METplus

Beta-METplus

Generated by Doxygen 1.8.13

Contents

Chapter 1

METplus Scripts

Welcome to the METplus scripting system documentation. This manual seeks to document every aspect of the METplus scripts, at both a high level and low level. Every function, argument, class, script, module, member variable and module-level variable is documented, and there are examples in many places of how to extend the scripts and Python libraries. There are also pages that give a high-level description of how the scripts work.

This website layout and content is based on and uses the original work that was developed for the HWRF project by Samuel Trahan from EMC-NCEP-NOAA.

Note

Content is currently being added to this manual. Pages may move, merge, be renamed, reformatted, or edited.

1.1 METplus Terms Of Use

- Model Evaluation Tools Plus (METplus) TERMS OF USE - IMPORTANT!

1.2 What is All of This?

METplus is a Python scripting structure (wrapper) around the the MET series of statistical tools.

You can find more information about MET and METplus on these websites:

What	Where
MET	http://www.dtcenter.org/met/users/
METplus*	https://www.github.com/NCAR/METplus

***The METplus GitHub repository is currently public. A user account is necessary if you plan on contributing code or modifying your code.**

1.3 Installing METplus

- Installing METplus from the Repository

After you have installed the METplus source code, you must then configure it to run on your machine.

1.4 Configuring and Running METplus

METplus and MET are designed to be highly configurable. Detailed information about configuring METPlus is found in this page and its subpages:

If you downloaded METplus from the GitHub repository, this page is for you. It explains in detail how to configure, and run METplus:

- Model Evaluation Tools Plus (METplus) (May 2017)
- METplus Configuration Guide

When we have set up and tested METplus for Rocoto, we will provide a detailed guide on how to run METplus using Rocoto here:

- METplus Rocoto Workflow

We will be adding more pages on these topics in the near future, including where to find log files, and troubleshooting problems.

1.5 Developing in METplus

Users new to the METplus scripting system should read the high-level overview pages before delving into the detailed documentation.

- METplus System Overview — provides a high-level overview of the structure of the METplus scripting system.
- ush – METplus utility scripts for wrapping MET.
- Package "produtil" — The produtil Python package creates a platform-independent environment for running METplus. This package is independent of the METplus system and can be used for other numerical weather and ocean prediction systems. It implements critical functionality missing from the Python 2.6 standard library. There are many alternatives to standard library functions and classes, which provide bug fixes to bugs in Python, adds logging and error checking, and provides workarounds for known problems on some platforms.
- produtil.config — Parses UNIX conf files and makes the result readily available. This is part of the produtil package and is referenced here as a convenience.

You can also explore the "Classes" and "Packages" tabs at the top of this page.

1.6 Generating the Website

This manual is a living document. It is generated from special comments in the scripts themselves by a program called Doxygen, a documentation generation suite. Users can generate the entire website, and a LaTeX version of the same, if they have **Doxygen version 1.8.9.1 or later**.

```
git clone https://www.github.com/NCAR/METplus
cd METplus/sorc/
make doc
cd ../doc
# copy the entire contents of the html/ directory to a web server
```

As the METplus code is updated, the documentation should be updated as well, and any public version of this manual should add the new version.

Chapter 2

EMC-NCEP-NOAA produtil library utility package

The produtil directory is a Platform-independent weather and ocean forecasting utility package. Developed at the National Oceanic and Atmospheric Administration (NOAA).

- Package "produtil" — The produtil Python package creates a platform-independent environment for running METplus. This package is independent of the METplus system and can be used for other numerical weather and ocean prediction systems. It implements critical functionality missing from the Python 2.6 standard library. There are many alternatives to standard library functions and classes, which provide bug fixes to bugs in Python, adds logging and error checking, and provides workarounds for known problems on some platforms.
- produtil.config — Parses UNIX conf files and makes the result readily available. This is part of the produtil package and is referenced here as a convenience.

Chapter 3

METplus Configuration Guide

Todo edit me: confguide.dox, THIS PAGE is a STARTING PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY REVIEWED AND EDITED.

This page explains how to modify the ∗.conf files in the METplus parm directory, and explains the purpose of the various conf files. There are a number of subpages with more detailed information:

- Local System Configuration (metplus_system.conf)
- available_configurations link to page that has examples of various configurations to run METplus.

Todo Add input source and METplus configuration pages.

The ∗.conf files configure various aspects of the METplus system. Users can also override these configuration settings manually on the command line. In the future, programmatically in the metplus.prelaunch() function.

These are the standard conf files read in by all workflows, in the order they are read in:

File	Purpose
parm/metplus.conf	Detailed configuration of most aspects of METplus.
parm/user.template.conf	Basic configuration settings.

The first file, metplus.conf, should not need to be modified unless one is making extensive changes to METplus, or adding new functionality to the underlying scripts. Instead, one can specify additional configuration files that override the default settings. See available_configurations for a list of alternate METplus configurations, and how to enable them.

The user.template.conf should be modified manually. In fact, it is meant as a starting point for a user to override the values set in metplus.conf in order to run METplus in the users environment.

This page documents how to modify the METplus parm/metplus_<xyz>.conf and user.template.conf file.

Todo edit me: confguide.dox, THIS PAGE is a STARTING PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY REVIEWED AND EDITED. It should be filled in with the information and configuration variables found in the README file Model Evaluation Tools Plus (METplus) (May 2017)

3.1 metplus.conf: [dir] Configure Run Areas

For Example:

```
[dir]
PROJ_DIR = /d1/data/SBU
MODEL_DATA_DIR = MODEL_DATA_DIR = {PROJ_DIR}/reduced_model_data
```

3.2 Local System Configuration (metplus_system.conf)

This page documents how to modify the METplus parm/metplus_system.conf amd user.template.conf file.

Todo edit me: metplus-conf.dox, THIS PAGE is a STARTING PLACEHOLDER. THIS PAGE NEEDS TO BE FU↩ LLY REVIEWED AND EDITED. It should be filled in with the information and configuration variables found in the README file Model Evaluation Tools Plus (METplus) (May 2017)

3.2.1 metplus.conf: [dir] Configure Run Areas

For Example:

```
[dir]
PROJ_DIR = /d1/data/SBU
MODEL_DATA_DIR = MODEL_DATA_DIR = {PROJ_DIR}/reduced_model_data
```

Chapter 4

All Configuration Files

Documentation for all configuration files in the parm/ directory. Subpages include:

- All configuration sections and options.
- File parm/metplus_system.conf
- File parm/metplus_runtime.conf
- File parm/metplus_data.conf

4.1 File parm/metplus_system.conf

This is a UNIX conf file that contains all information relating to the METplus configuration. UNIX conf is used because of how easy it is to parse (even GrADS can do it). The syntax:

```
[section]
var = value
```

For generation of namelists for WRF, WPS and other Fortran programs, we use this syntax:

```
[section]
namelist.nlvar = value
```

to set the value of namelist &namelist's nlvar variable. Also, the special variable "namelist" lists additional conf sections to recurse into to get more namelist variables after the current conf section is parsed. Any variable will only be set once: the first time it is seen.

Sets basic configuration options used by all components.

This section sets basic configuration options used by all components. Several special variables in this section are set by the ProdConfig object itself, which will overwrite them if they're set in this file: YMDHM = analysis time (201304261830 = April 26, 2013, 18:30 UTC) YMDH = analysis time excluding minute (2013042618) YMD = analysis time, excluding hour and minute year, YYYY = analysis time's year (ie.: 2013) YY = last two digits of year century, CC = first two digits of year month, MM = analysis time's month (ie.: 04) day, DD = analysis time's day (ie.: 26) hour, cyc, HH = analysis time's hour (ie.: 18) minute, min = analysis time's minute (ie.: 30)

There may be additional variables depending on what subclass (if any) of the ProdConfig is used. You must specify the mandatory EXPT value, which is the name of the experiment to run.

Commonly used base METplus variables are defined here, but can be over-ridden in subsequent configuration files indicated at the command line.

Commonly used base MET variables

This is a configuration override file. This file sets options in the following sections:

- [dir]
- [exe]

4.1.1 Section [exe]

Options in this section:

- WGRIB2 — NON-MET executables, used by METplus to perform specific tasks.
- RM_EXE
- CUT_EXE
- TR_EXE
- NCAP2_EXE
- CONVERT_EXE
- NCDUMP_EXE
- EGREP_EXE

4.1.1.1 [exe] WGRIB2

NON-MET executables, used by METplus to perform specific tasks.

```
[exe]
WGRIB2 = /path/to
```

Defined in File parm/metplus_system.conf

4.1.1.2 [exe] RM_EXE

```
[exe]
RM_EXE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.1.3 [exe] CUT_EXE

```
[exe]
CUT_EXE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.1.4 [exe] TR_EXE

```
[exe]
TR_EXE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.1.5 [exe] NCAP2_EXE

```
[exe]
NCAP2_EXE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.1.6 [exe] CONVERT_EXE

```
[exe]
CONVERT_EXE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.1.7 [exe] NCDUMP_EXE

```
[exe]
NCDUMP_EXE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.1.8 [exe] EGREP_EXE

```
[exe]
EGREP_EXE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.2 Section [dir]

Options in this section:

- METPLUS_BASE — METPLUS_BASE indicates the location of METplus code
- PARM_BASE
- OUTPUT_BASE
- MET_BUILD_BASE
- MET_BASE
- LOG_DIR — Output directories
- TMP_DIR

4.1.2.1 [dir] METPLUS_BASE

METPLUS_BASE indicates the location of METplus code

```
[dir]
METPLUS_BASE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.2.2 [dir] PARM_BASE

```
[dir]
PARM_BASE = {METPLUS_BASE}/parm
```

Defined in File parm/metplus_system.conf

4.1.2.3 [dir] OUTPUT_BASE

```
[dir]
OUTPUT_BASE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.2.4 [dir] MET_BUILD_BASE

```
[dir]
MET_BUILD_BASE = /path/to
```

Defined in File parm/metplus_system.conf

4.1.2.5 [dir] MET_BASE

```
[dir]
MET_BASE = {MET_BUILD_BASE}/share/met
```

Defined in File parm/metplus_system.conf

4.1.2.6 [dir] LOG_DIR

Output directories

```
[dir]
LOG_DIR = {OUTPUT_BASE}/logs
```

Defined in File parm/metplus_system.conf

4.1.2.7 [dir] TMP_DIR

```
[dir]
TMP_DIR = /path/to
```

Defined in File parm/metplus_system.conf

4.2 File parm/metplus_runtime.conf

This is a UNIX conf file that contains all information relating to the METplus configuration. UNIX conf is used because of how easy it is to parse (even GrADS can do it). The syntax:

```
[section]
var = value
```

to set the value of namelist &namelist's nlvar variable. Also, the special variable "namelist" lists additional conf sections to recurse into to get more namelist variables after the current conf section is parsed. Any variable will only be set once: the first time it is seen.

Sets basic configuration options used by all components.

This section sets basic configuration options used by all components. Several special variables in this section are set by the ProdConfig object itself, which will overwrite them if they're set in this file: YMDHM = analysis time (201304261830 = April 26, 2013, 18:30 UTC) YMDH = analysis time excluding minute (2013042618) YMD = analysis time, excluding hour and minute year, YYYY = analysis time's year (ie.: 2013) YY = last two digits of year century, CC = first two digits of year month, MM = analysis time's month (ie.: 04) day, DD = analysis time's day (ie.: 26) hour, cyc, HH = analysis time's hour (ie.: 18) minute, min = analysis time's minute (ie.: 30)

There may be additional variables depending on what subclass (if any) of the ProdConfig is used. You must specify the mandatory EXPT value, which is the name of the experiment to run.

This is a configuration override file. This file sets options in the following sections:

- [config]

4.2.1 Section [config]

Options in this section:

- EXPT — Experiment name, used for finding installation locations
- LOOP_METHOD — Options are processes, times processes: run each process to its completion before beginning the next process in the process list times: run every process in the process list for an init time, then run every process again for each subsequent init time in the init time list.
- PROCESS_LIST — Indicate the processes to run in master script (master_met_plus.py) All processes correspond to the class name in the Python files in the ush directory
- INIT_TIME_FMT
- INIT_BEG
- INIT_END
- INIT_INC
- LOG_LEVEL — Levels: DEBUG, INFO, WARNING, ERROR, CRITICAL
- LOG_FILENAME — NOTE: current YYYYMMDD is inserted before the rightmost . filename extension
- METPLUS_CONF
- CONFIG_DIR

4.2.1.1 [config] EXPT

Experiment name, used for finding installation locations

```
[config]
EXPT = METplus
```

Defined in File parm/metplus_runtime.conf

4.2.1.2 [config] LOOP_METHOD

Options are processes, times processes: run each process to its completion before beginning the next process in the process list times: run every process in the process list for an init time, then run every process again for each subsequent init time in the init time list.

```
[config]
LOOP_METHOD = processes
```

Defined in File parm/metplus_runtime.conf

4.2.1.3 [config] PROCESS_LIST

Indicate the processes to run in master script (master_met_plus.py) All processes correspond to the class name in the Python files in the ush directory

```
[config]
PROCESS_LIST = Usage
```

Defined in File parm/metplus_runtime.conf

4.2.1.4 [config] INIT_TIME_FMT

```
[config]
INIT_TIME_FMT = %Y%m%d
```

Defined in File parm/metplus_runtime.conf

4.2.1.5 [config] INIT_BEG

```
[config]
INIT_BEG = 20141214
```

Defined in File parm/metplus_runtime.conf

4.2.1.6 [config] INIT_END

```
[config]
INIT_END = 20141216
```

Defined in File parm/metplus_runtime.conf

4.2.1.7 [config] INIT_INC

```
[config]
INIT_INC = 21600
```

Defined in File parm/metplus_runtime.conf

4.2.1.8 [config] LOG_LEVEL

Levels: DEBUG, INFO, WARNING, ERROR, CRITICAL

```
[config]
LOG_LEVEL = DEBUG
```

Defined in File parm/metplus_runtime.conf

4.2.1.9 [config] LOG_FILENAME

NOTE: current YYYYMMDD is inserted before the rightmost . filename extension

```
[config]
LOG_FILENAME = {LOG_DIR}/master_met_plus.log
```

Defined in File parm/metplus_runtime.conf

4.2.1.10 [config] METPLUS_CONF

```
[config]
METPLUS_CONF = {OUTPUT_BASE}/metplus_final.conf
```

Defined in File parm/metplus_runtime.conf

4.2.1.11 [config] CONFIG_DIR

```
[config]
CONFIG_DIR = {METPLUS_BASE}/parm/met_config
```

Defined in File parm/metplus_runtime.conf

4.3 File parm/metplus_data.conf

This is a UNIX conf file that contains all information relating to the METplus configuration. UNIX conf is used because of how easy it is to parse (even GrADS can do it). The syntax:

```
[section]
var = value
```

to set the value of namelist &namelist's nlvar variable. Also, the special variable "namelist" lists additional conf sections to recurse into to get more namelist variables after the current conf section is parsed. Any variable will only be set once: the first time it is seen.

Sets basic configuration options used by all components.

This section sets basic configuration options used by all components. Several special variables in this section are set by the ProdConfig object itself, which will overwrite them if they're set in this file: YMDHM = analysis time (201304261830 = April 26, 2013, 18:30 UTC) YMDH = analysis time excluding minute (2013042618) YMD = analysis time, excluding hour and minute year, YYYY = analysis time's year (ie.: 2013) YY = last two digits of year century, CC = first two digits of year month, MM = analysis time's month (ie.: 04) day, DD = analysis time's day (ie.: 26) hour, cyc, HH = analysis time's hour (ie.: 18) minute, min = analysis time's minute (ie.: 30)

There may be additional variables depending on what subclass (if any) of the ProdConfig is used. You must specify the mandatory EXPT value, which is the name of the experiment to run.

Input data directories are defined here.

NOTE: These are EXAMPLE FILENAME TEMPLATES Define your own filename templates here.

GFS_FCST_FILE_TMPL = gfs_4_{init?fmt=Ymd}_{init?fmt=H}00_{lead?fmt=HHH}.grb2 GFS_FCST_NC_↩ FILE_TMPL = gfs_4_{init?fmt=Ymd}_{init?fmt=H}00_{lead?fmt=HHH}.nc GFS_ANLY_FILE_TMPL = gfs↩ _4_{valid?fmt=Ymd}_{valid?fmt=H}00_000.grb2 GFS_ANLY_NC_FILE_TMPL = gfs_4_{valid?fmt=Ymd}_↩ {valid?fmt=H}00_000.nc

This is a configuration override file. This file sets options in the following sections:

- [dir]
- [filename_templates]

4.3.1 Section [filename_templates]

4.3.2 Section [dir]

Options in this section:

- PROJ_DIR — This is the location of your input files for METplus
- MODEL_DATA_DIR

4.3.2.1 [dir] PROJ_DIR

This is the location of your input files for METplus

```
[dir]
PROJ_DIR = /path/to
```

Defined in File parm/metplus_data.conf

4.3.2.2 [dir] MODEL_DATA_DIR

```
[dir]
MODEL_DATA_DIR = {PROJ_DIR}/model_data
```

Defined in File parm/metplus_data.conf

Chapter 5

All Configuration Options

This page documents configuration options for all known sections:

5.1 Section [config]

Options in this section:

- EXPT — Experiment name, used for finding installation locations
- LOOP_METHOD — Options are processes, times processes: run each process to its completion before beginning the next process in the process list times: run every process in the process list for an init time, then run every process again for each subsequent init time in the init time list.
- PROCESS_LIST — Indicate the processes to run in master script (master_met_plus.py) All processes correspond to the class name in the Python files in the ush directory
- INIT_TIME_FMT
- INIT_BEG
- INIT_END
- INIT_INC
- LOG_LEVEL — Levels: DEBUG, INFO, WARNING, ERROR, CRITICAL
- LOG_FILENAME — NOTE: current YYYYMMDD is inserted before the rightmost . filename extension
- METPLUS_CONF
- CONFIG_DIR

5.1.1 [config] EXPT

Experiment name, used for finding installation locations

```
[config]
EXPT = METplus
```

Defined in File parm/metplus_runtime.conf

5.1.2 [config] LOOP_METHOD

Options are processes, times processes: run each process to its completion before beginning the next process in the process list times: run every process in the process list for an init time, then run every process again for each subsequent init time in the init time list.

```
[config]
LOOP_METHOD = processes
```

Defined in File parm/metplus_runtime.conf

5.1.3 [config] PROCESS_LIST

Indicate the processes to run in master script (master_met_plus.py) All processes correspond to the class name in the Python files in the ush directory

```
[config]
PROCESS_LIST = Usage
```

Defined in File parm/metplus_runtime.conf

5.1.4 [config] INIT_TIME_FMT

```
[config]
INIT_TIME_FMT = %Y%m%d
```

Defined in File parm/metplus_runtime.conf

5.1.5 [config] INIT_BEG

```
[config]
INIT_BEG = 20141214
```

Defined in File parm/metplus_runtime.conf

5.1.6 [config] INIT_END

```
[config]
INIT_END = 20141216
```

Defined in File parm/metplus_runtime.conf

5.1.7 [config] INIT_INC

```
[config]
INIT_INC = 21600
```

Defined in File parm/metplus_runtime.conf

5.1.8 [config] LOG_LEVEL

Levels: DEBUG, INFO, WARNING, ERROR, CRITICAL

```
[config]
LOG_LEVEL = DEBUG
```

Defined in File parm/metplus_runtime.conf

5.1.9 [config] LOG_FILENAME

NOTE: current YYYYMMDD is inserted before the rightmost . filename extension

```
[config]
LOG_FILENAME = {LOG_DIR}/master_met_plus.log
```

Defined in File parm/metplus_runtime.conf

5.1.10 [config] METPLUS_CONF

```
[config]
METPLUS_CONF = {OUTPUT_BASE}/metplus_final.conf
```

Defined in File parm/metplus_runtime.conf

5.1.11 [config] CONFIG_DIR

```
[config]
CONFIG_DIR = {METPLUS_BASE}/parm/met_config
```

Defined in File parm/metplus_runtime.conf

5.2 Section [dir]

Options in this section:

- METPLUS_BASE — METPLUS_BASE indicates the location of METplus code
- PARM_BASE
- OUTPUT_BASE
- MET_BUILD_BASE
- MET_BASE
- LOG_DIR — Output directories
- TMP_DIR
- PROJ_DIR — This is the location of your input files for METplus
- MODEL_DATA_DIR

5.2.1 [dir] METPLUS_BASE

METPLUS_BASE indicates the location of METplus code

```
[dir]
METPLUS_BASE = /path/to
```

Defined in File parm/metplus_system.conf

5.2.2 [dir] PARM_BASE

```
[dir]
PARM_BASE = {METPLUS_BASE}/parm
```

Defined in File parm/metplus_system.conf

5.2.3 [dir] OUTPUT_BASE

```
[dir]
OUTPUT_BASE = /path/to
```

Defined in File parm/metplus_system.conf

5.2.4 [dir] MET_BUILD_BASE

```
[dir]
MET_BUILD_BASE = /path/to
```

Defined in File parm/metplus_system.conf

5.2.5 [dir] MET_BASE

```
[dir]
MET_BASE = {MET_BUILD_BASE}/share/met
```

Defined in File parm/metplus_system.conf

5.2.6 [dir] LOG_DIR

Output directories

```
[dir]
LOG_DIR = {OUTPUT_BASE}/logs
```

Defined in File parm/metplus_system.conf

5.2.7 [dir] TMP_DIR

```
[dir]
TMP_DIR = /path/to
```

Defined in File parm/metplus_system.conf

5.2.8 [dir] PROJ_DIR

This is the location of your input files for METplus

```
[dir]
PROJ_DIR = /path/to
```

Defined in File parm/metplus_data.conf

5.2.9 [dir] MODEL_DATA_DIR

```
[dir]
MODEL_DATA_DIR = {PROJ_DIR}/model_data
```

Defined in File parm/metplus_data.conf

5.3 Section [exe]

Options in this section:

- WGRIB2 — NON-MET executables, used by METplus to perform specific tasks.
- RM_EXE
- CUT_EXE
- TR_EXE
- NCAP2_EXE
- CONVERT_EXE
- NCDUMP_EXE
- EGREP_EXE

5.3.1 [exe] WGRIB2

NON-MET executables, used by METplus to perform specific tasks.

```
[exe]
WGRIB2 = /path/to
```

Defined in File parm/metplus_system.conf

5.3.2 [exe] RM_EXE

```
[exe]
RM_EXE = /path/to
```

Defined in File parm/metplus_system.conf

5.3.3 [exe] CUT_EXE

```
[exe]
CUT_EXE = /path/to
```

Defined in File parm/metplus_system.conf

5.3.4 [exe] TR_EXE

```
[exe]
TR_EXE = /path/to
```

Defined in File parm/metplus_system.conf

5.3.5 [exe] NCAP2_EXE

```
[exe]
NCAP2_EXE = /path/to
```

Defined in File parm/metplus_system.conf

5.3.6 [exe] CONVERT_EXE

```
[exe]
CONVERT_EXE = /path/to
```

Defined in File parm/metplus_system.conf

5.3.7 [exe] NCDUMP_EXE

```
[exe]
NCDUMP_EXE = /path/to
```

Defined in File parm/metplus_system.conf

5.3.8 [exe] EGREP_EXE

```
[exe]
EGREP_EXE = /path/to
```

Defined in File parm/metplus_system.conf

5.4 Section [filename_templates]

Chapter 6

METplus Installation Guide

Todo edit me: install-main.dox, THIS PAGE is a STARTING PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY REVIEWED AND EDITED.

Obtain METplus from the public GitHub repository: https://github.com/NCAR/METplus

6.1 Installation from the Public Repository

Most of this guide explains how to compile, install and run the HWRF system from the public repository:

- Installing METplus from the Repository

If you are downloading from the DTC Subversion server, then that is the page for you.

6.2 Public METplus Release Tarballs

If you download METplus tarballs off of the DTC website, this guide will not help you. The installation process is different. See the DTC webpage here:

http://www.dtcenter.org/met/users/docs/

and search for the version of METplus that you are using under "METplus Documents."

6.3 Installing METplus from the Repository

This page explains how to install METplus from the github repository housed here:

Todo edit me: install.dox, THIS PAGE NEEDS TO BE FULLY REVIEWD AND EDITED.

A user account is required to access the METplus repository.

- METplus GitHub Developer Page

Specifically, it explains how to install from the branch, tag or trunk that you checked out. The guide is actually generated from special comments and documentation files inside that repository.

6.3.1 Prerequisites

6.3.1.1 Prerequisites: Scripting Languages

You may need to install some additional software. If you have Linux, MacOS or open-source BSD distribution, these are likely already installed, or can be installed via your OS installation command (apt-get, yum, etc.)

Language	Why	Command	To obtain
POSIX sh	Job setup	/bin/sh	Always present on POSIX-compliant operating systems.
Python 2.7	Workflow	python	https://www.python.org/downloads/release
GNU make	Build system (for documentation)	gmake	http://www.gnu.org/software/make/
Doxygen 1.8.↩ 9.1	Create documentation		http://www.stack.nl/~dimitri/doxygen/dow
MET	For MET applications		https://dtcenter.org/met/users
R 3.25	If using MET plot_tcmpr.R		included in MET download (see above)

Note that Python must be version 2.x, and at least version 2.7. Python 3 is a completely different language than Python 2, and the METplus scripts are all Python 2 scripts. You can determine your version of Python using this command:

```
python --version
```

If your version of Python is 3, you may also have a "python2" program:

```
python2 --version
```

If your "python" command is version 3, and python2 is version 2, you can still run METplus. However, you will need to edit the ∗.py files in ush/, scripts/ and rocoto/, and change:

```
#! /bin/env python
```

to:

```
#! /bin/env python2
```

6.3.1.2 Prerequisites: Workflow Automation Programs

Currently, large-scale METplus verification has not been set up or tested. When METplus supports large-scale tasks, it will first support Rocoto, then eventually ecFlow.

What	Why	Command	To obtain
Rocoto	Workflow Automation	rocoto	https://github.com/christopherwharrop/rocoto/releases
ecFlow	Alternative to Rocoto	ecflow-client	https://software.ecmwf.int/wiki/display/ECFLOW/Releas

6.3.2 Step 1: METplus Repository Checkout

The first step is to check out METplus from the repository.

```
git clone https://www.github.com/NCAR/METplus
```

6.3.2.1 Step 1.2: Now Configure and Run

- Configuring and Running METplus

Chapter 7

Local Data Configuration (metplus_data.conf)

This page documents how to modify the METplus parm/metplus_data.conf amd user.template.conf file.

Todo edit me: metplus-conf.dox, THIS PAGE is a STARTING PLACEHOLDER. THIS PAGE NEEDS TO BE FU↩ LLY REVIEWED AND EDITED. It should be filled in with the information and configuration variables found in the README file Model Evaluation Tools Plus (METplus) (May 2017)

7.1 metplus.conf: [dir] Configure Run Areas

For Example:

```
[dir]
PROJ_DIR = /d1/data/SBU
MODEL_DATA_DIR = MODEL_DATA_DIR = {PROJ_DIR}/reduced_model_data
```

Chapter 8

METplus System Overview

8.1 Introduction

METplus provides wrappers to MET in an effort to make MET easily configurable and easy to run for both new and experienced MET users. METplus is written entirely in Python.

8.2 Overall System Design

As of the beta release of METplus, the following MET applications have corresponding METplus "wrappers":

- Grid-Stat
- PCP-Combine
- MODE
- Regrid-Data-Plane
- Series-Analysis
- TC-Pairs
- TC-Stat
- tcmpr_plotter.R

*In the future, all MET applications will have corresponding METplus wrappers, including METViewer.

8.2.1 Workflow Layer

Content to be added when integration to Rocoto workflow has been complete.

8.3 Portability Layer

The Portability Layer is a Python package (see produtil) which implements cross-platform methods of doing common tasks. For example, it implements a way of running MPI, OpenMP and serial programs in a cross-platform manner. It can perform file operations with improved logging, interact with the batch system, identify limitations of the cluster, deal with restricted data classes, manipulate resource limits, and interact with a database file. Currently, METplus uses this for logging, configuration files, and running/executing MET and other commands.

- produtil

Chapter 9

Model Evaluation Tools Plus (METplus) (May 2017)

Welcome to the documentation for METplus. METplus is a set of Python wrapper scripts around the MET verification tools (and eventually METViewer, a tool used for plotting MET output verification statistics).

Background and Future

METplus development began in 2016 with initial development for the cyclone-relative verification for the Stony Brook University (SBU) project. Development in 2017 will focus on replicating the Global Deterministic National Centers for Environmental Prediction (NCEP) Verification and future work will focus on ensemble, meso, and storm scale verification at NCEP and public support.

Dependencies

The MET verification tools package is required to be installed on your system prior to running the METplus wrapper scripts.

METplus was developed using Python version 2.7.9. Python version 2.7 or greater is required.

- METplus requires the following to be installed on your system:
  - ncdump utility
    - http://www.unidata.ucar.edu/downloads/netcdf/index.jsp
  - ncap2 utility
    - http://nco.sourceforge.net/
  - convert utility (part of ImageMagick)
    - https://www.imagemagick.org/script/binary-releases.php
  - wgrib2 utility
    - http://www.cpc.noaa.gov/products/wesley/wgrib2/compile_questions.html
  - egrep utility
    - http://directory.fsf.org/wiki/Grep
  - rm, cut, tr utilities (standard on Linux)

Version Control

METplus uses GIT for version control in a public GitHub repository: NCAR/METplus.

Getting the Code and Test Data

Get the METplus package by running:

wget http://www.dtcenter.org/met/users/downloads/METplus/METplus_vX.2017XXXX.tar

Get the METplus test data by running:

wget http://www.dtcenter.org/met/users/downloads/METplus//METplus_test_data.20170109.tar.

Decide where you would like to put the code and copy the METplus Package to that location. Unpack the gzipped tar file in that directory by running:

tar -xf METplus_vX.2017XXXX.tar

Decide where you would like to put the test data and copy the METplus test data to that location. Unpack the gzipped tar file in that directory by running:

tar -zxf METplus_test_data.20170109.tar.gz

Configuring the Environment

Configure the following environment variables in your login shell. The example below assumes C shell. Open up your .cshrc (or similar file) using the editor of your choice and add the following:

To your PYTHONPATH, add:

(full path to METplus/ush):${PYTHONPATH} (replacing the text and () with the full path)

If you do not currently have a PYTHONPATH, add:

setenv PYTHONPATH (full path to METplus/ush) (replacing the text and () with the full path)

To your PATH (path), add:

setenv PATH ${PATH}:(full path to METplus/ush) (replacing the text and () with the full path)

Optional: Add the METplus job log file, JLOGFILE

setenv JLOGFILE (full path/filename) (replacing the text in () with the desired full path and filename of your job log file.

Save the changes and source your .cshrc (or similar file) by running, for example:

```
source ~/.cshrc
```

Configuration Files

There are two sets of configuration files - one for running METplus and one for running MET.

- METplus

The main configuration file for METplus is metplus.conf, which is located in the "parm" subdirectory.

Users have the ability to override specific fields in metplus.conf in their own config file. As a starting point, an example file with some typical fields to override is user.template.conf

- MET

The configuration files for the MET tools are also located in the "parm" subdirectory. Currently, the applicable configuration files are TCPairsETCConfig (for the extra tropical cyclone TCPairs run) and the SeriesAnalysis↩ Config_by_init and SeriesAnalysisConfig_by_lead.

Configuration Setup

The user should look at metplus.conf to modify necessary and desired fields. The information below will cover the various variables:

NON-MET EXECUTABLES

Some of these fields may need to be modified based on the location of the executables on your system, but some may be standard. Note that the WGRIB2 executable is not currently in a standard location and will need to be modified.

COMMONLY USED BASE VARIABLES

MET_BUILD_BASE is the base location for the MET release that you will be using. Please set that to an appropriate location.

OUTPUT_BASE is the base area for where the user would like to store their output data.

PARM_BASE is the parm subdirectory for the METplus configuration files.

MET EXECUTABLES

These fields rely on MET_BUILD_BASE and its "bin" subdirectory.

INPUT DATA DIRECTORIES

These fields indicate where your input data is located.

For example, the METplus_test_data.20170109.tar.gz, includes a "reduced_model_data" directory, which contains GFS data, and a "track_data" directory, which contains extra tropical cyclone track data. If you wanted to put this data at "/d1/data/SBU", you would set the following:

PROJ_DIR = /d1/data/SBU MODEL_DATA_DIR = GFS_DIR = {PROJ_DIR}/reduced_model_data TRACK_DAT↩ A_DIR = {PROJ_DIR}/track_data

OUTPUT DIRECTORIES

These fields include a log directory and a tmp directory along with other output directories. The TRACK_DATA_S↩ UBDIR_MOD refers to the subdirectory where the track data will be written, reformatted to be in true ATCF format, which the MET tools need for processing.

FILENAME TEMPLATES

These fields contain templates for filenames and filename prefixes and regular expressions.

CONFIGURATION FILES

These fields indicate which configuration files to use for MET.

LISTS AND SETTINGS

PROCESS_LIST is the list of processes that the user wants the master script to run. For example, a full run from start to finish for running series analysis by lead, would be:

PROCESS_LIST = run_tc_pairs.py, extract_tiles.py, series_by_lead.py

STAT_LIST is the list of statistics to be computed (e.g. STAT_LIST = TOTAL, FBAR, OBAR). NOTE: Currently, "TOTAL" is a REQUIRED cnt statistic used by the series analysis scripts, so it must be in the STAT_LIST.

INIT_DATE_BEG is the beginning date in the format YYYYMMDD (e.g. 20141201) for the initialization time. INIT_DATE_END is the ending date in the format YYYYMMDD (e.g. 20150331) for the initialization time. INIT_H↩ OUR_INC is the hour increment in the format H $<$ 10 or HH $>=$ 10 (e.g. 6) INIT_HOUR_END is the last increment hour you'd like to process in the format (e.g. For the sample data provided, GFS has 00, 06, 12, and 18, so this value would be "18")

VAR_LIST OR EXTRACT_TILES_VAR_LIST is the list of variables of interest with their levels: Values SHOU↩ LD NOT be present in both the VAR_LIST and the EXTRACT_TILES_VAR_LIST. e.g. VAR_LIST = HGT/P500, PRMSL/Z0, TMP/Z2 EXTRACT_TILES_VAR_LIST =

The following are used for performing series analysis based on lead time: FHR_BEG is the beginning forecast time. FHR_END is the ending forecast time. FHR_INC is the forecast hour increment.

NLAT and NLON are the dimensions of the tile.

DLAT and DLON is the resolution of the data in degrees.

LON_ADJ and LAT_ADJ are the degrees to subtract from the center lat and lon to calculate the lower left lat (lat_ll) and lower left lon (lon_ll) for a grid that is 2n X 2m, where n = LAT_ADJ degrees and m = LON_ADJ degrees. For example, where n=15 and m=15, this results in a 30 deg X 30 deg grid.

TC PAIRS filtering options

These variables contains the options used for the call to MET's tc_pairs code.

TC-STAT filtering options

These variables contains the filtering options for the call to MET's tc_stat code.

OVERWRITE OPTIONS

These variables exist so that you can choose whether or not to overwrite already processed data sets.

PLOTTING

These variables contains the possible plotting options

REGRIDDING

Tese variables contain the possible regridding options. REGRID_USING_MET_TOOL is currently set to FALSE, as METplus is currently using wgrib2, as opposed to regrid_data_plane, for part of its processing.

TESTING

These options are currently used by the developers and shouldn't need to be modified.

LOGGING

These variables contain the logging options. A LOG_LEVEL of "DEBUG" will likely provide too much information for the general user, so the user may wish to start off with "INFO" instead.

How to Run?

Once you have set up user.template.conf you can simply run:

master_met_plus.py -c user.template.conf

Release Notes

Alpha Release Notes:

2017 May 9: METplus is now using the NOAA/NCEP/EMC produtil package. This changed how configuration files, logging, subprocess execution, and some file operations are implemented.

2017 Jan:

- Initial release of the code.

Chapter 10

Model Evaluation Tools Plus (METplus) TERMS OF USE - IMPORTANT!

USE OF THIS SOFTWARE IS SUBJECT TO THE FOLLOWING TERMS AND CONDITIONS:

1. License. Subject to these terms and conditions, University Corporation for Atmospheric Research (UCAR) grants you a non-exclusive, royalty-free license to use, create derivative works, publish, distribute, disseminate, transfer, modify, revise and copy the Model Evaluation Tools Plus (MET plus) software.

   You shall not sell, license or transfer for a fee the Software, or any work that in any manner contains the Software.

2. Disclaimer of Warranty on Software. Use of the Software is at your sole risk. The Software is provided "AS IS" and without warranty of any kind and UCAR EXPRESSLY DISCLAIMS ALL WARRANTIES AND/OR COND↩ ITIONS OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT OF A THIRD PARTY'S INTELLECTUAL PROPER↩ TY, MERCHANTABILITY OR SATISFACTORY QUALITY AND FITNESS FOR A PARTICULAR PURPOSE. THE PARTIES EXPRESSLY DISCLAIM THAT THE UNIFORM COMPUTER INFORMATION TRANSACTI↩ ONS ACT (UCITA) APPLIES TO OR GOVERNS THIS AGREEMENT. No oral or written information or advice given by UCAR or a UCAR authorized representative shall create a warranty or in any way increase the scope of this warranty. Should the Software prove defective, you (and neither UCAR nor any UCAR representative) assume the cost of all necessary correction.

3. Limitation of Liability. UNDER NO CIRCUMSTANCES, INCLUDING NEGLIGENCE, SHALL UCAR BE LIA↩ BLE FOR ANY DIRECT, INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES INCLUD↩ ING LOST REVENUE, PROFIT OR DATA, WHETHER IN AN ACTION IN CONTRACT OR TORT ARISING OUT OF OR RELATING TO THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF UCAR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4. Compliance with Law. All Software and any technical data delivered under this Agreement are subject to U.S. export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with all applicable laws and regulations in connection with use and distribution of the Software, including export control laws, and you acknowledge that you have responsibility to obtain any required license to export, re-export, or import as may be required.

5. No Endorsement/No Support. The names UCAR/NCAR, National Center for Atmospheric Research and the University Corporation for Atmospheric Research may not be used in any advertising or publicity to endorse or promote any products or commercial entity unless specific written permission is obtained from UCAR. The Software is provided without any support or maintenance, and without any obligation to provide you with modifications, improvements, enhancements, or updates of the Software.

6. Controlling Law and Severability. This Agreement shall be governed by the laws of the United States and the State of Colorado. If for any reason a court of competent jurisdiction finds any provision, or portion thereof, to be unenforceable, the remainder of this Agreement shall continue in full force and effect. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is hereby expressly excluded.

7. Termination. Your rights under this Agreement will terminate automatically without notice from UCAR if you fail to comply with any term(s) of this Agreement. You may terminate this Agreement at any time by destroying the Software and any related documentation and any complete or partial copies thereof. Upon termination, all rights granted under this Agreement shall terminate. The following provisions shall survive termination: Sections 2, 3, 6 and 9.

8. Complete Agreement. This Agreement constitutes the entire agreement between the parties with respect to the use of the Software and supersedes all prior or contemporaneous understandings regarding such subject matter. No amendment to or modification of this Agreement will be binding unless in writing and signed by UCAR.

9. Notices and Additional Terms. Copyright in Software is held by UCAR. You must include, with each copy of the Software and associated documentation, a copy of this Agreement and the following notice:

   "The source of this material is the Research Applications Laboratory at the National Center for Atmospheric Research, a program of the University Corporation for Atmospheric Research (UCAR) pursuant to a Cooperative Agreement with the National Science Foundation; Copyright 2007 University Corporation for Atmospheric Research. All Rights Reserved."

   The following notice shall be displayed on any scholarly works associated with, related to or derived from the Software:

   "Model Evaluation Tools Plus (METplus) was developed at the National Center for Atmospheric Research (NCAR) through grants from the National Oceanic and Atmospheric Administration (NOAA). NCAR is sponsored by the United States National Science Foundation (NSF)."

By using or downloading the Software, you agree to be bound by the terms and conditions of this Agreement.

Chapter 11

Running METplus

Todo edit me: run.dox, THIS PAGE is a PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY EDITED and REVIEWD..

This page and its subpages explain how to run METplus, track what it is doing during the execution, and figure out what is going wrong.

11.1 Running METplus

There are several ways to run METpus:

- Use a workflow automation system.
  - Use ecFlow, if you work in NCEP Central Operations (NCO).
  - Use Rocoto, if you don't work in NCO.
- In interactive batch jobs with METplus wrappers.
- Manual execution for debugging:
  - Directly run ex-scripts from the shell.
  - Manually run METPlus Python functions.

The direct methods are more laborious than automation systems. However, direct methods are more useful for debugging if one is developing new capabilities in METplus, or porting it to new supercomputers. For details on each method, visit one of the three above links.

For information about METplus wrappers, see the METplus User's Guide on the public METplus website: http://www.dtcenter.org/met/users/

11.2 Monitoring METplus

After you start the Rocoto-based or ecFlow-based workflow, a sequence of jobs will run, and some may fail, requiring user intervention. In order to know this, you must monitor the progress of the HWRF forecast cycles. Both ecFlow and Rocoto have means by which to check which jobs are queued, submitted, completed or failed. See the METplus Rocoto Workflow page for details on doing this in Rocoto.

11.2.1 METplus Directory Structure

Knowing where data will show up can tell you a lot about what is going on in METplus and why. The METplus system has several key directories:

- WORKmetplus — the work directory for each cycle. Each storm and cycle has its own work directory. In the Rocoto-based workflow, this directory contains log files.
- intercom — a directory used to trade data between jobs for one storm and cycle. This is inside the WORKhwrf directory.
- com — the com directory for each cycle. A job for one storm or cycle will never access another storm or cycle data except through its com directory.
- log — contains log files that are not specific to a storm or cycle.

For great detail on the METplus directory structure:

Todo Add METplus directory structure page.

11.2.2 Detailed Logging

Logging in METplus is quite extensive. For details on METplus logging, we refer you to an entire page on the matter:

- metplus-log-files

11.3 METplus Rocoto Workflow

Todo edit me: rotoco.dox, THIS PAGE is a PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY CREATED ONCE Rocoto support is available in METplus.

METplus has not yet been set up to support the Rocoto workflow manager, but will be in the near future.

Chapter 12

METplus wiki contents

Model Evaluation Tools Plus (METplus) wiki contents

Welcome to the METplus wiki contents documentation.

The source of this documentation is from the GitHub NCAR METplus website.

This is just an example page of providing the content in a doxygen page.

wiki content diagrams

`extract_tiles_Activity`

`extract_tiles.py_Sequence_Diagram`

`GitHub_process`

`series_by_lead_Activity`

`series_by_lead_Sequence_Diagram`

Current Use Cases

`Current Uses Cases`

Example in-line image

Chapter 13

Wiki Contents

THIS FILE is not meant to by referenced by any pages. It purely exists to pull in the wiki_contents images file.

USE THIS FILE to List all image files that you want to have available in your documentation via links.

Note:

Using and processing images in doxygen also requires you set the IMAGE_PATH in the doxygen config file, Doxyfile.in

The only way doxygen pulls in image files and places them in the html output directory it generates is with the

If you plan on using images in-line, with the image command, than you really don't need to list your file here. However, if you want to use a link to an image file (like below) than you want to make sure the image file is present and exists in the html directory generated by doxygen. You do that, by using the image special-command and listing it in this file.

These are in-line images, using the image command causes them to get placed in the html ouput directory.

wiki_contents/diagrams

wiki_contents/images

Chapter 14

METplus Repository README File

Welcome to the documentation for the Model Evaluation Tools Plus (METplus).

This is the METplus repository Top level README.md

Basic DOCUMENTATION - getting started

ALL Documentation specific to this repository can be found in the doc/ directory.

The ORIGINAL setup text documentation in a markdown file is found here.

- doc/README_install.md — installation, configuration, running
- doc/README_terms_of_use.md — legal Terms Of Use

METplus is a Python scripting infrastructure around the MET verification tools (and eventually METViewer, a tool used for plotting MET output verification statistics).

This infrastructure utilizes the NCEP produtil package. A Platform-independent weather and ocean forecasting utility package. Developed at the National Oceanic and Atmospheric Administration (NOAA).

Website Documentation

Users can generate an entire METplus documentation website, if they have Doxygen version 1.8.9.1 or later installed.

```
cd METplus/sorc
make doc
In your browser, open the page  METplus/doc/html/index.html
```

Terms of Use

Model Evaluation Tools Plus (METplus) TERMS OF USE - IMPORTANT!

Install/Configure/Run Guide

Model Evaluation Tools Plus (METplus) (May 2017)

Produtil Guide

EMC-NCEP-NOAA produtil library utility package

GitHub METplus - User Account Required

METplus GitHub Developer Page

GitHub wiki contents

METplus wiki contents

Chapter 15

Todo List

Page Installing METplus from the Repository

edit me: install.dox, THIS PAGE NEEDS TO BE FULLY REVIEWD AND EDITED.

Page Local Data Configuration (metplus_data.conf)

edit me: metplus-conf.dox, THIS PAGE is a STARTING PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY REVIEWED AND EDITED. It should be filled in with the information and configuration variables found in the README file Model Evaluation Tools Plus (METplus) (May 2017)

Page Local System Configuration (metplus_system.conf)

edit me: metplus-conf.dox, THIS PAGE is a STARTING PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY REVIEWED AND EDITED. It should be filled in with the information and configuration variables found in the README file Model Evaluation Tools Plus (METplus) (May 2017)

Page METplus Configuration Guide

edit me: confguide.dox, THIS PAGE is a STARTING PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY REVIEWED AND EDITED.

Add input source and METplus configuration pages.

Page METplus Installation Guide

edit me: install-main.dox, THIS PAGE is a STARTING PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY REVIEWED AND EDITED.

Page METplus Rocoto Workflow

edit me: rotoco.dox, THIS PAGE is a PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY CREATED ONCE Rocoto support is available in METplus.

Page Running METplus

edit me: run.dox, THIS PAGE is a PLACEHOLDER. THIS PAGE NEEDS TO BE FULLY EDITED and REVIE↩ WD..

Add METplus directory structure page.

Chapter 16

Bug List

Member produtil.mpi_impl.mpi_impl_base.CMDFGen.__init__ (self, base, lines, cmd_envar='SCR_CMDFI↩ LE', model_envar=None, filename_arg=False, kwargs)

The base_suffix keyword is used for both the suffix and prefix

Member produtil::run.runbg (arg, capture=False, kwargs)

produtil.run.runbg() is not implemented

Member produtil::run.waitprocs (procs, logger=None, timeout=None, usleep=1000)

produtil.run.waitprocs() is untested

Chapter 17

Namespace Index

17.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 18

Hierarchical Index

18.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 19

Class Index

19.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 20

Namespace Documentation

20.1 command_builder Namespace Reference

20.1.1 Detailed Description

```
Program Name: CommandBuilder.py
Contact(s): George McCabe
Abstract:
History Log:  Initial version
Usage: Create a subclass
Parameters: None
Input Files: N/A
Output Files: N/A
```

Classes

- class CommandBuilder

20.2 confdoc Namespace Reference

Generates the doc/config-files.dox, which documents configuration files.

20.2.1 Detailed Description

Generates the doc/config-files.dox, which documents configuration files.

Run by the documentation generator in `sorc/doc/compile`. Syntax:

```
../../ush/confdoc.py ../../parm/metplus.conf ... more ... > ../../doc/config-files.dox
```

That will read in the listed conf files, process Doxygen-like comments, and generate multiple pages of documentation. There will be one page for each conf file, another page listing all sections and options, and a final top-level page

The documentation comments are similar to the syntax Doxygen uses for Python, but with the addition of ";;" comments for documenting options on the same line they are defined:

```
## Brief description of section
#
# Detailed description of section
[section]
option1 = value ;; Brief description of option1

## Brief description of option2
#
# Detailed description of option2
option2 = value
```

The descriptions can contain the usual Doxygen and Markdown syntax.

There are a number of pages and sections generated with the following anchors. These are the page anchors:

- conf-files — Main page that lists all subpages.
- conf-options — Subpage that contains all section and option documentation
- conf-file-[filename] — Page for the specified file. Any dots (".") are replaced with underscores ("_") in the filename.

These are the section anchors:

- conf-sec-runwrf — Documentation section for the [runwrf] section
- conf-sec-runwrf-wm3c_ranks — Documentation subsection for the wm3c_ranks option in the [runwrf] section. Any percent signs ("%") in the option name are replaced with "-P-"
- conf-[filename]-runwrf — Documentation for the [runwrf] section in the specified file
- conf-[filename]-runwrf-wm3c_ranks — Documentation for the [runwrf] section's wm3c_ranks option in the specified file.

Classes

- class coredoc

  Subclass of override, for documenting the core configuration files.
- class docbase

  Stores documentation for all configuration options and sections.
- class override

  Subclass of docbase for documenting files that override the base configuration.
- class parsefile

  Config file parser.

Functions

- def main (args)

  Main program for confdoc.

20.2.2 Function Documentation

20.2.2.1 main()

```
def confdoc.main (
              args )
```

Main program for confdoc.

See the confdoc documentation for details.

Definition at line 772 of file confdoc.py.

20.3 config_metplus Namespace Reference

The initial METplus configure script for parsing the command line options, arguments and setting up the METPL↩ US_CONF file.

20.3.1 Detailed Description

The initial METplus configure script for parsing the command line options, arguments and setting up the METPL↩ US_CONF file.

This module setup() function should be called at the start of each task to setup a configuration object used by all the processing tasks. Each task that calls this MUST have run produtil.setup

Functions

- def usage (filename=None, logger=None)

  How to call this function.
- def setup (filename=None, logger=None)

  The METplus setup fuction.

Variables

- logger = None

  The logging.Logger for log messages.
- **send_dbn**
- **False**
- **jobname**
- **jlogfile**
- **exc_info**

20.3.2 Function Documentation

20.3.2.1 setup()

```
def config_metplus.setup (
              filename = None,
              logger = None )
```

The METplus setup fuction.

Parameters

filename	the filename of the calling module.
logger	a logging.logger for log messages

The setup function that process command line options and arguements and returns a configuration object.

Definition at line 54 of file config_metplus.py.

Referenced by tc_pairs_wrapper.TcPairsWrapper.build_tc_pairs(), series_by_lead_wrapper.SeriesByLead↩ Wrapper.create_animated_gifs(), series_by_init_wrapper.SeriesByInitWrapper.create_fcst_anly_to_ascii_file(), tcmpr_plotter_wrapper.TCMPRPlotterWrapper.retrieve_optionals(), and extract_tiles_wrapper.ExtractTiles↩ Wrapper.run_at_time().

20.3.2.2 usage()

```
def config_metplus.usage (
              filename = None,
              logger = None )
```

How to call this function.

Parameters

filename	the filename of the calling module.
logger	a logging.logger for log messages

Definition at line 32 of file config_metplus.py.

Referenced by setup().

20.4 extract_tiles_wrapper Namespace Reference

20.4.1 Detailed Description

```
Program Name: ExtractTiles.py
Contact(s): Julie Prestopnik, Minna Win
Abstract: Extracts tiles to be used by series_analysis
History Log:  Initial version
Usage: ExtractTiles.py
Parameters: None
Input Files: tc_pairs data
Output Files: tiled grib2 files
Condition codes: 0 for success, 1 for failure
```

Classes

- class ExtractTilesWrapper

  Takes tc-pairs data and regrids paired data to an n x m grid as specified in the config file.

Variables

- **send_dbn**
- **False**
- **jobname**
- **jlogfile**
- **CONFIG_INST** = config_metplus.setup()
- **ET** = ExtractTilesWrapper(CONFIG_INST, logger=None)
- **exc_info**

20.5 ExtraTropicalCyclonePlotter Namespace Reference

A Python class that generates plots of extra tropical cyclone forecast data, replicating the NCEP tropical and extra tropical cyclone tracks and erification plots http://www.emc.ncep.noaa.gov/mmb/gplou/emchurr/glblgen/.

20.5.1 Detailed Description

A Python class that generates plots of extra tropical cyclone forecast data, replicating the NCEP tropical and extra tropical cyclone tracks and erification plots http://www.emc.ncep.noaa.gov/mmb/gplou/emchurr/glblgen/.

20.6 gempak_to_cf_wrapper Namespace Reference

20.6.1 Detailed Description

```
Program Name: gempak_to_cf.py
Contact(s): George McCabe
Abstract: Runs GempakToCF
History Log:  Initial version
Usage:
Parameters: None
Input Files:
Output Files:
Condition codes: 0 for success, 1 for failure
```

Classes

- class GempakToCFWrapper

20.7 grid_stat_wrapper Namespace Reference

20.7.1 Detailed Description

```
Program Name: grid_stat_wrapper.py
Contact(s): George McCabe
Abstract:
History Log:  Initial version
Usage:
Parameters: None
Input Files:
Output Files:
Condition codes: 0 for success, 1 for failure
```

Classes

- class GridStatWrapper

20.8 mode_wrapper Namespace Reference

20.8.1 Detailed Description

```
Program Name: mode_wrapper.py
Contact(s): George McCabe
Abstract: Runs mode
History Log:  Initial version
Usage:
Parameters: None
Input Files:
Output Files:
Condition codes: 0 for success, 1 for failure
```

Classes

- class ModeWrapper

20.9 pcp_combine_wrapper Namespace Reference

20.9.1 Detailed Description

```
Program Name: pcp_combine_wrapper.py
Contact(s): George McCabe
Abstract: Runs pcp_combine to merge multiple forecast files
History Log:  Initial version
Usage:
Parameters: None
Input Files: grib2 files
Output Files: pcp_combine files
Condition codes: 0 for success, 1 for failure
```

Classes

- class PcpCombineWrapper

20.10 produtil Namespace Reference

Platform-independent weather and ocean forecasting utility package.

20.10.1 Detailed Description

Platform-independent weather and ocean forecasting utility package.

The produtil package is a general production weather and ocean forecasting utility package. It implements a number of classes and functions needed to implement a reliable, cross-platform weather or ocean forecasting system. This package is entirely model-independent: nothing in it is specific to, or reliant on, the HWRF model.

Note that before you use anything in this module, you must first call the produtil.setup.setup() function, and that function should only be called once per process. Generally this is done at the top of the main program.

20.10.2 File and Product Manipulation

There are a number of file and directory manipulation routines in the produtil package. In many cases, these replace Python standard library routines that either have known bugs or lack logging functionality. If a function exists in produtil and the Python standard library, it is best to use the produtil version to avoid Python's bugs.

- produtil.fileop — Many simple routines to manipulate files and directories. Works around many Python bugs and adds logging to file manipulation routines.
- produtil.acl — A wrapper around libacl. This is used by the produtil.fileop.deliver_file() to copy access control lists (ACLs)
- produtil.cd — Two classes to implement safe cd-in-cd-out blocks using the Python "with" construct. Also implements temporary directories and deletion of pre-existing directories if requested.
- produtil.locking — File locking that works around Lustre, GPFS and Panasas bugs.
- produtil.retry — Retry operations.
- produtil.rstprod — Handle NOAA restricted data requirements.
- produtil.dbn_alert — Trigger DBNet alerts.
- produtil.datastore — A database and product tracking.
- produtil.atparse — A simple text preparser.

20.10.3 Program Execution

The produtil package has flexible, shell-like syntaxes for specifying program execution, including complex MPI execution with multiple executables. Most critically, this package works around a bug in Python's subprocess module, which forgets to close pipes after a fork() call, causing deadlocks in multi-stage pipelines. That bug renders Python's subprocess module worthless for complex pipelines. The produtil.run does not suffer from that problem.

- produtil.run — shell-like syntax for running programs, including a cross-platform way of requesting MPI and OpenMP program execution.
- produtil.prog, produtil.mpiprog — Object tree that underlies the produtil.run implementation.
- produtil.mpi_impl — Contains one module for each MPI implementation supported by produtil.
- produtil.pipeline — Launches and monitors processes for produtil.run.

You should never need to access the mpi_impl or pipeline modules directly, and you should only need the prog and mpiprog modules for type checking. (For example, is my argument a produtil.prog.ImmutableRunner?) In nearly all cases, you can use the produtil.run functions to access the full functionality of all of the program execution modules.

20.10.4 Other Utilities

- produtil.setup — Contains the produtil.setup.setup() function, which initializes the entire produtil package.
- produtil.log — Initialization of the logging module. Sets up logging to match what is required in the NCEP production environment. This is highly configurable (as is the Python logging module).
- produtil.sigsafety — raises an exception on fatal signals, instead of causing an immediate uncontrolled exit. This is connected to the produtil.locking module to work around bugs in Lustre, Panasas and GPFS file locking.
- produtil.rusage — monitor and limit process resource usage
- produtil.batchsystem — Query information about the batch system and current batch job.
- produtil.cluster — Query information about the cluster.

Namespaces

- acl

  Manipulates Access Control Lists (ACL)
- atparse

  ATParser is a text parser that replaces strings with variables and function output.
- batchsystem

  Provides information about the batch system.
- cd

  Change directory, handle temporary directories.
- cluster

  Provides information about the cluster on which this job is running.
- config

  Parses UNIX conf files and makes the result readily available.
- datastore

  Stores products and tasks in an sqlite3 database file.
- dbnalert

  This module runs the NCO dbn_alert program, or logs dbn_alert messages if run with dbn alerts disabled.
- fileop

  This module provides a set of utility functions to do filesystem operations.
- listing

  Contains the Listing class, which emulates "ls -l".
- locking

  Handles file locking using Python "with" blocks.
- log

  Configures logging.
- mpi_impl

  Converts a group of MPI ranks to a runnable command.
- mpiprog

  Object structure for describing MPI programs.
- numerics

  Time manipulation and other numerical routines.
- pipeline

  Internal module that launches and monitors processes.
- prog

  Implements the produtil.run: provides the object tree for representing shell commands.

- retry
  Contains retry_io() which automates retrying operations.
- rstprod
  Handles data restriction classes.
- run
  A shell-like syntax for running serial, MPI and OpenMP programs.
- rusage
  This module allows querying resource usage and limits, as well as setting resource limits.
- setup
  Contains setup(), which initializes the produtil package.
- sigsafety
  Sets up signal handlers to ensure a clean exit.
- tempdir
  This module is an alias for produtil.cd, for backward compatibility.
- workpool
  Contains the WorkPool class, which maintains pools of threads that perform small tasks.

Variables

- string **version** = '4.0'

20.11 produtil.acl Namespace Reference

Manipulates Access Control Lists (ACL)

20.11.1 Detailed Description

Manipulates Access Control Lists (ACL)

This module is a wrapper around the C libacl library, which provides support for POSIX Access Control Lists, as defined by the abandoned draft standard "IEEE 1003.1e draft 17". Only the widely-supported features are implemented. It is intended to be used with the Linux libacl, but might be portable to other versions if the module-scope acl_library variable is changed to the name of your "dll" or "so" file for libacl and values of ACL_TYPE_ACC↩ESS and ACL_TYPE_DEFAULT are changed. In addition, one must change the means by which errno is accessed if switching from glibc to another C library.

Classes

- class ACL
  ACL class wrapped around the libacl library:
- class ACLCannotGet
  Raised when the libacl library could not get a file's ACL.
- class ACLCannotSet
  Raised when the libacl library could not set a file's ACL.
- class ACLCannotStringify
  Raised when libacl cannot convert an ACL to text.
- class ACLError
  Superclass of any ACL errors.
- class ACLLibraryError
  Raised when the libacl library could not be loaded.
- class ACLMissingError
  Raised when a function that requires an ACL object received None, or an invalid ACL.

Functions

- def load_libc ()

  Library loading routine:
- def load_libacl ()

  Loads the libacl library.
- def acl_to_text (acl)

  Returns a string representation of the given access control list object.
- def acl_get_file (filename, access=ACL_TYPE_ACCESS)

  Returns an object that represents the access control list for the specified file.
- def acl_get_fd (fd)

  Returns an object that represents the access control list for an open file descriptor.
- def acl_set_file (filename, acl, access=ACL_TYPE_ACCESS)

  Sets the named file's access control list.
- def acl_set_fd (fd, acl)

  Given an open file descriptor, sets the corresponding file's access control list.
- def acl_from_text (txt)

  Converts text to an access control list.
- def copy_acl_fd (fromfd, tofd)

  Simplified wrappers that perform common tasks:

Variables

- libacl = None

  The loaded libacl library from ctypes.cdll.LoadLibrary.
- libc = None

  The loaded libc library from ctypes.cdll.LoadLibrary.
- int ACL_TYPE_ACCESS = 32768

  The ACL_TYPE for Access Control Lists, defined in the libacl header files.
- int ACL_TYPE_DEFAULT = 16384

  The ACL_TYPE for Default Access Control Lists defined in the libacl header files.
- string acl_library = 'libacl.so.1'

  The ACL library name or path for input to ctypes.cdll.LoadLibrary.
- string c_library = 'libc.so.6'

  The C library name for input to ctypes.cdll.LoadLibrary.
- get_errno = None

  Function that returns the value of errno.

20.11.2 Function Documentation

20.11.2.1 acl_from_text()

```
def produtil.acl.acl_from_text (
              txt )
```

Converts text to an access control list.

Parameters

txt	a text access control list

Returns

a new ACL object

Definition at line 335 of file acl.py.

Referenced by produtil.acl.ACL.from_text().

20.11.2.2 acl_get_fd()

```
def produtil.acl.acl_get_fd (
              fd )
```

Returns an object that represents the access control list for an open file descriptor.

Parameters

fd	the integer file descriptor or open file object

Returns

a new ACL object

Definition at line 312 of file acl.py.

Referenced by produtil.acl.ACL.from_fd().

20.11.2.3 acl_get_file()

```
def produtil.acl.acl_get_file (
              filename,
              access = ACL_TYPE_ACCESS )
```

Returns an object that represents the access control list for the specified file.

Parameters

filename	the name of the file of interest
access	ACL_TYPE_ACCESS or ACL_TYPE_DEFAULT

Returns

a new ACL

Definition at line 304 of file acl.py.

Referenced by produtil.acl.ACL.from_file().

20.11.2.4 acl_set_fd()

```
def produtil.acl.acl_set_fd (
              fd,
              acl )
```

Given an open file descriptor, sets the corresponding file's access control list.

Parameters

fd	the file descriptor or file object
acl	the ACL object to change

Returns

acl

Definition at line 327 of file acl.py.

Referenced by produtil.acl.ACL.to_fd().

20.11.2.5 acl_set_file()

```
def produtil.acl.acl_set_file (
              filename,
              acl,
              access = ACL_TYPE_ACCESS )
```

Sets the named file's access control list.

Parameters

filename	the name of the file of interest
acl	the destination ACL object
access	ACL_TYPE_ACCESS or ACL_TYPE_DEFAULT

Returns

acl

Definition at line 319 of file acl.py.

Referenced by produtil.acl.ACL.to_file().

20.11.2.6 acl_to_text()

```
def produtil.acl.acl_to_text (
              acl )
```

Returns a string representation of the given access control list object.

Parameters

acl	an ACL object

Returns

the string equivalent

Definition at line 297 of file acl.py.

Referenced by produtil.acl.ACL.to_text().

20.11.2.7 copy_acl_fd()

```
def produtil.acl.copy_acl_fd (
              fromfd,
              tofd )
```

Simplified wrappers that perform common tasks:

Copy an access control list from one object to another

Copies a POSIX Access Control List (ACL) from one open file to another. The arguments should be either UNIX file descriptors, or the return values from open(). This routine is quicker than using the ACL() object due to avoidance of creating unnecessary Python objects. However, the access control list information is discarded in this routine, so it can only be used when the sole need is to copy the information from one file to another.

Parameters

fromfd	the source file descriptor
tofd	the target file descriptor

Definition at line 344 of file acl.py.

20.11.2.8 load_libacl()

```
def produtil.acl.load_libacl ( )
```

Loads the libacl library.

Loads the libacl library whose name is specified in the module scope acl_library variable. This function is called automatically when needed; you should never need to call it directly.

Definition at line 97 of file acl.py.

Referenced by produtil.acl.ACL.__init__(), and produtil.acl.copy_acl_fd().

20.11.2.9 load_libc()

```
def produtil.acl.load_libc ( )
```

Library loading routine:

Loads the libc library.

Loads the standard C library, which is needed to test the value of errno in order to report errors. This function is called automatically when needed; you should never need to call it directly.

Definition at line 76 of file acl.py.

Referenced by produtil.acl.load_libacl().

20.11.3 Variable Documentation

20.11.3.1 acl_library

```
produtil.acl.acl_library = 'libacl.so.1'
```

The ACL library name or path for input to ctypes.cdll.LoadLibrary.

This is intended to be modified externally from this module if needed before using the produtil.acl module.

Definition at line 60 of file acl.py.

20.11.3.2 ACL_TYPE_ACCESS

```
produtil.acl.ACL_TYPE_ACCESS = 32768
```

The ACL_TYPE for Access Control Lists, defined in the libacl header files.

This must match the value in the header.

Definition at line 49 of file acl.py.

20.11.3.3 ACL_TYPE_DEFAULT

```
produtil.acl.ACL_TYPE_DEFAULT = 16384
```

The ACL_TYPE for Default Access Control Lists defined in the libacl header files.

This must match the value in the header.

Definition at line 54 of file acl.py.

20.11.3.4 c_library

```
produtil.acl.c_library = 'libc.so.6'
```

The C library name for input to ctypes.cdll.LoadLibrary.

This is intended to be modified externally from this module if needed before using the produtil.acl module.

Definition at line 66 of file acl.py.

20.11.3.5 get_errno

```
produtil.acl.get_errno = None
```

Function that returns the value of errno.

Used for testing for errors in libacl routines.

Definition at line 71 of file acl.py.

Referenced by produtil.acl.copy_acl_fd(), produtil.acl.ACL.from_fd(), produtil.acl.ACL.from_file(), produtil.acl.AC↩
L.from_text(), produtil.acl.ACL.to_fd(), produtil.acl.ACL.to_file(), and produtil.acl.ACL.to_text().

20.11.3.6 libacl

```
produtil.acl.libacl = None
```

The loaded libacl library from ctypes.cdll.LoadLibrary.

Definition at line 40 of file acl.py.

20.11.3.7 libc

```
produtil.acl.libc = None
```

The loaded libc library from ctypes.cdll.LoadLibrary.

Definition at line 44 of file acl.py.

20.12 produtil.atparse Namespace Reference

ATParser is a text parser that replaces strings with variables and function output.

20.12.1 Detailed Description

ATParser is a text parser that replaces strings with variables and function output.

Classes

- class ATParser

  Takes input files or other data, and replaces certain strings with variables or functions.
- class NoSuchVariable

  Raised when a script requests an unknown variable.
- class ParserSyntaxError

  Raised when the parser encounters a syntax error.
- class ScriptAbort

  *Raised when an "@** abort" directive is reached in a script.*
- class ScriptAssertion

  Raised when a script @[VARNAME:?message] is encountered, and the variable does not exist.

Functions

- def replace_backslashed (text)

  Turns \t to tab, \n to end of line, \r to carriage return, \b to backspace and \(octal) to other characters.

Variables

- functions

  List of functions recognized.
- outer = dict(active=True,in_if_block=False,in_ifelse_block=False,used_if=False,ignore=False)

  Parser state for the portion of the file outside @[] and @∗∗ blocks.
- if_unused_if = dict(active=False,in_if_block=True,in_ifelse_block=False,used_if=False,ignore=False)

  @ var if_unused_if Parser state for within @∗∗if blocks that are inactive
- if_active_if = dict(active=True,in_if_block=True,in_ifelse_block=False,used_if=True,ignore=False)

  Parser state for within @∗∗ if blocks that are active.
- if_used_if = dict(active=False,in_if_block=True,in_ifelse_block=True,used_if=True,ignore=False)

  Parser state for after the end of an @∗∗ if block.
- if_active_else = dict(active=True,in_if_block=False,in_ifelse_block=True,used_if=True,ignore=False)

  Parser state for inside an "else" block.
- if_inactive_else = dict(active=False,in_if_block=False,in_ifelse_block=True,used_if=True,ignore=False)

  Parser state for inside an "else" block that was not used.
- ignore_if_block = dict(active=False,in_if_block=True,in_ifelse_block=False,used_if=False,ignore=True)

  Parser state for an "if" block that was skipped.
- ignore_else_block = dict(active=False,in_if_block=False,in_ifelse_block=True,used_if=False,ignore=True)

  Parser state for an "else" block that was skipped.

20.12.2 Function Documentation

20.12.2.1 replace_backslashed()

```
def produtil.atparse.replace_backslashed (
              text )
```

Turns \t to tab, \n to end of line, \r to carriage return, \b to backspace and \(octal) to other characters.

Parameters

text	the text to scan

Definition at line 47 of file atparse.py.

Referenced by produtil.atparse.ATParser.replace_vars().

20.12.3 Variable Documentation

20.12.3.1 functions

`produtil.atparse.functions`

Initial value:

```
1 = dict(lc=lambda x:str(x).lower(),
2                uc=lambda x:str(x).upper(),
3                len=lambda x:str(len(x)),
4                trim=lambda x:str(x).strip())
```

List of functions recognized.

Definition at line 9 of file atparse.py.

20.13 produtil.batchsystem Namespace Reference

Provides information about the batch system.

20.13.1 Detailed Description

Provides information about the batch system.

This module is intended to be used to communicate with the batch system. At present, it just knows how to guess the job name and id, as well as a "longname" that combines the two.

Classes

- class FakeClass

  A special class for constants.

Functions

- def set_default_name (default_name)

  Set default for all job names.
- def set_jobname (jobname)

  Sets the value that jobname() should return.
- def set_jobid (jobid)

  Sets the value that jobid() should return.
- def set_joblongname (joblongname)

  Sets the value that joblongname() should return.
- def getenvs (names, fallback=None)

  Get an environment variable, with various fallback options.
- def jobname (fallback=UNSPECIFIED)

  Get the batch job name.
- def jobid (fallback=UNSPECIFIED)

  Get the batch job ID.
- def joblongname (jobid_fallback=UNSPECIFIED, jobname_fallback=UNSPECIFIED)

  Get the job longname.

Variables

- UNSPECIFIED = FakeClass()

  Constant for unspecified arguments.
- string NONAME = "NONAME"

  Name for jobs that have no name.

20.13.2 Function Documentation

20.13.2.1 getenvs()

```
def produtil.batchsystem.getenvs (
              names,
              fallback = None )
```

Get an environment variable, with various fallback options.

Tries the list of environment variable names, returning the first one that exists and is non-blank. If none are found, returns the fallback.

Parameters

names	the list of environment variables
fallback	the fallback option if none are set

Definition at line 68 of file batchsystem.py.

Referenced by produtil.batchsystem.jobid(), produtil.batchsystem.joblongname(), and produtil.batchsystem.↩ jobname().

20.13.2.2 jobid()

```
def produtil.batchsystem.jobid (
              fallback = UNSPECIFIED )
```

Get the batch job ID.

Returns the batch system job id for the batch job that is running this program, if known. If set_jobid was called, returns its value. Otherwise, tries the NCO $pid first, then the various batch system environment variables. If none are found, and the fallback is specified, returns the fallback. Otherwise, returns "o$PID" where $PID is the process ID.

Parameters

fallback	the fallback if no id is known

Definition at line 103 of file batchsystem.py.

Referenced by produtil.batchsystem.joblongname().

20.13.2.3 joblongname()

```
def produtil.batchsystem.joblongname (
              jobid_fallback = UNSPECIFIED,
              jobname_fallback = UNSPECIFIED )
```

Get the job longname.

Returns a human-readable job name that includes both the batch system job name and id. If set_↩ joblongname was called, returns its value. Next, returns the NCO $jobid variable if available, otherwise returns LL{jobid()}.o{jobname()} where jobid and jobname are the results of those two functions. The jobid_fallback and jobname_fallback are passed as the fallback parameters to the calls to jobid and jobname.

Parameters

jobid_fallback	the fallback if no id is known
jobname_fallback	the fallback if no name is known

Definition at line 120 of file batchsystem.py.

Referenced by produtil.batchsystem.jobid().

20.13.2.4 jobname()

```
def produtil.batchsystem.jobname (
              fallback = UNSPECIFIED )
```

Get the batch job name.

Returns the human-readable job name, if one exists. If set_jobname was called, returns its value. Otherwise, attempts to get it from the NCO $job environment variable first, then tries the batch system variables. If none are found, and fallback is specified, then the fallback is returned. Otherwise, the module-level NONAME variable is returned (which defaults to "NONAME").

Parameters

fallback	return value if no job name is known

Definition at line 84 of file batchsystem.py.

Referenced by produtil.log.configureLogging(), and produtil.batchsystem.joblongname().

20.13.2.5 set_default_name()

```
def produtil.batchsystem.set_default_name (
              default_name )
```

Set default for all job names.

Sets a default value to use for the job name and long name if it cannot be guessed from the environment. This is used by produtil.setup.setup's jobname= argument. This will override the fallback= arguments of both jobname() and joblongname()

Parameters

default_name	the name

Definition at line 39 of file batchsystem.py.

Referenced by produtil.setup.setup().

20.13.2.6 set_jobid()

```
def produtil.batchsystem.set_jobid (
              jobid )
```

Sets the value that jobid() should return.

Parameters

jobid	the id

Definition at line 56 of file batchsystem.py.

20.13.2.7 set_joblongname()

```
def produtil.batchsystem.set_joblongname (
              joblongname )
```

Sets the value that joblongname() should return.

Parameters

joblongname	the new long name

Definition at line 62 of file batchsystem.py.

20.13.2.8 set_jobname()

```
def produtil.batchsystem.set_jobname (
               jobname )
```

Sets the value that jobname() should return.

Parameters

jobname	the name

Definition at line 50 of file batchsystem.py.

20.13.3 Variable Documentation

20.13.3.1 NONAME

```
produtil.batchsystem.NONAME = "NONAME"
```

Name for jobs that have no name.

Definition at line 21 of file batchsystem.py.

20.13.3.2 UNSPECIFIED

```
produtil.batchsystem.UNSPECIFIED = FakeClass()
```

Constant for unspecified arguments.

Definition at line 17 of file batchsystem.py.

20.14 produtil.cd Namespace Reference

Change directory, handle temporary directories.

20.14.1 Detailed Description

Change directory, handle temporary directories.

This module provides a means by which to change to a different directory in a Python "with" block and change back out afterwards, regardless of what happens inside the block. It can, optionally, create a new directory, and optionally delete it at the end of the block. There are two classes:

- TempDir - creates a temporary directory with a randomly-generated name, chdirs to the directory, and chdirs back out afterwards. It can be configured to delete the directory afterwards (the default) or not.
- NamedDir - a subclass of TempDir that uses a specific directory rather than a randomly-generated one. By default, the directory is NOT deleted at the end of the block. That can be configured.

Classes

- class NamedDir

  This subclass of TempDir takes a directory name, instead of generating one automatically.
- class TempDir

  This class is intended to be used with the Python "with TempDir() as t" syntax.

Variables

- perm_add = stat.S_IRUSR | stat.S_IWUSR | stat.S_IXUSR | \

  Default permissions to add to new directories created by TempDir: user has all possible access.
- perm_remove = stat.S_IWOTH|stat.S_ISUID

  Permissions to remove from all directories: world write and setuid.

20.14.2 Variable Documentation

20.14.2.1 perm_add

```
produtil.cd.perm_add = stat.S_IRUSR | stat.S_IWUSR | stat.S_IXUSR | \
```

Default permissions to add to new directories created by TempDir: user has all possible access.

Group and other can read and execute.

Definition at line 29 of file cd.py.

20.14.2.2 perm_remove

```
produtil.cd.perm_remove = stat.S_IWOTH|stat.S_ISUID
```

Permissions to remove from all directories: world write and setuid.

This overrides perm_add.

Definition at line 36 of file cd.py.

20.15 produtil.cluster Namespace Reference

Provides information about the cluster on which this job is running.

20.15.1 Detailed Description

Provides information about the cluster on which this job is running.

Classes

- class Cluster

  Stores information about a computer cluster.
- class NOAAGAEA

  Represents the NOAA GAEA cluster.
- class NOAAJet

  The NOAA Jet Cluster.
- class NOAATheia
- class NOAAWCOSS

  Represents the NOAA WCOSS clusters, Tide, Gyre and the test system Eddy.
- class NOAAZeus

  Represents the NOAA Zeus cluster.
- class UCARYellowstone

  Represents the Yellowstone cluster.
- class WCOSSCray

  This subclass of NOAAWCOSS handles the new Cray portions of WCOSS: Luna and Surge.
- class WisconsinS4

  Represents the S4 cluster.

Functions

- def set_cluster (there)

  Sets the current cluster (module-level "here" variable) to the given value.
- def where ()

  Guesses what cluster the program is running on, and if it cannot, returns a cluster named "noname" with reasonable defaults.
- def longname ()

  Synonym for here.longname.
- def name ()

  Synonym for here.name.
- def group_quotas ()

  Synonym for here.group_quotas.
- def acl_support ()

  Synonym for here.acl_support.
- def no_access_control ()

  True if the cluster provides no means to control access to files.
- def use_acl_for_rstdata ()

  Synonym for here.use_acl_for_rstdata.
- def ncepprod ()

  Are we on NCEP production?
- def partition ()

  Returns system-specific information about what part of the system you are on.

Variables

- DO_NOT_SET = object()

  Special values for parameters that should not be set.
- here = None

  The Cluster object for the local cluster.

20.15.2 Function Documentation

20.15.2.1 acl_support()

```
def produtil.cluster.acl_support ( )
```

Synonym for here.acl_support.

Will call the "where()" function if "here" is uninitialized.

Definition at line 130 of file cluster.py.

20.15.2.2 group_quotas()

