api-simple-ipc

Simple-IPC API ==============

The Simple-IPC API is a collection of `ipc_` prefixed library routines and a basic communication protocol that allows an IPC-client process to send an application-specific IPC-request message to an IPC-server process and receive an application-specific IPC-response message.

Communication occurs over a named pipe on Windows and a Unix domain socket on other platforms. IPC-clients and IPC-servers rendezvous at a previously agreed-to application-specific pathname (which is outside the scope of this design) that is local to the computer system.

The IPC-server routines within the server application process create a thread pool to listen for connections and receive request messages from multiple concurrent IPC-clients. When received, these messages are dispatched up to the server application callbacks for handling. IPC-server routines then incrementally relay responses back to the IPC-client.

The IPC-client routines within a client application process connect to the IPC-server and send a request message and wait for a response. When received, the response is returned back to the caller.

For example, the `fsmonitor--daemon` feature will be built as a server application on top of the IPC-server library routines. It will have threads watching for file system events and a thread pool waiting for client connections. Clients, such as `git status`, will request a list of file system events since a point in time and the server will respond with a list of changed files and directories. The formats of the request and response are application-specific; the IPC-client and IPC-server routines treat them as opaque byte streams.

Comparison with sub-process model ---------------------------------

The Simple-IPC mechanism differs from the existing `sub-process.c` model (Documentation/technical/long-running-process-protocol.adoc) and used by applications like Git-LFS. In the LFS-style sub-process model, the helper is started by the foreground process, communication happens via a pair of file descriptors bound to the stdin/stdout of the sub-process, the sub-process only serves the current foreground process, and the sub-process exits when the foreground process terminates.

In the Simple-IPC model the server is a very long-running service. It can service many clients at the same time and has a private socket or named pipe connection to each active client. It might be started (on-demand) by the current client process or it might have been started by a previous client or by the OS at boot time. The server process is not associated with a terminal and it persists after clients terminate. Clients do not have access to the stdin/stdout of the server process and therefore must communicate over sockets or named pipes.

Server startup and shutdown ---------------------------

How an application server based upon IPC-server is started is also outside the scope of the Simple-IPC design and is a property of the application using it. For example, the server might be started or restarted during routine maintenance operations, or it might be started as a system service during the system boot-up sequence, or it might be started on-demand by a foreground Git command when needed.

Similarly, server shutdown is a property of the application using the simple-ipc routines. For example, the server might decide to shutdown when idle or only upon explicit request.

Simple-IPC protocol -------------------

The Simple-IPC protocol consists of a single request message from the client and an optional response message from the server. Both the client and server messages are unlimited in length and are terminated with a flush packet.

The pkt-line routines (linkgit:gitprotocol-common[5]) are used to simplify buffer management during message generation, transmission, and reception. A flush packet is used to mark the end of the message. This allows the sender to incrementally generate and transmit the message. It allows the receiver to incrementally receive the message in chunks and to know when they have received the entire message.

The actual byte format of the client request and server response messages are application specific. The IPC layer transmits and receives them as opaque byte buffers without any concern for the content within. It is the job of the calling application layer to understand the contents of the request and response messages.

Summary -------

Conceptually, the Simple-IPC protocol is similar to an HTTP REST request. Clients connect, make an application-specific and stateless request, receive an application-specific response, and disconnect. It is a one round trip facility for querying the server. The Simple-IPC routines hide the socket, named pipe, and thread pool details and allow the application layer to focus on the task at hand.

□□□ #2
□ □□□ □□□ 2026-10-31 20:16:10 UTC
□ □□□ □□□ 2026-10-31 20:16:11 UTC