GRID engine

The staff server has a version of the Oracle Grid Engine running. This is a so-called distributed resource management (DRM) system, which distributes user workload over the available processing resources. The system determines when and where a job is executed.

For more information, see the relevant manual page: `man sge_intro`

The shell command `qsub` allows you the send commands directly to the GRID batch system.

Below is an example, showing how to submit a simple compilation job:

```
bash$ qsub –S /bin/bash –m e –cwd <enter>
gcc –o hello hello.c <enter>
<ctrl-D>

Your job 587 ("STDIN") has been submitted
bash$
```

The options used in the example are:

`-S /bin/bash`	Use the bash shell as command interpreter
`-m e`	Send a message upon completing the job
`-cwd`	Use the current directory to read/write files. Omitting the `-cwd` switch implies `$HOME` as the working directory.

The source file `hello.c` used in the example should reside in the working directory.

The batch system will start the compilation process, and will send an e-mail message upon completion. The resulting compiled program `hello` will be written to the current working directory.

You can submit multiple jobs, which the batch system will handle in order of receipt. Upon completion of each job you will receive an e-mail message; the results can be found in the relevant working directory.

Use the command `qstat -f` to get a batch queue overview.

Redirecting output

You can redirect standard output (screen output) or standard error (error messages) to files of your choice.

Without specification, *default values* are used; these are:.

File location is your home directory (reading/writing).
Standard output is sent to file `job-name.o<job_id>`.
Standard error is sent to file `job-name.e<job_id>`.

Use the qsub `–N` option to supply a job name; this name will be used for output files.
Standard output will have extension `.o<job_id>`, and standard error will have `.e<job_id>`.

This is an example of a job submitted using qsub:

```
qsub –S /bin/bash –m e –M my_email@address –cwd –N counter my_script
```

```
bash$ Your job 567 ("counter") has been submitted
```

The options used are:

`-S /bin/bash`	shell used as command interpreter
`-M my_email@address`	e-mail address used to send end-of-job message to
`-N counter`	the job name, used for output; this will generate the files `counter.e<job_id>` en `counter.o<job_id>`, used for error messages and program output respectively.
`my_script`	the script to be executed; any executable script will do, but a counter is used as an example:

Contents of `my_script`

```
#!/bin/bash
for i in `seq 1 10`
 do
 echo "$i"
done
```

GRID environment variables

During job processing, the following variables are defined and available:

`HOME`	Home directory on the machine where the job is executed
`USER`	User ID of job owner
`JOB_ID`	Current job ID
`JOB_NAME`	Current job name
`HOSTNAME`	Name of the machine on which the job is running
`TASK_ID`	Array job task index number

Active Comments

Script lines starting with # will be regarded as comments. The Grid Engine system recognizes so-called *special comment lines*, used for passing options; these must start with #$.

Submitting multiple jobs

The example program `submit` (pag. 4.) reads a data file `data` and uses this as input for the `demo` script being submitted to the batch system with `qsub`.

The `submit` script actually just passes on a single variable. The `demo` script sends the variable value to standard output, which in the case of this example means the batch system will write the value to a file.

To start the job, simply enter the command `./submit`
The result will be nine jobs in the system queue, and after a while nine e-mail messages in your mailbox. In your current working directory, you will find the files `my_script_output` and `my_script_errors`. Using the `-N`, `-o` and `-e` switches, you can construct the relevant file names.

The example shows how to use active comment lines (mentioned above).
It is not always necessary to use all active comment lines as shown in the example.
Before calling `submit` you may want to change the *date-time to run* and the *email address* lines in the `demo` script. Remove the comment sign (#) at the beginning of a line to activate that line.
Without the *date-time to run* line the batch job will be executed immediately, otherwise it will be run at the given time (depending on othe jobs in the queue that have been submitted prior to your job).
Without the `-m e` line the GRID system will not send any messages regarding starting, ending or suspending the job; you will have to check manually for your job's status.