```
def produtil.cluster.group_quotas ( )
```

Synonym for here.group_quotas.

Will call the "where()" function if "here" is uninitialized.

Definition at line 124 of file cluster.py.

Referenced by produtil.fileop.deliver_file().

20.15.2.3 longname()

```
def produtil.cluster.longname ( )
```

Synonym for here.longname.

Will call the "where()" function if "here" is uninitialized.

Definition at line 112 of file cluster.py.

20.15.2.4 name()

```
def produtil.cluster.name ( )
```

Synonym for here.name.

Will call the "where()" function if "here" is uninitialized.

Definition at line 118 of file cluster.py.

20.15.2.5 ncepprod()

```
def produtil.cluster.ncepprod ( )
```

Are we on NCEP production?

Returns

True if the present machine is the NCEP production machine. Note that this function may read a text file when it is called, and the return value may change during the execution of the program if the program is running during a production switch.

Definition at line 149 of file cluster.py.

20.15.2.6 no_access_control()

```
def produtil.cluster.no_access_control ( )
```

True if the cluster provides no means to control access to files.

This is true if the cluster uses group ids for quotas, and provides no access control list support.

Definition at line 136 of file cluster.py.

Referenced by produtil.rstprod.RestrictionClass.__init__().

20.15.2.7 partition()

```
def produtil.cluster.partition ( )
```

Returns system-specific information about what part of the system you are on.

Definition at line 159 of file cluster.py.

20.15.2.8 set_cluster()

```
def produtil.cluster.set_cluster (
              there )
```

Sets the current cluster (module-level "here" variable) to the given value.

Bad things may happen if this is not a subclass of Cluster. #

Parameters

there	A Cluster object for this local cluster.

Definition at line 72 of file cluster.py.

Referenced by produtil.setup.setup().

20.15.2.9 use_acl_for_rstdata()

```
def produtil.cluster.use_acl_for_rstdata ( )
```

Synonym for here.use_acl_for_rstdata.

Will call the "where()" function if "here" is uninitialized.

Definition at line 143 of file cluster.py.

Referenced by produtil.rstprod.RestrictionClass.__init__(), and produtil.fileop.deliver_file().

20.15.2.10 where()

```
def produtil.cluster.where ( )
```

Guesses what cluster the program is running on, and if it cannot, returns a cluster named "noname" with reasonable defaults.

The result is stored in the module scope "here" variable.

Definition at line 80 of file cluster.py.

Referenced by produtil.cluster.acl_support(), produtil.cluster.group_quotas(), produtil.cluster.longname(), produtil.cluster.name(), produtil.cluster.ncepprod(), produtil.cluster.no_access_control(), produtil.cluster.partition(), produtil.setup.setup(), and produtil.cluster.use_acl_for_rstdata().

20.15.3 Variable Documentation

20.15.3.1 DO_NOT_SET

```
produtil.cluster.DO_NOT_SET = object()
```

Special values for parameters that should not be set.

Definition at line 13 of file cluster.py.

20.15.3.2 here

```
produtil.cluster.here = None
```

The Cluster object for the local cluster.

Do not modify.

Definition at line 70 of file cluster.py.

20.16 produtil.config Namespace Reference

Parses UNIX conf files and makes the result readily available.

20.16.1 Detailed Description

Parses UNIX conf files and makes the result readily available.

The produtil.config module reads configuration information for a production system from one or more *.conf files, via the Python ConfigParser module. This module also automatically fills in certain information, such as fields calculated from the tcvitals or date. The result is accessible via the ProdConfig class, which provides many ways of automatically accessing configuration options.

Classes

- class ConfFormatter

  Internal class that implements ProdConfig.strinterp()
- class ConfTimeFormatter

  internal function that implements time formatting
- class DuplicateTaskName

  Raised when more than one task is registered with the same name in an ProdConfig object.
- class Environment

  returns environment variables, allowing substitutions
- class ProdConfig

  a class that contains configuration information
- class ProdTask

  A subclass of produtil.datastore.Task that provides a variety of convenience functions related to unix conf files and logging.

Functions

- def qparse (format_string)

  Replacement for Formatter.parse which can be added to Formatter objects to turn {'...'} and {"..."} blocks into literal strings (the ...
- def confwalker (conf, start, selector, acceptor, recursevar)

  walks through a ConfigParser-like object performing some action
- def from_file (filename, quoted_literals=False)

  Reads the specified conf file into an ProdConfig object.
- def from_string (confstr, quoted_literals=False)

  Reads the given string as if it was a conf file into an ProdConfig object.

Variables

- **UNSPECIFIED** = object()
- ENVIRONMENT = Environment()

  an Environment object.
- dictionary FCST_KEYS

  the list of forecast time keys recognized by ConfTimeFormatter
- dictionary ANL_KEYS

  the list of analysis time keys recognized by ConfTimeFormatter
- dictionary **ANL_M6_KEYS**
- dictionary **ANL_P6_KEYS**
- TIME_DIFF_KEYS = set(['fahr','famin','fahrmin'])

  the list of "forecast time minus analysis time" keys recognized by ConfTimeFormatter
- NOTFOUND = object()

  a special constant that represents a key not being found

20.16.2 Function Documentation

20.16.2.1 confwalker()

```
def produtil.config.confwalker (
                conf,
                start,
                selector,
                acceptor,
                recursevar )
```

walks through a ConfigParser-like object performing some action

Recurses through a ConfigParser-like object "conf" starting at section "start", performing a specified action. The special variable whose name is in recursevar specifies a list of additional sections to recurse into. No section will be processed more than once, and sections are processed in breadth-first order. For each variable seen in each section (including recursevar), this will call selector(sectionname, varname) to see if the variable should be processed. If selector returns True, then acceptor(section, varname, value) will be called.

Parameters

conf	the ConfigParser-like object
start	the starting section
selector	a function selector(section,option) that decides if an option needs processing (True) or not (False)
acceptor	a function acceptor(section,option,value) run on all options for which the selector returns True
recursevar	an option in each section that lists more sections the confwalker should touch. If the selector returns True for the recursevar, then the recursevar will be sent to the acceptor. However, it will be scanned for sections to recurse into even if the selector rejects it.

Definition at line 432 of file config.py.

20.16.2.2 from_file()

```
def produtil.config.from_file (
              filename,
              quoted_literals = False )
```

Reads the specified conf file into an ProdConfig object.

Creates a new ProdConfig object and instructs it to read the specified file.

Parameters

filename	the path to the file that is to be read

Returns

a new ProdConfig object

Definition at line 474 of file config.py.

20.16.2.3 from_string()

```
def produtil.config.from_string (
              confstr,
              quoted_literals = False )
```

Reads the given string as if it was a conf file into an ProdConfig object.

Creates a new ProdConfig object and reads the string data into it as if it was a config file

Parameters

confstr	the config data

Returns

a new ProdConfig object

Definition at line 486 of file config.py.

20.16.2.4 qparse()

```
def produtil.config.qparse (
              format_string )
```

Replacement for Formatter.parse which can be added to Formatter objects to turn {'...'} and {"..."} blocks into literal strings (the ...

part). Apply this by doing f=Formatter() ; f.parse=qparse.

Definition at line 176 of file config.py.

20.16.3 Variable Documentation

20.16.3.1 ANL_KEYS

produtil.config.ANL_KEYS

Initial value:

```
1 = { 'aYMDHM':'%Y%m%d%H%M', 'aYMDH':'%Y%m%d%H', 'aYMD':'%Y%m%d',
2            'ayear':'%Y', 'aYYYY':'%Y', 'aYY':'%y', 'aCC':'%C', 'acen':'%C',
3            'amonth':'%m', 'aMM':'%m', 'aday':'%d', 'aDD':'%d', 'ahour':'%H',
4            'acyc':'%H', 'aHH':'%H', 'aminute':'%M', 'amin':'%M' }
```

the list of analysis time keys recognized by ConfTimeFormatter

Definition at line 255 of file config.py.

20.16.3.2 ANL_M6_KEYS

dictionary produtil.config.ANL_M6_KEYS

Initial value:

```
1 = { 'am6YMDHM':'%Y%m%d%H%M', 'am6YMDH':'%Y%m%d%H', 'am6YMD':'%Y%m%d',
2            'am6year':'%Y', 'am6YYYY':'%Y', 'am6YY':'%y', 'am6CC':'%C', 'am6cen':'%C',
3            'am6month':'%m', 'am6MM':'%m', 'am6day':'%d', 'am6DD':'%d', 'am6hour':'%H',
4            'am6cyc':'%H', 'am6HH':'%H', 'am6minute':'%M', 'am6min':'%M' }
```

Definition at line 267 of file config.py.

20.16.3.3 ANL_P6_KEYS

dictionary produtil.config.ANL_P6_KEYS

Initial value:

```
1 = { 'ap6YMDHM':'%Y%m%d%H%M', 'ap6YMDH':'%Y%m%d%H', 'ap6YMD':'%Y%m%d',
2            'ap6year':'%Y', 'ap6YYYY':'%Y', 'ap6YY':'%y', 'ap6CC':'%C', 'ap6cen':'%C',
3            'ap6month':'%m', 'ap6MM':'%m', 'ap6day':'%d', 'ap6DD':'%d', 'ap6hour':'%H',
4            'ap6cyc':'%H', 'ap6HH':'%H', 'ap6minute':'%M', 'ap6min':'%M' }
```

Definition at line 279 of file config.py.

20.16.3.4 ENVIRONMENT

```
produtil.config.ENVIRONMENT = Environment()
```

an Environment object.

You should never need to instantiate another one.

Definition at line 66 of file config.py.

20.16.3.5 FCST_KEYS

```
produtil.config.FCST_KEYS
```

Initial value:

```
1 = { 'fYMDHM':'%Y%m%d%H%M', 'fYMDH':'%Y%m%d%H', 'fYMD':'%Y%m%d',
2             'fyear':'%Y', 'fYYYY':'%Y', 'fYY':'%y', 'fCC':'%C', 'fcen':'%C',
3             'fmonth':'%m', 'fMM':'%m', 'fday':'%d', 'fDD':'%d', 'fhour':'%H',
4             'fcyc':'%H', 'fHH':'%H', 'fminute':'%M', 'fmin':'%M' }
```

the list of forecast time keys recognized by ConfTimeFormatter

Definition at line 243 of file config.py.

20.17 produtil.datastore Namespace Reference

Stores products and tasks in an sqlite3 database file.

20.17.1 Detailed Description

Stores products and tasks in an sqlite3 database file.

This module maintains an sqlite3 database file that stores information about Products and Tasks. A Product is a file or group of files created by some Task. Both Product and Task classes derive from Datum, which is the base class of anything that can be stored in the Datastore.

Classes

- class CallbackExceptions

  Exception raised when a Product class encounters exceptions while calling its callback functions in Product.call_callbacks.
- class Datastore

  Stores information about Datum objects in a database.
- class Datum

  Superclass of anything that can be stored in a Datastore.
- class DatumException

  Superclass of all exceptions local to produtil.datastore.
- class DatumLockHeld

  Raised when a LockDatum is held by another Worker.
- class FakeException

  This is a fake exception used to get a stack trace.
- class FileProduct

  A subclass of Product that represents file delivery.
- class InvalidID

  Raised when a Datum or subclass receives a prodname or category name that is invalid.
- class InvalidOperation

  Raised when an invalid Datum operation is requested, such as delivering an UpstreamProduct.
- class Product

  A piece of data produced by a Task.
- class Task

  Represents a process or actor that makes a Product.
- class Transaction

  Datastore transaction support.
- class UnknownLocation

  Raised when delivering data, but no location is provided.
- class UpstreamFile

  Represents a Product created by an external workflow.

Functions

- def wait_for_products (plist, logger, renamer=None, action=None, renamer_args=None, action_args=None, sleeptime=20, maxtime=1800)

  Waits for products to be available and performs an action on them.

Variables

- string TASK_CATEGORY = '**task**'

  Special product category used for Tasks.
- int FAILED = -10

  Constant used for Task.state to indicate a run was attempted but failed.
- int UNSTARTED = 0

  Constant used for Task.state to indicate no attempt was made to run.
- int RUNNING = 10

  Constant used for Task.state to indicate the task is presently running.
- int PARTIAL = 20

  Constant used for Task.state to indicate the task was attempted but exited prematurely.
- int COMPLETED = 30

  Constant used for Task.state to indicate the task completed successfully.

20.17.2 Function Documentation

20.17.2.1 wait_for_products()

```
def produtil.datastore.wait_for_products (
              plist,
              logger,
              renamer = None,
              action = None,
              renamer_args = None,
              action_args = None,
              sleeptime = 20,
              maxtime = 1800 )
```

Waits for products to be available and performs an action on them.

Waits for a specified list of products to be available, and performs some action on each product when it becomes available. Sleeps sleeptime seconds between checks. Returns the number of products that were found before the maxtime was reached.

Parameters

plist	A Product or a list of Product objects.
logger	A logging.Logger object in which to log messages.
renamer	Optional: a function or callable object that provides a new name for each product. This is passed the product, the logger and the contents of ∗renamer_args. Default: os.path.basename(p.location)
action	Optional: an action to perform on each product. This is passed the product, the output of renamer, the logger and the contents of ∗action_args. Default: perform no action.
renamer_args	Optional: arguments to renamer.
action_args	Optional: arguments to action.
sleeptime	- after checking availability of all products, if at least one is unavailable, the code will sleep for this much time before rechecking. Will be overridden by 0.01 if it is set to something lower than that. Default: 20
maxtime	- maximum amount of time to spend in this routine before giving up.

Returns

the number of products that became available before the maximum wait time was hit.

Definition at line 979 of file datastore.py.

Referenced by produtil.datastore.UpstreamFile.deliver().

20.17.3 Variable Documentation

20.17.3.1 COMPLETED

```
produtil.datastore.COMPLETED = 30
```

Constant used for Task.state to indicate the task completed successfully.

Definition at line 130 of file datastore.py.

20.17.3.2 FAILED

```
produtil.datastore.FAILED = -10
```

Constant used for Task.state to indicate a run was attempted but failed.

Definition at line 101 of file datastore.py.

20.17.3.3 PARTIAL

```
produtil.datastore.PARTIAL = 20
```

Constant used for Task.state to indicate the task was attempted but exited prematurely.

Practically speaking, there is no way to tell the difference between RUNNING and PARTIAL since the job cannot ensure that it resets the state before unexpectedly exiting.

Definition at line 122 of file datastore.py.

20.17.3.4 RUNNING

```
produtil.datastore.RUNNING = 10
```

Constant used for Task.state to indicate the task is presently running.

Definition at line 113 of file datastore.py.

20.17.3.5 TASK_CATEGORY

```
produtil.datastore.TASK_CATEGORY = '**task**'
```

Special product category used for Tasks.

Definition at line 95 of file datastore.py.

20.17.3.6 UNSTARTED

```
produtil.datastore.UNSTARTED = 0
```

Constant used for Task.state to indicate no attempt was made to run.

Definition at line 107 of file datastore.py.

20.18 produtil.dbnalert Namespace Reference

This module runs the NCO dbn_alert program, or logs dbn_alert messages if run with dbn alerts disabled.

20.18.1 Detailed Description

This module runs the NCO dbn_alert program, or logs dbn_alert messages if run with dbn alerts disabled.

Classes

- class DBNAlert

  This class represents a call to dbn_alert, as a callable Python object.

Functions

- def find_dbn_alert ()

  Locates the dbn_alert executable based on environment variables, and returns it as a produtil.prog.Runner object.
- def init_logging (logger=None)

  Initializes logging for this module.
- def init_jobstring (jobname=None)

  Sets the job string (for dbn_alerts) to the specified value, or if unspecified, tries to guess one from the environment.
- def init_dbn_alert (send_dbn=None)

  DBN alert initialization helper function.
- def init_module (logger=None, jobname=None, send_dbn=None)

  Call to initialize this module.

Variables

- log = None

  logging.Logger object to send dbnalert messages
- job = None

  a string representing this job (from os.environ['job'] by default)
- bool send_dbn_alerts = True

  False = don't run dbn_alert.
- bool no_DBNROOT_warn = False

  True = I have already warned that $DBNROOT is unset.

20.18.2 Function Documentation

20.18.2.1 find_dbn_alert()

```
def produtil.dbnalert.find_dbn_alert ( )
```

Locates the dbn_alert executable based on environment variables, and returns it as a produtil.prog.Runner object.

Definition at line 32 of file dbnalert.py.

20.18.2.2 init_dbn_alert()

```
def produtil.dbnalert.init_dbn_alert (
              send_dbn = None )  [protected]
```

DBN alert initialization helper function.

This is part of the implementation of init_module: it decides whether to send DBNet alerts, and sets the module-scope send_dbn_alerts variable accordingly. That will then be used by DBNAlert objects to decide whether to actually call the dbn_alert program.

Parameters

send_dbn	Do we send dbn alerts?

Definition at line 147 of file dbnalert.py.

Referenced by produtil.dbnalert.init_module().

20.18.2.3 init_jobstring()

```
def produtil.dbnalert.init_jobstring (
              jobname = None )
```

Sets the job string (for dbn_alerts) to the specified value, or if unspecified, tries to guess one from the environment.

Definition at line 114 of file dbnalert.py.

Referenced by produtil.dbnalert.init_module().

20.18.2.4 init_logging()

```
def produtil.dbnalert.init_logging (
               logger = None )
```

Initializes logging for this module.

The argument is either a logging.Logger to log to, or the string name of the logging domain.

Definition at line 102 of file dbnalert.py.

Referenced by produtil.dbnalert.init_module().

20.18.2.5 init_module()

```
def produtil.dbnalert.init_module (
               logger = None,
               jobname = None,
               send_dbn = None )
```

Call to initialize this module.

Initializes the logging and job string for this module.

Parameters

logger	Either a logging.Logger object to receive log messages, or the string name of a logger domain.
jobname	The dbn_alert job string for this job.
send_dbn	Optional. If specified, this controls whether dbn_alert is actually run (True) or not (False). If unspecified, then the SENDDBN environment variable is used.

Definition at line 194 of file dbnalert.py.

Referenced by produtil.setup.setup().

20.18.3 Variable Documentation

20.18.3.1 send_dbn_alerts

```
produtil.dbnalert.send_dbn_alerts = True
```

False = don't run dbn_alert.

Just log the alerts.

Definition at line 26 of file dbnalert.py.

20.19 produtil.fileop Namespace Reference

This module provides a set of utility functions to do filesystem operations.

20.19.1 Detailed Description

This module provides a set of utility functions to do filesystem operations.

It replaces or improves upon several os, stat, and sys module functions by working around Python bugs, providing an API layer that allows forward compatibility to future Python versions, and adding logging capabilities.

Classes

- class CannotFindExe

  Thrown when find_exe cannot find an executable in the path or directory list.
- class CannotLinkMulti

  This exception is raised when the caller tries to create multiple symlinks in a single target, but the target is not a directory.
- class DeliveryFailed

  This exception is raised when a file cannot be delivered.
- class FileOpError

  This is the superclass of several exceptions relating to multi-file operations in produtil.fileop.
- class FileOpErrors

  This exception is raised when an operation that processes multiple files catches more than one exception.
- class FileWaiter

  A class that waits for files to meet some requirements.
- class FindExeInvalidExeName

  Thrown when find_exe is given an executable name that contains a directory path.
- class InvalidExecutable

  Thrown when a find_exe fails.
- class RelativePathError

  Raised when a relative path is given, but an absolute path is expected.
- class UnexpectedAbsolutePath

  This exception indicates that the renamer function sent to make_symlinks_in returned an absolute path.
- class VerificationFailed

  This exception is raised when a copy of a file has different content than the original.
- class WrongSymlink

  Raised when os.symlink makes a symlink to a target other than the one that was requested.

Functions

- def realcwd ()

  Returns the current working directory, expanding any symbolic links.
- def chdir (path, logger=None)

  Changes to the specified directory.
- def touch (filename, times=None)

  Open the file for append and set mtime and atime.
- def netcdfver (filename)

  What is the NetCDF version of this file?
- def gribver (filename)

  What is the GRIB version of this file?
- def makedirs (filename, numtries=10, logger=None)

  Make a directory tree, working around filesystem bugs.
- def remove_file (filename, info=True, logger=None)

  Deletes the specified file.
- def rmall (args, kwargs)

  Deletes the specified list of files.
- def lstat_stat (filename, raise_nonexist=False)

  Runs lstat and stat on a file as efficiently as possible.
- def isnonempty (filename)

  Returns True if the filename refers to an existent file that is non-empty, and False otherwise.
- def deliver_file (infile, outfile, keep=True, verify=False, blocksize=1048576, tempprefix=None, permmask=002, removefailed=True, logger=None, preserve_perms=True, preserve_times=True, preserve_group=None, copy_acl=None, moveok=True, force=True, copier=None)

  This moves or copies the file "infile" to "outfile" in a unit operation; outfile will never be seen in an incomplete state.
- def find_exe (name, dirlist=None, raise_missing=True)

  Searches the $PATH or a specified iterable of directory names to find an executable file with the given name.
- def symlink_read_test (filename, readsize=40, logger=None)

  Opens the specified file for reading and attempts to read data to it.
- def make_symlinks_in (sources, targetdir, force=False, renamer=None, logger=None, copy=False)

  Creates symbolic links from a set of source files to a target directory.
- def make_symlink (source, target, force=False, logger=None, max_tries=20)

  Creates a symbolic link "target" that points to "source".
- def replace_symlink (source, target, logger=None, max_tries=20)

  Do not call this routine directly: you want make_symlink instead.
- def unblock (stream, logger=None)

  Attempts to modify the given stream to be non-blocking.
- def call_fcntrl (stream, on, off, logger=None)

  Internal function that implements unblock()
- def fortlink (forts, force=False, basedir=None, logger=None)

  This is a convenience routine that makes many symbolic links to fort.N files for various integers N using make_symlink.
- def fortcopy (forts, basedir=None, logger=None, only_log_errors=False, kwargs)

  A convenience function for copying files to local fort.N files for various integers N using deliver_file(...,keep=True).
- def norm_expand_path (path=None, fullnorm=False)

  Normalizes path and expand home directories.
- def norm_abs_path (rel, fromdir=None)

  Return relative path.
- def check_last_lines (filename, searchstr, lastbytes=10000, logger=None)

  Checks the last few bytes of a file to see if the specified search string is present.

- def check_file (filename, min_size=None, min_mtime_age=None, min_atime_age=None, min_ctime_age=None, logger=None)

  Determines whether the specified file exists, and meets additional requirements.
- def wait_for_files (flist, logger=None, maxwait=1800, sleeptime=20, min_size=1, min_mtime_age=30, min_atime_age=None, min_ctime_age=None, min_fraction=1.0, log_each_file=True)

  Waits for files to meet requirements.

Variables

- **module_logger** = logging.getLogger('produtil.fileop')

20.19.2 Function Documentation

20.19.2.1 call_fcntrl()

```
def produtil.fileop.call_fcntrl (
              stream,
              on,
              off,
              logger = None )
```

Internal function that implements unblock()

Parameters

stream	the stream to modify
on	flags to turn on
off	flags to turn off
logger	a logging.Logger for messages

Returns

True on success, False otherwise.

Definition at line 751 of file fileop.py.

Referenced by produtil.fileop.unblock().

20.19.2.2 chdir()

```
def produtil.fileop.chdir (
              path,
              logger = None )
```

Changes to the specified directory.

Please use produtil.cd.NamedDir instead.

This is generally not a good idea since you will not cd back if an unhandled exception is raised. It is better to use the produtil.cd module, which provides ways to enter a directory in a "with" block and optionally delete it afterwards. Such functionality could also be implemented via a try...finally block.

Parameters

path	the path to cd to
logger	a logging.Logger for log messages

Definition at line 144 of file fileop.py.

20.19.2.3 check_file()

```
def produtil.fileop.check_file (
               filename,
               min_size = None,
               min_mtime_age = None,
               min_atime_age = None,
               min_ctime_age = None,
               logger = None )
```

Determines whether the specified file exists, and meets additional requirements.

Parameters

filename	The file to analyze.
min_size	If present, the file must be at least this many bytes.
min_mtime_age	If specified, the file must have been modified more than this many seconds in the past.
min_atime_age	if specified, the file atime must be at least this many seconds old. The meaning of atime varies, but usually means the last access time.
min_ctime_age	If specified, the file ctime must be at least this many seconds old. The meaning of ctime varies between platforms and file types, but usually means the file creation or inode change time. See stat(2) for details.
logger	a logging.Logger for log messages.

Note

This routine can also be used on directories, but one should avoid the min_size option when doing that.

Returns

True if requirements are met, False otherwise.

Definition at line 929 of file fileop.py.

Referenced by produtil.datastore.UpstreamFile.check(), produtil.fileop.FileWaiter.check(), and produtil.fileop.↩ check_last_lines().

20.19.2.4 check_last_lines()

```
def produtil.fileop.check_last_lines (
               filename,
               searchstr,
               lastbytes = 10000,
               logger = None )
```

Checks the last few bytes of a file to see if the specified search string is present.

Returns True if the string is present or False if the file existed but the string was not present. Will raise an exception if the file is non-existent or cannot be read.

Parameters

filename	The file to search (a string).
searchstr	The string to search for. Must not contain end-of-line chars.
lastbytes	The number of bytes at the end of the file to check. Can be larger than the file size.
logger	A logging.Logger for log messages.

Returns

True if the file contains the given string, False otherwise

Definition at line 895 of file fileop.py.

20.19.2.5 deliver_file()

```
def produtil.fileop.deliver_file (
               infile,
               outfile,
               keep = True,
               verify = False,
               blocksize = 1048576,
               tempprefix = None,
               permmask = 002,
               removefailed = True,
               logger = None,
               preserve_perms = True,
               preserve_times = True,
               preserve_group = None,
               copy_acl = None,
               moveok = True,
               force = True,
               copier = None )
```

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

If the caller specifies keep=False (default is True) and moveok=True, and the source and destination are on the same filesystem then the delivery is done with a simple move. Otherwise a copy is done to a temporary file on the same filesystem as the target. If verification is requested (verify=True) then the temporary file is verified by filecmp.cmp, before moving the temporary file to the final location.

When requested, and when possible, the permissions and ownership are preserved. Both copy_acl and preserve↩ _group have defaults set by the produtil.cluster module. If the cluster uses access control lists for data restriction classes, then copy_acl will be set to True, otherwise it is false. If group quotas are enabled, preserve_group is False, otherwise it is True.

Note

The original file is never deleted, but it may be moved to the target if keep=False. If a copy is done instead, the original file is still present.

Parameters

infile	the origin file
outfile	the destination file or its parent directory
keep	If False, the original file is no longer needed. If False and moveok=True, the file might be delivered by a "mv" operation, avoiding any data duplication (no "cp").
verify	If a "cp" is done, reopen the target and source and verify they are the same. Note that providing a copier will break the verification functionality if the copier changes the contents of the destination file (such as a copier that compresses).
blocksize	block size during copy operations
tempprefix	Prefix for temporary files during copy operations. Do not include directory paths in the tempprefix.
permmask	Permission bits to remove Default: world write (002)
removefailed	If True, delete temporary files if the delivery fails
logger	the logging.Logger for log messages.
preserve_perms	If True, copy the old file's permissions to the new file
preserve_times	If True, copy the old file's timestamps to the new file
preserve_group	If True, copy the old file's group ID to the new file
copy_acl	If True, copy the access control lists from one file to another
moveok	If True, delivery by "mv" is allowed. Must also set keep=False.
force	If False, delivery will be aborted (raise TargetFileExists) if the target file already exists.
copier	If present, this function or callable object is used to copy data from the source file to the temporary file before moving it to the target. The copier is called as copier(infile,temp_file_name,temp_file_object) Where the temp_file_name is the name of the destination file and the temp_file_object is an object that can be used to write to the file. The copier should NOT close the temp_file_object.

Definition at line 348 of file fileop.py.

Referenced by produtil.datastore.FileProduct.deliver(), produtil.fileop.fortcopy(), produtil.fileop.isnonempty(), and produtil.fileop.make_symlinks_in().

20.19.2.6 find_exe()

```
def produtil.fileop.find_exe (
                name,
                dirlist = None,
                raise_missing = True )
```

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

Returns the exectuable's location. If the executable cannot be found, and raise_missing=True, raises CannotFindExe, otherwise returns None. Raises FindExeInvalidExeName if "name" is not the same as its os.path.basename.

Parameters

name	The name of the executable to find.
dirlist	The list of directories to search, or None to search $PATH
raise_missing	If True, the CannotFindExe exception is raised for executables that cannot be found. If False, return None in that situation.

Definition at line 562 of file fileop.py.

Referenced by produtil.mpi_impl.mpirun_lsf.mpirunner(), produtil.mpi_impl.mpiexec_mpt.mpirunner(), produtil.mpi_impl.impi.mpirunner(), produtil.mpi_impl.mpiexec.mpirunner2(), produtil.mpi_impl.lsf_cray_intel.mpirunner2(), and produtil.mpi_impl.srun.mpirunner_impl().

20.19.2.7 fortcopy()

```
def produtil.fileop.fortcopy (
              forts,
              basedir = None,
              logger = None,
              only_log_errors = False,
              kwargs )
```

A convenience function for copying files to local fort.N files for various integers N using deliver_file(...,keep=True).

It works similarly to fortlink. The force= argument tells fortcopy to overwrite existing files. Otherwise, an exception will be raised if the destination file already exists. The optional basedir argument is the parent directory of the fort.N.

Call like this:

```
fortcopy({ 15:"/usr/local/share/file1",
           23:"./file2"})
```

And you will create files:

```
./fort.15 (copied from /usr/local/share/file1)
./fort.23 (copied from ./file2)
```

All other keyword arguments are sent to deliver_file.

Parameters

forts	Mapping from Fortran unit number to copy target.
basedir	Where to put the files instead of the current directory.
logger	A logging.Logger for log messages.
only_log_errors	Only log failed operations instead of logging everything.
kwargs	All other keyword arguments are passed to deliver_file()

Definition at line 815 of file fileop.py.

20.19.2.8 fortlink()

```
def produtil.fileop.fortlink (
              forts,
              force = False,
              basedir = None,
              logger = None )
```

This is a convenience routine that makes many symbolic links to fort.N files for various integers N using make_↩ symlink.

It works similarly to fortcopy. The optional basedir is the relative directory. The optional force argument is passed on to make_symlink and has the usual meaning: replace existing files.

Call like this:

```
fortlink({ 15:"/usr/local/share/file1",
           23:"./file2"})
```

And you will create these symbolic links:

```
./fort.15 -> /usr/local/share/file1
./fort.23 -> ./file2
```

as with other symlink routines in this module, set force=True to remove target fort.N files if they already exist.

Parameters

forts	Mapping from Fortran unit number to link target.
force	Remove target files if they exist.
basedir	Where to make the links instead of the current directory.
logger	A logging.Logger for log messages.

Definition at line 781 of file fileop.py.

20.19.2.9 gribver()

```
def produtil.fileop.gribver (
              filename )
```

What is the GRIB version of this file?

Returns the GRIB file version: 1 or 2. If the file is not a GRIB file, or if the answer is indeterminate, returns None. Only the first GRIB record is tested.

Parameters

filename	the path to the file to test

Definition at line 200 of file fileop.py.

20.19.2.10 isnonempty()

```
def produtil.fileop.isnonempty (
              filename )
```

Returns True if the filename refers to an existent file that is non-empty, and False otherwise.

Parameters

filename	The file to test.

Definition at line 332 of file fileop.py.

Referenced by produtil.config.ProdConfig.from_args(), and produtil.fileop.gribver().

20.19.2.11 lstat_stat()

```
def produtil.fileop.lstat_stat (
              filename,
              raise_nonexist = False )
```

Runs lstat and stat on a file as efficiently as possible.

Returns (lstat(filename),stat(filename)) where each is None if it fails due to non-existence. Does this in as few filesystem metadata operations as possible. Will raise an exception if the stat fails for any reason other than non-existence of a file, or if the file or linked file is non-existent and raise_nonexist=True.

Parameters

filename	The file to test.
raise_nonexist	Should we raise an exception if the file does not exist?

Returns

a tuple (L,S) where L is the lstat return value, and S is the stat return value. Each will be None if the file or link target do not exist.

Definition at line 306 of file fileop.py.

Referenced by produtil.fileop.deliver_file(), produtil.fileop.isnonempty(), and produtil.fileop.make_symlinks_in().

20.19.2.12 make_symlink()

```
def produtil.fileop.make_symlink (
               source,
               target,
               force = False,
               logger = None,
               max_tries = 20 )
```

Creates a symbolic link "target" that points to "source".

If the target already exists and is NOT a directory, then the file will be replaced. The replacement is done in a unit operation so that the target will always exist (unless the operation fails).

Parameters

source	The file to link to.
target	The name of the link.
force	If True, and target exists, delete it first.
logger	a logging.Logger for log messages.

Definition at line 666 of file fileop.py.

Referenced by produtil.fileop.fortlink(), and produtil.fileop.make_symlinks_in().

20.19.2.13 make_symlinks_in()

```
def produtil.fileop.make_symlinks_in (
               sources,
               targetdir,
               force = False,
               renamer = None,
               logger = None,
               copy = False )
```

Creates symbolic links from a set of source files to a target directory.

If "force" is True, then any existing files will first be deleted.

The "renamer" can be a function that generates paths of the symlinks, relative to targetdir, for each symlink in "sources". If the return value from "renamer" is an absolute path, an exception will be thrown. If the return value is None, then no link will be made.

Example: make_symlinks_in(['/path/to/a','/path/to/b'],'.', renamer=lambda s: os.path.basename(s)+'.linkified')

will create a.linkified, linked to /path/to/a, and b.linkified, linked to /path/to/b in directory "."

Parameters

sources	The list of files to link to.
targetdir	The directory in which to place the links.
force	Remove existing files if needed.
renamer	Function to generate link names.
logger	A logging.Logger for log messages.
copy	If True, files are copied instead of linked.

Definition at line 609 of file fileop.py.

Referenced by produtil.fileop.symlink_read_test().

20.19.2.14 makedirs()

```
def produtil.fileop.makedirs (
              filename,
              numtries = 10,
              logger = None )
```

Make a directory tree, working around filesystem bugs.

This makedirs implementation works around a common bug: if two processes try to recursively make a directory tree simultaneously, makedirs can fail when two processes make the same path component at the same time. This implementation automatically retries in that situation.

Parameters

filename	the directory path
numtries	the number of times to retry
logger	a logging.Logger for log messages

Definition at line 223 of file fileop.py.

Referenced by produtil.locking.LockFile.acquire_impl(), produtil.config.ProdConfig.makedirs(), tc_pairs_wrapper.↩ TcPairsWrapper.run_all_times(), and config_launcher.METplusLauncher.sanity_check().

20.19.2.15 netcdfver()

```
def produtil.fileop.netcdfver (
              filename )
```

What is the NetCDF version of this file?

Returns one of three strings based on the NetCDF version of the given file, or returns None if the file is not NetCDF:

- "CDF1" = NetCDF classic format
- "CDF2" = NetCDF 64-bit offset format
- "HDF5" = HDF5 file, and hence possibly a NetCDF4 file.
- None = Not NetCDF and not HDF5

Parameters

filename	the name of the file to test

Definition at line 176 of file fileop.py.

20.19.2.16 norm_abs_path()

```
def produtil.fileop.norm_abs_path (
              rel,
              fromdir = None )
```

Return relative path.

This routine generates relative file paths (using os.path.relpath) that are relative to the specified "from" directory fromdir. The fromdir will be first sent through norm_expand_path to eliminate system-specific weirdness, such as a/./b, a/../b, ∼username and so on. This will raise RelativePathError if the resulting path is not absolute.

Parameters

rel	the path
fromdir	the directory from which we want the relative path

Definition at line 881 of file fileop.py.

20.19.2.17 norm_expand_path()

```
def produtil.fileop.norm_expand_path (
              path = None,
              fullnorm = False )
```

Normalizes path and expand home directories.

Calls os.path.normpath and os.path.expanduser on its argument, or on os.getcwd() if no argument is supplied (or if path=None). This removes extraneous a/./b, a/../b, expands ∼username and ∼, and other system-specific expansions. See the Python documentation of normpath and expanduser for details. Will also call realpath and normcase if fullnorm=True. Raises RelativePathError if the resulting path is not absolute.

Parameters

path	the path to expand
fullnorm	If True, call os.path.normcase() and os.path.realpath() normapth and expanduser.

Definition at line 857 of file fileop.py.

Referenced by produtil.fileop.norm_abs_path().

20.19.2.18 realcwd()

```
def produtil.fileop.realcwd ( )
```

Returns the current working directory, expanding any symbolic links.

Definition at line 139 of file fileop.py.

20.19.2.19 remove_file()

```
def produtil.fileop.remove_file (
              filename,
              info = True,
              logger = None )
```

Deletes the specified file.

Does nothing if the filename is None, is the empty string or already does not exist. Otherwise, the file is deleted.

Parameters

filename	The file to delete.
info	Optional: indicates that warnings about a file already not existing should be sent to the logger at INFO level (info=True) instead of WARNING (info=False).
logger	the logging.Logger for messages

Definition at line 250 of file fileop.py.

Referenced by produtil.fileop.rmall(), and produtil.datastore.FileProduct.undeliver().

20.19.2.20 replace_symlink()

```
def produtil.fileop.replace_symlink (
              source,
              target,
              logger = None,
              max_tries = 20 )
```

Do not call this routine directly: you want make_symlink instead.

This routine creates a new symbolic link and renames that link to "target." That always replaces target with a symbolic link to source, even if target did not already exist.

Parameters

source	the file to link from
target	the file to link to
logger	a logging.Logger for messages

Definition at line 700 of file fileop.py.

Referenced by produtil.fileop.make_symlink().

20.19.2.21 rmall()

```
def produtil.fileop.rmall (
              args,
              kwargs )
```

Deletes the specified list of files.

Deletes files listed in "args". Each one is passed to remove_file. Exceptions that derive from EnvironmentError are collected, and will be raised at the end, thus allowing removal of later files to continue if earlier ones failed. If only one file causes an exception, that exception will be raised, otherwise FileOpErrors will be raised

Parameters

args	The files to delete.
kwargs	Keyword arguments passed to remove_file().

Definition at line 275 of file fileop.py.

20.19.2.22 symlink_read_test()

```
def produtil.fileop.symlink_read_test (
              filename,
              readsize = 40,
              logger = None )
```

Opens the specified file for reading and attempts to read data to it.

Logs the process. Will NOT raise any I/O or system errors; they are ignored. This is a workaround for a bug in Cray: symlinks to recently created files cannot be read by the compute node unless the batch node reads from them first (or unless you wait a while).

Definition at line 593 of file fileop.py.

20.19.2.23 touch()

```
def produtil.fileop.touch (
              filename,
              times = None )
```

Open the file for append and set mtime and atime.

Opens the specified file in append mode, but writes nothing. Sets the access and modification times.

Parameters

filename	the string filename
times	A 2-tuple of numbers, of the form (atime, mtime). These are UNIX epoch times (seconds since 1970 began in UTC).

Definition at line 163 of file fileop.py.

20.19.2.24 unblock()

```
def produtil.fileop.unblock (
              stream,
              logger = None )
```

Attempts to modify the given stream to be non-blocking.

This only works with streams that have an underlying POSIX fileno, such as those from open.

Will re-raise any exception received, other than AttributeError and EnvironmentError. Hence, I/O errors and attempts to make a non-fileno stream non-blocking will produce a False return value, while anything else will raise an exception.

Parameters

stream	the stream to unblock
logger	a logging.Logger for log messages

Returns

True on success, False otherwise.

Definition at line 736 of file fileop.py.

20.19.2.25 wait_for_files()

```
def produtil.fileop.wait_for_files (
              flist,
              logger = None,
              maxwait = 1800,
              sleeptime = 20,
              min_size = 1,
              min_mtime_age = 30,
              min_atime_age = None,
              min_ctime_age = None,
              min_fraction = 1.0,
              log_each_file = True )
```

Waits for files to meet requirements.

This is a simple wrapper around the FileWaiter class for convenience. It is equivalent to creating a FileWaiter with the provided arguments, and calling its checkfiles routine.

Parameters

flist	the file or list of files to wait for. This is simply sent into self.add.
logger	a logging.Logger for messages
maxwait	maximum seconds to wait
sleeptime	sleep time in seconds between checks
min_size	minimum file size
min_mtime_age	minimum modification time age,
min_atime_age	minimum access time age.
min_ctime_age	time since last file status change (see stat(2))
min_fraction	the minimum fraction of the provided files that must match the above requirements in order for FileWaiter.wait to return True. Default is 1.0, which means all of them.
log_each_file	log messages about each file checked

Definition at line 1136 of file fileop.py.

Referenced by produtil.fileop.FileWaiter.checkfiles().

20.20 produtil.listing Namespace Reference

Contains the Listing class, which emulates "ls -l".

20.20.1 Detailed Description

Contains the Listing class, which emulates "ls -l".

Classes

- class Listing

  Imitates the shell "ls -l" program.

20.21 produtil.locking Namespace Reference

Handles file locking using Python "with" blocks.

20.21.1 Detailed Description

Handles file locking using Python "with" blocks.

This module implements a Python with construct that can hold a lock and release it at the end of the "with" block. It also implements a safety feature to allow the program to disable locking, ensuring a fatal exception (LockingDisabled) if anything tries to lock a file. That functionality is connected to the produtil.sigsafety module, which will disable locking if a fatal signal is received.

```
import produtil.locking
with produtil.locking.LockFile("some.lockfile"):
    ... do things while the file is locked...
...  the file is now unlocked ...
```

Classes

- class LockFile

  Automates locking of a lockfile.
- class LockHeld

  This exception is raised when a LockFile cannot lock a file because another process or thread has locked it already.
- class LockingDisabled

  This exception is raised when a thread attempts to acquire a lock while Python is exiting according to produtil.sigsafety.

Functions

- def disable_locking ()

  Entirely disables all locking in this module.

Variables

- locks = set()

  Part of the internal implementation of this module: the list of existing locks (LockFile objects) that may be held.
- bool locks_okay = True

  Part of the internal implementation of this module: if True, locking is allowed, if False, locking is forbidden.

20.21.2 Function Documentation

20.21.2.1 disable_locking()

```
def produtil.locking.disable_locking ( )
```

Entirely disables all locking in this module.

If this is called, any locking attempts will raise LockingDisabled. That exception derives directly from BaseException, which well-written Python code will never catch, hence ensuring a rapid, abnormal exit of the program. This routine should never be called directly: it is only used as part of the implementation of the produtil.sigsafety, to prevent file locking after catching a terminal signal, hence allowing the program to exit as quickly as possible, and present a stack trace to any locations that attempt locking.

Definition at line 36 of file locking.py.

Referenced by produtil.sigsafety.hup_handler(), and produtil.sigsafety.term_handler().

20.21.3 Variable Documentation

20.21.3.1 locks

```
produtil.locking.locks = set()
```

Part of the internal implementation of this module: the list of existing locks (LockFile objects) that may be held.

Definition at line 28 of file locking.py.

20.21.3.2 locks_okay

```
produtil.locking.locks_okay = True
```

Part of the internal implementation of this module: if True, locking is allowed, if False, locking is forbidden.

When this is False, LockingDisabled is raised on any attempt to acquire a lock.

Definition at line 34 of file locking.py.

20.22 produtil.log Namespace Reference

Configures logging.

20.22.1 Detailed Description

Configures logging.

This module configures logging for stdout, stderr and the jlogfile. It also contains the jlogger, a logger.Logger object that is used to log directly to the jlogfile, and jlogdomain: a string name of the logger domain for the jlogfile.

Classes

- class JLogFormatter

  This subclass of MasterLogFormatter does not include exception information in the log file.
- class JLogHandler

  Custom LogHandler for the jlogfile.
- class MasterLogFormatter

  This is a custom log formatter that inserts the thread or process (logthread) that generated the log message.
- class MasterLogHandler

  Custom LogHandler for the master process of a multi-process job.
- class ThreadLogger

  Custom logging.Logger that inserts thread information.

Functions

- def postmsg (message)

  Sends the message to the jlogfile logging stream at level INFO.
- def set_jlogfile (filename)

  Tells the jlogger to log to the specified file instead of the current jlogfile.
- def stdout_is_stderr ()

  Returns True if it can determine that stdout and stderr are the same file or terminal.
- def mpi_redirect (threadname, stderrfile, stdoutfile, threadlevel=logging.WARNING, masterlevel=logging.I↩ NFO, openmode=None, logger=None)

  Used to split to multiple logging streams.
- def configureLogging (jlogfile=None, level=logging.INFO, jloglevel=logging.INFO, japplevel=logging.ERROR, eloglevel=logging.WARNING, ologlevel=logging.NOTSET, thread_logger=False, masterdomain='master')

  Configures log output to stderr, stdout and the jlogfile.

Variables

- string logthread = ''

  string for log messages to indicate thread number/name
- string jlogdomain = 'jlog'

  Logging domain for the jlogfile.
- jlogger = logging.getLogger(jlogdomain)

  A logging.Logger for the jlogdomain.
- jloghandler = None

  A logging.LogHandler for the jlogger.
- masterlogger = None

  Master log stream for MPI-split jobs.
- masterdomain = None

  Logging domain for the masterlogger.

20.22.2 Function Documentation

20.22.2.1 configureLogging()

```
def produtil.log.configureLogging (
              jlogfile = None,
              level = logging.INFO,
              jloglevel = logging.INFO,
              japplevel = logging.ERROR,
              eloglevel = logging.WARNING,
              ologlevel = logging.NOTSET,
              thread_logger = False,
              masterdomain = 'master' )
```

Configures log output to stderr, stdout and the jlogfile.

Configures log file locations and logging levels for all streams.

Note

Important notes when choosing levels:

- level - sets the global minimum log level. Anything below this level will be discarded regardless of other settings.
- jloglevel - this limit is applied before japplevel

Parameters

jlogfile	path to the jlogfile. Default: use os.environ('jlogfile') if set. Otherwise, stderr.
level	minimum logging level globally. Set to INFO by default. Change this to logging.DEBUG if you're debugging the program.
jloglevel	minimum logging level to send to jlogfile
japplevel	minimum logging level to send to jlogfile from all domains except that specified in jlogdomain. Be careful when changing this as it logs directly to the WCOSS-wide jlogfile in operations.
eloglevel	minimum logging level to send to stderr from ALL logs Set to None to disable stderr logging
ologlevel	minimum logging level to send to stdout from ALL logs Default: logging.NOTSET (no filtering) Set to None to disable stdout logging.
thread_logger	True to include the thread name in log messages.
masterdomain	The logging domain that will send messages to the main log stream for the job, even within individual ranks of mpi-split jobs

Definition at line 339 of file log.py.

Referenced by produtil.log.mpi_redirect(), and produtil.setup.setup().

20.22.2.2 mpi_redirect()

```
def produtil.log.mpi_redirect (
               threadname,
               stderrfile,
               stdoutfile,
               threadlevel = logging.WARNING,
               masterlevel = logging.INFO,
               openmode = None,
               logger = None )
```

Used to split to multiple logging streams.

When the Python script splits itself into multiple processes via MPI, this function is called to redirect stdout to stdoutfile, stderr to stderrfile, and produce a new logging stream to the original stderr, with a logging level set to masterlevel. That new logging stream is called the "master log" and will receive any messages at level masterlevel or higher, and any messages sent to the jlogdomain.

This can also be used to redirect ONLY stdout, in which case no master logging stream is set up. That is requested by stderrfile=None.

Parameters

threadname	the name of this process for logging purposes
stderrfile	file to receive stderr
stdoutfile	file to receive stdout
masterlevel	log level to send to master log stream
openmode	integer mode to use when opening files
logger	a logging.Logger for logging errors while splitting the log stream.

Definition at line 244 of file log.py.

Referenced by produtil.log.JLogHandler.set_jlogfile().

20.22.2.3 postmsg()

```
def produtil.log.postmsg (
              message )
```

Sends the message to the jlogfile logging stream at level INFO.

This is identical to:

```
jlogger.info(message).
```

Parameters

message	the message to log.

Definition at line 58 of file log.py.

Referenced by tc_pairs_wrapper.TcPairsWrapper.build_tc_pairs(), tc_stat_wrapper.TcStatWrapper.build_tc_↩ stat(), series_by_lead_wrapper.SeriesByLeadWrapper.create_animated_gifs(), series_by_init_wrapper.Series↩ ByInitWrapper.create_fcst_anly_to_ascii_file(), extra_tropical_cyclone_plotter.ExtraTropicalCyclonePlotter.get↩ _basemap(), tcmpr_plotter_wrapper.TCMPRPlotterWrapper.retrieve_optionals(), extract_tiles_wrapper.Extract↩ TilesWrapper.run_at_time(), and config_metplus.setup().

20.22.2.4 set_jlogfile()

```
def produtil.log.set_jlogfile (
              filename )
```

Tells the jlogger to log to the specified file instead of the current jlogfile.

Also updates the jlogfile environment variable. The argument must be a filename.

Parameters

filename	the new jlogfile

Definition at line 68 of file log.py.

20.22.2.5 stdout_is_stderr()

```
def produtil.log.stdout_is_stderr ( )
```

Returns True if it can determine that stdout and stderr are the same file or terminal.

Returns False if it can determine they are not, or if the result is inconclusive.

Definition at line 135 of file log.py.

Referenced by produtil.log.configureLogging().

20.22.3 Variable Documentation

20.22.3.1 masterlogger

```
produtil.log.masterlogger = None
```

Master log stream for MPI-split jobs.

When a job is split via mpi_redirect, this logger will send data to the master log stream at lower log levels. This is configurable via calls to mpi_redirect()

Definition at line 39 of file log.py.

20.23 produtil.mpi_impl Namespace Reference

Converts a group of MPI ranks to a runnable command.

20.23.1 Detailed Description

Converts a group of MPI ranks to a runnable command.

20.23.2 Produtil MPI Implementation

This package and its submodules implement execution of externals MPI programs. This package is not intended to be used directly, instead one should use produtil.run. This package appears to the outside to be a module that implements a common interface to various local MPI implementations. This is done by automatically detecting which MPI implementation is in use, and then importing the entire contents of the corresponding sub-module of produtil.mpi_impl. See the submodules for details on each implementation:

- produtil.mpi_impl.mpiexec — MPICH or public MPVAPICH2
- produtil.mpi_impl.impi — Intel MPI
- produtil.mpi_impl.mpiexec_mpt — SGI MPT
- produtil.mpi_impl.mpirun_lsf — LSF wrapped around IBMPE
- produtil.mpi_impl.no_mpi — For a purely serial environment.

20.23.3 Subroutines Imported from Implementation Modules

The following subroutines are imported from one of those modules. They are added to the mpi_impl package level to make the mpi_impl look identical to the underlying implementation module:

- openmp(arg,threads) - given a Runner, set it up to use OpenMP If threads is provided, it is the number of threads to use. Otherwise, no thread count is specified and it is assumed that the underlying OpenMP implementation will use the correct number.
- can_run_mpi() - does this computer support running MPI programs?
- bigexe_prepend(arg,∗∗kwargs) - Modifies an executable to run on a compute node instead of the batch node. This is intended for future support of the Cray architecture, where the batch script runs on a batch node, and must call "aprun" to execute a program on a remote compute node. This is the function that one would use to prepend "aprun" and its various arguments. This functionality is not presently tested.
- mpirunner(arg,allranks=False,∗∗kwargs) - Implementation of produtil.run.mpirun(). Given an object that is a subclass of produtil.mpiprog.MPIRanksBase, construct and return a produtil.prog.Runner that will execute that MPI command. The allranks=True option requests that the program use all available MPI ranks. An exception should be raised if the program also requests a specific number of ranks (other than 1).

  There are two different types of MPI programs that mpirunner must handle. One is MPI execution of non-MPI programs, which the caller requests via produtil.run.mpiserial. Some MPI implementations support running non-MPI programs directly, while others don't. The external C program "mpiserial" provides an MPI wrapper program to work around that lack of support. It is a simple MPI program that directs each rank to execute a shell command. The other variety of program mpirunner must handle is, of course, MPI programs. These are differentiated via: (serial,parallel)=arg.check_serial() If serial is true, the program is serial, if parallel is true, the program is parallel. If both are true, MPIMixed should be raised.

  The mpirunner must also handle the allranks=True vs. False cases. If allranks=True, the caller is requesting that the provided MPI program be run on all available ranks. If the MPI program also provides a rank specification (detected via arg.nranks()!=1) then the MPI_COMM_WORLD is overspecified and the mpirunner must raise MPIAllRanksError.

These are the detection routines imported from each submodule, except for no_mpi. The name of the routine is "detect()" in its module, and is renamed during import to the package-level namespace:

- impi_detect() — returns True if the Intel MPI should be used
- mpiexec_detect() - returns True if the MPICH or MVAPICH2 MPI should be used
- mpiexec_mpt_detect() - returns True if the SGI MPT should be used
- mpirun_lsf_detect() - returns True if LSF IBMPE should be used

20.23.4 New MPI Implementations

To implement a new MPI implementation, one must create a new submodule of mpi_impl. It is best to examine the existing modules and mimic them when doing this. Most architectures are similar to either the mpirun_lsf (which uses command files) or mpiexec (which provides arguments to mpiexec on the command line). In addition, the external program "mpiserial" provides a means by which to execute a list of serial programs via an MPI invocation for MPI implementations that do not natively support that (such as the problematic SGI MPT). Furthermore, some MPI implementations may have bugs or limitations that one must work around via setting environment variables (such as SGI MPT with its numerous hard-coded limits). The mpirunner and openmp functions should work around those problems.

Note that there are two utilities designed to simplify the implementation of a new MPI module:

- produtil.mpiprog.MPIRanksBase.to_arglist() – walks the tree of objects automatically generating an mpi invocation command (mpiexec, mpirun, etc.) with arguments, based on a provided set of rules. This is how the three existing modules make their MPI commands. It is quite simple to use, and handles the hard work of walking the object tree for you.
- produtil.mpi_impl.mpi_impl_base.CMDFGen - provides a way of easily writing a command file based on produtil.mpiprog.MPISerial objects. This is for MPI implementations such as IBMPE that require a file listing the commands to run on each MPI rank. It is also needed when using mpiserial to execute non-MPI programs under MPI

Once you have a new MPI implementation module, you must edit produtil/mpi_impl/__init__.py to detect your MPI implementation and correctly import the module. The produtil/mpi_impl/__init__.py must import that module's detect() function, and detect whether the MPI implementation should be used. If it should be, then **init**.py must import the relevant symbols from your module into the package-level namespace. There are instructions in the code in **init**.py on how to modify it to achieve these steps.

Namespaces

- impi

  Adds Intel MPI support to produtil.run.
- inside_aprun

  Adds support for running serial programs when one is inside an aprun execution.
- lsf_cray_intel

  Adds support for LSF+aprun with the Intel OpenMP to produtil.run.
- mpi_impl_base

  Utilities like CMDFGen to simplify adding new MPI implementations to the produtil.run suite of modules.
- mpiexec

  Adds MPICH or MVAPICH2 support to produtil.run.
- mpiexec_mpt

  Adds SGI MPT support to produtil.run.
- mpirun_lsf

  Adds LSF+IBMPE support to produtil.run.
- no_mpi

  Stub funcitons to allow produtil.mpi_impl to run when MPI is unavailable.
- srun

  Adds SLURM srun support to produtil.run.

20.24 produtil.mpi_impl.impi Namespace Reference

Adds Intel MPI support to produtil.run.

20.24.1 Detailed Description

Adds Intel MPI support to produtil.run.

This module is part of the produtil.mpi_impl package – see **init**.py for details. This implements the Intel MPI, but may work for other MPI implementations that use the "mpirun" command and OpenMP implementations that use the KMP_NUM_THREADS or OMP_NUM_THREADS environment variables.

Warning

This module assumes the TOTAL_TASKS environment variable is set to the maximum number of MPI ranks the program has available to it. That is used when the mpirunner is called with the allranks=True option.

Functions

- def runsync (logger=None)

  Runs the "sync" command as an exe().
- def openmp (arg, threads)

  Adds OpenMP support to the provided object.
- def detect ()

  Detects whether Intel MPI is available.
- def can_run_mpi ()

  Does this module represent an MPI implementation? Returns True.
- def make_bigexe (exe, kwargs)

  Returns an ImmutableRunner that will run the specified program.
- def mpirunner (arg, allranks=False, kwargs)

  Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Variables

- mpirun_path = produtil.fileop.find_exe('mpirun',raise_missing=False)

  Path to the mpirun program, or None if it could not be found.
- **module_logger** = logging.getLogger('lsf_cray_intel')

20.24.2 Function Documentation

20.24.2.1 can_run_mpi()

```
def produtil.mpi_impl.impi.can_run_mpi ( )
```

Does this module represent an MPI implementation? Returns True.

Definition at line 66 of file impi.py.

20.24.2.2 detect()

```
def produtil.mpi_impl.impi.detect ( )
```

Detects whether Intel MPI is available.

Definition at line 51 of file impi.py.

20.24.2.3 make_bigexe()

```
def produtil.mpi_impl.impi.make_bigexe (
              exe,
              kwargs )
```

Returns an ImmutableRunner that will run the specified program.

Returns

an empty list

Parameters

exe	The executable to run on compute nodes.
kwargs	Ignored.

Definition at line 70 of file impi.py.

20.24.2.4 mpirunner()

```
def produtil.mpi_impl.impi.mpirunner (
              arg,
              allranks = False,
              kwargs )
```

Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Parameters

arg	a tree of produtil.mpiprog.MPIRanksBase objects
allranks	if True, and only one rank is requested by arg, then all MPI ranks will be used
kwargs	passed to produtil.mpi_impl.mpi_impl_base.CMDFGen when mpiserial is in use.

Returns

a produtil.prog.Runner that will run the selected MPI program

Definition at line 77 of file impi.py.

20.24.2.5 openmp()

```
def produtil.mpi_impl.impi.openmp (
              arg,
              threads )
```

Adds OpenMP support to the provided object.

Parameters

arg	An produtil.prog.Runner or produtil.mpiprog.MPIRanksBase object tree
threads	the number of threads, or threads per rank, an integer

Definition at line 34 of file impi.py.

20.24.2.6 runsync()

```
def produtil.mpi_impl.impi.runsync (
              logger = None )
```

Runs the "sync" command as an exe().

Definition at line 26 of file impi.py.

20.24.3 Variable Documentation

20.24.3.1 mpirun_path

```
produtil.mpi_impl.impi.mpirun_path = produtil.fileop.find_exe('mpirun',raise_missing=False)
```

Path to the mpirun program, or None if it could not be found.

Definition at line 22 of file impi.py.

20.25 produtil.mpi_impl.inside_aprun Namespace Reference

Adds support for running serial programs when one is inside an aprun execution.

20.25.1 Detailed Description

Adds support for running serial programs when one is inside an aprun execution.

This module is part of the mpi_impl package – see produtil.mpi_impl for details. This implements execution of serial programs when one is inside an aprun execution.

Functions

- def runsync (logger=None)
  Runs the "sync" command as an exe().
- def **detect** ()
- def openmp (arg, threads)
  When more than one thread is requested, this raises OpenMPDisabled to indicate OpenMP is not allowed.
- def mpirunner (arg, kwargs)
  Raises an exception to indicate MPI is not supported.
- def can_run_mpi ()
  Returns False to indicate MPI is not supported.
- def make_bigexe (exe, kwargs)
  Returns an ImmutableRunner that will run the specified program.

Variables

- **module_logger** = logging.getLogger('lsf_cray_intel')

20.25.2 Function Documentation

20.25.2.1 can_run_mpi()

```
def produtil.mpi_impl.inside_aprun.can_run_mpi ( )
```

Returns False to indicate MPI is not supported.

Definition at line 50 of file inside_aprun.py.

20.25.2.2 make_bigexe()

```
def produtil.mpi_impl.inside_aprun.make_bigexe (
              exe,
              kwargs )
```

Returns an ImmutableRunner that will run the specified program.

Returns

an empty list

Parameters

exe	The executable to run on compute nodes.
kwargs	Ignored.

Definition at line 54 of file inside_aprun.py.

20.25.2.3 mpirunner()

```
def produtil.mpi_impl.inside_aprun.mpirunner (
              arg,
              kwargs )
```

Raises an exception to indicate MPI is not supported.

Parameters

arg,kwargs	Ignored.

Definition at line 45 of file inside_aprun.py.

20.25.2.4 openmp()

```
def produtil.mpi_impl.inside_aprun.openmp (
              arg,
              threads )
```

When more than one thread is requested, this raises OpenMPDisabled to indicate OpenMP is not allowed.

Parameters

arg	An produtil.prog.Runner or produtil.mpiprog.MPIRanksBase object tree
threads	the number of threads, or threads per rank, an integer

Definition at line 32 of file inside_aprun.py.

20.25.2.5 runsync()

```
def produtil.mpi_impl.inside_aprun.runsync (
              logger = None )
```

Runs the "sync" command as an exe().

Definition at line 16 of file inside_aprun.py.

20.26 produtil.mpi_impl.lsf_cray_intel Namespace Reference

Adds support for LSF+aprun with the Intel OpenMP to produtil.run.

20.26.1 Detailed Description

Adds support for LSF+aprun with the Intel OpenMP to produtil.run.

This module is part of the mpi_impl package – see produtil.mpi_impl for details. This implements the bizarre combination of LSF, Cray aprun with Intel OpenMP.

Functions

- def get_p_state_turbo ()

  Value to send to aprun –p-state option for Intel Turbo Mode.
- def runsync (logger=None)

  Runs the "sync" command as an exe().
- def openmp (arg, threads)

  Adds OpenMP support to the provided object.
- def **aprun_ln_sf** (source, target, content, logger=None)
- def detect ()

  Determines if Cray aprun should be used to run MPI programs by looking for the aprun program in $PATH.
- def can_run_mpi ()

  Does this module represent an MPI implementation? Returns True.
- def make_bigexe (exe, kwargs)

  Returns an ImmutableRunner that will run the specified program.
- def **mpirunner** (arg, allranks=False, logger=None, kwargs)
- def mpirunner2 (arg, allranks=False, logger=None, kwargs)

  Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Variables

- **module_logger** = logging.getLogger('lsf_cray_intel')
- **aprun_path** = produtil.fileop.find_exe('aprun',raise_missing=False)
- p_state_turbo = None

  Value to send to aprun –p-state option for Intel Turbo Mode.

20.26.2 Function Documentation

20.26.2.1 can_run_mpi()

```
def produtil.mpi_impl.lsf_cray_intel.can_run_mpi ( )
```

Does this module represent an MPI implementation? Returns True.

Definition at line 134 of file lsf_cray_intel.py.

20.26.2.2 get_p_state_turbo()

```
def produtil.mpi_impl.lsf_cray_intel.get_p_state_turbo ( )
```

Value to send to aprun –p-state option for Intel Turbo Mode.

This is the largest value printed out by

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

Which is either the highest allowed sustained clock speed, or the magic number for Turbo Mode.

Definition at line 30 of file lsf_cray_intel.py.

Referenced by produtil.mpi_impl.lsf_cray_intel.mpirunner2().

20.26.2.3 make_bigexe()

```
def produtil.mpi_impl.lsf_cray_intel.make_bigexe (
              exe,
              kwargs )
```

Returns an ImmutableRunner that will run the specified program.

Returns

an empty list

Note

This function does NOT search $PATH. That ensures the $PATH will be expanded on the compute node instead. Use produtil.fileop.find_exe() if you want to explicitly search the PATH before execution

Parameters

exe	The executable to run on compute nodes.
kwargs	Ignored.

Definition at line 138 of file lsf_cray_intel.py.

20.26.2.4 mpirunner2()

```
def produtil.mpi_impl.lsf_cray_intel.mpirunner2 (
              arg,
              allranks = False,
              logger = None,
              kwargs )
```

Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Parameters

arg	a tree of produtil.mpiprog.MPIRanksBase objects
allranks	if True, and only one rank is requested by arg, then all MPI ranks will be used
logger	a logging.Logger for log messages
kwargs	passed to produtil.mpi_impl.mpi_impl_base.CMDFGen when mpiserial is in use.

Returns

a produtil.prog.Runner that will run the selected MPI program

Definition at line 161 of file lsf_cray_intel.py.

Referenced by produtil.mpi_impl.lsf_cray_intel.make_bigexe().

20.26.2.5 openmp()

```
def produtil.mpi_impl.lsf_cray_intel.openmp (
              arg,
              threads )
```

Adds OpenMP support to the provided object.

Parameters

arg	An produtil.prog.Runner or produtil.mpiprog.MPIRanksBase object tree
threads	the number of threads, or threads per rank, an integer

Definition at line 67 of file lsf_cray_intel.py.

20.26.2.6 runsync()

```
def produtil.mpi_impl.lsf_cray_intel.runsync (
              logger = None )
```

Runs the "sync" command as an exe().

Definition at line 61 of file lsf_cray_intel.py.

20.26.3 Variable Documentation

20.26.3.1 p_state_turbo

```
produtil.mpi_impl.lsf_cray_intel.p_state_turbo = None
```

Value to send to aprun –p-state option for Intel Turbo Mode.

This is the largest value printed out by

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

Which is either the highest allowed sustained clock speed, or the magic number for Turbo Mode.

Definition at line 28 of file lsf_cray_intel.py.

20.27 produtil.mpi_impl.mpi_impl_base Namespace Reference

Utilities like CMDFGen to simplify adding new MPI implementations to the produtil.run suite of modules.

20.27.1 Detailed Description

Utilities like CMDFGen to simplify adding new MPI implementations to the produtil.run suite of modules.

This module contains classes and functions to assist developers in extending the functionality of the produtil.mpi_impl package. The main highlight is the CMDFGen, which generates command files. Some MPI implementations, and the mpiserial program, want to read a file with one line per MPI rank telling what program to run on each rank. For example, LSF+IBMPE and LoadLeveler+IBMPE work this way if one wants to run different programs on different ranks.

Classes

- class CMDFGen

  Generates files with one line per MPI rank, telling what program to run on each rank.
- class MPIAllRanksError

  Raised when the allranks=True keyword is sent to mpirun or mpirunner, but the MPI program specification has more than one rank.
- class MPIConfigError

  Base class of MPI configuration exceptions.
- class MPIDisabled

  Thrown to MPI is not supported.
- class MPIMixed

  Thrown to indicate serial and parallel processes are being mixed in a single mpi_comm_world.
- class MPISerialMissing

  Raised when the mpiserial program is required, but is missing.
- class OpenMPDisabled

  Raised when OpenMP is not supported by the present implementation.
- class WrongMPI

  Unused: raised when the wrong MPI implementation is accessed.

Variables

- **module_logger** = logging.getLogger('produtil.mpi_impl')

20.28 produtil.mpi_impl.mpiexec Namespace Reference

Adds MPICH or MVAPICH2 support to produtil.run.

20.28.1 Detailed Description

Adds MPICH or MVAPICH2 support to produtil.run.

This module is part of the mpi_impl package – see produtil.mpi_impl for details. This implements the Hydra MPI wrapper and MPICH MPI implementation with Intel OpenMP, but may work for other MPI implementations that use the "mpiexec" command and OpenMP implementations that use the KMP_NUM_THREADS or OMP_NUM_THR↩EADS environment variables.

Warning

This module assumes the TOTAL_TASKS environment variable is set to the maximum number of MPI ranks the program has available to it. That is used when the mpirunner is called with the allranks=True option.

Functions

- def runsync (logger=None)

  Runs the "sync" command as an exe().
- def openmp (arg, threads)

  Adds OpenMP support to the provided object.
- def detect ()

  Detects whether the MPICH mpi implementation is available by looking for the mpiexec program in $PATH.
- def can_run_mpi ()

  Does this module represent an MPI implementation? Returns True.
- def make_bigexe (exe, kwargs)

  Returns an ImmutableRunner that will run the specified program.
- def **mpirunner** (arg, allranks=False, logger=None, kwargs)
- def mpirunner2 (arg, allranks=False, kwargs)

  Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Variables

- mpiexec_path = produtil.fileop.find_exe('mpiexec',raise_missing=False)

  Path to the mpiexec program.
- **module_logger** = logging.getLogger('mpiexec')

20.28.2 Function Documentation

20.28.2.1 can_run_mpi()

```
def produtil.mpi_impl.mpiexec.can_run_mpi ( )
```

Does this module represent an MPI implementation? Returns True.

Definition at line 69 of file mpiexec.py.

20.28.2.2 detect()

```
def produtil.mpi_impl.mpiexec.detect ( )
```

Detects whether the MPICH mpi implementation is available by looking for the mpiexec program in $PATH.

Definition at line 64 of file mpiexec.py.

20.28.2.3 make_bigexe()

```
def produtil.mpi_impl.mpiexec.make_bigexe (
              exe,
              kwargs )
```

Returns an ImmutableRunner that will run the specified program.

Returns

an empty list

Parameters

exe	The executable to run on compute nodes.
kwargs	Ignored.

Definition at line 73 of file mpiexec.py.

20.28.2.4 mpirunner2()

```
def produtil.mpi_impl.mpiexec.mpirunner2 (
              arg,
              allranks = False,
              kwargs )
```

Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Parameters

arg	a tree of produtil.mpiprog.MPIRanksBase objects
allranks	if True, and only one rank is requested by arg, then all MPI ranks will be used
kwargs	passed to produtil.mpi_impl.mpi_impl_base.CMDFGen when mpiserial is in use.

Returns

a produtil.prog.Runner that will run the selected MPI program

Warning

Assumes the TOTAL_TASKS environment variable is set if allranks=True

Definition at line 87 of file mpiexec.py.

Referenced by produtil.mpi_impl.mpiexec.make_bigexe().

20.28.2.5 openmp()

```
def produtil.mpi_impl.mpiexec.openmp (
              arg,
              threads )
```

Adds OpenMP support to the provided object.

Parameters

arg	An produtil.prog.Runner or produtil.mpiprog.MPIRanksBase object tree
threads	the number of threads, or threads per rank, an integer

Definition at line 34 of file mpiexec.py.

20.28.2.6 runsync()

```
def produtil.mpi_impl.mpiexec.runsync (
              logger = None )
```

Runs the "sync" command as an exe().

Definition at line 26 of file mpiexec.py.

20.29 produtil.mpi_impl.mpiexec_mpt Namespace Reference

Adds SGI MPT support to produtil.run.

20.29.1 Detailed Description

Adds SGI MPT support to produtil.run.

This module is part of the produtil.mpi_impl package. It underlies the produtil.run.openmp, produtil.run.mpirun , and produtil.run.mpiserial functions, providing the implementation needed to run with the SGI MPT MPI implementation.

Warning

> This module assumes the TOTAL_TASKS environment variable is set to the maximum number of MPI ranks the program has available to it. That is used when the mpirunner is called with the allranks=True option.

Functions

- def runsync (logger=None)

  Runs the "sync" command as an exe().
- def openmp (arg, threads)

  Adds OpenMP support to the provided object.
- def detect ()

  Detects whether the SGI MPT is available by looking for mpiexec_mpt.
- def guess_nthreads (default)

  Tries to guess the number of threads in use.
- def can_run_mpi ()

  Does this module represent an MPI implementation? Returns True.
- def make_bigexe (exe, kwargs)

  Returns an ImmutableRunner that will run the specified program.
- def mpirunner (arg, allranks=False, kwargs)

  Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Variables

- mpiexec_mpt_path = produtil.fileop.find_exe('mpiexec_mpt',raise_missing=False)

  Path to the mpiexec_mpt program.
- **module_logger** = logging.getLogger('lsf_cray_intel')

20.29.2 Function Documentation

20.29.2.1 can_run_mpi()

```
def produtil.mpi_impl.mpiexec_mpt.can_run_mpi ( )
```

Does this module represent an MPI implementation? Returns True.

Definition at line 63 of file mpiexec_mpt.py.

20.29.2.2 detect()

```
def produtil.mpi_impl.mpiexec_mpt.detect ( )
```

Detects whether the SGI MPT is available by looking for mpiexec_mpt.

Definition at line 48 of file mpiexec_mpt.py.

20.29.2.3 guess_nthreads()

```
def produtil.mpi_impl.mpiexec_mpt.guess_nthreads (
              default )
```

Tries to guess the number of threads in use.

Parameters

default	the value to return if the function cannot guess

Definition at line 52 of file mpiexec_mpt.py.

20.29.2.4 make_bigexe()

```
def produtil.mpi_impl.mpiexec_mpt.make_bigexe (
```

```
                exe,
                kwargs )
```

Returns an ImmutableRunner that will run the specified program.

Returns

an empty list

Parameters

exe	The executable to run on compute nodes.
kwargs	Ignored.

Definition at line 67 of file mpiexec_mpt.py.

20.29.2.5 mpirunner()

```
def produtil.mpi_impl.mpiexec_mpt.mpirunner (
                arg,
                allranks = False,
                kwargs )
```

Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Parameters

arg	a tree of produtil.mpiprog.MPIRanksBase objects
allranks	if True, and only one rank is requested by arg, then all MPI ranks will be used
kwargs	passed to produtil.mpi_impl.mpi_impl_base.CMDFGen when mpiserial is in use.

Returns

a produtil.prog.Runner that will run the selected MPI program

Warning

Assumes the TOTAL_TASKS environment variable is set if allranks=True

Definition at line 74 of file mpiexec_mpt.py.

20.29.2.6 openmp()

```
def produtil.mpi_impl.mpiexec_mpt.openmp (
                arg,
                threads )
```

Adds OpenMP support to the provided object.

Parameters

arg	An produtil.prog.Runner or produtil.mpiprog.MPIRanksBase object tree
threads	the number of threads, or threads per rank, an integer

Definition at line 32 of file mpiexec_mpt.py.

20.29.2.7 runsync()

```
def produtil.mpi_impl.mpiexec_mpt.runsync (
              logger = None )
```

Runs the "sync" command as an exe().

Definition at line 24 of file mpiexec_mpt.py.

20.29.3 Variable Documentation

20.29.3.1 mpiexec_mpt_path

```
produtil.mpi_impl.mpiexec_mpt.mpiexec_mpt_path = produtil.fileop.find_exe('mpiexec_mpt',raise↩
_missing=False)
```

Path to the mpiexec_mpt program.

Definition at line 21 of file mpiexec_mpt.py.

20.30 produtil.mpi_impl.mpirun_lsf Namespace Reference

Adds LSF+IBMPE support to produtil.run.

20.30.1 Detailed Description

Adds LSF+IBMPE support to produtil.run.

This module is part of the produtil.mpi_impl package. It underlies the produtil.run.openmp, produtil.run.mpirun , and produtil.run.mpiserial functions, providing the implementation needed to run with LSF combined with the IBMPE MPI implementation. It may work with other MPI implementations connected to LSF, as long as they use mpirun.lsf to launch MPI programs.

Note

Unlike other MPI implementations, LSF does not allow changing of the number of MPI ranks used when running an MPI program. You can only run on all provided ranks, or one rank. Hence the TOTAL_TASKS variable used elsewhere in produtil, is ignored here.

Functions

- def runsync (logger=None)

  Runs the "sync" command as an exe().
- def openmp (arg, threads)

  Adds OpenMP support to the provided object.
- def detect ()

  Determines if LSF+IBMPE should be used to run MPI programs by looking for the mpirun.lsf program in $PATH.
- def can_run_mpi ()

  Does this module represent an MPI implementation? Returns True.
- def make_bigexe (exe, kwargs)

  Returns an ImmutableRunner that will run the specified program.
- def mpirunner (arg, allranks=False, logger=None, kwargs)

  Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Variables

- mpirun_lsf_path = produtil.fileop.find_exe('mpirun.lsf',raise_missing=False)

  Path to the mpirun.lsf program, or None if it isn't found.
- **module_logger** = logging.getLogger('lsf_cray_intel')

20.30.2 Function Documentation

20.30.2.1 can_run_mpi()

```
def produtil.mpi_impl.mpirun_lsf.can_run_mpi ( )
```

Does this module represent an MPI implementation? Returns True.

Definition at line 54 of file mpirun_lsf.py.

20.30.2.2 make_bigexe()

```
def produtil.mpi_impl.mpirun_lsf.make_bigexe (
              exe,
              kwargs )
```

Returns an ImmutableRunner that will run the specified program.

Returns

an empty list

Parameters

exe	The executable to run on compute nodes.
kwargs	Ignored.

Definition at line 58 of file mpirun_lsf.py.

20.30.2.3 mpirunner()

```
def produtil.mpi_impl.mpirun_lsf.mpirunner (
              arg,
              allranks = False,
              logger = None,
              kwargs )
```

Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Parameters

arg	a tree of produtil.mpiprog.MPIRanksBase objects
allranks	if True, and only one rank is requested by arg, then all MPI ranks will be used
logger	a logging.Logger for log messages
kwargs	passed to produtil.mpi_impl.mpi_impl_base.CMDFGen when mpiserial is in use.

Returns

a produtil.prog.Runner that will run the selected MPI program

Note

LSF does not support modifying the number of MPI ranks to use when running a program. You can only use all provided ranks, or one rank.

Definition at line 65 of file mpirun_lsf.py.

20.30.2.4 openmp()

```
def produtil.mpi_impl.mpirun_lsf.openmp (
              arg,
              threads )
```

Adds OpenMP support to the provided object.

Parameters

arg	An produtil.prog.Runner or produtil.mpiprog.MPIRanksBase object tree
threads	the number of threads, or threads per rank, an integer

Definition at line 33 of file mpirun_lsf.py.

20.30.2.5 runsync()

```
def produtil.mpi_impl.mpirun_lsf.runsync (
               logger = None )
```

Runs the "sync" command as an exe().

Definition at line 25 of file mpirun_lsf.py.

20.30.3 Variable Documentation

20.30.3.1 mpirun_lsf_path

