#!/bin/sh

bsub -o out.txt -e error.txt "cat pymol.pml | pymol -c -p"

#####################################################################

# Cameron Mura

# March 2002

#

# PyMOL script to make simple ribbon diagram of the Pae SurE dimer

# One monomer will be colored red, the other blue.

#####################################################################

# load pdb file and call the PyMOL object "sure":

load 4A8J.pdb

# set background color to white:

bg\_color black

rotate y, 90

# Hide all the atoms, then make the cartoon and display just the ribbons:

hide everything

cartoon auto

show cartoon

show ribbon

show sticks

set cartoon\_fancy\_helices=1

set cartoon\_flat\_sheets = 1

set cartoon\_smooth\_loops = 1

clip slab, 60

set antialias = 1

set gamma = 1.15

set orthoscopic = 1

# Do ray-tracing, write out .png files, and quit PyMOL:

ray 1600, 1200

png 4A8J.png, 1600, 1200

quit

1.

most basic script

top: Shellscript

bottom: PyMOL script

Topics: Shebang, how to run a script / required permissions (./script.sh, /bin/sh script.sh, script.sh), location of scripts (per-project or all together)

#!/bin/sh

# script xyz.sh - render PDB files with PyMOL

#

# Author: Frank Thommen, EMBL Heidelberg

# Date: 22-May-2013

# License: Take it or leave it

#

# Changes:

#

#

bsub -o out.txt -e error.txt "cat pymol.pml | pymol -c -p"

2.

Added comments

Topics: Comments, documentation

#!/bin/sh

# script xyz.sh - render PDB files with PyMOL

#

# Author: Frank Thommen, EMBL Heidelberg

# Date: 22-May-2013

# License: Take it or leave it

#

# Changes:

#

#

if [ ! -e 4A8J.pdb ]; then

echo "Sorry, file 4A8J.pdb not found"

exit

fi

bsub -o out.txt -e error.txt "cat pymol.pml | pymol -c -p"

3.

Added if statement to make sure the pdb file exists

Topics: if, conditions

#!/bin/sh

# script xyz.sh - render PDB files with PyMOL

#

# Author: Frank Thommen, EMBL Heidelberg

# Date: 22-May-2013

# License: Take it or leave it

#

# Changes:

#

#

FILE=4A8J.pdb

if [ ! -e $FILE ]; then

echo "Sorry, file $FILE not found"

exit

fi

bsub -o out.txt -e error.txt "cat pymol.pml | pymol -c -p"

4.

Added variable $FILE for the file, which immediately leads to 5, because we’d like to use the variable in the PyMOL script, too

Topics: Variables

#!/bin/sh

# script xyz.sh - render PDB files with PyMOL

#

# Author: Frank Thommen, EMBL Heidelberg

# Date: 22-May-2013

# License: Take it or leave it

#

# Changes:

#

#

ID=4A8J

if [ ! -e $ID.pdb ]; then

echo "Sorry, file $ID.pdb not found"

exit

fi

bsub -o out.txt -e error.txt "cat pymol-var.pml | sed "s/PDBID/$ID/" | pymol -c -p" > /dev/null

#####################################################################

# Cameron Mura

# March 2002

#

# PyMOL script to make simple ribbon diagram of the Pae SurE dimer

# One monomer will be colored red, the other blue.

#####################################################################

# load pdb file and call the PyMOL object "sure":

load PDBID.pdb

[...]

png PDBID.png, 1600, 1200

quit

5.

Now using $ID instead of $FILE, adding the extension in the script (top) and using the modified pymol.pml script (bottom)

#!/bin/sh

# script xyz.sh - render PDB files with PyMOL

#

# Author: Frank Thommen, EMBL Heidelberg

# Date: 22-May-2013

# License: Take it or leave it

#

# Changes:

#

#

ID=4A8J

if [ ! -e $ID.pdb ]; then

echo "Sorry, file $ID.pdb not found"

exit

fi

bsub -o out.txt -e error.txt "cat pymol-var.pml | sed "s/PDBID/$ID/" | pymol -c -p" > /dev/null

until test -e $ID.png; do

sleep 5

echo "ID $ID still being processed..."

done

echo "ID $ID finished processing!"

6.

Added while loop which checks for the end result of processing

Topics: while, until

#!/bin/sh

# script xyz.sh - render PDB files with PyMOL

#

# Author: Frank Thommen, EMBL Heidelberg

# Date: 22-May-2013

# License: Take it or leave it

#

# Changes:

#

#

while [ "$#" -gt 0 ]; do

if [ $1 = "--debug" ]; then

set -x

shift

elif [ $1 = "--help" ]; then

echo "USAGE: $0 [--debug] [--help] ID [ID [...]]"

exit

else

ID=$1

shift

fi

done

if [ ! -e $ID.pdb ]; then

echo "Sorry, file $ID.pdb not found"

exit

fi

bsub -o out.txt -e error.txt "cat pymol-var.pml | sed "s/PDBID/$ID/" | pymol -c -p" > /dev/null

until test -e $ID.png; do

sleep 5

echo "ID $ID still being processed..."

done

echo "ID $ID finished processing!"

