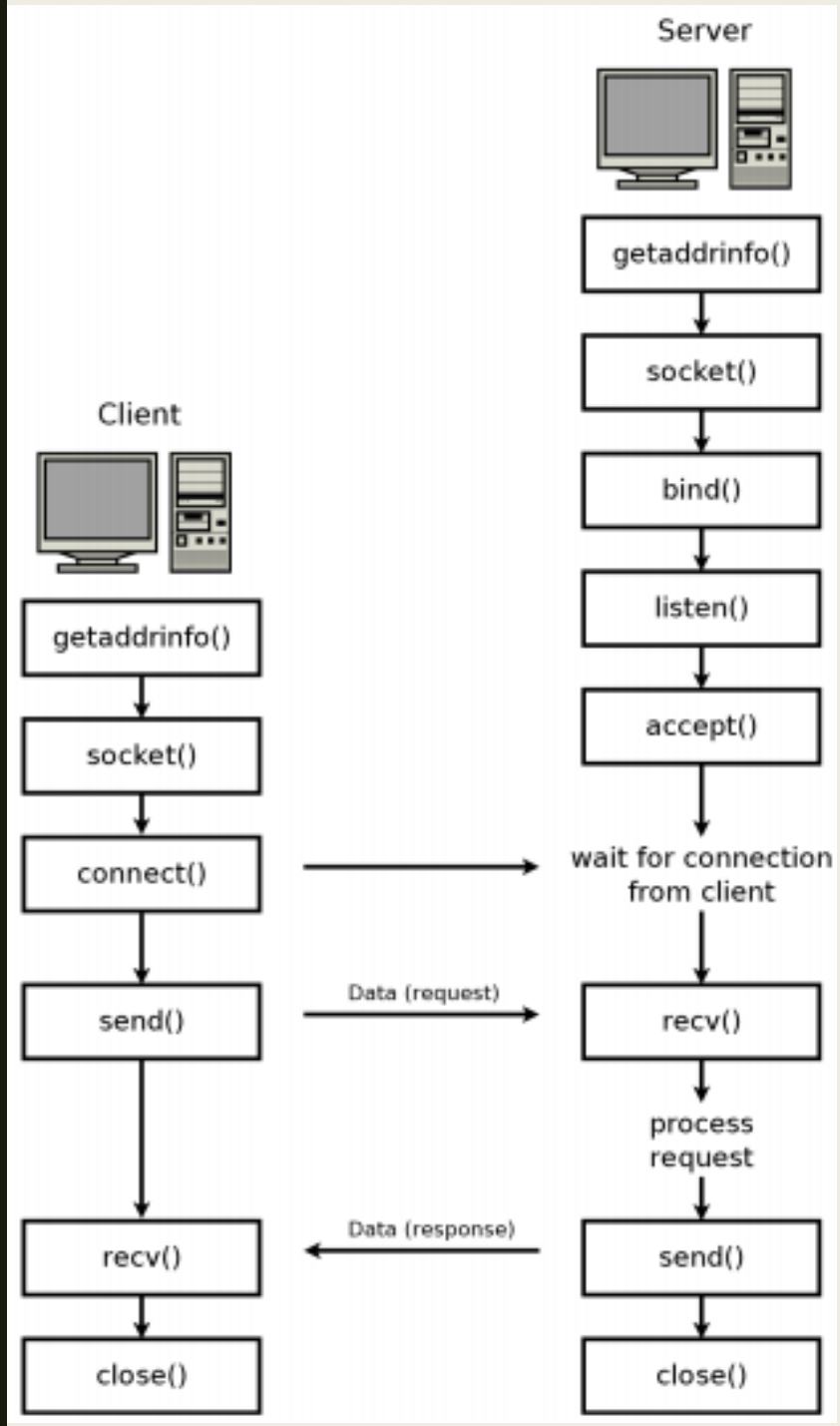
- IPv6 Socket Address Structure

```
struct in6_addr {
  uint8_t  s6_addr[16];                 /* 128-bit IPv6 address */
                                        /* network byte ordered */
};


struct sockaddr_in6 {
  uint8_t           sin6_len;           /* length of this struct (28) */
  sa_family_t       sin6_family;        /* AF_INET6 */
  in_port_t         sin6_port;          /* transport layer port# */
                                        /* network byte ordered */
  uint32_t          sin6_flowinfo;      /* flow information, undefined */
  struct in6_addr   sin6_addr;          /* IPv6 address */
                                         /* network byte ordered */
  uint32_t          sin6_scope_id;      /* set of interfaces for a scope */
};
```

