Package produtil

Sam Trahan
January 2016

Package “produtil”

- Why?
  - Python modules have many bugs.
    - (especially RedHat Python 2.6.6)
  - Built-in modules lack logging and error checking.
  - Python core is unreliable at backward compatibility
  - Scientists are  at security and reliable code
- Solution:
  - Add a layer between HWRF and Python core that works around these problems.

produtil.run
Serial Programs

- `exe('echo')` = which echo
  - find program “echo” and return a produtil.prog.Runner that knows how to run the program
- `run(exe('echo'))` = echo
- `run(exe('echo')['hello','world'])` = echo hello world
- `checkrun(exe('echo')['hello','world')`

  = set -e ; echo hello world
- `runstr(exe('echo'))` = s=`echo`
- bigexe vs. exe vs. batchexe:
  - Use exe()
  - Others are for Cray support: bigexe and exe use aprun, batchexe does not.

produtil.run
MPI Ranks

- `mpi('wrf.exe')  =  which wrf.exe`
- `mpi('wrf.exe')*200`
  - I want to run 200 ranks of wrf.exe
- `mpi('cpl.exe')*9+mpi('ocn.exe')*90`

  `+ mpi('wave.exe')*120`

  `+ mpi('wrf.exe')*600`
  - 9 cpl.exe ranks, 90 ocn.exe ranks, 120 wave.exe ranks, 600 wrf.exe ranks.
- NONE OF THESE RUN A PROGRAM

produtil.run
Running an MPI Program

- `mpirun(mpi('cpl.exe')*9+mpi('ocn.exe')*90`
  `+mpi('wave.exe')*120+mpi('wrf.exe')*600)`
  - Returns:
  - `exe('mpirun')['-np','9','cpl.exe',':',`
    `'-np','90','wave.exe',':','-np',`
    `'600','wrf.exe']`
- produtil.mpi_impl converts from mpirun(...) to exe(...)

  `checkrun(mpirun(mpi('cpl.exe')*9+mpi('ocn.exe')*90`
  `+mpi('wave.exe')*120+mpi('wrf.exe')*600))`
- Runs that command!

produtil.run
Running an MPI Program

- `checkrun(mpirun(mpi('post.exe'),all_ranks=True))`
  - Run post.exe on as many MPI ranks as possible.
- `openmp(exe('relocate.exe'),threads=16)`
  - Run relocate.exe on 16 OpenMP threads.
- `runstr(mpirun(mpi('mympi.exe')*32) |`
  `exe('myserial')['alg=bubblesort'])`
  - Run mympi.exe on 32 MPI ranks, and
  - pipe that into myserial with argument alg=bubblesort

produtil.run Internals

- exe(), batchexe(), etc.
  - Create a Runner, which stores information about the program, or pipeline of programs, to run.
- mpi(), mpiserial()
  - Create a tree of MPIRanksBase objects describing an MPI program to run.
- mpirun()
  - Converts the MPIRanksBase tree to a Runner
- checkrun(), run(), runstr()
  - Runs a Runner.
- alias()
  - Converts a Runner to an ImmutableRunner. If you try to modify the result (add arguments, env vars, etc.) it will create a new Runner.
  - This avoids mistakenly re-using an old Runner

Data Delivery
produtil.datastore Product callbacks

- List of functions to call when a product is delivered.
- Allows one to add dbnalerts or other delivery actions without having to know internal workflow.
- Example:

```
prod=Product(...)
def my_callback(prod):
    print “%s: delivered to %s”%(
        prod.did, prod.location)
```

Data Delivery
produtil.dbnalert

- Pre-made delivery callbacks for DBN alerts.

```
from hwrf_expt import tracker
hr3=tracker.product('atcf3hourly')
hr3.add_callback(DBNAlert([
    'MODEL','HWRF_ASCII','{job}'
    '{location}']))
```

- In shell:

