Unix Domain Sockets: Local IPC with File System-Based Endpoints

Unix Domain Sockets (UDS) are a powerful form of **inter-process communication (IPC)** available on Unix-like operating systems. Unlike TCP/IP sockets that communicate over a network, UDS operates **locally** between processes on the same host, using the **file system as the addressing namespace**.

📁 How It Works

1. **Server process** creates a socket and binds it to a pathname.
2. **Client process** connects to that path to initiate communication.
3. They exchange data via `read/write`, `send/recv`.

The kernel mediates all interactions without going through the network stack.

📦 Advantages

- **Faster than TCP/IP**: No overhead of IP stack, no checksum/fragmentation.
- **Secure**: Access controlled via file system permissions (`chmod`, `chown`).
- **No Port Conflicts**: Uses file paths instead of numeric ports.
- **Pass File Descriptors**: Can send open file descriptors using `sendmsg()` and `SCM_RIGHTS`.

⚙️ Real-World Integration Example

Nginx Configuration

```
upstream php_backend {
    server unix:/run/php/php7.4-fpm.sock;
}
```

This avoids TCP overhead by communicating over a Unix socket instead of localhost TCP.

⚠️ Considerations

- Path length limit (often 108 bytes for `sun_path`).
- Must manually clean up the socket file on shutdown.
- Only usable on the same host – no cross-network communication.

Unix Domain Sockets are an ideal choice when building fast, secure, and local IPC mechanisms. They’re widely used across system-level applications and container ecosystems.