Socket Address Structures

- New Generic Socket Address Structure

```
struct sockaddr_storage {
  uint8_t       ss_len;     /* length of this struct (implementation
                               dependent) */
  sa_family_t   ss_family;  /* address family: AF_xxx value */


 /* implementation-dependent elements to provide:
   * a) alignment sufficient to fulfill the alignment requirements of
   *    all socket address types that the system supports.
   * b) enough storage to hold any type of socket address that the
   *    system supports.
   */
};
```



TCP: client & server

1. The server then creates the socket with a call to socket()
2. The socket must be bound to the listening IP address and port - call to bind()
3. then calls listen(), which puts the socket in a state where it listens for new connections
4. then call accept(), which will wait until a client establishes a connection to the server
5. When the new connection has been established, accept() returns a *new* socket
6. This *new* socket can be used to exchange data with the client using send() and recv()

**Meanwhile, the first socket remains listening for new connections, and repeated calls to accept()

Socket Core Functions (TCP)

- int socket (int family, int type, int protocol)
- int bind(int sockfd, struct sockaddr *my_addr,int addrlen)
- int listen(int sockfd, int backlog)
- int connect (int sockfd, struct sockaddr *serv_addr, int addrlen)
- int accept (int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen)
- int send(int sockfd, const void *msg, int len, int flags)
- int recv (int sockfd, void *buf, int len, unsigned int flags)
- int close (int sockfd)

Lab2_client_checkpoint.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <errno.h>

#define GETSOCKETERRNO() (errno)
#define SERVER_ADDRESS "127.0.0.1"
#define SERVER_ADDRESS6 "::1"
#define MAXLINE 4096

#include <stdio.h>
#include <string.h>
#include <time.h>
```

Lab2_serv_dual_checkpoint.c

```c
int main() {

struct addrinfo hints;
struct addrinfo *servinfo;
struct sockaddr_storage client_address;
int socket_listen;
char buf[MAXLINE];
int n =0;
int socket_client;

    printf("Configuring local address ... \n\n");
    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_INET6;          //Dual Stack
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_flags = AI_PASSIVE;

    getaddrinfo(0, "7777", &hints, &servinfo);
```


```c
int main() {

struct addrinfo hints;
struct addrinfo *servinfo;
struct sockaddr_storage client_address;
int socket_listen;
char buf[MAXLINE];
int n =0;
int socket_client;

    printf("Configuring local address...\n\n");
    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_INET6;          //Dual Stack
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_flags = AI_PASSIVE;

    getaddrinfo(0, "7777", &hints, &servinfo);
```

Lab2_serv_dual_checkpoint.c

- Creating Socket (before setting sockopt)
- Setsockopt for Dual Stack
  - After the call to socket() and before the call to bind()
  - IPV6_V6ONLY is enabled by default, so
  - we clear it by setting it to 0

```
int option = 0;
if (setsockopt(socket_listen, IPPROTO_IPV6, IPV6_V6ONLY, (void*)&option, sizeof(option))) {
    fprintf(stderr, "setsockopt() failed. (%d)\n", GETSOCKETERRNO());
    return 1; //Dual Stack
}
```

```
int option = 0;
   if (setsockopt(socket_listen, IPPROTO_IPV6, IPV6_V6ONLY, (void*)&option,
sizeof(option))) {
      fprintf(stderr, "setsockopt() failed. (%d)\n", GETSOCKETERRNO());
      return 1; //Dual Stack
   }
```

Lab2_serv_dual_checkpoint.c

- Bind()
- Listen
- Accept
- Recv()