```
produtil.mpi_impl.mpirun_lsf.mpirun_lsf_path = produtil.fileop.find_exe('mpirun.lsf',raise_↩
missing=False)
```

Path to the mpirun.lsf program, or None if it isn't found.

Definition at line 22 of file mpirun_lsf.py.

20.31 produtil.mpi_impl.no_mpi Namespace Reference

Stub funcitons to allow produtil.mpi_impl to run when MPI is unavailable.

20.31.1 Detailed Description

Stub funcitons to allow produtil.mpi_impl to run when MPI is unavailable.

This module is part of the produtil.mpi_impl package. It underlies the produtil.run.openmp, produtil.run.mpirun , and produtil.run.mpiserial functions, providing the implementation needed to run when MPI is unavailable.

Functions

- def runsync (logger=None)

  Runs the "sync" command as an exe().
- def openmp (arg, threads)

  Does nothing.
- def mpirunner (arg, kwargs)

  Raises an exception to indicate MPI is not supported.
- def can_run_mpi ()

  Returns False to indicate MPI is not supported.
- def make_bigexe (exe, kwargs)

  Returns an ImmutableRunner that will run the specified program.

Variables

- **module_logger** = logging.getLogger('lsf_cray_intel')

20.31.2 Function Documentation

20.31.2.1 can_run_mpi()

```
def produtil.mpi_impl.no_mpi.can_run_mpi ( )
```

Returns False to indicate MPI is not supported.

Definition at line 40 of file no_mpi.py.

20.31.2.2 make_bigexe()

```
def produtil.mpi_impl.no_mpi.make_bigexe (
              exe,
              kwargs )
```

Returns an ImmutableRunner that will run the specified program.

Returns

an empty list

Parameters

exe	The executable to run on compute nodes.
kwargs	Ignored.

Definition at line 43 of file no_mpi.py.

20.31.2.3 mpirunner()

```
def produtil.mpi_impl.no_mpi.mpirunner (
              arg,
              kwargs )
```

Raises an exception to indicate MPI is not supported.

Parameters

arg,kwargs	Ignored.

Definition at line 36 of file no_mpi.py.

20.31.2.4 openmp()

```
def produtil.mpi_impl.no_mpi.openmp (
              arg,
              threads )
```

Does nothing.

This implementation does not support OpenMP.

Parameters

arg	An produtil.prog.Runner or produtil.mpiprog.MPIRanksBase object tree
threads	the number of threads, or threads per rank, an integer

Definition at line 21 of file no_mpi.py.

20.31.2.5 runsync()

```
def produtil.mpi_impl.no_mpi.runsync (
              logger = None )
```

Runs the "sync" command as an exe().

Definition at line 14 of file no_mpi.py.

20.32 produtil.mpi_impl.srun Namespace Reference

Adds SLURM srun support to produtil.run.

20.32.1 Detailed Description

Adds SLURM srun support to produtil.run.

This module is part of the mpi_impl package – see produtil.mpi_impl for details. This translates produtil.run directives to SLURM srun commands.

Functions

- def runsync (logger=None)

  Runs the "sync" command as an exe().
- def openmp (arg, threads)

  Adds OpenMP support to the provided object.
- def detect ()

  Detects whether the SLURM srun command is available by looking for it in the $PATH.
- def can_run_mpi ()

  Does this module represent an MPI implementation? Returns True.
- def make_bigexe (exe, kwargs)

  Returns an ImmutableRunner that will run the specified program.
- def mpirunner (arg, allranks=False, kwargs)

  Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.
- def mpirunner_impl (arg, allranks=False, kwargs)

  This is the underlying implementation of mpirunner and should not be called directly.

Variables

- srun_path = produtil.fileop.find_exe('srun',raise_missing=False)

  Path to the srun program.
- **module_logger** = logging.getLogger('lsf_cray_intel')

20.32.2 Function Documentation

20.32.2.1 can_run_mpi()

```
def produtil.mpi_impl.srun.can_run_mpi ( )
```

Does this module represent an MPI implementation? Returns True.

Definition at line 46 of file srun.py.

20.32.2.2 detect()

```
def produtil.mpi_impl.srun.detect ( )
```

Detects whether the SLURM srun command is available by looking for it in the $PATH.

Definition at line 41 of file srun.py.

20.32.2.3 make_bigexe()

```
def produtil.mpi_impl.srun.make_bigexe (
              exe,
              kwargs )
```

Returns an ImmutableRunner that will run the specified program.

Returns

an empty list

Parameters

exe	The executable to run on compute nodes.
kwargs	Ignored.

Definition at line 50 of file srun.py.

20.32.2.4 mpirunner()

```
def produtil.mpi_impl.srun.mpirunner (
              arg,
              allranks = False,
              kwargs )
```

Turns a produtil.mpiprog.MPIRanksBase tree into a produtil.prog.Runner.

Parameters

arg	a tree of produtil.mpiprog.MPIRanksBase objects
allranks	if True, and only one rank is requested by arg, then all MPI ranks will be used
kwargs	passed to produtil.mpi_impl.mpi_impl_base.CMDFGen when mpiserial is in use.

Returns

a produtil.prog.Runner that will run the selected MPI program

Definition at line 57 of file srun.py.

20.32.2.5 mpirunner_impl()

```
def produtil.mpi_impl.srun.mpirunner_impl (
              arg,
              allranks = False,
              kwargs )
```

This is the underlying implementation of mpirunner and should not be called directly.

Definition at line 69 of file srun.py.

Referenced by produtil.mpi_impl.srun.mpirunner().

20.32.2.6 openmp()

```
def produtil.mpi_impl.srun.openmp (
              arg,
              threads )
```

Adds OpenMP support to the provided object.

Parameters

arg	An produtil.prog.Runner or produtil.mpiprog.MPIRanksBase object tree
threads	the number of threads, or threads per rank, an integer

Definition at line 25 of file srun.py.

20.32.2.7 runsync()

```
def produtil.mpi_impl.srun.runsync (
              logger = None )
```

Runs the "sync" command as an exe().

Definition at line 18 of file srun.py.

20.33 produtil.mpiprog Namespace Reference

Object structure for describing MPI programs.

20.33.1 Detailed Description

Object structure for describing MPI programs.

Do not load this module directly. It is meant to be loaded only by the produtil.run module.

This module handles execution of MPI programs, and execution of groups of non-MPI programs through an MPI interface (which requires all sorts of tricks). This module is also the interface to the various produtil.mpi_impl.* modules that generate the shell command to run MPI programs. This module is built on top of the produtil.prog module and uses it to run the MPI-launching program for your local cluster (mpiexec, mpirun, poe, etc.)

In addition, this module contains code to simplify adding new MPI implementations to the produtil.mpi_impl subpackage. High-level code, such as the HWRF scripts, use the produtil.run module to generate object trees of MPIRanksBase objects. The produtil.mpi_impl subpackages then implement an mpirunner function that turns those into a produtil.prog.Runner to be directly executed. The MPIRanksBase object, and its subclasses, implement a few utilites to automate that for you:

- to_arglist — converts the MPI ranks to an mpi launcher command as a produtil.prog.Runner, or to an array of strings for a command file.
- nranks — calculates the number of requested MPI ranks
- expand_iter — iterates over groups of identical MPI ranks
- check_serial — tells whether this program is running MPI programs, or running serial programs as if they were MPI (or both, which most MPI implementations don't support)

For MPI implementations that require a command file, see the produtil.mpi_impl.mpi_impl_base CMDFGen class to have the produtil.prog module automatically write the command file before executing the program. The produtil.mpi_impl.mpirun_lsf shows an example of how to use it.

See the produtil.run module for full documentation.

Classes

- class ComplexProgInput

  Raised when something that cannot be expressed as a pure MPI rank is given as a pure MPI rank.
- class InputsNotStrings

  Raised when the validation scripts were expecting string arguments or string executable names, but something else was found.
- class MPIProgSyntaxError

  Base class of syntax errors in MPI program specifications.
- class MPIRank

  Represents a single MPI rank.
- class MPIRanksBase

  This is the abstract superclass of all classes that represent one or more MPI ranks, including MPI ranks that are actually serial programs.
- class MPIRanksMPMD

  Represents a group of MPI programs, each of which have some number of ranks assigned.
- class MPIRanksSPMD

  Represents one MPI program duplicated across many ranks.
- class MPISerial

  Represents a single rank of an MPI program that is actually running a serial program.
- class NotMPIProg

  Raised when an MPI program was expected but something else was given.
- class NotSerialProg

  Raised when a serial program was expected, but something else was given.

Functions

- def **collapse** (runner)

20.34 produtil.numerics Namespace Reference

Time manipulation and other numerical routines.

20.34.1 Detailed Description

Time manipulation and other numerical routines.

This module implements various simple numerical algorithms, such as partial sorting, time manipulation or fraction-to-date conversions. It also contains two array-like classes that take datetime objects as indices.

Classes

- class InvalidTimespan

  Superclass of exceptions relating to groups of one or more distinct times and relationships between them.
- class InvalidTimestep

  Raised when a timestep is invalid, such as a negative timestep for a situation that requires a positive one.
- class NoNearbyValues

  Raised when an operation has a set of known times, but another provided time is not near one of those known times.
- class NoTimespan

  Raised when a timespan was expected, but none was available.
- class NotInTimespan

  Raised when a time is outside the range of times being processed by a function.
- class partial_ordering

  Sorts a pre-determined list of objects, placing unknown items at a specified location.
- class TimeArray

  A time-indexed array that can only handle equally spaced times.
- class TimeContainer

  Abstract base class that maps from time to objects.
- class TimeError

  Base class used for time-related exceptions.
- class TimeMapping

  Maps from an ordered list of times to arbitrary data.

Functions

- def great_arc_dist (xlon1, ylat1, xlon2, ylat2)

  Great arc distance between two points on Earth.
- def fcst_hr_min (time, start)

  Return forecast time in hours and minutes.
- def randint_zeromean (count, imax, randomizer=None)

  Generates "count" numbers uniformly distributed between -imax and imax, inclusive, with a mean of zero.
- def split_fraction (f)

  Splits a fraction into components.
- def within_dt_epsilon (time1, time2, epsilon)

  Returns True if time1 is within epsilon of time2, and False otherwise.
- def timedelta_epsilon (times, rel=None, default=None, sort=False, numerator=10)

  Decides a reasonable epsilon for time equality comparisons.
- def to_fraction (a, b=None, negok=False)

  Converts an object or two to a fraction.
- def to_datetime_rel (d, rel)

  Converts objects to a datetime relative to another datetime.
- def to_datetime (d)

  Converts the argument to a datetime.
- def to_timedelta (a, b=None, negok=True)

  Converts an object to a datetime.timedelta.
- def minutes_seconds_rest (fraction)

  Splits the given fractions.Fraction of seconds into integer minutes, seconds and fractional remainder <=0.
- def nearest_datetime (start, target, timestep)

  Return the nearest datetime.datetime to a target.
- def is_at_timestep (start, target, timestep)

  Returns True if the target time lies exactly on a timestep, and False otherwise.
- def str_timedelta (dt)

  Converts a timedelta to a string.

20.34.2 Function Documentation

20.34.2.1 fcst_hr_min()

```
def produtil.numerics.fcst_hr_min (
              time,
              start )
```

Return forecast time in hours and minutes.

Given a forecast datetime.datetime and an analysis datetime.datetime, this returns a tuple containing the forecast hour and minute, rounded to the nearest integer minute.

Parameters

time	forecast time as a datetime.datetime
start	analysis time as a datetime.datetime

Returns

a tuple (ihours,iminutes)

Definition at line 159 of file numerics.py.

Referenced by produtil.config.ConfTimeFormatter.get_value().

20.34.2.2 great_arc_dist()

```
def produtil.numerics.great_arc_dist (
              xlon1,
              ylat1,
              xlon2,
              ylat2 )
```

Great arc distance between two points on Earth.

Calculates the great arc distance in meters between two points using the Haversine method. Uses the local Earth radius at the latitude half-way between the two points.

Parameters

xlon1,ylat1	first point, degrees
xlon2,ylat2	second point, degrees

Returns

distance in meters

Definition at line 132 of file numerics.py.

20.34.2.3 is_at_timestep()

```
def produtil.numerics.is_at_timestep (
              start,
              target,
              timestep )
```

Returns True if the target time lies exactly on a timestep, and False otherwise.

Parameters

start	the fixed start time of allowed return values
target	the time desired
timestep	the times between allowed return values

Definition at line 487 of file numerics.py.

20.34.2.4 minutes_seconds_rest()

```
def produtil.numerics.minutes_seconds_rest (
              fraction )
```

Splits the given fractions.Fraction of seconds into integer minutes, seconds and fractional remainder <=0.

Parameters

fraction	the fraction to convert, assumed to be in seconds

Returns

a tuple (minutes,seconds,rest) as integers

Definition at line 453 of file numerics.py.

20.34.2.5 nearest_datetime()

```
def produtil.numerics.nearest_datetime (
              start,
              target,
              timestep )
```

Return the nearest datetime.datetime to a target.

Given a start time, a target time and a timestep, determine the nearest time not earlier than the target that lies exactly on a timestep. Input start and target can be anything understood by to_datetime, and the timestep can be anything understood by to_fraction. Return value is a datetime.datetime object.

Parameters

start	the fixed start time of allowed return values
target	the time desired
timestep	the times between allowed return values

Definition at line 467 of file numerics.py.

20.34.2.6 randint_zeromean()

```
def produtil.numerics.randint_zeromean (
              count,
```

```
        imax,
        randomizer = None )
```

Generates "count" numbers uniformly distributed between -imax and imax, inclusive, with a mean of zero.

Parameters

count	number of numbers to return
imax	maximum value of any number
randomizer	the random module, or something that looks like it

Definition at line 179 of file numerics.py.

20.34.2.7 split_fraction()

```
def produtil.numerics.split_fraction (
        f )
```

Splits a fraction into components.

Splits a fraction.Fraction into integer, numerator and denominator parts. For example, split_fraction(Fraction(13,7)) will return (1,6,7) since 1+6/7=13/7.

Returns

a tuple (integer,numerator,denominator)

Definition at line 217 of file numerics.py.

Referenced by produtil.numerics.str_timedelta().

20.34.2.8 str_timedelta()

```
def produtil.numerics.str_timedelta (
        dt )
```

Converts a timedelta to a string.

Converts dt to a string of the format "DD:HH:MM:SS+num/den"

- DD - number of days
- HH - number of hours
- MM - minutes
- SS - seconds
- num/den - fractional part The to_fraction is used to get the fractional part.

Parameters

dt	anything convertible to a datetime.timedelta.

Definition at line 500 of file numerics.py.

20.34.2.9 timedelta_epsilon()

```
def produtil.numerics.timedelta_epsilon (
              times,
              rel = None,
              default = None,
              sort = False,
              numerator = 10 )
```

Decides a reasonable epsilon for time equality comparisons.

Given an iterable of datetime objects (or anything accepted by to_datetime_rel), computes the minimum time difference between any two adjacent times and divides it by "numerator" (default: 10). The "rel" argument is the relative time when calling to_datetime_rel. If unspecified, it will be the first time seen (in which case that time must be acceptable to to_datetime). The "default" is the return value when all elements in "times" are identical, or when there is only one element. If the default is unspecified and it is needed, then NoTimespan is raised. If sort is specified and True, then the times will be sorted before anything is done (it is False by default).

Parameters

times	a list of example times for comparison
rel	a reference time to which the times will be compared
default	if too few unique times are found, this is returned If that happens and no default is provided, an exception is raised
sort	if True, sort times
numerator	A measure of how close times should be. The least time difference will be at least numerator times the epsilon

Returns

an epsilon value

Definition at line 253 of file numerics.py.

20.34.2.10 to_datetime()

```
def produtil.numerics.to_datetime (
              d )
```

Converts the argument to a datetime.

If the argument is already a datetime, it is simply returned. Otherwise it must be a string of the format YYYYMM↩ DDHH, YYYYMMDDHHMM, or YYYY-MM-DD HH:MM:SS.

Parameters

d	the object being converted

Returns

the resulting datetime.datetime object

Definition at line 379 of file numerics.py.

Referenced by produtil.numerics.TimeArray.__init__(), produtil.numerics.TimeMapping.__init__(), produtil.numerics.TimeContainer.get(), produtil.numerics.TimeMapping.index_of(), produtil.numerics.is_at_timestep(), produtil.numerics.nearest_datetime(), produtil.numerics.timedelta_epsilon(), produtil.config.ProdConfig.timestrinterp(), produtil.numerics.to_datetime_rel(), and produtil.numerics.within_dt_epsilon().

20.34.2.11 to_datetime_rel()

```
def produtil.numerics.to_datetime_rel (
              d,
              rel )
```

Converts objects to a datetime relative to another datetime.

Given a datetime object "rel", converts "d" to a datetime. Object "d" can be anything accepted by to_datetime, or anything accepted as a single argument by to_timedelta. If it is a timedelta, it is added to "rel" to get the final time.

Parameters

d	the object being converted
rel	the datetime.datetime to which it is to be converted

Definition at line 352 of file numerics.py.

Referenced by produtil.numerics.TimeArray.__init__(), produtil.numerics.TimeArray.index_of(), produtil.numerics.nearest_datetime(), produtil.numerics.timedelta_epsilon(), and produtil.config.ProdConfig.timestrinterp().

20.34.2.12 to_fraction()

```
def produtil.numerics.to_fraction (
              a,
              b = None,
              negok = False )
```

Converts an object or two to a fraction.

This routine is a wrapper around fraction.Fraction() which accepts additional inputs. Fractions are needed to provide higher precision in time and resolution calculations (and also to match the WRF behavior). The arguments

are the same as for the fractions.Fraction constructor, but additional calling conventions are accepted: a single float argument, a single datetime.timedelta object, or a string containing an integer and a fraction. If a float is provided, its denominator is restricted to be no more than 1000000. If the resulting fraction is not larger than 0, then InvalidTimestep is thrown unless negok=True.

Examples:

```
to_fraction(0.855)            # 0.855 seconds
to_fraction(33)               # 33 seconds
to_fraction('12/5')           # 2.4 seconds
to_fraction('7+1/2')          # 7.5 seconds
to_fraction(0.0000001) # ERROR! Value of the float() is less
                       # than 1e-6
to_fraction(1,10000000) # Valid as a fraction, even though value
                        # is less than 1e-6
```

Parameters

a,b	the objects to convert
negok	if True, negative numbers are okay.

Definition at line 302 of file numerics.py.

Referenced by produtil.numerics.TimeArray.__init__(), produtil.numerics.TimeArray.index_of(), produtil.numerics.is_at_timestep(), produtil.numerics.minutes_seconds_rest(), produtil.numerics.nearest_datetime(), produtil.numerics.TimeContainer.neartime(), produtil.numerics.str_timedelta(), produtil.numerics.timedelta_epsilon(), produtil.numerics.to_timedelta(), and produtil.numerics.within_dt_epsilon().

20.34.2.13 to_timedelta()

```
def produtil.numerics.to_timedelta (
              a,
              b = None,
              negok = True )
```

Converts an object to a datetime.timedelta.

Returns a datetime.timedelta object. If "a" is a timedelta, then that value is returned. If "a" is a string of the format 08:13 or 08:13:12, optionally preceded by a "-" then it is interpreted as HH:MM or HH:MM:SS, respectively (where a "-" negates). Otherwise, the arguments are sent to the to_fraction() function, and the result is converted into a timedelta.

Parameters

a	object being converted
b	second object, if a time range is sent
negok	if True, negative timespans are allowed

Returns

a datetime.timedelta

Definition at line 404 of file numerics.py.

Referenced by produtil.numerics.TimeArray.__init__(), produtil.numerics.str_timedelta(), produtil.numerics.↩ timedelta_epsilon(), and produtil.numerics.to_datetime_rel().

20.34.2.14 within_dt_epsilon()

```
def produtil.numerics.within_dt_epsilon (
              time1,
              time2,
              epsilon )
```

Returns True if time1 is within epsilon of time2, and False otherwise.

Parameters

time1,time2	the times being compared
epsilon	how close they need to be in order to be considered equal.

Returns

True if the times are within epsilon of each other, False if not

Definition at line 231 of file numerics.py.

20.35 produtil.pipeline Namespace Reference

Internal module that launches and monitors processes.

20.35.1 Detailed Description

Internal module that launches and monitors processes.

Do not use this module directly: it is part of the internal implementation of the produtil.prog and produtil.run modules. It converts a produtil.prog.Runner object to processes, and monitors the processes until they exit, sending and receiving data as needed. This replaces the built-in "subprocess" module which is not capable of general-purpose pipeline execution.

Classes

- class Constant

  A class used to implement named constants.
- class NoMoreProcesses

  Raised when the produtil.sigsafety package catches a fatal signal.
- class Pipeline

  This class is a wrapper around launch and manage.

Functions

- def unblock (stream, logger=None)

  Attempts to modify the given stream to be non-blocking.
- def call_fcntrl (stream, on, off, logger=None)

  Internal function that implements unblock()
- def pipe (logger=None)

  Creates a pipe that will be closed on exec.
- def padd (p)

  Adds a file descriptor to the list to close before exec.
- def pclose (i)

  Closes a file descriptor, removing it from the list that must be closed on exec.
- def pclose_all (i=None, o=None, e=None, logger=None)

  Closes all file descriptors sent to padd.
- def launch (cmd, env=None, stdin=None, stdout=None, stderr=None, debug=False, cd=None)

  Starts the specified command (a list), with the specified environment (or None to copy this process's environment).
- def filenoify (f)

  Tries to convert f to a fileno.
- def kill_for_thread (th)

  Sends a TERM signal to all processes that the specified thread (a threading.Thread) is waiting for.
- def kill_all ()

  Sends a TERM signal to all processes that this module is managing.
- def manage (proclist, inf=None, outf=None, errf=None, instr=None, logger=None, childset=None, sleeptime=None)

  Watches a list of processes, handles their I/O, returns when all processes have exited and all I/O is complete.
- def **simple_run** (cmd, env=None, stdin=None, stdout=None, stderr=None, debug=False, cd=None, logger=None)

Variables

- plock = threading.Lock()

  A global lock for this module.
- pipes_to_close = set()

  Set of pipes that must be closed after forking to avoid deadlocks.
- PIPE = Constant('PIPE')

  Indicates that stdout, stdin or stderr should be a pipe.
- ERR2OUT = Constant('ERR2OUT')

  Request that stderr and stdout be the same stream.

20.35.2 Function Documentation

20.35.2.1 call_fcntrl()

```
def produtil.pipeline.call_fcntrl (
              stream,
              on,
              off,
              logger = None )
```

Internal function that implements unblock()

Parameters

stream	the stream to modify
on	flags to turn on
off	flags to turn off
logger	a logging.Logger for messages

Returns

True on success, False otherwise.

Definition at line 70 of file pipeline.py.

Referenced by produtil.pipeline.pipe(), and produtil.pipeline.unblock().

20.35.2.2 filenoify()

```
def produtil.pipeline.filenoify (
              f )
```

Tries to convert f to a fileno.

Returns

an integer UNIX file descriptor

Parameters

f	ERR2OUT, PIPE, an integer fileno or a file-like object with a fileno() function.

Definition at line 290 of file pipeline.py.

Referenced by produtil.pipeline.manage().

20.35.2.3 kill_for_thread()

```
def produtil.pipeline.kill_for_thread (
              th )
```

Sends a TERM signal to all processes that the specified thread (a threading.Thread) is waiting for.

Definition at line 307 of file pipeline.py.

Referenced by produtil.workpool.WorkPool.kill_threads().

20.35.2.4 launch()

```
def produtil.pipeline.launch (
               cmd,
               env = None,
               stdin = None,
               stdout = None,
               stderr = None,
               debug = False,
               cd = None )
```

Starts the specified command (a list), with the specified environment (or None to copy this process's environment).

Parameters

stdin,stdout,stderr	Specifies the stdin, stdout and stderr streams. The special value PIPE means "make a pipe," and sending stderr=ERR2OUT requests redirection of stderr to stdout.
cd	The optional "cd" argument specifies a directory to cd into, in the child process, before executing the command. Of course, you shouldn't care about any of this because you should be using the produtil.run package.
cmd	the command to run
env	the subprocess's environment, or None to use mine
debug	if True, send debug messages

Definition at line 142 of file pipeline.py.

Referenced by produtil.pipeline.Pipeline.__repr__(), produtil.pipeline.manage(), and produtil.pipeline.pclose_all().

20.35.2.5 manage()

```
def produtil.pipeline.manage (
              proclist,
              inf = None,
              outf = None,
              errf = None,
              instr = None,
              logger = None,
              childset = None,
              sleeptime = None )
```

Watches a list of processes, handles their I/O, returns when all processes have exited and all I/O is complete.

Warning

You should not be calling this function unless you are modifying the implementation of Pipeline. Use the produtil.run module instead of calling launch() and manage().

Parameters

proclist	the list of processes to watch
inf	the input file

Parameters

outf	the output file
errf	the error file
instr	the input string, instead of an input file
childset	the set of child process ids
sleeptime	sleep time between checks of child processes
logger	Logs to the specified object, at level DEBUG, if a logger is specified.

Returns

a tuple containing the stdout string (or None), the stderr string (or None) and a dict mapping from process id to the return value from os.wait4 called on that process.

Definition at line 330 of file pipeline.py.

Referenced by produtil.pipeline.Pipeline.communicate(), and produtil.pipeline.kill_all().

20.35.2.6 padd()

```
def produtil.pipeline.padd (
              p )
```

Adds a file descriptor to the list to close before exec.

Parameters

p	the file descriptor

Definition at line 110 of file pipeline.py.

Referenced by produtil.pipeline.launch().

20.35.2.7 pclose()

```
def produtil.pipeline.pclose (
              i )
```

Closes a file descriptor, removing it from the list that must be closed on exec.

Parameters

i	the file descriptor

Definition at line 116 of file pipeline.py.

Referenced by produtil.pipeline.launch(), and produtil.pipeline.manage().

20.35.2.8 pclose_all()

```
def produtil.pipeline.pclose_all (
              i = None,
              o = None,
              e = None,
              logger = None )
```

Closes all file descriptors sent to padd.

Parameters

i	my stdin, which should not be closed
o	my stdout, which should not be closed
e	my stderr, which should not be closed
logger	a logging.Logger for debug messages

Definition at line 127 of file pipeline.py.

Referenced by produtil.pipeline.launch().

20.35.2.9 pipe()

```
def produtil.pipeline.pipe (
              logger = None )
```

Creates a pipe that will be closed on exec.

Except that it does not seem to be reliably closed on exec, so there are other workarounds in this module.

Parameters

logger	a logging.Logger for log messages

Definition at line 99 of file pipeline.py.

Referenced by produtil.pipeline.launch().

20.35.2.10 unblock()

```
def produtil.pipeline.unblock (
              stream,
              logger = None )
```

Attempts to modify the given stream to be non-blocking.

This only works with streams that have an underlying POSIX fileno, such as those from open.

Will re-raise any exception received, other than AttributeError and EnvironmentError. Hence, I/O errors and attempts to make a non-fileno stream non-blocking will produce a False return value, while anything else will raise an exception.

Parameters

stream	the stream to unblock
logger	a logging.Logger for log messages

Returns

True on success, False otherwise.

Definition at line 55 of file pipeline.py.

Referenced by produtil.pipeline.manage().

20.35.3 Variable Documentation

20.35.3.1 ERR2OUT

```
produtil.pipeline.ERR2OUT = Constant('ERR2OUT')
```

Request that stderr and stdout be the same stream.

Definition at line 54 of file pipeline.py.

20.35.3.2 PIPE

```
produtil.pipeline.PIPE = Constant('PIPE')
```

Indicates that stdout, stdin or stderr should be a pipe.

Definition at line 50 of file pipeline.py.

20.35.3.3 pipes_to_close

```
produtil.pipeline.pipes_to_close = set()
```

Set of pipes that must be closed after forking to avoid deadlocks.

Definition at line 46 of file pipeline.py.

20.35.3.4 plock

```
produtil.pipeline.plock = threading.Lock()
```

A global lock for this module.

Definition at line 42 of file pipeline.py.

20.36 produtil.prog Namespace Reference

Implements the produtil.run: provides the object tree for representing shell commands.

20.36.1 Detailed Description

Implements the produtil.run: provides the object tree for representing shell commands.

Do not load this module directly except for type checking (instanceof(o,produtil.prog.Runner)). It is meant to be used only by the produtil.run module. This module is part of the implementation of a shell-like syntax for running programs. The rest of the implementation is in the produtil.run and produtil.pipeline modules. MPI programs are implemented by the produtil.mpiprog and produtil.mpi_impl.

This module implements a shell-like syntax of running shell programs from Python. This module should not be used directly: the produtil.run implements critical parts of the functionality. Specifically, this module implements the Runner and ImmutableRunner classes. It also knows how to convert them to produtil.pipeline.Pipeline objects for actual execution.

- Runner — This class represents a process that could be run. It keeps track of all possible aspects of running a process, including the command, arguments, environment variables, stdout stream, stderr stream, stdin stream, and a list of functions or callable objects to run before executing the problem. Provides public functions to modify the Runner.
- ImmutableRunner — A Runner that cannot be changed: when modifying the Runner, it returns a new object. This is to implement shell aliases. For example, one could make an ImmutableRunner for program to index GRIB2 files. All the user would have to do is add the GRIB2 file as an argument, and capture the output.

Note that the actual work of creating the Runner or ImmutableRunner, or turning them into Pipeline objects done by the produtil.run module. Turning MPI programs into Runner objects is done by the produtil.mpiprog module and produtil.mpi_impl package, with the public interface in produtil.run. Hence, nobody would ever load this module directly, except for type checking (ie.: to see if your argument is a Runner before passing it to produtil.run.checkrun)..

Classes

- class EqualInEnv

  Raised when converting a Runner or pipeline of Runners to a POSIX sh string if there is an equal ("=") sign in an environment variable name.
- class EqualInExecutable

  Raised when converting a Runner or pipeline of Runners to a posix sh string if a Runner's executable contains an equal ("=") sign.
- class FileOpener

  This is part of the internal implementation of Runner, used to convert it to a produtil.pipeline.Pipeline for execution.
- class ImmutableRunner

  An copy-on-write version of Runner.
- class InvalidPipeline

  Raised when the caller specifies an invalid input or output when piping a Runner into or out of another object.
- class MultipleStderr

  Raised when the caller specifies more than one destination for a Runner's stderr.
- class MultipleStdin

  Raised when the caller specifies more than one source for the stdin of a Runner.
- class MultipleStdout

  Raised when the caller specifies more than one destination for a Runner's stdout.
- class NoSuchRedirection

  Raised when trying to convert a pipeline of Runners to a POSIX sh string, if a redirection in the pipeline cannot be expressed in POSIX sh.
- class NotValidPosixSh

  Base class of exceptions that are raised when converting a Runner or pipeline of Runners to a POSIX sh command, if the Runner cannot be expressed as POSIX sh.
- class NotValidPosixShString

  Raised when converting a Runner or pipeline of Runners to a POSIX sh string.
- class OutIsError

  Instructs a Runner to send stderr to stdout.
- class OverspecifiedStream

  Raised when one tries to specify the stdout, stderr or stdin to go to, or come from, more than one location.
- class ProgSyntaxError

  Base class of exceptions raised when a Runner is given arguments that make no sense.
- class Runner

  Represents a single stage of a pipeline to execute.
- class StreamGenerator

  This is part of the internal implementation of Runner, and is used to convert it to a produtil.pipeline.Pipeline for execution.
- class StreamReuser

  Arranges for a stream-like object to be sent to the stdout, stderr or stdin of a Runner.
- class StringInput

  Represents sending a string to a process's stdin.

Functions

- def shvarok (s)

  Returns True if the specified environment variable name is a valid POSIX sh variable name, and False otherwise.
- def shstrok (s)

  Returns True if the specified string can be expressed as a POSIX sh string, and false otherwise.
- def shbackslash (s)

  Given a Python str, returns a backslashed POSIX sh string, or raises NotValidPosixShString if that cannot be done.

20.36.2 Function Documentation

20.36.2.1 shbackslash()

```
def produtil.prog.shbackslash (
              s )
```

Given a Python str, returns a backslashed POSIX sh string, or raises NotValidPosixShString if that cannot be done.

Parameters

s	a string to backslash

Definition at line 105 of file prog.py.

Referenced by produtil.prog.FileOpener.to_shell(), produtil.prog.StringInput.to_shell(), produtil.prog.Runner.to_↩shell(), and produtil.mpiprog.MPIRank.to_shell().

20.36.2.2 shstrok()

```
def produtil.prog.shstrok (
              s )
```

Returns True if the specified string can be expressed as a POSIX sh string, and false otherwise.

Parameters

s	a string

Definition at line 95 of file prog.py.

Referenced by produtil.prog.shbackslash().

20.36.2.3 shvarok()

```
def produtil.prog.shvarok (
              s )
```

Returns True if the specified environment variable name is a valid POSIX sh variable name, and False otherwise.

Parameters

s	an environment variable name

Definition at line 86 of file prog.py.

20.37 produtil.retry Namespace Reference

Contains retry_io() which automates retrying operations.

20.37.1 Detailed Description

Contains retry_io() which automates retrying operations.

Functions

- def retry_io (max_tries, sleep_time, operation, opargs=[], logger=None, fail=None, failargs=[], giveup=None, giveupargs=[], randsleep=True, backoff=1.3, first_warn=0, giveup_quiet=False)

  This function automates retrying an unreliable operation several times until it succeeds.

20.37.2 Function Documentation

20.37.2.1 retry_io()

```
def produtil.retry.retry_io (
              max_tries,
              sleep_time,
              operation,
              opargs = [],
              logger = None,
              fail = None,
              failargs = [],
              giveup = None,
              giveupargs = [],
              randsleep = True,
              backoff = 1.3,
              first_warn = 0,
              giveup_quiet = False )
```

This function automates retrying an unreliable operation several times until it succeeds.

This subroutine will retry the operation up to a maximum number of times. If the operation fails too many times, then the last exception thrown by the operation is passed on (raised) to the caller.

Parameters

max_tries	Maximum number of times to attempt the operation (mandatory)
sleep_time	Time to sleep between tries
operation	A function or callable object that may thrown an Exception
opargs	A list containing arguments to the operation

Parameters

logger	A logging.Logger object to use for logging, or None to disable logging.
fail	A string to print, or a function to call, when the operation fails but more retries are possible
failargs	Optional: a list of arguments to fail, or None to disable
giveup	A string to print, or a function to call when the operation fails too many times, causing retry_io to give up. Default: same as fail
giveupargs	Optional: a list of arguments to giveup, or None to disable
randsleep	Set to True (default) to enable an exponential backoff algorithm, which will increase the sleep time between tries
backoff	The exponent for the exponential backoff algorithm
first_warn	The first failure at which to warn via the logger
giveup_quiet	If True, a WARNING-level message is sent to the logger if the operation fails more than max_tries times.

Returns

The return value of the operation.

Note

If fail or giveup are functions, they are passed the contents of failargs (default: opargs) or giveupargs (default: failargs or opargs) with several additional arguments appended. Those arguments are the exception that was caught, the number of attempts so far, the max_tries, the sleep_time, and then a boolean that is true iff the operation is about to be retried.

Definition at line 13 of file retry.py.

20.38 produtil.rstprod Namespace Reference

Handles data restriction classes.

20.38.1 Detailed Description

Handles data restriction classes.

Implements access control mechanisms for NOAA data. Although this was written for the NOAA Restricted Data (rstprod), it can be used for general access control. It is also more general than NOAA, so long as one correctly initializes the produtil.cluster module. The mechanism used depends on the cluster, due to varying capabilities throughout. Some do not implement access control mechanisms that are usable for the restricted data (such as NOAA Jet). For those systems, RstNoAccessControl is raised if one attempts to restrict a file.

Classes

- class RestrictionClass

  This is a python class intended to be used to automate restricting data to a specific restriction class using access control lists or group ownership.
- class RstBadGroup

  Raised when a group's id or name could not be determined.
- class RstNoAccessControl

  Raised when the cluster has no access control mechanisms.
- class RstprodError

  The base class of all exceptions specific to the rstprod module.

Functions

- def acl_text_for_rstclass (groupname, mode)

  Generates the access control list for the specified restriction class (groupname) and nine bit access permissions (mode).
- def make_rstprod_tagger (group='rstprod', use_acl=None, logger=None)

  Creates the rstprod_tagger object for use by tag_rstprod.
- def tag_rstprod (target, logger=None)

  Places a file or directory under the rstprod restriction class.

Variables

- okay_mode = stat.S_IRUSR|stat.S_IWUSR|stat.S_IXUSR | \

  File permission bits (from the stat module) that are allowed to be set on restricted access data.
- rstprod_tagger = None

  The RestrictionClass object used for tag_rstprod.

20.38.2 Function Documentation

20.38.2.1 acl_text_for_rstclass()

```
def produtil.rstprod.acl_text_for_rstclass (
              groupname,
              mode )
```

Generates the access control list for the specified restriction class (groupname) and nine bit access permissions (mode).

Parameters

groupname	the restricted file unix group
mode	required access mode (world access will be removed even if it is present in mode)

Definition at line 37 of file rstprod.py.

Referenced by produtil.rstprod.RestrictionClass.make_acl_dict().

20.38.2.2 tag_rstprod()

```
def produtil.rstprod.tag_rstprod (
              target,
              logger = None )
```

Places a file or directory under the rstprod restriction class.

This command will attempt to raise RstprodForbidden if it is run on a cluster that is not supposed to have rstprod data (only GAEA, Zeus and WCOSS are allowed).

This routine uses the approved rstprod protection mechanisms on each cluster:

- Zeus — place the file in the rstprod access control list, and make it unreadable to anyone else.
- WCOSS — place the file in group rstprod and remove permissions for others.
- GAEA — same as WCOSS

Note that the NOAA Jet cluster is not allowed to contain restricted data, so this routine will raise RstprodForbidden on that cluster.

Definition at line 279 of file rstprod.py.

20.38.3 Variable Documentation

20.38.3.1 okay_mode

```
produtil.rstprod.okay_mode = stat.S_IRUSR|stat.S_IWUSR|stat.S_IXUSR | \
```

File permission bits (from the stat module) that are allowed to be set on restricted access data.

When Access Control List (ACL) based access control is used, the group bits refer to the rstprod's permissions in the ACL, rather than the owning group.

Definition at line 34 of file rstprod.py.

20.38.3.2 rstprod_tagger

```
produtil.rstprod.rstprod_tagger = None
```

The RestrictionClass object used for tag_rstprod.

Create this with make_rstprod_tagger

Definition at line 272 of file rstprod.py.

20.39 produtil.run Namespace Reference

A shell-like syntax for running serial, MPI and OpenMP programs.

20.39.1 Detailed Description

A shell-like syntax for running serial, MPI and OpenMP programs.

This module implements a shell-like syntax for launching MPI and non-MPI programs from Python. It recognizes three types of executables: mpi, "small serial" (safe for running on a batch node) and "big serial" (which should be run via aprun if applicable). There is no difference between "small serial" and "big serial" programs except on certain architectures (like Cray) where the job script runs on a heavily-loaded batch node and has compute nodes assigned for running other programs.

20.39.2 Program Types

There are three types of programs: mpi, serial and "big non-MPI." A "big" executable is one that is either OpenMP, or is a serial program that cannot safely be run on heavily loaded batch nodes. On Cray architecture machines, the job script runs on a heavily-populated "batch" node, with some compute nodes assigned for "large" programs. In such environments, the "big" executables are run on compute nodes and the small ones on the batch node.

- mpi('exename') = an executable "exename" that calls MPI_Init and MPI_Finalize exactly once each, in that order.
- exe('exename') = a small non-MPI program safe to run on a batch node
- bigexe('exename') = a big non-MPI program that must be run on a compute node it may or may not use other forms of parallelism

You can also make reusable aliases to avoid having to call those functions over and over (more on that later). Examples:

- Python: wrf=mpi('./wrf.exe')
- Python: lsl=alias(exe('/bin/ls')['-l'].env(LANG='C',LS_COLORS='never'))

Those can then be reused later on as if the code is pasted in, similar to a shell alias.

20.39.3 Serial Execution Syntax

Select your serial programs by exe('name') for small serial programs and bigexe('name') for big serial programs. The return value of those functions can then be used with a shell-like syntax to specify redirection and piping. Example:

- shell version: ls -l / | wc -l
- Python version: run(exe('ls')['-l','/'] | exe('wc')['-l'])

Redirection syntax similar to the shell ($<$ $>$ and $<<$ operators):

```
run( ( exe('myprogram')['arg1','arg2','...'] < 'infile' ) > 'outfile')
```

Note the extra set of parentheses: you cannot do "exe('prog') $<$ infile

outfile" because of the order of precedence of Python operators

Append also works:

```
run(exe('myprogram')['arg1','arg2','...'] >> 'appendfile')
```

You can also send strings as input with <<

```
run(exe('myprogram')['arg1','arg2','...'] << 'some input string')
```

One difference from shells is that < and << always modify the beginning of the pipeline:

- shell: cat < infile | wc -l
- Python #1: (exe('cat') < 'infile') | exe('wc')['-l']
- Python #2: exe('cat') | (exe('wc')['-l'] < 'infile')

Note that the last second one, equivalent to `cat|wc -l<infile`, would NOT work in a shell since you would be giving wc -l two inputs.

20.39.4 Parallel Execution Syntax

Use mpi('exename') to select your executable, use [] to set arguments, use multiplication to set the number of ranks and use addition to combine different executables together into a multiple program multiple data (MPMD) MPI program.

Run ten copies of ls -l:

```
run(mpirun(mpiserial(('ls')['-l'])*10))
```

Run HyCOM coupled HWRF: one wm3c.exe, 30 hycom.exe and 204 wrf.exe:

```
run(mpirun(mpi('wm3c.exe') + mpi('hycom.exe')*30 + mpi('wrf.exe')*204))
```

You can set environment variables, pipe MPI output and handle redirection using the mpirun() function, which converts MPI programs into an bigexe()-style object (Runner):

Shell version:

```
result=$( mpirun -n 30 hostname | sort -u | wc -l )
```

Python version:

```
result=runstr( mpirun(mpi('hostname')*30) | exe['sort']['-u'] | exe['wc']['-l'] )
```

20.39.5 Aliases

If you find yourself frequently needing the same command, or you need to store a command for multiple uses, then then you should define an alias. Let's say you want "long output" format Japanese language "ls" output:

```
exe('ls')['-l','/path/to/dir'].env(LANG='JP')
```

but you find yourself running that on many different directories. Then you may want to make an alias:

```
jplsl=alias(exe('ls')['-l'].env(LANG='JP'))
```

The return value jplsl can be treated as an exe()-like return value since it was from exe() originally, but any new arguments will be appended to the original set:

```
run(jplsl['/path/to/dir'])
```

Note that if we did this:

```
badlsl=exe('ls')['-l'].env(LANG='JP')  # Bad! No alias!
run(badlsl['/'])  # will list /
run(badlsl['/home'])  # will list / and /home
run(badlsl['/usr/bin']) # will list / /home and /usr/bin

goodlsl=alias(exe('ls')['-l'].env(LANG='JP')
run(goodlsl['/'])  # will list /
run(goodlsl['/home'])  # will list /home
run(goodlsl['/usr/bin']) # will list /usr/bin
```

Then the run(badlsl['/home']) would list /home AND / which is NOT what we want. Why does it do that? It is because badlsl is not an alias — it is a regular output from exe(), so every time we call its [] operator, we add an argument to the original command. When we call alias() it returns a copy-on-write version (goodlsl), where every call to [] creates a new object.

Note that alias() also works with pipelines, but most operations will only modify the last the command in the pipeline (or the first, for operations that change stdin).

Classes

- class ExitStatusException

  Raised to indicate that a program generated an invalid return code.

- class InvalidRunArgument

  Raised to indicate that an invalid argument was sent into one of the run module functions.

Functions

- def alias (arg)

  Attempts to generate an unmodifiable "copy on write" version of the argument.
- def batchexe (name, kwargs)

  Returns a prog.ImmutableRunner object that represents a small serial program that can be safely run on a busy batch node.
- def exe (name, kwargs)

  Returns a prog.ImmutableRunner object that represents a large serial program that must be run on a compute node.
- def bigexe (name, kwargs)

  Alias for exe() for backward compatibility.
- def mpirun (arg, kwargs)

  Converts an MPI program specification into a runnable shell program suitable for run(), runstr() or checkrun().
- def make_pipeline (arg, capture, kwargs)

  This internal implementation function generates a prog.PopenCommand object for the specified input, which may be a prog.Runner or mpiprog.MPIRanksBase.
- def runbg (arg, capture=False, kwargs)

  Not implemented: background execution.
- def waitprocs (procs, logger=None, timeout=None, usleep=1000)

  Not implemented: background process monitoring.
- def runsync (logger=None)

  Runs the "sync" command as an exe().
- def run (arg, logger=None, sleeptime=None, kwargs)

  Executes the specified program and attempts to return its exit status.
- def checkrun (arg, logger=None, kwargs)

  This is a simple wrapper round run that raises ExitStatusException if the program exit status is non-zero.
- def openmp (arg, threads=None)

  Sets the number of OpenMP threads for the specified program.
- def runstr (arg, logger=None, kwargs)

  Executes the specified program or pipeline, capturing its stdout and returning that as a string.
- def mpi (arg, kwargs)

  Returns an MPIRank object that represents the specified MPI executable.
- def mpiserial (arg, kwargs)

  Generates an mpiprog.MPISerial object that represents an MPI rank that executes a serial (non-MPI) program.

Variables

- module_logger = logging.getLogger('produtil.run')

  Default logger used by some functions if no logger is given.

20.39.6 Function Documentation

20.39.6.1 alias()

```
def produtil.run.alias (
              arg )
```

Attempts to generate an unmodifiable "copy on write" version of the argument.

The returned copy will generate a modifiable duplicate of itself if you attempt to change it.

Returns

a produtil.prog.ImmutableRunner

Parameters

arg	a produtil.prog.Runner or produtil.prog.ImmutableRunner

Definition at line 220 of file run.py.

20.39.6.2 batchexe()

```
def produtil.run.batchexe (
              name,
              kwargs )
```

Returns a prog.ImmutableRunner object that represents a small serial program that can be safely run on a busy batch node.

Parameters

name	the executable name or path
kwargs	passed to produtil.prog.Runner.__init__

Returns

a new produtil.prog.ImmutableRunner

Definition at line 234 of file run.py.

20.39.6.3 bigexe()

```
def produtil.run.bigexe (
              name,
              kwargs )
```

Alias for exe() for backward compatibility.

Use exe() instead.

Definition at line 254 of file run.py.

20.39.6.4 checkrun()

```
def produtil.run.checkrun (
              arg,
              logger = None,
              kwargs )
```

This is a simple wrapper round run that raises ExitStatusException if the program exit status is non-zero.

Parameters

arg	the produtil.prog.Runner to execute (output of exe(), bigexe() or mpirun()
logger	a logging.Logger to log messages
kwargs	The optional run=[] argument can provide a different list of acceptable exit statuses.

Definition at line 398 of file run.py.

20.39.6.5 exe()

```
def produtil.run.exe (
              name,
              kwargs )
```

Returns a prog.ImmutableRunner object that represents a large serial program that must be run on a compute node.

Note

This function does NOT search $PATH on Cray. That ensures the $PATH will be expanded on the compute node instead. Use produtil.fileop.find_exe() if you want to explicitly search the PATH before execution.

Parameters

name	the executable name or path
kwargs	passed to produtil.prog.Runner.__init__

Returns

a new produtil.prog.ImmutableRunner

Definition at line 242 of file run.py.

Referenced by produtil.run.bigexe().

20.39.6.6 make_pipeline()

```
def produtil.run.make_pipeline (
              arg,
              capture,
              kwargs )
```

This internal implementation function generates a prog.PopenCommand object for the specified input, which may be a prog.Runner or mpiprog.MPIRanksBase.

Parameters

arg	the produtil.prog.Runner to convert. This is the output of exe(), bigexe() or mpirun()
capture	if True, capture the stdout into a string
kwargs	additional keyword arguments, same as for mpirun()

Definition at line 276 of file run.py.

Referenced by produtil.run.run(), produtil.run.runbg(), and produtil.run.runstr().

20.39.6.7 mpi()

```
def produtil.run.mpi (
              arg,
              kwargs )
```

Returns an MPIRank object that represents the specified MPI executable.

Parameters

arg	the MPI program to run
kwargs	logger=L for a logging.Logger to log messages

Definition at line 465 of file run.py.

20.39.6.8 mpirun()

```
def produtil.run.mpirun (
              arg,
              kwargs )
```

Converts an MPI program specification into a runnable shell program suitable for run(), runstr() or checkrun().

Options for kwargs:

- allranks=True — to run on all available MPI ranks. This cannot be used if a specific number of ranks (other than 1) was requested in the arg.
- logger=L — a logging.Logger for log messages
- Other platform-specific arguments. See produtil.mpi_impl for details.

Parameters

arg	the mpiprog.MPIRanksBase describing the MPI program to run. This is the output of the mpi() or mpiserial() function.
kwargs	additional arguments to control output.

Returns

a prog.Runner object for the specified mpiprog.MPIRanksBase object.

Definition at line 258 of file run.py.

20.39.6.9 mpiserial()

```
def produtil.run.mpiserial (
              arg,
              kwargs )
```

Generates an mpiprog.MPISerial object that represents an MPI rank that executes a serial (non-MPI) program.

The given value MUST be from bigexe() or exe(), NOT from mpi().

Parameters

arg	the MPI program to run
kwargs	logger=L for a logging.Logger to log messages

Definition at line 472 of file run.py.

20.39.6.10 openmp()

```
def produtil.run.openmp (
              arg,
              threads = None )
```

Sets the number of OpenMP threads for the specified program.

Warning

Generally, when using MPI with OpenMP, the batch system must be configured correctly to handle this or unexpected errors will result.

Parameters

arg	The "arg" argument must be from mpiserial, mpi, exe or bigexe.
threads	The optional "threads" argument is an integer number of threads. If it is not specified, the maximum possible number of threads will be used. Note that using threads=None with mpirun(...,allranks=True) will generally not work unless the batch system has already configured the environment correctly for an MPI+OpenMP task with default maximum threads and ranks.

Returns

see run()

Definition at line 415 of file run.py.

20.39.6.11 run()

```
def produtil.run.run (
              arg,
              logger = None,
              sleeptime = None,
              kwargs )
```

Executes the specified program and attempts to return its exit status.

In the case of a pipeline, the highest exit status seen is returned. For MPI programs, exit statuses are unreliable and generally implementation-dependent, but it is usually safe to assume that a program that runs MPI_Finalize() and exits normally will return 0, and anything that runs MPI_Abort(MPI_COMM_WORLD) will return non-zero. Programs that exit due to a signal will return statuses >255 and can be interpreted with WTERMSIG, WIFSIGNALLED, etc.

Parameters

arg	the produtil.prog.Runner to execute (output of exe(), bigexe() or mpirun()
logger	a logging.Logger to log messages
sleeptime	time to sleep between checks of child process
kwargs	ignored

Definition at line 376 of file run.py.

Referenced by produtil.run.checkrun().

20.39.6.12 runbg()

```
def produtil.run.runbg (
              arg,
              capture = False,
              kwargs )
```

Not implemented: background execution.

Runs the specified process in the background. Specify capture=True to capture the command's output. Returns a produtil.prog.PopenCommand. Call poll() to determine process completion, and use the stdout_data property to get the output after completion, if capture=True was specified.

Bug produtil.run.runbg() is not implemented

Warning

this is not implemented

Parameters

arg	the produtil.prog.Runner to execute (output of exe(), bigexe() or mpirun()
capture	if True, capture output
kwargs	same as for mpirun()

Definition at line 303 of file run.py.

20.39.6.13 runstr()

```
def produtil.run.runstr (
              arg,
              logger = None,
              kwargs )
```

Executes the specified program or pipeline, capturing its stdout and returning that as a string.

If the exit status is non-zero, then NonZeroExit is thrown.

Example:

```
runstr(exe('false'),ret=(1))
```

succeeds if "false" returns 1, and raises ExitStatusError otherwise.

Parameters

arg	The "arg" argument must be from mpiserial, mpi, exe or bigexe.
logger	a logging.Logger for logging messages
kwargs	You can specify an optional list or tuple "ret" that contains an alternative list of valid return codes. All return codes are zero or positive: negative values represent signal-terminated programs (ie.: SIGTERM produces -15, SIGKILL produces -9, etc.)

Definition at line 434 of file run.py.

20.39.6.14 runsync()

```
def produtil.run.runsync (
              logger = None )
```

Runs the "sync" command as an exe().

Definition at line 372 of file run.py.

20.39.6.15 waitprocs()

```
def produtil.run.waitprocs (
              procs,
              logger = None,
              timeout = None,
              usleep = 1000 )
```

Not implemented: background process monitoring.

Waits for one or more backgrounded processes to complete. Logs to the specified logger while doing so. If a timeout is specified, returns False after the given time if some processes have not returned. The usleep argument is the number of microseconds to sleep between checks (can be a fraction). The first argument, procs specifies the processes to check. It must be a produtil.prog.Pipeline (return value from runbg) or an iterable (list or tuple) of such.

Bug produtil.run.waitprocs() is untested

Warning

This is not tested and probably does not work.

Parameters

procs	the processes to watch
logger	the logging.Logger for log messages
timeout	how long to wait before giving up
usleep	sleep time between checks

Definition at line 324 of file run.py.

20.40 produtil.rusage Namespace Reference

This module allows querying resource usage and limits, as well as setting resource limits.

20.40.1 Detailed Description

This module allows querying resource usage and limits, as well as setting resource limits.

It is a wrapper around the Python resource module.

Setting resource limits:

```
use logging, produtil.rusage
logger=logging.logger("rusage")
produtil.rusage.setrlimit(logger,data=1e9,nofile=500,aspace=2e9,stack=5e8)
```

Printing resource limits to a logger:

```
use logging, produtil.rusage
logger=logging.logger("rusage")
u.produtil.rusage.getrlimit(logger) # writes the limits to the logger
# Limits are also in the returned object "u"
# Send None instead of logger to avoid logging.
```

Classes

- class RLimit

  Gets the resource limits set on this process: core, cpu, fsize, data, stack, rss, nproc, nofile, memlock, aspace Each is set to a tuple containing the soft and hard limit.
- class RUsage

  Contains resource usage (rusage) information that can be used with a Python "with" construct to collect the resources utilized by a block of code, or group of subprocesses executing during that block.
- class RUsageReport

  *Raised when caller makes an RUsage, and tries to generate its report, before calling its **enter** or **exit** routines.*

Functions

- def setrlimit (logger=None, ignore=False, hard=False, kwargs)

  Sets resource limits.
- def getrlimit (logger=None)

  Gets the current resource limits.

Variables

- rtypemap

  Maps the name used in this module for each resource class to the name used by the Python resource module.
- rnamemap

  Maps the name used in this module for each resource class, to a short human-readable string explaining the resource's meaning.
- tuple rusage_keys

  A tuple containing all rusage keys.
- rusage_meanings

  A mapping from rusage key to a human-readable explanation of the meaning.
- rusage = RUsage

  Alias for produtil.rusage.RUsage.

20.40.2 Function Documentation

20.40.2.1 getrlimit()

```
def produtil.rusage.getrlimit (
              logger = None )
```

Gets the current resource limits.

If logger is not None, sends the limits to the logger at level INFO.

Returns

a RLimit object with resource information

Definition at line 132 of file rusage.py.

20.40.2.2 setrlimit()

```
def produtil.rusage.setrlimit (
               logger = None,
               ignore = False,
               hard = False,
               kwargs )
```

Sets resource limits.

Parameters

ignore	If ignore=True, ignores any errors from getrlimit or setrlimit.
hard	If hard=True, attempts to set hard limits, which generally requires administrator privileges.
logger	The logger argument sets the logger (default: produtil.setrlimit logging domain).
kwargs	The kwargs should be a list of resource limits. Accepted resource limits: • core = core file size (RLIMIT_CORE) • cpu = max. cpu usage (RLIMIT_CPU) • fsize = max. file size (RLIMIT_FSIZE) • data = max. heap size (RLIMIT_DATA) • stack = max. stack size (RLIMIT_STACK) • rss = max. resident set size (RLIMIT_RSS) • nproc = max. processes (RLIMIT_NPROC) • nofile = max. open files (RLIMIT_NOFILE or RLIMIT_OFILE) • memlock= max locked memory (RLIMIT_MEMLOCK) • aspace = max. address space (RLIMIT_AS) See "man setrlimit" for details.

Definition at line 60 of file rusage.py.

20.40.3 Variable Documentation

20.40.3.1 rnamemap

```
produtil.rusage.rnamemap
```

Initial value:

```
1 = dict(core='core file size',
2               cpu='cpu usage',
3               fsize='max. file size',
4               data='max. heap size',
5               stack='max. stack size',
6               rss='max. resident set size',
7               nproc='max. processes',
8               nofile='max. open files',
9               #ofile='max. open files (BSD)',
10               memlock='max. locked memory',
11               #vmem='max. virtual memory',
12               aspace='max. address space')
```

Maps the name used in this module for each resource class, to a short human-readable string explaining the resource's meaning.

Definition at line 45 of file rusage.py.

20.40.3.2 rtypemap

produtil.rusage.rtypemap

Initial value:

```
1 = dict(core=resource.RLIMIT_CORE,
2                cpu=resource.RLIMIT_CPU,
3                fsize=resource.RLIMIT_FSIZE,
4                data=resource.RLIMIT_DATA,
5                stack=resource.RLIMIT_STACK,
6                rss=resource.RLIMIT_RSS,
7                nproc=resource.RLIMIT_NPROC,
8                nofile=resource.RLIMIT_NOFILE,
9                #ofile=resource.RLIMIT_OFILE,
10                memlock=resource.RLIMIT_MEMLOCK,
11                #vmem=resource.RLIMIT_VMEM,
12                aspace=resource.RLIMIT_AS)
```

Maps the name used in this module for each resource class to the name used by the Python resource module.

Definition at line 27 of file rusage.py.

20.40.3.3 rusage_keys

produtil.rusage.rusage_keys

Initial value:

```
1 = ('ru_utime','ru_stime','ru_maxrss','ru_ixrss','ru_idrss',
2              'ru_isrss','ru_minflt','ru_majflt','ru_nswap','ru_inblock',
3              'ru_oublock','ru_msgsnd','ru_msgrcv','ru_nsignals',
4              'ru_nvcsw','ru_nivcsw')
```

A tuple containing all rusage keys.

Definition at line 144 of file rusage.py.

20.40.3.4 rusage_meanings

produtil.rusage.rusage_meanings

Initial value:

```
1 = dict(ru_utime='time in user mode',
2                      ru_stime='time in system mode',
3                      ru_maxrss='maximum resident set size',
4                      ru_ixrss='shared memory size',
5                      ru_idrss='unshared memory size',
6                      ru_isrss='unshared stack size',
7                      ru_minflt='page faults not requiring I/O',
8                      ru_majflt='page faults requiring I/O',
9                      ru_nswap='number of swap outs',
10                      ru_inblock='block input operations',
11                      ru_oublock='block output operations',
12                      ru_msgsnd='messages sent',
13                      ru_msgrcv='messages received',
14                      ru_nsignals='signals received',
15                      ru_nvcsw='voluntary context switches',
16                      ru_nivcsw='involuntary context switches')
```

A mapping from rusage key to a human-readable explanation of the meaning.

Definition at line 152 of file rusage.py.

20.41 produtil.setup Namespace Reference

Contains setup(), which initializes the produtil package.

20.41.1 Detailed Description

Contains setup(), which initializes the produtil package.

This module contains the setup() function that should be called once by every Python process started, immediately after Python starts.

Functions

- def setup (ignore_hup=False, dbnalert_logger=None, jobname=None, cluster=None, send_dbn=None, thread_logger=False, thread_stack=2 ∗∗24, kwargs)

  Initializes the produtil package.

20.41.2 Function Documentation

20.41.2.1 setup()

```
def produtil.setup.setup (
               ignore_hup = False,
               dbnalert_logger = None,
               jobname = None,
               cluster = None,
               send_dbn = None,
               thread_logger = False,
               thread_stack = 2**24,
               kwargs )
```

Initializes the produtil package.

Calls the module initialization functions for all other modules in the produtil package.

At present, it:

1. Installs signal handlers that will cleanly abort the process.
2. Sets up logging to the jlogfile, if $jlogfile is in the environment.
3. Sets up logging to stdout and stderr.
4. Sets up the produtil.dbnalert module so DBNAlert objects will function properly
5. Sets the produtil.cluster's idea of what cluster it is on. If no cluster is specified, the produtil.cluster is instructed to guess.

This is a wrapper around the produtil.sigsafety, and produtil.log, and other module initializers. Note that one could call each module's initialization functions directly instead. However, one would have to keep up with changes to the produtil package during upgrades in order to do that.

Parameters

ignore_hup	if True, this program will ignore SIGHUP. Use this for UNIX daemon processes. Turned off (False) by default, causing SIGHUP to be a terminal signal.
dbnalert_logger	sent to dbnalert.init_module's logger argument to initialize the logging domain for informational messages about dbn alerts
jobname	dbn_alert job string
cluster	if specified and not None, sent to produtil.cluster's set_cluster. Otherwise, produtil.cluster.where() is called to guess the cluster, or set suitable defaults.
send_dbn	should dbn alerts be sent?
thread_logger	if True, log messages will include thread name.
thread_stack	passed to threading.stack_size(); the stack size in bytes for new threads. The default is 2**24, which is 16 MB. See the threading module for details. Set to None to disable changing of the threading stack size.
kwargs	all other keyword args sent to produtil.log.configureLogging()

Definition at line 15 of file setup.py.

Referenced by tc_pairs_wrapper.TcPairsWrapper.build_tc_pairs(), tc_stat_wrapper.TcStatWrapper.build_tc_stat(), series_by_lead_wrapper.SeriesByLeadWrapper.create_animated_gifs(), series_by_init_wrapper.SeriesByInitWrapper.create_fcst_anly_to_ascii_file(), extra_tropical_cyclone_plotter.ExtraTropicalCyclonePlotter.get_basemap(), tcmpr_plotter_wrapper.TCMPRPlotterWrapper.retrieve_optionals(), extract_tiles_wrapper.ExtractTilesWrapper.run_at_time(), and config_metplus.setup().

20.42 produtil.sigsafety Namespace Reference

Sets up signal handlers to ensure a clean exit.

20.42.1 Detailed Description

Sets up signal handlers to ensure a clean exit.

This module is a workaround for a deficiency of Python. When Python receives a fatal signal other than SIGINT, it exits immediately without freeing utilized resources or otherwise cleaning up. This module causes Python to raise a fatal exception, that does NOT derive from Exception, if a fatal signal is received. Note that there is a critical flaw in this design: raising an exception in a signal handler only raises it in the main (initial) thread. Other threads must call the produtil.sigsafety.checksig function as frequently as possible to check if a signal has been caught. That function will raise the appropriate exception if a signal was caught, or return immediately otherwise.

The reason this HAD to be added to produtil is that the lack of proper signal handling caused major problems. In particular, it completely broke file locking on Lustre and Panasas. Both filesystems will sometimes forget a file lock is released if the lock was held by a process that exited abnormally. There were also unverified cases of this happening with GPFS. Correctly handling SIGTERM, SIGQUIT, SIGHUP and SIGINT has solved that problem thus far.

The base class of any exception thrown due to a signal is CaughtSignal. It has two subclasses: FatalSignal, which is raised when a fatal signal is received, and HangupSignal. The HangupSignal is raised by SIGHUP, unless the install_handlers requests otherwise. Scripts should catch HangupSignal if the program is intended to ignore hangups. However, nothing should ever catch FatalSignal. Only **exit** and finalize blocks should be run in that situation, and they should run as quickly as possible.

The install_handlers installs the signal handlers: term_handler and optionally hup_handler. The raise_signals option specifies the list of signals that will raise FatalSignal, defaulting to SIGTERM, SIGINT and SIGQUIT. If SIGHUP is added to that list, then it will raise FatalSignal as well. Otherwise, the ignore_hup option controls the SIGHUP behavior: if True, SIGHUP is simply ignored, otherwise it raises HangupSignal.

One can call install_handlers directly, though it is recommended to call produtil.setup.setup instead.

Classes

- class CaughtSignal

  Base class of the exceptions thrown when a signal is caught.
- class FatalSignal

  Raised when a fatal signal is caught, as defined by the call to install_handlers.
- class HangupSignal

  With the default settings to install_handlers, this is raised when a SIGHUP is caught.

Functions

- def checksig ()

  This should be called frequently from worker threads to determine if the main thread has received a signal.
- def uninstall_handlers ()

  Resets all signal handlers to their system-default settings (SIG_DFL).
- def hup_handler (signum, frame)

  This is the signal handler for raising HangupSignal: it is used only for SIGHUP, and only if that is not specified in raise_signals and ignore_hup=False.
- def term_handler (signum, frame)

  This is the signal handler for raising FatalSignal.
- def install_handlers (ignore_hup=False, raise_signals=defaultsigs)

  Installs signal handlers that will raise exceptions.

Variables

- list defaultsigs = [signal.SIGTERM,signal.SIGINT,signal.SIGQUIT]

  Default signals for which to install terminal handlers.
- modifiedsigs = list()

  List of signals modified by install_handlers.
- caught_signal = None

  The signal number of the signal that was caught or None if no signal has been caught.
- caught_class = None

  The class that should be raised due to the caught signal, or None if no signal has been caught.

20.42.2 Function Documentation

20.42.2.1 checksig()

```
def produtil.sigsafety.checksig ( )
```

This should be called frequently from worker threads to determine if the main thread has received a signal.

If a signal was caught this function will raise the appropriate subclass of CaughtSignal. Otherwise, it returns None.

Definition at line 97 of file sigsafety.py.

Referenced by produtil.workpool.WorkPool.add_work().

20.42.2.2 hup_handler()

```
def produtil.sigsafety.hup_handler (
              signum,
              frame )
```

This is the signal handler for raising HangupSignal: it is used only for SIGHUP, and only if that is not specified in raise_signals and ignore_hup=False.

Parameters

signum,frame	signal information

Definition at line 129 of file sigsafety.py.

20.42.2.3 install_handlers()

```
def produtil.sigsafety.install_handlers (
```

```
                ignore_hup = False,
                raise_signals = defaultsigs )
```

Installs signal handlers that will raise exceptions.

Parameters

ignore_hup	If True, SIGHUP is ignored, else SIGHUP will raise HangupSignal
raise_signals	- List of exceptions that will raise FatalSignal. If SIGHUP is in this list, that overrides any decision made through ignore_hup.

Definition at line 153 of file sigsafety.py.

Referenced by produtil.setup.setup().

20.42.2.4 term_handler()

```
def produtil.sigsafety.term_handler (
                signum,
                frame )
```

This is the signal handler for raising FatalSignal.

Parameters

signum,frame	signal information

Definition at line 141 of file sigsafety.py.

20.42.2.5 uninstall_handlers()

```
def produtil.sigsafety.uninstall_handlers ( )
```

Resets all signal handlers to their system-default settings (SIG_DFL).

Does NOT restore the original handlers.

This function is a workaround for a design flaw in Python threading: you cannot kill a thread. This workaround restores default signal handlers after a signal is caught, ensuring the next signal will entirely terminate Python. Only the term_handler calls this function, so repeated hangups will still be ignored if the code desires it.

Some may note you can kill a Python thread on Linux using a private function but it is not available on all platforms and breaks GC. Another common workaround in Python is to use Thread.daemon, but that kills the thread immediately, preventing the thread from killing external processes or cleaning up other resources upon parent exit.

Definition at line 109 of file sigsafety.py.

Referenced by produtil.sigsafety.term_handler().

20.42.3 Variable Documentation

20.42.3.1 caught_class

```
produtil.sigsafety.caught_class = None
```

The class that should be raised due to the caught signal, or None if no signal has been caught.

This is initialized by the signal handlers, and used by checksig to raise exceptions due to caught signals.

Definition at line 95 of file sigsafety.py.

20.42.3.2 caught_signal

```
produtil.sigsafety.caught_signal = None
```

The signal number of the signal that was caught or None if no signal has been caught.

This is initialized by the signal handlers, and used by checksig to raise exceptions due to caught signals.

Definition at line 88 of file sigsafety.py.

20.42.3.3 defaultsigs

```
produtil.sigsafety.defaultsigs = [signal.SIGTERM,signal.SIGINT,signal.SIGQUIT]
```

Default signals for which to install terminal handlers.

Definition at line 48 of file sigsafety.py.

20.43 produtil.tempdir Namespace Reference

This module is an alias for produtil.cd, for backward compatibility.

20.43.1 Detailed Description

This module is an alias for produtil.cd, for backward compatibility.

20.44 produtil.workpool Namespace Reference

Contains the WorkPool class, which maintains pools of threads that perform small tasks.

20.44.1 Detailed Description

Contains the WorkPool class, which maintains pools of threads that perform small tasks.

Classes

- class WorkPool

  A pool of threads that perform some list of tasks.
- class WorkTask

  Stores a piece of work.
- class WrongThread

  Raised when a thread unrelated to a WorkPool attempts to interact with the WorkPool.

Functions

- def do_nothing ()

  Does nothing.

Variables

- TERMINATE = WorkTask(do_nothing)

  Special constant WorkTask used to terminate a WorkPool.

20.44.2 Function Documentation

20.44.2.1 do_nothing()

```
def produtil.workpool.do_nothing ( )
```

Does nothing.

Used to implement worker termination.

Definition at line 76 of file workpool.py.

20.44.3 Variable Documentation

20.44.3.1 TERMINATE

```
produtil.workpool.TERMINATE = WorkTask(do_nothing)
```

Special constant WorkTask used to terminate a WorkPool.

Do not modify.

Definition at line 82 of file workpool.py.

20.45 regrid_data_plane_wrapper Namespace Reference

20.45.1 Detailed Description

```
Program Name: regrid_data_plane.py
Contact(s): George McCabe
Abstract: Runs regrid_data_plane
History Log:  Initial version
Usage:
Parameters: None
Input Files: nc files
Output Files: nc files
Condition codes: 0 for success, 1 for failure
```

Classes

- class RegridDataPlaneWrapper

20.46 SeriesByLeadWrapper Namespace Reference

Performs any optional filtering of input tcst data then performs regridding via either MET regrid_data_plane or wgrib2, then builds up the commands to perform a series analysis by lead time by invoking the MET tool series_↩ analysis.

20.46.1 Detailed Description

Performs any optional filtering of input tcst data then performs regridding via either MET regrid_data_plane or wgrib2, then builds up the commands to perform a series analysis by lead time by invoking the MET tool series_↩ analysis.

NetCDF plots are generated by invoking the MET tool plot_data_plane. The NetCDF plots are then converted to .png and Postscript, and an animated GIF representative of the entire series is generated.

Call as follows:

```
SeriesByLeadWrapper.py [-c /path/to/user.template.conf]
```

20.47 string_template_substitution Namespace Reference

20.47.1 Detailed Description

```
Program Name: string_template_substitution.py
Contact(s): Julie Prestopnik, NCAR RAL DTC, jpresto@ucar.edu
Abstract: Supporting functions for parsing and creating filename templates
History Log:  Initial version for METPlus
Usage: Usually imported in other Python code for individual function calls
Parameters: Varies
Input Files: None
Output Files: None
Condition codes: Varies
```

Classes

- class StringExtract
- class StringSub

Functions

- def date_str_to_datetime_obj (str)
- def multiple_replace (dict, text)
- def get_lead_accum_time_seconds (logger, time_string)

Variables

- string **TEMPLATE_IDENTIFIER_BEGIN** = "{"
- string **TEMPLATE_IDENTIFIER_END** = "}"
- string **FORMATTING_DELIMITER** = "?"
- string **FORMATTING_VALUE_DELIMITER** = "="
- string **FORMAT_STRING** = "fmt"
- string **VALID_STRING** = "valid"
- string **LEAD_STRING** = "lead"
- string **INIT_STRING** = "init"
- string **ACCUM_STRING** = "accum"
- string **LEAD_ACCUM_FORMATTING_DELIMITER** = "%"
- int **SECONDS_PER_HOUR** = 3600
- int **MINUTES_PER_HOUR** = 60
- int **SECONDS_PER_MINUTE** = 60
- int **TWO_DIGIT_PAD** = 2
- int **THREE_DIGIT_PAD** = 3
- **GLOBAL_LOGGER** = None

20.47.2 Function Documentation

20.47.2.1 date_str_to_datetime_obj()

```
def string_template_substitution.date_str_to_datetime_obj (
              str )

Convert year month day string to a datetime object.
Works with YYYYMMDDHHMMSS, YYYYMMDDHHMM, YYYYMMDDHH, YYYYMMDD
```

Definition at line 48 of file string_template_substitution.py.

Referenced by string_template_substitution.StringSub.doStringSub().

20.47.2.2 get_lead_accum_time_seconds()

```
def string_template_substitution.get_lead_accum_time_seconds (
              logger,
              time_string )

Returns the number of seconds for the time string in the format
    [H]HH[MMSS]
```

Definition at line 81 of file string_template_substitution.py.

Referenced by string_template_substitution.StringSub.dateCalcInit(), and string_template_substitution.String↩Sub.dateCalcValid().

20.47.2.3 multiple_replace()

```
def string_template_substitution.multiple_replace (
              dict,
              text )

Replace in 'text' all occurences of any key in the
given dictionary by its corresponding value.  Returns the new string.
```

Definition at line 70 of file string_template_substitution.py.

Referenced by string_template_substitution.StringSub.doStringSub().

20.48 task_info Namespace Reference

20.48.1 Detailed Description

```
Program Name: task_info.py
Contact(s): George McCabe
Abstract:
History Log:  Initial version
Usage: Create a subclass
Parameters: None
Input Files: N/A
Output Files: N/A
```

Classes

- class TaskInfo

20.49 tc_pairs_wrapper Namespace Reference

20.49.1 Detailed Description

```
Program Name: tc_pairs_wrapper.py
Contact(s): Julie Prestopnik, Minna Win
Abstract: Invokes the MET tool tc_pairs to parse ADeck and BDeck ATCF files,
    filter the data, and match them up
History Log:  Initial version
Usage:
Parameters: None
Input Files: adeck and bdeck files
Output Files: tc_pairs files
Condition codes: 0 for success, 1 for failure
```

Classes

- class TcPairsWrapper

  Wraps the MET tool, tc_pairs to parse and match ATCF adeck and bdeck files.

Variables

- **send_dbn**
- **jobname**
- **jlogfile**
- **False**
- **CONFIG_INST** = config_metplus.setup()
- **TCP** = TcPairsWrapper(CONFIG_INST, logger=None)
- **exc_info**

20.50 tc_stat_wrapper Namespace Reference

Program Name: TcStatWrapper.py Contact(s): Julie Prestopnik, Minna Win Abstract: Subset tc_pairs data using MET tool TC-STAT for use in ExtractTiles.py or series analysis (via SeriesByLead.py or series_by_init.py) History log: Initial version Usage: TcStatWrapper.py Parameters: None Input Files: tc_pairs data Output Files: subset of tc_pairs data Condition codes: 0 for success, 1 for failure.

20.50.1 Detailed Description

Program Name: TcStatWrapper.py Contact(s): Julie Prestopnik, Minna Win Abstract: Subset tc_pairs data using MET tool TC-STAT for use in ExtractTiles.py or series analysis (via SeriesByLead.py or series_by_init.py) History log: Initial version Usage: TcStatWrapper.py Parameters: None Input Files: tc_pairs data Output Files: subset of tc_pairs data Condition codes: 0 for success, 1 for failure.

Classes

- class TcStatWrapper

  Wrapper for the MET tool, tc_stat, which is used to filter tropical cyclone pair data.

Variables

- **send_dbn**
- **False**
- **jobname**
- **jlogfile**
- **CONFIG** = config_launcher.load_baseconfs(sys.argv[2])
- **TCS** = TcStatWrapper(CONFIG)
- **exc_info**

20.51 TCMPRPlotterWrapper Namespace Reference

A Python class than encapsulates the plot_tcmpr.R plotting script.

20.51.1 Detailed Description

A Python class than encapsulates the plot_tcmpr.R plotting script.

Generates plots for input files with .tcst format and creates output subdirectory based on the input tcst file. The plot_tcmpr.R plot also supports additional filtering by calling MET tool tc_stat. This wrapper extends plot_tcmpr.R by allowing the user to specify as input a directory (to support plotting all files in the specified directory and its subdirectories). The user can now either indicate a file or directory in the (required) -lookin option. Call as follows:

```
tcmpr_plotter_wrapper.py [-c /path/to/user.template.conf]
```

20.52 TcStatWrapper Namespace Reference

Wrapper to the MET tool tc_stat, which is used for filtering tropical cyclone pair data.

20.52.1 Detailed Description

Wrapper to the MET tool tc_stat, which is used for filtering tropical cyclone pair data.

20.53 UsageWrapper Namespace Reference

Provides a default process for master_metplus.py.

20.53.1 Detailed Description

Provides a default process for master_metplus.py.

Indicates what processes are currently available. Call as follows:

