Network Programming

Lecture 2—Elementary Sockets I

Lei Wang
lei.wang@dlut.edu.cn

Dalian University of Technology

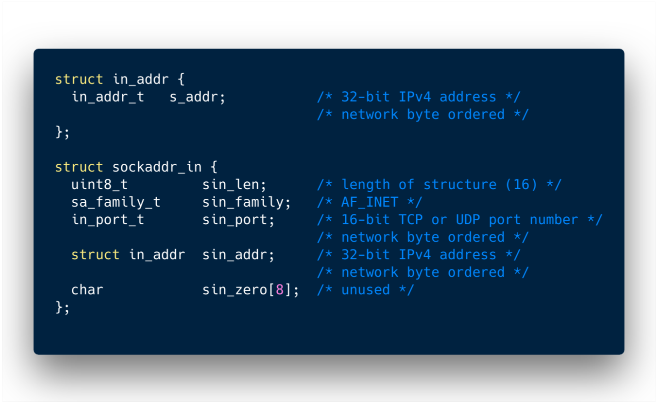
Oct 24, 2018

Part 2. Elementary Sockets I

Socket Address Structures

- Most socket functions require a pointer to a socket address structure as an argument.
- Each supported protocol suite defines its own socket address structure.
- The names of these structures begin with `sockaddr_` and end with a unique suffix for each protocol suite.

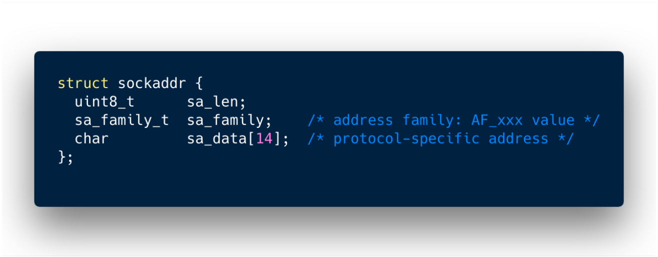
IPv4 Socket Address Structure



```
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 */
};
```

