Subject: Device Driver
Posted by harmer on Fri, 08 Apr 1994 08:36:34 GMT
View Forum Message <> Reply to Message

Hi
I am trying to write a C device driver and link this up to
IDL. I started by using 'call external' but found it a bit slow - the device
driver had to be reinitialised on each call, it seemed that anything
done in previous instances of 'call external' is forgotten when control
returns to IDL. Am I doing something wrong?
Any suggestions on the best way of linking a device driver to IDL would be
gratefully accepted as I am a real novice in this area
Another question I have is: what is the 'export.h' file in the IDL
directory for? In the release notes for version 2.2.2 it is mentioned in the
same breath as'linkimage'.
Thanks in advance
Mark Harmer

Subject: re:Device Driver
Posted by harmer on Wed, 13 Apr 1994 06:21:53 GMT
View Forum Message <> Reply to Message

Date: Sat, 9 Apr 94 13:08:35 -0500
From: roberson@hamer.ibd.nrc.ca (Walter Roberson)
Message-Id: <9404091808.AA09821@hamer.ibd.nrc.ca>
To: harmer@ph.und.ac.za
Subject: Re: Device Driver
Newsgroups: comp.lang.idl-pvwave
In-Reply-To: <harmer.1.2DA51792@ph.und.ac.za>
Organization: NRC Institute for Biodiagnostics
Cc: roberson@hamer.ibd.nrc.ca

In article <harmer.1.2DA51792@ph.und.ac.za> you write:
> Hi
> I am trying to write a C device driver and link this up to
> IDL. I started by using 'call external' but found it a bit slow - the device
> driver had to be reinitialised on each call, it seemed that anything
> done in previous instances of 'call external' is forgotten when control
> returns to IDL. Am I doing something wrong?

Often devices are set up to de-initialize themselves when the last user
of the device closes the device. For example, terminal lines may be
set to de-initialize back to the default settings when they are not in
use. The easy way to work around this is to keep something open and connected
to the device for as long as you need it, and then shut it down.

One way to do this would be to SPAWN with NOWAIT the routine that

initialized the driver, with that routine set to go to sleep after it
did the real work. That does a popen() to the SPAWN'd routine and returns
a file descriptor that you can read from or write to if you needed to
communicate with the routine. I suggest perhaps putting the routine to
sleep by having it do a read() or gets() call on standard input; then
when you want the routine to die, just send any \n terminated line of
data down the pipe. CLOSE the file descriptor afterwards.

In theory, the spawned process should die if you close the pipe without
having told the spawned process to exit. Unfortunately, instead it seems
to go zombie, so it is best to find some explicit way to make it die
[eg, by having it exit when it gets some input.]

In comparison, at the pure unix level, if I needed to hold a device
open, I would normally just do something like:

(initialization_command; sleep 30000) < /dev/the_device &

for example

(stty 38400 -icanon -echo; sleep 30000) < /dev/ttyd5 &

Walter Roberson roberson@ibd.nrc.ca
