**IPC Systems – Windows**

**Ordinary/anonymous Pipes**

* Ordinary Pipes allow communication in standard producer-consumer style
* Producer writes to one end (the write-end of the pipe)
* Consumer reads from the other end (the read-end of the pipe)
* Ordinary pipes are therefore unidirectional
* Used for communication between **related processes** (e.g., parent and child processes).
* Require parent-child relationship between communicating processes
* Exists only while processes are running and disappears after termination.
* Windows calls these anonymous pipes

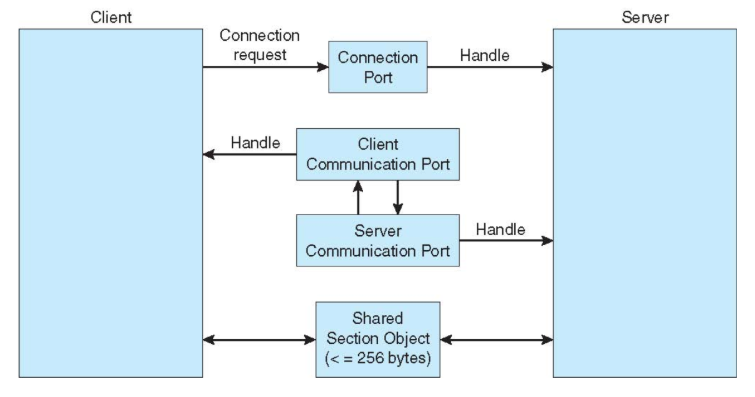
**Named Pipes**

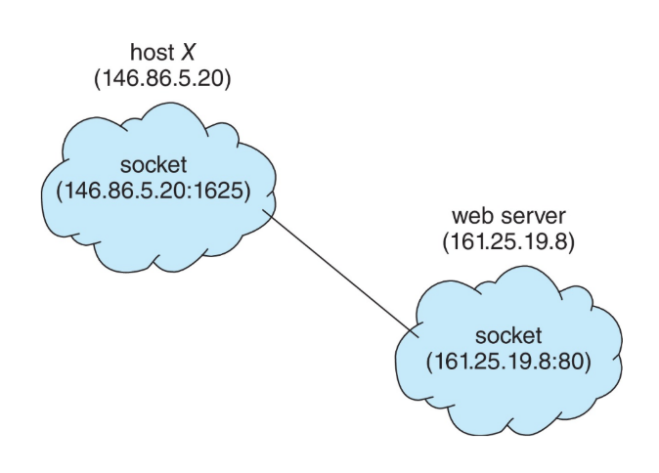
* Named Pipes are more powerful than ordinary pipes
* Communication is bidirectional
* Used for communication between **unrelated** or **independent** processes.
* No parent-child relationship is necessary between the communicating processes
* Several processes can use the named pipe for communication
* Persists even after the processes finish execution
* Provided on both UNIX and Windows systems

Windows provides **Inter-Process Communication (IPC)** mechanisms to facilitate data exchange between processes. One of the core IPC methods in **Windows** is through **Advanced Local Procedure Call (ALPC)**, which is message-passing-centric.

**Advanced Local Procedure Call (ALPC) in Windows**

1. **Purpose**:
   * Used for high-speed **message-passing** between processes on the **same system**.
   * Mainly utilized for **local** communication between user-space and kernel-space.
2. **Characteristics**:
   * **Local-Only**: Works **within** the same system, not across networks.
   * **Port-Based**: Uses **ports (mailboxes)** to manage and maintain communication.
   * **Synchronous & Asynchronous**: Supports both forms of communication.
   * **Security**: Provides secure communication through **access control**.
3. **Communication Process**:
   * **Step 1**: The **client** opens a handle to the **server’s connection port** (an object that facilitates communication).
   * **Step 2**: The **client** sends a **connection request** to the server.
   * **Step 3**: The **server** creates **two private ports**:
     + One for actual **data transfer**.
     + One for handling **callbacks** or additional communication.
   * **Step 4**: The **server** gives the client a handle to one of these ports.
   * **Step 5**: Both **client** and **server** use their assigned **port handles** to exchange **messages** and **listen** for responses.
4. **Applications**:
   * Communication between **Windows subsystems** (e.g., Win32 and NT kernel).
   * Used by **Windows Services** for secure, efficient messaging.
   * Common in **antivirus** software and **driver** communication.



**IPC Systems - Client-Server Systems**

**1. Sockets**

A socket is essentially an endpoint for communication between two processes over a network.

It acts as an interface through which a process can send and receive data.

Sockets are used to establish connections between processes, regardless of whether they are on the same machine or on different machines.

* **Addressing Network Services:**
  + A socket address is typically represented as a combination of an IP address and a port number.
  + The IP address identifies the host (the computer) on the network.
  + The port number identifies a specific process or service running on that host.
  + This concatenation allows a single host to run multiple network services simultaneously.
* **Example:**
  + 161.25.19.8:1625 indicates that the communication is directed to port 1625 on the host with the IP address 161.25.19.8.
* **Local Communication:**
  + The IP address 127.0.0.1, also known as the "loopback address," is a special address that refers to the local host (the computer on which the process is running).
  + It's used for communication between processes on the same machine, without involving the external network.
  + This is very useful for testing network applications.

**2. Remote Procedure Calls (RPC) in Client-Server Systems**

**Remote Procedure Calls (RPC)** enable processes to execute procedures (functions) **remotely**—across **different machines** in a distributed system.

1. **Purpose**:
   * Allow a **client** process to **invoke** a procedure (function) on a **remote** server as if it were a **local** function.
   * Abstracts the complexity of **network communication** from the programmer.
2. **Stubs and Marshalling:**

RPCs use "stubs" (proxy functions) to handle the network communication and data conversion.

"Marshalling" is the process of packaging the parameters into a message for transmission, and "unmarshalling" is the reverse process.

1. **How RPC Works**:
   * **Step 1**: The **client** sends a **request** to the server by invoking a **stub** (a proxy function that mimics a local call).
   * **Step 2**: The request is **serialized** (converted into a format suitable for transmission).
   * **Step 3**: The message is sent to the **server** using network protocols (**TCP/IP** or **UDP**).
   * **Step 4**: The **server** receives and **deserializes** the request.
   * **Step 5**: The **server** executes the requested **procedure** and **sends a response** back.
   * **Step 6**: The **client** receives the **result** and resumes execution.
2. **Types of RPC**:
   * **Synchronous RPC**: Client waits for a **response** before proceeding.
   * **Asynchronous RPC**: Client sends a request and **continues** without waiting.
3. **RPC in Windows**:
   * Uses **Microsoft RPC** (MSRPC), based on the **Distributed Computing Environment (DCE)** model.
   * Facilitates **networked communication** between Windows applications.
   * Used by Windows for services like **file sharing**, **Active Directory**, and **remote administration**.
4. **Advantages of RPC**:
   * **Transparency**: Hides network complexities from the user.
   * **Efficiency**: Enables **distributed systems** to work seamlessly.
   * **Language Agnostic**: Works across different programming languages.
5. **Examples of RPC Usage**:
   * **Remote Desktop Protocol (RDP)** for managing Windows systems remotely.
   * **Database servers** where clients make **queries** to remote databases.
   * **Distributed File Systems** where file operations are performed across networks.

A diagram of a server

AI-generated content may be incorrect.