pyfact Documentation

Release 0.1

Werner Lustermann

April 05, 2012

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

pyfact provides support for analysing FACT data using python. This comprises:

- classes for accessing and analysing FACT data
- tools for specific tasks
- general python language support
- examples

If you are new to python you need to learn the basics. In any case it would be wise to study the following book:

A Primer on Scientific Programming with Python - Hans Petter Langtangen

pyfact is supposed to follow closely the programming style thought in this book.

python documentation pages

astropython.org

1.1 Python version and modules used

As long as you work on the ISDC FACT cluster most things should be just fine. |python version: 2.6.6 |— add later: list of modules

1.2 Style guide and coding conventions

The code of pyfact follows the ‘Style Guide for python code <http://www.python.org/dev/peps/pep-0008/>.

CHAPTER

TWO

GETTING STARTED

1. Get an account at the FACT data center at ISDC
2. Setup your environment variables
3. Checkout the pyfact software repository
4. Try a few examples

2.1 1. Important links

FACT data center at ISDC <http://www.isdc.unige.ch/fact/datacenter>** here you find instructions how to get an account, how to set it up and how to use it

SVN repository <https://fact.isdc.unige.ch/svn/fact/> hosts all FACT software (including pyfact)

FACT run database <https://www.fact-project.org/run_db/db/fact_runinfo.php> here you find information about all available runs

eLogbook <https://www.fact-project.org/logbook/> telescope logbook used since Dec. 1, 2012; for logbook information before this date have a look here: <http://fact.ethz.ch/FACTelog/index.jsp>

FACT La Palma pages <https://www.fact-project.org/> plenty of information concerning the telescope in La Palma

2.2 2. Environment variables

set the following env variables depending on the SHELL you are using, for instance in .bashrc or .tcshrc

- **ROOTSYS**: /swdev_nfs/root_v5.28.00
- **PATH**: add $ROOTSYS/bin
- **PATH**: add /swdev_nfs/FACT++
- **LD_LIBRARY_PATH**: $ROOTSYS/lib:/swdev_nfs/FACT++/.libs
- **PYTHONPATH**: $ROOTSYS/lib
- **PYTHONPATH: add all directories where python should search for modules. At least:**
  - pyscripts/pyfact
  - pyscript/tools
  - pyscripts/ecamples

the absolute path depends on where you have checked (or will check) out the pyscripts repository

2.3 3. Check out the repository

svn co https://fact.isdc.unige.ch/svn/fact/tools/pyscripts/

2.4 4. Create pyfits_h.so library

FACT data are stored in (gzipped) fits files, but the data files are too large to be read by the existing tools:

pyfits <http://www.stsci.edu/institute/software_hardware/pyfits> which uses cfitsio.

To mitigate this problem a C++ class defined in fits.h [1] is used. A simple possibility to create an interface of this C++ class is to use ROOT and later to import it using the pyroot module.

got to the directory: pyscripts/pyfact

simple:

```
[lusterma@isdc-cn02 pyfact]$ root
ROOT 5.28/00 (trunk@37585, Dec 14 2010, 15:20:27 on linuxx8664gcc)
CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.

root [0] .L fits.h++
Info in <TUnixSystem::ACLiC>: creating shared library /home_nfs/isdc/lusterma/pyscripts/pyfact/./fits
root [1]
```

This might not work for whatenver reason then it should be done like this:

```
$rootcint -f my_dict.C -c fits.h izstream.h
$g++ -fPIC -c -I$ROOTSYS/include my_dict.C -o my_dict.o
$g++ -o fits_h.so -shared my_dict.o
```

Now you can check if the reading rawdata is working:

$ pyfact.py

this should print some data from a few events

2.5 5. Run examples

[1] fits.h (here named pyfits.h) is part of MARS and was written by T.Bretz, it was added to this repository for convienience