7.

Added commandline argument processing with possibility to define the ID on the commandline

Topics: shift, set, while, conditions

#!/bin/sh

# script xyz.sh - render PDB files with PyMOL

#

# Author: Frank Thommen, EMBL Heidelberg

# Date: 22-May-2013

# License: Take it or leave it

#

# Changes:

#

#

while [ "$#" -gt 0 ]; do

case $1 in

--debug)

set -x

shift

;;

--help)

echo "USAGE: $0 [--debug] [--help] ID [ID [...]]"

exit

;;

\*)

ID=$1

shift

;;

esac

done

if [ ! -e $ID.pdb ]; then

echo "Sorry, file $ID.pdb not found"

exit

fi

bsub -o out.txt -e error.txt "cat pymol-var.pml | sed "s/PDBID/$ID/" | pymol -c -p" > /dev/null

until test -e $ID.png; do

sleep 5

echo "ID $ID still being processed..."

done

echo "ID $ID finished processing!"

8.

Replaced if-elif-elif in commandline argument processing by case for better readability

Topics: case

#!/bin/sh

# script xyz.sh - render PDB files with PyMOL

#

# [...]

# process commandline parameters and arguments

#

while [ "$#" -gt 0 ]; do

case $1 in

--debug)

set -x

shift

;;

--help)

echo "USAGE: $0 [--debug] [--help] ID [ID [...]]"

exit

;;

\*)

IDS=$\*

break

;;

esac

done

echo "Processing IDS $IDS"

# Submit rendering job for all given IDs

#

for ID in $IDS; do

if [ ! -e $ID.pdb ]; then

echo "Sorry, file $ID.pdb not found"

else

bsub -o out.$ID.txt -e error.$ID.txt "cat pymol-var.pml | sed "s/PDBID/$ID/" | pymol -c -p" > /dev/null

fi

done

# check for the existence of processed .png files

#

ALLDONE=0

until test $ALLDONE -eq 1; do

sleep 5

ALLDONE=1

for ID in $IDS; do

if [ -e $ID.png ]; then

echo "ID $ID finished processing!"

else

echo "ID $ID still being processed..."

ALLDONE=0

fi

done

done

echo "All done"

9.

Expanded commandline processing and processing loop to support multiple IDs. Have different logfiles per job.

Added comments per functional block

Topics: $\*, documentation, variables as flags, break/continue

#!/bin/sh

# script xyz.sh - render PDB files with PyMOL

#

# [...]

# process commandline parameters and arguments

#

while [ "$#" -gt 0 ]; do

case $1 in

--debug)

set -x

shift

;;

--help)

echo "USAGE: $0 [--debug] [--help] ID [ID [...]]"

exit

;;

\*)

IDS=$\*

shift $#

;;

esac

done

echo "Processing IDS $IDS"

EXITSTATUS=0

# Submit rendering job for all given IDs

#

for ID in $IDS; do

if [ ! -e $ID.pdb ]; then

echo "Sorry, file $ID.pdb not found"

else

bsub -o out.$ID.txt -e error.$ID.txt "cat pymol-var.pml | sed "s/PDBID/$ID/" | pymol -c -p" > /dev/null

[ $? -ne 0 ] && EXITSTATUS =1

fi

done

# check for the existence of processed .png files

#

ALLDONE=0

until test $ALLDONE -eq 1; do

sleep 5

ALLDONE=1

for ID in $IDS; do

if [ -e $ID.png ]; then

echo "ID $ID finished processing!"

else

echo "ID $ID still being processed..."

ALLDONE=0

fi

done

done

if [ $EXITSTATUS -ne 0 ]; then

echo "An error occured..."

else

echo "All done successfully"

fi

exit $EXITSTATUS

10.

ensuring a correct and controlled exit status

Topics: exitstatus, $?, compact formulation of conditions, command grouping, alternative way of stopping the commandline argument loop (shift $#)

Known problems and possible enhancements:

* allow intermixed arguments and commandline parameters
* handle the situation, where a .png file already exists (e.g. by skipping this rendering)
* handle the situation when no ID is given
* Make the status lines more informative by using `bjobs` output (requires fetching the job ID from the `bsub` output
* retrieve IDs from existing \*.pdb files where no \*.png exists