ISSN 2039-7941
Anno 2018_Numero 396

rapporti tecnici INGV

PyGLog: a Python software for handling GNSS metadata and log-files

396

Istituto Nazionale di Geofisica e Vulcanologia

Direttore Responsabile
Silvia MATTONI

Editorial Board
Luigi CUCCI - Editor in Chief (INGV-RM1)
Raffaele AZZARO (INGV-CT)
Mario CASTELLANO (INGV-NA)
Viviana CASTELLI (INGV-BO)
Rosa Anna CORSARO (INGV-CT)
Mauro DI VITO (INGV-NA)
Marcello LIOTTA (INGV-PA)
Mario MATTIA (INGV-CT)
Milena MORETTI (INGV-ONT)
Nicola PAGLIUCA (INGV-RM1)
Umberto SCIACCA (INGV-RM2)
Alessandro SETTIMI
Salvatore STRAMONDO (INGV-ONT)
Andrea TERTULLIANI (INGV-RM1)
Aldo WINKLER (INGV-RM2)

Segreteria di Redazione
Francesca Di Stefano - Referente
Rossella Celi
Tel. +39 06 51860068
redazionecen@ingv.it

in collaborazione con:
Barbara Angioni (RM1)

REGISTRAZIONE AL TRIBUNALE DI ROMA N.173 | 2014, 23 LUGLIO
© 2014 INGV Istituto Nazionale di Geofisica e Vulcanologia
Rappresentante legale: Carlo DOGLIONI
Sede: Via di Vigna Murata, 605 | Roma

ISSN 2039-7941
Anno 2018_Numero 396

rapporti tecnici INGV

PYGLOG: A PYTHON SOFTWARE FOR HANDLING GNSS METADATA AND LOG-FILES

Daniele Randazzo[1], Enrico Serpelloni[1], Adriano Cavaliere[2]

[1]**INGV** (Istituto Nazionale di Geofisica e Vulcanologia, Osservatorio Nazionale Terremoti)
[2]**INGV** (Istituto Nazionale di Geofisica e Vulcanologia, Sezione di Bologna)

396

How to cite: Randazzo D., Serpelloni E., Cavaliere A., (2018). PyGLog: a Python software for handling GNSS metadata and log-files. Rapp. Tec. INGV, 396: 1-26.

Index

Abstract

We describe a software developed to handle metadata of Global Navigation Satellite System (GNSS) stations. The number of available GNSS sites in the Euro-Mediterranean and African area has grown significantly in the last decade, pushing toward the development of automatic procedures for the analysis of the raw observations. Currently >3000 stations are routinely processed at the GPS data analysis center based on the GAMIT/GLOBK software operating at INGV-Bologna. Here we describe a software, written in Python, developed with the goal of processing metadata associated with continuously operating GNSS stations. The metadata is, generally, an ASCII file associated with each station, containing information about the geodetic antenna, the receiver, the radome model and the antenna offset (distance between the phase center of the antenna and the reference point, depending on the mechanical structure of the antenna mount). Commonly, but not always, GNSS stations metadata are provided in the form of log-files, which are however most of time compiled by human operators and later made available on the web for public access. On the research side these metadata are fundamental for a proper data processing and an accurate estimate of geophysical information, and need to be converted in a particular standard format (station info file), depending on the software adopted for data reduction, with coherent information to not stall the elaboration. The check of metadata coming from a huge amount of GNSS stations from different networks is a time consuming task, which compromises the efficiency of the research job. The software presented in this work aims at automatically update the repository of thousands of station metadata files, checking the coherence of the information and creating the station info files, in different formats, needed for processing, thus requiring a minimum effort for data processing to the operators.

Key words

Global Navigation Satellite System, Global Positioning System (GPS) – Continuous GPS (CGPS) – station info – log file – metadata – offset – repository – data processing – station-info.

Introduction

The “PyGLog” software is intended to automatically retrieve the log files (metadata) from different GNSS networks servers, to keep the local repository updated, to properly interpret the information contained in the original log-files, to compare new and old log files and to create the station-info files in different formats, compatible with most common geodetic-level software tools, limiting the occurrence of errors due to human interaction on the metadata. The software also creates a specific log file associated to the GNSS network, in order to save the information about the status of elaborated stations; this file is sent, via email, to alert the operator whenever an error occur.

In order to allow the software to manage different GPS networks, the following issues have been faced:

- The GNSS network data and metadata are provided on the web either through FTP server connection or through dynamic link with an HTTP server. While the first provides direct access to the files and related information, the HTTP protocol generally provides a not standard web page, requiring to parse the HTML code in order to find the link and other information. To properly detect the new and old log-file, it’s relevant to know and compare the “timestamp” of the file, which is the date and time of its creation; when it’s not available, the current time is used.
- Some Data Repository uses UNAVCO’s GSAC software (https://www.unavco.org/software/data-management/gsac/gsac.html) for data search and downloads. In this case only the GAMIT format station.info and SINEX files are available and a different approach is required to retrieve the metadata.
- Some Repository provides log-file in compressed format, to unzip before processing.
- The filenames do not have a standard format and often the timestamp is not included.
- Most of the web servers provide the filename in the format “ssss_yyyymmdd” (“s” = station_id, “y” = year, “m” = month and “d” = day) but it’s also possible to find multiple files with the same station_id and different dates, requiring to select only the newest one from the source.
- Inside the station log-file the parameters are described by the “field name” and are grouped into sub

paragraphs as shown in Appendix A; Although rare, not all of the files are provided in English language, requiring thus to interpret the fields.
- The information inside the log-files such as the date-time format, the antenna, the receiver and radome model often does not match the expected format or code.
- The need to process a scalable number of GNSS networks.
- The need to recognize new available potential interesting stations falling inside the region of interest.