Contents of the file submit

```
#!/bin/bash
  for i in `cat data`
   do
    qsub demo $i
  done
#
qstat –f
```

Contents of the file data

```
1 2 3
4 5 6
7 8 9
```

Contents of the file demo

```
#$ -S /bin/bash

# date-time to run, format [[CC]yy]MMDDhhmm[.SS]
# Start job at Jul 5, 2011 at 10.23, change to any future time
# #$ -a 201107051023.00       # Remove ‘# ‘ to activate this line

# Name of job
#$ -N my_script

# Send e-mail when job starts/ends/suspends (#$ -m bes)
# Send e-mail when job ends (e)
#$ -m e

# Send info to e-mail address, change to valid address
# #$ -M my_emailaddress@uu.nl  # Remove ‘# ‘ to activate line

# Input/output files to be found in current working directory
# ($HOME directory is default)
#$ -cwd

# define output file name
#$ -o $JOB_NAME_output

# define error file name
#$ -e $JOB_NAME_errors

# Run the actual program
echo "Data contents $1"
```

Copy the file contents above to your working directory.
Make sure the scripts are executable before calling them!
Use the chmod command to do this:

```
chmod +x demo
chmod +x submit
```

Submit jobs to the batch system

```
bash$ ./submit
```

You will see a list of submitted files, followed by an overvie of queued jobs.

```
Your job 664 ("my_script") has been submitted
Your job 665 ("my_script") has been submitted
Your job 666 ("my_script") has been submitted
Your job 667 ("my_script") has been submitted
Your job 668 ("my_script") has been submitted
Your job 669 ("my_script") has been submitted
Your job 670 ("my_script") has been submitted
Your job 671 ("my_script") has been submitted
Your job 672 ("my_script") has been submitted
queuename                      qtype resv/used/tot. load_avg arch
states
---------------------------------------------------------------------------
long@staff.science.uu.nl       BP    0/1/6          1.16     lx24-amd64

############################################################################
 - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS
############################################################################
    664 0.00000 my_script solis-id     qw    07/19/2011 08:58:08     1
    665 0.00000 my_script solis-id     qw    07/19/2011 08:58:08     1
    666 0.00000 my_script solis-id     qw    07/19/2011 08:58:08     1
    667 0.00000 my_script solis-id     qw    07/19/2011 08:58:09     1
    668 0.00000 my_script solis-id     qw    07/19/2011 08:58:09     1
    669 0.00000 my_script solis-id     qw    07/19/2011 08:58:09     1
    670 0.00000 my_script solis-id     qw    07/19/2011 08:58:09     1
    671 0.00000 my_script solis-id     qw    07/19/2011 08:58:09     1
    672 0.00000 my_script solis-id     qw    07/19/2011 08:58:09     1
```

Any errors occuring during script execution will be written to `my_script_errors`.

Any results of the demo script will be written to `my_script_output`

```
bash$ cat my_script_output
Data contents 1
Data contents 2
Data contents 3
Data contents 4
Data contents 5
Data contents 6
Data contents 7
Data contents 9
Data contents 8
```

As you can see, output does not necessarily end up in the expected sequence in the output file. The output sequence depends on available processing resources.

Example of an end-of-job e-mail

```
Job 670 (my_script) Complete
 User             = solis-id
 Queue            = long@staff.science.uu.nl
 Host             = staff.science.uu.nl
 Start Time       = 07/19/2011 08:58:34
 End Time         = 07/19/2011 08:58:34
 User Time        = 00:00:00
 System Time      = 00:00:00
 Wallclock Time   = 00:00:00
 CPU              = 00:00:00
 Max vmem         = NA
 Exit Status      = 0
```

Tip

With long-running jobs, it is recommended to write intermediate results to the output file. The server may temporarily be unavailable due to maintenance or other reasons, and your work may be lost.