SOCKETSERVER

Shannon.Noe at flightaware dot com
Short review of TCP server programming

1. `socket()` // Creates a socket
2. `bind()` // Assign address
3. `listen()` // Join the network
4. `accept()` // Establish connection
Where to create worker processes?

Classic:
socket() bind() listen() accept() fork()

Pre-Fork:
socket() bind() listen() fork() accept()

SO_REUSEPORT:
exec()/fork() socket() bind() listen() accept()
Footnote

SO_REUSEPORT is in TCL TIP 465
OS HAS CONTROL WITH MULTIPLE LISTEN FDS

Multiple accepts are scheduled by OS

SO_REUSEPORT is hashed to processes by address.

Low number IP addresses low scalability on Linux
For a good implementation see Cloudflare's blogs
How to get classic single accept with multiple workers?

Exclusive locks and coordination - Apache Proxy/Broker TCP in userspace
SCM_RIGHTS Apache mod and socketserververtcl
What is SCM_RIGHTS?

Part of the Unix socket specification.

SCM_RIGHTS is a control message which can be sent over SOL_SOCKET.

Provides the ability to pass file descriptors.
SOCKETSERVER

TCL extension which provides a means to send and receive SCM_RIGHTS messages.

This makes it possible to pass TCL sockets.

Programming model follows TCL's core socket command.
package require socketserver

::socketserver::socket server 9901

proc handle_readable ...

proc handle_accept {fd ipaddr port} {
    fileevent $fd readable [list handle_readable $fd]
}

proc make_worker {} {
    set pid [fork]
    if {$pid == 0} {
        # This is the child
        ::socketserver::socket client handle_accept
        vwait done
    }
}

make_worker

vwait done
proc handle_accept {fd} {
    fconfigure $fd -encoding utf-8 -buffering line -blocking 1 -
    while {1} {
        set line [gets $fd]
        if {[string first "quit" $line] != -1} {
            break
        }
    }
    puts $fd "[pid] $line"
}
puts "client closing socket"
close $fd
# Now that we have closed, we are ready for another socket
::socketserver::socket client -port 8888 handle_accept
fork() new client process

accept() get new FD

TCP session established

sendmsg(FD, SCM_RIGHTS)

TCL FD Event

recvmsg(FD, SCM_RIGHTS)

handle_listen(sock, ip, port)

"HELO"

"OK"

close() re-register callback proc