This technical report describes the strategy used to reach these goals using Python, which is a cross-platform, open source and object-oriented programming language but, as explained in paragraph 3, actually it's launched by a bash script that makes it executable only on Linux. This software code has been developed on Ubuntu using the "PyCharm Community" smart editor which is a free, open source and lightweight IDE for Python & Scientific Development by JetBrains (downloadable from the link:
https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=linux&code=PCC)

To understand the kind of elaboration performed by the software, it's interesting to observe in Appendix A how a typical GNSS log file appears.

1. Strategy of data management

The "PyGLog" software is developed to manage the GNSS networks as a list of independent objects running in background as sub-processes, each of which with a different working directory, different configuration file and output directory. For this reason, here we will describe a single sub process associated to a network, which has the same principle used by all sub processes; Figure 1 shows the flowchart of the software in order to explain the sequence of operations.

As in the flowchart, the software loads the GNSS network configuration then according to the timetable, iterate through the loop. The available information are loaded into reference arrays, then the process compares the timestamp of the new station log-files from the web with the old one in the local repository: the new files belonging to the NetList (white list) are downloaded into a dedicated folder and are compared, field by field with the old one in order to find and report the differences. The files not in the NetList are considered as new "GPS station's log-files", they are downloaded directly into the repository but the process also checks for the coordinates of each GPS station, reporting when they fall inside a pre-defined area of interest. The most important part of the process is the elaboration to create the StationInfo files: to validate the data are performed multiple checks:

- The field names are interpreted according to the language, then the parameters are loaded into a structured array (defined as object).
- Date-time properly retrieved: the software is able to interpret different date-time formats, providing a large flexibility of interpretation.
- Period alignment check: reports when the date-time of installation are not consecutive in the paragraphs of the Antenna or the Receiver, avoiding or limiting possible errors which cause the process to stop.
- Check the GPS antenna, the Receiver and Antenna Radome codes: the software retrieves the codes from the log-file and looks for a match in the reference file list, which is the receiver/antenna file provided by IGS: *ftp://igs.org/pub/station/general/rcvr_ant.tab*. This file details naming conventions for IGS equipment descriptions in GNSS site logs, RINEX headers and SINEX. When no match is found the software uses a "Translation Table" which is an ASCII file where all the non-standard codes and the equivalent correct ones are saved. This guarantee that the StationInfo files will contain only IGS standard codes for the antennas, receivers and radomes. The Translation Table file is edited either by the software and by the operator (see par. 2.3.4) and helps in automatically handling persisting errors in log-files where non-IGS standard codes are used and where the networks managers do not apply the requested changes to the original log-files.

After the fields of the log files have been verified, if all the information are properly interpreted the software creates the Station-Info file which is a formatted ASCII file, containing the main information of the GPS station required by different geodetic software (mainly GAMIT/GLOBK, GIPSY and BERNESE), with

as many rows as the periods, ready to be processed. Figure 2 shows a typical GAMIT/GLOBK Station-Info file, where each column reports the following information:

- "*SITE": Station Id, the unique name used to identify the station
- "Station Name": the real name of the station, based on the location
- "Session Start": the beginning of a period, with the year, the Julian day and the time. A new period is defined every time a change occurs in any of the GPS parameters
- "Session End": the end of the period. When the period refers to the present, the end date is indicated with the year "9999"
- "Ant Ht", "HtCod", "Ant N" and "Ant E": antenna parameters
- "Receiver Type", "Vers", "SwVer" and "Receiver SN": receiver parameters
- "Antenna Type", "Dome" and "Antenna SN": Antenna and Duomo parameters

The flowchart in Figure 1 completes with service duties as deleting the old files, sending the report emails and waiting for the next timetable schedule.

