

TCP Server+perl

Source: <http://coding.derkeiler.com/Archive/Perl/comp.lang.perl.misc/2007-04/msg00091.html>

- *From:* "Gauri" <himagauri@xxxxxxxxxx>
 - *Date:* 2 Apr 2007 10:49:25 -0700
-

I'm trying to do socket communication between two machines Linux and MVS.

On Linux side, the code is programmed in Perl and on the MVS side in Rexx.

The Perl code I have is designed as a TCP Server which should process several Clients at the same time.

I've used Forks for this. I close the Parent Server and make the child/Client processes orphans because I want to spawn of a new child/Client each time a new connection is established at the Socket.

But here is what happens. Lets say in the beginning 4 connections are established by the socket, 4 child processes are spawned off, When each child completes processing it closes/ends. In the meanwhile 4 more connections are already established at the socket and are waiting to be spawned off (and ideally each new child should be spawned off immediately, irrespective if the previous children have finished processing or not.).

However in my current scenario, if one of the child process takes more time to complete, the 4 new established connections still wait, for the slower child to finish. It's only when all 4 children spawned off in the first go are completed, that the 4 new waiting processes are spawned off.

I want each child to spawn off as soon as the connection is made.

I believe that perhaps my parent is not being closed properly. Here is the piece of code.

Any suggestions are welcome.

-Thanks,
Gauri

```
# Main loop control variable  
$time_to_die = 0;
```

```
# Set up signal handler, which just sets the global to exit the main  
loop
```

TCP Server+perl

```

$SIG{INT} = $SIG{TERM} = $SIG{HUP} = \&SIG_HANDLER;

# Build the socket: standard Perl boilerplate
socket(SERVER, PF_INET, SOCK_STREAM, getprotobyname('tcp'));
setsockopt(SERVER, SOL_SOCKET, SO_REUSEADDR, 1);
$my_addr = sockaddr_in($PORT, INADDR_ANY);
bind(SERVER, $my_addr) or do {
    $s="Couldn't bind to port $PORT";
    mylog($s, "crit");
    die "$s : $!\n";
};
listen(SERVER, SOMAXCONN) or do {
    $s="Couldn't listen on port $PORT";
    mylog($s, "crit");
    die("$s : $!\n");
};

# Main listen/fork loop: straight-ahead forking server. You exit the
main
# loop by sending a signal to the program, which then sets the global
variable
# $time_to_die in the signal handler, and then the next time through
the
# loop, the program exits and shuts down.

until($time_to_die) {
    $client = accept(CLIENT, SERVER);
    select CLIENT; # By default, print to socket
    $| = 1; # Turn on autoflush so prints happen
    # immediately and there's no
    buffering
    next unless ($client);
    ($port, $packed_ip) = sockaddr_in($client);
    $dq = inet_ntoa($packed_ip);
    mylog("Connection from $dq : $port");
    # Fork and exec main server loop
    $pid = fork();
    die "fork: $!" unless defined $pid;
    if ($pid) {
        # Parent
    } else {
        mylog("Child starting with connection from $dq : $port");
        close(SERVER); # child doesn't need it
        mainloop(); # Go do the actual work
        mylog("Child exiting");
        exit 0;
    }
}

continue {
    close(CLIENT); # Parent doesn't need it

```

TCP Server+perl

```
}  
  
mylog("End of main event loop reached (signal, most likely): Exiting  
now.");  
  
close (SERVER);  
exit 0;  
  
# End of mainline code  
  
.
```