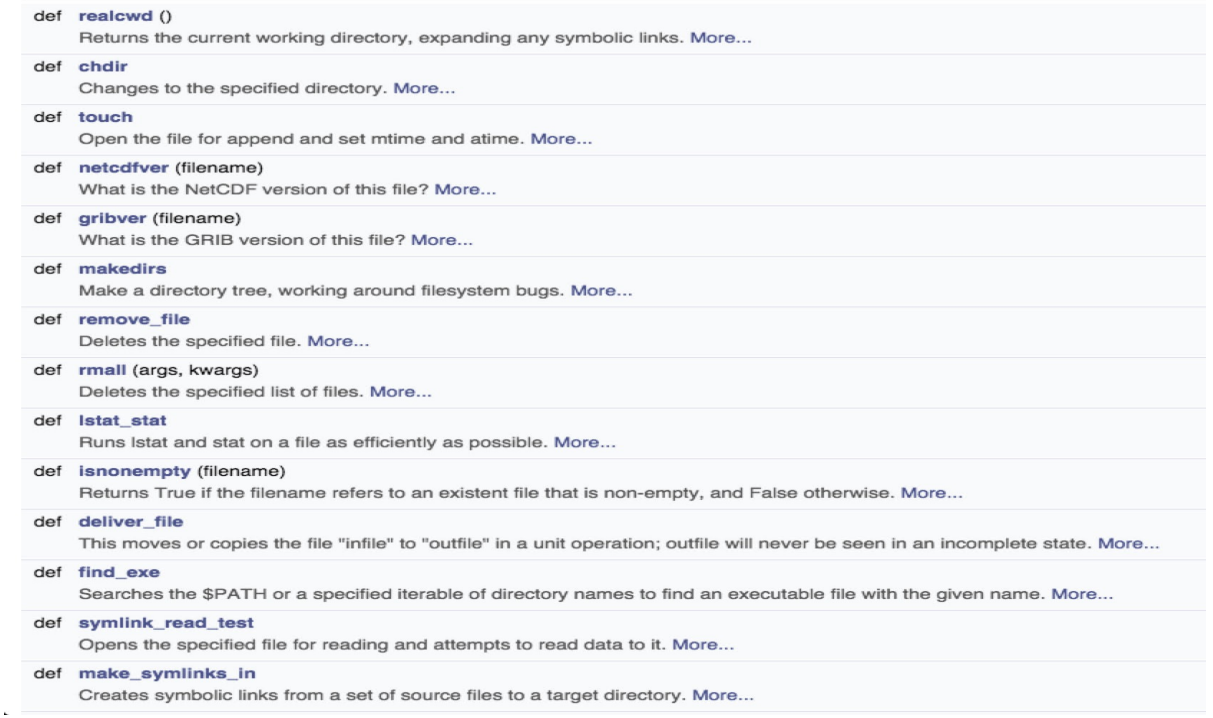
```
dbn_alert MODEL HWRF_ASCII \
    jhwrf_output /path/to/track.atcfunix
```

File Manipulation
produtil.fileop

- Many common shell commands:
  - ln -s = make_symlink
  - rm = remove_file
  - pwd = realcwd
  - cd = chdir, but don't use it. Use produtil.cd
  - touch = touch
  - mkdir -p = makedirs
  - rm -rf = rmall
  - lstat_stat = stat
  - isnonempty = test -s

File Manipulation
produtil.fileop

- Many common shell commands:
  - which = find_exe
  - ln -s * /path/to/dir/ = make_symlinks_in
- http://www.emc.ncep.noaa.gov/HWRF/scripts/namespaceprodutil.html



def **realcwd** ()
Returns the current working directory, expanding any symbolic links. More...

def **chdir**
Changes to the specified directory. More...

def **touch**
Open the file for append and set mtime and atime. More...

def **netcdfver** (filename)
What is the NetCDF version of this file? More...

def **gribver** (filename)
What is the GRIB version of this file? More...

def **makedirs**
Make a directory tree, working around filesystem bugs. More...

def **remove_file**
Deletes the specified file. More...

def **rmall** (args, kwargs)
Deletes the specified list of files. More...

def **lstat_stat**
Runs lstat and stat on a file as efficiently as possible. More...

def **isnonempty** (filename)
Returns True if the filename refers to an existent file that is non-empty, and False otherwise. More...

def **deliver_file**
This moves or copies the file "infile" to "outfile" in a unit operation; outfile will never be seen in an incomplete state. More...

def **find_exe**
Searches the $PATH or a specified iterable of directory names to find an executable file with the given name. More...

def **symlink_read_test**
Opens the specified file for reading and attempts to read data to it. More...

def **make_symlinks_in**
Creates symbolic links from a set of source files to a target directory. More...

File Manipulation
produtil.fileop.deliver_file

- Deliver a file without leaving a partial version:
  - deliver_file(“fromfile”,”tofile”,[options])
  - Copy “fromfile” to “/path/to/tmp.tofile.random”
  - move “/path/to/tmp.tofile.random” to “/path/to/tofile”
- Options:
  - keep=False = may use “mv” if possible
  - verify=True = verify tmp file before renaming to final
  - removefailed=True = delete temp file if failed
  - force=True = delete target if it already exists
  - preserve_*=True = retain various attributes
  - ... others ...