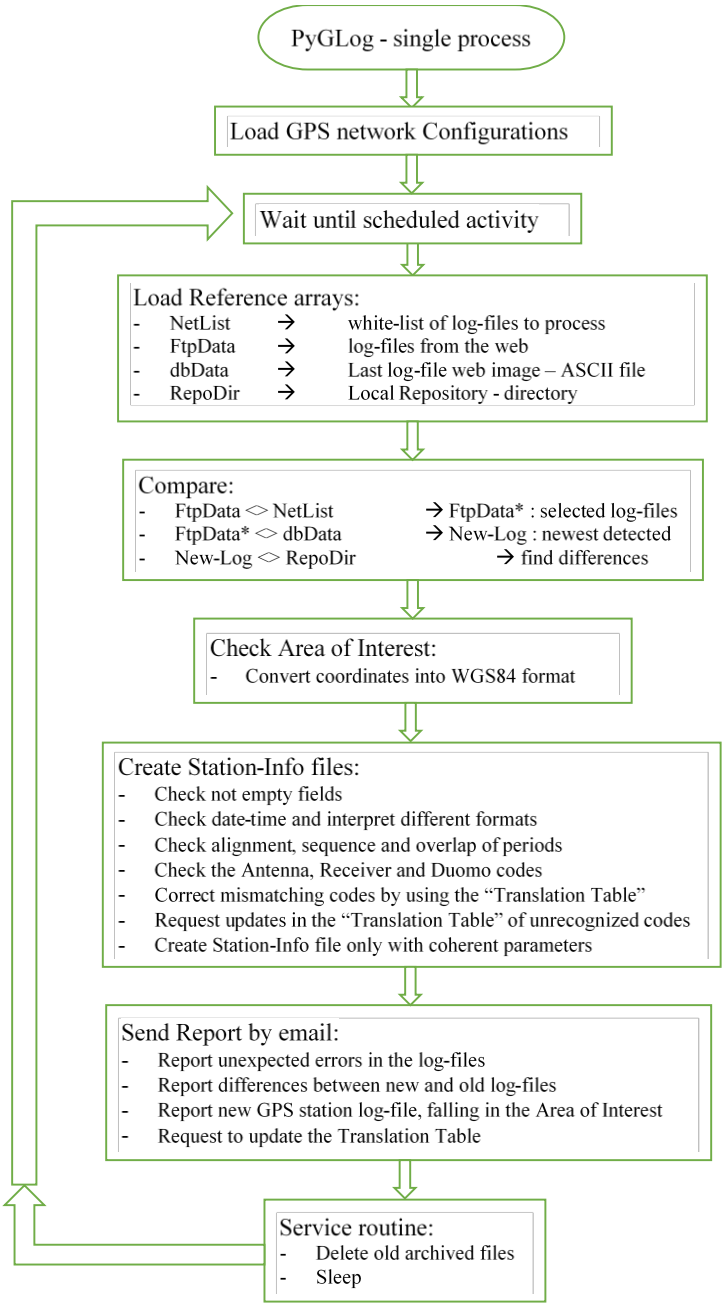
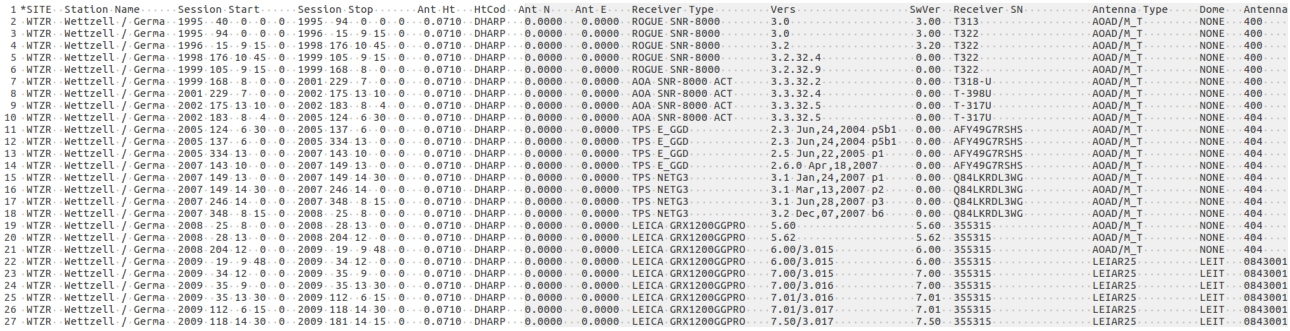


Figure 1. Flow chart of a single process to create the Station-Info files.



*SITE	Station Name	Session Start	Session Stop	Ant Ht	HtCod	Ant N	Ant E	Receiver Type	Vers	SwVer	Receiver SN	Antenna Type	Dome	Antenna
WTZR	Wettzell / Germa	1995 40 0 0	1995 94 0 0	0.0710	DHARP	0.0000	0.0000	ROGUE SNR-8000	3.0	3.00	T313	AOAD/M_T	NONE	400
WTZR	Wettzell / Germa	1995 94 0 0	1996 15 9 15	0.0710	DHARP	0.0000	0.0000	ROGUE SNR-8000	3.0	3.00	T322	AOAD/M_T	NONE	400
WTZR	Wettzell / Germa	1996 15 9 15	1998 176 10 45	0.0710	DHARP	0.0000	0.0000	ROGUE SNR-8000	3.2	3.20	T322	AOAD/M_T	NONE	400
WTZR	Wettzell / Germa	1998 176 10 45	1999 105 9 15	0.0710	DHARP	0.0000	0.0000	ROGUE SNR-8000	3.2.32.4	0.00	T322	AOAD/M_T	NONE	400
WTZR	Wettzell / Germa	1999 105 9 15	1999 168 8 0	0.0710	DHARP	0.0000	0.0000	ROGUE SNR-8000	3.2.32.9	0.00	T322	AOAD/M_T	NONE	400
WTZR	Wettzell / Germa	1999 168 8 0	2001 229 7 0	0.0710	DHARP	0.0000	0.0000	AOA SNR-8000 ACT	3.3.32.2	0.00	T318-U	AOAD/M_T	NONE	400
WTZR	Wettzell / Germa	2001 229 7 0	2002 175 13 10	0.0710	DHARP	0.0000	0.0000	AOA SNR-8000 ACT	3.3.32.4	0.00	T-398U	AOAD/M_T	NONE	400
WTZR	Wettzell / Germa	2002 175 13 10	2002 183 8 4 0	0.0710	DHARP	0.0000	0.0000	AOA SNR-8000 ACT	3.3.32.5	0.00	T-317U	AOAD/M_T	NONE	400
WTZR	Wettzell / Germa	2002 183 8 4 0	2005 124 6 30 0	0.0710	DHARP	0.0000	0.0000	AOA SNR-8000 ACT	3.3.32.5	0.00	T-317U	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2005 124 6 30 0	2005 137 6 0 0	0.0710	DHARP	0.0000	0.0000	TPS E_GGD	2.3 Jun,24,2004 p5b1	0.00	AFY49G7RSHS	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2005 137 6 0 0	2005 334 13 0 0	0.0710	DHARP	0.0000	0.0000	TPS E_GGD	2.3 Jun,24,2004 p5b1	0.00	AFY49G7RSHS	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2005 334 13 0 0	2007 143 10 0 0	0.0710	DHARP	0.0000	0.0000	TPS E_GGD	2.5 Jun,22,2005 p1	0.00	AFY49G7RSHS	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2007 143 10 0 0	2007 149 13 0 0	0.0710	DHARP	0.0000	0.0000	TPS E_GGD	2.6.0 Apr,18,2007	0.00	AFY49G7RSHS	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2007 149 13 0 0	2007 149 14 30 0	0.0710	DHARP	0.0000	0.0000	TPS NETG3	3.1 Jan,24,2007 p1	0.00	Q84LKRDL3WG	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2007 149 14 30	2007 246 14 0 0	0.0710	DHARP	0.0000	0.0000	TPS NETG3	3.1 Mar,13,2007 p2	0.00	Q84LKRDL3WG	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2007 246 14 0 0	2007 348 8 15 0	0.0710	DHARP	0.0000	0.0000	TPS NETG3	3.1 Jun,28,2007 p3	0.00	Q84LKRDL3WG	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2007 348 8 15 0	2008 25 8 0 0	0.0710	DHARP	0.0000	0.0000	TPS NETG3	3.2 Dec,07,2007 b6	0.00	Q84LKRDL3WG	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2008 25 8 0 0	2008 28 13 0 0	0.0710	DHARP	0.0000	0.0000	LEICA GRX1200GGPRO	5.60	5.60	355315	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2008 28 13 0 0	2008 204 12 0 0	0.0710	DHARP	0.0000	0.0000	LEICA GRX1200GGPRO	5.62	5.62	355315	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2008 204 12 0 0	2009 19 9 48 0	0.0710	DHARP	0.0000	0.0000	LEICA GRX1200GGPRO	6.00/3.015	6.00	355315	AOAD/M_T	NONE	404
WTZR	Wettzell / Germa	2009 19 9 48 0	2009 34 12 0 0	0.0710	DHARP	0.0000	0.0000	LEICA GRX1200GGPRO	6.00/3.015	6.00	355315	LEIAR25	LEIT	0843001
WTZR	Wettzell / Germa	2009 34 12 0 0	2009 35 9 0 0	0.0710	DHARP	0.0000	0.0000	LEICA GRX1200GGPRO	7.00/3.015	7.00	355315	LEIAR25	LEIT	0843001
WTZR	Wettzell / Germa	2009 35 9 0 0	2009 35 13 30 0	0.0710	DHARP	0.0000	0.0000	LEICA GRX1200GGPRO	7.00/3.016	7.00	355315	LEIAR25	LEIT	0843001
WTZR	Wettzell / Germa	2009 35 13 30 0	2009 112 6 15 0	0.0710	DHARP	0.0000	0.0000	LEICA GRX1200GGPRO	7.01/3.016	7.01	355315	LEIAR25	LEIT	0843001
WTZR	Wettzell / Germa	2009 112 6 15 0	2009 118 14 30 0	0.0710	DHARP	0.0000	0.0000	LEICA GRX1200GGPRO	7.01/3.017	7.01	355315	LEIAR25	LEIT	0843001
WTZR	Wettzell / Germa	2009 118 14 30 0	2009 181 14 15 0	0.0710	DHARP	0.0000	0.0000	LEICA GRX1200GGPRO	7.50/3.017	7.50	355315	LEIAR25	LEIT	0843001