```
usage_wrapper.py [-c /path/to/user.template.conf]
```

20.54 ush Namespace Reference

METplus utility scripts for wrapping MET.

20.54.1 Detailed Description

METplus utility scripts for wrapping MET.

The ush directory is the resting place of most of the Python scripts in the METplus package. It contains the main utility scripts for calling and wrapping the MET tools suite.

Utility scripts of general interest:

master_met_plus — This is the main script that calls all the tasks in the PROCESS_LIST.

run_tc_pairs — Runs tc_pairs to parse ADeck and BDeck ATCF files, filter the data, and match them up.

extract_tiles — Extracts tiles to be used by series_analysis.

series_by_lead — Perform a series analysis of extra tropical cyclone paired data based on lead time (forecast hour).

series_by_init — Invoke the series analysis script based on the init time in the format YYYYMMDD_hh.

run_tc_stat — Subset tc_pairs data using MET tool TC-STAT for use in extract_tiles or series analysis.

tc2cyclone_relative — Convert MET TC-Pairs output (A-deck and B-deck track files) into format for SBU cyc-new.dat and match.dat.

TCMPRPlotter — A Python wrapper to the plot_tcmpr.R plotting script Generates plots for input files with .tcst format and creates output subdirectory based on the input tcst file.

met_util — A collection of utility functions used to perform necessary series analysis tasks and other METPlus related tasks.

Reference links to the produtil package

Package "produtil" — The produtil Python package creates a platform-independent environment for running METplus.

produtil.config — Parses UNIX conf files and makes the result readily available. This is part of the produtil package and is referenced here as a convenience.

Chapter 21

Class Documentation

21.1 produtil.acl.ACL Class Reference

ACL class wrapped around the libacl library:

21.1.1 Detailed Description

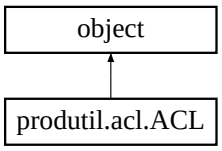
ACL class wrapped around the libacl library:

Inquire and manipulate access control lists (ACLs).

Represents a POSIX Access Control List (ACL). This is a wrapper around the libacl library, and implements only widely-supported ACL features. Data is stored internally in C structures, which are allocated and freed automatically as needed.

Definition at line 139 of file acl.py.

Inheritance diagram for produtil.acl.ACL:



Public Member Functions

- def __init__ (self)

  Create a blank, invalid, ACL.
- def __del__ (self)

  Free the memory used by the ACL in libacl.
- def free (self)

  Frees resources used by the libacl library to store this ACL's underlying C structures.
- def have_acl (self)

  Returns True if this ACL has data, and False otherwise.

- def from_text (self, acl)

  Attempts to convert the given string to an ACL, storing the result in this object.
- def from_file (self, filename, which=ACL_TYPE_ACCESS)

  Copies the files's ACL into this object.
- def from_fd (self, fd)

  Get an access control list from a file descriptor.
- def to_fd (self, fd)

  Updates a file's file descriptor.
- def to_file (self, filename, access=ACL_TYPE_ACCESS)

  Updates a file's access control list.
- def to_text (self)

  Converts an ACL to text.
- def __str__ (self)

  => self.to_text()

21.1.2 Constructor & Destructor Documentation

21.1.2.1 __init__()

```
def produtil.acl.ACL.__init__ (
              self )
```

Create a blank, invalid, ACL.

You should use the various from_∗ routines to fill it with valid data.

Definition at line 146 of file acl.py.

21.1.2.2 __del__()

```
def produtil.acl.ACL.__del__ (
              self )
```

Free the memory used by the ACL in libacl.

Definition at line 152 of file acl.py.

21.1.3 Member Function Documentation

21.1.3.1 free()

```
def produtil.acl.ACL.free (
              self )
```

Frees resources used by the libacl library to store this ACL's underlying C structures.

Definition at line 155 of file acl.py.

Referenced by produtil.acl.ACL.__del__(), produtil.acl.ACL.from_file(), and produtil.acl.ACL.from_text().

21.1.3.2 from_fd()

```
def produtil.acl.ACL.from_fd (
              self,
              fd )
```

Get an access control list from a file descriptor.

Obtains an Access Control List from the specified file object or file descriptor number. You can also pass any object that has a "fileno()" member function. Any prior ACL information in this object will be freed.

Parameters

fd	an integer file descriptor or a file object.

Returns

self

Definition at line 205 of file acl.py.

21.1.3.3 from_file()

```
def produtil.acl.ACL.from_file (
              self,
              filename,
              which = ACL_TYPE_ACCESS )
```

Copies the files's ACL into this object.

Specify which type of access control list via the second argument: ACL_TYPE_ACCESS or ACL_TYPE_DEFAULT. Any prior ACL information in this object will be freed.

Parameters

filename	the name of the file whose ACL is desired
which	which access control list is desired; ACL_TYPE_ACCESS or ACL_TYPE_DEFAULT.

Returns

self

Definition at line 185 of file acl.py.

21.1.3.4 from_text()

```
def produtil.acl.ACL.from_text (
              self,
              acl )
```

Attempts to convert the given string to an ACL, storing the result in this object.

Any prior ACL information in this object will be freed.

Parameters

acl	the access control list description

Definition at line 167 of file acl.py.

21.1.3.5 have_acl()

```
def produtil.acl.ACL.have_acl (
              self )
```

Returns True if this ACL has data, and False otherwise.

Definition at line 164 of file acl.py.

21.1.3.6 to_fd()

```
def produtil.acl.ACL.to_fd (
              self,
              fd )
```

Updates a file's file descriptor.

Sets the ACL for the specified file descriptor to the ACL stored in this object. Raises ACLMissingError if this object has no ACL information.

Parameters

fd	an integer file descriptor or open file object

Definition at line 224 of file acl.py.

21.1.3.7 to_file()

```
def produtil.acl.ACL.to_file (
              self,
              filename,
              access = ACL_TYPE_ACCESS )
```

Updates a file's access control list.

Sets the ACL for the specified file to the ACL stored in this object. Specify access=ACL_TYPE_DEFAULT to obtain the default access control list (Default ACL) or ACL_TYPE_ACCESS for the access control list. Raises ACLMissingError if this object has no ACL information.

Parameters

filename	the name of the file whose ACL is to be updated
access	ACL_TYPE_ACCESS or ACL_TYPE_DEFAULT

Definition at line 245 of file acl.py.

21.1.3.8 to_text()

```
def produtil.acl.ACL.to_text (
              self )
```

Converts an ACL to text.

Returns a string representation of this ACL from acl_to_text. Returns the empty string ('') if this ACL has no data.

Definition at line 268 of file acl.py.

Referenced by produtil.acl.ACL.__str__().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/acl.py

21.2 produtil.acl.ACLCannotGet Class Reference

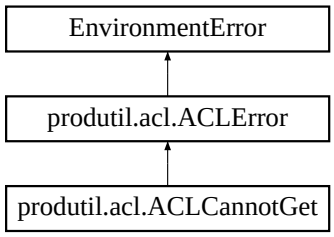
Raised when the libacl library could not get a file's ACL.

21.2.1 Detailed Description

Raised when the libacl library could not get a file's ACL.

Definition at line 33 of file acl.py.

Inheritance diagram for produtil.acl.ACLCannotGet:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/acl.py

21.3 produtil.acl.ACLCannotSet Class Reference

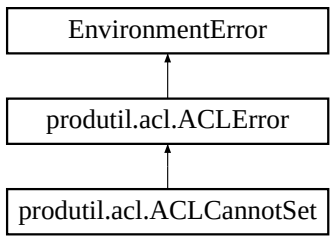
Raised when the libacl library could not set a file's ACL.

21.3.1 Detailed Description

Raised when the libacl library could not set a file's ACL.

Definition at line 35 of file acl.py.

Inheritance diagram for produtil.acl.ACLCannotSet:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/acl.py

21.4 produtil.acl.ACLCannotStringify Class Reference

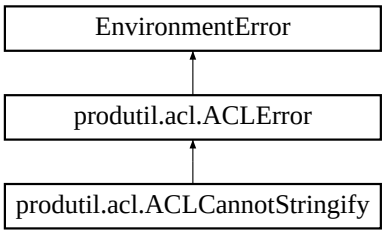
Raised when libacl cannot convert an ACL to text.

21.4.1 Detailed Description

Raised when libacl cannot convert an ACL to text.

Definition at line 31 of file acl.py.

Inheritance diagram for produtil.acl.ACLCannotStringify:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/acl.py

21.5 produtil.acl.ACLError Class Reference

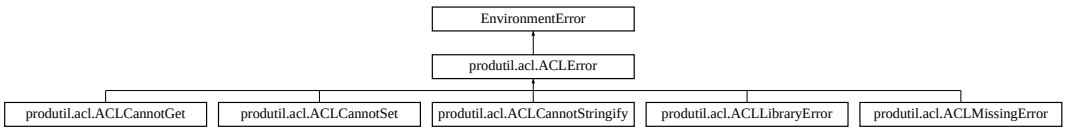
Superclass of any ACL errors.

21.5.1 Detailed Description

Superclass of any ACL errors.

Definition at line 15 of file acl.py.

Inheritance diagram for produtil.acl.ACLError:



Public Member Functions

- def __init__ (self, message, errno)

  ACLError constructor.

Public Attributes

- errno

  The errno value when the error happened.

21.5.2 Constructor & Destructor Documentation

21.5.2.1 __init__()

```
def produtil.acl.ACLError.__init__ (
              self,
              message,
              errno )
```

ACLError constructor.

Parameters

message	the description of the error
errno	the system errno from the error

Definition at line 17 of file acl.py.

21.5.3 Member Data Documentation

21.5.3.1 errno

```
produtil.acl.ACLError.errno
```

The errno value when the error happened.

Definition at line 22 of file acl.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/acl.py

21.6 produtil.acl.ACLLibraryError Class Reference

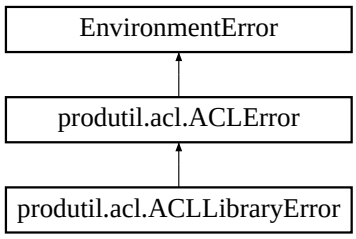
Raised when the libacl library could not be loaded.

21.6.1 Detailed Description

Raised when the libacl library could not be loaded.

Definition at line 26 of file acl.py.

Inheritance diagram for produtil.acl.ACLLibraryError:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/acl.py

21.7 produtil.acl.ACLMissingError Class Reference

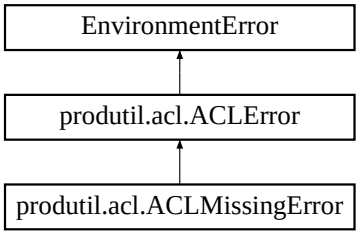
Raised when a function that requires an ACL object received None, or an invalid ACL.

21.7.1 Detailed Description

Raised when a function that requires an ACL object received None, or an invalid ACL.

Definition at line 28 of file acl.py.

Inheritance diagram for produtil.acl.ACLMissingError:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/acl.py

21.8 produtil.atparse.ATParser Class Reference

Takes input files or other data, and replaces certain strings with variables or functions.

21.8.1 Detailed Description

Takes input files or other data, and replaces certain strings with variables or functions.

The calling convention is quite simple:

```
ap=ATParser(varhash={"NAME":"Katrina", "STID":"12L"})
ap.parse_file("input-file.txt")
lines="line 1\nline 2\nline 3 of @[NAME]"
ap.parse_lines(lines,"(string-data)")
ap.parse_stream(sys.stdin,"(stdin)")
```

Inputs are general strings with @[...] and @∗∗ escape sequences which follow familiar shell syntax (but with @[...] instead of ${...}):

```
My storm is @[NAME] and the RSMC is @[RSMC:-${center:-unknown}].
```

In this case, it would print:

```
My storm is Katrina and the RSMC is unknown.
```

since NAME is set, but RSMC and center are unset.

There are also block if statements:

```
@** if NAME==BILLY
storm is billy
@** elseif name==KATRINA
storm is katrina
@** else
another storm
@** endif
```

and a variety of other things:

```
@[<anotherfile.txt]  # read another file
@[var=value] # assign a variable
@[var:=value] # assign a variable and insert the value in the output stream
@[var2:?] # abort if var2 is not assigned, otherwise insert var2's contents
@[var3==BLAH?thencondition:elsecondition] # if-then-else substitution
@[var3!=BLAH?thencondition:elsecondition] # same, but with a "not equal"
@[var4:-substitution] # insert var4, or this substitution if var4 is unset
@[var5:+text] # insert text if var5 is set
```

There are also a small number of functions that modify text before it is sent to stdout. (The original variable is unmodified, only the output text is changed.)

```
@[var1.uc] # uppercase value of var1
@[var1.lc] # lowercase value of var1
@[var1.len] # length of var1
@[var1.trim] # var1 with leading and trailing whitespace removed
```

Definition at line 98 of file atparse.py.

Public Member Functions

- def __init__ (self, stream=sys.stdout, varhash=None, logger=None, max_lines=1000000)

  ATParser constructor.
- def max_lines (self)

  The maximum number of lines to read.
- def infile (self)

  The current input file name.
- def parse_stream (self, stream, streamname)

  Read a stream and parse its contents.
- def parse_file (self, filename)

  Read a file and parse its contents.
- def getvar (self, varname)

  Return the value of a variable, or None if it is unset.
- def str_state (self)

  Return a string description of the parser stack for debugging.
- def parse_lines (self, lines, filename)

  Given a multi-line string, parse the contents line-by-line.
- def parse_line (self, line, filename, lineno)

  Parses one line of text.

Public Attributes

- varhash

  The dict of variables.

Protected Member Functions

- def warn (self, text)

  Print a warning to the logger, if we have a logger.
- def applyfun (self, val, fun1, morefun)

  Applies a function to text.
- def from_var (self, varname, optional)

  Return the value of a variable with functions applied.
- def optional_var (self, varname)

  Return the value of a variable with functions applied, or '' if the variable is unset.
- def require_var (self, varname)

  Return the value of a variable with functions applied, raising an exception if the variable is unset.
- def replace_vars (self, text)

  Expand @[...] blocks in a string.
- def require_file (self, filename_pattern)

  Read the contents of a file and return it.
- def var_or_command (self, data)

  Expand one ${...} or @[...] block.
- def require_data (self, data)

  Expand text within an @[...] block.
- def active (self)

  Is the current block active?
- def top_state (self, what=None)

Return the top parser state without removing it.

- def push_state (self, state)

  Push a new state to the top of the parser state stack.

- def pop_state (self)

  Remove and return the top parser state.

- def replace_state (self, state)

  Replace the top parser state.

21.8.2 Constructor & Destructor Documentation

21.8.2.1 __init__()

```
def produtil.atparse.ATParser.__init__ (
              self,
              stream = sys.stdout,
              varhash = None,
              logger = None,
              max_lines = 1000000 )
```

ATParser constructor.

Parameters

stream	the output stream
varhash	a dict of variables. All values must be strings. If this is unspecified, os.environ will be used.
logger	the logging.Logger to read.
max_lines	the maximum number of lines to read

Definition at line 156 of file atparse.py.

21.8.3 Member Function Documentation

21.8.3.1 applyfun()

```
def produtil.atparse.ATParser.applyfun (
              self,
              val,
              fun1,
              morefun )  [protected]
```

Applies a function to text.

Parameters

val	the text
fun1	the function to apply
morefun	more functions to apply

Definition at line 195 of file atparse.py.

Referenced by produtil.atparse.ATParser.applyfun(), produtil.atparse.ATParser.from_var(), and produtil.atparse.↩ ATParser.var_or_command().

21.8.3.2 from_var()

```
def produtil.atparse.ATParser.from_var (
              self,
              varname,
              optional )  [protected]
```

Return the value of a variable with functions applied.

Parameters

varname	the variable name, including functions
optional	if False, raise an exception if the variable is unset. If True, return '' for unset variables.

Definition at line 215 of file atparse.py.

Referenced by produtil.atparse.ATParser.from_var(), produtil.atparse.ATParser.optional_var(), and produtil.↩ atparse.ATParser.require_var().

21.8.3.3 getvar()

```
def produtil.atparse.ATParser.getvar (
              self,
              varname )
```

Return the value of a variable, or None if it is unset.

Definition at line 287 of file atparse.py.

Referenced by produtil.atparse.ATParser.var_or_command().

21.8.3.4 infile()

```
def produtil.atparse.ATParser.infile (
              self )
```

The current input file name.

Definition at line 188 of file atparse.py.

Referenced by produtil.atparse.ATParser.from_var().

21.8.3.5 max_lines()

```
def produtil.atparse.ATParser.max_lines (
              self )
```

The maximum number of lines to read.

Definition at line 184 of file atparse.py.

Referenced by produtil.atparse.ATParser.parse_line().

21.8.3.6 optional_var()

```
def produtil.atparse.ATParser.optional_var (
              self,
              varname )  [protected]
```

Return the value of a variable with functions applied, or '' if the variable is unset.

Parameters

varname	the name of the variable.

Definition at line 233 of file atparse.py.

Referenced by produtil.atparse.ATParser.parse_line().

21.8.3.7 parse_file()

```
def produtil.atparse.ATParser.parse_file (
              self,
              filename )
```

Read a file and parse its contents.

Parameters

filename	the name of this file for error messages

Definition at line 270 of file atparse.py.

21.8.3.8 parse_line()

```
def produtil.atparse.ATParser.parse_line (
              self,
              line,
              filename,
              lineno )
```

Parses one line of text.

Parameters

line	the line of text.
filename	the name of the source file, for error messages
lineno	the line number within the source file, for error messages

Definition at line 462 of file atparse.py.

Referenced by produtil.atparse.ATParser.parse_file(), produtil.atparse.ATParser.parse_lines(), and produtil.↩ atparse.ATParser.parse_stream().

21.8.3.9 parse_lines()

```
def produtil.atparse.ATParser.parse_lines (
              self,
              lines,
              filename )
```

Given a multi-line string, parse the contents line-by-line.

Parameters

lines	the multi-line string
filename	the name of the file it was from, for error messages

Definition at line 453 of file atparse.py.

Referenced by produtil.atparse.ATParser.parse_line().

21.8.3.10 parse_stream()

```
def produtil.atparse.ATParser.parse_stream (
              self,
              stream,
              streamname )
```

Read a stream and parse its contents.

Parameters

stream	the stream (an opened file)
streamname	a name for this stream for error messages

Definition at line 261 of file atparse.py.

21.8.3.11 replace_state()

```
def produtil.atparse.ATParser.replace_state (
              self,
              state )  [protected]
```

Replace the top parser state.

Parameters

state	the new parser state

Definition at line 447 of file atparse.py.

Referenced by produtil.atparse.ATParser.parse_line().

21.8.3.12 replace_vars()

```
def produtil.atparse.ATParser.replace_vars (
              self,
              text )  [protected]
```

Expand @[...] blocks in a string.

Parameters

text	the string

Returns

a new string with expansions performed

Definition at line 247 of file atparse.py.

Referenced by produtil.atparse.ATParser.parse_line(), produtil.atparse.ATParser.require_file(), and produtil.↩ atparse.ATParser.var_or_command().

21.8.3.13 require_data()

```
def produtil.atparse.ATParser.require_data (
              self,
              data )  [protected]
```

Expand text within an @[...] block.

Parameters

data	the contents of the block

Definition at line 373 of file atparse.py.

Referenced by produtil.atparse.ATParser.parse_line().

21.8.3.14 require_file()

```
def produtil.atparse.ATParser.require_file (
              self,
              filename_pattern )  [protected]
```

Read the contents of a file and return it.

Parameters

filename_pattern	a filename with ${} or @[] blocks in it.

Definition at line 279 of file atparse.py.

Referenced by produtil.atparse.ATParser.parse_line(), and produtil.atparse.ATParser.require_data().

21.8.3.15 require_var()

```
def produtil.atparse.ATParser.require_var (
              self,
              varname )  [protected]
```

Return the value of a variable with functions applied, raising an exception if the variable is unset.

Parameters

varname	the name of the variable.

Definition at line 240 of file atparse.py.

Referenced by produtil.atparse.ATParser.replace_vars(), and produtil.atparse.ATParser.var_or_command().

21.8.3.16 str_state()

```
def produtil.atparse.ATParser.str_state (
              self )
```

Return a string description of the parser stack for debugging.

Definition at line 390 of file atparse.py.

21.8.3.17 top_state()

```
def produtil.atparse.ATParser.top_state (
              self,
              what = None )  [protected]
```

Return the top parser state without removing it.

Parameters

what	why the state is being examined. This is for error messages.

Definition at line 423 of file atparse.py.

Referenced by produtil.atparse.ATParser.parse_line().

21.8.3.18 var_or_command()

```
def produtil.atparse.ATParser.var_or_command (
              self,
              data )  [protected]
```

Expand one ${...} or @[...] block.

Parameters

data	the contents of the block

Definition at line 291 of file atparse.py.

Referenced by produtil.atparse.ATParser.replace_vars(), and produtil.atparse.ATParser.require_data().

21.8.3.19 warn()

```
def produtil.atparse.ATParser.warn (
              self,
              text )  [protected]
```

Print a warning to the logger, if we have a logger.

Parameters

text	the warning text.

Definition at line 177 of file atparse.py.

Referenced by produtil.atparse.ATParser.applyfun(), and produtil.atparse.ATParser.parse_line().

21.8.4 Member Data Documentation

21.8.4.1 varhash

```
produtil.atparse.ATParser.varhash
```

The dict of variables.

This is NOT the dict sent to the constructor — a copy was made. That means it is safe to modify the variables all you want, even if os.environ was used.

Definition at line 164 of file atparse.py.

Referenced by produtil.atparse.ATParser.from_var(), produtil.atparse.ATParser.getvar(), and produtil.atparse.AT↩ Parser.var_or_command().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/atparse.py

21.9 produtil.datastore.CallbackExceptions Class Reference

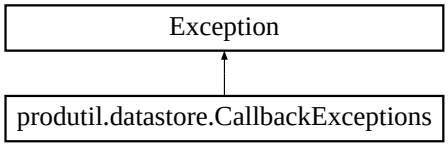
Exception raised when a Product class encounters exceptions while calling its callback functions in Product.call_callbacks.

21.9.1 Detailed Description

Exception raised when a Product class encounters exceptions while calling its callback functions in Product.call_callbacks.

Definition at line 692 of file datastore.py.

Inheritance diagram for produtil.datastore.CallbackExceptions:



Public Member Functions

- def __init__ (self, message, exlist)

  CallbackExceptions constructor.

Public Attributes

- messagebase

  The original message sent to the constructor before per-exception messages were appended.
- exlist

  The list of exceptions raised.

21.9.2 Constructor & Destructor Documentation

21.9.2.1 __init__()

```
def produtil.datastore.CallbackExceptions.__init__ (
              self,
              message,
              exlist )
```

CallbackExceptions constructor.

Parameters

message	The beginning of the exception message. Each exception's message will be appended to this.
exlist	The list of exceptions.

Definition at line 696 of file datastore.py.

21.9.3 Member Data Documentation

21.9.3.1 exlist

`produtil.datastore.CallbackExceptions.exlist`

The list of exceptions raised.

Definition at line 702 of file datastore.py.

21.9.3.2 messagebase

`produtil.datastore.CallbackExceptions.messagebase`

The original message sent to the constructor before per-exception messages were appended.

Definition at line 701 of file datastore.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.10 produtil.fileop.CannotFindExe Class Reference

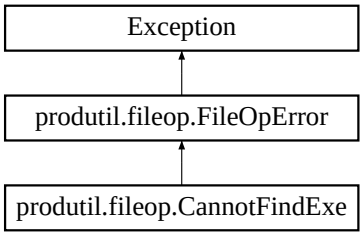
Thrown when find_exe cannot find an executable in the path or directory list.

21.10.1 Detailed Description

Thrown when find_exe cannot find an executable in the path or directory list.

Definition at line 74 of file fileop.py.

Inheritance diagram for produtil.fileop.CannotFindExe:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.11 produtil.fileop.CannotLinkMulti Class Reference

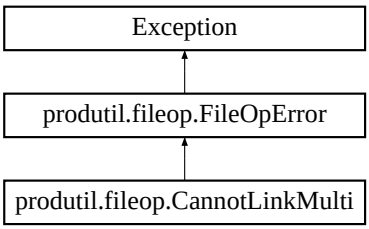
This exception is raised when the caller tries to create multiple symlinks in a single target, but the target is not a directory.

21.11.1 Detailed Description

This exception is raised when the caller tries to create multiple symlinks in a single target, but the target is not a directory.

Definition at line 62 of file fileop.py.

Inheritance diagram for produtil.fileop.CannotLinkMulti:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.12 produtil.sigsafety.CaughtSignal Class Reference

Base class of the exceptions thrown when a signal is caught.

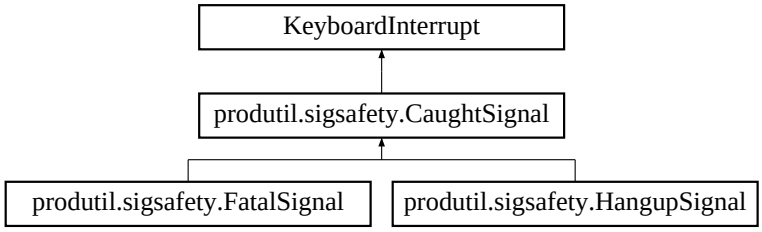
21.12.1 Detailed Description

Base class of the exceptions thrown when a signal is caught.

Note that this does not derive from Exception, to ensure it is not caught accidentally. At present, it derives directly from KeyboardInterrupt, though that may be changed in the future to BaseException.

Definition at line 58 of file sigsafety.py.

Inheritance diagram for produtil.sigsafety.CaughtSignal:



Public Member Functions

- def __init__ (self, signum)
  CaughtSignal constructor.
- def __str__ (self)
  A string description of this error.

Public Attributes

- signum
  The integer signal number.

21.12.2 Constructor & Destructor Documentation

21.12.2.1 __init__()

```
def produtil.sigsafety.CaughtSignal.__init__ (
            self,
            signum )
```

CaughtSignal constructor.

Parameters

signum	the signal that was caught (an int)

Definition at line 64 of file sigsafety.py.

21.12.3 Member Function Documentation

21.12.3.1 __str__()

```
def produtil.sigsafety.CaughtSignal.__str__ (
              self )
```

A string description of this error.

Definition at line 72 of file sigsafety.py.

21.12.4 Member Data Documentation

21.12.4.1 signum

```
produtil.sigsafety.CaughtSignal.signum
```

The integer signal number.

Definition at line 68 of file sigsafety.py.

Referenced by produtil.sigsafety.CaughtSignal.__str__().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/sigsafety.py

21.13 produtil.cluster.Cluster Class Reference

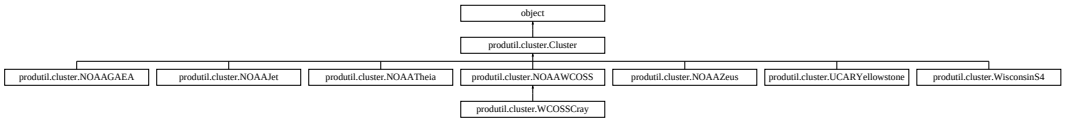
Stores information about a computer cluster.

21.13.1 Detailed Description

Stores information about a computer cluster.

Definition at line 15 of file cluster.py.

Inheritance diagram for produtil.cluster.Cluster:



Public Member Functions

- def __init__ (self, group_quotas, acl_support, production, name, longname, use_acl_for_rstdata=None)

  Sets all public member variables.
- def partition (self)

  What part of the cluster you are on; this information is system-specific.

Public Attributes

- group_quotas

  True if group membership is used to manage disk quotas.
- acl_support

  True if the system uses Access Control Lists (ACLs) to control access to files.
- production

  True if this system is production (real-time forecasting) environment, and False otherwise.
- name

  a short name of this cluster.
- longname

  a long name of this cluster.
- use_acl_for_rstdata

  True if the scripts should use ACLs to protect restricted access data.

21.13.2 Constructor & Destructor Documentation

21.13.2.1 __init__()

```
def produtil.cluster.Cluster.__init__ (
              self,
              group_quotas,
              acl_support,
              production,
              name,
              longname,
              use_acl_for_rstdata = None )
```

Sets all public member variables.

All are mandatory except use_acl_for_rstdata. The default for use_acl_for_rstdata is the logical AND of group_↩ quotas and acl_support.

Definition at line 18 of file cluster.py.

21.13.3 Member Function Documentation

21.13.3.1 partition()

```
produtil.cluster.Cluster.partition (
              self )
```

What part of the cluster you are on; this information is system-specific.

For example, on WCOSS, this may be "phase1" or "phase2" or "cray"

Definition at line 65 of file cluster.py.

21.13.4 Member Data Documentation

21.13.4.1 acl_support

```
produtil.cluster.Cluster.acl_support
```

True if the system uses Access Control Lists (ACLs) to control access to files.

Definition at line 23 of file cluster.py.

21.13.4.2 group_quotas

```
produtil.cluster.Cluster.group_quotas
```

True if group membership is used to manage disk quotas.

If this is True, then the scripts should never copy the group ID when copying files.

Definition at line 22 of file cluster.py.

21.13.4.3 longname

```
produtil.cluster.Cluster.longname
```

a long name of this cluster.

Definition at line 27 of file cluster.py.

21.13.4.4 name

`produtil.cluster.Cluster.name`

a short name of this cluster.

Must be a valid Python identifier string.

Definition at line 26 of file cluster.py.

Referenced by produtil.cluster.Cluster.partition(), and produtil.cluster.NOAAWCOSS.production().

21.13.4.5 production

`produtil.cluster.Cluster.production`

True if this system is production (real-time forecasting) environment, and False otherwise.

Most systems should set this to False.

Definition at line 25 of file cluster.py.

21.13.4.6 use_acl_for_rstdata

`produtil.cluster.Cluster.use_acl_for_rstdata`

True if the scripts should use ACLs to protect restricted access data.

If this is True, the scripts should copy ACLs when copying files. The produtil.acl supplies a way to do that on some Linux machines.

Definition at line 30 of file cluster.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cluster.py

21.14 produtil.mpi_impl.mpi_impl_base.CMDFGen Class Reference

Generates files with one line per MPI rank, telling what program to run on each rank.

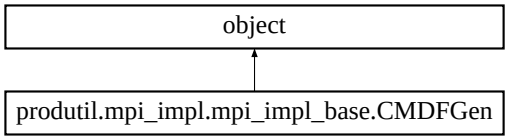
21.14.1 Detailed Description

Generates files with one line per MPI rank, telling what program to run on each rank.

This class is used to generate command files for mpiserial, poe or mpirun.lsf. Command files are files with one MPI rank per line containing a shell command to run for that rank. Generally the input (lines) is generated by the to_arglist function in a subclass of produtil.mpiprog.MPIRanksBase. See the produtil.mpi_impl.mpirun_lsf for an example of how to use this.

Definition at line 32 of file mpi_impl_base.py.

Inheritance diagram for produtil.mpi_impl.mpi_impl_base.CMDFGen:



Public Member Functions

- def __init__ (self, base, lines, cmd_envar='SCR_CMDFILE', model_envar=None, filename_arg=False, kwargs)

  CMDFGen constructor.
- def __call__ (self, runner, logger=None)

  Adds the environment variables to `runner` and creates the command file.

Public Attributes

- filename

  command file's filename
- tmpprefix

  temporary file prefix
- tmpsuffix

  temporary file suffix
- tmpdir

  temporary file directory
- cmd_envar

  Environment variable to set telling the path to the command file.
- model_envar

  Environment variable to set to "MPMD".
- **filename_arg**
- cmdf_contents

  String containing the command file contents.

21.14.2 Constructor & Destructor Documentation

21.14.2.1 __init__()

```
def produtil.mpi_impl.mpi_impl_base.CMDFGen.__init__ (
              self,
              base,
              lines,
              cmd_envar = 'SCR_CMDFILE',
              model_envar = None,
              filename_arg = False,
              kwargs )
```

CMDFGen constructor.

Parameters

base	type of command file being generated. See below.
lines	the command file contents as a list of strings, one per line
cmd_envar	environment variable to set to the command file path
model_envar	environment variable to set to "MPMD"
kwargs	Sets the command file name. See below.
filename_arg	If True, the name of the command file is appended to the program argument list.

The command file is generated from tempfile.NamedTemporaryFile, passing several arguments from kwargs, if provided, or suitable defaults otherwise. There are several arguments used. In all cases, replace "base" with the contents of the `base` argument:

- base_suffix — temporary file suffix (default: "base.")
- base_prefix — temporary file prefix (default: ".cmdf")
- base_tempdir — directory in which to create the file

Bug The base_suffix keyword is used for both the suffix and prefix

Definition at line 43 of file mpi_impl_base.py.

21.14.3 Member Function Documentation

21.14.3.1 __call__()

```
def produtil.mpi_impl.mpi_impl_base.CMDFGen.__call__ (
              self,
              runner,
              logger = None )
```

Adds the environment variables to `runner` and creates the command file.

Parameters

out	*runner*	A produtil.prog.Runner to modify
	logger	a logging.Logger for log messages

Definition at line 115 of file mpi_impl_base.py.

21.14.4 Member Data Documentation

21.14.4.1 cmdf_contents

`produtil.mpi_impl.mpi_impl_base.CMDFGen.cmdf_contents`

String containing the command file contents.

Definition at line 79 of file mpi_impl_base.py.

Referenced by produtil.mpi_impl.mpi_impl_base.CMDFGen.__call__().

The documentation for this class was generated from the following file:

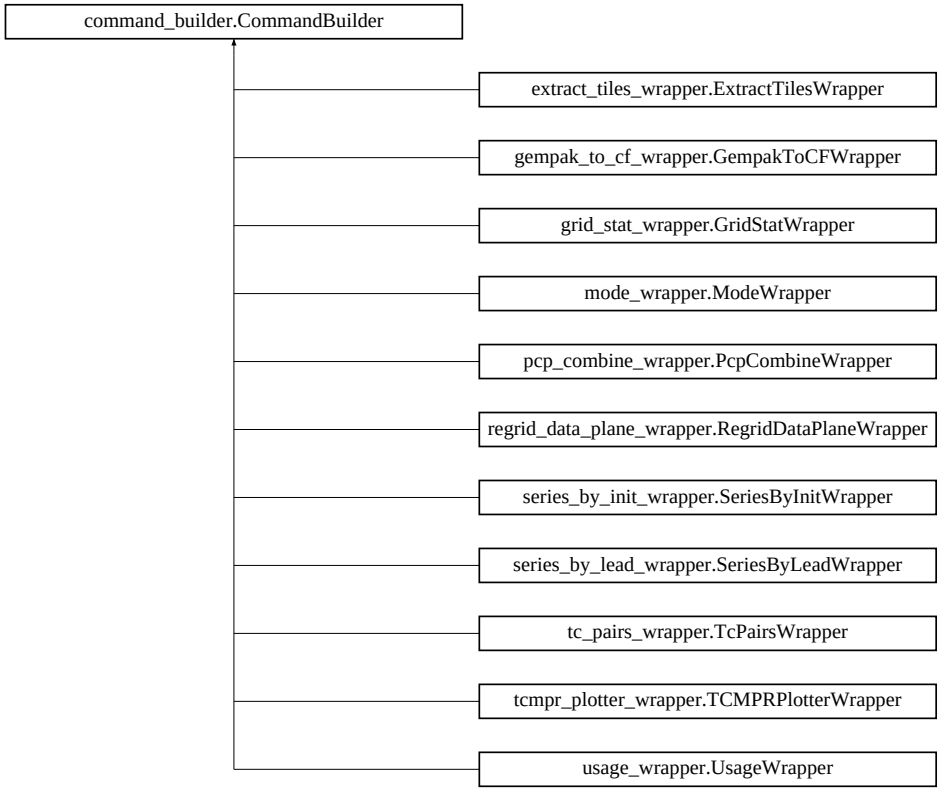
- /home/minnawin/wip_10-31/METplus/ush/produtil/mpi_impl/mpi_impl_base.py

21.15 command_builder.CommandBuilder Class Reference

21.15.1 Detailed Description

Definition at line 30 of file command_builder.py.

Inheritance diagram for command_builder.CommandBuilder:



Public Member Functions

- def __init__ (self, p, logger)
- def **clear** (self)
- def **set_debug** (self, debug)
- def **add_arg** (self, arg)
- def **add_input_file** (self, filename)
- def **get_input_files** (self)
- def **set_input_dir** (self, d)
- def set_output_path (self, outpath)
- def get_output_path (self)
- def **set_output_filename** (self, outfile)
- def **set_output_dir** (self, outdir)
- def **set_param_file** (self, param)
- def **add_env_var** (self, key, name)
- def **get_env** (self)
- def print_env (self)
- def print_env_copy (self, vars)
- def **print_env_item** (self, item)
- def get_command (self)
- def build (self)
- def **run_all_times** (self)

Public Attributes

- **p**
- **logger**
- **debug**
- **app_name**
- **app_path**
- **env**
- **args**
- **input_dir**
- **infiles**
- **outdir**
- **outfile**
- **param**

21.15.2 Constructor & Destructor Documentation

21.15.2.1 __init__()

```
def command_builder.CommandBuilder.__init__ (
              self,
              p,
              logger )
```

```
Retrieve parameters from corresponding param file
```

Definition at line 33 of file command_builder.py.

21.15.3 Member Function Documentation

21.15.3.1 build()

```
def command_builder.CommandBuilder.build (
              self )
```

```
Build and run command
```

Definition at line 145 of file command_builder.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.build_and_run_series_request().

21.15.3.2 get_command()

```
def command_builder.CommandBuilder.get_command (
              self )
```

```
Build command to run from arguments
```

Definition at line 113 of file command_builder.py.

Referenced by command_builder.CommandBuilder.build().

21.15.3.3 get_output_path()

```
def command_builder.CommandBuilder.get_output_path (
              self )
```

```
Combine output directory and filename then return result
```

Definition at line 72 of file command_builder.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.create_out_arg().

21.15.3.4 print_env()

```
def command_builder.CommandBuilder.print_env (
              self )
```

```
Print all environment variables set for this application
```

Definition at line 91 of file command_builder.py.

21.15.3.5 print_env_copy()

```
def command_builder.CommandBuilder.print_env_copy (
              self,
              vars )
```

```
Print list of environment variables that can be easily \
copied into terminal
```

Definition at line 96 of file command_builder.py.

21.15.3.6 set_output_path()

```
def command_builder.CommandBuilder.set_output_path (
              self,
              outpath )
```

```
Split path into directory and filename then save both
```

Definition at line 67 of file command_builder.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/command_builder.py

21.16 produtil.mpiprog.ComplexProgInput Class Reference

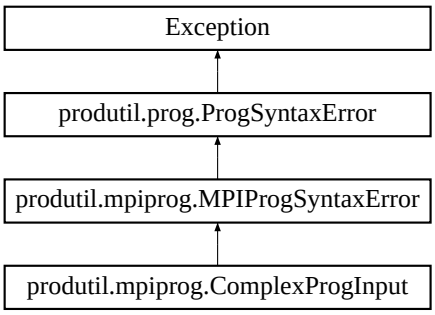
Raised when something that cannot be expressed as a pure MPI rank is given as a pure MPI rank.

21.16.1 Detailed Description

Raised when something that cannot be expressed as a pure MPI rank is given as a pure MPI rank.

Definition at line 55 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.ComplexProgInput:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.17 produtil.config.ConfFormatter Class Reference

Internal class that implements ProdConfig.strinterp()

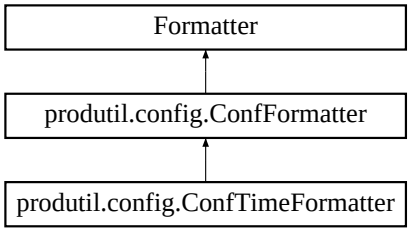
21.17.1 Detailed Description

Internal class that implements ProdConfig.strinterp()

This class is part of the implementation of ProdConfig: it is used to interpolate strings using a syntax similar to string.format(), but it allows recursion in the config sections, and it also is able to use the [config] and [dir] sections as defaults for variables not found in the current section.

Definition at line 68 of file config.py.

Inheritance diagram for produtil.config.ConfFormatter:



Public Member Functions

- def __init__ (self, quoted_literals=False)

  Constructor for ConfFormatter.
- def **quoted_literals** (self)
- def **slow_format** (self, format_string, args, kwargs)
- def **slow_vformat** (self, format_string, args, kwargs)
- def get_value (self, key, args, kwargs)

  Return the value of variable, or a substitution.

Public Attributes

- **format**
- **vformat**
- **parse**

21.17.2 Member Function Documentation

21.17.2.1 get_value()

```
def produtil.config.ConfFormatter.get_value (
              self,
              key,
              args,
              kwargs )
```

Return the value of variable, or a substitution.

Never call this function. It is called automatically by str.format. It provides the value of an variable, or a string substitution.

Parameters

key	the string key being analyzed by str.format()
args	the indexed arguments to str.format()
kwargs	the keyword arguments to str.format()

Definition at line 116 of file config.py.

Referenced by produtil.config.ConfFormatter.__init__().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/config.py

21.18 produtil.config.ConfTimeFormatter Class Reference

internal function that implements time formatting

21.18.1 Detailed Description

internal function that implements time formatting

Like its superclass, ConfFormatter, this class is part of the implementation of ProdConfig, and is used to interpolate strings in a way similar to string.format(). It works the same way as ConfFormatter, but accepts additional keys generated based on the forecast and analysis times:

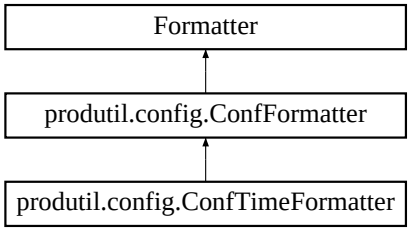
```
fYMDHM - 201409171200 = forecast time September 17, 2014 at 12:00 UTC
fYMDH  - 2014091712
fYMD   - 20140917
fyear  - 2014
fYYYY  - 2014
fYY    - 14   (year % 100)
fCC    - 20   (century)
fcen   - 20
fmonth - 09
fMM    - 09
fday   - 17
fDD    - 17
fhour  - 12
fcyc   - 12
fHH    - 12
fminute - 00
fmin   - 00
```

Replace the initial "f" with "a" for analysis times. In addition, the following are available for the time difference between forecast and analysis time. Suppose the forecast is twenty-three hours and nineteen minutes (23:19) after the analysis time:

```
fahr - 23
famin - 1399   ( = 23*60+19)
fahrmin - 19
```

Definition at line 303 of file config.py.

Inheritance diagram for produtil.config.ConfTimeFormatter:



Public Member Functions

- def __init__ (self, quoted_literals=False)

  constructor for ConfTimeFormatter
- def get_value (self, key, args, kwargs)

  return the value of a variable, or a substitution

Additional Inherited Members

21.18.2 Member Function Documentation

21.18.2.1 get_value()

```
def produtil.config.ConfTimeFormatter.get_value (
              self,
              key,
              args,
              kwargs )
```

return the value of a variable, or a substitution

Never call this function. It is called automatically by str.format. It provides the value of an variable, or a string substitution.

Parameters

key	the string key being analyzed by str.format()
args	the indexed arguments to str.format()
kwargs	the keyword arguments to str.format()

Definition at line 342 of file config.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/config.py

21.19 produtil.pipeline.Constant Class Reference

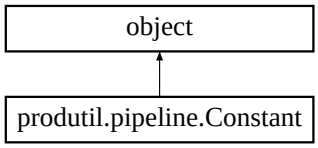
A class used to implement named constants.

21.19.1 Detailed Description

A class used to implement named constants.

Definition at line 23 of file pipeline.py.

Inheritance diagram for produtil.pipeline.Constant:



Public Member Functions

- def __init__ (self, s, r=None)

  Creates a named constant.
- def __str__ (self)

  Returns the s argument to the constructor.
- def __repr__ (self)

  Returns the r argument of the constructor.

21.19.2 Constructor & Destructor Documentation

21.19.2.1 __init__()

```
def produtil.pipeline.Constant.__init__ (
              self,
              s,
              r = None )
```

Creates a named constant.

Parameters

s	the return value of **str**() = str(self)
r	the return value of **repr**() = repr(self)

Definition at line 25 of file pipeline.py.

21.19.3 Member Function Documentation

21.19.3.1 __repr__()

```
def produtil.pipeline.Constant.__repr__ (
              self )
```

Returns the r argument of the constructor.

Definition at line 36 of file pipeline.py.

Referenced by produtil.prog.Runner.__str__().

21.19.3.2 __str__()

```
def produtil.pipeline.Constant.__str__ (
              self )
```

Returns the s argument to the constructor.

Definition at line 33 of file pipeline.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/pipeline.py

21.20 confdoc.coredoc Class Reference

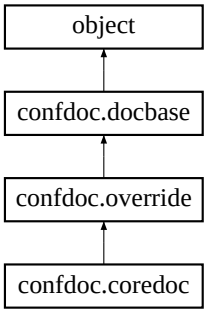
Subclass of override, for documenting the core configuration files.

21.20.1 Detailed Description

Subclass of override, for documenting the core configuration files.

Definition at line 549 of file confdoc.py.

Inheritance diagram for confdoc.coredoc:



Public Member Functions

- def __init__ (self, basename, parent)

  coredoc constructor
- def print_subdoc (self, s)

  Prints the documentation to the specified stream.

Additional Inherited Members

21.20.2 Constructor & Destructor Documentation

21.20.2.1 __init__()

```
def confdoc.coredoc.__init__ (
              self,
              basename,
              parent )
```

coredoc constructor

Parameters

basename	the file basename
parent	The parent docbase

Definition at line 552 of file confdoc.py.

21.20.3 Member Function Documentation

21.20.3.1 print_subdoc()

```
def confdoc.coredoc.print_subdoc (
              self,
              s )
```

Prints the documentation to the specified stream.

Parameters

s	The stream, ideally a StringIO.StringIO.

Definition at line 557 of file confdoc.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/confdoc.py

21.21 produtil.datastore.Datastore Class Reference

Stores information about Datum objects in a database.

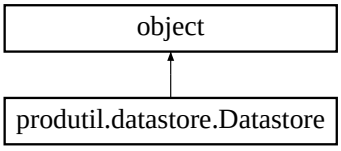
21.21.1 Detailed Description

Stores information about Datum objects in a database.

Stores and retrieves Datum objects from an sqlite3 database. Uses file locking workarounds for bugs in RedHat Enterprise Linux's sqlite3 library, which is compiled without proper locking. Has support for caching, and multithreading. Each object in this database has a type, a string location, an integer "available" parameter, and an arbitrary list of (key,value) metadata pairs. This object can safely be accessed by multiple threads in the local process, and handles concurrency between processes via file locking.

Definition at line 134 of file datastore.py.

Inheritance diagram for produtil.datastore.Datastore:



Public Member Functions

- def __init__ (self, filename, logger=None, locking=True)

  Datastore constructor.
- def transaction (self)

  Starts a transaction on the database in the current thread.
- def dump (self)

  Print database contents to the terminal.

Public Attributes

- filename

  The path to the sqlite3 database file.
- db

  The underlying sqlite3 database object.

21.21.2 Constructor & Destructor Documentation

21.21.2.1 __init__()

```
def produtil.datastore.Datastore.__init__ (
              self,
              filename,
              logger = None,
              locking = True )
```

Datastore constructor.

Creates a Datastore for the specified sqlite3 file. Uses the given logging.Logger object for logging messages. Set locking=False to disable all file locking. That is generally unwise, and should only be used when reading the database. That functionality is supplied, and critical, for monitoring another user's jobs. One cannot lock another user's file, so the "no locking" option is the only way to analyze the other user's simulation.

Parameters

filename	the filename passed to sqlite3.connect
logger	a logging.Logger to use for logging messages
locking	should file locking be used? It is unwise to turn off file locking.

Warning

Setting locking=False will disable file locking at both the Datastore level, and within sqlite3 itself. This can lead to database corruption if two processes try to write at the same time. This functionality is provided for the rare situation where you are unable to write to a database, such as when reading other users' sqlite3 database files.

Definition at line 146 of file datastore.py.

21.21.3 Member Function Documentation

21.21.3.1 dump()

```
def produtil.datastore.Datastore.dump (
              self )
```

Print database contents to the terminal.

This function is only meant for debugging. It dumps to the terminal an arguably human-readable display of the complete database state via the print command.

Definition at line 241 of file datastore.py.

21.21.3.2 transaction()

```
def produtil.datastore.Datastore.transaction (
              self )
```

Starts a transaction on the database in the current thread.

Definition at line 225 of file datastore.py.

Referenced by produtil.datastore.Datum.__delitem__(), produtil.datastore.Datum.__setitem__(), produtil.datastore.Datastore.dump(), and produtil.datastore.Datum.set_loc_avail().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.22 produtil.datastore.Datum Class Reference

Superclass of anything that can be stored in a Datastore.

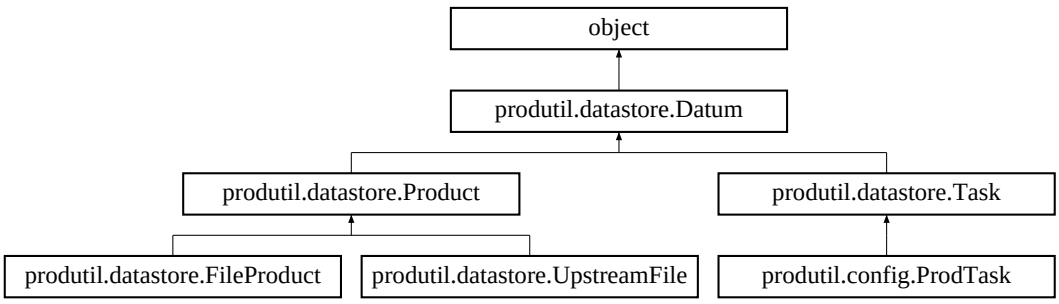
21.22.1 Detailed Description

Superclass of anything that can be stored in a Datastore.

This is the superclass of anything that can be placed in a datastore. It has a category, a product name (prodname for short), a location, availability (an int) and arbitrary metadata (key,value) pairs. It caches database metadata in self._meta, which is directly accessed by the Datastore class. Cache data will be discarded once its age is older than self._cacheage.

Definition at line 426 of file datastore.py.

Inheritance diagram for produtil.datastore.Datum:



Public Member Functions

- def __init__ (self, dstore, prodname, category, meta=None, cache=30, location=None, kwargs)

  Datum constructor.
- def __enter__ (self)

  Acquires this object's thread lock.
- def __exit__ (self, etype, evalue, traceback)

  Releases this object's thread lock.
- def validate (self)

  Validates this object's Datastore, prodname and category.
- def getid (self)

  Returns the database ID of this datum.
- def getdatastore (self)

  Returns the datastore of this datum.
- def transaction (self)

  Creates, but does not lock, a Transaction for this datum's datastore.
- def getprodtype (self)

  Returns the product type of this Datum.
- def getprodname (self)

  Returns the product name part of the database ID.
- def getcategory (self)

  Returns the product category part of the database ID.
- def getlocation (self)

  Returns the "location" field of this Datum's database entry.
- def setlocation (self, v)

  Sets the "location" field of this Datum's database entry.
- def __hash__ (self)

  Integer hash function.
- def __str__ (self)

  Human-readable description of this Datum.
- def __repr__ (self)

  Python code-like description of this Datum.
- def __cmp__ (self, other)

  Compares two Datums' prodnames and categories.
- def set_loc_avail (self, loc, avail)

  Sets the location and availability of this Datum in a single transaction.
- def update (self)

  Discards all cached metadata and refreshes it from the Datastore.
- def __getitem__ (self, k)

  Returns the value of the specified metadata key or raises KeyError.
- def meta (self, k, default=None)

  Return the value of a metadata key.
- def get (self, k, default=None)

  Alias for self.meta() Returns the value of the specified metadata key or returns default if it is unset.
- def __setitem__ (self, k, v)

  Sets the value of the specified metadata key.
- def __delitem__ (self, k)

  Deletes the specified metadata key, which must not be "available" or "location".
- def __contains__ (self, k)

  Determines if a metadata key is set.
- def iteritems (self)

  Iterates over all metadata (key,value) pairs for this Datum, including "available" and "location".

Properties

- prodname = property(getprodname,None,None,)

  Read-only property, an alias for getprodname(): the product name part of the database ID.
- category = property(getcategory,None,None,)

  Read-only property, an alias for getcategory(), the category name part of the database ID.
- prodtype

  Read-only property, an alias for getprodtype(), the product type.
- did

  Read-only property, an alias for getid().
- dstore

  Read-only property, an alias for getdatastore(), the Datastore in which this Datum resides.
- location

  Read-write property, an alias for getlocation() and setlocation().

21.22.2 Constructor & Destructor Documentation

21.22.2.1 __init__()

```
def produtil.datastore.Datum.__init__ (
              self,
              dstore,
              prodname,
              category,
              meta = None,
              cache = 30,
              location = None,
              kwargs )
```

Datum constructor.

Creates a Datum in the given Datastore dstore, under the specified category and product name prodname. The datastore id used is "category::prodname". The value for "cache" is the number of seconds to retain cached metadata before going back to disk to reread it. That only applies to data "get" operations: setting a data or metadata value will cause an immediate write to the database. Also, **contains** ("var" in self) will force a fetch from the database if the requested metadata variable is not in the cached copy of the database.

Values for location and meta are only the DEFAULT values, and will be ignored if other values are already set in the database. The location is only used if the product is not already in the database: its location will be set to the provided values upon entry. Similarly, the metadata is only set in this call if there isn't already metadata for the product with the given metadata keys.

Parameters

dstore	The Datastore for this Datum.
prodname	The product name portion of the Datum ID
category	The category part of the Datum ID
meta	A dict of metadata values.
cache	Metadata cache lifetime in seconds.
location	The initial value for location, if it is not set already in the database.
kwargs	Ignored

Definition at line 435 of file datastore.py.

21.22.3 Member Function Documentation

21.22.3.1 __cmp__()

```
def produtil.datastore.Datum.__cmp__ (
              self,
              other )
```

Compares two Datums' prodnames and categories.

Parameters

other	the other datum to compare against

Definition at line 576 of file datastore.py.

21.22.3.2 __contains__()

```
def produtil.datastore.Datum.__contains__ (
              self,
              k )
```

Determines if a metadata key is set.

Returns

True if the specified metadata key is set, and False otherwise. Immediately returns True for 'available' and 'location' without checking the metadata cache.

Parameters

k	the key of interest

Definition at line 667 of file datastore.py.

21.22.3.3 __delitem__()

```
def produtil.datastore.Datum.__delitem__ (
              self,
              k )
```

Deletes the specified metadata key, which must not be "available" or "location".

Parameters

k	the key to delete

Definition at line 657 of file datastore.py.

21.22.3.4 __enter__()

```
def produtil.datastore.Datum.__enter__ (
              self )
```

Acquires this object's thread lock.

This is used to manage cached data.

Definition at line 484 of file datastore.py.

21.22.3.5 __exit__()

```
def produtil.datastore.Datum.__exit__ (
              self,
              etype,
              evalue,
              traceback )
```

Releases this object's thread lock.

This is used to manage cached data.

Parameters

etype,evalue,traceback	Exception information.

Definition at line 488 of file datastore.py.

21.22.3.6 __getitem__()

```
def produtil.datastore.Datum.__getitem__ (
              self,
              k )
```

Returns the value of the specified metadata key or raises KeyError.

Forces a cache update if k is not in the cache.

Definition at line 619 of file datastore.py.

21.22.3.7 __hash__()

```
def produtil.datastore.Datum.__hash__ (
              self )
```

Integer hash function.

Definition at line 566 of file datastore.py.

21.22.3.8 __repr__()

```
def produtil.datastore.Datum.__repr__ (
              self )
```

Python code-like description of this Datum.

Definition at line 572 of file datastore.py.

Referenced by produtil.prog.Runner.__str__().

21.22.3.9 __setitem__()

```
def produtil.datastore.Datum.__setitem__ (
              self,
              k,
              v )
```

Sets the value of the specified metadata key.

Parameters

k	the key
v	the value

Definition at line 649 of file datastore.py.

21.22.3.10 __str__()

```
def produtil.datastore.Datum.__str__ (
              self )
```

Human-readable description of this Datum.

Definition at line 569 of file datastore.py.

21.22.3.11 get()

```
def produtil.datastore.Datum.get (
              self,
              k,
              default = None )
```

Alias for self.meta() Returns the value of the specified metadata key or returns default if it is unset.

Does NOT force a cache update if k is missing from the cache. To force a cache update, use **getitem** instead.

Parameters

k	The key of interest.
default	The value to return if no value is seen.

Returns

The metadata value or the default.

Definition at line 637 of file datastore.py.

Referenced by produtil.datastore.UpstreamFile.check(), produtil.config.ProdTask.confget(), produtil.config.ProdTask.getdir(), produtil.config.ProdTask.getexe(), and produtil.datastore.Datum.meta().

21.22.3.12 getcategory()

```
def produtil.datastore.Datum.getcategory (
              self )
```

Returns the product category part of the database ID.

Definition at line 520 of file datastore.py.

21.22.3.13 getdatastore()

```
def produtil.datastore.Datum.getdatastore (
              self )
```

Returns the datastore of this datum.

Definition at line 504 of file datastore.py.

21.22.3.14 getid()

```
def produtil.datastore.Datum.getid (
              self )
```

Returns the database ID of this datum.

Definition at line 501 of file datastore.py.

21.22.3.15 getlocation()

```
def produtil.datastore.Datum.getlocation (
              self )
```

Returns the "location" field of this Datum's database entry.

Definition at line 524 of file datastore.py.

21.22.3.16 getprodname()

```
def produtil.datastore.Datum.getprodname (
              self )
```

Returns the product name part of the database ID.

Definition at line 517 of file datastore.py.

21.22.3.17 getprodtype()

```
def produtil.datastore.Datum.getprodtype (
              self )
```

Returns the product type of this Datum.

Returns the product type of this Datum. This is generally the name of the Python class that created the entry in the database.

Definition at line 510 of file datastore.py.

21.22.3.18 iteritems()

```
def produtil.datastore.Datum.iteritems (
              self )
```

Iterates over all metadata (key,value) pairs for this Datum, including "available" and "location".

Definition at line 677 of file datastore.py.

21.22.3.19 meta()

```
def produtil.datastore.Datum.meta (
              self,
              k,
              default = None )
```

Return the value of a metadata key.

Returns the value of the specified metadata key or returns default if it is unset. Does NOT force a cache update if k is missing from the cache. To force a cache update, use **getitem** instead.

Parameters

k	The key of interest.
default	The value to return if no value is seen.

Returns

The metadata value or the default.

Definition at line 625 of file datastore.py.

Referenced by produtil.config.ProdTask.__init__(), produtil.config.ProdTask.get_outdir(), and produtil.config.ProdTask.get_workdir().

21.22.3.20 set_loc_avail()

```
def produtil.datastore.Datum.set_loc_avail (
              self,
              loc,
              avail )
```

Sets the location and availability of this Datum in a single transaction.

Parameters

loc	the new location, a string
avail	the new availability, an int

Definition at line 583 of file datastore.py.

Referenced by produtil.datastore.UpstreamFile.check(), and produtil.datastore.FileProduct.deliver().

21.22.3.21 setlocation()

```
def produtil.datastore.Datum.setlocation (
              self,
              v )
```

Sets the "location" field of this Datum's database entry.

Parameters

v	the new location

Definition at line 527 of file datastore.py.

21.22.3.22 transaction()

```
def produtil.datastore.Datum.transaction (
              self )
```

Creates, but does not lock, a Transaction for this datum's datastore.

Definition at line 507 of file datastore.py.

Referenced by produtil.datastore.Datum.__delitem__(), produtil.datastore.Datum.__init__(), produtil.datastore.Datum.__setitem__(), and produtil.datastore.Datum.set_loc_avail().

21.22.3.23 update()

```
def produtil.datastore.Datum.update (
              self )
```

Discards all cached metadata and refreshes it from the Datastore.

Definition at line 614 of file datastore.py.

Referenced by produtil.datastore.Product.check().

21.22.3.24 validate()

```
def produtil.datastore.Datum.validate (
              self )
```

Validates this object's Datastore, prodname and category.

Definition at line 493 of file datastore.py.

Referenced by produtil.datastore.Datum.__init__(), and produtil.mpiprog.MPIRank.__init__().

21.22.4 Property Documentation

21.22.4.1 category

```
produtil.datastore.Datum.category = property(getcategory,None,None,)  [static]
```

Read-only property, an alias for getcategory(), the category name part of the database ID.

Definition at line 540 of file datastore.py.

21.22.4.2 did

```
produtil.datastore.Datum.did  [static]
```

Initial value:

```
= property(getid,None,None,
                )
```

Read-only property, an alias for getid().

The full database ID.

Definition at line 551 of file datastore.py.

Referenced by produtil.datastore.Datum.__str__(), produtil.datastore.Product.call_callbacks(), produtil.datastore.↩
FileProduct.deliver(), and produtil.datastore.Datum.set_loc_avail().

21.22.4.3 dstore

```
produtil.datastore.Datum.dstore  [static]
```

Initial value:

```
= property(getdatastore,None,None,
                    )
```

Read-only property, an alias for getdatastore(), the Datastore in which this Datum resides.

Definition at line 557 of file datastore.py.

Referenced by produtil.datastore.Datum.__repr__(), and produtil.datastore.Datum.set_loc_avail().

21.22.4.4 location

```
produtil.datastore.Datum.location  [static]
```

Initial value:

```
= property(getlocation,setlocation,None,
                     )
```

Read-write property, an alias for getlocation() and setlocation().

The location on disk of this Datum.

Definition at line 563 of file datastore.py.

Referenced by produtil.datastore.UpstreamFile.check(), produtil.datastore.FileProduct.deliver(), and produtil.↩ datastore.FileProduct.undeliver().

21.22.4.5 prodname

```
produtil.datastore.Datum.prodname = property(getprodname,None,None,)  [static]
```

Read-only property, an alias for getprodname(): the product name part of the database ID.

Definition at line 535 of file datastore.py.

21.22.4.6 prodtype

```
produtil.datastore.Datum.prodtype  [static]
```

Initial value:

```
= property(getprodtype,None,None,
                       )
```

Read-only property, an alias for getprodtype(), the product type.

This is generally the name of the Python class that created the entry in the database.

Definition at line 546 of file datastore.py.

Referenced by produtil.datastore.Datum.__repr__(), and produtil.datastore.Datum.__str__().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.23 produtil.datastore.DatumException Class Reference

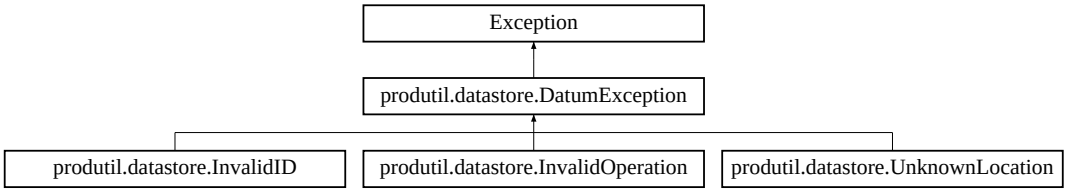
Superclass of all exceptions local to produtil.datastore.

21.23.1 Detailed Description

Superclass of all exceptions local to produtil.datastore.

Definition at line 24 of file datastore.py.

Inheritance diagram for produtil.datastore.DatumException:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.24 produtil.datastore.DatumLockHeld Class Reference

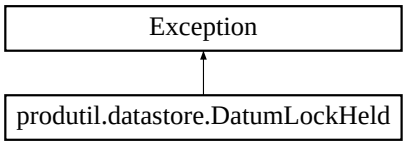
Raised when a LockDatum is held by another Worker.

21.24.1 Detailed Description

Raised when a LockDatum is held by another Worker.

Definition at line 27 of file datastore.py.

Inheritance diagram for produtil.datastore.DatumLockHeld:



Public Member Functions

- def __init__ (self, did, owner, owninfo, ownseen, ownlegacy, checktime)
  DatumLockHeld constructor.
- def __str__ (self)
  Human-readable representation of this exception.
- def __repr__ (self)
  String representation of this object.

Public Attributes

- did
  The database ID of the datum whose lock is held.
- owner
  The owner of the lock.
- owninfo
  implementation-defined information about the owner
- ownseen
  last time the owner checked in
- ownlegacy
  length of time the lock is valid
- checktime
  suggestion of how often to check the lock

21.24.2 Constructor & Destructor Documentation

21.24.2.1 __init__()

```
def produtil.datastore.DatumLockHeld.__init__ (
              self,
              did,
              owner,
              owninfo,
              ownseen,
              ownlegacy,
              checktime )
```

DatumLockHeld constructor.

Parameters

did	the database ID of the datum whose lock is held
owner	the owner of the lock
owninfo	implementation-defined information about the owner
ownseen	last time the owner checked in
ownlegacy	length of time the lock is valid
checktime	suggestion of how often to check the lock

Definition at line 29 of file datastore.py.

21.24.3 Member Function Documentation

21.24.3.1 __repr__()

```
def produtil.datastore.DatumLockHeld.__repr__ (
              self )
```

String representation of this object.

Definition at line 72 of file datastore.py.

Referenced by produtil.prog.Runner.__str__().

21.24.3.2 __str__()

```
def produtil.datastore.DatumLockHeld.__str__ (
              self )
```

Human-readable representation of this exception.

Definition at line 61 of file datastore.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.25 produtil.dbnalert.DBNAlert Class Reference

This class represents a call to dbn_alert, as a callable Python object.

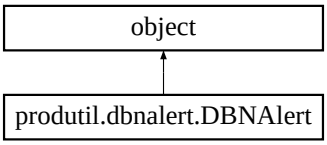
21.25.1 Detailed Description

This class represents a call to dbn_alert, as a callable Python object.

It allows the instructions on how to make the call to be stored for later use by a produtil.datastore.Product object's add_callback and call_callbacks functions.

Definition at line 47 of file dbnalert.py.

Inheritance diagram for produtil.dbnalert.DBNAlert:



Public Member Functions

- def __init__ (self, args, loglevel=logging.WARNING, alert_exe=None)

  Create a new DBNAlert object that can be used to send an alert later on.
- def __call__ (self, kwargs)

  Expands strings specified in the constructor and calls dbn_alert with the results.

Public Attributes

- alert_args

  Array of arguments to the alert function.
- alert_exe

  Alert executable.
- loglevel

  Desired logging level.

21.25.2 Constructor & Destructor Documentation

21.25.2.1 __init__()

```
def produtil.dbnalert.DBNAlert.__init__ (
              self,
              args,
              loglevel = logging.WARNING,
              alert_exe = None )
```

Create a new DBNAlert object that can be used to send an alert later on.

Parameters

args	The arguments to dbn_alert.
alert_exe	The dbn_alert executable name.
loglevel	A Python logging level to log messages before each alert.

Definition at line 52 of file dbnalert.py.

21.25.3 Member Function Documentation

21.25.3.1 __call__()

```
def produtil.dbnalert.DBNAlert.__call__ (
              self,
              kwargs )
```

Expands strings specified in the constructor and calls dbn_alert with the results.

If dbn alerts are disabled, then the fact that a dbn alert would be run is logged, but dbn_alert is NOT called.

Parameters

kwargs	string formatting variables for the dbn alert arguments

Definition at line 79 of file dbnalert.py.

21.25.4 Member Data Documentation

21.25.4.1 loglevel

```
produtil.dbnalert.DBNAlert.loglevel
```

Desired logging level.

Definition at line 69 of file dbnalert.py.

Referenced by produtil.dbnalert.DBNAlert.__call__().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/dbnalert.py

21.26 produtil.fileop.DeliveryFailed Class Reference

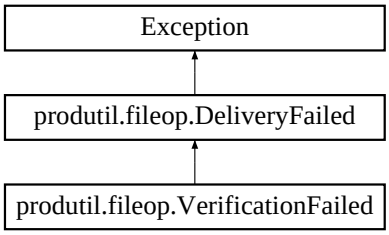
This exception is raised when a file cannot be delivered.

21.26.1 Detailed Description

This exception is raised when a file cannot be delivered.

Definition at line 87 of file fileop.py.

Inheritance diagram for produtil.fileop.DeliveryFailed:



Public Member Functions

- def __init__ (self, message, fromfile, tofile)
  DeliveryFailed constructor.
- def __str__ (self)
  Human-readable description of this error.
- def __repr__ (self)
  Pythonic representation of this error.

Public Attributes

- message
  String description of the problem.
- fromfile
  The source file.
- tofile
  The target file.

21.26.2 Constructor & Destructor Documentation

21.26.2.1 __init__()

```
def produtil.fileop.DeliveryFailed.__init__ (
              self,
              message,
              fromfile,
              tofile )
```

DeliveryFailed constructor.

Parameters

message	why the delivery failed
fromfile	what was being delivered
tofile	delivery destination

Definition at line 89 of file fileop.py.

21.26.3 Member Function Documentation

21.26.3.1 __repr__()

```
def produtil.fileop.DeliveryFailed.__repr__ (
              self )
```

Pythonic representation of this error.

Definition at line 110 of file fileop.py.

Referenced by produtil.prog.Runner.__str__().

21.26.3.2 __str__()

```
def produtil.fileop.DeliveryFailed.__str__ (
              self )
```

Human-readable description of this error.

Definition at line 106 of file fileop.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.27 confdoc.docbase Class Reference

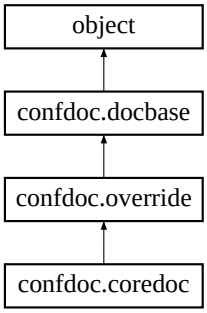
Stores documentation for all configuration options and sections.

21.27.1 Detailed Description

Stores documentation for all configuration options and sections.

Definition at line 56 of file confdoc.py.

Inheritance diagram for confdoc.docbase:



Public Member Functions

- def __init__ (self)

  docbase constructor
- def set_subdoc (self, basename, doc)

  Sets the documentation object that will contain file-specific information for the given file.
- def fileanch (self, basename)

  Returns the anchor for a configuration file.
- def secanch (self, section, where='sec')

  Returns the anchor for a specified section.
- def optanch (self, section, option, where='sec')

  Returns the anchor for a specified option.
- def make_brief (self, detail)

  Given a detailed description for something that has no brief description, return the brief description.
- def add_section (self, section, brief, detail, replace=None)

  Adds documentation for a conf section.
- def section_inc (self, section, inc)

  Sets the @inc list for a conf section.
- def file_block (self, basename, brief, detail)

  Adds a documentation block that is not associated with any section or option.
- def add_option (self, section, option, ivalue, brief, detail, basename=None, replace=None)

  Adds documentation for an option in a conf section.
- def print_doc (self, s)

  Writes Doxygen documentation to the stream "s" which is assumed to be a StringIO.

Public Attributes

- sections
  Mapping of section name to description.
- secref
  Mapping from section name to anchor name.
- **secrefset**
- secfile
  Mapping from doxified filename to list of sections in that file.
- secinc
  Mapping from section name to the list of sections that are @inc included by that section.
- seclist
  List of section names, in the order they were first seen.
- options
  Mapping from option anchor name to the description of the option.
- optbrief
  Mapping from option anchor to the option's brief description.
- secbrief
  Mapping from section name to the section's brief description.
- **fileset**
- **filelist**
- blocks
  Mapping from file basename to the list of documentation blocks in that file that were not associated with any section or option.

21.27.2 Constructor & Destructor Documentation

21.27.2.1 __init__()

```
def confdoc.docbase.__init__ (
              self )
```

docbase constructor

Initializes the documentation to an empty configuration suite. No sections, no options, no files, etc.

Definition at line 58 of file confdoc.py.

21.27.3 Member Function Documentation

21.27.3.1 add_option()

```
def confdoc.docbase.add_option (
              self,
              section,
              option,
              ivalue,
              brief,
              detail,
              basename = None,
              replace = None )
```

Adds documentation for an option in a conf section.

Parameters

section	The conf section name
option	The name of the option in that section
ivalue	A shortened form of the option value
brief	The brief documentation
detail	The detailed documentation
basename	The file basename
replace	If True, any existing documentation is replaced. Otherwise, it is ignored.

Definition at line 252 of file confdoc.py.

21.27.3.2 add_section()

```
def confdoc.docbase.add_section (
              self,
              section,
              brief,
              detail,
              replace = None )
```

Adds documentation for a conf section.

Parameters

section	The conf section name
brief	The brief documentation
detail	The detailed documentation or None
replace	If True, any existing documentation is replaced. Otherwise, it is ignored.

Definition at line 205 of file confdoc.py.

Referenced by confdoc.docbase.print_doc().

21.27.3.3 file_block()

```
def confdoc.docbase.file_block (
              self,
              basename,
              brief,
              detail )
```

Adds a documentation block that is not associated with any section or option.

Parameters

basename	the file that contains the block
brief	the brief documentation
detail	the detailed documentation

Definition at line 237 of file confdoc.py.

21.27.3.4 fileanch()

```
def confdoc.docbase.fileanch (
              self,
              basename )
```

Returns the anchor for a configuration file.

Parameters

basename	the basename of the configuration file.

Definition at line 118 of file confdoc.py.

Referenced by confdoc.docbase.optanch().

21.27.3.5 make_brief()

```
def confdoc.docbase.make_brief (
              self,
              detail )
```

Given a detailed description for something that has no brief description, return the brief description.

Parameters

detail	the detailed description

Returns

A brief description, or None if no suitable description was found.

Definition at line 190 of file confdoc.py.

Referenced by confdoc.docbase.add_section().

21.27.3.6 optanch()

```
def confdoc.docbase.optanch (
              self,
              section,
              option,
              where = 'sec' )
```

Returns the anchor for a specified option.

Parameters

section	The conf file section name.
option	The option name.
where	Configuration override file name. Default: "sec" which is the special name used for the page that stores information about ALL configuration options.

Definition at line 132 of file confdoc.py.

Referenced by confdoc.docbase.add_option(), confdoc.override.find_optbrief(), and confdoc.docbase.optanch().

21.27.3.7 print_doc()

```
def confdoc.docbase.print_doc (
              self,
              s )
```

Writes Doxygen documentation to the stream "s" which is assumed to be a StringIO.

Parameters

s	a StringIO.StringIO to receive documentation.

Definition at line 278 of file confdoc.py.

21.27.3.8 secanch()

```
def confdoc.docbase.secanch (
              self,
              section,
              where = 'sec' )
```

Returns the anchor for a specified section.

Parameters

section	The conf file section name.
where	Configuration override file name. Default: "sec" which is the special name used for the page that stores information about ALL configuration options.

Definition at line 123 of file confdoc.py.

Referenced by confdoc.docbase.add_section(), confdoc.docbase.optanch(), and confdoc.override.print_sec_opt().

21.27.3.9 section_inc()

```
def confdoc.docbase.section_inc (
              self,
              section,
              inc )
```

Sets the @inc list for a conf section.

Parameters

section	The conf section name
inc	The contents of the @inc= option

Definition at line 226 of file confdoc.py.

21.27.3.10 set_subdoc()

```
def confdoc.docbase.set_subdoc (
              self,
              basename,
              doc )
```

Sets the documentation object that will contain file-specific information for the given file.

Parameters

basename	The file basename
doc	The documentation object, a subclass of docbase

Definition at line 77 of file confdoc.py.

Referenced by confdoc.override.__init__().

21.27.4 Member Data Documentation

21.27.4.1 blocks

```
confdoc.docbase.blocks
```

Mapping from file basename to the list of documentation blocks in that file that were not associated with any section or option.

These blocks are placed at the top of that file's documentation page.

Definition at line 75 of file confdoc.py.

Referenced by confdoc.docbase.file_block(), and confdoc.docbase.print_doc().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/confdoc.py

21.28 produtil.config.DuplicateTaskName Class Reference

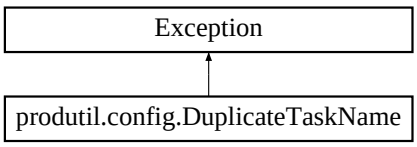
Raised when more than one task is registered with the same name in an ProdConfig object.

21.28.1 Detailed Description

Raised when more than one task is registered with the same name in an ProdConfig object.

Definition at line 29 of file config.py.

Inheritance diagram for produtil.config.DuplicateTaskName:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/config.py

21.29 produtil.config.Environment Class Reference

returns environment variables, allowing substitutions

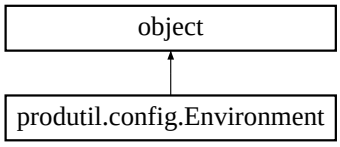
21.29.1 Detailed Description

returns environment variables, allowing substitutions

This class is used to read (but not write) environment variables and provide default values if an environment variable is unset or blank. It is only meant to be used in string formats, by passing ENV=ENVIRONMENT. There is a global constant in this module, ENVIRONMENT, which is an instance of this class. You should never need to instantiate another one.

Definition at line 35 of file config.py.

Inheritance diagram for produtil.config.Environment:



Public Member Functions

- def __contains__ (self, s)

  *Determines if **getitem** will return something (True) or raise KeyError (False).*
- def __getitem__ (self, s)

  Same as os.environ[s] unless s contains "|-".

21.29.2 Member Function Documentation

21.29.2.1 __contains__()

```
def produtil.config.Environment.__contains__ (
              self,
              s )
```

Determines if **getitem** will return something (True) or raise KeyError (False).

Same as "s in os.environ" unless s contains "|-", in which case, the result is True.

Definition at line 44 of file config.py.

21.29.2.2 __getitem__()

