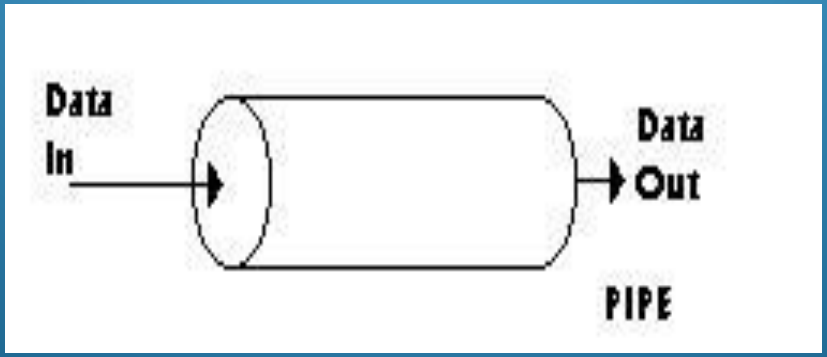
CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Processes that are **related** by creation hierarchy and run on the same machine can use a fast kernel based stream communication mechanism known as a pipe.

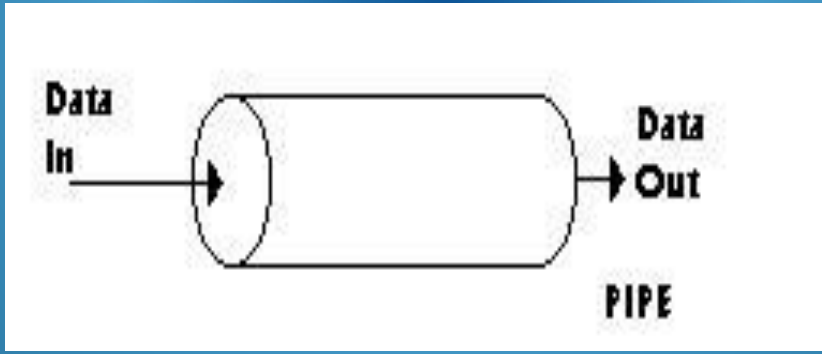
Pipes are FIFO byte stream communication channels implemented with finite size memory buffers maintained by the kernel.



CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

A process writes data to one end of the pipe, and usually another process reads it from the other end. A pipe exists for the lifetime of the processes that have access to it.



A pipe can only be used between **related processes** and is generally used as a unidirectional communication channel from parent to child or vice versa. The parent would create the pipe before creating the child so that the child can inherit access to it.

CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Creating a pipe from a C program

The syntax of the pipe system call is as follows:

```
pipe(fdptr);
```

fdptr is a pointer to an array of two integers

i.e. declared in C as

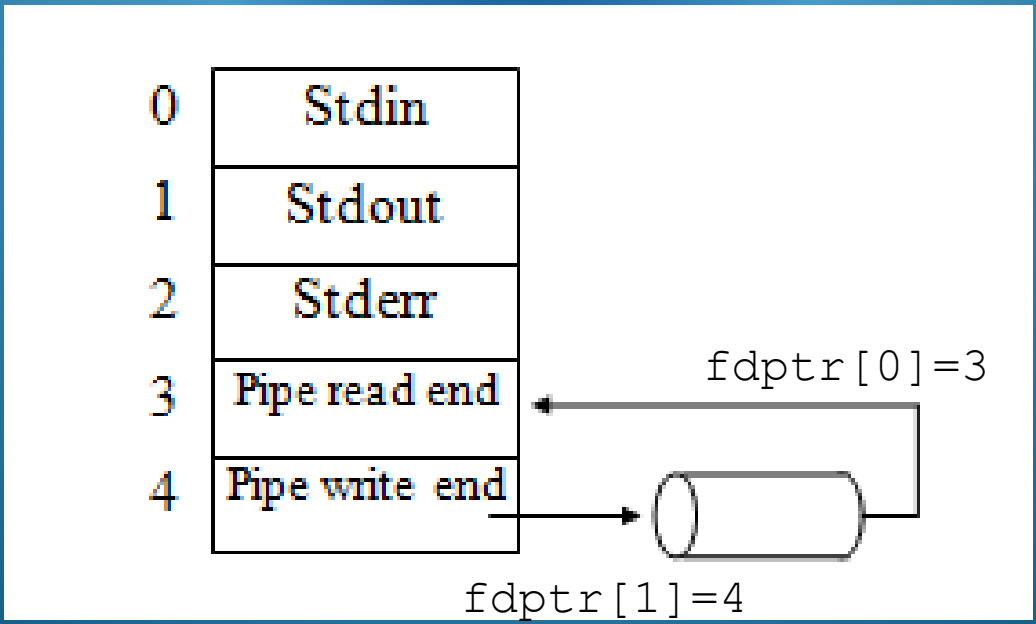
```
int fdptr[2];
```

After the system call to pipe, this array will contain two file descriptors which identify both ends of the new pipe.

CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

`fdptr[0]` is the read descriptor for the pipe and `fdptr[1]` is the write descriptor.
Details of process I/O devices are stored in the process descriptor table.



CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Reading and Writing to Pipes

Data structures needed

```
int fdptr[2], n;
char strbuff[5];
pipe(fdptr);
```

The write system call is used to put data into the pipe.

```
write(fdptr[1], "Welcome to Unix pipes", 21);
```

4

To receive information, a process uses the read system call to get data out of the pipe.

```
n = read(fdptr[0], strbuff, 5);
```

3

CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Reading and Writing to Pipes

```
n = read(fdptr[0], strbuff, 5);
```

If the pipe is empty, the read function blocks the calling process until data can be read from the pipe.

If the pipe is not empty but the process attempts to read more data than is in the pipe, the read succeeds returning all data in the pipe and the count of bytes actually read.

CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Named Pipes

A variation of this mechanism known as a *named pipe*, can be used for communication between unrelated processes that have access to the same file name space.

A named pipe is implemented as a special FIFO file by the kernel, rather than as a memory buffer, and so is accessible to independent processes through the file system's shared name space.

Named pipes can be shared across machines with a common file system.

CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Creating and Using Named Pipes

To create a named pipe file called "`mypipe`" in the current directory, use:-

```
mknod("mypipe", 010777, 0);
```

The file type is a fifo special file indicated by the code 010

The access permissions to the pipe allow read, write and execute permission for the owner, group and world with a value 777

The last parameter to mknod is irrelevant for fifo pipes.

CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Processes open **named pipes** in the same way as regular files, so unrelated processes can communicate by opening the pipe and either reading or writing it.

A process can open a pipe in read or write mode.

```
namedpipe = open("mypipe",O_WRONLY);
namedpipe = open("mypipe",O_RDONLY);
```

`O_WRONLY` and `O_RDONLY` are constants found in the header file "`fcntl.h`"

```
#include <fcntl.h>
```

CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Semantics

A process opening the pipe for reading will block if there is no process that has it open for writing.

Similarly, a process opening the pipe for writing will block if there is no process that has it open for reading.

The format of the read and write operations is as follows:-

```
n = read(namedpipe, buffer, nbytes);
write(namedpipe, buffer, nbytes);
```

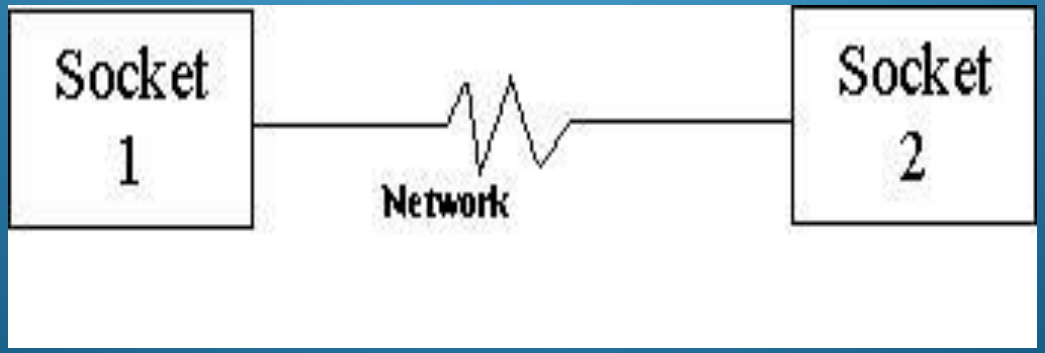
CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Network Sockets

Sockets are the more usual and general purpose means of communication between **independent processes** and can be used when neither kernel data structures or files can be shared, for example communicating across the Internet.

A communication channel can be visualized as a pair of communication end-points.

