DEVLINUX
ACADEMY

IPC SOCKET

“Code Your Embedded Future”

https://devlinux.vn

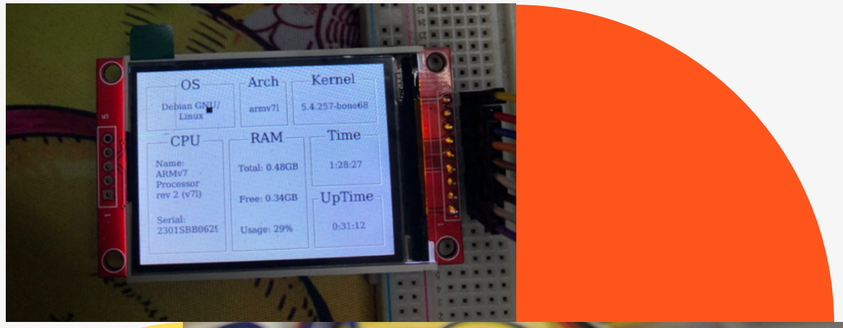
Cộng đồng lập trình Nhúng & Vi Mạch

SUBSCRIBE @devlinux097

DEVLINUX ACADEMY

“Code Your Embedded Future”

AGENDA



- I. Introduction
- II. Work Flow
- III. Sockets: Internet Domain Socker
- IV. Sockets: Unix Domain

https://devlinux.vn

Cộng đồng lập trình Nhúng & Vi Mạch

SUBSCRIBE @devlinux097

DEVLINUX
ACADEMY

INTRODUCTION

https://devlinux.vn
Cộng đồng lập trình Nhúng & Vi Mạch
SUBSCRIBE @devlinux097

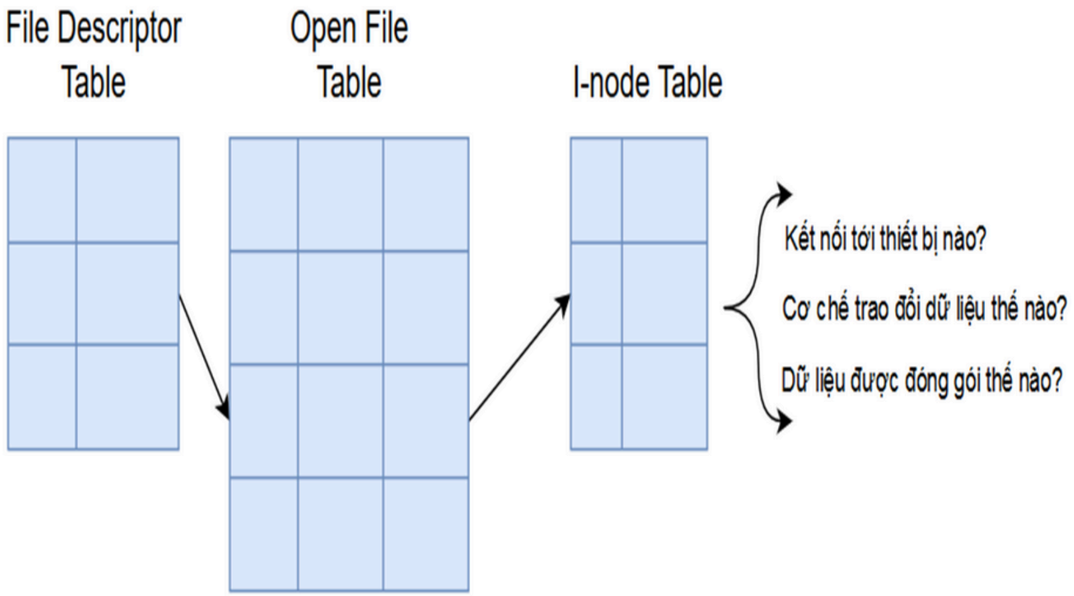
“Code Your Embedded Future”

What is socket?

Sockets are communication mechanisms that allow processes to communicate with each other whether they are on the same device or different devices.

Sockets are represented by a socket descriptor file.