```
def produtil.config.Environment.__getitem__ (
              self,
              s )
```

Same as os.environ[s] unless s contains "|-".

ENVIRONMENT["VARNAME|-substitute"] will return os.environ[VARNAME] if VARNAME is defined and non-empty in os.environ. Otherwise, it will return "substitute".

Definition at line 49 of file config.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/config.py

21.30 produtil.prog.EqualInEnv Class Reference

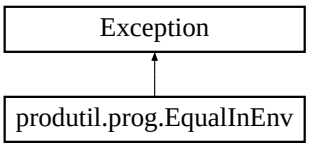
Raised when converting a Runner or pipeline of Runners to a POSIX sh string if there is an equal ("=") sign in an environment variable name.

21.30.1 Detailed Description

Raised when converting a Runner or pipeline of Runners to a POSIX sh string if there is an equal ("=") sign in an environment variable name.

Definition at line 81 of file prog.py.

Inheritance diagram for produtil.prog.EqualInEnv:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.31 produtil.prog.EqualInExecutable Class Reference

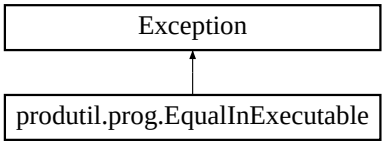
Raised when converting a Runner or pipeline of Runners to a posix sh string if a Runner's executable contains an equal ("=") sign.

21.31.1 Detailed Description

Raised when converting a Runner or pipeline of Runners to a posix sh string if a Runner's executable contains an equal ("=") sign.

Definition at line 77 of file prog.py.

Inheritance diagram for produtil.prog.EqualInExecutable:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.32 produtil.run.ExitStatusException Class Reference

Raised to indicate that a program generated an invalid return code.

21.32.1 Detailed Description

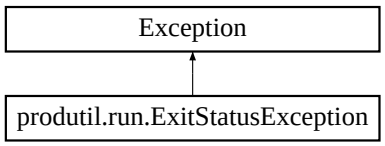
Raised to indicate that a program generated an invalid return code.

Examine the "returncode" member variable for the returncode value. Negative values indicate the program was terminated by a signal while zero and positive values indicate the program exited. The highest exit status of the pipeline is returned when a pipeline is used.

For MPI programs, the exit status is generally unreliable due to implementation-dependent issues, but this package attempts to return the highest exit status seen. Generally, you can count on MPI implementations to return zero if you call MPI_Finalize() and exit normally, and non-zero if you call MPI_Abort with a non-zero argument. Any other situation will produce unpredictable results.

Definition at line 179 of file run.py.

Inheritance diagram for produtil.run.ExitStatusException:



Public Member Functions

- def __init__ (self, message, status)
  ExitStatusException constructor.
- def status (self)
  An alias for self.returncode: the exit status.
- def __str__ (self)
  A string description of the error.
- def __repr__ (self)
  A pythonic description of the error for debugging.

Public Attributes

- message
  A string description for what went wrong.
- returncode
  The return code, including signal information.

21.32.2 Constructor & Destructor Documentation

21.32.2.1 __init__()

```
def produtil.run.ExitStatusException.__init__ (
              self,
              message,
              status )
```

ExitStatusException constructor.

Parameters

message	a description of what went wrong
status	the exit status

Definition at line 201 of file run.py.

21.32.3 Member Function Documentation

21.32.3.1 __repr__()

```
def produtil.run.ExitStatusException.__repr__ (
              self )
```

A pythonic description of the error for debugging.

Definition at line 216 of file run.py.

21.32.3.2 __str__()

```
def produtil.run.ExitStatusException.__str__ (
              self )
```

A string description of the error.

Definition at line 213 of file run.py.

21.32.3.3 status()

```
def produtil.run.ExitStatusException.status (
              self )
```

An alias for self.returncode: the exit status.

Definition at line 209 of file run.py.

21.32.4 Member Data Documentation

21.32.4.1 returncode

`produtil.run.ExitStatusException.returncode`

The return code, including signal information.

Definition at line 206 of file run.py.

Referenced by produtil.run.ExitStatusException.__repr__(), produtil.run.ExitStatusException.__str__(), and produtil.run.ExitStatusException.status().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/run.py

21.33 extract_tiles_wrapper.ExtractTilesWrapper Class Reference

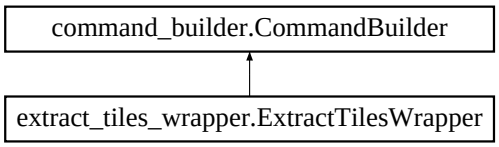
Takes tc-pairs data and regrids paired data to an n x m grid as specified in the config file.

21.33.1 Detailed Description

Takes tc-pairs data and regrids paired data to an n x m grid as specified in the config file.

Definition at line 36 of file extract_tiles_wrapper.py.

Inheritance diagram for extract_tiles_wrapper.ExtractTilesWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def **run_all_times** (self)
- def run_at_time (self, cur_init)

  Get TC-paris data then regrid tiles centered on the storm.

Public Attributes

- **app_path**
- **app_name**
- **tc_pairs_dir**
- **overwrite_flag**
- **addl_filter_opts**
- **filtered_out_dir**
- **tc_stat_exe**
- **init_beg**
- **init_end**
- **init_hour_inc**
- **init_hour_end**
- **logger**
- **config**

21.33.2 Member Function Documentation

21.33.2.1 run_at_time()

```
def extract_tiles_wrapper.ExtractTilesWrapper.run_at_time (
              self,
              cur_init )
```

Get TC-paris data then regrid tiles centered on the storm.

Get TC-pairs track data and GFS model data, do any necessary processing then regrid the forecast and analysis files to a 30 x 30 degree tile centered on the storm. Args:

Returns:

```
None: invokes regrid_data_plane to create a netCDF file from two
extratropical storm track files.
```

Definition at line 99 of file extract_tiles_wrapper.py.

Referenced by command_builder.CommandBuilder.build().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/extract_tiles_wrapper.py

21.34 extra_tropical_cyclone_plotter.ExtraTropicalCyclonePlotter Class Reference

Generate plots of extra tropical storm forecast tracks.

21.34.1 Detailed Description

Generate plots of extra tropical storm forecast tracks.

Reads input from ATCF files generated from MET TC-Pairs

Definition at line 20 of file extra_tropical_cyclone_plotter.py.

Public Member Functions

- def **__init__** (self, p, logger)
- def retrieve_data (self)

  Retrieve data from track files and return the min and max lon.
- def get_columns_and_indices (self, header)

  Parse the header for the columns of interest and store the information in a dictionary where the key is the column name and value is the index/column number.
- def create_plot (self)

  Create the plot, with a Basemap of the projection type requested in the metplus.conf file.
- def get_basemap (self)

  Retrieves the projection from the user's configuration file and returns the basemap and projection type.

Static Public Member Functions

- def extract_date_and_time_from_init (init_time_str)

  Extract and return the YYYYMMDD portion and the hh portion from the init time taken from the .tcst file.
- def extract_lead_hr (lead_str)

  Extract and return the lead hours from the hhmmss lead string.
- def **set_lead_group** (track_dict, init_hh)
- def **rescale_lon** (lon)

Public Attributes

- **input_data**
- **output_dir**
- **init_date**
- **lead_hr**
- **projection**
- **model**
- **title**
- **unique_storm_id**
- **storm_id_dict**
- **logger**
- **columns_of_interest**
- **llcrnrlat**
- **llcrnrlon**
- **urcrnrlat**
- **urcrnrlon**
- **resolution**

21.34.2 Member Function Documentation

21.34.2.1 create_plot()

```
def extra_tropical_cyclone_plotter.ExtraTropicalCyclonePlotter.create_plot (
              self )
```

Create the plot, with a Basemap of the projection type requested in the metplus.conf file.

Definition at line 273 of file extra_tropical_cyclone_plotter.py.

21.34.2.2 get_basemap()

```
def extra_tropical_cyclone_plotter.ExtraTropicalCyclonePlotter.get_basemap (
              self )
```

Retrieves the projection from the user's configuration file and returns the basemap and projection type.

Args: None

```
Returns:
```

Definition at line 443 of file extra_tropical_cyclone_plotter.py.

Referenced by extra_tropical_cyclone_plotter.ExtraTropicalCyclonePlotter.create_plot().

21.34.2.3 get_columns_and_indices()

```
def extra_tropical_cyclone_plotter.ExtraTropicalCyclonePlotter.get_columns_and_indices (
              self,
              header )
```

Parse the header for the columns of interest and store the information in a dictionary where the key is the column name and value is the index/column number.

Returns: column_dict: A dictionary containing the column name and its index

Definition at line 235 of file extra_tropical_cyclone_plotter.py.

Referenced by extra_tropical_cyclone_plotter.ExtraTropicalCyclonePlotter.retrieve_data().

21.34.2.4 retrieve_data()

```
def extra_tropical_cyclone_plotter.ExtraTropicalCyclonePlotter.retrieve_data (
              self )
```

Retrieve data from track files and return the min and max lon.

Returns: None

Definition at line 47 of file extra_tropical_cyclone_plotter.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/extra_tropical_cyclone_plotter.py

21.35 produtil.batchsystem.FakeClass Class Reference

A special class for constants.

21.35.1 Detailed Description

A special class for constants.

Definition at line 12 of file batchsystem.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/batchsystem.py

21.36 produtil.datastore.FakeException Class Reference

This is a fake exception used to get a stack trace.

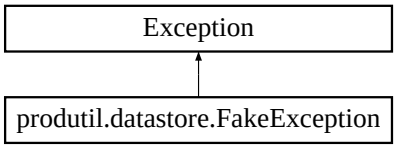
21.36.1 Detailed Description

This is a fake exception used to get a stack trace.

It will never be raised outside this module.

Definition at line 20 of file datastore.py.

Inheritance diagram for produtil.datastore.FakeException:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.37 produtil.sigsafety.FatalSignal Class Reference

Raised when a fatal signal is caught, as defined by the call to install_handlers.

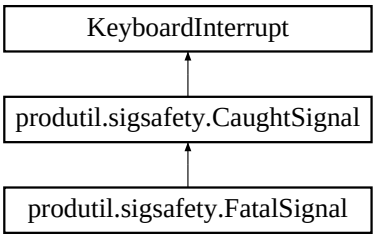
21.37.1 Detailed Description

Raised when a fatal signal is caught, as defined by the call to install_handlers.

Note that this does not derive from Exception.

Definition at line 79 of file sigsafety.py.

Inheritance diagram for produtil.sigsafety.FatalSignal:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/sigsafety.py

21.38 produtil.prog.FileOpener Class Reference

This is part of the internal implementation of Runner, used to convert it to a produtil.pipeline.Pipeline for execution.

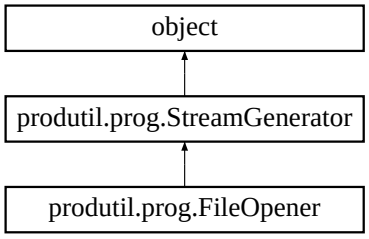
21.38.1 Detailed Description

This is part of the internal implementation of Runner, used to convert it to a produtil.pipeline.Pipeline for execution.

It represents stdin, stdout or stderr being connected to an open file. It instructs the Runner to open the file before starting the process.

Definition at line 134 of file prog.py.

Inheritance diagram for produtil.prog.FileOpener:



Public Member Functions

- def __init__ (self, filename, mode, err=False)

  FileOpener constructor.
- def copy (self)

  Creates a shallow copy of this object.
- def to_shell (self)

  Creates a POSIX sh representation of the part of the command that requests redirection.
- def intmode (self)

  Returns an integer version of mode suitable for os.open.
- def __repr__ (self)

  Returns a string representation of this object as valid Python code.
- def repr_for_in (self)

  Part of the implementation of Runner.__repr__, this returns the filename and ",string=False".
- def repr_for_out (self)

  Part of the implementation of Runner.__repr__, this returns the filename and ",string=False".
- def repr_for_err (self)

  Same as repr_for_out.

Public Attributes

- filename

  the name of the file being opened
- mode

  how the file is being opened
- err

  If True, this is for stderr.

21.38.2 Constructor & Destructor Documentation

21.38.2.1 __init__()

```
def produtil.prog.FileOpener.__init__ (
              self,
              filename,
              mode,
              err = False )
```

FileOpener constructor.

Parameters

filename	the name of the file being opened
mode	how it is being opened
err	if True, this is for stderr

Definition at line 140 of file prog.py.

21.38.3 Member Function Documentation

21.38.3.1 __repr__()

```
def produtil.prog.FileOpener.__repr__ (
              self )
```

Returns a string representation of this object as valid Python code.

Definition at line 188 of file prog.py.

Referenced by produtil.prog.Runner.__str__().

21.38.3.2 copy()

```
def produtil.prog.FileOpener.copy (
              self )
```

Creates a shallow copy of this object.

Definition at line 157 of file prog.py.

Referenced by produtil.prog.ImmutableRunner.runner().

21.38.3.3 repr_for_err()

```
def produtil.prog.FileOpener.repr_for_err (
              self )
```

Same as repr_for_out.

Definition at line 201 of file prog.py.

21.38.3.4 repr_for_in()

```
def produtil.prog.FileOpener.repr_for_in (
              self )
```

Part of the implementation of Runner.__repr__, this returns the filename and ",string=False".

Definition at line 192 of file prog.py.

21.38.3.5 repr_for_out()

```
def produtil.prog.FileOpener.repr_for_out (
              self )
```

Part of the implementation of Runner.__repr__, this returns the filename and ",string=False".

It also appends ",append=X" where X is the true/false flag for appending to the file.

Definition at line 196 of file prog.py.

Referenced by produtil.prog.StreamGenerator.repr_for_err(), and produtil.prog.FileOpener.repr_for_err().

21.38.3.6 to_shell()

```
def produtil.prog.FileOpener.to_shell (
              self )
```

Creates a POSIX sh representation of the part of the command that requests redirection.

Definition at line 160 of file prog.py.

21.38.4 Member Data Documentation

21.38.4.1 err

```
produtil.prog.FileOpener.err
```

If True, this is for stderr.

Definition at line 147 of file prog.py.

Referenced by produtil.prog.Runner.__init__(), produtil.prog.FileOpener.copy(), and produtil.prog.FileOpener.to_↩shell().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.39 produtil.fileop.FileOpError Class Reference

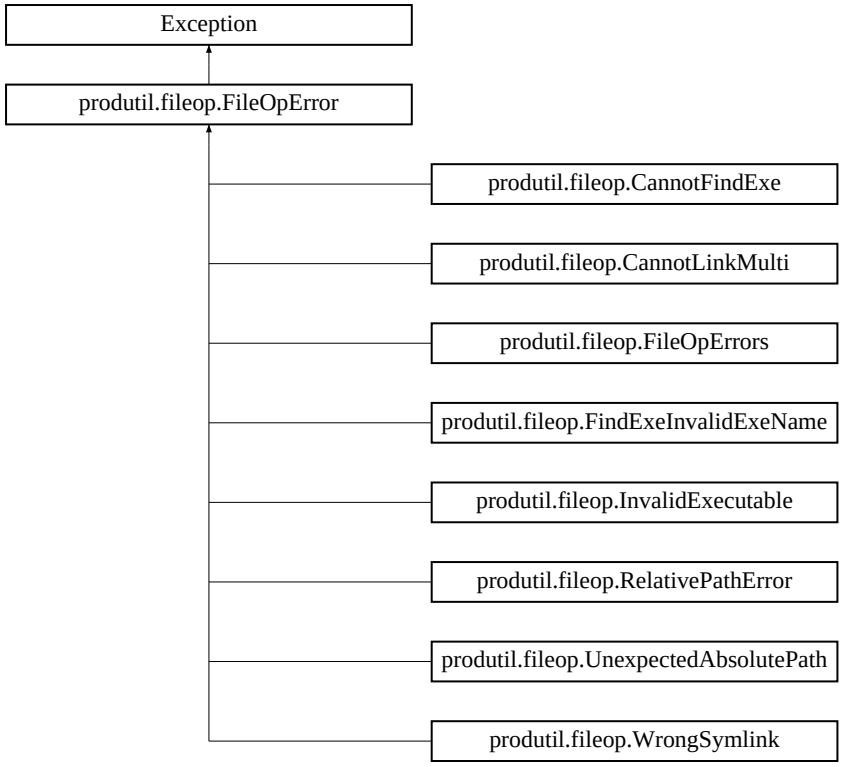
This is the superclass of several exceptions relating to multi-file operations in produtil.fileop.

21.39.1 Detailed Description

This is the superclass of several exceptions relating to multi-file operations in produtil.fileop.

Definition at line 26 of file fileop.py.

Inheritance diagram for produtil.fileop.FileOpError:



Public Member Functions

- def __init__ (self, message, filename, more=[ ])

  FileOpError constructor.
- def __str__ (self)

  A string description of the problem.
- def __iter__ (self)

  Iterates over a list of tuples, (from,to,message) where from is the source file, to is the destination file and message is a description of the problem with that pair.

Public Attributes

- message

  The error message.
- filename

  The name of the problematic file.
- more

  A list of tuples, (from,to,message) where from is the source file, to is the destination file and message is a description of the problem with that pair.

21.39.2 Constructor & Destructor Documentation

21.39.2.1 __init__()

```
def produtil.fileop.FileOpError.__init__ (
              self,
              message,
              filename,
              more = [] )
```

FileOpError constructor.

Parameters

message	the error message
filename	the name of the problematic file
more	a list of tuples, (from,to,message) where from is the source file, to is the destination file and message is a description of the problem with that pair.

Definition at line 29 of file fileop.py.

21.39.3 Member Function Documentation

21.39.3.1 __iter__()

```
def produtil.fileop.FileOpError.__iter__ (
              self )
```

Iterates over a list of tuples, (from,to,message) where from is the source file, to is the destination file and message is a description of the problem with that pair.

Definition at line 53 of file fileop.py.

21.39.3.2 __str__()

```
def produtil.fileop.FileOpError.__str__ (
              self )
```

A string description of the problem.

Definition at line 50 of file fileop.py.

21.39.4 Member Data Documentation

21.39.4.1 more

`produtil.fileop.FileOpError.more`

A list of tuples, (from,to,message) where from is the source file, to is the destination file and message is a description of the problem with that pair.

Definition at line 38 of file fileop.py.

Referenced by produtil.fileop.FileOpError.__iter__().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.40 produtil.fileop.FileOpErrors Class Reference

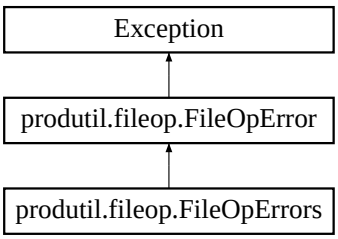
This exception is raised when an operation that processes multiple files catches more than one exception.

21.40.1 Detailed Description

This exception is raised when an operation that processes multiple files catches more than one exception.

Definition at line 59 of file fileop.py.

Inheritance diagram for produtil.fileop.FileOpErrors:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.41 produtil.datastore.FileProduct Class Reference

A subclass of Product that represents file delivery.

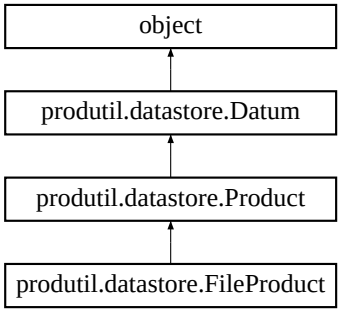
21.41.1 Detailed Description

A subclass of Product that represents file delivery.

This subclass of Product represents a file that is delivered by this workflow. The deliver() subroutine actually copies the file, and undeliver() deletes it. The produtil.fileop.remove_file() and produtil.fileop.deliver_file() are used for this purpose.

Definition at line 856 of file datastore.py.

Inheritance diagram for produtil.datastore.FileProduct:



Public Member Functions

- def undeliver (self, delete=True, logger=None)

  Undoes the effect of deliver()
- def deliver (self, location=None, frominfo=None, keep=True, logger=None, copier=None)

  Delivers the file to a destination.

Public Attributes

- **available**

Additional Inherited Members

21.41.2 Member Function Documentation

21.41.2.1 deliver()

```
def produtil.datastore.FileProduct.deliver (
               self,
               location = None,
               frominfo = None,
               keep = True,
               logger = None,
               copier = None )
```

Delivers the file to a destination.

Delivers the file to a destination location specified. The origin is in the "frominfo" argument. Delivery is done by produtil.fileop.deliver_file. The keep, copier and logger arguments are passed on unmodified.

Parameters

location	The new location.
frominfo	Where to get the file from.
keep	If True, the original file is always kept. If False, the original file may be moved to the destination instead of copied.
logger	a logging.Logger for log messages
copier	Passed to the copier argument of produtil.fileop.deliver_file()

Postcondition

The file is at the location specified, and the database location and availability are updated accordingly.

Definition at line 875 of file datastore.py.

Referenced by produtil.datastore.FileProduct.undeliver().

21.41.2.2 undeliver()

```
def produtil.datastore.FileProduct.undeliver (
              self,
              delete = True,
              logger = None )
```

Undoes the effect of deliver()

Sets this Product's available attribute to False. If delete=True, will also delete the specified file.

Parameters

delete	if True, the file is deleted
logger	a logging.Logger for log messages

Definition at line 863 of file datastore.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.42 produtil.fileop.FileWaiter Class Reference

A class that waits for files to meet some requirements.

21.42.1 Detailed Description

A class that waits for files to meet some requirements.

Definition at line 984 of file fileop.py.

Public Member Functions

- def __init__ (self, flist=None, min_size=None, min_mtime_age=None, min_atime_age=None, min_ctime_age=None, min_fraction=1.0)

  Constructor for the FileWaiter.
- def add (self, flist)

  Adds a file, or iterable that iterates over files, to the list of files to wait for.
- def check (self, filename, logger=None)

  Checks to see if one file meets the requirements set in the constructor.
- def reset (self)

  Resets internal information about which files have been seen.
- def iterfound (self)

  Iterates over all files that were found.
- def countfound (self)

  Returns the number of files that were found.
- def countmissing (self)

  Returns the number of files that were NOT found.
- def checkfiles (self, maxwait=1800, sleeptime=20, logger=None, log_each_file=True)

  Looks for the requested files.

Public Attributes

- min_size

  The minimum file size.
- min_mtime_age

  Minimum age of the modification time.
- min_atime_age

  Minimum age of the access time.
- min_ctime_age

  Minimum age of the creation and/or inode access time.
- min_fraction

  The minimum fraction of files that must meet the requirements.

21.42.2 Constructor & Destructor Documentation

21.42.2.1 __init__()

```
def produtil.fileop.FileWaiter.__init__ (
              self,
              flist = None,
              min_size = None,
              min_mtime_age = None,
              min_atime_age = None,
              min_ctime_age = None,
              min_fraction = 1.0 )
```

Constructor for the FileWaiter.

Most arguments have the same meaning as check_file()

Parameters

flist	the file or list of files to wait for. This is simply sent into self.add.
min_size	minimum file size
min_mtime_age	minimum modification time age,
min_atime_age	minimum access time age.
min_ctime_age	time since last file status change (see stat(2))
min_fraction	the minimum fraction of the provided files that must match the above requirements in order for FileWaiter.wait to return True. Default is 1.0, which means all of them.

Definition at line 989 of file fileop.py.

21.42.3 Member Function Documentation

21.42.3.1 add()

```
def produtil.fileop.FileWaiter.add (
              self,
              flist )
```

Adds a file, or iterable that iterates over files, to the list of files to wait for.

If the same filename is received a second time, it is ignored.

Parameters

flist	a filename (string) or list of filenames

Definition at line 1026 of file fileop.py.

Referenced by produtil.fileop.FileWaiter.add(), and produtil.fileop.FileWaiter.checkfiles().

21.42.3.2 check()

```
def produtil.fileop.FileWaiter.check (
              self,
              filename,
              logger = None )
```

Checks to see if one file meets the requirements set in the constructor.

This default implementation calls check_file. This is in a separate member function so that a subclass can override the file checking method.

Returns

True if the file is "ready," and False if it is not.

Parameters

filename	the path to the file to check
logger	a logging.Logger for messages

Definition at line 1039 of file fileop.py.

Referenced by produtil.fileop.FileWaiter.checkfiles().

21.42.3.3 checkfiles()

```
def produtil.fileop.FileWaiter.checkfiles (
              self,
              maxwait = 1800,
              sleeptime = 20,
              logger = None,
              log_each_file = True )
```

Looks for the requested files.

Will loop, checking over and over up to maxwait seconds, sleeping sleeptime seconds between checks.

Parameters

maxwait	maximum seconds to wait
sleeptime	sleep time in seconds between checks
logger	a logging.Logger for messages
log_each_file	log messages about each file checked

Definition at line 1067 of file fileop.py.

Referenced by produtil.fileop.FileWaiter.countmissing().

21.42.3.4 countfound()

```
def produtil.fileop.FileWaiter.countfound (
              self )
```

Returns the number of files that were found.

Definition at line 1059 of file fileop.py.

21.42.3.5 countmissing()

```
def produtil.fileop.FileWaiter.countmissing (
              self )
```

Returns the number of files that were NOT found.

Definition at line 1062 of file fileop.py.

21.42.3.6 iterfound()

```
def produtil.fileop.FileWaiter.iterfound (
              self )
```

Iterates over all files that were found.

Definition at line 1055 of file fileop.py.

21.42.3.7 reset()

```
def produtil.fileop.FileWaiter.reset (
              self )
```

Resets internal information about which files have been seen.

Definition at line 1050 of file fileop.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.43 produtil.fileop.FindExeInvalidExeName Class Reference

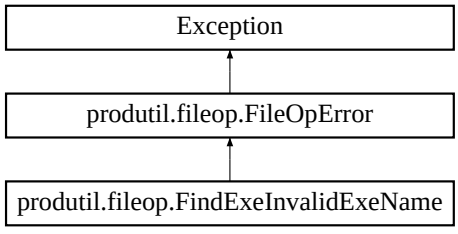
Thrown when find_exe is given an executable name that contains a directory path.

21.43.1 Detailed Description

Thrown when find_exe is given an executable name that contains a directory path.

Definition at line 71 of file fileop.py.

Inheritance diagram for produtil.fileop.FindExeInvalidExeName:



Additional Inherited Members

The documentation for this class was generated from the following file:

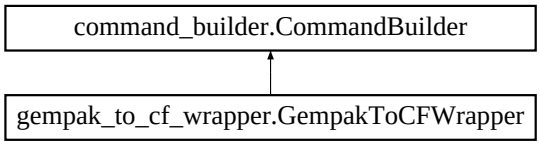
- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.44 gempak_to_cf_wrapper.GempakToCFWrapper Class Reference

21.44.1 Detailed Description

Definition at line 20 of file gempak_to_cf_wrapper.py.

Inheritance diagram for gempak_to_cf_wrapper.GempakToCFWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def **get_command** (self)

Public Attributes

- **app_name**
- **class_path**
- **outfile**

The documentation for this class was generated from the following file:

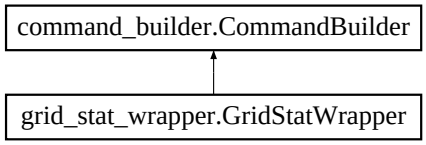
- /home/minnawin/wip_10-31/METplus/ush/gempak_to_cf_wrapper.py

21.45 grid_stat_wrapper.GridStatWrapper Class Reference

21.45.1 Detailed Description

Definition at line 31 of file grid_stat_wrapper.py.

Inheritance diagram for grid_stat_wrapper.GridStatWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def **set_output_dir** (self, outdir)
- def **get_command** (self)
- def **find_model** (self, model_type, lead, init_time)
- def **run_at_time** (self, init_time)
- def **run_at_time_once** (self, ti)

Public Attributes

- **app_path**
- **app_name**
- **outdir**

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/grid_stat_wrapper.py

21.46 produtil.sigsafety.HangupSignal Class Reference

With the default settings to install_handlers, this is raised when a SIGHUP is caught.

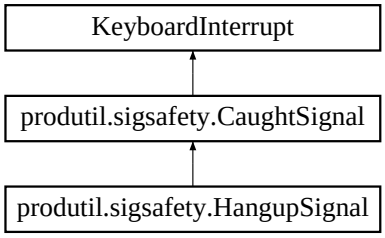
21.46.1 Detailed Description

With the default settings to install_handlers, this is raised when a SIGHUP is caught.

Note that this does not derive from Exception.

Definition at line 75 of file sigsafety.py.

Inheritance diagram for produtil.sigsafety.HangupSignal:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/sigsafety.py

21.47 produtil.prog.ImmutableRunner Class Reference

An copy-on-write version of Runner.

21.47.1 Detailed Description

An copy-on-write version of Runner.

This subclass of Runner is unmodifiable. It is meant to be used for re-usable exe()-like objects. For example, if one wants an object lsl that runs exe('ls')['-l'] with optional extra arguments, one could do:

lsl=ImmutableRunner(Runner('ls')['-l'])

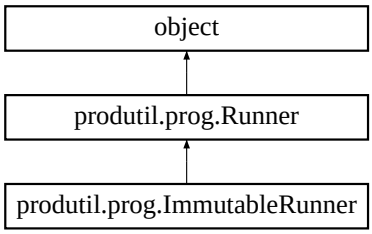
and then every time one does run(lsl[argument list]), it generates a new object without modifying the original lsl, ensuring later calls to lsl will have the same effect:

lsl['/'] lsl['~'] lsl['/'] # prints the same as the first

This is implemented by a copy-on-write method: if a modification is requested, a Runner is returned with the requested modifications.

Definition at line 897 of file prog.py.

Inheritance diagram for produtil.prog.ImmutableRunner:



Public Member Functions

- def __init__ (self, args, kwargs)

  Creates a new ImmutableRunner.
- def remove_prerun (self)

  Removes all prerun objects.
- def copy (self, typeobj=None)

  Creates a deep copy of this runner, except if stream objects are connected to stdin, stdout or stderr.
- def runner (self)

  Returns a modifiable version of this object (as a Runner).
- def copyenv (self)

  Creates a new Runner that is like self in all ways except that it uses the parent process environment.
- def clearenv (self)

  Creates a new Runner which is like self in all ways except that it uses an empty environment except for a few critical variables without which most programs cannot run.
- def cd (self, cd)

  Returns a new Runner that is like self, except that it cd's to the target directory before running.
- def env (self, kwargs)

Returns a new Runner that is like self in all ways except that the specified environment variables are set.

- def pipeto (self, other)

  Returns a new Runner that is like self in all ways, except that it has been piped into the other Runner.
- def inp (self, stdin, string=False)

  Returns a new Runner that is like self in all ways except that it has a different stdin.
- def out (self, stdout, append=False)

  Returns a new Runner that is like self in all ways except with a different stdout.
- def err (self, stderr, append=False)

  Returns a new Runner that is like self in all ways except with a different stderr.
- def err2out (self)

  Returns a new Runner that is like self in all ways except that stderr is piped into stdout.
- def prerun (self, arg)

  Returns a new Runner that is like self in all ways except that a new prerun function has been added.
- def __getitem__ (self, args)

  Returns a new Runner that is like self in all ways except with new arguments.
- def argins (self, index, arg)

  Returns a new Runner that is like self in all ways, except with the specified argument inserted.
- def setthreads (self, nthreads)

  Sets the number of threads requested by this program.
- def delthreads (self)

  Removes the request for threads.

Additional Inherited Members

21.47.2 Constructor & Destructor Documentation

21.47.2.1 __init__()

```
def produtil.prog.ImmutableRunner.__init__ (
              self,
              args,
              kwargs )
```

Creates a new ImmutableRunner.

All arguments to this constructor have the same meanings as the Runner constructor.

Parameters

args,kwargs	passed to Runner.__init__

Definition at line 918 of file prog.py.

21.47.3 Member Function Documentation

21.47.3.1 __getitem__()

```
def produtil.prog.ImmutableRunner.__getitem__ (
              self,
              args )
```

Returns a new Runner that is like self in all ways except with new arguments.

Parameters

args	the new argument or arguments

See also

Runner.__getitem__

Definition at line 1019 of file prog.py.

21.47.3.2 argins()

```
def produtil.prog.ImmutableRunner.argins (
              self,
              index,
              arg )
```

Returns a new Runner that is like self in all ways, except with the specified argument inserted.

Parameters

index	the index to insert before
arg	the argument to insert

Definition at line 1025 of file prog.py.

21.47.3.3 cd()

```
def produtil.prog.ImmutableRunner.cd (
              self,
              cd )
```

Returns a new Runner that is like self, except that it cd's to the target directory before running.

The directory must already exist before the program starts.

Parameters

cd	the directory to cd into, which must already exist.

Returns

the new Runner

Definition at line 972 of file prog.py.

21.47.3.4 clearenv()

```
def produtil.prog.ImmutableRunner.clearenv (
              self )
```

Creates a new Runner which is like self in all ways except that it uses an empty environment except for a few critical variables without which most programs cannot run.

(Retains PATH, USER, LOGNAME and HOME.)

Returns

a new Runner

Definition at line 965 of file prog.py.

21.47.3.5 copy()

```
def produtil.prog.ImmutableRunner.copy (
              self,
              typeobj = None )
```

Creates a deep copy of this runner, except if stream objects are connected to stdin, stdout or stderr.

In that case, those same stream objects are still connected.

Parameters

typeobj	the type of the output object. Do not use this unless you know what you're doing

Returns

a copy of self

Definition at line 936 of file prog.py.

21.47.3.6 copyenv()

```
def produtil.prog.ImmutableRunner.copyenv (
              self )
```

Creates a new Runner that is like self in all ways except that it uses the parent process environment.

Returns

the new Runner

Definition at line 960 of file prog.py.

21.47.3.7 delthreads()

```
def produtil.prog.ImmutableRunner.delthreads (
              self )
```

Removes the request for threads.

Same as self.threads=1

Definition at line 1046 of file prog.py.

21.47.3.8 env()

```
def produtil.prog.ImmutableRunner.env (
              self,
              kwargs )
```

Returns a new Runner that is like self in all ways except that the specified environment variables are set.

Parameters

kwargs	varname=value arguments of environment variables to set

Returns

the new Runner

Definition at line 979 of file prog.py.

21.47.3.9 err()

```
def produtil.prog.ImmutableRunner.err (
              self,
              stderr,
              append = False )
```

Returns a new Runner that is like self in all ways except with a different stderr.

Parameters

stderr	the stderr filename
append	if True, append to the file, otherwise truncate

Definition at line 1003 of file prog.py.

21.47.3.10 err2out()

```
def produtil.prog.ImmutableRunner.err2out (
              self )
```

Returns a new Runner that is like self in all ways except that stderr is piped into stdout.

Definition at line 1009 of file prog.py.

21.47.3.11 inp()

```
def produtil.prog.ImmutableRunner.inp (
              self,
              stdin,
              string = False )
```

Returns a new Runner that is like self in all ways except that it has a different stdin.

Parameters

stdin	the stdin string or filename
string	if True, stdin is a string

Definition at line 991 of file prog.py.

21.47.3.12 out()

```
def produtil.prog.ImmutableRunner.out (
              self,
```

```
            stdout,
            append = False )
```

Returns a new Runner that is like self in all ways except with a different stdout.

Parameters

stdout	the stdout filename
append	if True, append to the file, otherwise truncate

Definition at line 997 of file prog.py.

21.47.3.13 pipeto()

```
def produtil.prog.ImmutableRunner.pipeto (
            self,
            other )
```

Returns a new Runner that is like self in all ways, except that it has been piped into the other Runner.

Returns

the new Runner

Parameters

other	the Runner to pipe into.

Definition at line 985 of file prog.py.

21.47.3.14 prerun()

```
def produtil.prog.ImmutableRunner.prerun (
            self,
            arg )
```

Returns a new Runner that is like self in all ways except that a new prerun function has been added.

Parameters

arg	the new prerun function

See also

Runner.prerun()

Definition at line 1013 of file prog.py.

21.47.3.15 remove_prerun()

```
def produtil.prog.ImmutableRunner.remove_prerun (
              self )
```

Removes all prerun objects.

See also

prerun()

Definition at line 931 of file prog.py.

21.47.3.16 runner()

```
def produtil.prog.ImmutableRunner.runner (
              self )
```

Returns a modifiable version of this object (as a Runner).

Definition at line 946 of file prog.py.

21.47.3.17 setthreads()

```
def produtil.prog.ImmutableRunner.setthreads (
              self,
              nthreads )
```

Sets the number of threads requested by this program.

Definition at line 1041 of file prog.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.48 produtil.mpiprog.InputsNotStrings Class Reference

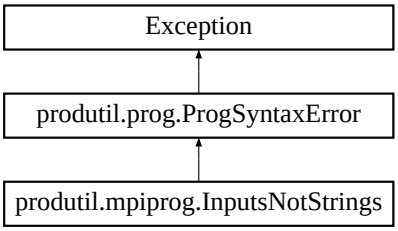
Raised when the validation scripts were expecting string arguments or string executable names, but something else was found.

21.48.1 Detailed Description

Raised when the validation scripts were expecting string arguments or string executable names, but something else was found.

Definition at line 64 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.InputsNotStrings:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.49 produtil.fileop.InvalidExecutable Class Reference

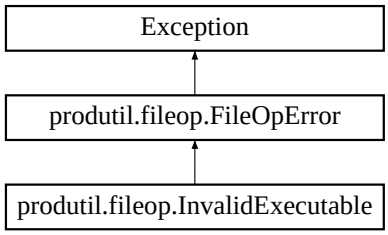
Thrown when a find_exe fails.

21.49.1 Detailed Description

Thrown when a find_exe fails.

Definition at line 69 of file fileop.py.

Inheritance diagram for produtil.fileop.InvalidExecutable:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.50 produtil.datastore.InvalidID Class Reference

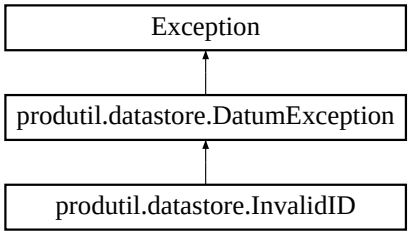
Raised when a Datum or subclass receives a prodname or category name that is invalid.

21.50.1 Detailed Description

Raised when a Datum or subclass receives a prodname or category name that is invalid.

Definition at line 78 of file datastore.py.

Inheritance diagram for produtil.datastore.InvalidID:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.51 produtil.datastore.InvalidOperation Class Reference

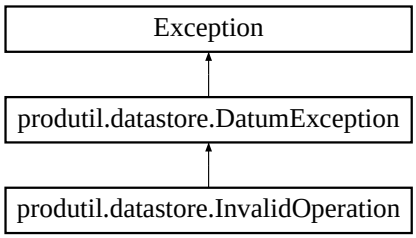
Raised when an invalid Datum operation is requested, such as delivering an UpstreamProduct.

21.51.1 Detailed Description

Raised when an invalid Datum operation is requested, such as delivering an UpstreamProduct.

Definition at line 80 of file datastore.py.

Inheritance diagram for produtil.datastore.InvalidOperation:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.52 produtil.prog.InvalidPipeline Class Reference

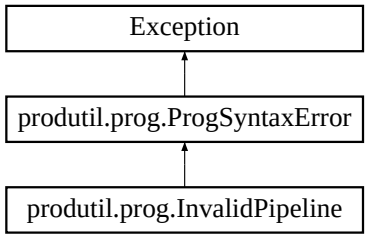
Raised when the caller specifies an invalid input or output when piping a Runner into or out of another object.

21.52.1 Detailed Description

Raised when the caller specifies an invalid input or output when piping a Runner into or out of another object.

Definition at line 61 of file prog.py.

Inheritance diagram for produtil.prog.InvalidPipeline:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.53 produtil.run.InvalidRunArgument Class Reference

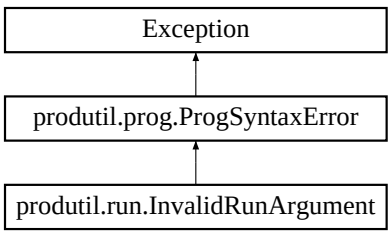
Raised to indicate that an invalid argument was sent into one of the run module functions.

21.53.1 Detailed Description

Raised to indicate that an invalid argument was sent into one of the run module functions.

Definition at line 175 of file run.py.

Inheritance diagram for produtil.run.InvalidRunArgument:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/run.py

21.54 produtil.numerics.InvalidTimespan Class Reference

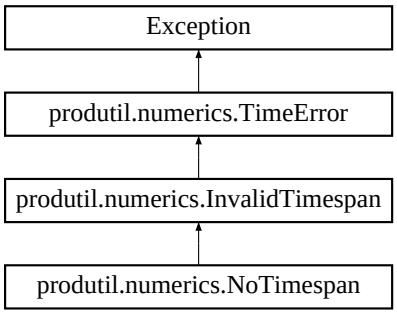
Superclass of exceptions relating to groups of one or more distinct times and relationships between them.

21.54.1 Detailed Description

Superclass of exceptions relating to groups of one or more distinct times and relationships between them.

Definition at line 30 of file numerics.py.

Inheritance diagram for produtil.numerics.InvalidTimespan:



Public Member Functions

- def __init__ (self, message, start, end)
  Constructor for InvalidTimespan.

Public Attributes

- start
  the start of the problematic timespan
- end
  the end of the problematic timespan

21.54.2 Constructor & Destructor Documentation

21.54.2.1 __init__()

```
def produtil.numerics.InvalidTimespan.__init__ (
              self,
              message,
              start,
              end )
```

Constructor for InvalidTimespan.

Parameters

message	the string explanation of the problem
start	the start of the timespan
end	the end of the timespan

Definition at line 39 of file numerics.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.55 produtil.numerics.InvalidTimestep Class Reference

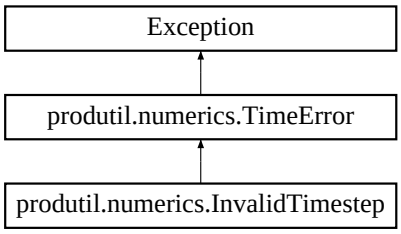
Raised when a timestep is invalid, such as a negative timestep for a situation that requires a positive one.

21.55.1 Detailed Description

Raised when a timestep is invalid, such as a negative timestep for a situation that requires a positive one.

Definition at line 20 of file numerics.py.

Inheritance diagram for produtil.numerics.InvalidTimestep:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.56 produtil.log.JLogFormatter Class Reference

This subclass of MasterLogFormatter does not include exception information in the log file.

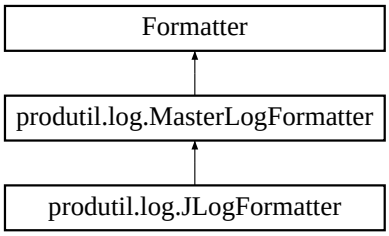
21.56.1 Detailed Description

This subclass of MasterLogFormatter does not include exception information in the log file.

This is done to prevent cluttering of the log file.

Definition at line 126 of file log.py.

Inheritance diagram for produtil.log.JLogFormatter:



Public Member Functions

- def formatException (self, ei)

  This subclass of MasterLogFormatter does not include exception information in the log file.

21.56.2 Member Function Documentation

21.56.2.1 formatException()

```
def produtil.log.JLogFormatter.formatException (
              self,
              ei )
```

This subclass of MasterLogFormatter does not include exception information in the log file.

This is done to prevent cluttering of the log file. Returns nothing to indicate no exception information should be printed.

Parameters

ei	the exception information to ignore

Definition at line 130 of file log.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/log.py

21.57 produtil.log.JLogHandler Class Reference

Custom LogHandler for the jlogfile.

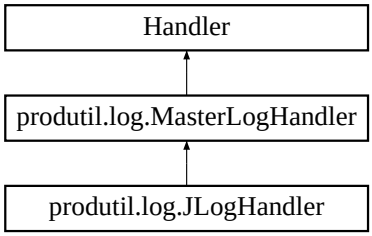
21.57.1 Detailed Description

Custom LogHandler for the jlogfile.

This is a custom logging Handler class for the jlogfile. It has a higher minimum log level for messages not sent to the jlogfile domain. Also, for every log message, the log file is opened, the message is written and the file is closed. This is done to mimic the postmsg command. Exception information is never sent to the log file.

Definition at line 194 of file log.py.

Inheritance diagram for produtil.log.JLogHandler:



Public Member Functions

- def emit (self, record)

  Write a log message.
- def set_jlogfile (self, filename)

  Set the location of the jlogfile.

21.57.2 Member Function Documentation

21.57.2.1 emit()

```
def produtil.log.JLogHandler.emit (
              self,
              record )
```

Write a log message.

Parameters

record	the log record

Note

See the Python logging module documentation for details.

Definition at line 203 of file log.py.

21.57.2.2 set_jlogfile()

```
def produtil.log.JLogHandler.set_jlogfile (
              self,
              filename )
```

Set the location of the jlogfile.

Parameters

filename	The path to the jlogfile.

Definition at line 231 of file log.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/log.py

21.58 produtil.listing.Listing Class Reference

Imitates the shell "ls -l" program.

21.58.1 Detailed Description

Imitates the shell "ls -l" program.

Imitate ls -l, but with a longer mtime string:

```
print Listing("/usr/local")
```

To include files whose names begin with "." add hidden=True:

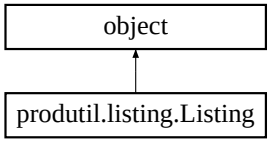
```
print Listing("/usr/local",hidden=True)
```

To log messages related to failures of lstat and readlink, pass a logging.Logger:

```
print Listing("/usr/local",hidden=True,logger=logger)
```

Definition at line 9 of file listing.py.

Inheritance diagram for produtil.listing.Listing:



Public Member Functions

- def __init__ (self, path=".", hidden=False, logger=None)

  Constructor for Listing:
- def __iter__ (self)

  Iterates over filenames in the listed directory.
- def iteritems (self)

  Iterates over name,data pairs in the listed directory.
- def iterkeys (self)

  Iterates over filenames in the listed directory.
- def list (self, hidden=False, logger=None)

  Updates the internal data structures with a new listing of the directory.
- def __str__ (self)

  Generates an ls -l style listing of the directory.

21.58.2 Constructor & Destructor Documentation

21.58.2.1 __init__()

```
def produtil.listing.Listing.__init__ (
              self,
              path = ".",
              hidden = False,
              logger = None )
```

Constructor for Listing:

Parameters

path	The directory path to list.
hidden	If True, files with names beginning with "." are listed.
logger	A logging.Logger for error messages.

Definition at line 27 of file listing.py.

21.58.3 Member Function Documentation

21.58.3.1 __iter__()

```
def produtil.listing.Listing.__iter__ (
              self )
```

Iterates over filenames in the listed directory.

Definition at line 35 of file listing.py.

21.58.3.2 __str__()

```
def produtil.listing.Listing.__str__ (
              self )
```

Generates an ls -l style listing of the directory.

Definition at line 84 of file listing.py.

21.58.3.3 iteritems()

```
def produtil.listing.Listing.iteritems (
              self )
```

Iterates over name,data pairs in the listed directory.

The "data" will be a tuple containing the output of lstat and the output of readlink.

Definition at line 39 of file listing.py.

21.58.3.4 iterkeys()

```
def produtil.listing.Listing.iterkeys (
              self )
```

Iterates over filenames in the listed directory.

Definition at line 45 of file listing.py.

Referenced by produtil.listing.Listing.__iter__().

21.58.3.5 list()

```
def produtil.listing.Listing.list (
              self,
              hidden = False,
              logger = None )
```

Updates the internal data structures with a new listing of the directory.

Arguments are the same as for the constructor.

Parameters

hidden	If True, files with names beginning with "." are listed.
logger	A logging.Logger for error messages.

Definition at line 50 of file listing.py.

Referenced by produtil.listing.Listing.__init__(), and produtil.listing.Listing.__str__().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/listing.py

21.59 produtil.locking.LockFile Class Reference

Automates locking of a lockfile.

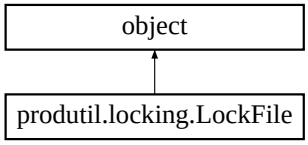
21.59.1 Detailed Description

Automates locking of a lockfile.

```
with LockFile("/path/to/lock.file"):
    ... do things while file is locked ...
...file is no longer locked.
```

Definition at line 66 of file locking.py.

Inheritance diagram for produtil.locking.LockFile:



Public Member Functions

- def __hash__ (self)

  Return a hash of this object.
- def __eq__ (self, other)

  Is this lock the same as that lock?
- def __init__ (self, filename, until=None, logger=None, max_tries=10, sleep_time=3, first_warn=0, giveup_↩ quiet=False)

  Creates an object that will lock the specified file.
- def acquire_impl (self)

  Internal implementation function; do not call directly.
- def release_impl (self)

  Internal implementation function; do not call directly.
- def acquire (self)

  Acquire the lock.
- def release (self)

  Release the lock.
- def __enter__ (self)

  Calls self.acquire() to acquire the lock.
- def __exit__ (self, etype, evalue, etraceback)

  Calls self.release() to release the lock.

21.59.2 Constructor & Destructor Documentation

21.59.2.1 __init__()

```
def produtil.locking.LockFile.__init__ (
              self,
              filename,
              until = None,
              logger = None,
              max_tries = 10,
              sleep_time = 3,
              first_warn = 0,
              giveup_quiet = False )
```

Creates an object that will lock the specified file.

Parameters

filename	the file to lock
until	Unused.
logger	Optional: a logging.Logger to log messages
max_tries	Optional: maximum tries before giving up on locking
sleep_time	Optional: approximate sleep time between locking attempts.
first_warn	Optional: first locking failure at which to write warnings to the logger
giveup_quiet	Optional: if True, do not log the final failure to lock

Definition at line 80 of file locking.py.

21.59.3 Member Function Documentation

21.59.3.1 __enter__()

```
def produtil.locking.LockFile.__enter__ (
              self )
```

Calls self.acquire() to acquire the lock.

Definition at line 149 of file locking.py.

21.59.3.2 __exit__()

```
def produtil.locking.LockFile.__exit__ (
              self,
              etype,
              evalue,
              etraceback )
```

Calls self.release() to release the lock.

Parameters

etype,evalue,etraceback	Exception information.

Definition at line 153 of file locking.py.

21.59.3.3 __hash__()

```
def produtil.locking.LockFile.__hash__ (
              self )
```

Return a hash of this object.

Definition at line 74 of file locking.py.

21.59.3.4 acquire()

```
def produtil.locking.LockFile.acquire (
              self )
```

Acquire the lock.

Will try for a while, and will raise LockHeld when giving up.

Definition at line 131 of file locking.py.

Referenced by produtil.locking.LockFile.__enter__().

21.59.3.5 acquire_impl()

```
def produtil.locking.LockFile.acquire_impl (
              self )
```

Internal implementation function; do not call directly.

Does the actual work of acquiring the lock, without retries, logging or sleeping. Will raise LockHeld if it cannot acquire the lock.

Definition at line 103 of file locking.py.

Referenced by produtil.locking.LockFile.acquire().

21.59.3.6 release()

```
def produtil.locking.LockFile.release (
              self )
```

Release the lock.

May raise exceptions on unexpected failures.

Definition at line 138 of file locking.py.

Referenced by produtil.locking.LockFile.__exit__().

21.59.3.7 release_impl()

```
def produtil.locking.LockFile.release_impl (
              self )
```

Internal implementation function; do not call directly.

Does the actual work of releasing the lock, without retries, logging or sleeping.

Definition at line 123 of file locking.py.

Referenced by produtil.locking.LockFile.release().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/locking.py

21.60 produtil.locking.LockHeld Class Reference

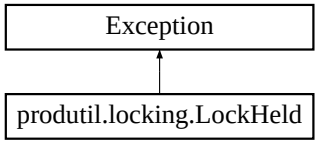
This exception is raised when a LockFile cannot lock a file because another process or thread has locked it already.

21.60.1 Detailed Description

This exception is raised when a LockFile cannot lock a file because another process or thread has locked it already.

Definition at line 55 of file locking.py.

Inheritance diagram for produtil.locking.LockHeld:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/locking.py

21.61 produtil.locking.LockingDisabled Class Reference

This exception is raised when a thread attempts to acquire a lock while Python is exiting according to produtil.sigsafety.

21.61.1 Detailed Description

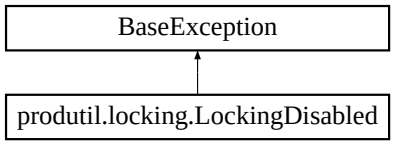
This exception is raised when a thread attempts to acquire a lock while Python is exiting according to produtil.sigsafety.

Warning

This is a subclass of BaseException, not Exception, to attempt to cleanly kill the thread.

Definition at line 59 of file locking.py.

Inheritance diagram for produtil.locking.LockingDisabled:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/locking.py

21.62 produtil.log.MasterLogFormatter Class Reference

This is a custom log formatter that inserts the thread or process (logthread) that generated the log message.

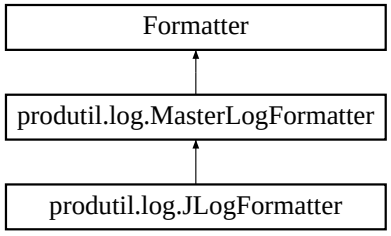
21.62.1 Detailed Description

This is a custom log formatter that inserts the thread or process (logthread) that generated the log message.

Also, it always directly calls formatException from format, ensuring that cached information is not used. That allows a subclass (JLogFormatter) to ignore exceptions.

Definition at line 76 of file log.py.

Inheritance diagram for produtil.log.MasterLogFormatter:



Public Member Functions

- def __init__ (self, fmt=None, datefmt=None, logthread=None)

  MasterLogFormatter constructor.
- def logthread (self)

  The name of the batch thread or process that generated log messages, if the LogRecord does not supply that already.
- def format (self, record)

  Replaces the logging.Formatter.format() function.
- def formatException (self, ei)

  Returns nothing to indicate no exception information should be printed.

21.62.2 Constructor & Destructor Documentation

21.62.2.1 __init__()

```
def produtil.log.MasterLogFormatter.__init__ (
              self,
              fmt = None,
              datefmt = None,
              logthread = None )
```

MasterLogFormatter constructor.

Parameters

fmt	the log message format
datefmt	the date format
logthread	the thread name for logging

Note

See the Python logging module documentation for details.

Definition at line 82 of file log.py.

21.62.3 Member Function Documentation

21.62.3.1 format()

```
def produtil.log.MasterLogFormatter.format (
              self,
              record )
```

Replaces the logging.Formatter.format() function.

We need to override this due to a "feature" in the Formatter.format: It ignores formatException (never calls it) and caches the exception info, even if the formatter is not supposed to output it.

Parameters

record	the log record to format

Note

See the Python logging module documentation for details.

Definition at line 97 of file log.py.

21.62.3.2 formatException()

```
def produtil.log.MasterLogFormatter.formatException (
              self,
              ei )
```

Returns nothing to indicate no exception information should be printed.

Parameters

ei	the exception information to ignore

Definition at line 121 of file log.py.

Referenced by produtil.log.MasterLogFormatter.format().

21.62.3.3 logthread()

```
def produtil.log.MasterLogFormatter.logthread (
              self )
```

The name of the batch thread or process that generated log messages, if the LogRecord does not supply that already.

Definition at line 91 of file log.py.

Referenced by produtil.log.MasterLogFormatter.format().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/log.py

21.63 produtil.log.MasterLogHandler Class Reference

Custom LogHandler for the master process of a multi-process job.

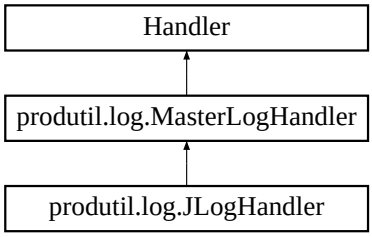
21.63.1 Detailed Description

Custom LogHandler for the master process of a multi-process job.

This is a custom logging Handler class used for multi-process or multi-job batch scripts. It has a higher minimum log level for messages not sent to the jlogfile domain. Also, for every log message, the log file is opened, the message is written and the file is closed. This is done to mimic the postmsg command. Exception information is never sent to the log file.

Definition at line 148 of file log.py.

Inheritance diagram for produtil.log.MasterLogHandler:



Public Member Functions

- def __init__ (self, logger, jlogdomain, otherlevels, joformat, jformat)
  Custom LogHandler for the master process of a multi-process job.
- def stringify_record (self, record)
  Convert a log record to a string.
- def emit (self, record)
  Write a log message.

21.63.2 Constructor & Destructor Documentation

21.63.2.1 __init__()

```
def produtil.log.MasterLogHandler.__init__ (
              self,
              logger,
              jlogdomain,
              otherlevels,
              joformat,
              jformat )
```

Custom LogHandler for the master process of a multi-process job.

This is a custom logging Handler class used for multi-process or multi-job batch scripts. It has a higher minimum log level for messages not sent to the jlogfile domain. Also, for every log message, the log file is opened, the message is written and the file is closed. This is done to mimic the postmsg command. Exception information is never sent to the log file. MasterLogHandler constructor

Parameters

logger	The logging.Logger for the master process.
jlogdomain	The logging domain for the jlogfile.
otherlevels	Log level for any extrema to go to the jlogfile.
joformat	Log format for other streams.
jformat	Log format for the jlogfile stream.

Definition at line 157 of file log.py.

21.63.3 Member Function Documentation

21.63.3.1 emit()

```
def produtil.log.MasterLogHandler.emit (
              self,
              record )
```

Write a log message.

Parameters

record	the log record

Note

See the Python logging module documentation for details.

Definition at line 186 of file log.py.

21.63.3.2 stringify_record()

```
def produtil.log.MasterLogHandler.stringify_record (
              self,
              record )
```

Convert a log record to a string.

Note

See the Python logging module documentation for details.

Returns

a string message to print

Definition at line 170 of file log.py.

Referenced by produtil.log.MasterLogHandler.emit(), and produtil.log.JLogHandler.emit().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/log.py

21.64 config_launcher.METplusLauncher Class Reference

A replacement for the produtil.config.ProdConfig used throughout the METplus system.

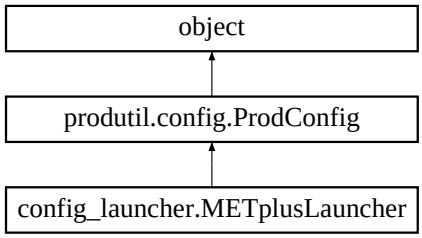
21.64.1 Detailed Description

A replacement for the produtil.config.ProdConfig used throughout the METplus system.

You should never need to instantiate one of these — the launch() and load() functions do that for you. This class is the underlying implementation of most of the functionality described in launch() and load()

Definition at line 373 of file config_launcher.py.

Inheritance diagram for config_launcher.METplusLauncher:



Public Member Functions

- def __init__ (self, conf=None)

  Creates a new METplusLauncher.
- def sanity_check (self)

  Runs nearly all sanity checks.

Additional Inherited Members

21.64.2 Constructor & Destructor Documentation

21.64.2.1 __init__()

```
def config_launcher.METplusLauncher.__init__ (
              self,
              conf = None )
```

Creates a new METplusLauncher.

Parameters

conf	The configuration file.

Definition at line 380 of file config_launcher.py.

21.64.3 Member Function Documentation

21.64.3.1 sanity_check()

```
def config_launcher.METplusLauncher.sanity_check (
              self )
```

Runs nearly all sanity checks.

Runs simple sanity checks on the METplus installation directory and configuration to make sure everything looks okay. May throw a wide variety of exceptions if sanity checks fail.

Definition at line 389 of file config_launcher.py.

The documentation for this class was generated from the following file:

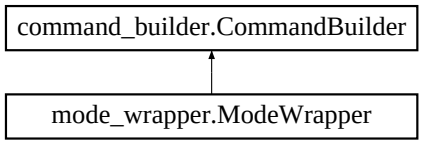
- /home/minnawin/wip_10-31/METplus/ush/config_launcher.py

21.65 mode_wrapper.ModeWrapper Class Reference

21.65.1 Detailed Description

Definition at line 27 of file mode_wrapper.py.

Inheritance diagram for mode_wrapper.ModeWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def **set_output_dir** (self, outdir)
- def **get_command** (self)
- def **run_at_time** (self, init_time, accum, ob_type, fcst_var)

Public Attributes

- **app_path**
- **app_name**
- **outdir**

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/mode_wrapper.py

21.66 produtil.mpi_impl.mpi_impl_base.MPIAllRanksError Class Reference

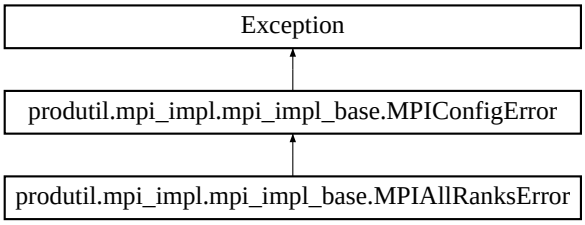
Raised when the allranks=True keyword is sent to mpirun or mpirunner, but the MPI program specification has more than one rank.

21.66.1 Detailed Description

Raised when the allranks=True keyword is sent to mpirun or mpirunner, but the MPI program specification has more than one rank.

Definition at line 23 of file mpi_impl_base.py.

Inheritance diagram for produtil.mpi_impl.mpi_impl_base.MPIAllRanksError:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpi_impl/mpi_impl_base.py

21.67 produtil.mpi_impl.mpi_impl_base.MPIConfigError Class Reference

Base class of MPI configuration exceptions.

21.67.1 Detailed Description

Base class of MPI configuration exceptions.

Definition at line 17 of file mpi_impl_base.py.

Inheritance diagram for produtil.mpi_impl.mpi_impl_base.MPIConfigError:

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpi_impl/mpi_impl_base.py

21.68 produtil.mpi_impl.mpi_impl_base.MPIDisabled Class Reference

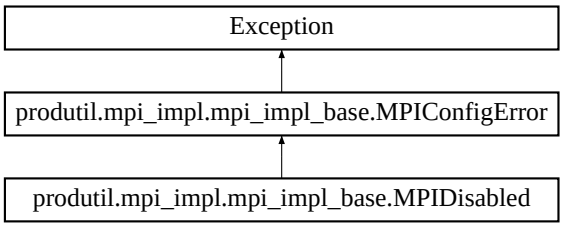
Thrown to MPI is not supported.

21.68.1 Detailed Description

Thrown to MPI is not supported.

Definition at line 28 of file mpi_impl_base.py.

Inheritance diagram for produtil.mpi_impl.mpi_impl_base.MPIDisabled:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpi_impl/mpi_impl_base.py

21.69 produtil.mpi_impl.mpi_impl_base.MPIMixed Class Reference

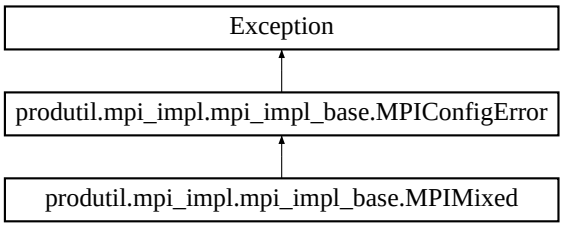
Thrown to indicate serial and parallel processes are being mixed in a single mpi_comm_world.

21.69.1 Detailed Description

Thrown to indicate serial and parallel processes are being mixed in a single mpi_comm_world.

Definition at line 26 of file mpi_impl_base.py.

Inheritance diagram for produtil.mpi_impl.mpi_impl_base.MPIMixed:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpi_impl/mpi_impl_base.py

21.70 produtil.mpiprog.MPIProgSyntaxError Class Reference

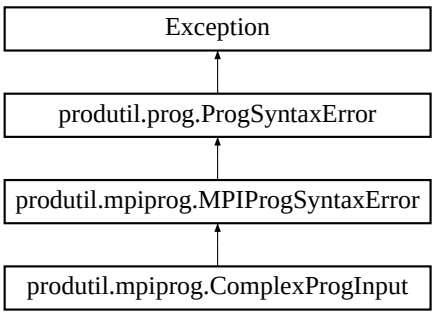
Base class of syntax errors in MPI program specifications.

21.70.1 Detailed Description

Base class of syntax errors in MPI program specifications.

Definition at line 53 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.MPIProgSyntaxError:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.71 produtil.mpiprog.MPIRank Class Reference

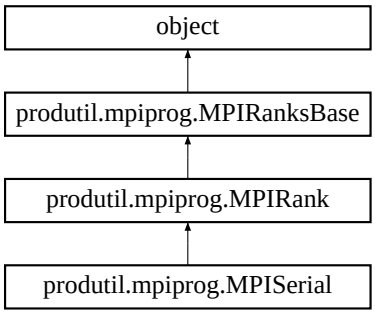
Represents a single MPI rank.

21.71.1 Detailed Description

Represents a single MPI rank.

Definition at line 718 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.MPIRank:



Public Member Functions

- def __init__ (self, arg, logger=None)

  MPIRank constructor.
- def getthreads (self)

  Returns the number of threads requested by this MPI rank, or by each MPI rank in this group of MPI ranks.
- def setthreads (self, nthreads)

  Sets the number of threads requested by each MPI rank within this group of MPI ranks.
- def delthreads (self)

  Removes the request for threads.
- def to_shell (self)

  Return a POSIX sh representation of this MPI rank, if possible.
- def __getitem__ (self, args)

  Adds arguments to this MPI rank's program.
- def __repr__ (self)

  Returns a Pythonic representation of this object for debugging.
- def get_logger (self)

  Returns a logging.Logger for this object, or None.
- def validate (self, more=None)

  Checks to see if this MPIRank is valid, or has errors.
- def args (self)

  Iterates over the executable arguments.
- def copy (self)

  Return a copy of self.
- def nranks (self)

  Returns 1: the number of MPI ranks.
- def ngroups (self)

Returns 1: the number of groups of identical ranks.

- def ranks (self)

  Yields self once: all MPI ranks.

- def groups (self, threads=False)

  Yields (self,1): all groups of identical ranks and the number per group.

- def __add__ (self, other)

  Creates an MPIRanksSPMD or MPIRanksMPMD with this MPIRank and the other ranks.

- def __mul__ (self, factor)

  Creates an MPIRanksSPMD with this MPIRank duplicated factor times.

- def __rmul__ (self, factor)

  Creates an MPIRanksSPMD with this MPIRank duplicated factor times.

- def __eq__ (self, other)

  Returns True if this MPIRank is equal to the other object.

- def check_serial (self)

  Returns (False,True): this is a pure parallel program.

Static Public Attributes

- **threads**

21.71.2 Constructor & Destructor Documentation

21.71.2.1 __init__()

```
def produtil.mpiprog.MPIRank.__init__ (
              self,
              arg,
              logger = None )
```

MPIRank constructor.

Parameters

arg	What program to run. Can be a produtil.prog.Runner, or some way of creating one, such as a program name or list of program+arguments.
logger	a logging.Logger for log messages or None to have no logger.

Definition at line 720 of file mpiprog.py.

21.71.3 Member Function Documentation

21.71.3.1 __add__()

```
def produtil.mpiprog.MPIRank.__add__ (
              self,
              other )
```

Creates an MPIRanksSPMD or MPIRanksMPMD with this MPIRank and the other ranks.

Parameters

other	The other ranks.

Definition at line 832 of file mpiprog.py.

21.71.3.2 __eq__()

```
def produtil.mpiprog.MPIRank.__eq__ (
              self,
              other )
```

Returns True if this MPIRank is equal to the other object.

Definition at line 854 of file mpiprog.py.

21.71.3.3 __getitem__()

```
def produtil.mpiprog.MPIRank.__getitem__ (
              self,
              args )
```

Adds arguments to this MPI rank's program.

Definition at line 774 of file mpiprog.py.

21.71.3.4 __mul__()

```
def produtil.mpiprog.MPIRank.__mul__ (
              self,
              factor )
```

Creates an MPIRanksSPMD with this MPIRank duplicated factor times.

Parameters

factor	the number of times to duplicate

Definition at line 842 of file mpiprog.py.

21.71.3.5 __repr__()

```
def produtil.mpiprog.MPIRank.__repr__ (
              self )
```

Returns a Pythonic representation of this object for debugging.

Definition at line 782 of file mpiprog.py.

Referenced by produtil.prog.Runner.__str__().

21.71.3.6 __rmul__()

```
def produtil.mpiprog.MPIRank.__rmul__ (
              self,
              factor )
```

Creates an MPIRanksSPMD with this MPIRank duplicated factor times.

Parameters

factor	the number of times to duplicate

Definition at line 848 of file mpiprog.py.

21.71.3.7 args()

```
def produtil.mpiprog.MPIRank.args (
              self )
```

Iterates over the executable arguments.

Definition at line 806 of file mpiprog.py.

Referenced by produtil.mpiprog.MPIRank.validate().

21.71.3.8 check_serial()

```
def produtil.mpiprog.MPIRank.check_serial (
              self )
```

Returns (False,True): this is a pure parallel program.

Definition at line 861 of file mpiprog.py.

21.71.3.9 copy()

```
def produtil.mpiprog.MPIRank.copy (
              self )
```

Return a copy of self.

This is a deep copy except for the logger which whose reference is copied.

Definition at line 809 of file mpiprog.py.

Referenced by produtil.mpiprog.MPIRank.__add__(), produtil.mpiprog.MPISerial.__add__(), produtil.mpiprog.M↩PIRank.__getitem__(), and produtil.prog.ImmutableRunner.runner().

21.71.3.10 delthreads()

```
def produtil.mpiprog.MPIRank.delthreads (
              self )
```

Removes the request for threads.

Definition at line 766 of file mpiprog.py.

21.71.3.11 get_logger()

```
def produtil.mpiprog.MPIRank.get_logger (
              self )
```

Returns a logging.Logger for this object, or None.

Definition at line 789 of file mpiprog.py.

21.71.3.12 getthreads()

```
def produtil.mpiprog.MPIRank.getthreads (
              self )
```

Returns the number of threads requested by this MPI rank, or by each MPI rank in this group of MPI ranks.

Definition at line 757 of file mpiprog.py.

21.71.3.13 groups()

```
def produtil.mpiprog.MPIRank.groups (
              self,
              threads = False )
```

Yields (self,1): all groups of identical ranks and the number per group.

Definition at line 825 of file mpiprog.py.

21.71.3.14 ngroups()

```
def produtil.mpiprog.MPIRank.ngroups (
              self )
```

Returns 1: the number of groups of identical ranks.

Definition at line 819 of file mpiprog.py.

21.71.3.15 nranks()

```
def produtil.mpiprog.MPIRank.nranks (
              self )
```

Returns 1: the number of MPI ranks.

Definition at line 816 of file mpiprog.py.

21.71.3.16 ranks()

```
def produtil.mpiprog.MPIRank.ranks (
              self )
```

Yields self once: all MPI ranks.

Definition at line 822 of file mpiprog.py.

21.71.3.17 setthreads()

```
def produtil.mpiprog.MPIRank.setthreads (
              self,
              nthreads )
```

Sets the number of threads requested by each MPI rank within this group of MPI ranks.

Definition at line 761 of file mpiprog.py.

21.71.3.18 to_shell()

```
def produtil.mpiprog.MPIRank.to_shell (
              self )
```

Return a POSIX sh representation of this MPI rank, if possible.

Definition at line 770 of file mpiprog.py.

21.71.3.19 validate()

```
def produtil.mpiprog.MPIRank.validate (
              self,
              more = None )
```

Checks to see if this MPIRank is valid, or has errors.

Parameters

more	Arguments to the executable to validate.

Returns

None if there are no errors, or raises a descriptive exception.

Definition at line 792 of file mpiprog.py.

Referenced by produtil.mpiprog.MPIRank.__init__().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.72 produtil.mpiprog.MPIRanksBase Class Reference

This is the abstract superclass of all classes that represent one or more MPI ranks, including MPI ranks that are actually serial programs.

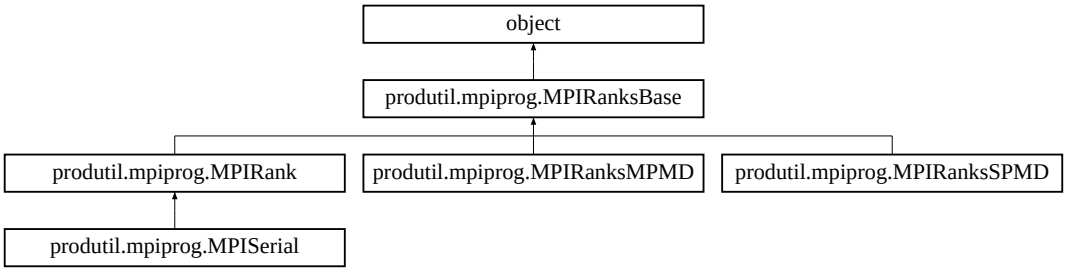
21.72.1 Detailed Description

This is the abstract superclass of all classes that represent one or more MPI ranks, including MPI ranks that are actually serial programs.

Subclasses of MPIRanksBase allow an MPI program to be represented as a tree of MPIRanksBase objects, in such a way that they can be easily converted to a produtil.prog.Runner object for execution. The actual conversion to a Runner is done in the produtil.mpi_impl package (see produtil/mpi_impl/__init__.py)

Definition at line 71 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.MPIRanksBase:



Public Member Functions

- def to_arglist (self, to_shell=False, expand=False, shell_validate=None, pre=[], before=[], between=[], after=[], post=[], extra={}, include_localopts=False)

  This is the underlying implementation of most of the mpi_impl modules, and hence make_runner as well.
- def haslocalopts (self)

  Returns True if setlocalopts(), addlocalopts() or addlocalopt() was called to add localopt values.
- def setlocalopts (self, localopts)

  Sets MPI options that are only meaningful to the currently used MPI configuration.
- def addlocalopts (self, localopts)

  Adds MPI options that are only meaningful to the currently used MPI configuration.
- def addlocalopt (self, localopts)

  Adds one MPI option to the local option list.
- def localoptiter (self)

  Iterates over local MPI configuration options for this rank or group of ranks.
- def getturbomode (self)

  Do we want turbo mode to be enabled for this set of ranks?
- def setturbomode (self, tm)

  Sets the turbo mode setting: on (True) or off (False).
- def delturbomode (self, tm)

  Removes the request for turbo mode to be on or off.
- def **turbo** (self, flag=True)
- def **rpn** (self, count=0)
- def make_runners_immutable (self)

  Returns a copy of this object where all child produtil.prog.Runner objects have been replaced with produtil.prog.ImmutableRunner objects.
- def get_logger (self)

  Returns a logger.Logger object for this MPIRanksBase or one from its child MPIRanksBase objects (if it has any).
- def check_serial (self)

  Returns a tuple (s,p) where s=True if there are serial ranks in this part of the MPI program, and p=True if there are parallel ranks.
- def getranks_per_node (self)

  Returns the number of MPI ranks per node requsted by this MPI rank, or 0 if unspecified.
- def setranks_per_node (self, rpn)

  Sets the number of MPI ranks per node requsted by this MPI rank.
- def delranks_per_node (self)

  Unsets the requested number of ranks per node.
- def nranks (self)

Returns the number of ranks in this part of the MPI program.

- def ranks (self)

  Iterates over all MPIRank objects in this part of the MPI program.
- def ngroups (self)

  Returns the number of groups of repeated MPI ranks in the MPI program.
- def groups (self, threads=False)

  Iterates over all groups of repeating MPI ranks in the MPI program returning tuples (r,c) containing a rank r and the count (number) of that rank c.
- def getthreads (self)

  Returns the number of threads requested by this MPI rank, or by each MPI rank in this group of MPI ranks.
- def setthreads (self, nthreads)

  Sets the number of threads requested by each MPI rank within this group of MPI ranks.
- def delthreads (self)

  Removes the request for threads.
- def __mul__ (self, factor)

  Returns a new set of MPI ranks that consist of this group of ranks repeated "factor" times.
- def __rmul__ (self, other)

  Returns a new set of MPI ranks that consist of this group of ranks repeated "factor" times.
- def __add__ (self, other)

  Returns a new set of MPI ranks that consist of this set of ranks with the "other" set appended.
- def __radd__ (self, other)

  Returns a new set of MPI ranks that consist of the "other" set of ranks with this set appended.
- def isplainexe (self)

  Determines if this set of MPI ranks can be represented by a single serial executable with a single set of arguments run without MPI.
- def to_shell (self)

  Returns a POSIX sh command that will execute the serial program, if possible, or raise a subclass of NotValidPosixSh otherwise.
- def expand_iter (self, expand, threads=False)

  This is a wrapper around ranks() and groups() which will call self.groups() if expand=False.
- def __repr__ (self)

  Returns a string representation of this object intended for debugging.
- def **__eq__** (self, other)

Static Public Attributes

- **turbomode**
- ranks_per_node

  The number of MPI ranks per node or 0 if no specific request is made.
- **threads**

21.72.2 Member Function Documentation

21.72.2.1 __add__()

```
def produtil.mpiprog.MPIRanksBase.__add__ (
            self,
            other )
```

Returns a new set of MPI ranks that consist of this set of ranks with the "other" set appended.

Parameters

other	the data to append

Definition at line 351 of file mpiprog.py.

21.72.2.2 __mul__()

```
def produtil.mpiprog.MPIRanksBase.__mul__ (
              self,
              factor )
```

Returns a new set of MPI ranks that consist of this group of ranks repeated "factor" times.

Parameters

factor	how many times to duplicate

Definition at line 341 of file mpiprog.py.

21.72.2.3 __radd__()

```
def produtil.mpiprog.MPIRanksBase.__radd__ (
              self,
              other )
```

Returns a new set of MPI ranks that consist of the "other" set of ranks with this set appended.

Parameters

other	the data to prepend

Definition at line 356 of file mpiprog.py.

21.72.2.4 __repr__()

```
def produtil.mpiprog.MPIRanksBase.__repr__ (
              self )
```

Returns a string representation of this object intended for debugging.

Definition at line 394 of file mpiprog.py.

Referenced by produtil.prog.Runner.__str__().

21.72.2.5 __rmul__()

```
def produtil.mpiprog.MPIRanksBase.__rmul__ (
              self,
              other )
```

Returns a new set of MPI ranks that consist of this group of ranks repeated "factor" times.

Parameters

other	how many times to duplicate

Definition at line 346 of file mpiprog.py.

21.72.2.6 addlocalopt()

```
def produtil.mpiprog.MPIRanksBase.addlocalopt (
              self,
              localopts )
```

Adds one MPI option to the local option list.

This is an option that is only meaningful to the currently used MPI configuration.

This function lets the ush-level scripts pass platform-specific information to the produtil.mpi_impl package, in order to make platform-specific changes to the way in which MPI programs are launched. These local options are a list of options that are sent for groups of MPI ranks. If the setlocalopts is called in a high-level group of ranks, such as MPIRanksMPMD, then it will apply to all ranks within.

Parameters

localopts	Options to set. These will append the given options to the end of the list of local options. Use addlocalopts() to add a list to the end, or setlocalopts() to replace the entire list.

Returns

self

Definition at line 203 of file mpiprog.py.

21.72.2.7 addlocalopts()