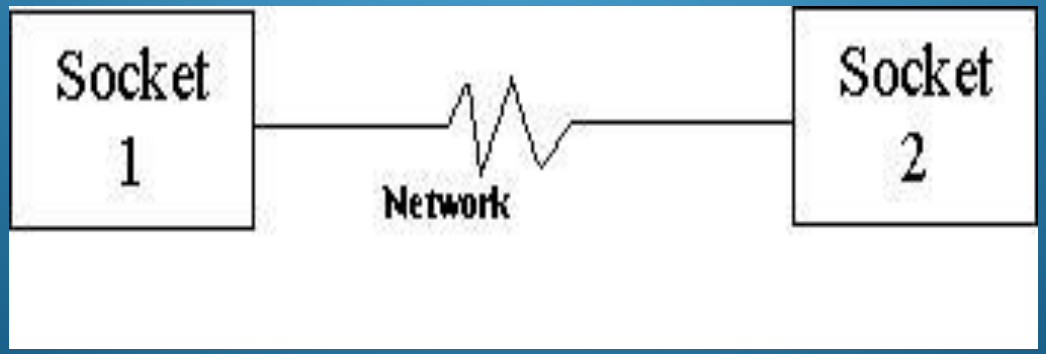


CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Network Sockets

A socket is a data structure created by an application to represent a communication channel. The socket must be bound or associated to some entity (service access point/port) within the communication system. By using communication primitives of the socket interface, the process then exchanges this data from its address space to the address space of the communication subsystem which handles delivery.

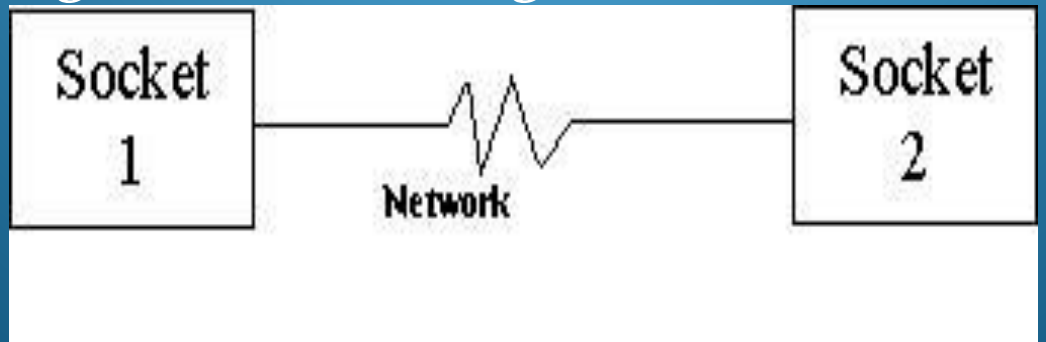


CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Network Sockets

The communication service (e.g TCP/IP) provides the glue for establishing connections between sockets belonging to different processes, which are often on different machines and networks and for transporting and routing messages through the network to a destination. On the Internet, IP addresses and port numbers are used as a means of identifying the endpoints of a connection to which a program's socket might be connected.