Figure 2. Part of a station-info ASCII file.

2. Description of the software

2.1 Main software: a scalable multiprocessing structure

The main software has been optimized to run several simultaneous processes, working continuously in background as "daemon". This particular kind of process is detached from the terminal, does not need to interact with the operator, works with the minimum CPU load and is not killed by the OS when the terminal is closed, even though it's always listening the OS signals. At the beginning the main process loads from file the "common configurations", including the list of the GNSS networks to analyze, and creates the sub processes. Each sub-process loads from file the specific parameters, including the link to download the related log-file and, according to the scheduled time, performs the checks and comparisons described in the flowchart, updates the local repository and creates the station-info files. Appendix B and C show how the sub processes are created.

2.2 Folder structure

The main folder is named "CGPSmetadata" and the structure is shown below in a convenient sort:

- CGPSmetadata
  - SRC
  - Configurations
  - DBfile
  - TransTable
  - DatiNet
    - GPS "NETWORK"
      - _FtpNewLog
      - _Log_Repo
      - _STF_Repo
  - LogMail_OLD
  - virtualenvCGPS

The "SRC" folder contains the source Python code; the code is broken in different files with extension ".py". In the same folder the ASCII file "CGPSmetadata.cfg" is used to initialize the main software with the common parameter for all sub-processes. The "Configurations" folder stores the ASCII files with the configurations for each GNSS network. The "DBfile" folder stores the files to keep track of the last status of the GPS networks. The "TransTable" folder stores the files used by the software as interface with the operator. All the above configuration files are described in Section. 2.3.

The working folder "DatiNet" collects and groups in sub-folders all the output files produced by the network processes. Inside each network folder, in the "_FtpNewLog" sub-folder the new log-files are downloaded, the "_Log_Repo" folder is the log-file repository while the "_STF_Repo" is the target folder where the station-info files are created.

The "LogMail_OLD" folder keeps the log-files produced by all processes: these files are used to report errors and service informations and are sent by email to alert the user. This folder is also used as garbage-collector because it keeps also the old dismissed GPS log-files, which are currently deleted by the software after 30 days. In section 4. is described a typical log-file of the software.

The “virtualenvCGPS” folder is the virtualized python environment, with specific libraries used by the software: the virtual environment allows to install specific libraries, with no modification of the standard python installed at system level. Currently are installed the “pyproj” library to allow the coordinate conversion and the “requests” library to unzip files.

2.3 Configuration files

The operator generally interacts with this software by editing the configuration files, which are simple ASCII text files, used to set the global or the specific parameters of the GNSS networks. While running, each process initializes with these parameters and begins the sequence of operations leading to the creation of the StationInfo file. Following we provide an explanation of these files.