```
def produtil.mpiprog.MPIRanksBase.addlocalopts (
              self,
              localopts )
```

Adds MPI options that are only meaningful to the currently used MPI configuration.

This function lets the ush-level scripts pass platform-specific information to the produtil.mpi_impl package, in order to make platform-specific changes to the way in which MPI programs are launched. These local options are a list of options that are sent for groups of MPI ranks. If the setlocalopts is called in a high-level group of ranks, such as MPIRanksMPMD, then it will apply to all ranks within.

Parameters

localopts	Iterable of options to set. These will extend the list of local options, adding the iterable of specified options to the end. Use addlocalopt() to add one option, or setlocalopt() to replace the entire list.

Returns

self

Definition at line 184 of file mpiprog.py.

21.72.2.8 check_serial()

```
def produtil.mpiprog.MPIRanksBase.check_serial (
              self )
```

Returns a tuple (s,p) where s=True if there are serial ranks in this part of the MPI program, and p=True if there are parallel ranks.

Note that it is possible that both could be True, which is an error. It is also possible that neither are True if there are zero ranks.

Definition at line 263 of file mpiprog.py.

21.72.2.9 delranks_per_node()

```
def produtil.mpiprog.MPIRanksBase.delranks_per_node (
              self )
```

Unsets the requested number of ranks per node.

Definition at line 285 of file mpiprog.py.

21.72.2.10 delthreads()

```
def produtil.mpiprog.MPIRanksBase.delthreads (
              self )
```

Removes the request for threads.

Definition at line 335 of file mpiprog.py.

21.72.2.11 delturbomode()

```
def produtil.mpiprog.MPIRanksBase.delturbomode (
              self,
              tm )
```

Removes the request for turbo mode to be on or off.

Definition at line 240 of file mpiprog.py.

21.72.2.12 expand_iter()

```
def produtil.mpiprog.MPIRanksBase.expand_iter (
              self,
              expand,
              threads = False )
```

This is a wrapper around ranks() and groups() which will call self.groups() if expand=False.

If expand=True, this will call ranks() returning a tuple (rank,1) for each rank.

Parameters

expand	If True, expand groups of identical ranks into one rank of each member
threads	If True, then a third element will be in each tuple: the number of requested threads per MPI rank.

Definition at line 373 of file mpiprog.py.

Referenced by produtil.mpiprog.MPIRanksBase.__repr__(), and produtil.mpiprog.MPIRanksBase.to_arglist().

21.72.2.13 get_logger()

```
def produtil.mpiprog.MPIRanksBase.get_logger (
              self )
```

Returns a logger.Logger object for this MPIRanksBase or one from its child MPIRanksBase objects (if it has any).

If no logger is found, None is returned.

Definition at line 258 of file mpiprog.py.

21.72.2.14 getranks_per_node()

```
def produtil.mpiprog.MPIRanksBase.getranks_per_node (
              self )
```

Returns the number of MPI ranks per node requsted by this MPI rank, or 0 if unspecified.

Definition at line 272 of file mpiprog.py.

21.72.2.15 getthreads()

```
def produtil.mpiprog.MPIRanksBase.getthreads (
              self )
```

Returns the number of threads requested by this MPI rank, or by each MPI rank in this group of MPI ranks.

If different ranks have different numbers of threads, returns the maximum requested. Returns None if no threads are requested.

Definition at line 316 of file mpiprog.py.

21.72.2.16 getturbomode()

```
def produtil.mpiprog.MPIRanksBase.getturbomode (
              self )
```

Do we want turbo mode to be enabled for this set of ranks?

Returns

None if unknown, True if turbo mode is explicitly enabled and False if turbo mode is explicitly disabled.

Definition at line 228 of file mpiprog.py.

21.72.2.17 groups()

```
def produtil.mpiprog.MPIRanksBase.groups (
              self,
              threads = False )
```

Iterates over all groups of repeating MPI ranks in the MPI program returning tuples (r,c) containing a rank r and the count (number) of that rank c.

Parameters

threads	If True, then a three-element tuple is iterated, (r,c,t) where the third element is the number of threads.

Definition at line 307 of file mpiprog.py.

Referenced by produtil.mpiprog.MPIRanksBase.delthreads(), produtil.mpiprog.MPIRanksBase.expand_iter(), and produtil.mpiprog.MPIRanksBase.setthreads().

21.72.2.18 haslocalopts()

```
def produtil.mpiprog.MPIRanksBase.haslocalopts (
              self )
```

Returns True if setlocalopts(), addlocalopts() or addlocalopt() was called to add localopt values.

Definition at line 162 of file mpiprog.py.

21.72.2.19 isplainexe()

```
def produtil.mpiprog.MPIRanksBase.isplainexe (
              self )
```

Determines if this set of MPI ranks can be represented by a single serial executable with a single set of arguments run without MPI.

Returns false by default: this function can only return true for MPISerial.

Definition at line 361 of file mpiprog.py.

21.72.2.20 localoptiter()

```
def produtil.mpiprog.MPIRanksBase.localoptiter (
              self )
```

Iterates over local MPI configuration options for this rank or group of ranks.

Definition at line 223 of file mpiprog.py.

21.72.2.21 make_runners_immutable()

```
def produtil.mpiprog.MPIRanksBase.make_runners_immutable (
              self )
```

Returns a copy of this object where all child produtil.prog.Runner objects have been replaced with produtil.prog.ImmutableRunner objects.

Definition at line 254 of file mpiprog.py.

21.72.2.22 ngroups()

```
def produtil.mpiprog.MPIRanksBase.ngroups (
              self )
```

Returns the number of groups of repeated MPI ranks in the MPI program.

Definition at line 303 of file mpiprog.py.

21.72.2.23 nranks()

```
def produtil.mpiprog.MPIRanksBase.nranks (
              self )
```

Returns the number of ranks in this part of the MPI program.

Definition at line 295 of file mpiprog.py.

Referenced by produtil.mpiprog.MPIRanksSPMD.__add__(), produtil.mpiprog.MPISerial.__add__(), and produtil.↩ mpiprog.MPIRanksBase.to_arglist().

21.72.2.24 ranks()

```
def produtil.mpiprog.MPIRanksBase.ranks (
              self )
```

Iterates over all MPIRank objects in this part of the MPI program.

Definition at line 299 of file mpiprog.py.

Referenced by produtil.mpiprog.MPIRanksBase.expand_iter().

21.72.2.25 setlocalopts()

```
def produtil.mpiprog.MPIRanksBase.setlocalopts (
              self,
              localopts )
```

Sets MPI options that are only meaningful to the currently used MPI configuration.

This function lets the ush-level scripts pass platform-specific information to the produtil.mpi_impl package, in order to make platform-specific changes to the way in which MPI programs are launched. These local options are a list of options that are sent for groups of MPI ranks. If the setlocalopts is called in a high-level group of ranks, such as MPIRanksMPMD, then it will apply to all ranks within.

Parameters

localopts	Options to set. These will replace any options already set. Use addlocalopts to append the end instead.

Returns

self

Definition at line 166 of file mpiprog.py.

21.72.2.26 setranks_per_node()

```
def produtil.mpiprog.MPIRanksBase.setranks_per_node (
              self,
              rpn )
```

Sets the number of MPI ranks per node requsted by this MPI rank.

Definition at line 277 of file mpiprog.py.

21.72.2.27 setthreads()

```
def produtil.mpiprog.MPIRanksBase.setthreads (
              self,
              nthreads )
```

Sets the number of threads requested by each MPI rank within this group of MPI ranks.

Definition at line 328 of file mpiprog.py.

21.72.2.28 setturbomode()

```
def produtil.mpiprog.MPIRanksBase.setturbomode (
              self,
              tm )
```

Sets the turbo mode setting: on (True) or off (False).

Definition at line 233 of file mpiprog.py.

21.72.2.29 to_arglist()

```
def produtil.mpiprog.MPIRanksBase.to_arglist (
                self,
                to_shell = False,
                expand = False,
                shell_validate = None,
                pre = [],
                before = [],
                between = [],
                after = [],
                post = [],
                extra = {},
                include_localopts = False )
```

This is the underlying implementation of most of the mpi_impl modules, and hence make_runner as well.

It converts this group of MPI ranks into a set of arguments suitable for sending to a Runner object or for writing to a command file. This is done by iterating over either all ranks (if expand=True) or groups of repeated ranks (if expand=False), converting their arguments to a list. It prepends an executable, and can insert other arguments in specified locations (given in the pre, before, between, after, and post arguments). It can also use the to_shell argument to convert programs to POSIX sh commands, and it performs simple string interpolation via the "extra" hash.

If to_shell=False then the executable and arguments are inserted directly to the output list. Otherwise (when to↩ _shell=True) the to_shell subroutine is called on the MPIRank object to produce a single argument that contains a shell command. That single argument is then used in place of the executable and arguments. Note that may raise NotValidPosixSh (or a subclass thereof) if the command cannot be expressed as a shell command. In addition, if shell_validate is not None, then it is called on each post-conversion shell argument, and the return value is used instead.

You can specify additional argument lists to be inserted in certain locations. Each argument in those lists will be processed through the % operator, specifying "extra" as the keyword list with two new keywords added: nworld is the number of ranks in the MPI program, and "n" is the number in the current group of repeated ranks if expand=False (n=1 if expand=True). Those argument lists are: pre, before, between, after and post.

Parameters

to_shell	If True, convert executable and arguments to a POSIX sh command instead of inserting them directly.
expand	If True, groups of repeated ranks are expanded.
shell_validate	A function to convert each argument to some "shell-acceptable" version.
pre	Inserted before everything else. This is where you would put the "mpiexec" and any global settings.
before	Inserted before each rank (if expand=True) or group (if expand=False)
between	Inserted between each rank (if expand=True) or group (if expand=False)
after	Inserted after each rank (if expand=True) or group (if expand=False)
post	Appended at the end of the list of arguments.
extra	used for string expansion
include_localopts	If True, then self._localopts is appended between the "before" argument and the command.

Definition at line 82 of file mpiprog.py.

21.72.2.30 to_shell()

```
def produtil.mpiprog.MPIRanksBase.to_shell (
              self )
```

Returns a POSIX sh command that will execute the serial program, if possible, or raise a subclass of NotValid↩ PosixSh otherwise.

Works only on single MPI ranks that are actually MPI wrappers around a serial program (ie.: from mpiserial).

Definition at line 367 of file mpiprog.py.

21.72.3 Member Data Documentation

21.72.3.1 ranks_per_node

```
produtil.mpiprog.MPIRanksBase.ranks_per_node  [static]
```

The number of MPI ranks per node or 0 if no specific request is made.

Definition at line 289 of file mpiprog.py.

Referenced by produtil.mpiprog.MPIRanksSPMD.__add__(), and produtil.mpiprog.MPIRanksBase.delturbomode().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.73 produtil.mpiprog.MPIRanksMPMD Class Reference

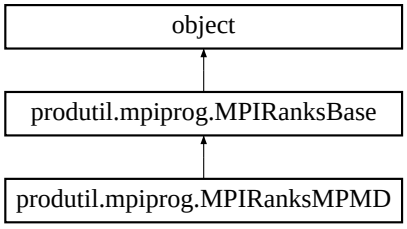
Represents a group of MPI programs, each of which have some number of ranks assigned.

21.73.1 Detailed Description

Represents a group of MPI programs, each of which have some number of ranks assigned.

Definition at line 559 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.MPIRanksMPMD:



Public Member Functions

- def __init__ (self, args)

  MPIRanksMPMD constructor.
- def **setturbomode** (self, tm)
- def **getturbomode** (self)
- def **delturbomode** (self)
- def **setranks_per_node** (self, tm)
- def **getranks_per_node** (self)
- def **delranks_per_node** (self)
- def **setlocalopts** (self, localopts)
- def **addlocalopts** (self, localopts)
- def **addlocalopt** (self, localopt)
- def make_runners_immutable (self)

  Tells each containing element to make its produtil.prog.Runners into produtil.prog.ImmutableRunners so that changes to them will not change the original.
- def __repr__ (self)

  Returns a pythonic description of this object.
- def ngroups (self)

  How many groups of identical repeated ranks are in this MPMD program?
- def nranks (self)

  How many ranks does this program request?
- def groups (self, threads=False)

  Iterates over tuples (rank,count) of groups of identical ranks.
- def ranks (self)

  Iterates over groups of repeated ranks returning the number of ranks each requests.
- def __add__ (self, other)

  Adds more ranks to this program.
- def __radd__ (self, other)

  Prepends more ranks to this program.
- def __mul__ (self, factor)

  Duplicates this MPMD program "factor" times.
- def __rmul__ (self, factor)

  Duplicates this MPMD program "factor" times.
- def check_serial (self)

  Checks to see if this program contains serial (non-MPI) or MPI components.
- def get_logger (self)

  Returns a logging.Logger for the first rank that has one.

Static Public Attributes

- **turbomode**
- **ranks_per_node**

21.73.2 Constructor & Destructor Documentation

21.73.2.1 __init__()

```
def produtil.mpiprog.MPIRanksMPMD.__init__ (
              self,
              args )
```

MPIRanksMPMD constructor.

Parameters

args	an array of MPIRanksBase to execute.

Definition at line 562 of file mpiprog.py.

21.73.3 Member Function Documentation

21.73.3.1 __add__()

```
def produtil.mpiprog.MPIRanksMPMD.__add__ (
              self,
              other )
```

Adds more ranks to this program.

Parameters

other	an MPIRanksMPMD or MPIRanksSPMD to add

Definition at line 668 of file mpiprog.py.

21.73.3.2 __mul__()

```
def produtil.mpiprog.MPIRanksMPMD.__mul__ (
              self,
              factor )
```

Duplicates this MPMD program "factor" times.

Parameters

factor	how many times to duplicate this program.

Definition at line 684 of file mpiprog.py.

21.73.3.3 __radd__()

```
def produtil.mpiprog.MPIRanksMPMD.__radd__ (
              self,
              other )
```

Prepends more ranks to this program.

Parameters

other	an MPIRanksMPMD or MPIRanksSPMD to prepend

Definition at line 676 of file mpiprog.py.

21.73.3.4 __repr__()

```
def produtil.mpiprog.MPIRanksMPMD.__repr__ (
              self )
```

Returns a pythonic description of this object.

Definition at line 627 of file mpiprog.py.

Referenced by produtil.prog.Runner.__str__().

21.73.3.5 __rmul__()

```
def produtil.mpiprog.MPIRanksMPMD.__rmul__ (
              self,
              factor )
```

Duplicates this MPMD program "factor" times.

Parameters

factor	how many times to duplicate this program.

Definition at line 690 of file mpiprog.py.

21.73.3.6 check_serial()

```
def produtil.mpiprog.MPIRanksMPMD.check_serial (
              self )
```

Checks to see if this program contains serial (non-MPI) or MPI components.

Returns

a tuple (serial,parallel) where serial is True if there are serial components, and parallel is True if there are parallel components.

Definition at line 695 of file mpiprog.py.

21.73.3.7 get_logger()

```
def produtil.mpiprog.MPIRanksMPMD.get_logger (
              self )
```

Returns a logging.Logger for the first rank that has one.

Definition at line 708 of file mpiprog.py.

21.73.3.8 groups()

```
def produtil.mpiprog.MPIRanksMPMD.groups (
              self,
              threads = False )
```

Iterates over tuples (rank,count) of groups of identical ranks.

Definition at line 651 of file mpiprog.py.

21.73.3.9 make_runners_immutable()

```
def produtil.mpiprog.MPIRanksMPMD.make_runners_immutable (
              self )
```

Tells each containing element to make its produtil.prog.Runners into produtil.prog.ImmutableRunners so that changes to them will not change the original.

Definition at line 622 of file mpiprog.py.

21.73.3.10 ranks()

```
def produtil.mpiprog.MPIRanksMPMD.ranks (
              self )
```

Iterates over groups of repeated ranks returning the number of ranks each requests.

Definition at line 662 of file mpiprog.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.74 produtil.mpiprog.MPIRanksSPMD Class Reference

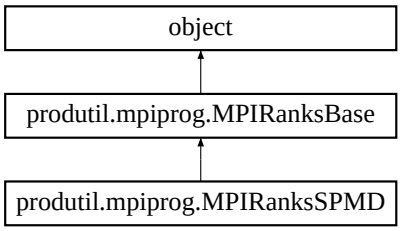
Represents one MPI program duplicated across many ranks.

21.74.1 Detailed Description

Represents one MPI program duplicated across many ranks.

Definition at line 421 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.MPIRanksSPMD:



Public Member Functions

- def __init__ (self, mpirank, count)

  MPIRanksSPMD constructor.
- def getranks_per_node (self)

  Returns the number of MPI ranks per node requsted by this MPI rank, or 0 if unspecified.
- def setranks_per_node (self, rpn)

  Sets the number of MPI ranks per node requsted by this MPI rank.
- def delranks_per_node (self)

  Unsets the requested number of ranks per node.
- def **setturbomode** (self, tm)
- def **getturbomode** (self)
- def **delturbomode** (self)
- def **setlocalopts** (self, localopts)
- def **addlocalopts** (self, localopts)
- def **addlocalopt** (self, localopt)
- def make_runners_immutable (self)

  Returns a new MPIRanksSPMD with an immutable version of self._mpirank.
- def __repr__ (self)

  Returns "X∗N" where X is the MPI program and N is the number of ranks.
- def ngroups (self)

  Returns 1 or 0: 1 if there are ranks and 0 if there are none.
- def groups (self, threads=False)

  Yields a tuple (X,N) where X is the mpi program and N is the number of ranks.
- def copy (self)

  Returns a deep copy of self.
- def ranks (self)

  Iterates over MPI ranks within self.
- def nranks (self)

  Returns the number of ranks this program requests.
- def __mul__ (self, factor)

  Multiply the number of requested ranks by some factor.
- def __rmul__ (self, factor)

  Multiply the number of requested ranks by some factor.
- def __add__ (self, other)

  Add some new ranks to self.
- def check_serial (self)

  Checks to see if this program contains serial (non-MPI) or MPI components.
- def get_logger (self)

  Returns my MPI program's logger.

Static Public Attributes

- **ranks_per_node**
- **turbomode**

21.74.2 Constructor & Destructor Documentation

21.74.2.1 __init__()

```
def produtil.mpiprog.MPIRanksSPMD.__init__ (
              self,
              mpirank,
              count )
```

MPIRanksSPMD constructor.

Parameters

mpirank	the program to run
count	how many times to run it

Definition at line 423 of file mpiprog.py.

21.74.3 Member Function Documentation

21.74.3.1 __add__()

```
def produtil.mpiprog.MPIRanksSPMD.__add__ (
              self,
              other )
```

Add some new ranks to self.

If they are not identical to the MPI program presently requested, this returns a new MPIRanksMPMD.

Definition at line 524 of file mpiprog.py.

21.74.3.2 __mul__()

```
def produtil.mpiprog.MPIRanksSPMD.__mul__ (
              self,
              factor )
```

Multiply the number of requested ranks by some factor.

Definition at line 514 of file mpiprog.py.

21.74.3.3 __repr__()

```
def produtil.mpiprog.MPIRanksSPMD.__repr__ (
              self )
```

Returns "X∗N" where X is the MPI program and N is the number of ranks.

Definition at line 480 of file mpiprog.py.

Referenced by produtil.prog.Runner.__str__().

21.74.3.4 __rmul__()

```
def produtil.mpiprog.MPIRanksSPMD.__rmul__ (
              self,
              factor )
```

Multiply the number of requested ranks by some factor.

Definition at line 519 of file mpiprog.py.

21.74.3.5 check_serial()

```
def produtil.mpiprog.MPIRanksSPMD.check_serial (
              self )
```

Checks to see if this program contains serial (non-MPI) or MPI components.

Returns

a tuple (serial,parallel) where serial is True if there are serial components, and parallel is True if there are parallel components. If there are no components, returns (False,False)

Definition at line 542 of file mpiprog.py.

21.74.3.6 copy()

```
def produtil.mpiprog.MPIRanksSPMD.copy (
              self )
```

Returns a deep copy of self.

Definition at line 497 of file mpiprog.py.

Referenced by produtil.mpiprog.MPIRanksSPMD.__add__(), produtil.mpiprog.MPIRank.__add__(), produtil.↩ mpiprog.MPISerial.__add__(), produtil.mpiprog.MPIRank.__getitem__(), produtil.mpiprog.MPIRanksSPMD.__↩ mul__(), produtil.mpiprog.MPIRanksSPMD.__rmul__(), and produtil.prog.ImmutableRunner.runner().

21.74.3.7 delranks_per_node()

```
def produtil.mpiprog.MPIRanksSPMD.delranks_per_node (
              self )
```

Unsets the requested number of ranks per node.

Definition at line 444 of file mpiprog.py.

21.74.3.8 get_logger()

```
def produtil.mpiprog.MPIRanksSPMD.get_logger (
              self )
```

Returns my MPI program's logger.

Definition at line 553 of file mpiprog.py.

21.74.3.9 getranks_per_node()

```
def produtil.mpiprog.MPIRanksSPMD.getranks_per_node (
              self )
```

Returns the number of MPI ranks per node requsted by this MPI rank, or 0 if unspecified.

Definition at line 434 of file mpiprog.py.

21.74.3.10 groups()

```
def produtil.mpiprog.MPIRanksSPMD.groups (
              self,
              threads = False )
```

Yields a tuple (X,N) where X is the mpi program and N is the number of ranks.

Definition at line 490 of file mpiprog.py.

21.74.3.11 make_runners_immutable()

```
def produtil.mpiprog.MPIRanksSPMD.make_runners_immutable (
              self )
```

Returns a new MPIRanksSPMD with an immutable version of self._mpirank.

Definition at line 476 of file mpiprog.py.

21.74.3.12 ngroups()

```
def produtil.mpiprog.MPIRanksSPMD.ngroups (
              self )
```

Returns 1 or 0: 1 if there are ranks and 0 if there are none.

Definition at line 484 of file mpiprog.py.

21.74.3.13 nranks()

```
def produtil.mpiprog.MPIRanksSPMD.nranks (
              self )
```

Returns the number of ranks this program requests.

Definition at line 508 of file mpiprog.py.

21.74.3.14 ranks()

```
def produtil.mpiprog.MPIRanksSPMD.ranks (
              self )
```

Iterates over MPI ranks within self.

Definition at line 503 of file mpiprog.py.

21.74.3.15 setranks_per_node()

```
def produtil.mpiprog.MPIRanksSPMD.setranks_per_node (
              self,
              rpn )
```

Sets the number of MPI ranks per node requsted by this MPI rank.

Definition at line 439 of file mpiprog.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.75 produtil.mpiprog.MPISerial Class Reference

Represents a single rank of an MPI program that is actually running a serial program.

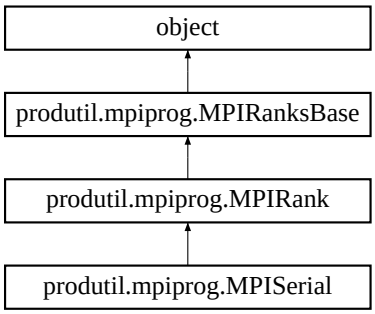
21.75.1 Detailed Description

Represents a single rank of an MPI program that is actually running a serial program.

This is supported directly by some MPI implementations while others require kludges to work properly.

Definition at line 867 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.MPISerial:



Public Member Functions

- def __init__ (self, runner, logger=None)

  MPISerial constructor.
- def make_runners_immutable (self)

  Creates a version of self with a produtil.prog.ImmutableRunner child.
- def copy (self)

  Duplicates self.
- def __repr__ (self)

  Returns a pythonic string representation of self for debugging.
- def args (self)

  Iterates over command arguments of the child serial program.
- def __add__ (self, other)

  Add some new ranks to self.
- def get_logger (self)

  Returns my logging.Logger that I use for log messages.
- def **runner** (self)
- def validate (self)

  Does nothing.
- def __eq__ (self, other)

  Returns True if other is an MPISerial with the same Runner, False otherwise.
- def check_serial (self)

  Returns (True,False) because this is a serial program (True,) and not a parallel program (,False).
- def isplainexe (self)

  Returns True if the child serial program is a plain executable, False otherwise.
- def to_shell (self)

  Returns a POSIX sh version of the child serial program.

Additional Inherited Members

21.75.2 Constructor & Destructor Documentation

21.75.2.1 __init__()

```
def produtil.mpiprog.MPISerial.__init__ (
              self,
              runner,
              logger = None )
```

MPISerial constructor.

Definition at line 871 of file mpiprog.py.

21.75.3 Member Function Documentation

21.75.3.1 __add__()

```
def produtil.mpiprog.MPISerial.__add__ (
              self,
              other )
```

Add some new ranks to self.

If they are not identical to the MPI program presently requested, this returns a new MPIRanksMPMD.

Definition at line 901 of file mpiprog.py.

21.75.3.2 __eq__()

```
def produtil.mpiprog.MPISerial.__eq__ (
              self,
              other )
```

Returns True if other is an MPISerial with the same Runner, False otherwise.

Parameters

other	the other object to compare against.

Definition at line 919 of file mpiprog.py.

21.75.3.3 __repr__()

```
def produtil.mpiprog.MPISerial.__repr__ (
              self )
```

Returns a pythonic string representation of self for debugging.

Definition at line 894 of file mpiprog.py.

Referenced by produtil.prog.Runner.__str__().

21.75.3.4 args()

```
def produtil.mpiprog.MPISerial.args (
              self )
```

Iterates over command arguments of the child serial program.

Definition at line 897 of file mpiprog.py.

21.75.3.5 check_serial()

```
def produtil.mpiprog.MPISerial.check_serial (
              self )
```

Returns (True,False) because this is a serial program (True,) and not a parallel program (,False).

Definition at line 928 of file mpiprog.py.

21.75.3.6 copy()

```
def produtil.mpiprog.MPISerial.copy (
              self )
```

Duplicates self.

Definition at line 885 of file mpiprog.py.

Referenced by produtil.prog.ImmutableRunner.runner().

21.75.3.7 get_logger()

```
def produtil.mpiprog.MPISerial.get_logger (
              self )
```

Returns my logging.Logger that I use for log messages.

Definition at line 909 of file mpiprog.py.

21.75.3.8 isplainexe()

```
def produtil.mpiprog.MPISerial.isplainexe (
              self )
```

Returns True if the child serial program is a plain executable, False otherwise.

See produtil.prog.Runner.isplainexe() for details.

Definition at line 932 of file mpiprog.py.

21.75.3.9 make_runners_immutable()

```
def produtil.mpiprog.MPISerial.make_runners_immutable (
              self )
```

Creates a version of self with a produtil.prog.ImmutableRunner child.

Definition at line 879 of file mpiprog.py.

21.75.3.10 to_shell()

```
def produtil.mpiprog.MPISerial.to_shell (
              self )
```

Returns a POSIX sh version of the child serial program.

Definition at line 937 of file mpiprog.py.

21.75.3.11 validate()

```
def produtil.mpiprog.MPISerial.validate (
              self )
```

Does nothing.

Definition at line 917 of file mpiprog.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.76 produtil.mpi_impl.mpi_impl_base.MPISerialMissing Class Reference

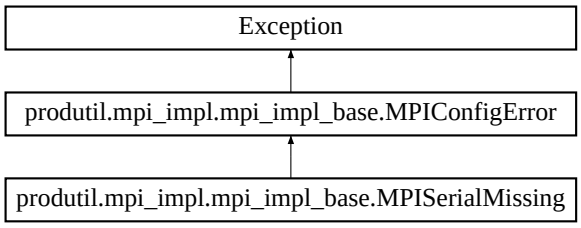
Raised when the mpiserial program is required, but is missing.

21.76.1 Detailed Description

Raised when the mpiserial program is required, but is missing.

Definition at line 21 of file mpi_impl_base.py.

Inheritance diagram for produtil.mpi_impl.mpi_impl_base.MPISerialMissing:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpi_impl/mpi_impl_base.py

21.77 produtil.prog.MultipleStderr Class Reference

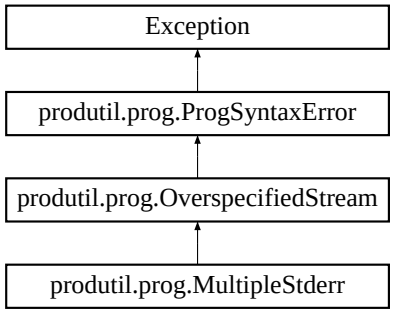
Raised when the caller specifies more than one destination for a Runner's stderr.

21.77.1 Detailed Description

Raised when the caller specifies more than one destination for a Runner's stderr.

Definition at line 58 of file prog.py.

Inheritance diagram for produtil.prog.MultipleStderr:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.78 produtil.prog.MultipleStdin Class Reference

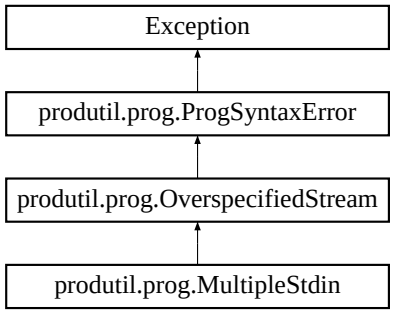
Raised when the caller specifies more than one source for the stdin of a Runner.

21.78.1 Detailed Description

Raised when the caller specifies more than one source for the stdin of a Runner.

Definition at line 52 of file prog.py.

Inheritance diagram for produtil.prog.MultipleStdin:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.79 produtil.prog.MultipleStdout Class Reference

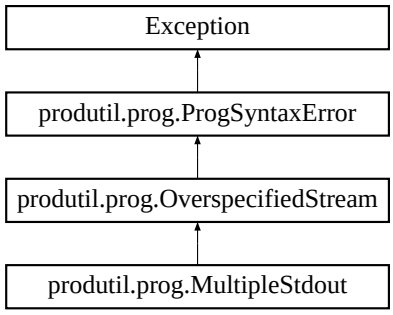
Raised when the caller specifies more than one destination for a Runner's stdout.

21.79.1 Detailed Description

Raised when the caller specifies more than one destination for a Runner's stdout.

Definition at line 55 of file prog.py.

Inheritance diagram for produtil.prog.MultipleStdout:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.80 produtil.cd.NamedDir Class Reference

This subclass of TempDir takes a directory name, instead of generating one automatically.

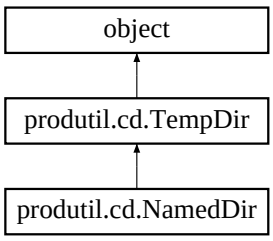
21.80.1 Detailed Description

This subclass of TempDir takes a directory name, instead of generating one automatically.

By default, it will NOT delete the directory upon **exit**. That can be overridden by specifying keep=False.

Definition at line 228 of file cd.py.

Inheritance diagram for produtil.cd.NamedDir:



Public Member Functions

- def __init__ (self, dirname, keep=True, logger=None, keep_on_error=True, add_perms=0, remove_perms=0, rm_first=False)

  Create a NamedDir for the specified directory.
- def name_make_dir (self)

  Replacement for the TempDir.name_make_dir.

Public Attributes

- dirname

  The directory name specified in the constructor.

21.80.2 Constructor & Destructor Documentation

21.80.2.1 __init__()

```
def produtil.cd.NamedDir.__init__ (
              self,
              dirname,
              keep = True,
              logger = None,
              keep_on_error = True,
              add_perms = 0,
              remove_perms = 0,
              rm_first = False )
```

Create a NamedDir for the specified directory.

The given logger is used to log messages. There are two deletion vs. non-deletion options:

Parameters

dirname	The directory name
keep	If False, the file is deleted upon successful return of the "with" block. If True, the file is kept upon successful return.
logger	A logging.logger for log messages
add_perms	Permissions to add to the directory after cding into it. Default: none.
remove_perms	Permissions to remove from the directory after cding into it. Default: none.
keep_on_error	Controls deletion upon catching of an Exception or GeneratorException (or subclass thereof).
rm_first	If the directory already exists, delete it first and make a new one before cding to it.

Definition at line 234 of file cd.py.

21.80.3 Member Function Documentation

21.80.3.1 name_make_dir()

```
def produtil.cd.NamedDir.name_make_dir (
              self )
```

Replacement for the TempDir.name_make_dir.

Uses the directory name specified in the constructor.

Definition at line 262 of file cd.py.

21.80.4 Member Data Documentation

21.80.4.1 dirname

```
produtil.cd.NamedDir.dirname
```

The directory name specified in the constructor.

Definition at line 258 of file cd.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cd.py

21.81 produtil.cluster.NOAAGAEA Class Reference

Represents the NOAA GAEA cluster.

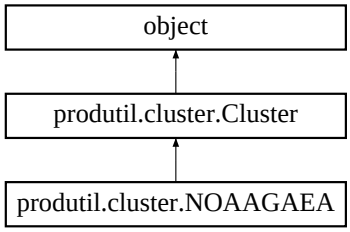
21.81.1 Detailed Description

Represents the NOAA GAEA cluster.

Allows ACLs to be used for restricted data, and specifies that group quotas are in use.

Definition at line 178 of file cluster.py.

Inheritance diagram for produtil.cluster.NOAAGAEA:



Public Member Functions

- def __init__ (self)
  constructor for NOAAGAEA

Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cluster.py

21.82 produtil.cluster.NOAAJet Class Reference

The NOAA Jet Cluster.

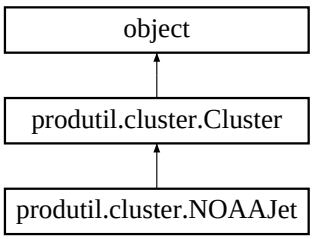
21.82.1 Detailed Description

The NOAA Jet Cluster.

Represents the NOAA Jet cluster, which has non-functional ACL support. Will report that ACLs are supported, but should not be used. Also, group quotas are in use. That means that there is no means by which to restrict access control, so no_access_control() will return True.

Definition at line 165 of file cluster.py.

Inheritance diagram for produtil.cluster.NOAAJet:



Public Member Functions

- def __init__ (self)
  constructor for NOAAJet

Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cluster.py

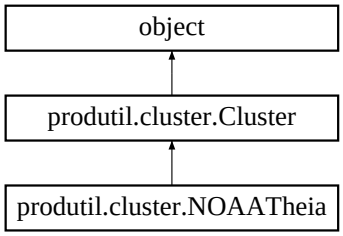
21.83 produtil.cluster.NOAATheia Class Reference

21.83.1 Detailed Description

```
Represents the NOAA Theia cluster.  Does not allow ACLs,
assumes no group quotas (fileset quotas instead).
```

Definition at line 210 of file cluster.py.

Inheritance diagram for produtil.cluster.NOAATheia:



Public Member Functions

- def **__init__** (self)

Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cluster.py

21.84 produtil.cluster.NOAAWCOSS Class Reference

Represents the NOAA WCOSS clusters, Tide, Gyre and the test system Eddy.

21.84.1 Detailed Description

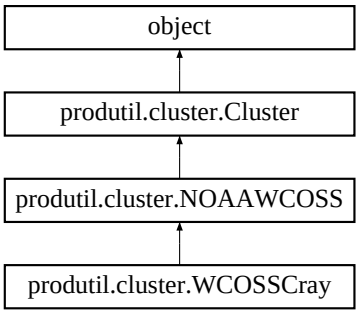
Represents the NOAA WCOSS clusters, Tide, Gyre and the test system Eddy.

Automatically determines which WCOSS the program is on based on the first letter of socket.gethostname(). Will report no ACL support, and no group quotas. Hence, the cluster should use group IDs for access control.

The production accessor is no longer a public member variable: it is now a property, which may open the /etc/prod file. The result of the self.production property is cached for up to prod_cache_time seconds. That time can be specified in the constructor, and defaults to 30 seconds.

Definition at line 217 of file cluster.py.

Inheritance diagram for produtil.cluster.NOAAWCOSS:



Public Member Functions

- def __init__ (self, prod_cache_time=30, name=None)

  Creates a NOAAWCOSS object, and optionally specifies the time for which the result of self.production should be cached.
- def uncache (self)

  Clears the cached value of self.production so the next call will return up-to-date information.
- def partition (self)

  Returns "phase1" on phase 1, or "phase2" on phase 2.
- def wcoss_phase (self)

  Returns integer 1 or 2 for WCOSS Phase 1 or WCOSS Phase 2, respectively.
- def production (self)

  Is this the WCOSS production machine?

Additional Inherited Members

21.84.2 Constructor & Destructor Documentation

21.84.2.1 __init__()

```
def produtil.cluster.NOAAWCOSS.__init__ (
              self,
              prod_cache_time = 30,
              name = None )
```

Creates a NOAAWCOSS object, and optionally specifies the time for which the result of self.production should be cached.

Default: 30 seconds.

Parameters

prod_cache_time	how long to cache the prod vs. dev information, in seconds

Definition at line 231 of file cluster.py.

21.84.3 Member Function Documentation

21.84.3.1 partition()

```
produtil.cluster.NOAAWCOSS.partition (
              self )
```

Returns "phase1" on phase 1, or "phase2" on phase 2.

The WCOSSCray.partition overrides this on the Cray.

Definition at line 259 of file cluster.py.

21.84.3.2 production()

```
def produtil.cluster.NOAAWCOSS.production (
              self )
```

Is this the WCOSS production machine?

The name of the WCOSS production machine: tide, gyre, surge or luna as determined by the /etc/prod file.

Returns

True or False: is this the WCOSS production machine?

Note

The return value may change during the execution of this program if a production switch happened. A cached value is returned if the values is not too old. To force a refresh, call uncache() first.

Warning

The check requires opening and parsing the /etc/prod file, so the runtime is likely several milliseconds when the cache times out.

Definition at line 291 of file cluster.py.

21.84.3.3 uncache()

```
def produtil.cluster.NOAAWCOSS.uncache (
              self )
```

Clears the cached value of self.production so the next call will return up-to-date information.

Definition at line 247 of file cluster.py.

21.84.3.4 wcoss_phase()

```
def produtil.cluster.NOAAWCOSS.wcoss_phase (
              self )
```

Returns integer 1 or 2 for WCOSS Phase 1 or WCOSS Phase 2, respectively.

Returns 0 if the request is invalid, such as on WCOSS Cray.

Scans /proc/cpuinfo for processor 32. If processor 32 is found, you are on Phase 2, which has 48 virtual processors per node. Otherwise, you are on Phase 1.

Note

Cached results are returned if available. Use uncache() to force regeneration of the information.

Returns

1 for WCOSS Phase 1, 2 for WCOSS Phase 2, or 0 for WCOSS Cray

Definition at line 266 of file cluster.py.

Referenced by produtil.cluster.NOAAWCOSS.partition().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cluster.py

21.85 produtil.cluster.NOAAZeus Class Reference

Represents the NOAA Zeus cluster.

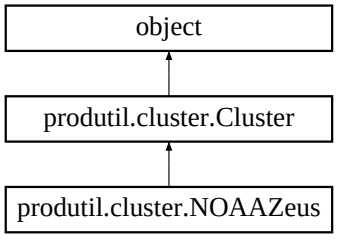
21.85.1 Detailed Description

Represents the NOAA Zeus cluster.

Allows ACLs to be used for restricted data, and specifies that group quotas are in use.

Definition at line 186 of file cluster.py.

Inheritance diagram for produtil.cluster.NOAAZeus:



Public Member Functions

- def __init__ (self)

  Constructor for NOAAZeus.

Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cluster.py

21.86 produtil.pipeline.NoMoreProcesses Class Reference

Raised when the produtil.sigsafety package catches a fatal signal.

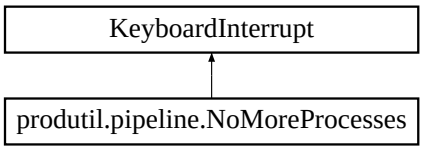
21.86.1 Detailed Description

Raised when the produtil.sigsafety package catches a fatal signal.

Indicates to callers that the thread should exit.

Definition at line 15 of file pipeline.py.

Inheritance diagram for produtil.pipeline.NoMoreProcesses:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/pipeline.py

21.87 produtil.numerics.NoNearbyValues Class Reference

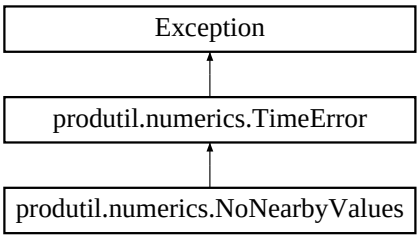
Raised when an operation has a set of known times, but another provided time is not near one of those known times.

21.87.1 Detailed Description

Raised when an operation has a set of known times, but another provided time is not near one of those known times.

Definition at line 26 of file numerics.py.

Inheritance diagram for produtil.numerics.NoNearbyValues:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.88 produtil.prog.NoSuchRedirection Class Reference

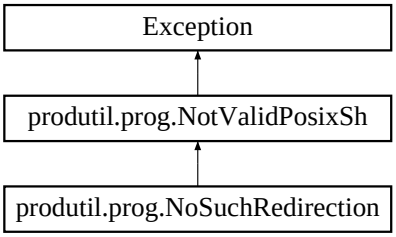
Raised when trying to convert a pipeline of Runners to a POSIX sh string, if a redirection in the pipeline cannot be expressed in POSIX sh.

21.88.1 Detailed Description

Raised when trying to convert a pipeline of Runners to a POSIX sh string, if a redirection in the pipeline cannot be expressed in POSIX sh.

Definition at line 69 of file prog.py.

Inheritance diagram for produtil.prog.NoSuchRedirection:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.89 produtil.atparse.NoSuchVariable Class Reference

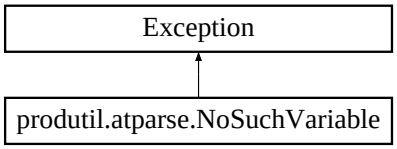
Raised when a script requests an unknown variable.

21.89.1 Detailed Description

Raised when a script requests an unknown variable.

Definition at line 21 of file atparse.py.

Inheritance diagram for produtil.atparse.NoSuchVariable:



Public Member Functions

- def __init__ (self, infile, varname, line=None)
  NoSuchVariable constructor.

Public Attributes

- infile
  The file that caused the problem.
- varname
  The problematic variable name.
- line
  The line number that caused the problem.

21.89.2 Constructor & Destructor Documentation

21.89.2.1 __init__()

```
def produtil.atparse.NoSuchVariable.__init__ (
              self,
              infile,
              varname,
              line = None )
```

NoSuchVariable constructor.

Parameters

infile	the input file that caused problems
varname	the variable that does not exist
line	the line number of the problematic line

Definition at line 23 of file atparse.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/atparse.py

21.90 produtil.numerics.NoTimespan Class Reference

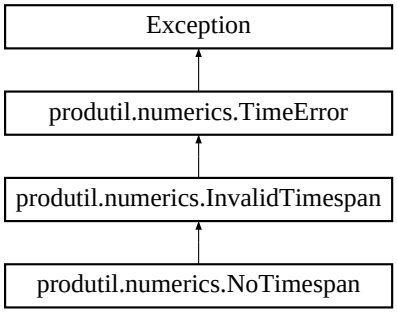
Raised when a timespan was expected, but none was available.

21.90.1 Detailed Description

Raised when a timespan was expected, but none was available.

Definition at line 49 of file numerics.py.

Inheritance diagram for produtil.numerics.NoTimespan:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.91 produtil.numerics.NotInTimespan Class Reference

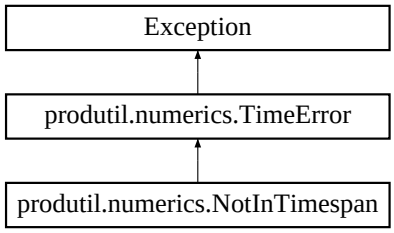
Raised when a time is outside the range of times being processed by a function.

21.91.1 Detailed Description

Raised when a time is outside the range of times being processed by a function.

Definition at line 23 of file numerics.py.

Inheritance diagram for produtil.numerics.NotInTimespan:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.92 produtil.mpiprog.NotMPIProg Class Reference

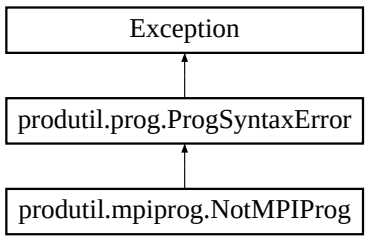
Raised when an MPI program was expected but something else was given.

21.92.1 Detailed Description

Raised when an MPI program was expected but something else was given.

Definition at line 58 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.NotMPIProg:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.93 produtil.mpiprog.NotSerialProg Class Reference

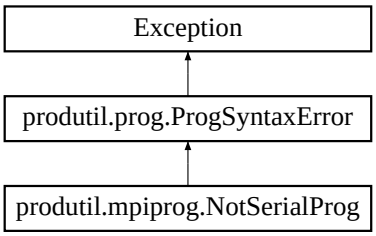
Raised when a serial program was expected, but something else was given.

21.93.1 Detailed Description

Raised when a serial program was expected, but something else was given.

Definition at line 61 of file mpiprog.py.

Inheritance diagram for produtil.mpiprog.NotSerialProg:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpiprog.py

21.94 produtil.prog.NotValidPosixSh Class Reference

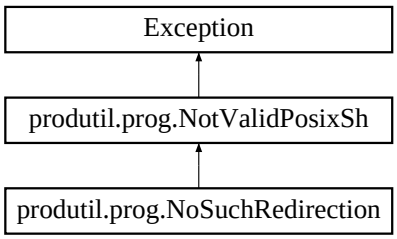
Base class of exceptions that are raised when converting a Runner or pipeline of Runners to a POSIX sh command, if the Runner cannot be expressed as POSIX sh.

21.94.1 Detailed Description

Base class of exceptions that are raised when converting a Runner or pipeline of Runners to a POSIX sh command, if the Runner cannot be expressed as POSIX sh.

Definition at line 65 of file prog.py.

Inheritance diagram for produtil.prog.NotValidPosixSh:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.95 produtil.prog.NotValidPosixShString Class Reference

Raised when converting a Runner or pipeline of Runners to a POSIX sh string.

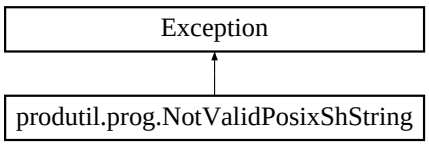
21.95.1 Detailed Description

Raised when converting a Runner or pipeline of Runners to a POSIX sh string.

If a string is sent to a program's stdin, this is raised when that string cannot be expressed in POSIX sh.

Definition at line 73 of file prog.py.

Inheritance diagram for produtil.prog.NotValidPosixShString:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.96 produtil.mpi_impl.mpi_impl_base.OpenMPDisabled Class Reference

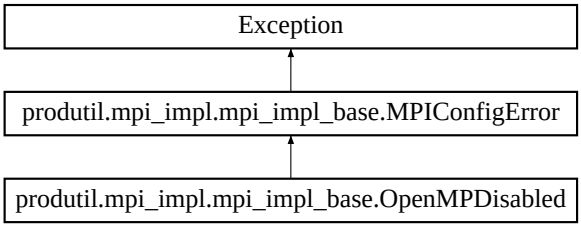
Raised when OpenMP is not supported by the present implementation.

21.96.1 Detailed Description

Raised when OpenMP is not supported by the present implementation.

Definition at line 30 of file mpi_impl_base.py.

Inheritance diagram for produtil.mpi_impl.mpi_impl_base.OpenMPDisabled:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpi_impl/mpi_impl_base.py

21.97 produtil.prog.OutIsError Class Reference

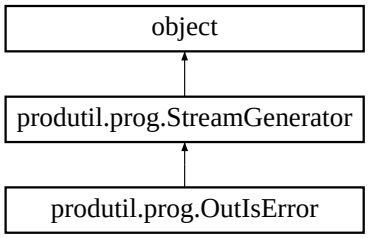
Instructs a Runner to send stderr to stdout.

21.97.1 Detailed Description

Instructs a Runner to send stderr to stdout.

Definition at line 271 of file prog.py.

Inheritance diagram for produtil.prog.OutIsError:



Public Member Functions

- def __init__ (self)
  OutIsError constructor.
- def copy (self)
  Returns a new OutIsError object.
- def to_shell (self)
  Returns "2>&1".
- def repr_for_in (self)
  This should never be called.
- def repr_for_out (self)
  Part of the representation of Runner.__repr__.
- def __eq__ (self, other)
  Is the other object an OutIsError?

21.97.2 Constructor & Destructor Documentation

21.97.2.1 __init__()

```
def produtil.prog.OutIsError.__init__ (
              self )
```

OutIsError constructor.

Definition at line 273 of file prog.py.

21.97.3 Member Function Documentation

21.97.3.1 __eq__()

```
def produtil.prog.OutIsError.__eq__ (
              self,
              other )
```

Is the other object an OutIsError?

Parameters

other	the other object to analyze.

Definition at line 292 of file prog.py.

21.97.3.2 copy()

```
def produtil.prog.OutIsError.copy (
              self )
```

Returns a new OutIsError object.

Definition at line 275 of file prog.py.

Referenced by produtil.prog.ImmutableRunner.runner().

21.97.3.3 repr_for_in()

```
def produtil.prog.OutIsError.repr_for_in (
              self )
```

This should never be called.

It returns ".err2out()".

Definition at line 284 of file prog.py.

21.97.3.4 repr_for_out()

```
def produtil.prog.OutIsError.repr_for_out (
              self )
```

Part of the representation of Runner.__repr__.

Returns ".err2out()" which instructs a Runner to send stderr to stdout.

Definition at line 287 of file prog.py.

Referenced by produtil.prog.StreamGenerator.repr_for_err().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.98 confdoc.override Class Reference

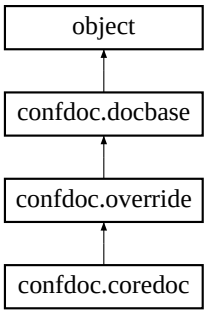
Subclass of docbase for documenting files that override the base configuration.

21.98.1 Detailed Description

Subclass of docbase for documenting files that override the base configuration.

Definition at line 354 of file confdoc.py.

Inheritance diagram for confdoc.override:



Public Member Functions

- def __init__ (self, basename, parent, filepart=None, replace=False)

  Class override constructor.
- def secanch (self, section, where=None)

  Returns the anchor for the specified section.
- def optanch (self, section, option, where=None)

  Returns the anchor for a specified section and option.
- def section_inc (self, section, inc)

  Sets the @inc list for a conf section.
- def file_block (self, basename, brief, detail)

  Adds a documentation block that is not associated with any section or option.
- def add_section (self, section, brief, detail, replace=True)

  Adds documentation for a conf section.
- def add_option (self, section, option, ivalue, brief, detail, basename=None, replace=True)

  Adds documentation for an option in a conf section.
- def find_optbrief (self, section, option)

  Finds the brief documentation for a section.
- def find_secbrief (self, section)

  Finds the brief documentation for a section.
- def find_secdoc (self, section, detail)

  Finds the brief and detailed documentation for a section.
- def print_subdoc (self, s)

  Prints the documentation to the specified stream.
- def print_sec_opt (self, s)

  Prints the section and option part of the documentation to the given stream.

Protected Member Functions

- def _secsec (self, section, brief, detail)

  Generates the contents of the documentation section that documents the specified conf section.

Additional Inherited Members

21.98.2 Constructor & Destructor Documentation

21.98.2.1 __init__()

```
def confdoc.override.__init__ (
              self,
              basename,
              parent,
              filepart = None,
              replace = False )
```

Class override constructor.

Parameters

basename	The file basename
parent	The parent docbase object that documents this group of conf files, or all conf options.
filepart	The modified filename for anchors. This should be the basename with "." replaced by "_".
replace	If True, replace existing documentation when new values are found, otherwise ignore new docs

Definition at line 357 of file confdoc.py.

21.98.3 Member Function Documentation

21.98.3.1 _secsec()

```
def confdoc.override._secsec (
              self,
              section,
              brief,
              detail )  [protected]
```

Generates the contents of the documentation section that documents the specified conf section.

Parameters

section	the conf section name
brief	the brief documentation
detail	the detailed documentation

Returns

the section text

Definition at line 397 of file confdoc.py.

21.98.3.2 add_option()

```
def confdoc.override.add_option (
              self,
              section,
              option,
              ivalue,
              brief,
              detail,
              basename = None,
              replace = True )
```

Adds documentation for an option in a conf section.

Parameters

section	The conf section name
option	The name of the option in that section
ivalue	A shortened form of the option value
brief	The brief documentation
detail	The detailed documentation
basename	The file basename
replace	If True, any existing documentation is replaced. Otherwise, it is ignored.

Definition at line 439 of file confdoc.py.

Referenced by confdoc.override.add_section().

21.98.3.3 add_section()

```
def confdoc.override.add_section (
              self,
              section,
              brief,
              detail,
              replace = True )
```

Adds documentation for a conf section.

Parameters

section	The conf section name
brief	The brief documentation
detail	The detailed documentation or None
replace	If True, any existing documentation is replaced. Otherwise, it is ignored.

Definition at line 427 of file confdoc.py.

21.98.3.4 file_block()

```
def confdoc.override.file_block (
              self,
              basename,
              brief,
              detail )
```

Adds a documentation block that is not associated with any section or option.

Parameters

basename	the file that contains the block
brief	the brief documentation
detail	the detailed documentation

Definition at line 418 of file confdoc.py.

21.98.3.5 find_optbrief()

```
def confdoc.override.find_optbrief (
              self,
              section,
              option )
```

Finds the brief documentation for a section.

Checks first this documentation object, and then the parent, searching for something that has documentation for the option.

Parameters

section	The conf section name
option	The name of the option in that section

Definition at line 455 of file confdoc.py.

Referenced by confdoc.override.print_sec_opt().

21.98.3.6 find_secbrief()

```
def confdoc.override.find_secbrief (
              self,
              section )
```

Finds the brief documentation for a section.

Checks first this documentation object, and then the parent, searching for something that has documentation for a section.

Parameters

section	The conf section name

Definition at line 470 of file confdoc.py.

Referenced by confdoc.override.print_sec_opt().

21.98.3.7 find_secdoc()

```
def confdoc.override.find_secdoc (
              self,
```

```
                section,
                detail )
```

Finds the brief and detailed documentation for a section.

Checks first this documentation object, and then the parent, searching for something that has documentation for a section.

Parameters

section	The conf section name
detail	detailed documentation, if available, and None otherwise

Returns

A tuple (brief,detail) containing any documentation found.

Definition at line 485 of file confdoc.py.

Referenced by confdoc.override._secsec().

21.98.3.8 optanch()

```
def confdoc.override.optanch (
                self,
                section,
                option,
                where = None )
```

Returns the anchor for a specified section and option.

Parameters

option	the option of interest
section	the section that contains the option
Optional	the conf file basename

Definition at line 386 of file confdoc.py.

Referenced by confdoc.override.find_optbrief().

21.98.3.9 print_sec_opt()

```
def confdoc.override.print_sec_opt (
                self,
                s )
```

Prints the section and option part of the documentation to the given stream.

Parameters

s	The stream, ideally a StringIO.StringIO.

Definition at line 513 of file confdoc.py.

Referenced by confdoc.override.print_subdoc(), and confdoc.coredoc.print_subdoc().

21.98.3.10 print_subdoc()

```
def confdoc.override.print_subdoc (
              self,
              s )
```

Prints the documentation to the specified stream.

Parameters

s	The stream, ideally a StringIO.StringIO.

Definition at line 507 of file confdoc.py.

21.98.3.11 secanch()

```
def confdoc.override.secanch (
              self,
              section,
              where = None )
```

Returns the anchor for the specified section.

Parameters

section	The conf section name
where	Optional: the conf file basename.

Definition at line 377 of file confdoc.py.

21.98.3.12 section_inc()

```
def confdoc.override.section_inc (
              self,
```

```
section,
inc )
```

Sets the @inc list for a conf section.

Parameters

section	The conf section name
inc	The contents of the @inc= option

Definition at line 410 of file confdoc.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/confdoc.py

21.99 produtil.prog.OverspecifiedStream Class Reference

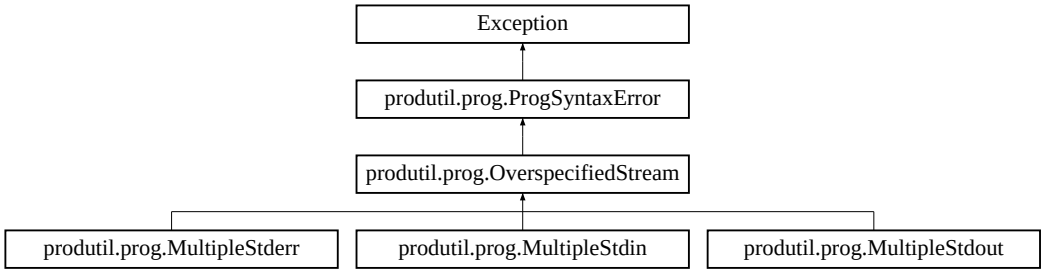
Raised when one tries to specify the stdout, stderr or stdin to go to, or come from, more than one location.

21.99.1 Detailed Description

Raised when one tries to specify the stdout, stderr or stdin to go to, or come from, more than one location.

Definition at line 49 of file prog.py.

Inheritance diagram for produtil.prog.OverspecifiedStream:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.100 confdoc.parsefile Class Reference

Config file parser.

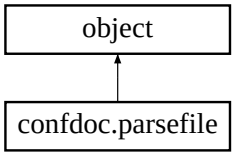
21.100.1 Detailed Description

Config file parser.

Parses comment blocks as described in the confdoc.

Definition at line 565 of file confdoc.py.

Inheritance diagram for confdoc.parsefile:



Public Member Functions

- def __init__ (self, filename, doc, maxread=500000)

  Opens the specified file, prepares to parse.
- def eot (self)

  Have we run out of lines to parse?
- def match (self, pattern)

  Calls re.match(pattern,...) on the current line, returning the result.
- def parse (self)

  Loops over all lines, parsing text and sending the result to global variables.
- def readoption (self, name)

  Reads later lines of a multi-line option=value assignment.
- def readblock (self)

  Reads the second and later lines of a multi-line comment block.

Public Attributes

- **doc**
- **lines**
- iline

  Current line number counting from 0.
- basename

  Basename of the current file.
- doxified

  The basename with "." replaced with "_".
- brief

  Brief portion of description that has not yet been assigned to an option or section.
- detail

  Detailed portion of description that has not yet been assigned to an option or section.
- section

  Section being parsed.
- **value**

21.100.2 Constructor & Destructor Documentation

21.100.2.1 __init__()

```
def confdoc.parsefile.__init__ (
              self,
              filename,
              doc,
              maxread = 500000 )
```

Opens the specified file, prepares to parse.

Only the first maxread bytes are read.

Parameters

filename	the ∗.conf file to read
doc	the docbase object to receive documentation
maxread	maximum number of lines to read

Definition at line 568 of file confdoc.py.

21.100.3 Member Function Documentation

21.100.3.1 match()

```
def confdoc.parsefile.match (
              self,
              pattern )
```

Calls re.match(pattern,...) on the current line, returning the result.

Parameters

pattern	argument to re.match: the pattern to match

Definition at line 607 of file confdoc.py.

Referenced by confdoc.parsefile.parse(), confdoc.parsefile.readblock(), and confdoc.parsefile.readoption().

21.100.3.2 parse()

```
def confdoc.parsefile.parse (
              self )
```

Loops over all lines, parsing text and sending the result to global variables.

Definition at line 615 of file confdoc.py.

21.100.3.3 readblock()

```
def confdoc.parsefile.readblock (
              self )
```

Reads the second and later lines of a multi-line comment block.

Assumes data is already in the self.brief variable.

Definition at line 735 of file confdoc.py.

Referenced by confdoc.parsefile.parse().

21.100.3.4 readoption()

```
def confdoc.parsefile.readoption (
              self,
              name )
```

Reads later lines of a multi-line option=value assignment.

Parameters

name	the option name

Definition at line 719 of file confdoc.py.

Referenced by confdoc.parsefile.parse().

21.100.4 Member Data Documentation

21.100.4.1 brief

`confdoc.parsefile.brief`

Brief portion of description that has not yet been assigned to an option or section.

Definition at line 580 of file confdoc.py.

Referenced by confdoc.parsefile.parse(), confdoc.parsefile.readblock(), and confdoc.parsefile.readoption().

21.100.4.2 detail

`confdoc.parsefile.detail`

Detailed portion of description that has not yet been assigned to an option or section.

Definition at line 581 of file confdoc.py.

Referenced by confdoc.parsefile.parse(), confdoc.parsefile.readblock(), and confdoc.parsefile.readoption().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/confdoc.py

21.101 produtil.atparse.ParserSyntaxError Class Reference

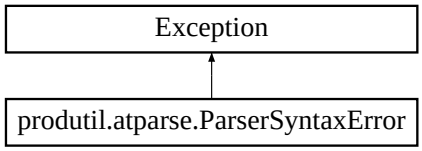
Raised when the parser encounters a syntax error.

21.101.1 Detailed Description

Raised when the parser encounters a syntax error.

Definition at line 14 of file atparse.py.

Inheritance diagram for produtil.atparse.ParserSyntaxError:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/atparse.py

21.102 produtil.numerics.partial_ordering Class Reference

Sorts a pre-determined list of objects, placing unknown items at a specified location.

21.102.1 Detailed Description

Sorts a pre-determined list of objects, placing unknown items at a specified location.

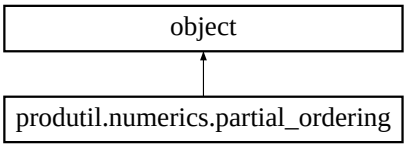
This class is a drop-in replacement for cmp in sorting routines. It represents a partial ordering of objects by specifying the order of a known subset of those objects, and inserting all unknown objects in a specified location in the that list (at the end, by default) in an order determined by cmp(a,b). Example:

```
p=partial_ordering([3,2,1]) # list is ordered as [3,2,1] with
                            #  everything else after that

sorted([0,1,2,3,6,4,5],p)  # = [3, 2, 1, 0, 4, 5, 6]
p(1,-99)  # = -1, so -99 goes after 1 since -99 is not
          #   in the partial ordering
p(1,3)    # = 1, so 3 goes before 1 since 3 is before 1
          #   in the partial ordering
p(5,10)   # = -1 since cmp(5,10)=-1
```

Definition at line 54 of file numerics.py.

Inheritance diagram for produtil.numerics.partial_ordering:



Public Member Functions

- def __init__ (self, ordering, unordered=None, backupcmp=cmp)

  partial_ordering constructor.
- def __call__ (self, a, b)

  Determine the ordering of a and b.

Public Attributes

- order

  Internal ordering information.
- backupcmp

  Backup comparison function for tiebreaking.
- unordered

  Unordered element index.

21.102.2 Constructor & Destructor Documentation

21.102.2.1 __init__()

```
def produtil.numerics.partial_ordering.__init__ (
               self,
               ordering,
               unordered = None,
               backupcmp = cmp )
```

partial_ordering constructor.

Creates a partial ordering. The subset that is ordered is specified by the ordered iterable "ordered" while the index at which to place unordered values is optionally specified by "unordered", which can be anything that can be cmp()'ed to an int. If "unordered" is missing, then all objects not in "ordered" will be placed at the end of any list. To place at the beginning of the list, give unordered=0. To insert between the first and second elements, specify 1, between second and third elements: specify unordered=2, and so on. Specify another tiebreaker "cmp" function with "backupcmp" (default: cmp).

Parameters

ordering	the ordering of known objects
unordered	Optional: where to put other objects
backupcmp	Tiebreaker comparison function.

Definition at line 75 of file numerics.py.

21.102.3 Member Function Documentation

21.102.3.1 __call__()

```
def produtil.numerics.partial_ordering.__call__ (
               self,
               a,
               b )
```

Determine the ordering of a and b.

Parameters

a,b	the objects to order

Returns

-1 if b>a, 1 if b<a, 0 if b and a belong in the same index.

Definition at line 119 of file numerics.py.

The documentation for this class was generated from the following file:

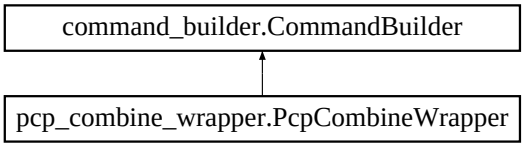
- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.103 pcp_combine_wrapper.PcpCombineWrapper Class Reference

21.103.1 Detailed Description

Definition at line 35 of file pcp_combine_wrapper.py.

Inheritance diagram for pcp_combine_wrapper.PcpCombineWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def **clear** (self)
- def **add_input_file** (self, filename, addon)
- def **getLastFile** (self, valid_time, search_time, template)
- def **get_lowest_forecast_at_valid** (self, valid_time, dtype)
- def **search_day** (self, dir, file_time, search_time, template)
- def **find_closest_before** (self, dir, time, template)
- def **get_accumulation** (self, valid_time, accum, ob_type, is_forecast=False)
- def **get_command** (self)
- def **run_at_time** (self, init_time)
- def **run_at_time_once** (self, valid_time, accum, ob_type, fcst_var, is_forecast=False)

Public Attributes

- **app_path**
- **app_name**
- **inaddons**
- **input_dir**
- **outfile**
- **outdir**

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/pcp_combine_wrapper.py

21.104 produtil.pipeline.Pipeline Class Reference

This class is a wrapper around launch and manage.

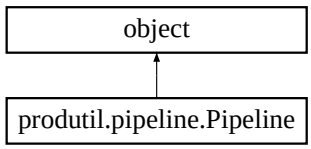
21.104.1 Detailed Description

This class is a wrapper around launch and manage.

It converts Runner objects to calls to "launch", and runs "manage" on the resulting processes.

Definition at line 564 of file pipeline.py.

Inheritance diagram for produtil.pipeline.Pipeline:



Public Member Functions

- def __init__ (self, runner, capture=False, logger=None, debug=False)

  Pipeline constructor.
- def __repr__ (self)

  Return a debug string representation of this Pipeline.
- def send_signal (self, sig)

  Sends a signal to all children.
- def terminate (self)

  Sends SIGTERM to all children.
- def kill (self)

  Sends SIGKILL to all children.
- def communicate (self, sleeptime=None)

  Writes to input, reads from output, waits for child processes, etc.
- def poll (self)

  Returns the exit status of the last element of the pipeline.
- def to_string (self)

  Calls self.communicate(), and returns the stdout from the pipeline (self.outstring).
- def outstring (self)

  The stdout from the pipeline.

21.104.2 Constructor & Destructor Documentation

21.104.2.1 __init__()

```
def produtil.pipeline.Pipeline.__init__ (
              self,
              runner,
              capture = False,
              logger = None,
              debug = False )
```

Pipeline constructor.

Parameters

runner	the produtil.prog.Runner to convert
capture	if True, capture the stdout of the runner
logger	a logging.Logger for messages
debug	if True, send debug messages

Definition at line 568 of file pipeline.py.

21.104.3 Member Function Documentation

21.104.3.1 __repr__()

```
def produtil.pipeline.Pipeline.__repr__ (
              self )
```

Return a debug string representation of this Pipeline.

Definition at line 589 of file pipeline.py.

Referenced by produtil.prog.Runner.__str__().

21.104.3.2 communicate()

```
def produtil.pipeline.Pipeline.communicate (
              self,
              sleeptime = None )
```

Writes to input, reads from output, waits for child processes, etc.

This is just a wrapper around the manage() function. It will return immediately if self.communicate has already completed earlier.

Parameters

sleeptime	the sleep time in seconds between checks

Definition at line 653 of file pipeline.py.

Referenced by produtil.pipeline.Pipeline.to_string().

21.104.3.3 kill()

```
def produtil.pipeline.Pipeline.kill (
              self )
```

Sends SIGKILL to all children.

Definition at line 644 of file pipeline.py.

21.104.3.4 outstring()

```
def produtil.pipeline.Pipeline.outstring (
              self )
```

The stdout from the pipeline.

Will be Null if the pipeline was redirected to a file, or if the constructor's capture option was not True.

Definition at line 693 of file pipeline.py.

Referenced by produtil.pipeline.Pipeline.to_string().

21.104.3.5 poll()

```
def produtil.pipeline.Pipeline.poll (
              self )
```

Returns the exit status of the last element of the pipeline.

If the process died due to a signal, returns a negative number.

Definition at line 670 of file pipeline.py.

21.104.3.6 send_signal()

```
def produtil.pipeline.Pipeline.send_signal (
              self,
              sig )
```

Sends a signal to all children.

Parameters

sig	the signal

Definition at line 633 of file pipeline.py.

Referenced by produtil.pipeline.Pipeline.kill(), and produtil.pipeline.Pipeline.terminate().

21.104.3.7 terminate()

```
def produtil.pipeline.Pipeline.terminate (
              self )
```

Sends SIGTERM to all children.

Definition at line 641 of file pipeline.py.

21.104.3.8 to_string()

```
def produtil.pipeline.Pipeline.to_string (
              self )
```

Calls self.communicate(), and returns the stdout from the pipeline (self.outstring).

The return value will be Null if the pipeline was redirected to a file or if the constructor's capture option was not True.

Definition at line 684 of file pipeline.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/pipeline.py

21.105 produtil.config.ProdConfig Class Reference

a class that contains configuration information

21.105.1 Detailed Description

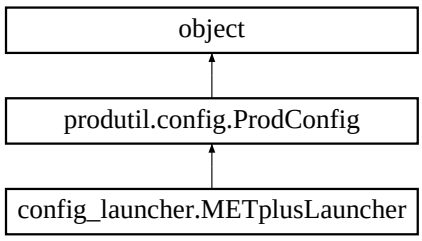
a class that contains configuration information

This class keeps track of configuration information for all tasks in a running model. It can be used in a read-only manner as if it was a ConfigParser object. All ProdTask objects require an ProdConfig object to keep track of registered task names via the register_task_name method, the current forecast cycle (cycle property) and the Datastore object (datastore property).

This class should never be instantiated directly. Instead, you should use the produtil.config.from_string or produtil.config.from_file to read configuration information from an in-memory string or a file.

Definition at line 499 of file config.py.

Inheritance diagram for produtil.config.ProdConfig:



Public Member Functions

- def __init__ (self, conf=None, quoted_literals=False)

  ProdConfig constructor.
- def **quoted_literals** (self)
- def fallback (self, name, details)

  Asks whether the specified fallback is allowed.
- def add_fallback_callback (self, function)

  Appends a function to the list of fallback callback functions called by fallback()
- def readstr (self, source)

  read config data and add it to this object
- def from_args (self, args=None, allow_files=True, allow_options=True, rel_path=None, verbose=False)

  Given a list of arguments, usually from sys.argv[1:], reads configuration files or sets option values.
- def read (self, source)

  reads and parses a config file
- def readfp (self, source)

  read config data from an open file
- def readstr (self, string)

  reads config data from an in-memory string
- def set_options (self, section, kwargs)

  set values of several options in a section
- def realtime (self)

  is this a real-time simulation?
- def set (self, section, key, value)

  set a config option
- def __enter__ (self)

  grab the thread lock
- def __exit__ (self, a, b, c)

  release the thread lock
- def register_task (self, name)

  add a ConfigurableTask to the database
- def log (self, sublog=None)

  returns a logging.Logger object
- def getdatastore (self)

  returns the Datastore
- def getcycle (self)

  get the analysis time
- def setcycle (self, cycle)

  set the analysis time
- def set_time_vars (self)

  internal function that sets time-related variables
- def add_section (self, sec)

  add a new config section
- def has_section (self, sec)

  does this section exist?
- def has_option (self, sec, opt)

  is this option set?
- def getdir (self, name, default=None, morevars=None, taskvars=None)

  query the "dir" section
- def getloc (self, name, default=None, morevars=None, taskvars=None)

  search the config, exe and dir sections in that order

- def getexe (self, name, default=None, morevars=None, taskvars=None)

  query the "exe" section
- def __getitem__ (self, arg)

  convenience function; replaces self.items and self.get
- def makedirs (self, args)

  calls produtil.fileop.makedirs() on directories in the [dir] section
- def keys (self, sec)

  get options in a section
- def sections (self)

  gets the list of all sections from a configuration object
- def items (self, sec, morevars=None, taskvars=None)

  get the list of (option,value) tuples for a section
- def write (self, fileobject)

  write the contents of this ProdConfig to a file
- def getraw (self, sec, opt, default=None)

  return the raw value of an option
- def strinterp (self, sec, string, kwargs)

  perform string expansion
- def timestrinterp (self, sec, string, ftime=None, atime=None, kwargs)

  performs string expansion, including time variables
- def getint (self, sec, opt, default=None, badtypeok=False, morevars=None, taskvars=None)

  get an integer value
- def getfloat (self, sec, opt, default=None, badtypeok=False, morevars=None, taskvars=None)

  get a float value
- def getstr (self, sec, opt, default=None, badtypeok=False, morevars=None, taskvars=None)

  get a string value
- def get (self, sec, opt, default=None, badtypeok=False, morevars=None, taskvars=None)

  get the value of an option from a section
- def options (self, sec)

  what options are in this section?
- def getboolean (self, sec, opt, default=None, badtypeok=False, morevars=None, taskvars=None)

  alias for getbool: get a bool value
- def getbool (self, sec, opt, default=None, badtypeok=False, morevars=None, taskvars=None)

  get a bool value

Properties

- datastore

  read-only property: the Datastore object for this simulation
- cycle

  the analysis cycle, a datetime.datetime object

21.105.2 Constructor & Destructor Documentation

21.105.2.1 __init__()

```
def produtil.config.ProdConfig.__init__ (
              self,
              conf = None,
              quoted_literals = False )
```

ProdConfig constructor.

Creates a new ProdConfig object.

Parameters

conf	the underlying ConfigParser.SafeConfigParser object that stores the actual config data
quoted_literals	if True, then {'...'} and {"..."} will be interpreted as quoting the contained ... text. Otherwise, those blocks will be considered errors.

Definition at line 513 of file config.py.

21.105.3 Member Function Documentation

21.105.3.1 __enter__()

```
def produtil.config.ProdConfig.__enter__ (
              self )
```

grab the thread lock

Grabs this ProdConfig's thread lock. This is only for future compatibility and is never used.

Definition at line 756 of file config.py.

21.105.3.2 __exit__()

```
def produtil.config.ProdConfig.__exit__ (
              self,
              a,
              b,
              c )
```

release the thread lock

Releases this ProdConfig's thread lock. This is only for future compatibility and is never used.

Parameters

a,b,c	unused

Definition at line 762 of file config.py.

21.105.3.3 __getitem__()

```
def produtil.config.ProdConfig.__getitem__ (
              self,
              arg )
```

convenience function; replaces self.items and self.get

This is a convenience function that provides access to the self.items or self.get functions.

- conf["section"] – returns a dict containing the results of self.items(arg)
- conf[a,b,c] – returns self.get(a,b,c) (b and c are optional)

Parameters

arg	the arguments: a list or string

Definition at line 952 of file config.py.

21.105.3.4 add_fallback_callback()

```
def produtil.config.ProdConfig.add_fallback_callback (
              self,
              function )
```

Appends a function to the list of fallback callback functions called by fallback()

Appends the given function to the list that fallback() searches while determining if a workflow emergency fallback option is allowed.

Parameters

function	a function f(allow,name,details)

See also

fallbacks()

Definition at line 574 of file config.py.

21.105.3.5 add_section()

```
def produtil.config.ProdConfig.add_section (
              self,
              sec )
```

add a new config section

Adds a section to this ProdConfig. If the section did not already exist, it will be initialized empty. Otherwise, this function has no effect.

Parameters

sec	the new section's name

Definition at line 874 of file config.py.

Referenced by produtil.config.ProdConfig.__init__().

21.105.3.6 fallback()

```
def produtil.config.ProdConfig.fallback (
              self,
              name,
              details )
```

Asks whether the specified fallback is allowed.

May perform other tasks, such as alerting the operator.

Calls the list of functions sent to add_fallback_callback. Each one receives the result of the last, and the final result at the end is returned. Note that ALL of the callbacks are called, even if one returns False; this is not a short-circuit operation. This is done to allow all reporting methods report to their operator and decide whether the fallback is allowed.

Each function called is f(allow,name,details) where:

- allow = True or False, whether the callbacks called thus far have allowed the fallback.
- name = The short name of the fallback.
- details = A long, human-readable description. May be several lines long.

Parameters

name	the name of the emergency situation

Warning

This function may take seconds or minutes to return. It could perform cpu- or time-intensive operations such as emailing an operator.

Definition at line 541 of file config.py.

21.105.3.7 from_args()

```
def produtil.config.ProdConfig.from_args (
              self,
              args = None,
              allow_files = True,
              allow_options = True,
              rel_path = None,
              verbose = False )
```

Given a list of arguments, usually from sys.argv[1:], reads configuration files or sets option values.

Reads list of strings of these formats:

- /path/to/file.conf — A configuration file to read.
- section.option=value — A configuration option to set in a specified section.

Will read files in the order listed, and then will override options in the order listed. Note that specified options override those read from files. Also, later files override earlier files.

Parameters

args	Typically argv[1:] or some other list of arguments.
allow_files	If True, filenames are allowed in args. Otherwise, they are ignored.
allow_options	If True, specified options (section.name=value) are allowed. Otherwise they are detected and ignored.
rel_path	Any filenames that are relative will be relative to this path. If None or unspecified, the current working directory as of the entry to this function is used.

Returns

self

Definition at line 599 of file config.py.

Referenced by produtil.config.ProdConfig.readstr().

21.105.3.8 get()