File Manipulation
produtil.fileop.fortcopy and fortlink

- Copy or link files to fort.[number] files for input to fortran programs.
  - fortlink({10:”infile”,30:”outfile”},[options])
    - ln -s “infile” fort.10
    - ln -s “outfile” fort.30
  - fortcopy({10:”infile1”,11:”infile2”},[options])
    - cp “infile1” fort.10
    - cp “infile2” fort.11

File Manipulation

produtil.fileop.check_file and wait_for_files

- check_file(filename,[options])
  - Check to see if file meets size, age, etc. requirements.
- wait_for_files([“file1”,”file2”,...],logger=logger, [options])
  - Wrapper around check_file that checks many times, up to some maximum wait time.

File Manipulation
produtil.listing.Listing

- `listing=produtil.listing.Listing(`
  `path,[options])`
- `print str(listing)` = ls -l [options] path
  - hidden=True = list hidden (.*) files.
  - logger=logger = a logger for logging messages
- The listing includes a mapping from name to file information:

```
for filename,info in listing:
    (lstat,linkpath)=info
    print “%s size is %d”%(
       filename,lstat.st_size)
```

Making and Changing Directory
produtil.cd

- TempDir class:

```
with TempDir() as t:
    ... do things in a temporary directory ...
... temp dir is gone ...
```

- NamedDir class:

```
with NamedDir('/path/to/dir') as nd:
    ... do things in /path/to/dir/ ...
... back to where we were ...
```

  - Does not delete /path/to/dir
- Many options.

Locking Files
produtil.lockfile

- Locks files. Includes workarounds for problematic filesystems.
  - `with LockFile('/path/to/file') as lf:`
  - `    ... do things with file locked ...`
  - `... file is now unlocked ...`

Resource Limits and Usage
produtil.rusage

- Set or get resource limits.
  - `print 'Stack limit is: ',getrlimit().stack`
  - `setrlimit(stack=6e9)`
  - `print 'Now limit is:',getrlimit().stack`
- Query resource usage.

```
with rusage(logger=logger) as ru:
    ... run programs ...
```

Logging
produtil.log

- Arranges NCEP standard log output syntax.
  - produtil.log.jlogger = sends output to jlogfile
  - postmsg(“message”) = send a message to jlogfile at INFO level.
  - set_jlogfile(“/path/to/jlogfile”) = sets the path to the jlogfile. Default comes from $jlogfile variable.
  - Other functions to handle multi-threaded jobs.

Cluster Interaction

produtil.cluster

- where() - guess which cluster we're on. Return an object that reports information about the cluster.
- longname() - full name of the cluster (ie.: jet.rdhpcs.noaa.gov)
- name() - short name (“jet”)
- ncepprod() - is this the NCEP production machine (True or False)
- Other information.

Cluster Interaction
produtil.batchsystem

- jobname - name of the batch job
- jobid - ID of the batch job
- joblongname - long name of the batch job

NOAA Restricted Data
produtil.rstprod

- tag_rstprod(“/path/to/file”,logger=logger)
  - Restricts the file or directory so that only those with approval to access NOAA restricted data can access it.
  - Knows about ACLs, rstprod group, etc.
  - Can be used for other data restriction classes (ie.: hur-decks)
- produtil.acl
  - Support library for produtil.rstprod.
  - Wrapper around libacl to handle filesystems without support for group-based restriction (ie.: Lustre)

Signal Safety
produtil.sigsafety

- Ensures:
  1. cleanup of locks, other resources, before exit.
  2. script exits quickly and cleanly
- After receiving a signal:
  - produtil.locking will not lock files
  - produtil.pipeline will kill processes
  - produtil.workpool will kill threads

Retrying Failed Operations
produtil.retry

- `produtil.retry.retry_io(`

  `max_tries,sleep_time,operation,[options])`

  - Try an operation many times until it succeeds or until we give up.
  - Exponential back-off.
  - Underlying implementation of other functionality

Threading
produtil.workpool

```
with produtil.workpool.WorkPool(5) as w:
    w.add_work(my_function,[arg,u,ments])
... all work is complete when “with” block exits
```

- Makes threading easier.
- Linked to produtil.sigsafety to automatically kill threads upon signal.
- Connected to logging capabilities.
- input, gsi and multistorm final merge jobs

Etcetera

- produtil.datastore - covered in previous presentation (HWRF Internals)
- produtil.atparse - text pre-parser
- produtil.run implementation:
  - mpi_impl - sub-package that implements MPI program execution
  - prog, mpiprog - represents programs as internal object structures, can manipulate.
  - produtil.pipeline - runs programs defined by objects within produtil.prog module.