2.3.1 Common configuration file

This file includes the following parameters:

- The list of GNSS networks to be elaborated.
- The main working folder.
- The range of coordinates, Longitude and Latitude in WGS84 format, to verify whether the stations fall in the area of interest, to detect new available stations.
- The Boolean flag to enable the email alert: each process sends its log-file with the report of errors and information about the elaboration to create the StationInfo files.
- The scheduled day of the week and time to run the process.

Section 4 describes how to extract the information from the report log-file, to have a full picture of the elaboration. This configuration file is designed to give flexibility in entering parameters, the software recognizes the specific keywords even changing the field’s order. In the same file it is possible to comment a row, or part of it, by using the hash character “#”.

2.3.2 GPS network configuration file

The configuration files related to the specific GNSS network include parameters such as the link and credential to connect the web server, in order to download the GPS log-files. The software interprets either the FTP or HTTP connection based on the link and gives the possibility to set the Boolean flag “gsac” to download the station info files based on GSAC, the UNAVCO’s Geodesy Seamless Archive Centers software system, which powers geodesy data repositories with specific web services (https://www.unavco.org/software/data-management/gsac/gsac.html). Once the parameters are set, the software behaves in different ways based on the kind of connection: it is worth remembering that while the FTP servers allow to manage the files for a standard connection, the HTTP servers show a web page made by human and it is necessary to parse the html code in order to retrieve the file information (when available).

2.3.3 Previous status image of the GPS network.

In order to detect a new GNSS log-file from the web server, it is necessary to obtain the information about the timestamp of the last retrieved one in the log-file repository. In the “DBfile” folder the files containing the previous status image of each GPS network are stored. Comparing these information with the new status updated from the web, it is possible to detect the new files to download. Figure 3 shows an example of DBfile: each row represents a single GPS station of the network with the timestamp of the last download.

```
1 bacu.log ——→ Bacu.log ——→ 20170731174724→http://geo.edu.al/gps/logs_AlbGNSS/Bacu.log
2 bera.log ——→ Bera.log ——→ 20170731174724→http://geo.edu.al/gps/logs_AlbGNSS/Bera.log
3 hima.log ——→ Hima.log ——→ 20170731174724→http://geo.edu.al/gps/logs_AlbGNSS/Hima.log
4 lesk.log ——→ Lesk.log ——→ 20170731174724→http://geo.edu.al/gps/logs_AlbGNSS/Lesk.log
5 mali.log ——→ Mali.log ——→ 20170731174724→http://geo.edu.al/gps/logs_AlbGNSS/Mali.log
6 pesh.log ——→ Pesh.log ——→ 20170731174724→http://geo.edu.al/gps/logs_AlbGNSS/Pesh.log
7 sara.log ——→ Sara.log ——→ 20170731174724→http://geo.edu.al/gps/logs_AlbGNSS/Sara.log
8 shko.log ——→ SHKO.log ——→ 20170731174724→http://geo.edu.al/gps/logs_AlbGNSS/SHKO.log
9 tira.log ——→ Tira.log ——→ 20170731174724→http://geo.edu.al/gps/logs_AlbGNSS/Tira.log
```

Figure 3. Example of “_fDB” file, representing the last status image of the web server.

The following Table 1 shows the meaning of each column:

Column N°	Brief Description	Description
1	`File name (lower case)`	allows to univocal compare the file names, even though they are duplicated with different cases
2	`File name (original case)`	associated with the original filename, used to retrieve the file from the server
3	`Time stamp`	is compared with the correspondent one from the server. When the timestamp is not provided by the web server, then it is automatically set to the current time
4	`Link to http server`	link to download the html file

Table 1. list of information from the GPS station log-files.

In the DBfile these information are separated by a <TAB> character, represented by an "arrow" as drawn in the text editor in the figure.

2.3.4 The Translation Table

During the process to create the StationInfo file, the software checks for the coherence of each field in the log-file. In particular it attempts to verify the GPS Antenna, Receiver and Radome codes with respect to standard IGS codes, which are listed in the "rcvr_ant.tab" and daily updated from the following link: https://igscb.jpl.nasa.gov/igscb/station/general/rcvr_ant.tab. Additionally, the software uses the GAMIT/GLOBK specific files *"rcvant.dat" and "antmod.dat"* as reference tables. The file "rcvant.dat" includes all antennas and receivers supported by the software and the *"rcvr_ant.tab"* includes antenna phase center models. It is worth noting that equivalent files are available for other geodetic software (GIPSY and BERNESE). Checking whether the receiver, antenna or radome listed in a Log-file are supported by the processing software allows to avoid fatal errors in the processing; Importantly, the choice that if a specific antenna-radome couple is not found in the phase center model file, the software automatically sets the radome field to "NONE" in the StationInfo file, in order to not stall the elaboration.

Because the GNSS log-file is often edited by human operators, sometime the entries used do not match the values in the reference tables due to human errors; in this case the software searches inside the translation table, which is an ASCII file with three paragraphs:

- "ANTCODE" refers to the antenna model;
- "RCVCODE" refers to the GNSS receiver model;
- "DOMECODE" refers to the antenna radome model.

