An IPC server (with Unix sockets) in Elixir

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

First publication of this article on 7 November 2022

https://www.bortzmeyer.org/elixir-ipc-server.html

While adding IPC support to an Internet server written in Elixir, I did not find a lot of documentation or examples about Unix sockets with Elixir. So, here it is.

Unix sockets (also mentioned as "domain" or "local") are a way to communicate between processes (hence IPC = Inter-Process Communication) **on the same machine**. As such, they are less general than TCP/IP sockets but they are more secure (you cannot connect from the outside, and you can retrieve the process identifier, user identity, etc of the client) and you can finely control access through normal file permissions since the socket appears as a file.

To have the server written in Elixir listen on such a socket, we first clean (we use the standard module File <https://hexdocs.pm/elixir/File.html>):

```
sockname = System.get_env("HOME") <> "/.test-ipc.sock"
File.rm(sockname) # We don't test the results, since we are not interested
   # if the file existed or not.
```

Then we can open the socket, using the Erlang socket library <https://www.erlang.org/doc/man/socket.html>, indicating the type `:local`:

```
{:ok, s} = :socket.open(:local, :stream, %{})
:ok = :socket.bind(s, %{family: :local, path: sockname})
# You can add File.chmod here to set permissions
:ok = :socket.listen(s)
```

Note that we use pattern matching to be sure the operations succeeded (we would have better error messages with a code testing the different cases).

If you get an `ErlangError {:invalid, {:protocol, %{}}}`, this is probably because your Elixir (actually, OTP) is too old. We need at least version 12 of Elixir (OTP 24). You can always catch the error to produce a better error message (see the complete source code).

Now, we can accept connections (the rest of the code is not specific to Unix local sockets) :

```
{:ok, client} = :socket.accept(socket)
```

And read from the clients, and write to them (here, we just echo the message), like we would do with more common TCP/IP sockets :

```
{:ok, result} = :socket.recv(socket, 0)
msg = String.trim(result)
Logger.info("Received \"#{msg}\"")

:ok = :socket.send(socket, msg <> "\r\n")
```

A complete echo server using these techniques is . Just run it :

```
% elixir server.exs
Listening on /home/stephane/.test-ipc.sock
```

Clients can be netcat :

```
% netcat -U ~/.test-ipc.sock

allo
allo


Tu es là?
Tu es là?
```

Or socat :

```
% echo foobar | socat - UNIX-CONNECT:/home/stephane/.test-ipc.sock
foobar
```

Or of course a client you write yourself.