```
def produtil.config.ProdConfig.get (
              self,
              sec,
              opt,
              default = None,
              badtypeok = False,
              morevars = None,
              taskvars = None )
```

get the value of an option from a section

Gets option opt from section sec, expands it and converts to a string. If the option is not found and default is specified, returns default. If badtypeok, returns default if the option is found, but cannot be converted. The morevars is used during string expansion: if {abc} is in the value of the given option, and morevars contains a key abc, then {abc} will be expanded using that value. The morevars is a dict that allows the caller to override the list of variables for string extrapolation.

Parameters

sec,opt	the section and option
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars,taskvars	dicts of more variables for string expansion

Definition at line 1246 of file config.py.

Referenced by produtil.config.ProdConfig.__getitem__(), produtil.datastore.UpstreamFile.check(), produtil.config.↩ ProdConfig.getcycle(), produtil.config.ProdConfig.getraw(), and produtil.config.ProdConfig.timestrinterp().

21.105.3.9 getbool()

```
def produtil.config.ProdConfig.getbool (
              self,
              sec,
              opt,
              default = None,
              badtypeok = False,
              morevars = None,
              taskvars = None )
```

get a bool value

Gets option opt from section sec and expands it; see "get" for details. Attempts to convert it to a bool

Parameters

sec,opt	the section and option
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars,taskvars	dicts of more variables for string expansion

Definition at line 1296 of file config.py.

Referenced by produtil.config.ProdConfig.fallback(), produtil.config.ProdConfig.getboolean(), and produtil.config.↩ ProdConfig.realtime().

21.105.3.10 getboolean()

```
def produtil.config.ProdConfig.getboolean (
              self,
              sec,
              opt,
              default = None,
              badtypeok = False,
              morevars = None,
              taskvars = None )
```

alias for getbool: get a bool value

This is an alias for getbool for code expecting a ConfigParser. Gets option opt from section sec and expands it; see "get" for details. Attempts to convert it to a bool

Parameters

sec,opt	the section and option
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars,taskvars	dicts of more variables for string expansion

Definition at line 1280 of file config.py.

21.105.3.11 getcycle()

```
def produtil.config.ProdConfig.getcycle (
              self )
```

get the analysis time

Returns the analysis time of this workflow as a datetime.datetime.

Definition at line 816 of file config.py.

21.105.3.12 getdatastore()

```
def produtil.config.ProdConfig.getdatastore (
              self )
```

returns the Datastore

Returns the produtil.datastore.Datastore object for this ProdConfig.

Definition at line 794 of file config.py.

21.105.3.13 getdir()

```
def produtil.config.ProdConfig.getdir (
              self,
              name,
              default = None,
              morevars = None,
              taskvars = None )
```

query the "dir" section

Search the "dir" section.

Returns

the specified key (name) from the "dir" section. Other options are passed to self.getstr.

Parameters

default	the default value if the option is unset
morevars	more variables for string substitution
taskvars	even more variables for string substitution
name	the option name to search for

Definition at line 903 of file config.py.

Referenced by produtil.config.ProdTask.__init__(), produtil.config.ProdTask.get_outdir(), and produtil.config.Prod↩Task.get_workdir().

21.105.3.14 getexe()

```
def produtil.config.ProdConfig.getexe (
              self,
              name,
              default = None,
```

```
                morevars = None,
                taskvars = None )
```

query the "exe" section

Search the "exe" section.

Returns

the specified key (name) from the "exe" section. Other options are passed to self.getstr.

Parameters

default	the default value if the option is unset
morevars	more variables for string substitution
taskvars	even more variables for string substitution
name	the option name to search for

Definition at line 938 of file config.py.

21.105.3.15 getfloat()

```
def produtil.config.ProdConfig.getfloat (
                self,
                sec,
                opt,
                default = None,
                badtypeok = False,
                morevars = None,
                taskvars = None )
```

get a float value

Gets option opt from section sec and expands it; see "get" for details. Attempts to convert it to a float

Parameters

sec,opt	the section and option
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars,taskvars	dicts of more variables for string expansion

Definition at line 1216 of file config.py.

21.105.3.16 getint()

```
def produtil.config.ProdConfig.getint (
               self,
               sec,
               opt,
               default = None,
               badtypeok = False,
               morevars = None,
               taskvars = None )
```

get an integer value

Gets option opt from section sec and expands it; see "get" for details. Attempts to convert it to an int.

Parameters

sec,opt	the section and option
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars,taskvars	dicts of more variables for string expansion

Definition at line 1201 of file config.py.

21.105.3.17 getloc()

```
def produtil.config.ProdConfig.getloc (
               self,
               name,
               default = None,
               morevars = None,
               taskvars = None )
```

search the config, exe and dir sections in that order

Find the location of a file in the named option. Searches the [config], [exe] and [dir] sections in order for an option by that name, returning the first one found.

Parameters

default	the default value if the option is unset
morevars	more variables for string substitution
taskvars	even more variables for string substitution
name	the option name to search for

Returns

the resulting value

Definition at line 917 of file config.py.

21.105.3.18 getraw()

```
def produtil.config.ProdConfig.getraw (
              self,
              sec,
              opt,
              default = None )
```

return the raw value of an option

Returns the raw value for the specified section and option, without string interpolation. That is, any {...} will be returned unmodified. Raises an exception if no value is set. Will not search other sections, unlike other accessors.

Parameters

sec	the section
opt	the option name
default	the value to return if the option is unset. If unspecified or None, NoOptionError is raised

Definition at line 1027 of file config.py.

21.105.3.19 getstr()

```
def produtil.config.ProdConfig.getstr (
              self,
              sec,
              opt,
              default = None,
              badtypeok = False,
              morevars = None,
              taskvars = None )
```

get a string value

Gets option opt from section sec and expands it; see "get" for details. Attempts to convert it to a str

Parameters

sec,opt	the section and option
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars,taskvars	dicts of more variables for string expansion

Definition at line 1231 of file config.py.

Referenced by produtil.config.ProdConfig.getdatastore(), produtil.config.ProdConfig.getdir(), produtil.config.Prod↩
Config.getexe(), produtil.config.ProdConfig.getloc(), and produtil.config.ProdConfig.makedirs().

21.105.3.20 has_option()

```
def produtil.config.ProdConfig.has_option (
              self,
              sec,
              opt )
```

is this option set?

Determines if an option is set in the specified section

Returns

True if this ProdConfig has the given option in the specified section, and False otherwise.

Parameters

sec	the section
opt	the name of the option in that section

Definition at line 893 of file config.py.

Referenced by produtil.config.ProdConfig.getloc().

21.105.3.21 has_section()

```
def produtil.config.ProdConfig.has_section (
              self,
              sec )
```

does this section exist?

Determines if a config section exists (even if it is empty)

Returns

True if this ProdConfig has the given section and False otherwise.

Parameters

sec	the section to check for

Definition at line 884 of file config.py.

21.105.3.22 items()

```
def produtil.config.ProdConfig.items (
              self,
              sec,
              morevars = None,
              taskvars = None )
```

get the list of (option,value) tuples for a section

Returns a section's options as a list of two-element tuples. Each tuple contains a config option, and the value of the config option after string interpolation. Note that the special config section inclusion option "@inc" is also returned.

Parameters

sec	the section
morevars	variables for string substitution
taskvars	yet more variables

Returns

a list of (option,value) tuples, where the value is after string expansion

Definition at line 997 of file config.py.

Referenced by produtil.config.ProdConfig.__getitem__().

21.105.3.23 keys()

```
def produtil.config.ProdConfig.keys (
              self,
              sec )
```

get options in a section

Returns a list containing the config options in the given section.

Parameters

sec	the string name of the section

Definition at line 984 of file config.py.

21.105.3.24 log()

```
def produtil.config.ProdConfig.log (
              self,
              sublog = None )
```

returns a logging.Logger object

Returns a logging.Logger object. If the sublog argument is provided, then the logger will be under that subdomain of the "produtil" logging domain. Otherwise, this ProdConfig's logger is returned.

Parameters

sublog	the logging subdomain, or None

Returns

a logging.Logger object

Definition at line 781 of file config.py.

Referenced by produtil.config.ProdConfig.from_args(), produtil.config.ProdConfig.getdatastore(), and config_launcher.METplusLauncher.sanity_check().

21.105.3.25 makedirs()

```
def produtil.config.ProdConfig.makedirs (
              self,
              args )
```

calls produtil.fileop.makedirs() on directories in the [dir] section

This is a simple utility function that calls produtil.fileop.makedirs() on some of the directories in the [dir] section.

Parameters

args	the keys in the [dir] section for the directories to make.

Definition at line 972 of file config.py.

21.105.3.26 options()

```
def produtil.config.ProdConfig.options (
              self,
              sec )
```

what options are in this section?

Returns a list of options in the given section

Parameters

sec	the section

Definition at line 1272 of file config.py.

Referenced by produtil.config.ProdConfig.items(), and produtil.config.ProdConfig.keys().

21.105.3.27 read()

```
def produtil.config.ProdConfig.read (
              self,
              source )
```

reads and parses a config file

Opens the specified config file and reads it, adding its contents to the configuration. This is used to implement the from_file module-scope function. You can use it again on an ProdConfig object to read additional files.

Parameters

source	the file to read

Returns

self

Definition at line 694 of file config.py.

Referenced by produtil.config.ProdConfig.from_args().

21.105.3.28 readfp()

```
def produtil.config.ProdConfig.readfp (
              self,
              source )
```

read config data from an open file

Reads a config file from the specified file-like object. This is used to implement the readstr.

Parameters

source	the opened file to read

Returns

self

Definition at line 706 of file config.py.

Referenced by produtil.config.ProdConfig.readstr().

21.105.3.29 readstr() [1/2]

```
def produtil.config.ProdConfig.readstr (
              self,
              source )
```

read config data and add it to this object

Given a string with conf data in it, parses the data.

Parameters

source	the data to parse

Returns

self

Definition at line 586 of file config.py.

Referenced by produtil.config.ProdConfig.readstr().

21.105.3.30 readstr() [2/2]

```
def produtil.config.ProdConfig.readstr (
              self,
              string )
```

reads config data from an in-memory string

Reads the given string as a config file. This is used to implement the from_string module-scope function. You can use it again to read more config data into an existing ProdConfig.

Parameters

string	the string to parse

Returns

self

Definition at line 716 of file config.py.

21.105.3.31 realtime()

```
def produtil.config.ProdConfig.realtime (
              self )
```

is this a real-time simulation?

Is this configuration for a real-time simulation? Defaults to True if unknown. This is the same as doing get-bool('config','realtime',True).

Definition at line 742 of file config.py.

21.105.3.32 register_task()

```
def produtil.config.ProdConfig.register_task (
              self,
              name )
```

add a ConfigurableTask to the database

Checks to ensure that there is no other task by this name, and records the fact that there is now a task. This is used by the ConfigurableTask to ensure only one task is made by any name.

Definition at line 769 of file config.py.

21.105.3.33 set()

```
def produtil.config.ProdConfig.set (
              self,
              section,
              key,
              value )
```

set a config option

Sets the specified config option (key) in the specified section, to the specified value. All three are converted to strings via str() before setting the value.

Definition at line 749 of file config.py.

Referenced by produtil.config.ProdConfig.__init__(), produtil.config.ProdConfig.from_args(), produtil.config.↩ ProdConfig.set_options(), produtil.config.ProdConfig.set_time_vars(), produtil.config.ProdConfig.setcycle(), and produtil.config.ProdConfig.timestrinterp().

21.105.3.34 set_options()

```
def produtil.config.ProdConfig.set_options (
              self,
              section,
              kwargs )
```

set values of several options in a section

Sets the value of several options in one section. The keywords arguments are the names of the options to set and the keyword values are the option values.

Parameters

section	the section being modified
kwargs	additional keyword arguments are the option names and values

Definition at line 728 of file config.py.

21.105.3.35 set_time_vars()

```
def produtil.config.ProdConfig.set_time_vars (
              self )
```

internal function that sets time-related variables

Sets many config options in the [config] section based on this workflow's analysis time. This is called automatically when the cycle property is assigned. You never need to call this function directly.

YMDHM - 201409171200 = forecast time September 17, 2014 at 12:00 UTC YMDH - 2014091712 YMD - 20140917 year - 2014 YYYY - 2014 YY - 14 (year % 100) CC - 20 (century) cen - 20 month - 09 MM - 09 day - 17 DD - 17 hour - 12 cyc - 12 HH - 12 minute - 00 min - 00

Definition at line 841 of file config.py.

Referenced by produtil.config.ProdConfig.setcycle().

21.105.3.36 setcycle()

```
def produtil.config.ProdConfig.setcycle (
              self,
              cycle )
```

set the analysis time

Sets the analysis time of this workflow. Also sets the [config] section's "cycle" option. Accepts anything that produtil.numerics.to_datetime recognizes.

Definition at line 824 of file config.py.

21.105.3.37 strinterp()

```
def produtil.config.ProdConfig.strinterp (
               self,
               sec,
               string,
               kwargs )
```

perform string expansion

Performs this ProdConfig's string interpolation on the specified string, as if it was a value from the specified section.

Parameters

sec	the section name
string	the string to expand
kwargs	more variables for string substitution

Definition at line 1044 of file config.py.

21.105.3.38 timestrinterp()

```
def produtil.config.ProdConfig.timestrinterp (
               self,
               sec,
               string,
               ftime = None,
               atime = None,
               kwargs )
```

performs string expansion, including time variables

Performs this ProdConfig's string interpolation on the specified string, as self.strinterp would, but adds in additional keys based on the given analysis and forecast times. The keys are the same as the keys added to [config] for the cycle, except with "a" prepended for the analysis time, or "f" for the forecast time. There are three more keys for the difference between the forecast an analysis time. The famin is the forecast time in minutes, rounded down. The fahr and fahrmin are the forecast hour, rounded down, and the remainder in minutes, rounded down to the next nearest minute.

If the analysis time is None or unspecified, then self.cycle is used. The atime can be anything understood by produtil.numerics.to_datetime and the ftime can be anything understood by produtil.numerics.to_datetime_rel, given the atime (or, absent atime, self.cycle) as the second argument.

This is implemented as a wrapper around the self._time_formatter object, which knows how to expand the a∗ and f∗ variables without having to generate all of them.

Parameters

sec	the section name
string	the string to expand
ftime	the forecast time or None
atime	the analysis time or None
kwargs	more variables for string expansion

Definition at line 1059 of file config.py.

21.105.3.39 write()

```
def produtil.config.ProdConfig.write (
               self,
               fileobject )
```

write the contents of this ProdConfig to a file

Writes the contents of an ProdConfig to the specified file, without interpolating (expanding) any strings. The file will be suitable for reading in to a new ProdConfig object in a later job. This is used to create the initial config file.

Parameters

fileobject	an opened file to write to

Definition at line 1016 of file config.py.

21.105.4 Property Documentation

21.105.4.1 cycle

```
produtil.config.ProdConfig.cycle  [static]
```

Initial value:

```
= property(getcycle,setcycle,None,
        )
```

the analysis cycle, a datetime.datetime object

Definition at line 838 of file config.py.

Referenced by produtil.config.ProdConfig.timestrinterp().

21.105.4.2 datastore

```
produtil.config.ProdConfig.datastore  [static]
```

Initial value:

```
= property(getdatastore,None,None, \
        )
```

read-only property: the Datastore object for this simulation

Definition at line 811 of file config.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/config.py

21.106 produtil.config.ProdTask Class Reference

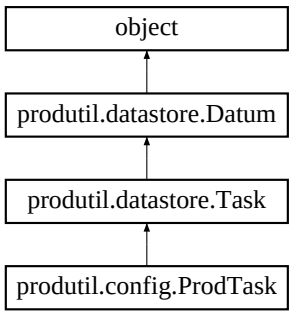
A subclass of produtil.datastore.Task that provides a variety of convenience functions related to unix conf files and logging.

21.106.1 Detailed Description

A subclass of produtil.datastore.Task that provides a variety of convenience functions related to unix conf files and logging.

Definition at line 1327 of file config.py.

Inheritance diagram for produtil.config.ProdTask:



Public Member Functions

- def __init__ (self, dstore, conf, section, taskname=None, workdir=None, outdir=None, taskvars=UNSPECI↩ FIED, kwargs)

  Creates an ProdTask.
- def get_workdir (self)

  Returns the directory the class should work in, as set by the "workdir" metadata value.
- def set_workdir (self, val)

  Sets the directory the class should work in.
- def get_outdir (self)

  Gets the directory that should receive output data.
- def set_outdir (self, val)

  Sets the directory that should receive output data.
- def tvset (self, opt, val)

  Sets a taskvar option's value.
- def tvdel (self, opt)

  Deletes an object-local value set by tvset.
- def tvget (self, opt)

  Gets a taskvar's value.
- def tvhave (self, opt=UNSPECIFIED)

  Is a taskvar set?
- def taskvars (self)

  The dict of object-local values used for string substitution.
- def confint (self, opt, default=None, badtypeok=False, section=None, morevars=None)

  Alias for self.conf.getint for section self.section.

- def confstr (self, opt, default=None, badtypeok=False, section=None, morevars=None)

  Alias for self.conf.getstr for section self.section.
- def confffloat (self, opt, default=None, badtypeok=False, section=None, morevars=None)

  Alias for self.conf.getfloat for section self.section.
- def confbool (self, opt, default=None, badtypeok=False, section=None, morevars=None)

  Alias for self.conf.getbool for section self.section.
- def confget (self, opt, default=None, badtypeok=False, section=None, morevars=None)

  Alias for self.conf.get for section self.section.
- def confitems (self, section=None, morevars=None)

  Alias for self.conf.items for section self.section.
- def confstrinterp (self, string, section=None, kwargs)

  Alias for self.icstr for backward compatibility.
- def conftimestrinterp (self, string, ftime, atime=None, section=None, kwargs)

  Alias for self.timestr for backward comaptibility.
- def confraw (self, opt, default=None, section=None)

  Get a raw configuration value before string expansion.
- def icstr (self, string, section=None, kwargs)

  Expands a string in the given conf section.
- def timestr (self, string, ftime=None, atime=None, section=None, kwargs)

  Expands a string in the given conf section, including time vars.
- def getdir (self, opt, default=None, morevars=None)

  Alias for produtil.config.ProdConfig.get() for the "dir" section.
- def getexe (self, opt, default=None, morevars=None)

  Alias for produtil.config.ProdConfig.get() for the "exe" section.
- def getconf (self)

  Returns this ProdTask's produtil.config.ProdConfig object.
- def getsection (self)

  Returns this ProdTask's section name in the ProdConfig.
- def log (self, subdom=None)

  Obtain a logging domain.

Properties

- workdir

  The directory in which this task should be run.
- outdir

  The directory in which this task should deliver its final output.
- conf

  This ProdTask's produtil.config.ProdConfig object.
- section

  The confsection in self.section for this ProdTask (read-only)

21.106.2 Constructor & Destructor Documentation

21.106.2.1 __init__()

```
def produtil.config.ProdTask.__init__ (
                self,
                dstore,
                conf,
                section,
                taskname = None,
                workdir = None,
                outdir = None,
                taskvars = UNSPECIFIED,
                kwargs )
```

Creates an ProdTask.

Parameters

dstore	passed to Datum: the Datastore object for this Task
conf	the conf object for this task
section	the conf section for this task
taskname	Optional: the taskname in the datastore. Default: the section name
workdir	directory in which this task should run. Any value set in the database will override this value.
outdir	directory where output should be copied. This argument must not be changed throughout the lifetime of the datstore database file.
taskvars	additonal variables for string expansion, sent to the taskvars arguments of produtil.config.ProdConfig member functions.
kwargs	passed to the parent class constructor.

Definition at line 1334 of file config.py.

21.106.3 Member Function Documentation

21.106.3.1 confbool()

```
def produtil.config.ProdTask.confbool (
                self,
                opt,
                default = None,
                badtypeok = False,
                section = None,
                morevars = None )
```

Alias for self.conf.getbool for section self.section.

Parameters

opt	the option name
section	Optional: the section. Default: self.section
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars	dict of more variables for string expansion

Definition at line 1512 of file config.py.

Referenced by produtil.config.ProdTask.conffloat().

21.106.3.2 conffloat()

```
def produtil.config.ProdTask.conffloat (
              self,
              opt,
              default = None,
              badtypeok = False,
              section = None,
              morevars = None )
```

Alias for self.conf.getfloat for section self.section.

Parameters

opt	the option name
section	Optional: the section. Default: self.section
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars	dict of more variables for string expansion

Definition at line 1499 of file config.py.

Referenced by produtil.config.ProdTask.confstr().

21.106.3.3 confget()

```
def produtil.config.ProdTask.confget (
              self,
              opt,
              default = None,
              badtypeok = False,
              section = None,
              morevars = None )
```

Alias for self.conf.get for section self.section.

Parameters

opt	the option name
section	Optional: the section. Default: self.section
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars	dict of more variables for string expansion

Definition at line 1525 of file config.py.

Referenced by produtil.config.ProdTask.confbool().

21.106.3.4 confint()

```
def produtil.config.ProdTask.confint (
              self,
              opt,
              default = None,
              badtypeok = False,
              section = None,
              morevars = None )
```

Alias for self.conf.getint for section self.section.

Parameters

opt	the option name
section	Optional: the section. Default: self.section.
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars	dict of more variables for string expansion

Definition at line 1473 of file config.py.

Referenced by produtil.config.ProdTask.taskvars().

21.106.3.5 confitems()

```
def produtil.config.ProdTask.confitems (
              self,
              section = None,
              morevars = None )
```

Alias for self.conf.items for section self.section.

Parameters

section	Optional: the section. Default: self.section.
morevars	variables for string substitution

Definition at line 1537 of file config.py.

21.106.3.6 confraw()

```
def produtil.config.ProdTask.confraw (
              self,
              opt,
              default = None,
              section = None )
```

Get a raw configuration value before string expansion.

Returns the raw, uninterpolated value for the specified option, raising an exception if that option is unset. Will not search other sections, and will not search the taskvars, unlike other conf accessors.

Parameters

opt	the option of interest
section	Optional: the section. Default: self.section
default	Optional: value to return if nothing is found.

Definition at line 1563 of file config.py.

21.106.3.7 confstr()

```
def produtil.config.ProdTask.confstr (
              self,
              opt,
              default = None,
              badtypeok = False,
              section = None,
              morevars = None )
```

Alias for self.conf.getstr for section self.section.

Parameters

opt	the option name
section	Optional: the section. Default: self.section
default	if specified and not None, then the default is returned if an option has no value or the section does not exist
badtypeok	is True, and the conversion fails, and a default is specified, the default will be returned.
morevars	dict of more variables for string expansion

Definition at line 1486 of file config.py.

Referenced by produtil.config.ProdTask.__init__(), and produtil.config.ProdTask.confint().

21.106.3.8 confstrinterp()

```
def produtil.config.ProdTask.confstrinterp (
              self,
              string,
              section = None,
              kwargs )
```

Alias for self.icstr for backward compatibility.

Parameters

string	the string to expand
section	Optional: the section in which to expand it. Default: self.section.
kwargs	more arguments for string substitution

Definition at line 1544 of file config.py.

21.106.3.9 conftimestrinterp()

```
def produtil.config.ProdTask.conftimestrinterp (
              self,
              string,
              ftime,
              atime = None,
              section = None,
              kwargs )
```

Alias for self.timestr for backward comaptibility.

Parameters

string	the string to expand
ftime	the forecast time
atime	Optional: the analysis time. Default: self.conf.cycle
section	Optional: the section in which to expand it. Default: self.section.
kwargs	more arguments for string substitution

Definition at line 1553 of file config.py.

Referenced by produtil.config.ProdTask.confstrinterp().

21.106.3.10 get_outdir()

```
def produtil.config.ProdTask.get_outdir (
              self )
```

Gets the directory that should receive output data.

This is in the "outdir" metadata value.

Definition at line 1404 of file config.py.

21.106.3.11 get_workdir()

```
def produtil.config.ProdTask.get_workdir (
              self )
```

Returns the directory the class should work in, as set by the "workdir" metadata value.

Definition at line 1386 of file config.py.

21.106.3.12 getconf()

```
def produtil.config.ProdTask.getconf (
              self )
```

Returns this ProdTask's produtil.config.ProdConfig object.

Definition at line 1635 of file config.py.

21.106.3.13 getdir()

```
def produtil.config.ProdTask.getdir (
              self,
              opt,
              default = None,
              morevars = None )
```

Alias for produtil.config.ProdConfig.get() for the "dir" section.

Parameters

opt	the option name
default	Optional: default value if nothing is found.
morevars	Optional: more variables for string substitution

Definition at line 1621 of file config.py.

Referenced by produtil.config.ProdTask.__init__(), produtil.config.ProdTask.get_outdir(), and produtil.config.Prod↩Task.get_workdir().

21.106.3.14 getexe()

```
def produtil.config.ProdTask.getexe (
              self,
              opt,
              default = None,
              morevars = None )
```

Alias for produtil.config.ProdConfig.get() for the "exe" section.

Parameters

opt	the option name
default	Optional: default value if nothing is found.
morevars	Optional: more variables for string substitution

Definition at line 1628 of file config.py.

21.106.3.15 getsection()

```
def produtil.config.ProdTask.getsection (
              self )
```

Returns this ProdTask's section name in the ProdConfig.

Definition at line 1643 of file config.py.

21.106.3.16 icstr()

```
def produtil.config.ProdTask.icstr (
              self,
              string,
              section = None,
              kwargs )
```

Expands a string in the given conf section.

Given a string, expand it as if it was a value in the specified conf section. Makes this objects tcvitals, if any, available via the "vit" variable while interpolating strings.

Parameters

string	the string to expand
section	Optional: the section in which to expand it. Default: self.section.
kwargs	more arguments for string substitution

Definition at line 1576 of file config.py.

Referenced by produtil.config.ProdTask.confstrinterp().

21.106.3.17 log()

```
def produtil.config.ProdTask.log (
              self,
              subdom = None )
```

Obtain a logging domain.

Creates or returns a logging.Logger. If subdom is None or unspecified, returns a cached logger for this task's logging domain. Otherwise, returns a logger for the specified subdomain of this task's logging domain.

Parameters

subdom	Optional: the desired logging domain

Definition at line 1651 of file config.py.

21.106.3.18 set_outdir()

```
def produtil.config.ProdTask.set_outdir (
              self,
              val )
```

Sets the directory that should receive output data.

Sets the "outdir" metadata value.

Parameters

val	the new output directory

Definition at line 1412 of file config.py.

21.106.3.19 set_workdir()

```
def produtil.config.ProdTask.set_workdir (
              self,
              val )
```

Sets the directory the class should work in.

This sets the "workdir" metadata value.

Parameters

val	the new work directory

Definition at line 1394 of file config.py.

21.106.3.20 taskvars()

```
def produtil.config.ProdTask.taskvars (
              self )
```

The dict of object-local values used for string substitution.

Definition at line 1467 of file config.py.

21.106.3.21 timestr()

```
def produtil.config.ProdTask.timestr (
              self,
              string,
              ftime = None,
              atime = None,
              section = None,
              kwargs )
```

Expands a string in the given conf section, including time vars.

Expands a string in the given conf section (default: self.section), and includes forecast and analysis time (default: conf.cycle) information in the variables that can be expanded. The mandatory ftime argument is the forecast time which will be used to expand values such as fHH, fYMDH, etc. The optional atime will be used to expand aHH, aYMDH, etc., and the two will be used together for forecast minus analysis fields like fahr. See produtil.config.↩ timestrinterp for details

As with self.icstr, this class's vitals are available via the "vit" variable while interpolating strings.

Parameters

string	the string to expand
ftime	the forecast time
atime	Optional: the analysis time. Default: self.conf.cycle
section	Optional: the section in which to expand it. Default: self.section.
kwargs	more arguments for string substitution

Definition at line 1592 of file config.py.

Referenced by produtil.config.ProdTask.conftimestrinterp().

21.106.3.22 tvdel()

```
def produtil.config.ProdTask.tvdel (
              self,
              opt )
```

Deletes an object-local value set by tvset.

Parameters

opt	the name of the taskvar to delete

Definition at line 1439 of file config.py.

21.106.3.23 tvget()

```
def produtil.config.ProdTask.tvget (
              self,
              opt )
```

Gets a taskvar's value.

Returns the value of an object-local (taskvar) option set by tvset.

Parameters

opt	the taskvar whose value should be returned

Definition at line 1445 of file config.py.

21.106.3.24 tvhave()

```
def produtil.config.ProdTask.tvhave (
              self,
              opt = UNSPECIFIED )
```

Is a taskvar set?

If an option is specified, determines if the given option has an object-local (taskvar) value. If no option is specified, returns True if ANY object-local values (taskvars) exist for any options.

Parameters

opt	Optional: the name of the taskvar being checked.

Definition at line 1453 of file config.py.

21.106.3.25 tvset()

```
def produtil.config.ProdTask.tvset (
              self,
              opt,
              val )
```

Sets a taskvar option's value.

Sets an object-local (taskvar) value for option "opt" to value "val". This will override config settings from the ProdConfig object. These are sent into the taskvars= parameter to the various ProdConfig member functions (hence the "tv" in "tvset").

Parameters

opt	the name of the taskvar
val	the string value of the option

Definition at line 1424 of file config.py.

Referenced by produtil.config.ProdTask.__init__().

21.106.4 Property Documentation

21.106.4.1 conf

```
produtil.config.ProdTask.conf  [static]
```

Initial value:

```
= property(getconf,None,None,
                 )
```

This ProdTask's produtil.config.ProdConfig object.

Definition at line 1641 of file config.py.

21.106.4.2 outdir

```
produtil.config.ProdTask.outdir  [static]
```

Initial value:

```
= property(get_outdir,set_outdir,None,
        )
```

The directory in which this task should deliver its final output.

Note that changing this will NOT update products already in the database.

Definition at line 1421 of file config.py.

21.106.4.3 section

```
produtil.config.ProdTask.section  [static]
```

Initial value:

```
= property(getsection,None,None,
        )
```

The confsection in self.section for this ProdTask (read-only)

Definition at line 1648 of file config.py.

Referenced by produtil.config.ProdTask.confraw().

21.106.4.4 workdir

```
produtil.config.ProdTask.workdir  [static]
```

Initial value:

```
= property(get_workdir,set_workdir,None,
        )
```

The directory in which this task should be run.

Definition at line 1401 of file config.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/config.py

21.107 produtil.datastore.Product Class Reference

A piece of data produced by a Task.

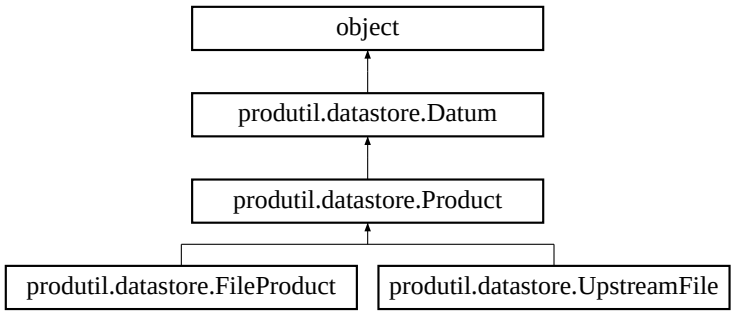
21.107.1 Detailed Description

A piece of data produced by a Task.

A Product is a piece of data that can be produced by a Task. Once the product is available, self.available or self.↩ is_available() will be True, and the self.location will be valid. The meaning of self.location is up to the implementer to decide, but it should be a full path to a location on disk for file products. As with all Datum objects, a Product also has arbitrary metadata.

Definition at line 716 of file datastore.py.

Inheritance diagram for produtil.datastore.Product:



Public Member Functions

- def add_callback (self, callback, args=None, states=None)

  Adds a delivery callback function.
- def has_callbacks (self)

  Returns True if this Product has any callback functions and False otherwise.
- def call_callbacks (self, logger=None)

  Calls all delivery callback functions.
- def check (self, kwargs)

  Asks the product to check its own availability and update the database.
- def deliver (self, kwargs)

  Asks the Product to deliver itself.
- def undeliver (self, kwargs)

  "Undelivers" a product.
- def setavailable (self, val)

  Sets the availability to the specified value.
- def is_available (self)

  Is the product available?
- def validate (self)

  Validates this object's Datastore, prodname and category.

Properties

- available

  Read-write property: is the product available?

21.107.2 Member Function Documentation

21.107.2.1 add_callback()

```
def produtil.datastore.Product.add_callback (
              self,
              callback,
              args = None,
              states = None )
```

Adds a delivery callback function.

Adds a delivery callback that is called when the product is delivered. This is intended to do such tasks as running an NCO dbn_alert, or copying to a website, or emailing someone. This function is only added in this local Python instance, not in the database file. Also, it is the responsibility of the subclasses to call self.call_callbacks() from self.deliver() to ensure the callbacks are run.

Example:

```
def callback(name,*args,**kwargs):
  print "My fancy product %s was delivered!"%(name,)
product.add_callback(callback,[product.prodname])
```

Parameters

callback	The callback function, which must be able to take any keyword or indexed arguments.
args	The indexed arguments to send.
states	Presently unused.

Definition at line 725 of file datastore.py.

21.107.2.2 call_callbacks()

```
def produtil.datastore.Product.call_callbacks (
              self,
              logger = None )
```

Calls all delivery callback functions.

Calls all data delivery callbacks for this Product. Collects any raised Exception subclasses until after all callbacks are called. Will raise CallbackExceptions if any exceptions are caught.

Subclasses should call this from either check, or deliver, as appropriate for the product type.

Parameters

logger	Optional: the logging.Logger for logging messages.

Definition at line 759 of file datastore.py.

Referenced by produtil.datastore.UpstreamFile.check(), and produtil.datastore.FileProduct.deliver().

21.107.2.3 check()

```
def produtil.datastore.Product.check (
              self,
              kwargs )
```

Asks the product to check its own availability and update the database.

Checks to see if this product is available. This is generally not a cheap operation, as it can take seconds or minutes and may fail. One should call "available" instead if cached information is sufficient.

Parameters

kwargs	Additional keyword arguments are unused. This is for use by subclasses.

Definition at line 786 of file datastore.py.

Referenced by produtil.fileop.FileWaiter.checkfiles().

21.107.2.4 deliver()

```
def produtil.datastore.Product.deliver (
              self,
              kwargs )
```

Asks the Product to deliver itself.

Delivers a product to its destination. This is not implemented by the base class. Note that this is generally an expensive operation which may take seconds or minutes, and may fail. It may involve copying many files, network access, or even pulling tapes from a silo. In the end, the location and availability are expected to be updated in the database.

Parameters

kwargs	Unused, to be used by subclasses.

Postcondition

available=True and location is non-empty.

Definition at line 798 of file datastore.py.

21.107.2.5 is_available()

```
def produtil.datastore.Product.is_available (
              self )
```

Is the product available?

Returns the "available" attribute of this Product in the database, converted to a boolean value via bool()

Definition at line 831 of file datastore.py.

21.107.2.6 setavailable()

```
def produtil.datastore.Product.setavailable (
              self,
              val )
```

Sets the availability to the specified value.

Sets the "available" attribute of this Product in the database after converting the given value to a bool and then int (int(bool(val))).

Parameters

val	the new availability

Definition at line 823 of file datastore.py.

21.107.2.7 undeliver()

```
def produtil.datastore.Product.undeliver (
              self,
              kwargs )
```

"Undelivers" a product.

The meaning of this function is implementation-dependent: it could mean deleting an output file, or any number of other actions. Regardless, it should result in self.available=False or an exception being thrown. Note that this is generally an expensive operation that could take seconds or minutes, and may fail. The default implementation simply sets available to False.

Postcondition

available=False

Definition at line 810 of file datastore.py.

21.107.2.8 validate()

```
def produtil.datastore.Product.validate (
              self )
```

Validates this object's Datastore, prodname and category.

Validates the Datastore, prodname and category of this Product. In addition to the requirements made by Datum, this function requires that the category not contain any double stars ("**").

Definition at line 843 of file datastore.py.

Referenced by produtil.mpiprog.MPIRank.__init__().

21.107.3 Property Documentation

21.107.3.1 available

```
produtil.datastore.Product.available  [static]
```

Initial value:

```
= property(is_available,setavailable,None,
                       )
```

Read-write property: is the product available?

Definition at line 840 of file datastore.py.

Referenced by produtil.datastore.Product.check(), produtil.datastore.UpstreamFile.check(), produtil.datastore.FileProduct.deliver(), produtil.datastore.Product.undeliver(), produtil.datastore.FileProduct.undeliver(), and produtil.datastore.UpstreamFile.undeliver().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.108 produtil.prog.ProgSyntaxError Class Reference

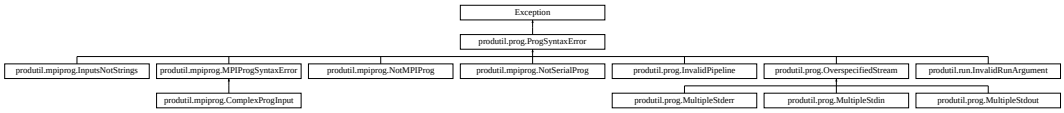
Base class of exceptions raised when a Runner is given arguments that make no sense.

21.108.1 Detailed Description

Base class of exceptions raised when a Runner is given arguments that make no sense.

Definition at line 46 of file prog.py.

Inheritance diagram for produtil.prog.ProgSyntaxError:



The documentation for this class was generated from the following file:

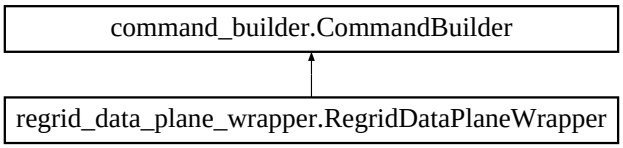
- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.109 regrid_data_plane_wrapper.RegridDataPlaneWrapper Class Reference

21.109.1 Detailed Description

Definition at line 29 of file regrid_data_plane_wrapper.py.

Inheritance diagram for regrid_data_plane_wrapper.RegridDataPlaneWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def **run_at_time** (self, init_time)
- def **run_at_time_once** (self, valid_time, accum, ob_type)

Public Attributes

- **app_path**
- **app_name**

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/regrid_data_plane_wrapper.py

21.110 produtil.fileop.RelativePathError Class Reference

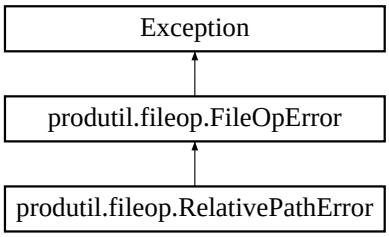
Raised when a relative path is given, but an absolute path is expected.

21.110.1 Detailed Description

Raised when a relative path is given, but an absolute path is expected.

Definition at line 77 of file fileop.py.

Inheritance diagram for produtil.fileop.RelativePathError:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.111 produtil.rstprod.RestrictionClass Class Reference

This is a python class intended to be used to automate restricting data to a specific restriction class using access control lists or group ownership.

21.111.1 Detailed Description

This is a python class intended to be used to automate restricting data to a specific restriction class using access control lists or group ownership.

Example:

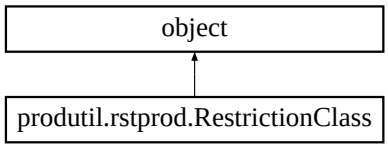
```
rc=RestrictionClass("rstprod")
rc.restrict_file("/path/to/some/dangerous/file")
```

It can also set the Default Access Control List if supplied a directory:

```
rc.restrict_file("/path/to/some/dangerous/directory/")
```

Definition at line 61 of file rstprod.py.

Inheritance diagram for produtil.rstprod.RestrictionClass:



Public Member Functions

- def __init__ (self, group, use_acl=None, logger=None)

  Create a new RestrictionClass object for the specified group.
- def groupname (self)

  The name of the group used for the restriction class.
- def groupid (self)

  The numeric ID of the group used for the restriction class.
- def use_acl (self)

  True if ACLs are used for access permission, False if setgid and chgrp are used.
- def acl_for (self, st_mode)

  Returns an produtil.acl.ACL object for the specified access mode.
- def restrict_file (self, filename, st_mode=None, logger=None)

  Adds the requested restrictions to the specified file or directory.
- def restrict_fd (self, fd, st_mode=None, logger=None)

Protected Member Functions

- def make_acl_dict (self)

  Internal function that generates the ACL dictionary.
- def chgrp_restrict (self, target, st_mode, chown, chmod, logger)

  Internal function that uses chgrp to restrict a file's access.
- def acl_restrict_file (self, target, st_mode, set_acl, logger)

  Internal function that restricts files using ACLs.

21.111.2 Constructor & Destructor Documentation

21.111.2.1 __init__()

```
def produtil.rstprod.RestrictionClass.__init__ (
              self,
              group,
              use_acl = None,
              logger = None )
```

Create a new RestrictionClass object for the specified group.

Parameters

group	The group may be the string group name, or the numeric group id.
use_acl	If use_acl is unspecified, then produtil.cluster.use_acl_for_rstdata() is used to decide.
logger	a logging.Logger for log messages

Definition at line 76 of file rstprod.py.

21.111.3 Member Function Documentation

21.111.3.1 acl_for()

```
def produtil.rstprod.RestrictionClass.acl_for (
              self,
              st_mode )
```

Returns an produtil.acl.ACL object for the specified access mode.

Will raise an exception if self.use_acl is False.

Parameters

st_mode	desired access mode

Definition at line 171 of file rstprod.py.

Referenced by produtil.rstprod.RestrictionClass.restrict_fd(), and produtil.rstprod.RestrictionClass.restrict_file().

21.111.3.2 acl_restrict_file()

```
def produtil.rstprod.RestrictionClass.acl_restrict_file (
              self,
              target,
              st_mode,
              set_acl,
              logger )  [protected]
```

Internal function that restricts files using ACLs.

This is an internal implementation function that should not be called directly. It handles the ACL case of restrict_file.

Parameters

target	the target file
st_mode	the desired access
set_acl	the acl-setting function
logger	a logging.Logger for log messages

Definition at line 206 of file rstprod.py.

Referenced by produtil.rstprod.RestrictionClass.restrict_file().

21.111.3.3 chgrp_restrict()

```
def produtil.rstprod.RestrictionClass.chgrp_restrict (
              self,
              target,
              st_mode,
              chown,
              chmod,
              logger )  [protected]
```

Internal function that uses chgrp to restrict a file's access.

This is an internal implementation function that should not be called directly. It handles the non-ACL (chgrp+setgid) case of restrict_file and restrict_gid.

Parameters

target	the target file
st_mode	the desired mode
chown	chowning function
chmod	chmodding function
logger	a logging.Logger for log messages

Definition at line 179 of file rstprod.py.

Referenced by produtil.rstprod.RestrictionClass.restrict_fd(), and produtil.rstprod.RestrictionClass.restrict_file().

21.111.3.4 make_acl_dict()

```
def produtil.rstprod.RestrictionClass.make_acl_dict (
              self )  [protected]
```

Internal function that generates the ACL dictionary.

This is part of the internal implementation of RestrictionClass and should not be used directly. It returns a dict() that maps from integer permission to an ACL object that will set an access control list appropriate for that permission. The user and restriction group will match the old user and group permissions, but other groups will have no permissions, and the "world" permissions will be 0.

Definition at line 132 of file rstprod.py.

Referenced by produtil.rstprod.RestrictionClass.__init__().

21.111.3.5 restrict_fd()

```
def produtil.rstprod.RestrictionClass.restrict_fd (
              self,
              fd,
              st_mode = None,
              logger = None )
```

```
Adds the requested restrictions to an opened file.  This
routine needs to stat the opened file to get the stat.st_mode.
@param st_mode To avoid a stat call, send st_mode into the optional argument.
@param fd the target file descriptor
@param logger a logging.Logger for log messages
```

Definition at line 247 of file rstprod.py.

21.111.3.6 restrict_file()

```
def produtil.rstprod.RestrictionClass.restrict_file (
              self,
              filename,
              st_mode = None,
              logger = None )
```

Adds the requested restrictions to the specified file or directory.

This routine needs to stat the opened file to get the stat.st_mode.

Parameters

st_mode	To avoid a stat call, send st_mode into the optional argument.
filename	the target file
logger	a logging.Logger for log messages

Definition at line 228 of file rstprod.py.

21.111.3.7 use_acl()

```
def produtil.rstprod.RestrictionClass.use_acl (
              self )
```

True if ACLs are used for access permission, False if setgid and chgrp are used.

Definition at line 166 of file rstprod.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/rstprod.py

21.112 produtil.rusage.RLimit Class Reference

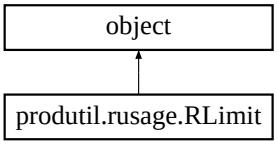
Gets the resource limits set on this process: core, cpu, fsize, data, stack, rss, nproc, nofile, memlock, aspace Each is set to a tuple containing the soft and hard limit.

21.112.1 Detailed Description

Gets the resource limits set on this process: core, cpu, fsize, data, stack, rss, nproc, nofile, memlock, aspace Each is set to a tuple containing the soft and hard limit.

Definition at line 98 of file rusage.py.

Inheritance diagram for produtil.rusage.RLimit:



Public Member Functions

- def __init__ (self, logger=None)
  RLimit constructor.
- def __str__ (self)
  Creates a multi-line string representation of the resource limits.

21.112.2 Constructor & Destructor Documentation

21.112.2.1 __init__()

```
def produtil.rusage.RLimit.__init__ (
              self,
              logger = None )
```

RLimit constructor.

Parameters

logger	a logging.Logger for log messages.

Definition at line 102 of file rusage.py.

21.112.3 Member Function Documentation

21.112.3.1 __str__()

```
def produtil.rusage.RLimit.__str__ (
              self )
```

Creates a multi-line string representation of the resource limits.

Definition at line 117 of file rusage.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/rusage.py

21.113 produtil.rstprod.RstBadGroup Class Reference

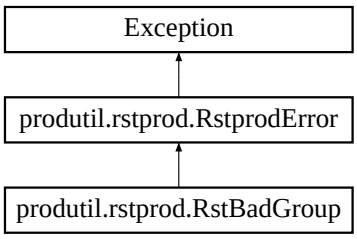
Raised when a group's id or name could not be determined.

21.113.1 Detailed Description

Raised when a group's id or name could not be determined.

Definition at line 21 of file rstprod.py.

Inheritance diagram for produtil.rstprod.RstBadGroup:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/rstprod.py

21.114 produtil.rstprod.RstNoAccessControl Class Reference

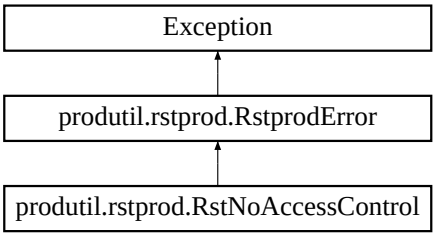
Raised when the cluster has no access control mechanisms.

21.114.1 Detailed Description

Raised when the cluster has no access control mechanisms.

Definition at line 19 of file rstprod.py.

Inheritance diagram for produtil.rstprod.RstNoAccessControl:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/rstprod.py

21.115 produtil.rstprod.RstprodError Class Reference

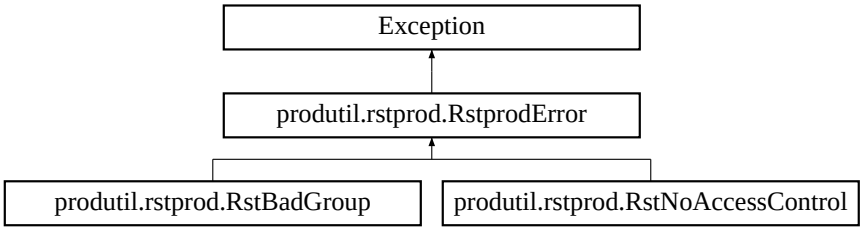
The base class of all exceptions specific to the rstprod module.

21.115.1 Detailed Description

The base class of all exceptions specific to the rstprod module.

Definition at line 17 of file rstprod.py.

Inheritance diagram for produtil.rstprod.RstprodError:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/rstprod.py

21.116 produtil.prog.Runner Class Reference

Represents a single stage of a pipeline to execute.

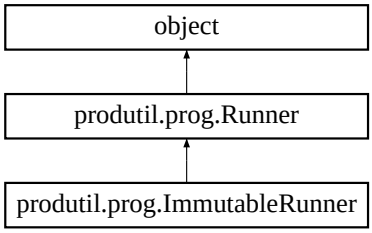
21.116.1 Detailed Description

Represents a single stage of a pipeline to execute.

This is a linked list class used to store information about a program or pipeline of programs to be run. It has the capability of converting itself to a Pipeline object (run(Runner)), or converting itself to a POSIX sh command (Runner.to_shell()). Note that some commands cannot be represented in POSIX sh, such as commands with non-ASCII characters or commands that have Python streams as their stdout or stdin. Those commands can still be run with a Pipeline, but trying to convert them to a POSIX sh command will throw NotValidPosixSh or a subclass thereof.

Definition at line 299 of file prog.py.

Inheritance diagram for produtil.prog.Runner:



Public Member Functions

- def __init__ (self, args, kwargs)

  Creates a new Runner.
- def getthreads (self)

  Returns the number of threads requested by this program.
- def setthreads (self, nthreads)

  Sets the number of threads requested by this program.
- def delthreads (self)

  Removes the request for threads.
- def first (self)

  Returns the first Runner in this pipeline.
- def remove_prerun (self)

  Removes all prerun objects.
- def prerun (self, arg)

  Adds a function or callable object to be called before running the program.
- def __getitem__ (self, args)

  Add one or more arguments to the executable.
- def __str__ (self)

  *Alias for **repr**()*
- def __repr__ (self)

  Attempts to produce valid Python code to represent this Runnable.
- def __eq__ (self, other)

  Returns True if the other object is a Runner that is equal to this one, and False otherwise.
- def isplainexe (self)

  Returns true if this is simply an executable with arguments (no redirection, no prerun objects, no environment modification, no piping), and False otherwise.
- def cd (self, dirpath)

  Requests that this process run in the specified directory.

- def __lt__ (self, stdin)

  Connects the given object to stdin, via inp(stdin,string=False).
- def __gt__ (self, stdout)

  Connects the given object to stdout, truncating it if it is a file.
- def __lshift__ (self, stdin)

  Sends the specified string into stdin.
- def __rshift__ (self, stdout)

  Appends stdout to the specified file.
- def __pos__ (self)

  Sends stderr to stdout.
- def __ge__ (self, outerr)

  Redirects stderr and stdout to the specified file, truncating it.
- def __or__ (self, other)

  Pipes this Runner to the other Runner.
- def argins (self, index, arg)

  Inserts the specified argument before the given index.
- def args (self)

  Iterates over the executable and arguments of this command.
- def copy (self, typeobj=None)

  Returns a deep copy of this object, almost.
- def copyenv (self)

  Instructs this command to duplicate the parent process environment (the default).
- def clearenv (self)

  Instructs this command to start with an empty environment except for certain critical variables without which most programs cannot run.
- def **getenv** (self, arg)
- def env (self, kwargs)

  Sets environment variables for this Runner.
- def to_shell (self)

  Returns a string that expresses this object as a POSIX sh shell command if possible, or raises a subclass of NotValidPosixSh if not.
- def runner (self)

  Returns self if self is modifiable, otherwise returns a modifiable copy of self.
- def pipeto (self, other)

  Specifies that this Runner will send its stdout to the other runner's stdin.
- def inp (self, stdin, string=False)

  Specifies that the first Runner in this pipeline takes input from the given file or string specified by stdin.
- def out (self, stdout, append=False)

  Specifies that this process sends output from its stdout stream to the given file or stream.
- def err2out (self)

  Sends stderr to stdout.
- def err (self, stderr, append=False)

  Specifies that this process sends output from its stderr stream to the given file or stream.

Static Public Attributes

- **threads**

21.116.2 Constructor & Destructor Documentation

21.116.2.1 __init__()

```
def produtil.prog.Runner.__init__ (
              self,
              args,
              kwargs )
```

Creates a new Runner.

The only non-keyword argument can be one of three things:

1. A Runner to copy. Every aspect of the Runner that can be copied will be. Note that if a stream-like object is connected to stdin, stdout or stderr, it will NOT be copied.
2. A list of strings. This will be used as the command path, and arguments.

Many options can be set via keyword arguments:

- clearenv=True - the environment should be cleared before running this command. Any arguments set by the env= keyword or the .env(...) member function ignore this. Also, PATH, USER, LOGNAME and HOME are retained since most programs cannot run without them.
- env=dict(var=value,...) - a dict of environment variables to set before running the Runner. Does NOT affect this parent's process, only the child process.
- in=filename - a file to send to stdin.
- instr=str - a string to send to stdin
- out=filename - a file to connect to stdout. Will truncate the file.
- outa=filename - same as "out=filename," but appends to the file.
- err2out - redirects stderr to stdout
- err=filename - a file to connect to stderr. Will truncate the file.
- erra=filename - same as "err=filename," but appends to the file.
- prerun=[obj,anotherobj,...] - sent to self.prerun, this is a list of functions or callable objects to run before executing the process. The objects are not called until execution is requested via self._gen.

Parameters

args	the arguments to the program
kwargs	other settings (see constructor description).

Definition at line 311 of file prog.py.

21.116.3 Member Function Documentation

21.116.3.1 __eq__()

```
def produtil.prog.Runner.__eq__ (
              self,
              other )
```

Returns True if the other object is a Runner that is equal to this one, and False otherwise.

Parameters

other	the object to compare

Definition at line 517 of file prog.py.

21.116.3.2 __ge__()

```
def produtil.prog.Runner.__ge__ (
              self,
              outerr )
```

Redirects stderr and stdout to the specified file, truncating it.

Same as err2out().out(filename,append=False)

Parameters

outerr	the stdout and stderr file

Returns

self

Definition at line 574 of file prog.py.

21.116.3.3 __getitem__()

```
def produtil.prog.Runner.__getitem__ (
              self,
              args )
```

Add one or more arguments to the executable.

Can ONLY accept strings, ints, floats or iterables (tuple, list). Strings, ints and floats are sent to _stringify_args, and the result is added to the end of the list of arguments to the command to run. For iterables (tuple, list), adds all elements to the list of arguments, passing each through _stringify_args.

Parameters

args	one or more arguments to add

Returns

self

Definition at line 461 of file prog.py.

21.116.3.4 __gt__()

```
def produtil.prog.Runner.__gt__ (
              self,
              stdout )
```

Connects the given object to stdout, truncating it if it is a file.

Same as out(stdout,append=False).

Parameters

stdout	the stdout object

Returns

self

Definition at line 552 of file prog.py.

21.116.3.5 __lshift__()

```
def produtil.prog.Runner.__lshift__ (
              self,
              stdin )
```

Sends the specified string into stdin.

Same as inp(stdin,string=True).

Parameters

stdin	the stdin file

Returns

self

Definition at line 558 of file prog.py.

21.116.3.6 __lt__()

```
def produtil.prog.Runner.__lt__ (
              self,
              stdin )
```

Connects the given object to stdin, via inp(stdin,string=False).

Parameters

stdin	the stdin object

Returns

self

Definition at line 547 of file prog.py.

21.116.3.7 __or__()

```
def produtil.prog.Runner.__or__ (
              self,
              other )
```

Pipes this Runner to the other Runner.

Same as pipeto(other).

Returns

other

Parameters

other	the other runner to pipe into

Definition at line 580 of file prog.py.

21.116.3.8 __pos__()

```
def produtil.prog.Runner.__pos__ (
              self )
```

Sends stderr to stdout.

Same as err2out().

Returns

self

Definition at line 570 of file prog.py.

21.116.3.9 __repr__()

```
def produtil.prog.Runner.__repr__ (
              self )
```

Attempts to produce valid Python code to represent this Runnable.

Generally, that can be done, unless an input string is too long, or a stream is connected to a Python object. In those cases, human-readable representations are given, which are not exactly Python code.

Definition at line 481 of file prog.py.

Referenced by produtil.prog.Runner.__str__().

21.116.3.10 __rshift__()

```
def produtil.prog.Runner.__rshift__ (
              self,
              stdout )
```

Appends stdout to the specified file.

Same as out(stdout,append=True).

Parameters

stdout	the stdout file

Returns

self

Definition at line 564 of file prog.py.

21.116.3.11 argins()

```
def produtil.prog.Runner.argins (
              self,
              index,
              arg )
```

Inserts the specified argument before the given index.

This function is intended for internal use only. It is used to implement threading on Cray, where arguments relating to threading have to be added after the Runner is generated.

Warning

It is generally not safe to call this function outside the produtil.mpi_impl subpackage since its modules may generate completely different commands than you asked in order to execute your requested programs.

Parameters

arg	a string argument to add
index	the index to insert before

Note

Index 0 is the executable, while later indices are arguments.

Definition at line 586 of file prog.py.

21.116.3.12 cd()

```
def produtil.prog.Runner.cd (
              self,
              dirpath )
```

Requests that this process run in the specified directory.

The directory must already exist before the program starts.

Parameters

dirpath	the directory to cd into, which must already exist.

Returns

self

Definition at line 540 of file prog.py.

Referenced by produtil.prog.Runner.__init__().

21.116.3.13 clearenv()

```
def produtil.prog.Runner.clearenv (
              self )
```

Instructs this command to start with an empty environment except for certain critical variables without which most programs cannot run.

(Retains PATH, USER, LOGNAME and HOME.)

Returns

self

Definition at line 644 of file prog.py.

Referenced by produtil.prog.Runner.__init__().

21.116.3.14 copy()

```
def produtil.prog.Runner.copy (
              self,
              typeobj = None )
```

Returns a deep copy of this object, almost.

If stdin, stdout or stderr are connected to streams instead of files or strings, then the streams are not copied. Instead, the exact same stream objects are connected to the same unit in the new Runner.

Parameters

typeobj	the type of the new object or None for Runner. Do not set this unless you know what you're doing.

Returns

the new object

Definition at line 610 of file prog.py.

Referenced by produtil.prog.ImmutableRunner.runner().

21.116.3.15 copyenv()

```
def produtil.prog.Runner.copyenv (
              self )
```

Instructs this command to duplicate the parent process environment (the default).

Returns

self

Definition at line 637 of file prog.py.

21.116.3.16 delthreads()

```
def produtil.prog.Runner.delthreads (
              self )
```

Removes the request for threads.

Definition at line 406 of file prog.py.

21.116.3.17 env()

```
def produtil.prog.Runner.env (
              self,
              kwargs )
```

Sets environment variables for this Runner.

The variables should be specified as keyword arguments.

Parameters

kwargs	varname=value arguments

Returns

self

Definition at line 681 of file prog.py.

21.116.3.18 err()

```
def produtil.prog.Runner.err (
              self,
              stderr,
              append = False )
```

Specifies that this process sends output from its stderr stream to the given file or stream.

The stderr object must be a string filename, or a stream. If append=False, and the stderr is a filename, the file will be truncated, if append=True then it is appended. Raises MultipleStderr if the stderr location is already specified.

Parameters

stderr	the stderr output file
append	if True, append to the file otherwise truncate

Returns

self

Definition at line 824 of file prog.py.

Referenced by produtil.prog.Runner.__init__().

21.116.3.19 err2out()

```
def produtil.prog.Runner.err2out (
              self )
```

Sends stderr to stdout.

Returns

self

Definition at line 815 of file prog.py.

Referenced by produtil.prog.Runner.__ge__(), produtil.prog.Runner.__init__(), and produtil.prog.Runner.__pos__().

21.116.3.20 first()

```
def produtil.prog.Runner.first (
              self )
```

Returns the first Runner in this pipeline.

Definition at line 413 of file prog.py.

21.116.3.21 getthreads()

```
def produtil.prog.Runner.getthreads (
              self )
```

Returns the number of threads requested by this program.

Definition at line 399 of file prog.py.

21.116.3.22 inp()

```
def produtil.prog.Runner.inp (
              self,
              stdin,
              string = False )
```

Specifies that the first Runner in this pipeline takes input from the given file or string specified by stdin.

If string=True, then stdin is converted to a string via str(), otherwise it must be a filename or a stream. Raises MultipleStdin if the stdin source is already specified.

Parameters

stdin	the input file or string
string	if True, stdin is a string. Otherwise, it is a file.

Returns

self

Definition at line 766 of file prog.py.

Referenced by produtil.prog.Runner.__lshift__(), produtil.prog.Runner.__lt__(), and produtil.prog.Runner.pipeto().

21.116.3.23 isplainexe()

```
def produtil.prog.Runner.isplainexe (
              self )
```

Returns true if this is simply an executable with arguments (no redirection, no prerun objects, no environment modification, no piping), and False otherwise.

Definition at line 532 of file prog.py.

21.116.3.24 out()

```
def produtil.prog.Runner.out (
              self,
              stdout,
              append = False )
```

Specifies that this process sends output from its stdout stream to the given file or stream.

The stdout object must be a string filename, or a stream. If append=False, and the stdout is a filename, the file will be truncated, if append=True then it is appended. Raises MultipleStdout if the stdout location is already specified

Parameters

stdout	the stdout file
append	if True, append to the file, otherwise truncate

Returns

self

Definition at line 793 of file prog.py.

Referenced by produtil.prog.Runner.__ge__(), produtil.prog.Runner.__gt__(), and produtil.prog.Runner.__rshift_↩
_().

21.116.3.25 pipeto()

```
def produtil.prog.Runner.pipeto (
              self,
              other )
```

Specifies that this Runner will send its stdout to the other runner's stdin.

This will raise MultipleStdout if this Runner's stdout target is already specified, or MultipleStdin if the other's stdin is already specified.

Parameters

other	the runner to pipe into

Returns

other

Definition at line 734 of file prog.py.

Referenced by produtil.prog.Runner.__or__().

21.116.3.26 prerun()

```
def produtil.prog.Runner.prerun (
              self,
              arg )
```

Adds a function or callable object to be called before running the program.

The callables should be very fast operations, and are executed by self._gen when creating the Pipeline. They take, as an argument, the Runner and an optional "logger" keyword argument that is either None, or a logging.Logger to use to log messages.

Parameters

arg	a callable object that takes self as an argument, and an optional keyword argument "logger" with a logging.Logger for log messages

Definition at line 426 of file prog.py.

Referenced by produtil.prog.Runner.__init__(), and produtil.prog.Runner.err().

21.116.3.27 remove_prerun()

```
def produtil.prog.Runner.remove_prerun (
              self )
```

Removes all prerun objects.

See also

prerun()

Returns

self

Definition at line 419 of file prog.py.

21.116.3.28 runner()

```
def produtil.prog.Runner.runner (
              self )
```

Returns self if self is modifiable, otherwise returns a modifiable copy of self.

This is intended to be used to implement unmodifiable subclasses of Runner

Returns

self

Definition at line 728 of file prog.py.

Referenced by produtil.prog.ImmutableRunner.argins(), produtil.prog.ImmutableRunner.pipeto(), and produtil.↩ prog.ImmutableRunner.runner().

21.116.3.29 setthreads()

```
def produtil.prog.Runner.setthreads (
              self,
              nthreads )
```

Sets the number of threads requested by this program.

Definition at line 402 of file prog.py.

21.116.3.30 to_shell()

```
def produtil.prog.Runner.to_shell (
              self )
```

Returns a string that expresses this object as a POSIX sh shell command if possible, or raises a subclass of NotValidPosixSh if not.

Definition at line 692 of file prog.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.117 produtil.rusage.RUsage Class Reference

Contains resource usage (rusage) information that can be used with a Python "with" construct to collect the resources utilized by a block of code, or group of subprocesses executing during that block.

21.117.1 Detailed Description

Contains resource usage (rusage) information that can be used with a Python "with" construct to collect the resources utilized by a block of code, or group of subprocesses executing during that block.

Example:

```
with produtil.rusage.RUsage(logger=logging.getLogger("usage")):
    ... do things ...
... stop doing things ...
```

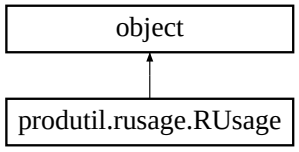
Just after the "with" block exits, the resource usage is printed to the given logger. The information can be retained for inspection instead:

```
u=produtil.rusage.RUsage(logger=logging.getLogger("usage"))
with u:
    ... do things ...
... stop doing things ...
# u.rusage_before is a dict of resource usage before the block
# u.time_before contains the time before the block
# u.rusage_after contains the resource usage at the end of the block
# u.time_after contains the time after the block
```

Note that the logger is optional: without it, nothing is logged.

Definition at line 175 of file rusage.py.

Inheritance diagram for produtil.rusage.RUsage:



Public Member Functions

- def __init__ (self, who=resource.RUSAGE_CHILDREN, logger=None)

  Creates an RUsage object for input to a "with" statement.
- def who (self)

  The "who" parameter to the constructor, which selects whether the usage measured should be of the child processes (RUSAGE_CHILDREN) or this process (RUSAGE_SELF) .
- def pagesize (self)

  System page size in bytes from resource.getpagesize().
- def __enter__ (self)

  Gets the resource usage and time at the top of the "with" block.
- def __exit__ (self, type, value, tb)

  Gets the resource usage and time at the end of a "with" block.
- def report (self)

  Generates a string report of the resource usage utilized.
- def __str__ (self)

  Generates a string report of the resource usage utilized.

Public Attributes

- logger

  The logging.Logger for log messages.
- rusage_before

  Resource usage before monitoring began.
- rusage_after

  The resource usage after monitoring ended.
- time_before

  The current time before usage monitoring began.
- time_after

  The current time after monitoring ended.

21.117.2 Constructor & Destructor Documentation

21.117.2.1 __init__()

```
def produtil.rusage.RUsage.__init__ (
              self,
              who = resource.RUSAGE_CHILDREN,
              logger = None )
```

Creates an RUsage object for input to a "with" statement.

Parameters

who	Pass who=resource.RUSAGE_SELF to get usage on this process or rusage.RUSAGE_CHILDREN (the default) to get resource usage on child processes.
logger	a logging.Logger for log messages

Definition at line 203 of file rusage.py.

21.117.3 Member Function Documentation

21.117.3.1 __enter__()

```
def produtil.rusage.RUsage.__enter__ (
              self )
```

Gets the resource usage and time at the top of the "with" block.

This function is called automatically by the Python interpreter at the beginning of a "with" block.

Definition at line 245 of file rusage.py.

21.117.3.2 __exit__()

```
def produtil.rusage.RUsage.__exit__ (
              self,
              type,
              value,
              tb )
```

Gets the resource usage and time at the end of a "with" block.

This is called automatically by Python at the end of a "with" block.

Parameters

type,value,tb	exception information

Definition at line 251 of file rusage.py.

21.117.3.3 __str__()

```
def produtil.rusage.RUsage.__str__ (
              self )
```

Generates a string report of the resource usage utilized.

Definition at line 278 of file rusage.py.

21.117.3.4 pagesize()

```
def produtil.rusage.RUsage.pagesize (
              self )
```

System page size in bytes from resource.getpagesize().

This is needed to interpret return values.

Definition at line 241 of file rusage.py.

21.117.3.5 report()

```
def produtil.rusage.RUsage.report (
              self )
```

Generates a string report of the resource usage utilized.

Accessible via str(self).

Definition at line 261 of file rusage.py.

Referenced by produtil.rusage.RUsage.__exit__(), and produtil.rusage.RUsage.__str__().

21.117.3.6 who()

```
def produtil.rusage.RUsage.who (
              self )
```

The "who" parameter to the constructor, which selects whether the usage measured should be of the child processes (RUSAGE_CHILDREN) or this process (RUSAGE_SELF) .

See **init** for details.

Definition at line 234 of file rusage.py.

21.117.4 Member Data Documentation

21.117.4.1 time_after

```
produtil.rusage.RUsage.time_after
```

The current time after monitoring ended.

Definition at line 214 of file rusage.py.

Referenced by produtil.rusage.RUsage.__exit__(), and produtil.rusage.RUsage.report().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/rusage.py

21.118 produtil.rusage.RUsageReport Class Reference

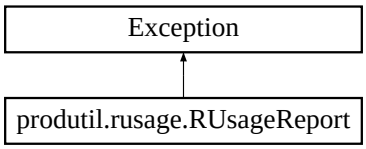
Raised when caller makes an RUsage, and tries to generate its report, before calling its **enter** or **exit** routines.

21.118.1 Detailed Description

Raised when caller makes an RUsage, and tries to generate its report, before calling its **enter** or **exit** routines.

Definition at line 171 of file rusage.py.

Inheritance diagram for produtil.rusage.RUsageReport:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/rusage.py

21.119 produtil.atparse.ScriptAbort Class Reference

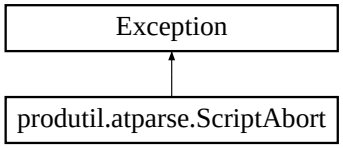
Raised when an "@∗∗ abort" directive is reached in a script.

21.119.1 Detailed Description

Raised when an "@∗∗ abort" directive is reached in a script.

Definition at line 19 of file atparse.py.

Inheritance diagram for produtil.atparse.ScriptAbort:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/atparse.py

21.120 produtil.atparse.ScriptAssertion Class Reference

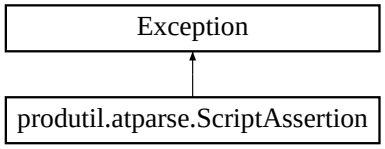
Raised when a script @[VARNAME:?message] is encountered, and the variable does not exist.

21.120.1 Detailed Description

Raised when a script @[VARNAME:?message] is encountered, and the variable does not exist.

Definition at line 16 of file atparse.py.

Inheritance diagram for produtil.atparse.ScriptAssertion:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/atparse.py

21.121 series_by_init_wrapper.SeriesByInitWrapper Class Reference

Performs series analysis based on init time by first performing any additional filtering via the wrapper to the MET tool tc_stat, tc_stat_wrapper.

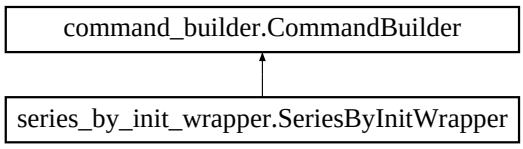
21.121.1 Detailed Description

Performs series analysis based on init time by first performing any additional filtering via the wrapper to the MET tool tc_stat, tc_stat_wrapper.

Next, the arguments to run the MET tool series_analysis is done

Definition at line 31 of file series_by_init_wrapper.py.

Inheritance diagram for series_by_init_wrapper.SeriesByInitWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def run_all_times (self)

  Invoke the series analysis script based on the init time in the format YYYYMMDD_hh.
- def is_netcdf_created (self)

  Check for the presence of NetCDF files in the series_analysis_init directory.
- def get_fcst_file_info (self, dir_to_search, cur_init, cur_storm)

  Get the number of all the gridded forecast n x m tile files for a given storm id and init time (that were created by extract_tiles).
- def get_ascii_storm_files_list (self, tile_dir)

  Creates the list of ASCII files that contain the storm id and init times.
- def build_and_run_series_request (self, sorted_filter_init, tile_dir)

  Build up the -obs, -fcst, -out necessary for running the series_analysis MET tool, then invoke series_analysis.
- def create_obs_fcst_arg (self, param_arg, ascii_file_base, cur_storm, cur_init)

  Create the argument to the -obs or -fcst flag to the MET tool, series_analysis.
- def create_out_arg (self, cur_storm, cur_init, name, level)

  Create/build the -out portion of the series_analysis command and creates the output directory.
- def **clear** (self)
- def **add_input_file** (self, filename, type_id)
- def **get_command** (self)
- def generate_plots (self, sorted_filter_init, tile_dir)

  Generate the plots from the series_analysis output.
- def get_storms_for_init (self, cur_init, out_dir_base)

  Retrieve all the filter files which have the .tcst extension.
- def create_fcst_anly_to_ascii_file (self, fcst_anly_grid_files, cur_init, cur_storm, fcst_anly_filename_base)

  Create ASCII file for either the FCST or ANLY files that are aggregated based on init time and storm id.

Public Attributes

- **p**
- **logger**
- **var_list**
- **stat_list**
- **regrid_with_met_tool**
- **extract_tiles_dir**
- **series_out_dir**
- **series_filtered_out_dir**
- **filter_opts**
- **fcst_ascii_file_prefix**
- **anly_ascii_file_prefix**
- **sbi_plotting_out_dir**
- **app_path**
- **app_name**
- **inaddons**
- **infiles**
- **outdir**
- **outfile**
- **args**

21.121.2 Member Function Documentation

21.121.2.1 build_and_run_series_request()

```
def series_by_init_wrapper.SeriesByInitWrapper.build_and_run_series_request (
              self,
              sorted_filter_init,
              tile_dir )
```

Build up the -obs, -fcst, -out necessary for running the series_analysis MET tool, then invoke series_analysis.

Args:

Parameters

sorted_filter_init	A list of the sorted directories corresponding to the init times that are the result of filtering. If filtering produced no results, this is the list of files created from running extract_tiles.
tile_dir	The directory where the input resides. Returns:

Definition at line 404 of file series_by_init_wrapper.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.run_all_times().

21.121.2.2 create_fcst_anly_to_ascii_file()

```
def series_by_init_wrapper.SeriesByInitWrapper.create_fcst_anly_to_ascii_file (
              self,
              fcst_anly_grid_files,
              cur_init,
              cur_storm,
              fcst_anly_filename_base )
```

Create ASCII file for either the FCST or ANLY files that are aggregated based on init time and storm id.

Args: fcst_anly_grid_files: A list of the FCST or ANLY gridded files under consideration.

cur_init: The initialization time of interest

cur_storm: The storm id of interest

fcst_anly_filename_base: The base name of the ASCII file (either ANLY_ASCII_FILES_ or FCST_ASCII_FILES_ which will be appended with the storm id.

Returns: None: Creates an ASCII file containing a list of either FCST or ANLY files based on init time and storm id.

Definition at line 745 of file series_by_init_wrapper.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.get_ascii_storm_files_list(), and series_by_init_↩ wrapper.SeriesByInitWrapper.get_storms_for_init().

21.121.2.3 create_obs_fcst_arg()

```
def series_by_init_wrapper.SeriesByInitWrapper.create_obs_fcst_arg (
              self,
              param_arg,
              ascii_file_base,
              cur_storm,
              cur_init )
```

Create the argument to the -obs or -fcst flag to the MET tool, series_analysis.

Args:

Definition at line 497 of file series_by_init_wrapper.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.build_and_run_series_request().

21.121.2.4 generate_plots()

```
def series_by_init_wrapper.SeriesByInitWrapper.generate_plots (
              self,
              sorted_filter_init,
              tile_dir )
```

Generate the plots from the series_analysis output.

Args:

Definition at line 588 of file series_by_init_wrapper.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.run_all_times(), and series_by_lead_wrapper.Series↩ ByLeadWrapper.run_all_times().

21.121.2.5 get_ascii_storm_files_list()

```
def series_by_init_wrapper.SeriesByInitWrapper.get_ascii_storm_files_list (
              self,
              tile_dir )
```

Creates the list of ASCII files that contain the storm id and init times.

The list is used to create an ASCII file which will be used as the option to the -obs or -fcst flag to the MET series↩ _analysis tool. Args:

Definition at line 325 of file series_by_init_wrapper.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.run_all_times().

21.121.2.6 get_fcst_file_info()

```
def series_by_init_wrapper.SeriesByInitWrapper.get_fcst_file_info (
              self,
              dir_to_search,
              cur_init,
              cur_storm )
```

Get the number of all the gridded forecast n x m tile files for a given storm id and init time (that were created by extract_tiles).

Determine the filename of the first and last files. This information is used to create the title value to the -title opt in plot_data_plane.

Args:

Returns: num, beg, end: A tuple representing the number of forecast tile files, and the first and last file.

sys.exit(1) otherwise

Definition at line 245 of file series_by_init_wrapper.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.generate_plots().

21.121.2.7 get_storms_for_init()

```
def series_by_init_wrapper.SeriesByInitWrapper.get_storms_for_init (
              self,
              cur_init,
              out_dir_base )
```

Retrieve all the filter files which have the .tcst extension.

Inside each file, extract the STORM_ID and append to the list, if the storm_list directory exists.

Args:

Returns: storm_list: A list of all the storms ids that correspond to this init time and actually has a directory in the init dir (additional filtering in a previous step may result in missing storm ids even though they are in the filter.tcst file)

Definition at line 703 of file series_by_init_wrapper.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.build_and_run_series_request(), series_by_init↩ _wrapper.SeriesByInitWrapper.generate_plots(), and series_by_init_wrapper.SeriesByInitWrapper.get_ascii_↩ storm_files_list().

21.121.2.8 is_netcdf_created()

```
def series_by_init_wrapper.SeriesByInitWrapper.is_netcdf_created (
              self )
```

Check for the presence of NetCDF files in the series_analysis_init directory.

Returns: is_created True if NetCDF files were found in the series_analysis_init/YYYYMMDD_hh/storm sub-directories, False otherwise.

Definition at line 212 of file series_by_init_wrapper.py.

Referenced by series_by_init_wrapper.SeriesByInitWrapper.run_all_times().

21.121.2.9 run_all_times()

```
def series_by_init_wrapper.SeriesByInitWrapper.run_all_times (
              self )
```

Invoke the series analysis script based on the init time in the format YYYYMMDD_hh.

Args:

Returns: None: Creates graphical plots of storm tracks

Definition at line 73 of file series_by_init_wrapper.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/series_by_init_wrapper.py

21.122 series_by_lead_wrapper.SeriesByLeadWrapper Class Reference

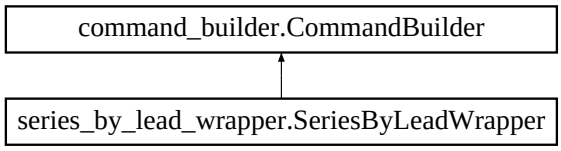
SeriesByLeadWrapper performs series analysis of paired data based on lead time and generates plots for each requested variable and statistic, as specified in a configuration/parameter file.

21.122.1 Detailed Description

SeriesByLeadWrapper performs series analysis of paired data based on lead time and generates plots for each requested variable and statistic, as specified in a configuration/parameter file.

Definition at line 32 of file series_by_lead_wrapper.py.

Inheritance diagram for series_by_lead_wrapper.SeriesByLeadWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def run_all_times (self)

  Perform a series analysis of extra tropical cyclone paired data based on lead time (forecast hour) This requires invoking the MET run_series_analysis binary, followed by generating graphics that are recognized by the MET viewer using the plot_data_plane and converting to postscript.
- def filter_with_tc_stat (self, tile_dir, init_times)

  Perform optional filtering using MET tc_stat.
- def perform_series_for_fhr_groups (self, tile_dir)

  Series analysis for groups based on forecast hours.
- def perform_series_for_all_fhrs (self, tile_dir, start, end, step)

  Performs a series analysis by lead time, based on a range and increment of forecast hours.
- def get_nseries (self, do_fhr_by_range, nc_var_file)

  Determine the number of series for this lead time and its associated variable via calculating the max series_cnt_T↩ OTAL value, maximum.
- def get_netcdf_min_max (self, do_fhr_by_range, nc_var_files, cur_stat)

  Determine the min and max for all lead times for each statistic and variable pairing.
- def retrieve_nc_files (self, do_fhr_by_range)

  Retrieve all the netCDF files created by MET series_analysis.
- def retrieve_fhr_tiles (self, tile_list, type_regex)

  Retrieves only the gridded tile files that correspond to the type.
- def find_matching_tile (self, fcst_file, anly_tiles)

  Find the corresponding ANLY 30x30 tile file to the fcst tile file.
- def cleanup_lead_ascii (self)

  Remove any pre-existing FCST and ANLY ASCII files created by previous runs of series_by_lead.
- def generate_plots (self, do_fhr_by_range)

  Generate the plots and animation GIFs for the series analysis results.
- def create_animated_gifs (self, do_fhr_by_range)

  Creates the animated GIF files from the .png files created in generate_plots().

Static Public Member Functions

- def get_var_ncfiles (do_fhr_by_range, cur_var, nc_list)

  Retrieve only the netCDF files corresponding to this statistic and variable pairing.
- def get_anly_or_fcst_files (filedir, file_type, filename_regex, cur_fhr)

  Get all the ANLY or FCST files by walking through the directories starting at filedir.

Public Attributes

- **p**
- **logger**
- **fhr_beg**
- **fhr_end**
- **fhr_inc**
- **fhr_group_beg_str**
- **fhr_group_end_str**
- **fhr_group_beg**
- **fhr_group_end**
- **fhr_group_labels**
- **var_list**
- **stat_list**
- **plot_data_plane_exe**
- **convert_exe**
- **ncap2_exe**
- **ncdump_exe**
- **rm_exe**
- **series_analysis_exe**
- **extract_tiles_dir**
- **series_lead_filtered_out_dir**
- **series_lead_out_dir**
- **tmp_dir**
- **background_map**
- **regrid_with_met_tool**
- **series_filter_opts**
- **fcst_ascii_regex**
- **anly_ascii_regex**
- **series_anly_configuration_file**
- **fcst_tile_regex**
- **anly_tile_regex**

21.122.2 Member Function Documentation

21.122.2.1 cleanup_lead_ascii()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.cleanup_lead_ascii (
              self )
```

Remove any pre-existing FCST and ANLY ASCII files created by previous runs of series_by_lead.