If the correct code is found, the software automatically uses it to elaborate the station-info file, otherwise it adds a new row in the proper paragraph, with the "fake" code in the first column and the "???" symbol in the second; the operator is then alerted by email to edit the translation table file and to indicate the correct code. Sometime it's useful to ignore the request for ambiguous radome model, in this case the operator adds the "#" char. All the translation table files are stored into the "TransTable" folder; the following Figure 4 shows a real case of translation table:

```
#Translation Table:
1) "???" = replace equivalent code
2) "???#" = ignore request, used for DUOCODE
#RCVCODE: format(FakeCode (tab) Translation) [20 char]
LEICA GRX1200GG PRO ---> LEICA GRX1200GGPRO
LEICA RS520 ---> LEICA SR52
TOPCON NET G3 ---> TPS NETG3
#END
---------

#ANTCODE: format(FakeCode (tab) Translation) [15 char]
LEICA  AR25 ---> LEIAR25
LEICA AS10 ---> LEIAS10
LEICA AT502 ---> LEIAT502
LEICA AT504 ---> LEIAT504
LEICA AT504GG ---> LEIAT504GG
LEICA AX1202GG ---> LEIAX1202GG
LEICAAT504GG ---> LEIAT504GG
TOPCON TPSCR.G3 ---> TPSCR.G3
TOPCONTPSCR.G3 ---> TPSCR.G3
TRIMBLE 29659.00 ---> TRM29659.00
TRIMBLE TRM29659.00 ---> TRM29659.00
TRIMBLE ZEPHYR GEODETIC - MODEL II (TRM 55971.00) ---> TRM55971.00
#END
---------

#DUOCODE: format(FakeCode (tab) Translation) [4 char]
AR25 ---> #???
AS10 ---> ???#
#END
```

Figure 4. Translation Table.

This strategy gives flexibility in the log-file elaboration, giving the autonomy to interpret human errors and allowing to generate correct (in the sense of avoiding software stops and fatal errors) station-info files.

3. Running the main software

The "pyGLog.py" software uses specific libraries, included in the associated virtual environment, which first needs to be activated. In order to facilitate the software execution, the shell script "start.sh" is created in the main folder. The main software iterates through the list of GPS networks and sequentially runs all sub-processes with the respective parameters. Below is the terminal with the running subprocesses, the parent pid (ppid) and the child process id (pid) are shown: it's interesting to notice that the sequence depends from the execution time, thus they are not sorted in the ascending order.

```
un@pc:/opt/CGPSmetadata$ ./start.sh
process  1 = GREF           ppid= 1672 pid= 7042
process  2 = SONEL          ppid= 1672 pid= 7045
process  3 = ERVA           ppid= 1672 pid= 7048
process  4 = ITACYL         ppid= 1672 pid= 7053
process  0 = EUREF          ppid= 1672 pid= 7039
process  6 = UNAVCO         ppid= 1672 pid= 7058
process  5 = CATNET         ppid= 1672 pid= 7055
process  8 = VENETO         ppid= 1672 pid= 7065
process  7 = RGP            ppid= 1672 pid= 7061
process  9 = STPOS          ppid= 1672 pid= 7068
process 11 = FREDNET        ppid= 1672 pid= 7073
process 10 = CAMPANIA       ppid= 1672 pid= 7072
```

```
process 15 = REP             ppid= 1672 pid= 7086
process 12 = FVG             ppid= 1672 pid= 7078
process 13 = GEODAF          ppid= 1672 pid= 7080
process 16 = IGNE            ppid= 1672 pid= 7088
process 17 = IGS             ppid= 1672 pid= 7092
process 18 = RENEP           ppid= 1672 pid= 7095
process 19 = RGAN            ppid= 1672 pid= 7098
process 14 = RING            ppid= 1672 pid= 7083
process 20 = GNSSPIEMONTE    ppid= 1672 pid= 7101
process 22 = SEGAL           ppid= 1672 pid= 7106
process 21 = NIGNET          ppid= 1672 pid= 7103
process 23 = ALBANIA         ppid= 1672 pid= 7109
process 24 = RAP             ppid= 1672 pid= 7112
```

To stop all the processes at once the shell script “stop.sh” was created in the main folder, which sends the kill message to each subprocess.

4. Extracting information from the Software LogFiles

While running each subprocess keeps track of useful information such as the server link, the run time, the number of log-files in the local repository and on the server side, the comparison between new and old stations log-files, the number of StationInfo files created and, importantly, gives information about errors during execution.

The log-files produced by the software of all the GPS networks are collected in the same “LogMail_OLD” folder and the filename includes the timestamp and network name, allowing to easily filter the files.

Using the Linux terminal commands from this folder with the appropriate “keywords”, it is possible to extract the main information from these files; follows a table with examples with the main usage of terminal commands:

`$ cat 2017-09-06_*.log`	Shows the logs produced by all the network in a specific date
`$ grep "STF files:" 2017-09-06*.log`	Shows the network name and the number of produced “Station-Info” files. Useful for diagnosis
`$ grep "AREA" 2017-09-06*.log`	Shows the network name and new potential GPS stations, falling in the area of interest
`$ grep "Compare"  2017-09-06_*.log`	Shows the network and GPS station name with changes in the metadata
`$ grep "not found" 2017-09*.log`	Shows the network and the number of GPS station not found
`$ grep "NO Log"  2017-09-06_*.log`	Shows empty logs
`$ grep "Error"  2017-09-06_*.log`	Shows generic Errors. Useful for diagnosis
`$ grep "Update TransTable" 2017-09*.log`	Shows which Translation Tables need updates

Starting from these information it is easy and fast to investigate the status of the processed networks.

Considerations

This software is currently running on the INGV-Bologna GPS data processing server and the StationInfo output files have been tested for errors with the GAMIT/GLOCK software, passing with success. The software is still under development and even though some change could occur on the procedures, the given scalable structure allows to keep the choices at the base of the project. The strength of this code is that the information coming from different sources, are grouped and converted in the same type of structured arrays, which allow to use generalized procedures for elaboration. The idea to validate the data through reference tables and translation tables gives flexibility, allowing to process the data even in presence of human errors in the GPS station log-file.

Future developments

Developing this software and facing different issues, has suggested the following considerations to improve the usage of this work:

- It's possible to add a procedure in the parent process which oversees all the child processes and alerts when a sub process stops or gives error and does not produces the expected StationInfo files.
- The usage of this software could be extended to other OS, as the same Python philosophy suggests.

