Subject: SPAWN
Posted by Nobuyuki Tasaka on Tue, 08 Apr 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Hi all,

If someone who have tried to compare SPAWN in the following
usage, please let me know the difference in terms of data
parsing speed, interface flexibility and routine's independency.

I would like to use SPAWN for calling database retriving routine
written in C.

1) SPAWN, "cmd", result

2) SPAWN, "cmd", /UNIT

3) SPAWN, "cmd"
(C routine writes data to memory map file and then IDL read it
as a Logical Unit File)

I'm looking forward to your help.
Nobu

Nobuyuki Tasaka
GE Medical Systems
3200 N Grandview Blvd,
Waukesha, WI 53188
(Phone) 414-521-6577
(E-mail) tasaka@mr.med.ge.com

Subject: Re: SPAWN
Posted by davidf on Fri, 11 Apr 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Nobuyuki Tasaka <tasaka@mr.med.ge.com> writes:

> If someone who have tried to compare SPAWN in the following
> usage, please let me know the difference in terms of data
> parsing speed, interface flexibility and routine's independency.
>
> I would like to use SPAWN for calling database retriving routine
> written in C.
>
> 1) SPAWN, "cmd", result

This is the least flexible way of using SPAWN, I think. You can only ask one "question" at a time and then you have the problem of parsing the "result" to get the answer you want.

> 2) SPAWN, "cmd", /UNIT

This method is much more flexible in terms of interaction with your database, but it is restricted to UNIX platforms. If you want to pass large amounts of data back and forth and can be slow and, of course, you have to make two copies of every piece of data, which is not resource friendly. Call_External would probably be a better choice if you decided to go down this path. It would give you the ability to share memory resources with your database access program.

> 3) SPAWN, "cmd"
> (C routine writes data to memory map file and then IDL read it
> as a Logical Unit File)

This has several huge advantages. First, it is extremely simple to implement. You don't need any special knowledge about operating sytems, linking protocols, etc. And second, it works on every platform IDL supports. You don't have to write special code when you decide to port from your Suns to the PC or Mac. On the downside, if you are transferring large amounts of data it can be slower than you would like.

If you are really serious about doing this, I would look into Call_External as a possibility. Or, perhaps even better, wait for IDL 5 and the new database connection tools. This problem may have already been solved for you! :-)

Cheers!

David

David Fanning, Ph.D.
Fanning Software Consulting
2642 Bradbury Court, Fort Collins, CO 80521
Phone: 970-221-0438 Fax: 970-221-4762
E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: SPAWN
Posted by alpha on Wed, 16 Apr 1997 07:00:00 GMT
View Forum Message <> Reply to Message

davidf@dfanning.com (David Fanning) writes:
> Nobuyuki Tasaka <tasaka@mr.med.ge.com> writes:

>> If someone who have tried to compare SPAWN in the following
>> usage, please let me know the difference in terms of data
>> parsing speed, interface flexibility and routine's independency.
>> I would like to use SPAWN for calling database retriving routine
>> written in C.
>>
>> 2) SPAWN, "cmd", /UNIT

> This method is much more flexible in terms of interaction with
> your database, but it is restricted to UNIX platforms. If you want
> to pass large amounts of data back and forth and can be slow and,
> of course,

ah yes, 130Kbyte/sec maximum

you have to make two copies of every piece of data,
> which is not resource friendly.

Call_External would probably be
> a better choice if you decided to go down this path. It would give
> you the ability to share memory resources with your database
> access program.

Call exeternal with FDDI - call by reference!
I made up to 8Mbytes/sec data transfer directly into IDL-arrays!

>> 3) SPAWN, "cmd"
>> (C routine writes data to memory map file and then IDL read it
>> as a Logical Unit File)

> This has several huge advantages. First, it is extremely simple
> to implement.

Really? can you tell something more about memory-map-Files?
is that a ramdisk-filesystem?

be aware that you will the problem of two copies of the same
data again!