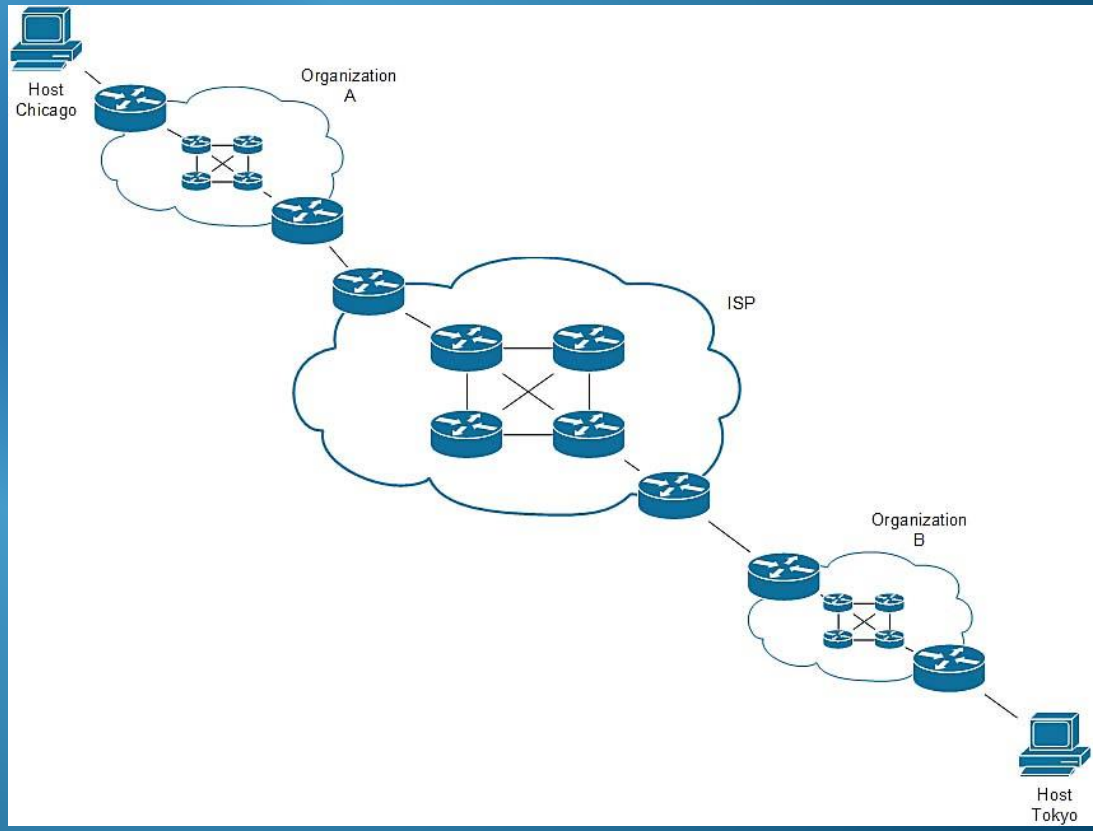


CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

Network Sockets

A communication subsystem which understands the same messaging protocols must be running on every point along the network over which the message travels. In an Internet context, this subsystem implements a communication protocol known as TCP/IP.



CS240 Operating Systems, Communications and Concurrency

Unix Interprocess Communication Mechanisms

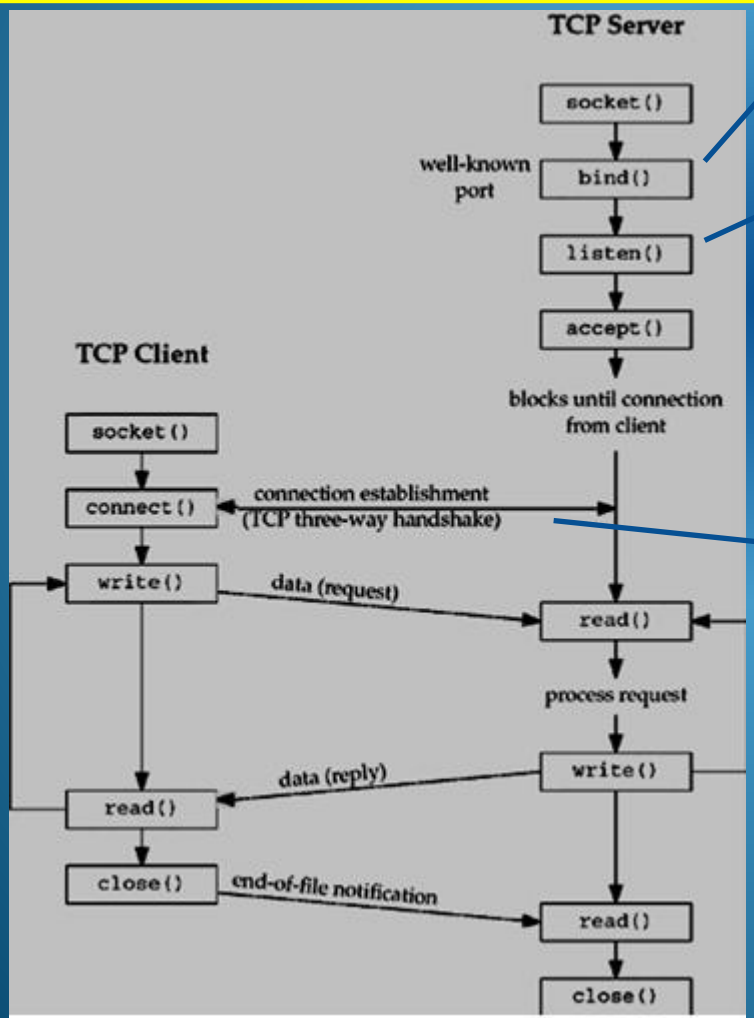
Different levels of reliability for message delivery through the socket may be specified to the transport control protocol (TCP) when it is created.

Faster delivery can be traded for less reliability.
DATAGRAM vs CONNECTION ORIENTED

A socket may be 'connected' to any named remote socket by the underlying communication protocol. After connection, bytes which are sent to the local socket are delivered to the remote socket. The remote process 'listens' to its socket and receives the data sent into the byte stream.

CS240 Operating Systems, Communications and Concurrency

Socket based Client/Server Communication



Associate socket with named IP port

Marks the socket as a passive socket, a socket that will be used to accept incoming connection requests using accept