Conclusions

The software turned out in a valid support to check a massive amount of GPS metadata, to quickly convert in station info files and to further elaborate the GPS data avoiding to stall the analysis. This translates into an advantage for the researcher who can devote himself to other more important aspects of his work.

Thanks

This work has been possible thanks to the close collaboration with the geodesy group operating in Bologna, which saw the possibility of optimizing its work by entrusting routine work to a process that, though quick and automated, is possible to be verified. Thanks especially to the colleagues for providing the specifications of implementation and supporting me along the entire software development path.

Sitography

https://www.python.org/
https://stackoverflow.com/
https://github.com/serverdensity/python-daemon/blob/master/daemon.py
https://github.com/
https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=linux&code=PCC
http://www.jejik.com/articles/2007/02/a_simple_unix_linux_daemon_in_python/
http://www.erlenstar.demon.co.uk/unix/faq_2.html#SEC16
https://www.unavco.org/software/data-management/gsac/gsac.html

Appendix A: structure of a GNSS log file

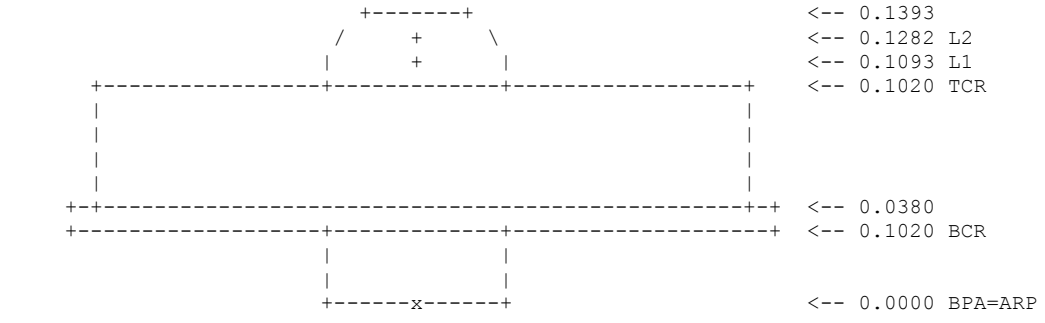
Follows an example of a typical well-structured GNSS log file, where the structure in sub-paragraphs highlights the historical parameter changes:

```
     ACOR Site Information Form (site log)
     International GNSS Service
     See Instructions at:
       ftp://igscb.jpl.nasa.gov/pub/station/general/sitelog_instr.txt

0.   Form

     Prepared by (full name)  : Pedro Gonzalo Lopez
     Date Prepared            : 2014-09-15
     Report Type              : UPDATE
     If Update:
      Previous Site Log       : acor_20121212.log
      Modified/Added Sections : 11

1.   Site Identification of the GNSS Monument

     Site Name                : A Coruna
     Four Character ID        : ACOR
     Monument Inscription     : None
     IERS DOMES Number        : 13434M001
     CDP Number               : 2101
     Monument Description     : STEEL MAST
       Height of the Monument : 3 m
       Monument Foundation    : CONCRETE BLOCK
       Foundation Depth       : 2 m
     Marker Description       : BRASS PLATE
     Date Installed           : 1998-03-06T10:10Z
     Geologic Characteristic  : CONGLOMERATE
       Bedrock Type           : IGNEOUS
       Bedrock Condition      : WEATHERED
       Fracture Spacing       : (0 cm/1-10 cm/11-50 cm/51-200 cm/over 200 cm)
       Fault zones nearby     : NO
         Distance/activity    : (multiple lines)
     Additional Information   : Station located in the tide gauge of
                              : A Coruna


2.   Site Location Information

     City or Town             : A Coruna
     State or Province        : A Coruna
     Country                  : Spain
     Tectonic Plate           : EURASIAN
     Approximate Position (ITRF)
       X coordinate (m)       : 4594489.939
       Y coordinate (m)       : -678368.073
       Z coordinate (m)       : 4357065.900
       Latitude (N is +)      : +432151.77
       Longitude (E is +)     : -0082356.17
       Elevation (m,ellips.)  :    67.0
     Additional Information   : ACOR is a reference point of the Spanish
                              : GPS Fiducial Network raised by the
                              : Instituto Geografico Nacional

3.   GNSS Receiver Information

3.1  Receiver Type            : ASHTECH UZ-12
     Satellite System         : GPS
     Serial Number            : 00224
     Firmware Version         : UE00-0A12
     Elevation Cutoff Setting : 0 deg
     Date Installed           : 1998-12-06T10:10Z
     Date Removed             : 2001-12-19
     Temperature Stabiliz.    : (deg C) +/- (deg C)
     Additional Information   : (multiple lines)

3.x  Receiver Type            : (A20, from rcvr_ant.tab; see instructions)
     Satellite System         : (GPS+GLO+GAL+BDS+QZSS+SBAS)
     Serial Number            : (A20, but note the last A5 is used in SINEX)
     Firmware Version         : (A11)
     Elevation Cutoff Setting : (deg)
     Date Installed           : (CCYY-MM-DDThh:mmZ)
```

```
     Date Removed             : (CCYY-MM-DDThh:mmZ)
     Temperature Stabiliz.    : (none or tolerance in degrees C)
     Additional Information   : (multiple lines)


4.   GNSS Antenna Information

4.1  Antenna Type             : ASH700936D_M    SNOW
     Serial Number            : 16122
     Antenna Reference Point  : BPA
     Marker->ARP Up Ecc. (m)  :   3.0420
     Marker->ARP North Ecc(m) :   0.0000
     Marker->ARP East Ecc(m)  :   0.0000
     Alignment from True N    : 0 deg
     Antenna Radome Type      : SNOW
     Radome Serial Number     :
     Antenna Cable Type       : (vendor & type number)
     Antenna Cable Length     : 30 m
     Date Installed           : 1998-12-06T10:10Z
     Date Removed             : 2007-03-17T23:59Z
