



University of
Zurich ^{UZH}

S3IT

Useful debugging commands

Riccardo Murri <riccardo.murri@uzh.ch>

S3IT: Services and Support for Science IT

University of Zurich

Recap of session-based scripts

Session-based scripts

All the scripts we've seen so far are *session-based scripts*.

A *session* is just a named collection of jobs.

A *session-based script* creates a session and runs all the tasks in it until completion.

Create a session

A session-based script **creates a session** and runs all the tasks in it until completion.

Create session S:

```
$ ./warholize.py bfly.jpg --session S
```

Run a session until done

A session-based script creates a session and **runs all the jobs in it until completion.**

Run jobs in session S, polling for updates every 5 seconds:

```
$ ./warholize.py bfly.jpg --session S --watch 5
```

You can stop a GC3Pie script by pressing *Ctrl+C*.
Run it again to resume activity from where it stopped.

Run a session until done

A session-based script creates a session and runs all the jobs in it until completion.

Run jobs in session S, polling for updates every 5 seconds:

```
$ ./warholize.py bfly.jpg --session S --watch 5
```

You can stop a GC3Pie script by pressing *Ctrl+C*.
Run it again to resume activity from where it stopped.

Inspecting sessions

Alternate display of session contents, I

Display top-level tasks in session S:

```
$ gsession list S
```

Exercise 3.A: Now try this yourself.

WTF??

```
> gsession list S
gc3.gc3libs: WARNING: Failed loading file '/home/ubuntu/S/jobs/WarholizeWorkflow.108': Import
...
LoadError: Failed retrieving object from file '/home/ubuntu/S/jobs/WarholizeWorkflow.108': 
gc3.gc3libs: WARNING: Ignoring error from loading 'ParallelTaskCollection.107': Failed retr
+-----+-----+-----+-----+
| JobID | Job name | State | Info |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

In order to work, all GC3Pie utilities need to access the Python script that generated the tasks and session.

To fix: set the PYTHONPATH variable to the directory containing your script:

```
$ export PYTHONPATH=$PWD
```

WTF??

```
> gsession list S
gc3.gc3libs: WARNING: Failed loading file '/home/ubuntu/S/jobs/WarholizeWorkflow.108': Import
...
LoadError: Failed retrieving object from file '/home/ubuntu/S/jobs/WarholizeWorkflow.108': 
gc3.gc3libs: WARNING: Ignoring error from loading 'ParallelTaskCollection.107': Failed retr
+-----+-----+-----+-----+
| JobID | Job name | State | Info |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

In order to work, all GC3Pie utilities need to access the Python script that generated the tasks and session.

To fix: set the `PYTHONPATH` variable to the directory containing your script:

```
$ export PYTHONPATH=$PWD
```

Alternate display of session contents, II

Display *all* tasks in session S:

```
$ gsession list --recursive S
```

Alternate display of session contents, III

Display summary of tasks in session S:

```
$ gstat -b -s S
```

Show log of actions in session

The `gession log` command prints out the sequence of actions and state changes for all tasks in a session:

```
$ gsession log ex2b
Jul 09 22:13:53 GrayscaleApp.6: Submitting to 'localhost'
Jul 09 22:13:53 GrayscaleApp.6: SUBMITTED
Jul 09 22:13:53 GrayscaleApp.6: Submitted to 'localhost'
Jul 09 22:13:58 GrayscaleApp.6: TERMINATING
Jul 09 22:13:58 GrayscaleApp.6: Final output downloaded to 'grayscale.d'
Jul 09 22:13:58 GrayscaleApp.6: TERMINATED
```

Dump contents of a task

The `ginfo` command allows you to dump the contents of a specific task, or all tasks in a session:

```
$ ginfo -v -s ex2b
GrayscaleApp.6
arguments: convert, bfly.jpg, -colorspace, gray, gray-bfly.jpg
[...]
execution:
[...]
history:
  - Submitting to 'localhost' at Sun Jul 10 00:27:02 2016
  [...]
lrms_execdir: /home/gc3pie/docs/programmers/tutorials/workflows/gc3libs.SMGNxrr
lrms_jobid: 23183
resource_name: localhost
[...]
inputs:
  file:///home/gc3pie/docs/programmers/tutorials/workflows/bfly.jpg: bfly.jpg
[...]
output_dir: grayscale.d
outputs:
  gray-lbfly.jpg: file, , gray-bfly.jpg, None, None, , None, None
  stderr.txt: file, , stderr.txt, None, None, , None, None
  stdout.txt: file, , stdout.txt, None, None, , None, None
[...]
stderr: stderr.txt
stdin: None
stdout: stdout.txt
[...]
```

Dump contents of a task, II

Note that **without the -v** option, `ginfo` limits its output to the `.execution` attribute of a Task/Application object:

```
$ ginfo -s S
GrayscaleApp.6
[...]
history:
  - Submitting to 'localhost' at Sun Jul 10 00:27:02 2016
  [...]
lrms_execdir: /home/gc3pie/docs/programmers/tutorials/workflows/gc3libs.SMGNx
lrms_jobid: 23183
resource_name: localhost
[...]
```

Session management

Abort all tasks in a session

The `gsession abort` command kills all tasks in a session.

```
$ gsession abort S
```

It produces normally no output; use the `-v` option to see a log of actions taken.

Note: it is important that sessions are terminated!

Otherwise, GC3Pie will consider part of the resources as still allocated to a task. This is especially evident when running on a single computer: after launching a few tasks, GC3Pie will stop and refuse to run anything.

Manual cleanup of the localhost resource

In case of incomplete cleanup, you will still run into this error:

```
ERROR: Resource localhost already running maximum allowed number of jobs
```

As a last resort, you can inspect directory

```
$HOME/.gc3/shellcmd.d:
```

```
$ ls $HOME/.gc3/shellcmd.d  
15843
```

Each of these files is an allocated execution slot in the localhost resource. Delete the files to free up the slot.

(The file name is the PID of the process, in case you want to check if a command is still running before you make GC3Pie forget about it.)

Aborting a single task

To stop and abort a single task, use the `gkill` command:

```
$ gkill -s S MyApplication.123
```

Combining commands

Selecting tasks from a session, I

The `gselect` command is the go-to tool for selective listing of tasks in a session. For example, to list finished tasks:

```
$ gselect -s S --state TERMINATED
```

The output of `gselect` is a list of task IDs, to be fed into another GC3Pie command. For example, to kill all queued tasks:

```
> gselect -s S --state SUBMITTED | xargs gkill -s S
```

Selecting tasks from a session, II

The `gselect` command has many different options to select tasks:

```
> gselect --help
usage: gselect [-h] [-V] [-v] [--config-files CONFIG_FILES] -s SESSION
               [--error-message REGEXP] [--input-file FILENAME]
               [--jobname REGEXP] [--jobid REGEXP] [-l STATES]
               [--output-file FILENAME] [--output-message REGEXP]
               [--successful] [--submitted-after DATE]
               [--submitted-before DATE] [--unsuccessful]
```

Exercise 3.B: Use `gselect` to print the IDs of the “TricolorizeImage” tasks in the last “Warholize” session.