Data Analysis Applications Group (DAAG) Update: Running IDL Batch Jobs

Justin Burruss
2004.06.18

You can schedule your IDL jobs to run overnight

- Since our computers are often busy during the day, it is sometimes useful to schedule jobs to run at night
- Very useful if you need to crunch numbers
- It is possible to schedule IDL jobs to run at a later time
- The keys are
  - use I/O redirection
  - use the at program

Put the IDL commands in an input file

- Here are the IDL commands we want to run

```
print, 'starting at ' + STRTRIM( SYSTIME() )
gadat, x, y, 'IP', 111203, /nomds
ys = ga_smooth( y, x, 50, /boxcar )
print, 'finished at '+ STRTRIM( SYSTIME() )
exit
```

- Put them in a file called idl_inputs.txt

- Then use I/O redirection so that IDL runs those commands

```
hydra % idl < idl_inputs.txt
```

Use I/O redirection to record IDL output in a log file

- We're running this overnight, so we want to keep a record of what happened in case something goes wrong
- The solution: log files
- Use I/O redirection to send output to a log file

```
hydra % idl < idl_inputs.txt >& idl_outputs.txt
```

- In csh and tcsh the >& redirection operator works on stdout and stderr.

Make a wrapper script for running your IDL command

- We'll use a Bourne shell script in this example
- The name of our script is idl_batch.sh

```
#! /bin/sh -f
input=/u/burruss/idl/batch_example/idl_inputs.txt
output=/u/burruss/idl/batch_example/idl_outputs.txt
idl < $input > $output 2>&1
```

- The **`#! /bin/sh -f`** stuff is how you start a Bourne shell script
- In Bourne shell, **`2>&1`** means "send stderr to stdout"
- We want to put errors in our logfile, so remember to redirect stderr, too.

Test the wrapper script

- At this point it's a good idea to test again
- To make the script executable, do this:

```
hydra % chmod a+x idl_batch.sh
```

- Try running it directly to see if it works

```
hydra % idl_batch.sh
```

- The timestamp on the logfile will change, and the contents will be updated
- If you have trouble, change the first line of the script to **`#! /bin/sh -fx`**
  - The extra **`x`** turns on shell script debugging

Finally, use at to schedule the script to run later

- Use the at program to schedule the script to run at a certain time in the future
- When testing, use time now to run immediately:

```
at –f idl_batch.sh now
```

- In general, the date format used by at is MMDDhhmm
- So to schedule something to run April 30th @ 8:30pm we would do this:

```
at –f idl_batch.sh –t 04302030
```

- If you want to run this batch job on a regular basis, use cron