Orion - Basic PBS (qsub) Scripts

Basic PBS Header Script

```
#!/bin/sh
#############################################################################
#
#    Example 'qsub' job submission script for Torque
#    More information is available in the qsub(1) manual page.
#
#############################################################################

# Use the Bash Shell
#PBS -S /bin/bash

# Put a unique job name here
# so that you can monitor it in the queue
#PBS -N NameOfJob

# Define the queue you're submitting to
#PBS -q production.q

# Notifying user by email at the beginning,end,abort,suspensions of the job run
#PBS -M USERNAME@clarkson.edu
#PBS -m eab

# Uncomment to pass all current environment variables
#PBS -V
# Uncomment to pass a single environment variable
# #PBS -v VAR_TO_PASS

# Redirecting standard output / error to files
# named "output" and "errors"
#PBS -o output
#PBS -e errors

# The max walltime for this job is 1 hour 31 minutes
# Set this longer than required for run!!!
# Set number of nodes/ppn to number of cores required: 8..1 is 8 nodes, 1 core per node
#PBS -l walltime=800:00:00,nodes=3:ppn=2

# Keep this.
#cd $PBS_O_WORKDIR
```

Basic Fluent.bash Script

```
#$ -S /bin/bash

# Put a unique job name here
# so that you can monitor it in the queue
#$ -N my_job_name

# Notifying user by email at the beginning,end,abort,suspensions of the job run
#$ -M youremailhere@clarkson.edu
#$ -m beas

# Tell GE to run the job from the current working directory
#$ -cwd

# Uncomment to pass all current environment variables
#$ -V
# Uncomment to pass a single environment variable
# #$ -v VAR_TO_PASS

# Redirecting standard output / error to files
# named "output" and "errors"
#$ -o output
#$ -e errors

# The max walltime for this job is 48 hours 00 minutes
# Set this longer than required for run!!!
#$ -l h_rt=48:00:00

# Parallel execution request for fluent. Set your number of processors here.
#$ -pe fluent_pe 2

# PUT SOME BASIC INFO INTO OUTPUT FILE
local_dir=$(pwd | cut -c 23-)
echo "This job was started in ${local_dir}"
echo "The job id is $JOB_ID"
echo "The start time was $(date)"

# Executing the program
# Fluent must be run as a batch job, not interactively
# THIS WEB PAGE HAS GOOD INSTRUCTIONS FOR RUNNING FLUENT
# http://www.hpcvl.org/faqs/application-software/fluent/how-do-i-setup-and-execute-a-fluent-parallel-production-
job

# Here is an easy way to make input.file
# start fluent at the command line with the
# command /share/apps/ansys_inc/v140/fluent/bin/fluent 3ddp -g
# issue command /file/start-journal
# use text menus to read and set up and case inputs, do 1 iteration, write data
# issue command /file/stop-journal
# exit
# then open journal file and modify it to do more iterations


# Here is an example input.file
#rc fan.cas
#/solve/initialize/initialize-flow
#/solve/iterate 1
#/file/write-data fan_1
#exit
#yes

/share/apps/ansys_inc/v140/fluent/bin/fluent -r14.0.0 2ddp -sgepe fluent_pe $NSLOTS -pinfiniband -cnf=$TMPDIR
/machines -ssh -g -i Test.jou -mpi=pcmpi

## Make a file for plotting convergence history
## Can be loaded into matlab/excel/tecplot to see convergence history
# Uncomment the following line to make convergence history file
# grep "^ *[0-9][ 0-9e+:.-]*$" output | sed "s/:/ /g" > cnvg.dat

echo "The stop time was $(date)"


# To submit this job type "qsub <the_name_of_this_file>"
# You can also use "qmon" (gui for queue) to submit, delete, monitor jobs...
# Other useful command line things: "qstat, qdel"
# To learn about queuing system type "man qsub"

# if a fluent job aborts, then you should use the kill script to delete the leftover processes
# chmod +x <kill-fluent-file>
# ./<kill-fluent-file>
```

Basic mpich.bash Script

Use the Bash Shell

```
#$ -S /bin/bash

# Put a unique job name here
# so that you can monitor it in the queue
#$ -N my_job_name

# Notifying user by email at the beginning,end,abort,suspensions of the job run
#$ -M youremailhere@clarkson.edu
#$ -m eas

# Tell GE to run the job from the current working directory
#$ -cwd

# Uncomment to pass all current environment variables
#$ -V
# Uncomment to pass a single environment variable
# #$ -v VAR

# Redirecting standard output / error
#$ -o qoutput
#$ -e qerrors

# The max walltime for this job is 31 minutes
#$ -l h_rt=600:31:00

# Parallel execution request for MPICH. Set your number of processors here.
#$ -pe mpich 2

local_dir=$(pwd)
echo "This job was started in ${local_dir}"
echo "The start time was $(date)"
echo "The job id is $JOB_ID"
echo "Got $NSLOTS processors."
echo "Machines:"
cat $TMPDIR/machines
echo ${PE}

# Executing the program
mpiexec -np $NSLOTS -machinefile $TMPDIR/machines <your_executable_name>
```

Basic serial.bash Script

```
# Use the Bash Shell
#$ -S /bin/bash

# Put a unique job name here
# so that you can monitor it in the queue
#$ -N my_job_name

# Notifying user by email at the beginning,end,abort,suspensions of the job run
#$ -M youremailhere@clarkson.edu
#$ -m eas

# Tell GE to run the job from the current working directory
#$ -cwd

# Uncomment to pass all current environment variables
#$ -V
# Uncomment to pass a single environment variable
# #$ -v VAR_TO_PASS

# Redirecting standard output / error to files
# named "output" and "errors"
#$ -o output
#$ -e errors

# The max walltime for this job is 1 hour 31 minutes
# Set this longer than required for run!!!
#$ -l h_rt=240:00:00

local_dir=$(pwd | cut -c 23-)
echo "This job was started in ${local_dir}"
echo "The start time was $(date)"
echo "The job id is $JOB_ID"
echo ${NSLOTS}
echo ${PE}

# Executing the program
<your_executable_name>
```