```
for (;;) {
    socklen_t client_len = sizeof(client_address);
    socket_client = accept (socket_listen,
            (struct sockaddr*) &client_address, &client_len);
    printf("%s\n","Received request ... ");

    while ( (n = recv(socket_client , buf, MAXLINE,0)) > 0)  {
```

Lab2_client_checkpoint.c

```c
int main() {
struct addrinfo hints;
struct addrinfo *server_addr;
struct sockaddr_storage server_from;
int socket_peer;
char sendline[MAXLINE], recvline[MAXLINE];
char buffer[1024];

    printf("Configuring remote address...\n");
    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_INET6;
    hints.ai_socktype = SOCK_STREAM;

    if (getaddrinfo(SERVER_ADDRESS6, "7777", &hints, &server_addr)) {
        fprintf(stderr, "getaddrinfo() failed. (%d)\n", GETSOCKETERRNO());
        return 1;
    }
```


```c
int main() {
struct addrinfo hints;
struct addrinfo *server_addr;
struct sockaddr_storage server_from;
int socket_peer;
char sendline[MAXLINE], recvline[MAXLINE];
char buffer[1024];

    printf("Configuring remote address...\n");
    memset(&hints, 0, sizeof(hints));
    hints.ai_family = AF_INET6;
    hints.ai_socktype = SOCK_STREAM;

    if (getaddrinfo(SERVER_ADDRESS6, "7777", &hints, &server_addr)) {
        fprintf(stderr, "getaddrinfo() failed. (%d)\n", GETSOCKETERRNO());
        return 1;
    }
```

Lab2_client_checkpoint.c

```
printf("Server address is: ");
char address_buffer[100];
char port_buffer[100];
getnameinfo(server_addr→ai_addr, server_addr→ai_addrlen,
        address_buffer, sizeof(address_buffer),
        port_buffer, sizeof(port_buffer),
        NI_NUMERICHOST  | NI_NUMERICSERV);
printf("%s \nPort is: %s\n", address_buffer, port_buffer);
```


```
printf("Server address is: ");
 char address_buffer[100];
 char port_buffer[100];
 getnameinfo(server_addr->ai_addr, server_addr->ai_addrlen,
        address_buffer, sizeof(address_buffer),
        port_buffer, sizeof(port_buffer),
        NI_NUMERICHOST  | NI_NUMERICSERV);
 printf("%s \nPort is: %s\n", address_buffer, port_buffer);
```

Lab2_client_checkpoint.c

- Creating socket
- connect(socket_peer, server_addr->ai_addr, server_addr->ai_addrlen)
- send()

```
        while (fgets(sendline, MAXLINE, stdin) != NULL) {
          send(socket_peer, sendline, strlen(sendline), 0);
          printf("String: ");
          fputs(recvline, stdout);
        }

    close(socket_peer);
    printf("Finished.\n");
    return 0;
}
```

```
┌──(kali㉿kali)-[~/lab_netPro/my_lab/lab2]
└─$ ./lab2_client
Configuring remote address ...
Server address is: ::1
Port is: 7777
Creating socket ...
Connected.
CPE
String: SUT
String: THAILAND
String: footbal
String:
```

```
┌──(kali㉿kali)-[~/lab_netPro/my_lab/lab2]
└─$ ./lab2_server
Configuring local address ...

Creating socket ...

Binding socket to local address ...

Listening ...

Waiting for connection ...

Received request ...
String received from the client:CPE

String received from the client:SUT

String received from the client:THAILAND

String received from the client:footbal
```

Dual Stack

```
Configuring remote address ...
Server address is: 127.0.0.1
Port is: 7777
Creating socket ...
Connected.
```

Reference

- Introduction to Sockets Programming in C using TCP/IP
  - Panagiota Fatourou