Subject: Stop a hanging execution
Posted by joe.llama on Tue, 26 Apr 2016 16:39:24 GMT
View Forum Message <> Reply to Message

Hi all,

I'm running a large chunk of code that calls multiple procedures and functions (the code was written by someone else). Every now and then, for a given set of input parameters, this code will hang when executing a certain procedure.

My question is, is there a way that if the code takes too long on one line of code to just return and stop executing? I've been investigating tic, and toc but I can't figure out how to say if toc is much longer than tic then stop.

Thanks!

Joe

Subject: Re: Stop a hanging execution
Posted by Craig Markwardt on Wed, 27 Apr 2016 13:24:19 GMT
View Forum Message <> Reply to Message

On Tuesday, April 26, 2016 at 12:39:26 PM UTC-4, joe....@lowell.edu wrote:
...

> My question is, is there a way that if the code takes too long on one line of code to just return and stop executing? I've been investigating tic, and toc but I can't figure out how to say if toc is much longer than tic then stop.

At the level of detail you gave, the short answer is no, there is no way to do this. You have a couple of choices.

One is to delve into the code and find a natural interruption point. There you can use GET_KBRD() to find out if a key has been pressed. Or even better, to quit if execution time exceeds some maximum.

Another choice is to use operating system functions. Mac and Linux OS's have a concept of process limits, which can include CPU execution time. The process will quit after the maximum is exceeded. Under bash, this would be achieved with "ulimit -t 600" for 600 seconds; the same thing under csh/tcsh is "limit cputime 600". You type this command at the console or in a script before running IDL, then run your IDL session.

Best wishes,
Craig Markwardt

Subject: Re: Stop a hanging execution

Posted by zacharyanorman on Thu, 28 Apr 2016 03:34:36 GMT

View Forum Message <> Reply to Message

Your best bet would probably be to use the IDl Workbench and actively press the stop button to see where you get stopped at. Alternatively, and something I use a lot, is placing print statements throughout your code. This really helps me know what processing is going on and gives me an idea of how much processing time I have left.

If you want to track time, then there are a couple of ways you can do this. You can use tic and toc like this, which actually returns the time that has passed in a variable so you can include a control statement in a loop to exit after a certain amount of time:

```
tic
wait, 2
time = toc()
print, time
```

Alternatively, you can use systime like this to get the time that has passed in seconds:

```
t_start = systime(/seconds)
wait, 2
time = systime(/seconds)
print, time - t_start
```