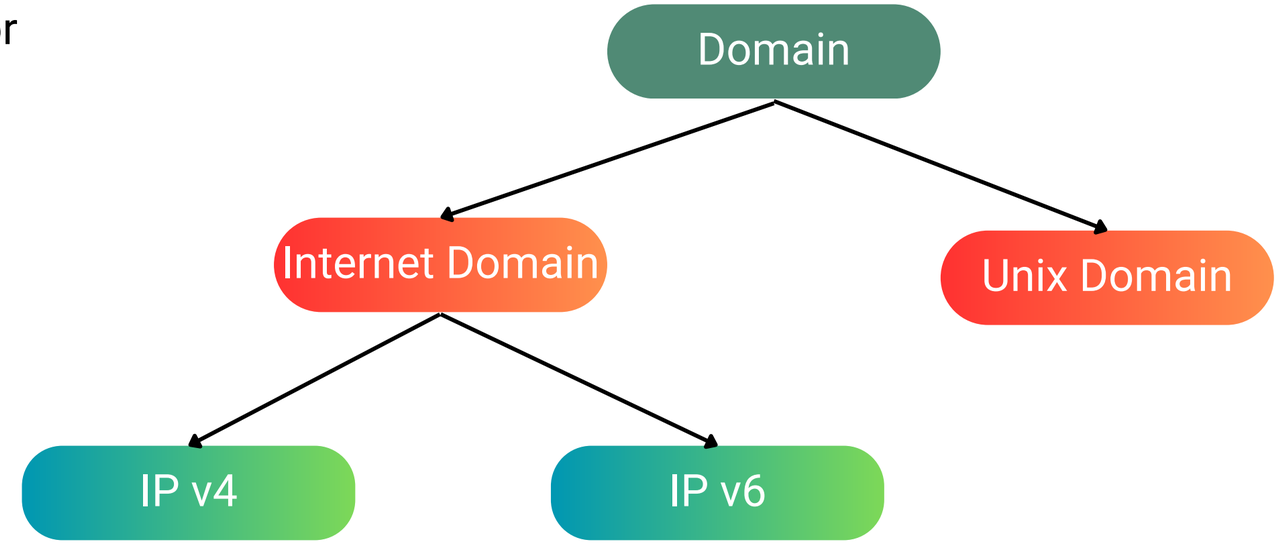
Information described in a socket file will include: Domain, Type, Protocol



Domain

The process that needs to communicate is on the same or different device. Sockets have two main domains:

- Internet Domain
- UNIX Domain.



Type Socket

Type

Description of the information transmission mechanism
Socket has 2 common types:

- Stream
- Datagram.

Stream Socket	Datagram Socket
Reliable (ensures data is received in order, with notification if errors occur)	Unreliable (data received may be out of order, data may be lost in transmission without notice)
Require connection creation before data exchange	No need to create a connection before exchanging data. Data can be sent even if the destination process
Often used when the data to be transmitted is a bit string.	Often used if the data to be transmitted is packets

Protocol

Protocol

How to package data.

From Domain and Type there will be a list of corresponding protocols for us to choose from.

Normally with a selected Domain and Type there is only one protocol that can be used so protocol usually has a value of 0.

Name	Purpose	Man page
AF_UNIX	Local communication	unix(7)
AF_LOCAL	Synonym for **AF_UNIX**	
AF_INET	IPv4 Internet protocols	ip(7)
AF_AX25	Amateur radio AX.25 protocol	**ax25**(4)
AF_IPX	IPX - Novell protocols	
AF_APPLETALK	AppleTalk	ddp(7)
AF_X25	ITU-T X.25 / ISO-8208 protocol	x25(7)
AF_INET6	IPv6 Internet protocols	ipv6(7)
AF_DECnet	DECet protocol sockets	
AF_KEY	Key management protocol, originally developed for usage with IPsec	
AF_NETLINK	Kernel user interface device	netlink(7)
AF_PACKET	Low-level packet interface	packet(7)
AF_RDS	Reliable Datagram Sockets (RDS) protocol	**rds**(7) **rds-rdma**(7)
AF_PPPOX	Generic PPP transport layer, for setting	

DEVLINUX
ACADEMY

WORK FLOW

https://devlinux.vn
Cộng đồng lập trình Nhúng & Vi Mạch
SUBSCRIBE @devlinux097

“Code Your Embedded Future”

Stream Socket

Stream sockets require a connection to be created before transmitting data.

The process that initiates the connection acts as the client, and the process that receives the connection request is the server.

