Regression Initiation: `rtest_init`

Victor M. Castillo
Methods Development Group

December 16, 1999

1 Scope

This `Python` script generates the `.info` file used by `rtest` to determine what element types are available within each sample case. A `grizinit` file is created that instructs GRIZ to dump the results of the `info` command to a file named `.info`.

1 ⟨* 1⟩≡

```
#!/usr/bin/env /grdev/regrtest/bin/python
```

⟨*init* 2⟩

2 ⟨*init* 2⟩≡ (1)

```
⟨rtest support 4⟩
#import os,re,string
from rtest_support import *

"""This routine does some basic initialization for the
regression testing.

A griz binary is called to generate an ascii file with
*/.info* about the sample file. The file is then parsed
to determine the relevant elements.

***This only needs to be run once for each sample***
"""

generate=1

GRIZBIN=TEST['bin'][0]
DB='d3plot'
for sample in CASE['SAMP']+CASE['SND']:
   print("Analyzing data for "+sample+"...")
   os.chdir(RHOME+sample+"/")
   if generate:
      g=open('grizinit','w+')    #create grizinit file
      g.write(HEADER)
      g.write("savtxt .info\n")
      g.write("info\n")
      g.write("endtxt\n")
      g.write("savtxt done\n")
      g.write("endtxt\n")
      g.write("quit\n")
      g.close()
      err=os.system(GRIZBIN+" -i "+DB)   #run griz with new grizinit file
      if not os.path.exists('done'):
         print("waiting ...")
      while not os.path.exists('done'):
         pass
      os.remove('done')
      os.remove('grizinit')
   #Now parse .info file
   if not os.path.exists('.info'):
      print("ERROR: No info file found.")
   info=open('.info','r+')
   raw = info.read()
   info.close()
   #
```

```
for key in IS.keys():
   etest = re.compile(IS[key][0])
   if etest.search(raw):
      np=string.replace(IS[key][0],':','')
      print(">>> "+np+" detected in "+sample)
```

2 Appendix: rtest_support

This file defines all of the Python dictionaries used for the regression testing. Dictionaries have an unordered set of *key:value* pairs where the key can be a string.

4 ⟨*rtest support* 4⟩≡ (2)

```
import os,sys,re,string

verbose=1
RHOME="/grdev/regrtest/"
GRIZ2=RHOME+"bin/griz2"
GRIZ4=RHOME+"bin/griz4"
TAURUS="../d3plot"
MILI="../m_plot"
H1="# This grizinit file was automatically generated\n"
H2="# by regrtest.py for regression testing of Griz4\n"
H3="# -Vic Castillo\n#\n"
HEADER=H1+H2+H3

TEST={}
TEST['bin']=GRIZ2,GRIZ4,GRIZ4
TEST['db']=TAURUS,TAURUS,MILI
TEST['dir']='G2T','G4T','G4M'

CASE={}
CASE['SAMP']='SAMP1','SAMP2','SAMP4','SAMP6','SAMP8'
CASE['SND']='SND1','SND2','SND3'

# Derived Results Dictionary
DR={}
drnd1='dispx','dispy','dispz','dispmag'
drnd2='velx','vely','velz','velmag'
drnd3='accx','accy','accz','accmag','pvmag'
DR['Nodal']=drnd1+drnd2+drnd3
drshr1='sx','sy','sz','sxy','syz','szx'
drshr2='press','seff','pdev1','pdev2','pdev3'
drshr3='maxshr','prin1','prin2','prin3'
DR['Share']=drshr1+drshr2+drshr3
drshl1='surf1','surf2','surf3','surf4','surf5','surf6'
drshl2='eff1','eff2','effmax'
DR['Shell']=drshl1+drshl2
drbrk1='ex','ey','ez','exy','eyz','ezx'
drbrk2='pdstrn1','pdstrn2','pdstrn3','pshrstr'
drbrk3='pstrn1','pstrn2','pstrn3','relvol','evol'
```

```
DR['Brick']=drbrk1+drbrk2+drbrk3

# Primal Results Dictionary
PR={}
PR['Nodal']='nodpos[ux]',
PR['Global']='ke',
PR['Mat']='matpe',
PR['Brick']='eeff',
PR['Shell']='eeff_mid',

# Time History Dictionary
# These are the results selected for the time histories
TH={}
TH['Nodal']='dispmag',
TH['Global']='ke',
TH['Mat']='matpe',
TH['Brick']='prin1',
TH['Shell']='effmax',

# Info String Dictionary
# These are the string patterns from the griz 'info' command.
# (Why does it not work without the commas?)
IS={}
IS['Nodal']='Nodes:',
IS['Brick']='Hex elements:',
IS['Shell']='Shell elements:',
# IS['Beam']='Beam elements:',

# These are the strings describing result types
RS={}
RS['DR']='Derived Results'
RS['PR']='Primative Results'
RS['TH']='Time History Results'
```