Figure: IPV4 socket address structure

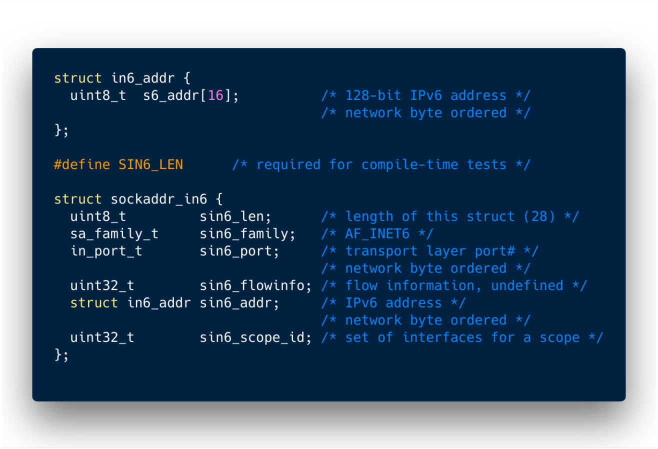
Generic Address Structure



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

Figure: Generic Address Structure

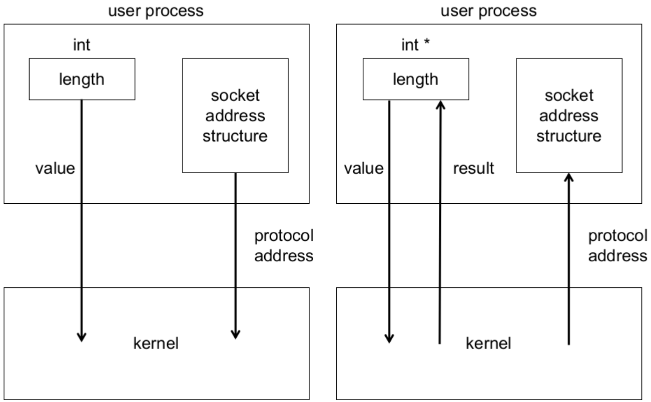
IPv6 Socket Address Structure



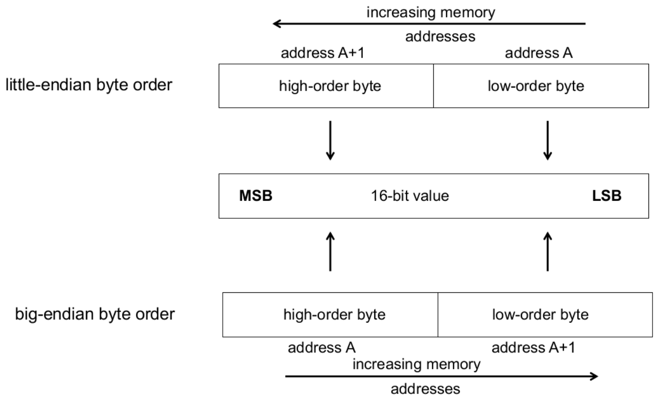
```
struct in6_addr {
  uint8_t  s6_addr[16];          /* 128-bit IPv6 address */
                                 /* network byte ordered */
};

#define SIN6_LEN      /* required for compile-time tests */

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 */
};
```

Figure: IPv6 Socket Address Structure

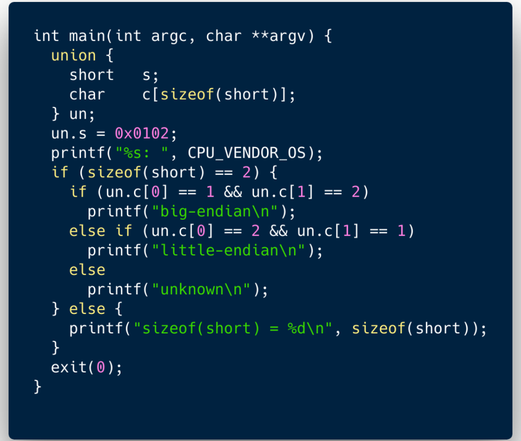
Value-Result Arguments



Byte Ordering



byteorder.c

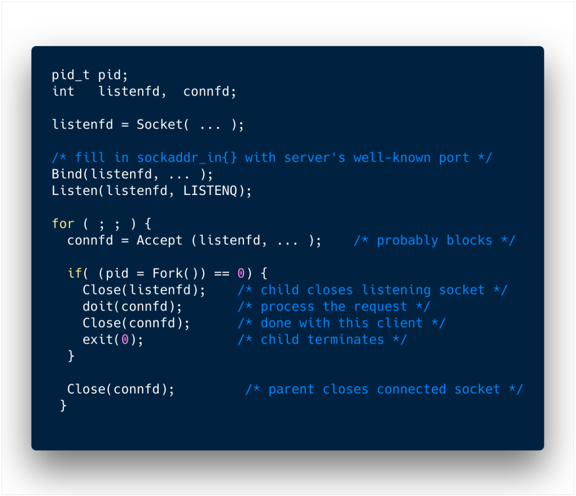


```c
int main(int argc, char **argv) {
  union {
    short   s;
    char    c[sizeof(short)];
  } un;
  un.s = 0x0102;
  printf("%s: ", CPU_VENDOR_OS);
  if (sizeof(short) == 2) {
    if (un.c[0] == 1 && un.c[1] == 2)
      printf("big-endian\n");
    else if (un.c[0] == 2 && un.c[1] == 1)
      printf("little-endian\n");
    else
      printf("unknown\n");
  } else {
    printf("sizeof(short) = %d\n", sizeof(short));
  }
  exit(0);
}
```

Elementary Sockets Functions

- `socket`
- `connect`
- `bind`
- `listen`
- `accept`

Concurrent Servers



```
pid_t pid;
int   listenfd,  connfd;

listenfd = Socket( ... );

/* fill in sockaddr_in{} with server's well-known port */
Bind(listenfd, ... );
Listen(listenfd, LISTENQ);

for ( ; ; ) {
  connfd = Accept (listenfd, ... );    /* probably blocks */

  if( (pid = Fork()) == 0) {
    Close(listenfd);     /* child closes listening socket */
    doit(connfd);        /* process the request */
    Close(connfd);       /* done with this client */
    exit(0);             /* child terminates */
  }

  Close(connfd);         /* parent closes connected socket */
 }
```

Figure: Concurrent Servers

`close` Function

- Descriptor Reference Counts
- `shutdown` for mandatory FIN

TCP Client/Server Example

- TCP Echo Client/Server
- Chapter 5—homework and experiment assignment

Summary

- All clients and servers call to `socket`, returning a socket descriptor.
- Clients then call `connect`, while servers call `bind`, `listen`, and `accept`.
- Sockets are normally closed with the standard close function, although another way is the `shutdown` function.