Args:

Returns: None: Removes any existing FCST and ANLY ASCII files which contains all the forecast and analysis gridded tiles.

Definition at line 1083 of file series_by_lead_wrapper.py.

21.122.2.2 create_animated_gifs()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.create_animated_gifs (
               self,
               do_fhr_by_range )
```

Creates the animated GIF files from the .png files created in generate_plots().

Args:

Definition at line 1275 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.run_all_times().

21.122.2.3 filter_with_tc_stat()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.filter_with_tc_stat (
               self,
               tile_dir,
               init_times )
```

Perform optional filtering using MET tc_stat.

Args:

Definition at line 247 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.run_all_times().

21.122.2.4 find_matching_tile()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.find_matching_tile (
               self,
               fcst_file,
               anly_tiles )
```

Find the corresponding ANLY 30x30 tile file to the fcst tile file.

Args:

Returns: anly_from_fcst (string): The name of the analysis tile file that corresponds to the same lead time as the input fcst tile.

Definition at line 1001 of file series_by_lead_wrapper.py.

21.122.2.5 generate_plots()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.generate_plots (
              self,
              do_fhr_by_range )
```

Generate the plots and animation GIFs for the series analysis results.

Args:

Returns: None

Definition at line 1113 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.run_all_times().

21.122.2.6 get_anly_or_fcst_files()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.get_anly_or_fcst_files (
              filedir,
              file_type,
              filename_regex,
              cur_fhr )  [static]
```

Get all the ANLY or FCST files by walking through the directories starting at filedir.

Args:

Returns: file_paths (string): a list of filenames (with full filepath)

Definition at line 1029 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.find_matching_tile(), series_by_lead_wrapper.↩ SeriesByLeadWrapper.perform_series_for_all_fhrs(), and series_by_lead_wrapper.SeriesByLeadWrapper.↩ perform_series_for_fhr_groups().

21.122.2.7 get_netcdf_min_max()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.get_netcdf_min_max (
              self,
              do_fhr_by_range,
              nc_var_files,
              cur_stat )
```

Determine the min and max for all lead times for each statistic and variable pairing.

Args:

Returns: tuple (vmin, vmax) vmin: The minimum vmax: The maximum

Definition at line 735 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.generate_plots().

21.122.2.8 get_nseries()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.get_nseries (
              self,
              do_fhr_by_range,
              nc_var_file )
```

Determine the number of series for this lead time and its associated variable via calculating the max series_cnt_↩ TOTAL value, maximum.

Args:

Returns: maximum (float): The maximum value of series_cnt_TOTAL of all the netCDF files for the variable cur_var.
None: If no max value is found.

Definition at line 644 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.generate_plots().

21.122.2.9 get_var_ncfiles()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.get_var_ncfiles (
              do_fhr_by_range,
              cur_var,
              nc_list )  [static]
```

Retrieve only the netCDF files corresponding to this statistic and variable pairing.

Args:

Returns: var_ncfiles: A list of netCDF files that correspond to this variable.

Definition at line 880 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.generate_plots().

21.122.2.10 perform_series_for_all_fhrs()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.perform_series_for_all_fhrs (
              self,
              tile_dir,
              start,
              end,
              step )
```

Performs a series analysis by lead time, based on a range and increment of forecast hours.
Invokes the MET tool Series-analysis

Args:

Returns: None

Definition at line 494 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.run_all_times().

21.122.2.11 perform_series_for_fhr_groups()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.perform_series_for_fhr_groups (
              self,
              tile_dir )
```

Series analysis for groups based on forecast hours.

Args:

Definition at line 309 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.run_all_times().

21.122.2.12 retrieve_fhr_tiles()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.retrieve_fhr_tiles (
              self,
              tile_list,
              type_regex )
```

Retrieves only the gridded tile files that correspond to the type.

Args:

Returns: fhr_tiles (string): A string of gridded tile names separated by newlines

Definition at line 966 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.perform_series_for_all_fhrs().

21.122.2.13 retrieve_nc_files()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.retrieve_nc_files (
              self,
              do_fhr_by_range )
```

Retrieve all the netCDF files created by MET series_analysis.

Args:

Definition at line 919 of file series_by_lead_wrapper.py.

Referenced by series_by_lead_wrapper.SeriesByLeadWrapper.generate_plots().

21.122.2.14 run_all_times()

```
def series_by_lead_wrapper.SeriesByLeadWrapper.run_all_times (
              self )
```

Perform a series analysis of extra tropical cyclone paired data based on lead time (forecast hour) This requires invoking the MET run_series_analysis binary, followed by generating graphics that are recognized by the MET viewer using the plot_data_plane and converting to postscript.

A pre-requisite is the presence of the filter file and storm files (set to nxm degree tiles as indicated in the param/config file) the specified init and lead times.

Create the following command to satisfy MET series_analysis: series_analysis -fcst <FILTERED_OUT_DI↩ R>/FCST_FILES_F<CUR_FHR> -obs <FILTERED_OUT_DIR>/ANLY_FILES_F<CUR_FHR> -out <OUT_↩ DIR>/series_F<CURR_FHR_<NAME>_<LEVEL>.nc -config SeriesAnalysisConfig_by_lead Args:

Returns: None: Invokes MET series_analysis and any other MET tool to perform series analysis. Then plots are generated for the variables and statistics (as indicated in the param/config file) corresponding to each forecast lead time.

Definition at line 102 of file series_by_lead_wrapper.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/series_by_lead_wrapper.py

21.123 produtil.prog.StreamGenerator Class Reference

This is part of the internal implementation of Runner, and is used to convert it to a produtil.pipeline.Pipeline for execution.

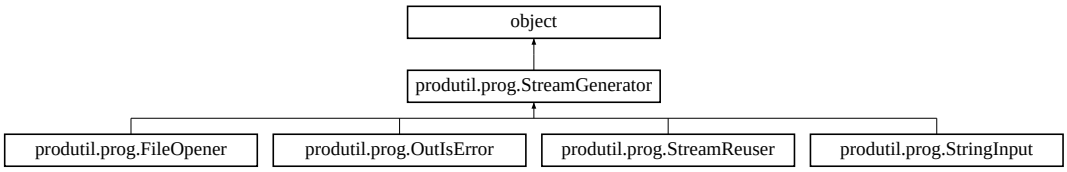
21.123.1 Detailed Description

This is part of the internal implementation of Runner, and is used to convert it to a produtil.pipeline.Pipeline for execution.

This is an abstract class whose subclasses create the Popen's stdout, stdin and stderr.

Definition at line 117 of file prog.py.

Inheritance diagram for produtil.prog.StreamGenerator:



Public Member Functions

- def for_input (self)
  Has no effect.
- def for_output (self)
  Has no effect.
- def repr_for_err (self)
  Returns the stderr value.

21.123.2 Member Function Documentation

21.123.2.1 for_input()

```
def produtil.prog.StreamGenerator.for_input (
              self )
```

Has no effect.

This exists only for debugging.

Definition at line 122 of file prog.py.

21.123.2.2 for_output()

```
def produtil.prog.StreamGenerator.for_output (
              self )
```

Has no effect.

This exists only for debugging.

Definition at line 125 of file prog.py.

21.123.2.3 repr_for_err()

```
def produtil.prog.StreamGenerator.repr_for_err (
              self )
```

Returns the stderr value.

The default implementation returns repr_for_out(), causing stderr to receive whatever stdout receives.

Definition at line 128 of file prog.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.124 produtil.prog.StreamReuser Class Reference

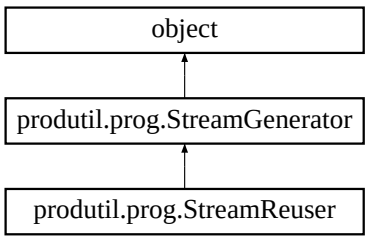
Arranges for a stream-like object to be sent to the stdout, stderr or stdin of a Runner.

21.124.1 Detailed Description

Arranges for a stream-like object to be sent to the stdout, stderr or stdin of a Runner.

Definition at line 240 of file prog.py.

Inheritance diagram for produtil.prog.StreamReuser:



Public Member Functions

- def __init__ (self, obj)

  Creates a StreamReuser for the specified stream-like object.
- def copy (self)

  Returns a shallow copy of this object.
- def to_shell (self)

  Raises NotValidPosixSh to indicate that the stream cannot be represented as POSIX sh.
- def repr_for_in (self)

  Returns repr(obj) where obj is the given stream-like object.
- def repr_for_out (self)

  Returns repr(obj) where obj is the given stream-like object.

Public Attributes

- obj

  the stream-like object to reuse.

21.124.2 Constructor & Destructor Documentation

21.124.2.1 __init__()

```
def produtil.prog.StreamReuser.__init__ (
              self,
              obj )
```

Creates a StreamReuser for the specified stream-like object.

Definition at line 243 of file prog.py.

21.124.3 Member Function Documentation

21.124.3.1 copy()

```
def produtil.prog.StreamReuser.copy (
              self )
```

Returns a shallow copy of this object.

Note that means that the underlying stream object is not copied.

Definition at line 249 of file prog.py.

Referenced by produtil.prog.ImmutableRunner.runner().

21.124.3.2 repr_for_in()

```
def produtil.prog.StreamReuser.repr_for_in (
              self )
```

Returns repr(obj) where obj is the given stream-like object.

Definition at line 262 of file prog.py.

21.124.3.3 repr_for_out()

```
def produtil.prog.StreamReuser.repr_for_out (
              self )
```

Returns repr(obj) where obj is the given stream-like object.

Definition at line 266 of file prog.py.

Referenced by produtil.prog.StreamGenerator.repr_for_err().

21.124.3.4 to_shell()

```
def produtil.prog.StreamReuser.to_shell (
              self )
```

Raises NotValidPosixSh to indicate that the stream cannot be represented as POSIX sh.

Definition at line 253 of file prog.py.

21.124.4 Member Data Documentation

21.124.4.1 obj

`produtil.prog.StreamReuser.obj`

the stream-like object to reuse.

Definition at line 246 of file prog.py.

Referenced by produtil.prog.StreamReuser.copy(), produtil.prog.StreamReuser.repr_for_in(), produtil.prog.StreamReuser.repr_for_out(), and produtil.prog.StreamReuser.to_shell().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.125 string_template_substitution.StringExtract Class Reference

21.125.1 Detailed Description

Definition at line 626 of file string_template_substitution.py.

Public Member Functions

- def **__init__** (self, log, temp, fstr)
- def **getValidTime** (self, fmt)
- def **getInitTime** (self, fmt)
- def **leadHour** (self)
- def **accumHour** (self)
- def **parseTemplate** (self)

Public Attributes

- **temp**
- **fstr**
- **validTime**
- **initTime**
- **leadTime**
- **accumTime**

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/string_template_substitution.py

21.126 produtil.prog.StringInput Class Reference

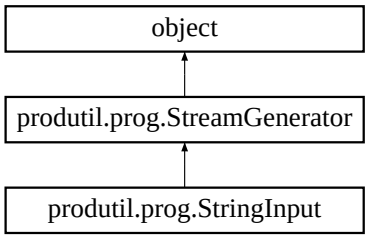
Represents sending a string to a process's stdin.

21.126.1 Detailed Description

Represents sending a string to a process's stdin.

Definition at line 205 of file prog.py.

Inheritance diagram for produtil.prog.StringInput:



Public Member Functions

- def __init__ (self, obj)

  Creates a StringInput that sends the specified object to stdin.
- def copy (self)

  Returns a shallow copy of this object.
- def __repr__ (self)

  Returns a string representation of this object as valid Python code.
- def to_shell (self)

  Converts this object, if possible, to an echo command followed by a pipe ("|").
- def repr_for_in (self)

  Part of the implementation of Runner.__repr__.

Public Attributes

- obj

  the object to send to stdin

21.126.2 Constructor & Destructor Documentation

21.126.2.1 __init__()

```
def produtil.prog.StringInput.__init__ (
              self,
              obj )
```

Creates a StringInput that sends the specified object to stdin.

Definition at line 207 of file prog.py.

21.126.3 Member Function Documentation

21.126.3.1 __repr__()

```
def produtil.prog.StringInput.__repr__ (
              self )
```

Returns a string representation of this object as valid Python code.

Definition at line 222 of file prog.py.

Referenced by produtil.prog.Runner.__str__().

21.126.3.2 copy()

```
def produtil.prog.StringInput.copy (
              self )
```

Returns a shallow copy of this object.

Definition at line 215 of file prog.py.

Referenced by produtil.prog.ImmutableRunner.runner().

21.126.3.3 repr_for_in()

```
def produtil.prog.StringInput.repr_for_in (
              self )
```

Part of the implementation of Runner.__repr__.

If possible, this creates valid Python code to represent specifying sending the given string to the stdin of a Runner. If the string is too long, it is abbreviated.

Definition at line 230 of file prog.py.

21.126.3.4 to_shell()

```
def produtil.prog.StringInput.to_shell (
              self )
```

Converts this object, if possible, to an echo command followed by a pipe ("|").

Definition at line 226 of file prog.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/prog.py

21.127 string_template_substitution.StringSub Class Reference

21.127.1 Detailed Description

```
log - log object
tmpl_str - template string to populate
kwargs - dictionary containing values for each template key

This class provides functionality for substituting values for
string templates.

Possible keys for vals:
   init - must be in YYYYmmddHH[MMSS] format
   valid - must be in YYYYmmddHH[MMSS] format
   lead - must be in HH[MMSS] format
   accum - must be in HH[MMSS] fomrat
   level - the level number as a string (?)
   model - the name of the model
   domain - the domain number (01, 02, etc.) read in as a string

See the description of doStringSub for further details.
```

Definition at line 122 of file string_template_substitution.py.

Public Member Functions

- def **__init__** (self, log, tmpl, kwargs)
- def dateCalcInit (self)
- def dateCalcValid (self)
- def dateCalcLead (self)
- def leadAccumFormat (self, parm_type, format_string)
- def doStringSub (self)

Public Attributes

- **logger**
- **tmpl**
- **kwargs**
- **lead_time_seconds**
- **accum_time_seconds**
- **negative_lead**
- **negative_accum**

21.127.2 Member Function Documentation

21.127.2.1 dateCalcInit()

```
def string_template_substitution.StringSub.dateCalcInit (
              self )
```

Calculate the init time from the valid and lead time

Definition at line 170 of file string_template_substitution.py.

Referenced by string_template_substitution.StringSub.doStringSub().

21.127.2.2 dateCalcLead()

```
def string_template_substitution.StringSub.dateCalcLead (
              self )
```

Calculate the lead time from the init and valid time

Definition at line 247 of file string_template_substitution.py.

Referenced by string_template_substitution.StringSub.doStringSub().

21.127.2.3 dateCalcValid()

```
def string_template_substitution.StringSub.dateCalcValid (
              self )
```

Calculate the valid time from the init and lead time

Definition at line 208 of file string_template_substitution.py.

Referenced by string_template_substitution.StringSub.doStringSub().

21.127.2.4 doStringSub()

```
def string_template_substitution.StringSub.doStringSub (
              self )


Populates the specified template with information from the
   kwargs dictionary.  The template structure is compresed of
   a fixed string populated with template place-holders inside curly
   braces, for example {tmpl_str}.  The tmpl_str must be present as
   a key in the kwargs dictionary, and the value will replace the
   {tmpl_str} in the returned string.

    In some cases, the template keys can have parameters containing
    formatting information. The format of the template in this case
    is {tmpl_str?parm=val}.  The supported parameters are:

    init, valid:
fmt - specifies a strftime format for the date/time
      e.g. %Y%m%d%H%M%S, %Y%m%d%H

      lead, accum:
fmt -  specifies an amount of time in [H]HH[MMSS] format
       e.g. %HH, %HHH, %HH%MMSS, %HHH%MMSS
```

Definition at line 445 of file string_template_substitution.py.

21.127.2.5 leadAccumFormat()

```
def string_template_substitution.StringSub.leadAccumFormat (
              self,
              parm_type,
              format_string )


Formats the lead or accum in the requested format
```

Definition at line 339 of file string_template_substitution.py.

Referenced by string_template_substitution.StringSub.doStringSub().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/string_template_substitution.py

21.128 produtil.datastore.Task Class Reference

Represents a process or actor that makes a Product.

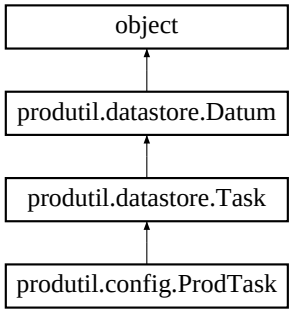
21.128.1 Detailed Description

Represents a process or actor that makes a Product.

A Task represents some work that needs to be done to produce Products. A task has a state (stored in the "available" metadata attribute), a location, whose meaning is up to the implementer to decide, and a logger.Logger. As with all Datum subclasses, a Task also has arbitrary metadata.

Definition at line 1052 of file datastore.py.

Inheritance diagram for produtil.datastore.Task:



Public Member Functions

- def __init__ (self, dstore, taskname, logger=None, kwargs)

  Task constructor.
- def jlogfile (self)

  returns the jlogfile logger.
- def postmsg (self, message, args, kwargs)

  same as produtil.log.jlogger.info()
- def setstate (self, val)

  Sets the state of this job.
- def getstate (self)

  Returns the job state.
- def strstate (self)

  A string representation of the job state.
- def gettaskname (self)

  Returns the task name part of the database ID, which is the same as the prodname.
- def products (self, args, kwargs)

  Iterate over the products this task produces.
- def log (self)

  Returns the logger object for this task.
- def clean (self)

  Cleans up any unneeded data used by this task.
- def unrun (self)

  Undoes the effect of run().
- def run (self)

  Performs the work this Task should do and generates all products.
- def is_completed (self)

  Is this task complete?
- def completed (self)

  Read-only property: is this task completed? Same as is_completed()
- def runpart (self)

  Run some of this task's work, deliver some products.

Properties

- state

  Read-write property: the job state.
- taskname

  Read-only property: the name of this task.

21.128.2 Constructor & Destructor Documentation

21.128.2.1 __init__()

```
def produtil.datastore.Task.__init__ (
              self,
              dstore,
              taskname,
              logger = None,
              kwargs )
```

Task constructor.

Creates a new Task from the given dataset and with the given task name.

Definition at line 1060 of file datastore.py.

21.128.3 Member Function Documentation

21.128.3.1 clean()

```
def produtil.datastore.Task.clean (
              self )
```

Cleans up any unneeded data used by this task.

Subclasses should override this function to clean up any unneeded temporary files or other unused resources consumed by the run() function. This default implementation does nothing.

Definition at line 1152 of file datastore.py.

21.128.3.2 completed()

```
def produtil.datastore.Task.completed (
              self )
```

Read-only property: is this task completed? Same as is_completed()

True if self.state==COMPLETED, False otherwise.

Definition at line 1189 of file datastore.py.

21.128.3.3 getstate()

```
def produtil.datastore.Task.getstate (
              self )
```

Returns the job state.

Returns the "available" attribute as an integer. This is used as the state of the Task. Typically, the return value should be one of: FAILED, UNSTARTED, RUNNING, PARTIAL, or COMPLETED.

Definition at line 1103 of file datastore.py.

21.128.3.4 gettaskname()

```
def produtil.datastore.Task.gettaskname (
              self )
```

Returns the task name part of the database ID, which is the same as the prodname.

Definition at line 1127 of file datastore.py.

21.128.3.5 is_completed()

```
def produtil.datastore.Task.is_completed (
              self )
```

Is this task complete?

Returns True if this task's state is COMPLETED, and False otherwise.

Definition at line 1180 of file datastore.py.

21.128.3.6 jlogfile()

```
def produtil.datastore.Task.jlogfile (
              self )
```

returns the jlogfile logger.

Returns a logging.Logger for the jlogfile. The jlogfile is intended to receive only major errors, and per-job start and completion information. This is equivalent to simply accessing produtil.log.jlogger.

Definition at line 1075 of file datastore.py.

21.128.3.7 log()

```
def produtil.datastore.Task.log (
              self )
```

Returns the logger object for this task.

Definition at line 1149 of file datastore.py.

21.128.3.8 postmsg()

```
def produtil.datastore.Task.postmsg (
              self,
              message,
              args,
              kwargs )
```

same as produtil.log.jlogger.info()

Sends a message to the multi-job shared log file at level INFO.

Definition at line 1084 of file datastore.py.

21.128.3.9 products()

```
def produtil.datastore.Task.products (
              self,
```

```
                args,
                kwargs )
```

Iterate over the products this task produces.

Iterates over some or all of the products produced by this task. The arguments are used to select subsets of the total set of products. Provide no arguments to get the full list of products. All subclasses should re-implement this method, and interpret the arguments in a way that makes sense to that class. The default implementation returns immediately without doing anything.

Definition at line 1136 of file datastore.py.

21.128.3.10 run()

```
def produtil.datastore.Task.run (
                self )
```

Performs the work this Task should do and generates all products.

Performs the work that this task is supposed to do. All subclasses should re-implement this method, and should set the state to COMPLETED the end. This implementation simply calls self.setstate(COMPLETED)

Postcondition

self.state=COMPLETED

Definition at line 1171 of file datastore.py.

Referenced by produtil.datastore.Task.runpart().

21.128.3.11 runpart()

```
def produtil.datastore.Task.runpart (
                self )
```

Run some of this task's work, deliver some products.

Performs a subset of the work that this task is supposed to do and returns. This is intended to be used for tasks that can be broken up into small pieces, such as post-processing all output files from a NWP simulation one by one. The default implementation simply calls self.run()

Definition at line 1193 of file datastore.py.

21.128.3.12 setstate()

```
def produtil.datastore.Task.setstate (
                self,
                val )
```

Sets the state of this job.

Sets the job stat to the specified value. This works by setting the "available" attribute to the specified integer. For compatibility with other scripts, this should be FAILED, UNSTARTED, RUNNING, PARTIAL or COMPLETED.

Definition at line 1094 of file datastore.py.

Referenced by produtil.datastore.Task.run().

21.128.3.13 strstate()

```
def produtil.datastore.Task.strstate (
              self )
```

A string representation of the job state.

Definition at line 1118 of file datastore.py.

21.128.3.14 unrun()

```
def produtil.datastore.Task.unrun (
              self )
```

Undoes the effect of run().

Cleans up this Task's work areas, "undelivers" all deliverables, and makes it look like the task has never been run. All subclasses should re-implement this method, and must also "unrun" everything their parent class runs. The default implementation simply calls self.clean() and sets the state to UNSTARTED.

Postcondition

self.state=UNSTARTED

Definition at line 1159 of file datastore.py.

21.128.4 Property Documentation

21.128.4.1 state

```
produtil.datastore.Task.state  [static]
```

Initial value:

```
= property(getstate,setstate,None,
        )
```

Read-write property: the job state.

Can be FAILED, UNSTARTED, RUNNING, PARTIAL or COMPLETED.

Definition at line 1114 of file datastore.py.

Referenced by produtil.datastore.Task.completed(), and produtil.datastore.Task.is_completed().

21.128.4.2 taskname

```
produtil.datastore.Task.taskname  [static]
```

Initial value:

```
= property(gettaskname,None,None,
        )
```

Read-only property: the name of this task.

Same as self.prodname.

Definition at line 1134 of file datastore.py.

Referenced by produtil.config.ProdTask.get_outdir(), produtil.config.ProdTask.get_workdir(), and produtil.config.↩ ProdTask.log().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.129 task_info.TaskInfo Class Reference

21.129.1 Detailed Description

Definition at line 28 of file task_info.py.

Public Member Functions

- def __init__ (self)
- def **getValidTime** (self)
- def **getInitTime** (self)

Public Attributes

- **init_time**
- **valid_time**
- **lead**
- **level**
- **fcst_var**
- **ob_type**

21.129.2 Constructor & Destructor Documentation

21.129.2.1 __init__()

```
def task_info.TaskInfo.__init__ (
              self )
```

```
Retrieve parameters from corresponding param file
```

Definition at line 31 of file task_info.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/task_info.py

21.130 tcmpr_plotter_wrapper.TCMPRPlotterWrapper Class Reference

A Python class than encapsulates the plot_tcmpr.R plotting script.

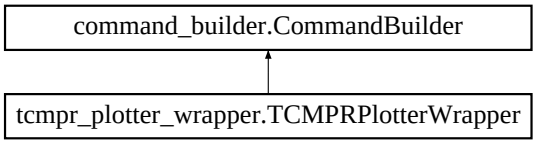
21.130.1 Detailed Description

A Python class than encapsulates the plot_tcmpr.R plotting script.

Generates plots for input files with .tcst format and creates output subdirectory based on the input tcst file. The plot_tcmpr.R plot also supports additional filtering by calling MET tool tc_stat. This wrapper extends plot_tcmpr.R by allowing the user to specify as input a directory (to support plotting all files in the specified directory and its subdirectories). The user can now either indicate a file or directory in the (required) -lookin option.

Definition at line 33 of file tcmpr_plotter_wrapper.py.

Inheritance diagram for tcmpr_plotter_wrapper.TCMPRPlotterWrapper:



Public Member Functions

- def __init__ (self, p, logger)

  Constructor for TCMPRPlotterWrapper Args:
- def run_all_times (self)

  Builds the command for invoking tcmpr.R plot script.
- def create_output_subdir (self, tcst_file)

  Extract the base portion of the tcst filename: eg amlqYYYYMMDDhh.gfso.nnnn in /d1/username/tc_pairs/YYYYM↩ M/amlqYYYYMMDDhh.gfso.nnnn and use this as the subdirectory (gets appended to the TCMPR output directory).
- def retrieve_optionals (self)

Public Attributes

- **config**
- **logger**
- **tcmpr_script**
- **input_data**
- **plot_config_file**
- **output_base_dir**
- **prefix**
- **title**
- **subtitle**
- **xlab**
- **ylab**
- **xlim**
- **ylim**
- **filter**
- **filtered_tcst_data**
- **dep_vars**
- **scatter_x**
- **scatter_y**
- **skill_ref**
- **series**
- **series_ci**
- **legend**
- **lead**
- **plot_types**
- **rp_diff**
- **demo_year**
- **hfip_baseline**
- **footnote_flag**
- **plot_config_options**
- **save_data**
- **no_ee**
- **no_log**
- **save**

21.130.2 Constructor & Destructor Documentation

21.130.2.1 __init__()

```
def tcmpr_plotter_wrapper.TCMPRPlotterWrapper.__init__ (
              self,
              p,
              logger )
```

Constructor for TCMPRPlotterWrapper Args:

Definition at line 45 of file tcmpr_plotter_wrapper.py.

21.130.3 Member Function Documentation

21.130.3.1 create_output_subdir()

```
def tcmpr_plotter_wrapper.TCMPRPlotterWrapper.create_output_subdir (
              self,
              tcst_file )
```

Extract the base portion of the tcst filename: eg amlqYYYYMMDDhh.gfso.nnnn in /d1/username/tc_pairs/YYYYM↩
M/amlqYYYYMMDDhh.gfso.nnnn and use this as the subdirectory (gets appended to the TCMPR output directory).

This allows the user to determine which plots correspond to the input track file.

Args:

Definition at line 250 of file tcmpr_plotter_wrapper.py.

21.130.3.2 retrieve_optionals()

```
def tcmpr_plotter_wrapper.TCMPRPlotterWrapper.retrieve_optionals (
              self )
```

```
Creates a list of the optional options if they are defined.
```

Definition at line 271 of file tcmpr_plotter_wrapper.py.

Referenced by tcmpr_plotter_wrapper.TCMPRPlotterWrapper.run_all_times().

21.130.3.3 run_all_times()

```
def tcmpr_plotter_wrapper.TCMPRPlotterWrapper.run_all_times (
              self )
```

Builds the command for invoking tcmpr.R plot script.

Args:

Returns:

Definition at line 124 of file tcmpr_plotter_wrapper.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/tcmpr_plotter_wrapper.py

21.131 tc_pairs_wrapper.TcPairsWrapper Class Reference

Wraps the MET tool, tc_pairs to parse and match ATCF adeck and bdeck files.

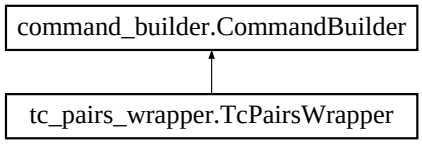
21.131.1 Detailed Description

Wraps the MET tool, tc_pairs to parse and match ATCF adeck and bdeck files.

Pre-processes extra tropical cyclone data.

Definition at line 41 of file tc_pairs_wrapper.py.

Inheritance diagram for tc_pairs_wrapper.TcPairsWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def read_modify_write_file (self, in_csvfile, storm_month, missing_values, out_csvfile)

  Reads, modifies and writes file Args:
- def run_all_times (self)

  Build up the command to invoke the MET tool tc_pairs.
- def setup_tropical_track_dirs (self, deck_input_file_path, deck_file_path, storm_month, missing_values)

  Set up the adeck or bdeck file paths.
- def build_tc_pairs (self, pairs_output_dir, date_file, adeck_file_path, bdeck_file_path)

  Build up the command that is used to run the MET tool, tc_pairs.

Public Attributes

- **config**
- **logger**

21.131.2 Member Function Documentation

21.131.2.1 build_tc_pairs()

```
def tc_pairs_wrapper.TcPairsWrapper.build_tc_pairs (
              self,
              pairs_output_dir,
              date_file,
              adeck_file_path,
              bdeck_file_path )
```

Build up the command that is used to run the MET tool, tc_pairs.

Args:

Definition at line 337 of file tc_pairs_wrapper.py.

Referenced by tc_pairs_wrapper.TcPairsWrapper.run_all_times(), and tc_pairs_wrapper.TcPairsWrapper.setup_↩
tropical_track_dirs().

21.131.2.2 read_modify_write_file()

```
def tc_pairs_wrapper.TcPairsWrapper.read_modify_write_file (
              self,
              in_csvfile,
              storm_month,
              missing_values,
              out_csvfile )
```

Reads, modifies and writes file Args:

Definition at line 53 of file tc_pairs_wrapper.py.

Referenced by tc_pairs_wrapper.TcPairsWrapper.setup_tropical_track_dirs().

21.131.2.3 setup_tropical_track_dirs()

```
def tc_pairs_wrapper.TcPairsWrapper.setup_tropical_track_dirs (
              self,
              deck_input_file_path,
              deck_file_path,
              storm_month,
              missing_values )
```

Set up the adeck or bdeck file paths.

If these correspond to a tropical storm, then perform additional processing via read_modify_write_file to conform to ATCF format.

Args:

Definition at line 272 of file tc_pairs_wrapper.py.

Referenced by tc_pairs_wrapper.TcPairsWrapper.run_all_times().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/tc_pairs_wrapper.py

21.132 tc_stat_wrapper.TcStatWrapper Class Reference

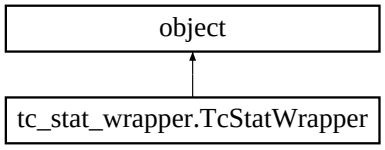
Wrapper for the MET tool, tc_stat, which is used to filter tropical cyclone pair data.

21.132.1 Detailed Description

Wrapper for the MET tool, tc_stat, which is used to filter tropical cyclone pair data.

Definition at line 41 of file tc_stat_wrapper.py.

Inheritance diagram for tc_stat_wrapper.TcStatWrapper:



Public Member Functions

- def **__init__** (self, p)
- def build_tc_stat (self, series_output_dir, cur_init, tile_dir, filter_opts)

  Create the call to MET tool TC-STAT to subset tc-pairs output based on the criteria specified in the parameter/config file.

Public Attributes

- **tc_exe**
- **logger**
- **config**

21.132.2 Member Function Documentation

21.132.2.1 build_tc_stat()

```
def tc_stat_wrapper.TcStatWrapper.build_tc_stat (
              self,
              series_output_dir,
              cur_init,
              tile_dir,
              filter_opts )
```

Create the call to MET tool TC-STAT to subset tc-pairs output based on the criteria specified in the parameter/config file.

Args:

Returns: None: if no error, then invoke MET tool TC-STAT and subsets tc-pairs data, creating a filter.tcst file.

Raises CalledProcessError

Definition at line 55 of file tc_stat_wrapper.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/tc_stat_wrapper.py

21.133 produtil.cd.TempDir Class Reference

This class is intended to be used with the Python "with TempDir() as t" syntax.

21.133.1 Detailed Description

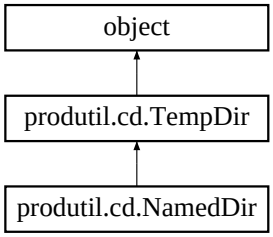
This class is intended to be used with the Python "with TempDir() as t" syntax.

Example:

```
with TempDir() as t:
    # we're now in the temporary directory
    ...do things...
# the temporary directory has been deleted now
```

Definition at line 38 of file cd.py.

Inheritance diagram for produtil.cd.TempDir:



Public Member Functions

- def __init__ (self, suffix='.tmp', prefix='tempdir.', dir=None, keep=False, logger=None, print_on_exception=True, add_perms=perm_add, remove_perms=perm_remove, keep_on_error=True, cd=True)

  Creates a TempDir.
- def name_make_dir (self)

  Decide the name of the directory, and create the directory.
- def mkdir_cd (self)

  Creates the temporary directory and chdirs the current process into that directory.
- def cd_out (self)

  Exit the temporary directory created by mkdir_cd and return to the original directory, if possible.
- def cd_rmdir (self)

  CD out and remove the old directory.
- def exception_info (self)

  Called to dump information to a log, or failing that, the terminal if an unexpected exception is caught.
- def __enter__ (self)

  This is a simple wrapper around mkdir_cd that is intended to be used with in a "with" block.
- def __exit__ (self, etype, value, traceback)

  Exit the 'with' block.

Public Attributes

- dirname

  The name of the target directory.
- suffix

  Temporary directory name suffix.
- prefix

  Temporary directory name prefix.
- print_on_exception

  Should we print exceptions before exiting the directory?
- dir

  The directory object.
- olddir

  The name of the directory we came from.

21.133.2 Constructor & Destructor Documentation

21.133.2.1 __init__()

```
def produtil.cd.TempDir.__init__ (
              self,
              suffix = '.tmp',
              prefix = 'tempdir.',
              dir = None,
              keep = False,
              logger = None,
              print_on_exception = True,
              add_perms = perm_add,
              remove_perms = perm_remove,
              keep_on_error = True,
              cd = True )
```

Creates a TempDir.

Definition at line 50 of file cd.py.

21.133.3 Member Function Documentation

21.133.3.1 __enter__()

```
def produtil.cd.TempDir.__enter__ (
              self )
```

This is a simple wrapper around mkdir_cd that is intended to be used with in a "with" block.

This subroutine is automatically called at the beginning of the block.

Definition at line 193 of file cd.py.

21.133.3.2 __exit__()

```
def produtil.cd.TempDir.__exit__ (
              self,
              etype,
              value,
              traceback )
```

Exit the 'with' block.

This is a simple wrapper around cd_rmdir that is intended to be used with in a "with" block. This subroutine is automatically called at the end of the block. It will call cd_rmdir to delete the directory unless an exception is thrown that is NOT a subclass of Exception or GeneratorExit. The removal is skipped to allow the program to exit quickly in case of a fatal signal (ie.: SIGQUIT, SIGTERM, SIGINT, SIGHUP).

Definition at line 199 of file cd.py.

21.133.3.3 cd_out()

```
def produtil.cd.TempDir.cd_out (
              self )
```

Exit the temporary directory created by mkdir_cd and return to the original directory, if possible.

Definition at line 135 of file cd.py.

Referenced by produtil.cd.TempDir.__exit__(), and produtil.cd.TempDir.cd_rmdir().

21.133.3.4 cd_rmdir()

```
def produtil.cd.TempDir.cd_rmdir (
              self )
```

CD out and remove the old directory.

This subroutine exits the temporary directory created by mkdir_cd, and then deletes that temporary directory. After this routine, the process will be in its original directory (from before the call to mkdir_cd) if possible, or otherwise it will be in the root directory (/).

It is the caller's responsibility to ensure this function is not called if keep_on_error=True and an error occurs.

Definition at line 141 of file cd.py.

Referenced by produtil.cd.TempDir.__exit__().

21.133.3.5 exception_info()

```
def produtil.cd.TempDir.exception_info (
              self )
```

Called to dump information to a log, or failing that, the terminal if an unexpected exception is caught.

Definition at line 181 of file cd.py.

Referenced by produtil.cd.TempDir.__exit__().

21.133.3.6 mkdir_cd()

```
def produtil.cd.TempDir.mkdir_cd (
              self )
```

Creates the temporary directory and chdirs the current process into that directory.

It calls self.name_make_dir() to do the naming and directory creation.

Definition at line 126 of file cd.py.

Referenced by produtil.cd.TempDir.__enter__().

21.133.3.7 name_make_dir()

```
def produtil.cd.TempDir.name_make_dir (
              self )
```

Decide the name of the directory, and create the directory.

Also create any path components leading up to the directory.

Definition at line 108 of file cd.py.

Referenced by produtil.cd.TempDir.mkdir_cd().

21.133.4 Member Data Documentation

21.133.4.1 dir

`produtil.cd.TempDir.dir`

The directory object.

Definition at line 78 of file cd.py.

Referenced by produtil.cd.TempDir.name_make_dir().

21.133.4.2 dirname

`produtil.cd.TempDir.dirname`

The name of the target directory.

Definition at line 72 of file cd.py.

Referenced by produtil.cd.TempDir.cd_rmdir(), produtil.cd.TempDir.exception_info(), produtil.cd.TempDir.mkdir_↩cd(), produtil.cd.TempDir.name_make_dir(), and produtil.cd.NamedDir.name_make_dir().

21.133.4.3 olddir

`produtil.cd.TempDir.olddir`

The name of the directory we came from.

Definition at line 79 of file cd.py.

Referenced by produtil.cd.TempDir.cd_out(), and produtil.cd.TempDir.mkdir_cd().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cd.py

21.134 produtil.log.ThreadLogger Class Reference

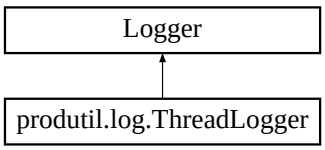
Custom logging.Logger that inserts thread information.

21.134.1 Detailed Description

Custom logging.Logger that inserts thread information.

Definition at line 45 of file log.py.

Inheritance diagram for produtil.log.ThreadLogger:



Public Member Functions

- def makeRecord (self, name, lvl, fn, lno, msg, args, kwargs)

  Replaces the logging.Logger.makeRecord() with a new implementation that inserts thread information from threading.current_thread()

21.134.2 Member Function Documentation

21.134.2.1 makeRecord()

```
def produtil.log.ThreadLogger.makeRecord (
              self,
              name,
              lvl,
              fn,
              lno,
              msg,
              args,
              kwargs )
```

Replaces the logging.Logger.makeRecord() with a new implementation that inserts thread information from threading.current_thread()

Definition at line 47 of file log.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/log.py

21.135 produtil.numerics.TimeArray Class Reference

A time-indexed array that can only handle equally spaced times.

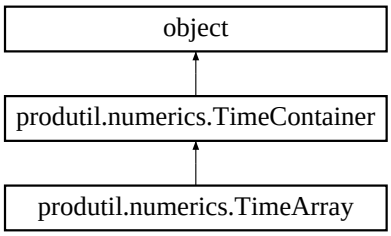
21.135.1 Detailed Description

A time-indexed array that can only handle equally spaced times.

This implements an array-like object that uses time as an index. It recognizes only discrete times as specified by a start, an end and a timestep. The end is only used as a guideline: if it does not lie exactly on a timestep, the next timestep end is used. Note that all methods in this class have the same meaning as the built-in list class, but accept times as input, except if specified otherwise. In all cases, the time "index" is anything accepted by to_datetime_rel(...,self.start).

Definition at line 722 of file numerics.py.

Inheritance diagram for produtil.numerics.TimeArray:



Public Member Functions

- def __init__ (self, start, end, timestep, init=None)

  TimeArray constructor.
- def index_of (self, when)

  Returns the index of the specified time in the internal storage arrays or raises NotInTimespan if the time is not in the timespan.

Public Attributes

- start

  Start time.
- end

  End time.
- timestep

  Timestep between times.

21.135.2 Constructor & Destructor Documentation

21.135.2.1 __init__()

```
def produtil.numerics.TimeArray.__init__ (
              self,
              start,
              end,
              timestep,
              init = None )
```

TimeArray constructor.

Definition at line 733 of file numerics.py.

21.135.3 Member Function Documentation

21.135.3.1 index_of()

```
def produtil.numerics.TimeArray.index_of (
              self,
              when )
```

Returns the index of the specified time in the internal storage arrays or raises NotInTimespan if the time is not in the timespan.

Definition at line 759 of file numerics.py.

21.135.4 Member Data Documentation

21.135.4.1 end

```
produtil.numerics.TimeArray.end
```

End time.

Do not modify.

Definition at line 748 of file numerics.py.

21.135.4.2 start

`produtil.numerics.TimeArray.start`

Start time.

Do not modify.

Definition at line 747 of file numerics.py.

21.135.4.3 timestep

`produtil.numerics.TimeArray.timestep`

Timestep between times.

Do not modify.

Definition at line 749 of file numerics.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.136 produtil.numerics.TimeContainer Class Reference

Abstract base class that maps from time to objects.

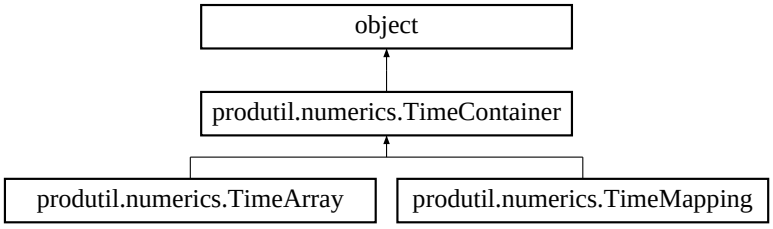
21.136.1 Detailed Description

Abstract base class that maps from time to objects.

This is the abstract base class of any class that represents a mapping from a set of times to a set of data. It provides the underlying implementation of TimeArray and TimeMapping. This is not exported by "from produtil.↩ numerics import ∗" to prevent accidental use.

Definition at line 526 of file numerics.py.

Inheritance diagram for produtil.numerics.TimeContainer:



Public Member Functions

- def __init__ (self, times, init=None)

  TimeContainer constructor.
- def at_index (self, index)

  Returns the data at the given index, or raises KeyError if no data exists.
- def index_of (self, when)

  This virtual function should be implemented in a subclass.
- def lasttime (self)

  Returns the last time in the array or list of times, even if it has no data.
- def firsttime (self)

  Returns the first time in the array or list of times, even if it has no data.
- def __getitem__ (self, when)

  Returns the item at the latest time that is not later than "when".
- def neartime (self, when, epsilon=None)

  Returns a tuple containing the time nearest to "when" without going over, and the index of that time.
- def get (self, when, default)

  Returns the item at the latest time that is not later than "when.".
- def __setitem__ (self, when, val)

  Finds the latest time that is not later than "when" in self._times.
- def __iter__ (self)

  Iterates over all data.
- def itervalues (self)

  Iterates over data for all known times that have data.
- def iterkeys (self)

  Iterates over all times that have data.
- def iteritems (self)

  Iterates over all known times that have data, returning a tuple containing the time and the data at that time.
- def __reversed__ (self)

  Iterates over all known times that have data, in reverse order.
- def __delitem__ (self, when)

  Finds the latest time that is not later than "when" and removes the data it is mapped to.
- def __contains__ (self, when)

  Finds the latest time that is not later than "when" in self._times.
- def __len__ (self)

  Returns the number of times that have data.
- def times (self)

  Iterates over all times in this TimeContainer.
- def datatimes (self)

  Iterates over all times in this TimeContainer that map to data.
- def __str__ (self)

  Returns a string representation of this object.
- def datatimes_reversed (self)

  Iterates in reverse order over all times in this TimeContainer that map to data.

21.136.2 Constructor & Destructor Documentation

21.136.2.1 __init__()

```
def produtil.numerics.TimeContainer.__init__ (
               self,
               times,
               init = None )
```

TimeContainer constructor.

Initializes the internal arrays for the given list of times, which must not be empty. Note that strange, potentially bad things will happen if there are duplicate times.

Implementation notes:

```
self._times[N]: a list of times
self._data[N]: the data for each time
self._assigned[N]: True if there is data for this time,
  False otherwise
```

where N is the length of the input times array. The _data will be filled with init() and _assigned filled with True if init is present and not None, otherwise _assigned will be False.

Definition at line 535 of file numerics.py.

21.136.3 Member Function Documentation

21.136.3.1 __contains__()

```
def produtil.numerics.TimeContainer.__contains__ (
               self,
               when )
```

Finds the latest time that is not later than "when" in self._times.

Returns True if there is data mapped to that time and False otherwise.

Definition at line 678 of file numerics.py.

21.136.3.2 __delitem__()

```
def produtil.numerics.TimeContainer.__delitem__ (
              self,
              when )
```

Finds the latest time that is not later than "when" and removes the data it is mapped to.

Later calls to **getitem**, get, **iter** and so on, will not return any data for this time.

Definition at line 669 of file numerics.py.

21.136.3.3 __getitem__()

```
def produtil.numerics.TimeContainer.__getitem__ (
              self,
              when )
```

Returns the item at the latest time that is not later than "when".

If there is no data assigned at that time, then KeyError is raised.

Definition at line 590 of file numerics.py.

21.136.3.4 __iter__()

```
def produtil.numerics.TimeContainer.__iter__ (
              self )
```

Iterates over all data.

Definition at line 640 of file numerics.py.

21.136.3.5 __len__()

```
def produtil.numerics.TimeContainer.__len__ (
              self )
```

Returns the number of times that have data.

Definition at line 688 of file numerics.py.

21.136.3.6 __reversed__()

```
def produtil.numerics.TimeContainer.__reversed__ (
              self )
```

Iterates over all known times that have data, in reverse order.

Definition at line 661 of file numerics.py.

21.136.3.7 __setitem__()

```
def produtil.numerics.TimeContainer.__setitem__ (
              self,
              when,
              val )
```

Finds the latest time that is not later than "when" in self._times.

Assigns "val" to that time.

Definition at line 630 of file numerics.py.

21.136.3.8 __str__()

```
def produtil.numerics.TimeContainer.__str__ (
              self )
```

Returns a string representation of this object.

Definition at line 700 of file numerics.py.

21.136.3.9 at_index()

```
def produtil.numerics.TimeContainer.at_index (
              self,
              index )
```

Returns the data at the given index, or raises KeyError if no data exists.

Definition at line 567 of file numerics.py.

21.136.3.10 datatimes()

```
def produtil.numerics.TimeContainer.datatimes (
              self )
```

Iterates over all times in this TimeContainer that map to data.

Definition at line 695 of file numerics.py.

21.136.3.11 datatimes_reversed()

```
def produtil.numerics.TimeContainer.datatimes_reversed (
              self )
```

Iterates in reverse order over all times in this TimeContainer that map to data.

Definition at line 711 of file numerics.py.

21.136.3.12 firsttime()

```
def produtil.numerics.TimeContainer.firsttime (
              self )
```

Returns the first time in the array or list of times, even if it has no data.

Definition at line 586 of file numerics.py.

21.136.3.13 get()

```
def produtil.numerics.TimeContainer.get (
              self,
              when,
              default )
```

Returns the item at the latest time that is not later than "when.".

Definition at line 619 of file numerics.py.

21.136.3.14 index_of()

```
def produtil.numerics.TimeContainer.index_of (
              self,
              when )
```

This virtual function should be implemented in a subclass.

It should return the index of the time to use for the given time.

Definition at line 573 of file numerics.py.

Referenced by produtil.numerics.TimeContainer.__contains__(), produtil.numerics.TimeContainer.__delitem_↩
_(), produtil.numerics.TimeContainer.__getitem__(), produtil.numerics.TimeContainer.__setitem__(), produtil.↩
numerics.TimeContainer.get(), and produtil.numerics.TimeContainer.neartime().

21.136.3.15 iteritems()

```
def produtil.numerics.TimeContainer.iteritems (
              self )
```

Iterates over all known times that have data, returning a tuple containing the time and the data at that time.

Definition at line 655 of file numerics.py.

21.136.3.16 iterkeys()

```
def produtil.numerics.TimeContainer.iterkeys (
              self )
```

Iterates over all times that have data.

Definition at line 650 of file numerics.py.

21.136.3.17 itervalues()

```
def produtil.numerics.TimeContainer.itervalues (
              self )
```

Iterates over data for all known times that have data.

Definition at line 645 of file numerics.py.

21.136.3.18 lasttime()

```
def produtil.numerics.TimeContainer.lasttime (
              self )
```

Returns the last time in the array or list of times, even if it has no data.

Definition at line 581 of file numerics.py.

21.136.3.19 neartime()

```
def produtil.numerics.TimeContainer.neartime (
              self,
              when,
              epsilon = None )
```

Returns a tuple containing the time nearest to "when" without going over, and the index of that time.

Definition at line 599 of file numerics.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.137 produtil.numerics.TimeError Class Reference

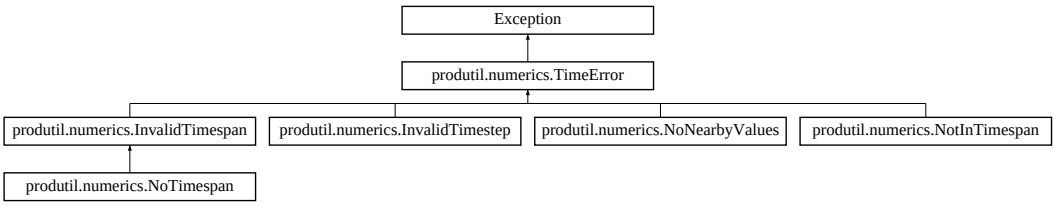
Base class used for time-related exceptions.

21.137.1 Detailed Description

Base class used for time-related exceptions.

Definition at line 18 of file numerics.py.

Inheritance diagram for produtil.numerics.TimeError:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.138 produtil.numerics.TimeMapping Class Reference

Maps from an ordered list of times to arbitrary data.

21.138.1 Detailed Description

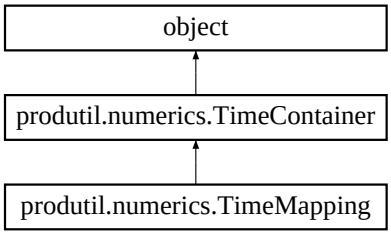
Maps from an ordered list of times to arbitrary data.

A TimeMapping is a mapping from an ordered list of times to a set of data. The full set of possible times is given in the constructor: it is not possible to add new times to the mapping after creation of the TimeMapping. A TimeMapping is more expensive than a TimeArray since every [] lookup (**getitem**) has to do a binary search. However, it is more general since the times do not have to be exactly at an interval.

Note that a TimeArray can replace a TimeMapping if a small enough timestep (the greatest common factor) is used since most of the TimeContainer methods only work on the indices that have data. However, if some timespans have dense timesteps and the rest are sparse, there may be significant memory and CPU savings in using a TimeMapping.

Definition at line 779 of file numerics.py.

Inheritance diagram for produtil.numerics.TimeMapping:



Public Member Functions

- def __init__ (self, times, init=None)

  TimeMapping constructor.
- def index_of (self, when)

  Returns the index of the specified time in the internal storage arrays or raises NotInTimespan if the time is not in the timespan.

21.138.2 Constructor & Destructor Documentation

21.138.2.1 __init__()

```
def produtil.numerics.TimeMapping.__init__ (
              self,
              times,
              init = None )
```

TimeMapping constructor.

Definition at line 796 of file numerics.py.

21.138.3 Member Function Documentation

21.138.3.1 index_of()

```
def produtil.numerics.TimeMapping.index_of (
              self,
              when )
```

Returns the index of the specified time in the internal storage arrays or raises NotInTimespan if the time is not in the timespan.

Definition at line 804 of file numerics.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/numerics.py

21.139 produtil.datastore.Transaction Class Reference

Datastore transaction support.

21.139.1 Detailed Description

Datastore transaction support.

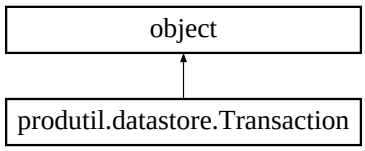
Implements sqlite3 transactions on a Datastore. A transaction is a set of reads and updates that should either ALL be done, or NONE at all. Transactions also speed up the script, sometimes by as much of a factor of 300, by grouping I/O operations to the database into one large chunk. However, one must be careful in using them since it keeps the database locked for an extended period of time.

This class should not be used directly. Instead, one should do this to a Datum (Task or Product) object:

with datum_object.transaction() as t: ... do things to the datum object ... transaction is now complete, database is updated.

Definition at line 270 of file datastore.py.

Inheritance diagram for produtil.datastore.Transaction:



Public Member Functions

- def __init__ (self, ds)
  Transaction constructor.
- def __enter__ (self)
  Locks the database for the current thread, if it isn't already locked.
- def __exit__ (self, etype, evalue, traceback)
  Releases the database lock if this is the last Transaction released for the current thread.
- def query (self, stmt, subvals=())
  Performs an SQL query returning the result of cursor.fetchall()
- def mutate (self, stmt, subvals=())
  Performs an SQL database modification, returning the result of cursor.lastrowid.
- def init_datum (self, d, meta=True)
  Add a Datum to the database if it is not there already.
- def update_datum (self, d)
  Update database availability and location records.
- def refresh_meta (self, d, or_add=True)
  Replace Datum metadata with database values, add new metadata to database.
- def set_meta (self, d, k, v)
  Sets metadata key k to value v for the given Datum.
- def del_meta (self, d, k)
  Delete metadata from the database.
- ds
  The Datastore containing the database for which this is a transaction.

Public Attributes

- **ds**

21.139.2 Constructor & Destructor Documentation

21.139.2.1 __init__()

```
def produtil.datastore.Transaction.__init__ (
              self,
              ds )
```

Transaction constructor.

Creates the Transaction object but does NOT initiate the transaction.

Definition at line 287 of file datastore.py.

21.139.3 Member Function Documentation

21.139.3.1 __enter__()

```
def produtil.datastore.Transaction.__enter__ (
              self )
```

Locks the database for the current thread, if it isn't already locked.

Definition at line 294 of file datastore.py.

21.139.3.2 __exit__()

```
def produtil.datastore.Transaction.__exit__ (
              self,
              etype,
              evalue,
              traceback )
```

Releases the database lock if this is the last Transaction released for the current thread.

Definition at line 303 of file datastore.py.

21.139.3.3 del_meta()

```
def produtil.datastore.Transaction.del_meta (
              self,
              d,
              k )
```

Delete metadata from the database.

Deletes the specified metadata key k, which must not be "location" or "available".

Definition at line 410 of file datastore.py.

21.139.3.4 init_datum()

```
def produtil.datastore.Transaction.init_datum (
              self,
              d,
              meta = True )
```

Add a Datum to the database if it is not there already.

Given a Datum, add the object to the database if it is not there already and fill the object with data from the database.

Definition at line 327 of file datastore.py.

Referenced by produtil.datastore.Transaction.refresh_meta().

21.139.3.5 mutate()

```
def produtil.datastore.Transaction.mutate (
              self,
```

```
                stmt,
                subvals = () )
```

Performs an SQL database modification, returning the result of cursor.lastrowid.

Definition at line 320 of file datastore.py.

Referenced by produtil.datastore.Transaction.del_meta(), produtil.datastore.Transaction.init_datum(), produtil.↩ datastore.Transaction.set_meta(), and produtil.datastore.Transaction.update_datum().

21.139.3.6 query()

```
def produtil.datastore.Transaction.query (
                self,
                stmt,
                subvals = () )
```

Performs an SQL query returning the result of cursor.fetchall()

Definition at line 314 of file datastore.py.

Referenced by produtil.datastore.Transaction.init_datum(), and produtil.datastore.Transaction.refresh_meta().

21.139.3.7 refresh_meta()

```
def produtil.datastore.Transaction.refresh_meta (
                self,
                d,
                or_add = True )
```

Replace Datum metadata with database values, add new metadata to database.

Given a Datum d, discards and replaces d._meta with the current metadata, location and availability. Will raise an exception if the product does not exist in the database.

Definition at line 368 of file datastore.py.

Referenced by produtil.datastore.Transaction.init_datum().

21.139.3.8 set_meta()

```
def produtil.datastore.Transaction.set_meta (
              self,
              d,
              k,
              v )
```

Sets metadata key k to value v for the given Datum.

Modifies the database entries for key k and datum d to have the value v. If k is location or available, then the product table will be updated instead.

Definition at line 395 of file datastore.py.

21.139.3.9 update_datum()

```
def produtil.datastore.Transaction.update_datum (
              self,
              d )
```

Update database availability and location records.

Given a Datum whose location and availability is set, update that information in the database, adding the Datum if necessary.

Definition at line 357 of file datastore.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.140 produtil.cluster.UCARYellowstone Class Reference

Represents the Yellowstone cluster.

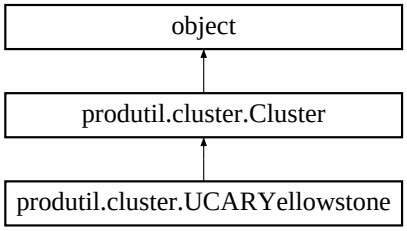
21.140.1 Detailed Description

Represents the Yellowstone cluster.

Does not allow ACLs, assumes group quotas.

Definition at line 194 of file cluster.py.

Inheritance diagram for produtil.cluster.UCARYellowstone:



Public Member Functions

- def __init__ (self)

  Constructor for UCARYellowstone.

Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cluster.py

21.141 produtil.fileop.UnexpectedAbsolutePath Class Reference

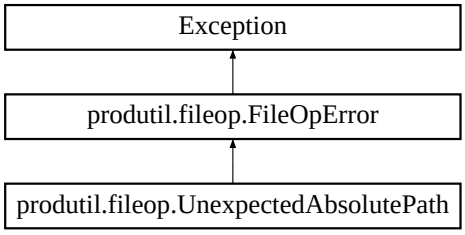
This exception indicates that the renamer function sent to make_symlinks_in returned an absolute path.

21.141.1 Detailed Description

This exception indicates that the renamer function sent to make_symlinks_in returned an absolute path.

Definition at line 66 of file fileop.py.

Inheritance diagram for produtil.fileop.UnexpectedAbsolutePath:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.142 produtil.datastore.UnknownLocation Class Reference

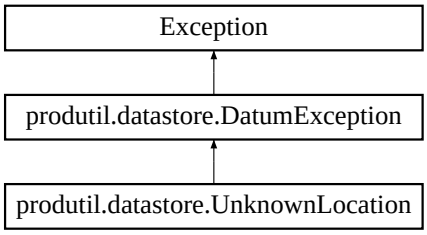
Raised when delivering data, but no location is provided.

21.142.1 Detailed Description

Raised when delivering data, but no location is provided.

Definition at line 82 of file datastore.py.

Inheritance diagram for produtil.datastore.UnknownLocation:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.143 produtil.datastore.UpstreamFile Class Reference

Represents a Product created by an external workflow.

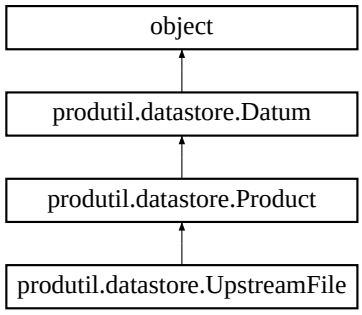
21.143.1 Detailed Description

Represents a Product created by an external workflow.

This subclass of Product represents a file that is created by an external workflow. It implements a check() call that determines if the file is larger than a minimum size and older than a minimum age. Once the file is large enough and old enough, it sets the location and availability. Any calls to undeliver() or deliver() raise InvalidOperation.

Definition at line 915 of file datastore.py.

Inheritance diagram for produtil.datastore.UpstreamFile:



Public Member Functions

- def check (self, frominfo=None, minsize=None, minage=None, logger=None)

  Checks the specified file to see if it is available.
- def undeliver (self)

  Undelivering an UpstreamFile merely sets the internal "available" flag to False.
- def deliver (self, location=None, frominfo=None)

  Raises InvalidOperation.

Public Attributes

- **available**

Additional Inherited Members

21.143.2 Member Function Documentation

21.143.2.1 check()

```
def produtil.datastore.UpstreamFile.check (
              self,
              frominfo = None,
              minsize = None,
              minage = None,
              logger = None )
```

Checks the specified file to see if it is available.

Looks for the file on disk. Updates the "available" and "location" attributes of this Product. Uses two metadata values to decide whether the file is "available" if it exists:

self["minsize"] - minimum size in bytes. Default: 0 self["minage"] - minimum age in seconds. Default: 20

Both can be overridden by corresponding arguments. Note that one must be careful with the minimum age on poorly-maintained clusters due to clock skews.

Definition at line 924 of file datastore.py.

Referenced by produtil.fileop.FileWaiter.checkfiles().

21.143.2.2 deliver()

```
def produtil.datastore.UpstreamFile.deliver (
              self,
              location = None,
              frominfo = None )
```

Raises InvalidOperation.

You cannot deliver an upstream file. The upstream workflow must do that for you. Call check() instead to see if the file has been delivered.

Definition at line 969 of file datastore.py.

21.143.2.3 undeliver()

```
def produtil.datastore.UpstreamFile.undeliver (
              self )
```

Undelivering an UpstreamFile merely sets the internal "available" flag to False.

It does not remove the data.

Definition at line 965 of file datastore.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/datastore.py

21.144 usage_wrapper.UsageWrapper Class Reference

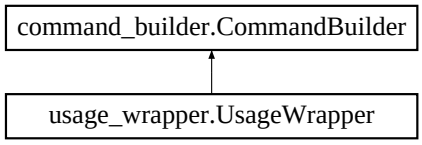
A default process, prints out usage when nothing is defined in the PROCESS_LIST of the parm/metplus_↩ config/metplus_runtime.conf and no lower level config files are included.

21.144.1 Detailed Description

A default process, prints out usage when nothing is defined in the PROCESS_LIST of the parm/metplus_↩ config/metplus_runtime.conf and no lower level config files are included.

Definition at line 19 of file usage_wrapper.py.

Inheritance diagram for usage_wrapper.UsageWrapper:



Public Member Functions

- def **__init__** (self, p, logger)
- def **run_all_times** (self)

Public Attributes

- **logger**
- **available_processes**

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/usage_wrapper.py

21.145 produtil.fileop.VerificationFailed Class Reference

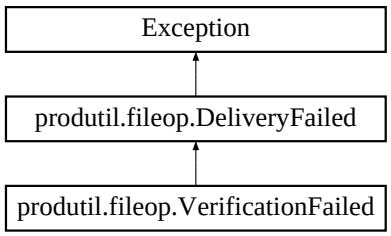
This exception is raised when a copy of a file has different content than the original.

21.145.1 Detailed Description

This exception is raised when a copy of a file has different content than the original.

Definition at line 115 of file fileop.py.

Inheritance diagram for produtil.fileop.VerificationFailed:



Public Member Functions

- def __init__ (self, message, fromfile, tofile, verifyfile)

  VerificationFailed constructor.
- def __str__ (self)

  Human-readable description of this error.
- def __repr__ (self)

  Pythonic representation of this error.

Public Attributes

- verifyfile

  The file to verify.

21.145.2 Constructor & Destructor Documentation

21.145.2.1 __init__()

```
def produtil.fileop.VerificationFailed.__init__ (
              self,
              message,
              fromfile,
              tofile,
              verifyfile )
```

VerificationFailed constructor.

Definition at line 118 of file fileop.py.

21.145.3 Member Function Documentation

21.145.3.1 __repr__()

```
def produtil.fileop.VerificationFailed.__repr__ (
              self )
```

Pythonic representation of this error.

Definition at line 133 of file fileop.py.

Referenced by produtil.prog.Runner.__str__().

21.145.3.2 __str__()

```
def produtil.fileop.VerificationFailed.__str__ (
              self )
```

Human-readable description of this error.

Definition at line 129 of file fileop.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.146 produtil.cluster.WCOSSCray Class Reference

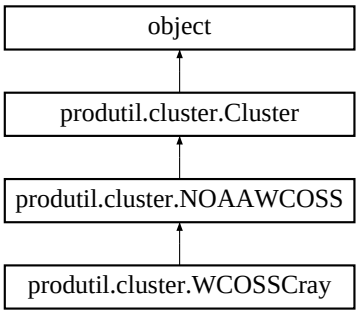
This subclass of NOAAWCOSS handles the new Cray portions of WCOSS: Luna and Surge.

21.146.1 Detailed Description

This subclass of NOAAWCOSS handles the new Cray portions of WCOSS: Luna and Surge.

Definition at line 322 of file cluster.py.

Inheritance diagram for produtil.cluster.WCOSSCray:



Public Member Functions

- def partition (self)

  Returns "cray" to indicate the user is on the Cray side of WCOSS.
- def wcoss_phase (self)

  Returns 0 to indicate that this is not the IBM part of WCOSS.

Additional Inherited Members

21.146.2 Member Function Documentation

21.146.2.1 wcoss_phase()

```
def produtil.cluster.WCOSSCray.wcoss_phase (
              self )
```

Returns 0 to indicate that this is not the IBM part of WCOSS.

Returns

0

Definition at line 347 of file cluster.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cluster.py

21.147 produtil.cluster.WisconsinS4 Class Reference

Represents the S4 cluster.

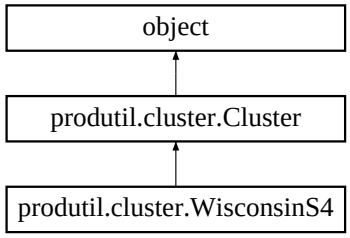
21.147.1 Detailed Description

Represents the S4 cluster.

Does not allow ACLs, assumes group quotas.

Definition at line 202 of file cluster.py.

Inheritance diagram for produtil.cluster.WisconsinS4:



Public Member Functions

- def __init__ (self)

  Constructor for WisconsinS4.

Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/cluster.py

21.148 produtil.workpool.WorkPool Class Reference

A pool of threads that perform some list of tasks.

21.148.1 Detailed Description

A pool of threads that perform some list of tasks.

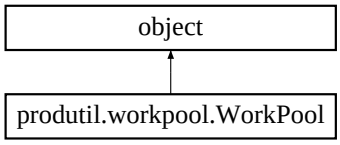
There is a function add_work() that adds a task to be performed.

Example: print the numbers from 1 to 10 in no particular order, in three threads:

```
def printit(num):
  print str(num)
with WorkPool(3) as w:
  print "three threads are waiting for work"
  for x in xrange(10):
    w.add_work(printit,[x+1])
  print "all threads have work, but the work may not be complete"
  w.barrier()
  print "all work is now complete."
print "once you get here, all workpool threads exited"
```

Definition at line 84 of file workpool.py.

Inheritance diagram for produtil.workpool.WorkPool:



Public Member Functions

- def __init__ (self, nthreads, logger=None, raise_at_exit=False)

  Create a WorkPool with the specified number of worker threads.
- def __enter__ (self)

  Does nothing.
- def __exit__ (self, etype, value, traceback)

  Called at the bottom of a "with" block.
- def exceptions (self)

  Iterates over all exceptions from worker threads.
- def nthreads (self)

  The number of worker threads.
- def add_work (self, work, args=None)

  Adds a piece of work to be done.
- def start_threads (self, n)

  Starts n new threads.
- def kill_threads (self)

  Kills all worker threads.
- def barrier (self)

  Waits for all threads to reach the barrier function.

Public Attributes

- logger

  a logging.Logger for log messages
- die

  If True, all threads should exit immediately.

21.148.2 Constructor & Destructor Documentation

21.148.2.1 __init__()

```
def produtil.workpool.WorkPool.__init__ (
              self,
              nthreads,
              logger = None,
              raise_at_exit = False )
```

Create a WorkPool with the specified number of worker threads.

The nthreads must be at least 1.

Definition at line 103 of file workpool.py.

21.148.3 Member Function Documentation

21.148.3.1 __enter__()

```
def produtil.workpool.WorkPool.__enter__ (
              self )
```

Does nothing.

Called from atop a "with" block.

Definition at line 125 of file workpool.py.

21.148.3.2 __exit__()

```
def produtil.workpool.WorkPool.__exit__ (
              self,
              etype,
              value,
              traceback )
```

Called at the bottom of a "with" block.

If no exception was raised, and no "break" encountered, then waits for work to complete, and then kills threads. Upon a fatal signal or break, kills threads as quickly as possible.

Definition at line 128 of file workpool.py.

21.148.3.3 add_work()

```
def produtil.workpool.WorkPool.add_work (
              self,
              work,
              args = None )
```

Adds a piece of work to be done.

It must be a callable object. If there are no worker threads, the work() is called immediately. The args are passed, if present.

Definition at line 201 of file workpool.py.

Referenced by produtil.workpool.WorkPool.barrier().

21.148.3.4 barrier()

```
def produtil.workpool.WorkPool.barrier (
              self )
```

Waits for all threads to reach the barrier function.

This can only be called by the master thread.

Upon calling, the master thread adds a WorkTask for each thread, telling the thread to call self.barrier(). Once all threads have reached that point, the barrier returns in all threads.

Definition at line 339 of file workpool.py.

Referenced by produtil.workpool.WorkPool.__exit__(), and produtil.workpool.WorkPool.barrier().

21.148.3.5 exceptions()

```
def produtil.workpool.WorkPool.exceptions (
              self )
```

Iterates over all exceptions from worker threads.

Definition at line 149 of file workpool.py.

Referenced by produtil.workpool.WorkPool.__exit__().

21.148.3.6 kill_threads()

```
def produtil.workpool.WorkPool.kill_threads (
              self )
```

Kills all worker threads.

Can only be called from the thread that made this object.

Definition at line 310 of file workpool.py.

Referenced by produtil.workpool.WorkPool.__exit__().

21.148.3.7 nthreads()

```
def produtil.workpool.WorkPool.nthreads (
              self )
```

The number of worker threads.

Definition at line 197 of file workpool.py.

Referenced by produtil.workpool.WorkPool.add_work(), and produtil.workpool.WorkPool.barrier().

21.148.3.8 start_threads()

```
def produtil.workpool.WorkPool.start_threads (
              self,
              n )
```

Starts n new threads.

Can only be called from the thread that made this object.

Definition at line 276 of file workpool.py.

21.148.4 Member Data Documentation

21.148.4.1 die

`produtil.workpool.WorkPool.die`

If True, all threads should exit immediately.

Definition at line 288 of file workpool.py.

Referenced by produtil.workpool.WorkPool.add_work(), and produtil.workpool.WorkPool.kill_threads().

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/workpool.py

21.149 produtil.workpool.WorkTask Class Reference

Stores a piece of work.

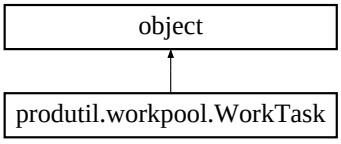
21.149.1 Detailed Description

Stores a piece of work.

This is an internal implementation class. Do not use it directly. It stores one piece of work to be done by a worker thread in a WorkPool.

Definition at line 16 of file workpool.py.

Inheritance diagram for produtil.workpool.WorkTask:



Public Member Functions

- def __init__ (self, work, args=None)

  Create a WorkTask whose job is to call work()
- def args (self)

  The arguments to the work function.

Public Attributes

- work

  The function this WorkTask should call.

Properties

- exception

  The exception that was raised by the work function.
- done

  Is this work task done?

21.149.2 Constructor & Destructor Documentation

21.149.2.1 __init__()

```
def produtil.workpool.WorkTask.__init__ (
              self,
              work,
              args = None )
```

Create a WorkTask whose job is to call work()

Definition at line 21 of file workpool.py.

21.149.3 Property Documentation

21.149.3.1 done

produtil.workpool.WorkTask.done [static]

Initial value:

```
= property(_get_done,_set_done,_del_done,
                 )
```

Is this work task done?

Definition at line 73 of file workpool.py.

21.149.3.2 exception

produtil.workpool.WorkTask.exception [static]

Initial value:

```
= property(_get_exception,_set_exception,_del_exception,
                       )
```

The exception that was raised by the work function.

Definition at line 55 of file workpool.py.

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/workpool.py

21.150 produtil.mpi_impl.mpi_impl_base.WrongMPI Class Reference

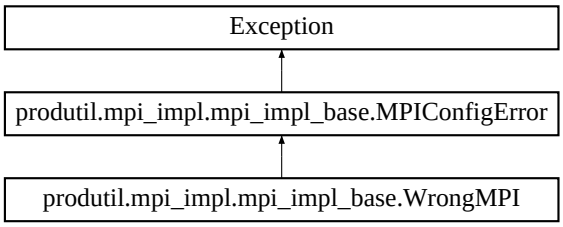
Unused: raised when the wrong MPI implementation is accessed.

21.150.1 Detailed Description

Unused: raised when the wrong MPI implementation is accessed.

Definition at line 19 of file mpi_impl_base.py.

Inheritance diagram for produtil.mpi_impl.mpi_impl_base.WrongMPI:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/mpi_impl/mpi_impl_base.py

21.151 produtil.fileop.WrongSymlink Class Reference

Raised when os.symlink makes a symlink to a target other than the one that was requested.

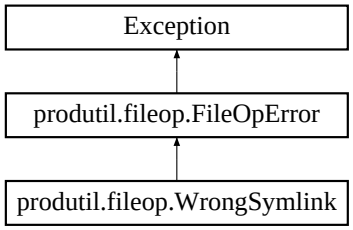
21.151.1 Detailed Description

Raised when os.symlink makes a symlink to a target other than the one that was requested.

This is present to detect a bug in Cray where os.symlink randomly makes a symlink to the wrong place.

Definition at line 81 of file fileop.py.

Inheritance diagram for produtil.fileop.WrongSymlink:



Additional Inherited Members

The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/fileop.py

21.152 produtil.workpool.WrongThread Class Reference

Raised when a thread unrelated to a WorkPool attempts to interact with the WorkPool.

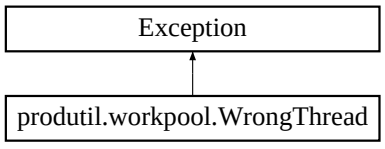
21.152.1 Detailed Description

Raised when a thread unrelated to a WorkPool attempts to interact with the WorkPool.

Only the thread that called the constructor, and the threads created by the WorkPool can interact with it.

Definition at line 10 of file workpool.py.

Inheritance diagram for produtil.workpool.WrongThread:



The documentation for this class was generated from the following file:

- /home/minnawin/wip_10-31/METplus/ush/produtil/workpool.py