Sockets

- The original method for process communication in UNIX is *pipes*.
- A disadvantage with pipes is that they can only be used by processes that have the same parent process.
- When communicating across a network a more general method is needed.
- Sockets is a very general communications interface developed at Berkeley.
- A *socket* is an endpoint of communication.
- A *socket* usually has an *address* bound to it.
- Different networks use different addressing methods.
- To handle different addressing methods a socket belongs to a *communication domain* that determines the addressing method to be used.

Sockets

Communication domains

All processes that communicates in the same *communication domain* use the same *address format*.

Two communication domains are implemented in all systems:

- AF_UNIX. The UNIX domain. This domain is used for local communication and uses file system path names as addresses.
- AF_INET. The Internet domain. This domain uses the Internet protocols (TCP/IP) and Internet addresses (a 32 bit host number and a 16 bit port number).

Socket types

There are several socket types, which represent classes of service.

Each type may or may not be implemented in any communication domain.

The socket type like the communication domain are selected when the socket is created.

Examples of socket types are:

- SOCK_STREAM. Provides reliable duplex sequenced data streams (virtual circuit). In the Internet domain this is supported by TCP (Transmission Control Protocol). In the UNIX domain , pipes may be implemented as a pair of stream sockets.
- SOCK_DGRAM. Provides an unreliable datagram service. Datagrams do not use a connection and requires a complete address to be included in each packet. Supported by UDP in the Internet domain.
- SOCK_RAW. Raw sockets allow direct access to lower level protocols. In the Internet domain, IP can be reached with a raw socket (requires root privileges).

System calls for sockets

There is a number of system calls specific to the socket mechanism:

Socket Creates a socket. Returns a socket descriptor to be used in the same way as a file descriptor.

bind Bind a socket to an address so it can be addressed from another process.

listen Tells the kernel that the process is ready to receive connections and how many pending connections to queue until connections are refused.

accept Accept a single connection. Blocks until the socket is called. Returns a new socket descriptor for the new connection.

connect Connect to a named socket in another process.

sendto Send a message to a datagram socket.

recvfrom Receive a message from a datagram socket.

Socket - Server code

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <sys/un.h>
#include <unistd.h>
int main() {
   int sd, ns, len;
   struct sockaddr_un server_address, client_address;
   char buf[256];
    //create a socket for the server
    sd = socket(AF_UNIX, SOCK_STREAM, 0);
   //Name the socket
   server_address.sun_family = AF_UNIX;
   strcpy(server_address.sun_path, "sockname");
   len = sizeof(server_address);
   if (bind(sd, (struct sockaddr *)&server_address, len) < 0)
       printf("Cannot bind\n");

   //Create a connection queue and wait for clients.
   listen(sd, 2);
   while(1) {
      //Accept a connection.
      len = sizeof(client_address);
      ns = accept(sd, (struct sockaddr *)&client_address, &len);
      if (fork() == 0) { /* child */
         close(sd);
         read(ns, buf, sizeof(buf));
         printf("server read %s\n",buf);
         exit(0);
      }
      close(ns);
   }
}
```


Socket - Client code

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
int main ()
{
   int sd, len;
   struct sockaddr_un address;
   int result;

   //create a socket for the client
   sd = socket(AF_UNIX, SOCK_STREAM, 0);

   //Name the socket as agreed with the server
   address.sun_family=AF_UNIX;
   strcpy(address.sun_path,"sockaddr");
   len = sizeof(address);

   //Now connect our socket to the server's socket
   result = connect(sd, (struct sockaddr *)&address, len);
   if(result == -1) exit (1);
   write(sd, "hi guy", 6);
   //Close the socked
   close(sd);
   return (0);
}
```

Network support

The network usually used for long distance communication is the global Internet.

This network uses the Internet protocols which originate from the Arpanet.

For performance reasons, the lower level protocols (IP, TCP, UDP) are implemented within the kernel.

The upper level protocols on the other hand are implemented by user level programs.

Examples of such protocols are:

FTP - File Transfer Protocol

Telnet - Remote login

Both FTP and Telnet originate from the early days of the Arpanet (1971) and completely lack all security mechanisms.

Today these protocols should be replaced with newer encrypting protocols such as *scp* and *ssh*.

Network Support

The most well known model for describing communication architectures is the ISO OSI model.

The Internet protocols are more related to the “Arpanet Reference Model” ARM.

ARM was developed for Arpanet and is older than the OSI model.

The OSI model usually have one protocol per layer.

ARM allows several protocols in each layer

ARM have the following layers:

Process/Applications Corresponds to the application, presentation and session layers in the OSI model. The ftp, telnet and ssh protocols are at this level.

Host-host Corresponds to the transport layer and part of the network layer in the OSI model. TCP and IP are at this level.

Network_interface Corresponds to part of the network layer and the data link layer in the OSI model. The Ethernet driver is at this level.

Network_hardware This level is not formally included in the ARM model but every network has hardware corresponding to the physical level in the OSI model.

Network support

Adding new protocols

Because TCP and UDP provides transparent communication between two processes, changes in the protocols above this level only impose changes in user level programs.

Changes in the transport protocols (TCP, UDP) will require changes in all involved operating systems.

Changing IP is a serious undertaking that will require changes in all operating systems and in all routers (in the world). (The work to replace IPv4 started in 1990 and still IPv6 is only used by a fraction of all computers)