You don't need any special knowledge about
> operating sytems, linking protocols, etc. And second, it works

> on every platform IDL supports. You don't have to write special
> code when you decide to port from your Suns to the PC or Mac.
> On the downside, if you are transferring large amounts of data
> it can be slower than you would like.

Ah.. si: how is the rate?

> If you are really serious about doing this, I would look into
> Call_External as a possibility. Or, perhaps even better, wait
> for IDL 5 and the new database connection tools. This problem
> may have already been solved for you! :-)

ah yes! ODBC is the best of course!

so, CU

Panther

--
Panther in the Jungle
-BELIEVE AND DECEIVE-
http://www.ang-physik
.uni-kiel.de/~hendrik

Subject: Re: spawn
Posted by Craig Markwardt on Wed, 16 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

marc <m_schellens@hotmail.com> writes:

> I want to spawn under UNIX one program many times (multiprocessor
> machine). So I could sent it into the background using the UNIT keyword,
> but how can I determine when the program finishes?
> I run some basic tests and it seems that IDL leaves the unit open.
> Is there a way or have I (ab)use the filesystem for this (via
> lockfiles)?

A clarification is in order. SPAWN with the UNIT keyword does *not*
run the process in the background. The unit stays open because IDL
attaches the output of the process to the unit, and IDL is waiting for
you to read the output.

You can run the process in the background by adding '&' like you
normally would. Then be sure that you redirect the output and CLOSE
the unit.

To find out whether a background process is finished is difficult.
Usually it's just easiest to write a flag to a temporary file when the
job is done, and have IDL periodically monitor that file for changes.
One can probably do it with the unix system call waitpid(), but you
that would require compiling a DLM, not for the faint of heart.

Good luck,
Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: spawn
Posted by marc schellens[1] on Thu, 17 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Another thing would be interesting to know:
How to catch the stderr output of a process?
(result paramter only returns stdout output)

Subject: Re: spawn
Posted by marc schellens[1] on Thu, 17 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> marc <m_schellens@hotmail.com> writes:
>
>> I want to spawn under UNIX one program many times (multiprocessor
>> machine). So I could sent it into the background using the UNIT keyword,
>> but how can I determine when the program finishes?
>> I run some basic tests and it seems that IDL leaves the unit open.
>> Is there a way or have I (ab)use the filesystem for this (via
>> lockfiles)?
>
> A clarification is in order. SPAWN with the UNIT keyword does *not*
> run the process in the background. The unit stays open because IDL
> attaches the output of the process to the unit, and IDL is waiting for
> you to read the output.
>
> You can run the process in the background by adding '&' like you
> normally would. Then be sure that you redirect the output and CLOSE

> the unit.
>
> To find out whether a background process is finished is difficult.
> Usually it's just easiest to write a flag to a temporary file when the
> job is done, and have IDL periodically monitor that file for changes.
> One can probably do it with the unix system call waitpid(), but you
> that would require compiling a DLM, not for the faint of heart.
>
> Good luck,
> Craig

The & at the end does only work in shell mode. Wiht SPAWN,/NOSHELL,...
you get an error message because the process treats it as an option. but
with SPAWN,/NOSHELL,UNIT=u,... IDL returns immediatly back to the
command line (also without /NOSHELL), or continues execution (at least
under Solaris and Linux, IDL 5.2).

But even then fstat(u) doesn't tell you anything about the process
status. Al least you can kill the process with FREE_LUN,u .

Right now I spawn every some seconds a 'ps' command and compare the
pid's (from PID parameter) to see if the process is already running.
But this 'ps' bothers, since the behaviour is little different between
Solaris and Linux.

However, thanks,
:-) marc

Subject: Re: spawn
Posted by Benno Puetz on Mon, 06 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

marc wrote:

> Another thing would be interesting to know:
> How to catch the stderr output of a process?
> (result paramter only returns stdout output)

How about redirecting it to a file and reading it afterwards?

Even redirecting to stdout might work if it can be distinguished ...

