
Subject: Re: handling of different(many) processes
Posted by [davidf](#) on Wed, 16 Dec 1998 08:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

R.Bauer (R.Bauer@fz-juelich.de) writes:

- > I like to have a mechanism which allows me to put some routines into a
- > idl background process while an other process is running.
- > I believe this is possible with widgets.
- > But what did I have to do with normal routines like:
- >
- > reading 30 MB into a structure
- > calculating something
- >
- > while saving to a sav file returning to the user.
- >
- > or returning to the user while saving to a sav file.
- >
- > At the moment I have to save to a sav file and after it's finished I am
- > able to return.
- >
- > Any ideas are welcome!

Ain't a gonna happen! Sorry. :-(

Cheers,

David

P.S. Maybe my favorite line from the movie "The Princess Bride" is
a consolation: "Get used to disappointment."

--

David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: handling of different(many) processes
Posted by [korpela](#) on Thu, 17 Dec 1998 08:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

In article <MPG.10e1bed32ce7c9569896af@news.frii.com>,
David Fanning <davidf@dfanning.com> wrote:
> R.Bauer (R.Bauer@fz-juelich.de) writes:
>

```
>> I like to have a mechanism which allows me to put some routines into a
>> idl background process while an other process is running.
>
> Ain't a gonna happen! Sorry. :-(
```

Not so fast.... We just had a thread here a couple weeks ago discussing this very thing. It certainly is possible to do all of these things under UNIX. It takes some use of call external and/or linkimage. At the end of this message, I'll provide the necessary routines for some multiprocessing support. They will work on many, if not most, UNIX systems. (I'm using them under SunOS). I wouldn't know how to do this on a Windows system. (And frankly I don't want to)

```
>> while saving to a sav file returning to the user.
```

This is easy using the routines below...

PROC_FORK is a function that creates a process. PROC_FORK returns 0 to the new process and returns the new process identifier to the old process. An if statement is used to determine which process you are in.

The following line will execute SAVE,/ALL in the background.

```
IDL> if PROC_FORK() eq 0 then begin & SAVE,/ALL & PROC_EXIT & endif
```

The PROC_EXIT line is necessary to tell the child process to exit after the save is complete, otherwise it will try to get command line input.

```
>> reading 30 MB into a structure
>> calculating something
```

These are more difficult as changes to memory aren't automatically shared between the processes. (I.E. if the background process changes a variable, the change won't show up in the foreground.) To do calculations in the background you need shared memory, which can be supplied by my VARRAY package. Unfortunately, VARRAY doesn't yet support structures.

Here's an example of a procedure that processes things in the child process and the parent process simultaneously.

```
function test
; create a 1024x1024 shared memory float array
a=VARRAY(float(0),1024,1024,/writable)
; process the [*;0:511] elements in the background
; process the [*;512:1023] elements in the foreground
if PROC_FORK() eq 0 then begin
; Child process
```

```

a[*,0:511]=randomn(seed,1024,512)
PROC_EXIT
endif else begin
; Parent process
a[*,512:1023]=randomu(seed,1024,512)
PROC_WAIT
endelse
return,a
end

```

The call to PROC_WAIT is necessary to make sure that the returned array doesn't get used until the processing is complete. PROC_WAIT stops a process until all of its child processes have finished. That can make it difficult to use more than two processes at a time.

It is possible to write a PROC_WAIT that waits for a specific child process to finish, but that's more complicated than what I've provided here.

Anyway, here are the functions that I wrote. I may eventually add a package of multitasking functions for IDL to my web site.

Eric

```

function proc_fork
; PROC_FORK.PRO Copyright 1998, Eric Korpela, All rights reserved
; Forks a new IDL child process, returns its PID to the old process
; Returns 0 the the new process
common libc,name
if not(keyword_set(name)) then name=findfile("/lib/libc.so.*.*")
pid=call_external(name[0],"_fork")
return,pid
end

```

```

pro proc_wait
; PROC_WAIT.PRO Copyright 1998, Eric Korpela, All rights reserved
; Waits for all child processes to finish
common libc,name
if not(keyword_set(name)) then name=findfile("/lib/libc.so.*.*")
dummy=call_external(name[0],"_wait")
end

```

```

pro proc_kill,pid
; PROC_KILL.PRO Copyright 1998, Eric Korpela, All rights reserved
; kills a process.

```

```
spawn,"/bin/kill "+string(pid)
end
```

```
pro proc_exit
; PROC_EXIT.PRO Copyright 1998, Eric Korpela, All rights reserved
; exits the current process
common libc,name
if not(keyword_set(name)) then name=findfile("/lib/libc.so.*.*")
pid=call_external(name[0],"_getpid")
proc_kill,pid
end
```

--

Eric Korpela | An object at rest can never be
korpela@ssl.berkeley.edu | stopped.
Click for home page.