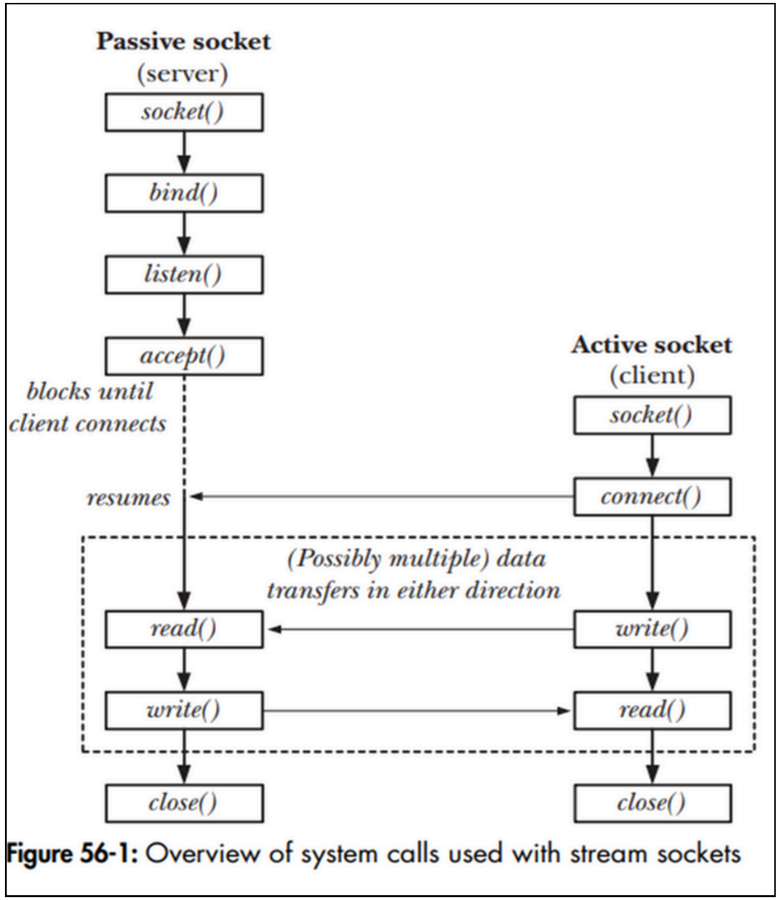


Figure 56-1: Overview of system calls used with stream sockets

Datagram Socket

In Datagram sockets, the roles of client and server are quite vague.

Basically, processes can send data to an address regardless of whether that address exists or not.

In the transmission and reception process, we temporarily consider the process that wants to send data as the client and the process that receives data as the server.