--
Benno Puetz
Kernspintomographie
Max-Planck-Institut f. Psychiatrie Tel.: +49-89-30622-413
Kraepelinstr. 10 Fax : +49-89-30622-520

80804 Muenchen, Germany

Subject: Re: Spawn
Posted by Allan Whiteford on Thu, 31 Jul 2008 11:56:27 GMT
View Forum Message <> Reply to Message

Joost Aan de Brugh wrote:
> Hello all,
>
> I was wondering if you can navigate efficiently through directories
> with Spawn (Calling shell commands)
>
> For example:
> IDL> Spawn,"pwd"
> <Some directory>
> IDL> Spawn,"cd .."
> IDL> Spawn,"pwd"
> <Still same directory, because after the cd .., something is reset>
>
> I solved this problem by making an executable file which takes care of
> the directories.
> Spawn,condition?"OneFile":"OtherFile"
> And the executable file OneFile and OtherFile do the same but in a
> slightly different directory.
>
> I actually have no problem with this, but I was wondering if there is
> a nicer way to do this.
>
> Kind regards,
> Joost

Joost,

Use the IDL cd procedure:

```
IDL> spawn,'pwd'
/home/allan
IDL> cd,'..'
IDL> spawn,'pwd'
/home
```

Thanks,

Allan

Subject: Re: Spawn
Posted by Bob[3] on Thu, 31 Jul 2008 12:33:09 GMT
View Forum Message <> Reply to Message

On Jul 31, 7:44 am, Joost Aan de Brugh <joost...@gmail.com> wrote:
> Hello all,
>
> I was wondering if you can navigate efficiently through directories
> with Spawn (Calling shell commands)
>
> For example:
> IDL> Spawn,"pwd"
> <Some directory>
> IDL> Spawn,"cd .."
> IDL> Spawn,"pwd"
> <Still same directory, because after the cd .., something is reset>

I believe the problem is because each time SPAWN is executed a new
shell is opened, so that you will be in IDL's current working
directory by default.

Subject: Re: Spawn
Posted by Joost Aan de Brugh on Thu, 31 Jul 2008 12:37:31 GMT
View Forum Message <> Reply to Message

Hello Allan,

This works well. And with the Current=olddir in the cd (and then
cd,olddir at the end) it is entirely solved.

Thanks.

Joost

>
> Joost,
>
> Use the IDL cd procedure:
>
> IDL> spawn,'pwd'
> /home/allan
> IDL> cd,'..'
> IDL> spawn,'pwd'
> /home
>
> Thanks,
>

> Allan

Subject: Re: Spawn
Posted by mankoff on Thu, 31 Jul 2008 12:38:03 GMT
View Forum Message <> Reply to Message

On Jul 31, 7:44 am, Joost Aan de Brugh <joost...@gmail.com> wrote:
> Hello all,
>
> I was wondering if you can navigate efficiently through directories
> with Spawn (Calling shell commands)
>
> For example:
> IDL> Spawn,"pwd"
> <Some directory>
> IDL> Spawn,"cd .."
> IDL> Spawn,"pwd"
> <Still same directory, because after the cd .., something is reset>
>
> I solved this problem by making an executable file which takes care of
> the directories.
> Spawn,condition?"OneFile":"OtherFile"
> And the executable file OneFile and OtherFile do the same but in a
> slightly different directory.
>
> I actually have no problem with this, but I was wondering if there is
> a nicer way to do this.
>
> Kind regards,
> Joost

You can use the internal 'cd' command in IDL:

IDL> spawn, 'foo'
IDL> cd, '..'
IDL> spawn, 'foo' ; different foo, in parent directory.

Subject: Re: Spawn
Posted by Michael Galloy on Thu, 31 Jul 2008 17:21:45 GMT
View Forum Message <> Reply to Message

On Jul 31, 6:37 am, Joost Aan de Brugh <joost...@gmail.com> wrote:
> Hello Allan,
>
> This works well. And with the Current=olddir in the cd (and then

> cd,olddir at the end) it is entirely solved.

Yes, the most used routine in my library is:

```
pro pwd
  cd, current=c
  print, c
end
```

Mike
--
www.michaelgalloy.com
Tech-X Corporation
Software Developer II