CHAPTER

THREE

CLASSES

3.1 pyfact.py

3.1.1 rawdata access

class pyfact.**RawData**(*data_file_name*, *calib_file_name*, *user_action_calib=<function <lambda> at 0xab028b4>*, *baseline_file_name=''*, *return_dict=None*)

raw data access and calibration

- open raw data file and drs calibration file
- performs amplitude calibration
- performs baseline substraction if wanted
- provides all data in an array: row = number of pixel col = length of region of interest

initialize object

open data file and calibration data file get basic information about the data in data_file_name allocate buffers for data access

data_file_name : fits or fits.gz file of the data including the path calib_file_name : fits or fits.gz file containing DRS calibration data baseline_file_name : npy file containing the baseline values

baseline_correct()

subtract baseline from the data

baseline_read_values(*file*, *bsl_hist='bsl_sum/hplt_mean'*)

open ROOT file with baseline histogram and read baseline values file name of the root file bsl_hist path to the histogram containing the basline values

board_times

time when the FAD was triggered, in some strange units...

calib_file

drs calibration file

calibrate_drs_amplitude()

perform the drs amplitude calibration of the event data

data

1D array with raw data

data_file

data file (fits object)

info()
print run information

nevents
number of events in the data run

next()
used by __iter__

next_event()
load the next event from disk and calibrate it

npix
number of pixels (should be 1440)

nroi
region of interest (number of DRS slices read)

start_cells
slice where drs readout started

3.1.2 fnames of a data run

class pyfact.**fnames**(*specifier=['012', '023', '2011', '11', '24'], rpath='/scratch_nfs/res/bsl/', zipped=True*)
organize file names of a FACT data run

specifier [list of strings defined as:] ['DRS calibration file', 'Data file', 'YYYY', 'MM', 'DD']

rpath : directory path for the results; YYYYMMDD will be appended to rpath zipped : use zipped (True) or unzipped (Data)

info()
print complete filenames

make(*specifier*, *rpath*, *zipped*)
create (make) the filenames

names : dictionary of filenames, tags { 'data', 'drscal', 'results' } data : name of the data file drscal : name of the drs calibration file results : radikal of file name(s) for results (to be completed by suffixes)

CHAPTER

FOUR

CLASSES

4.1 pyfact.py

4.1.1 rawdata access

4.1.2 fnames of a data run

class pyfact.**fnames**(*specifier=['012', '023', '2011', '11', '24'], rpath='/scratch_nfs/res/bsl/', zipped=True*)

organize file names of a FACT data run

specifier [list of strings defined as:] ['DRS calibration file', 'Data file', 'YYYY', 'MM', 'DD']

rpath : directory path for the results; YYYYMMDD will be appended to rpath zipped : use zipped (True) or unzipped (Data)

info()

print complete filenames

make(*specifier*, *rpath*, *zipped*)

create (make) the filenames

names : dictionary of filenames, tags { 'data', 'drscal', 'results' } data : name of the data file drscal : name of the drs calibration file results : radikal of file name(s) for results (to be completed by suffixes)

4.2 FIR filter functions

CHAPTER

FIVE

EXAMPLES

5.1 calling a system command

Using the os module any command executable on the command line can be called within a script. This is in particular true for your own python scripts:

```
import os
os.system('echo long listing of dir; pwd; ls -l')
```

or suppose you created a script my_script.py:

```
from os import system
system('python my_scrip.py')
```

CHAPTER

SIX

PHYTON IN FACT

Python is installed on the fact cluster at ISDC. Presently python 2.4.3 an upgrade, is planned soon.

The presence of the following python modules is supposed. They are not part of the standard distribution.:

- numpy: basis of numerical computation in python
- scipy: scientific computation in python including Fourier transforms, signal processing
- matplotlib: provides a matlab like plotting environment (still missing!!!)
- pyfits: easy access to fits files

Other packages used in pyfact being part of a standard python installation:

- os: misc functions providing access to the operating system. Among others a function system is provided, allowing to run system commands.

CHAPTER

SEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

INDEX