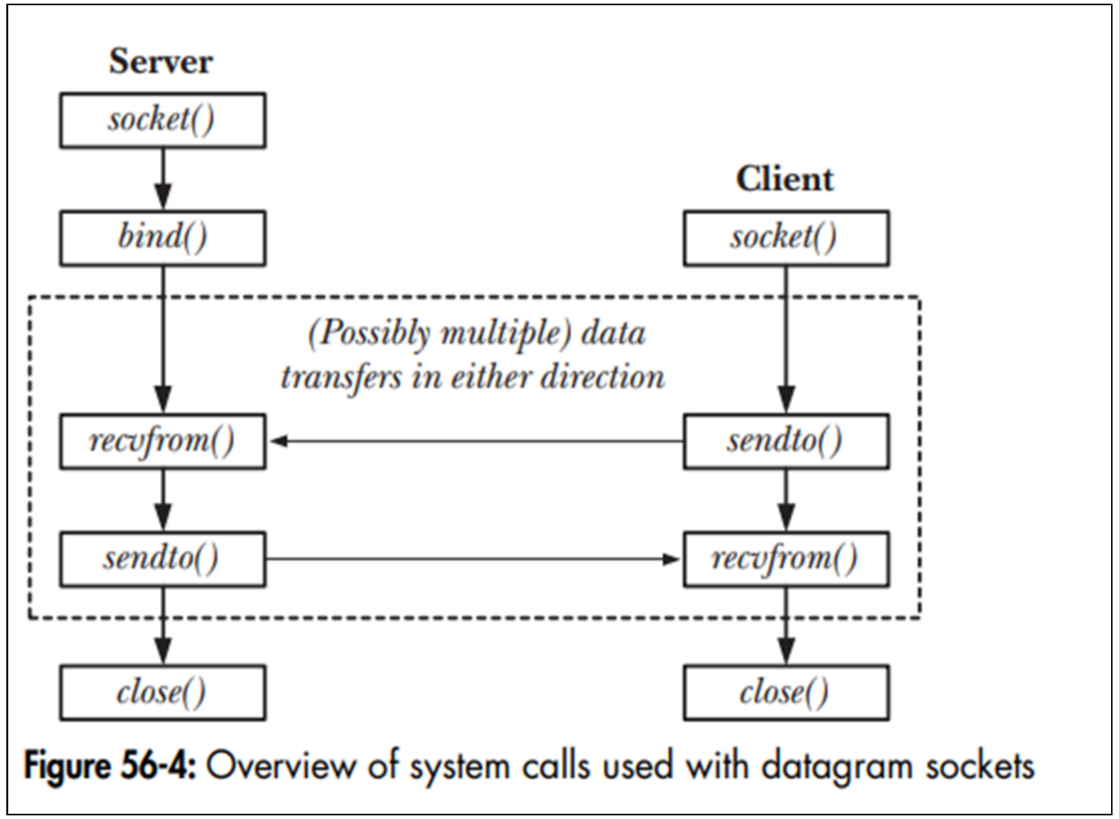


Figure 56-4: Overview of system calls used with datagram sockets

DEVLINUX
ACADEMY

SOCKET: INTERNET DOMAIN SOCKET

https://devlinux.vn
Cộng đồng lập trình Nhúng & Vi Mạch
SUBSCRIBE @devlinux097

“Code Your Embedded Future”

Internet Socket Address

Used to communicate between processes on different devices.

Domain: AF_INET/AF_INET6

Socket has only one address type, sockaddr

However, for convenience in using with different socket domains, people define additional address structs for each domain and then cast them to struct socaddr

struct sockaddr

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

IPv4 Socket Address

struct sockaddr_in

```
struct sockaddr_in {
    sa_family_t sin_family;  /* Address family (AF_INET) */
    in_port_t sin_port;      /* Port number */
    struct in_addr sin_addr;  /* IPv4 address */
    unsigned char __pad[X];  /* Pad to size of 'sockaddr' structure (16 bytes) */
};
```

struct in_addr

```
struct in_addr { /* IPv4 4-byte address */
    in_addr_t s_addr; /* Unsigned 32-bit integer */
};
```

IPv6 Socket Address

struct sockaddr_in6

```
struct sockaddr_in {
    sa_family_t sin6_family;   /* Address family (AF_INET6) */
    in_port_t sin6_port;       /* Port number */
    struct in6_addr sin6_addr;  /* IPv6 address */
   uint32_t sin6_scope_id;    /* Scope ID (new in kernel 2.4) */
};
```

struct in6_addr

```
struct in_addr { /* IPv6 address structure */
   uint8_t s6_addr[16];       /* 16 bytes == 128 bits */
};
```

Convert Socket Address

The address of an Internet Socket is represented by an IP address and a port. Both are stored on the device as integers.

Devices using different hardware architectures will store addresses in different orders.

Sockets use a common convention for storing addresses called network byte order (actually Big-endian order)

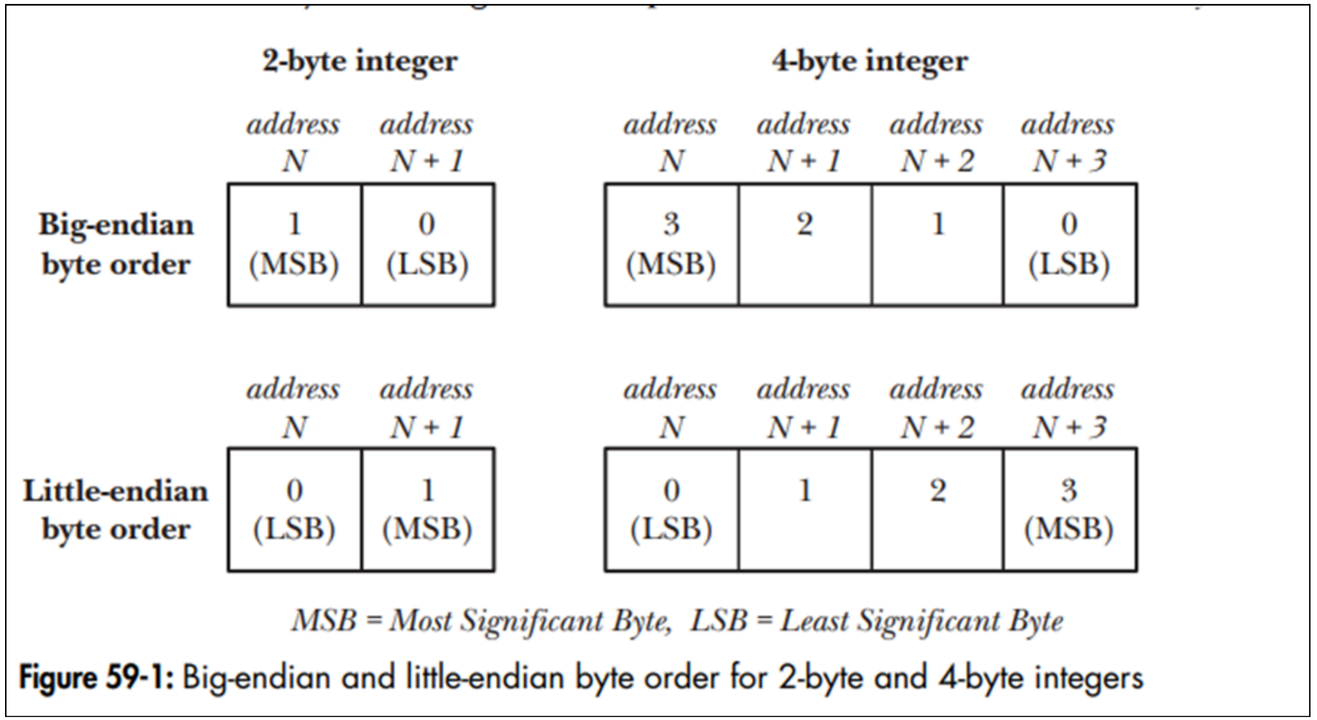


Figure 59-1: Big-endian and little-endian byte order for 2-byte and 4-byte integers

Convert Socket Address

Functions used to convert Socket addresses

```
#include <arpa/inet.h>
uint16_t htons(uint16_t host_uint16);
    Returns host_uint16 converted to network byte order

uint32_t htonl(uint32_t host_uint32);
    Returns host_uint32 converted to network byte order
```

```
#include <arpa/inet.h>
uint16_t ntohs(uint16_t net_uint16);
    Returns net_uint16 converted to host byte order

uint32_t ntohl(uint32_t net_uint32);
    Returns net_uint32 converted to host byte order
```

DEVLINUX
ACADEMY

SOCKET: UNIX DOMAIN SOCKET

https://devlinux.vn
Cộng đồng lập trình Nhúng & Vi Mạch
SUBSCRIBE @devlinux097

“Code Your Embedded Future”

Unix Socket Address

Linux supports Internet Sockets for communication between processes on the same device or over a network.

Although Internet Sockets can be used for communication on the same device, UNIX Domain Socket (UDS) is faster and easier to use in this case.

The domain used for UNIX Domain Socket is: AF_UNIX (or AF_LOCAL)

UNIX Domain Socket supports 2 main types:

- SOCK_STREAM (stream-like communication - TCP-like)
- SOCK_DGRAM (datagram-like communication - UDP-like)

Protocol is always 0

- socket(AF_UNIX, SOCK_STREAM, 0)

Unix Socket Address

After running bind() to assign the address to the socket, a socket file will be created according to path_name

A socket cannot be assigned to an existing path_name.

A path_name can only be assigned to a socket.

The path_name can be an absolute path (/home/phonglt/path_name) or a relative path (./path_name)

Although the socket is characterized by a socket file, we cannot use open() to connect to the socket.

After the socket is closed or the program is shut down, the path_name file still exists. If we want to delete this file, we can use unlink() or remove().

struct sockaddr_un

```
struct sockaddr_un {
    sa_family_t sun_family; /* Always AF_UNIX */
    char sun_path[108]; /* Null-terminated socket pathname */
};
```

Unix Socket Address

To connect or send data to a socket requires the process to have write permission to the file path_name.

The bind() command will create a socket file with full permissions for all accounts, but we can change their permissions using umask() or simply change the permissions of the directory containing the socket file.

DEVLINUX
ACADEMY

Thanks for Your Attention!

Liên hệ với chúng tôi

https://devlinux.vn

Cộng đồng lập trình Nhúng & Vi Mạch

SUBSCRIBE @devlinux097