     Additional Information   : New antenna denomination; no physical
                              : change

4.x  Antenna Type             : (A20, from rcvr_ant.tab; see instructions)
     Serial Number            : (A*, but note the last A5 is used in SINEX)
     Antenna Reference Point  : (BPA/BCR/XXX from "antenna.gra"; see instr.)
     Marker->ARP Up Ecc. (m)  : (F8.4)
     Marker->ARP North Ecc(m) : (F8.4)
     Marker->ARP East Ecc(m)  : (F8.4)
     Alignment from True N    : (deg; + is clockwise/east)
     Antenna Radome Type      : (A4 from rcvr_ant.tab; see instructions)
     Radome Serial Number     :
     Antenna Cable Type       : (vendor & type number)
     Antenna Cable Length     : (m)
     Date Installed           : (CCYY-MM-DDThh:mmZ)
     Date Removed             : (CCYY-MM-DDThh:mmZ)
     Additional Information   : (multiple lines)

5.   Surveyed Local Ties

5.x  Tied Marker Name         :
     Tied Marker Usage        : (SLR/VLBI/LOCAL CONTROL/FOOTPRINT/etc)
     Tied Marker CDP Number   : (A4)
     Tied Marker DOMES Number : (A9)
     Differential Components from GNSS Marker to the tied monument (ITRS)
       dx (m)                 : (m)
       dy (m)                 : (m)
       dz (m)                 : (m)
     Accuracy (mm)            : (mm)
     Survey method            : (GPS CAMPAIGN/TRILATERATION/TRIANGULATION/etc)
     Date Measured            : (CCYY-MM-DDThh:mmZ)
     Additional Information   : (multiple lines)

6.   Frequency Standard

6.1  Standard Type            : INTERNAL
       Input Frequency        : (if external)
       Effective Dates        : 1998-12-06/CCYY-MM-DD
       Notes                  : (multiple lines)

6.x  Standard Type            : (INTERNAL or EXTERNAL H-MASER/CESIUM/etc)
       Input Frequency        : (if external)
       Effective Dates        : (CCYY-MM-DD/CCYY-MM-DD)
       Notes                  : (multiple lines)


7.   Collocation Information

7.1  Instrumentation Type     : TIDE GAUGE
       Status                 : PERMANENT
       Effective Dates        : 1955-01-15/CCYY-MM-DD
       Notes                  : Sensor Model   : THALIMEDES
                              : Manufacturer   : AOTT
                              : Data Frequency : 10 min
                              : Accuracy (mm)  : 1 mm

7.x  Instrumentation Type     : (GPS/GLONASS/DORIS/PRARE/SLR/VLBI/TIME/etc)
       Status                 : (PERMANENT/MOBILE)
```

```
       Effective Dates        : (CCYY-MM-DD/CCYY-MM-DD)
       Notes                  : (multiple lines)


8.   Meteorological Instrumentation

8.1.x  Humidity Sensor Model  :
       Manufacturer           :
       Serial Number          :
       Data Sampling Interval : (sec)
       Accuracy (% rel h)     : (% rel h)
       Aspiration             : (UNASPIRATED/NATURAL/FAN/etc)
       Height Diff to Ant     : (m)
       Calibration date       : (CCYY-MM-DD)
       Effective Dates        : (CCYY-MM-DD/CCYY-MM-DD)
       Notes                  : (multiple lines)

8.2.x  Pressure Sensor Model  :
       Manufacturer           :
       Serial Number          :
       Data Sampling Interval : (sec)
       Accuracy               : (hPa)
       Height Diff to Ant     : (m)
       Calibration date       : (CCYY-MM-DD)
       Effective Dates        : (CCYY-MM-DD/CCYY-MM-DD)
       Notes                  : (multiple lines)

8.3.x  Temp. Sensor Model     :
       Manufacturer           :
       Serial Number          :
       Data Sampling Interval : (sec)
       Accuracy               : (deg C)
       Aspiration             : (UNASPIRATED/NATURAL/FAN/etc)
       Height Diff to Ant     : (m)
       Calibration date       : (CCYY-MM-DD)
       Effective Dates        : (CCYY-MM-DD/CCYY-MM-DD)
       Notes                  : (multiple lines)

8.4.x  Water Vapor Radiometer :
       Manufacturer           :
       Serial Number          :
       Distance to Antenna    : (m)
       Height Diff to Ant     : (m)
       Calibration date       : (CCYY-MM-DD)
       Effective Dates        : (CCYY-MM-DD/CCYY-MM-DD)
       Notes                  : (multiple lines)

8.5.x  Other Instrumentation  : (multiple lines)


9.   Local Ongoing Conditions Possibly Affecting Computed Position

9.1.x  Radio Interferences    : (TV/CELL PHONE ANTENNA/RADAR/etc)
       Observed Degradations  : (SN RATIO/DATA GAPS/etc)
       Effective Dates        : (CCYY-MM-DD/CCYY-MM-DD)
       Additional Information : (multiple lines)

9.2.x  Multipath Sources      : (METAL ROOF/DOME/VLBI ANTENNA/etc)
       Effective Dates        : (CCYY-MM-DD/CCYY-MM-DD)
       Additional Information : (multiple lines)

9.3.x  Signal Obstructions    : (TREES/BUILDINGS/etc)
       Effective Dates        : (CCYY-MM-DD/CCYY-MM-DD)
       Additional Information : (multiple lines)


10.  Local Episodic Effects Possibly Affecting Data Quality

10.x  Date                    : (CCYY-MM-DD/CCYY-MM-DD)
      Event                   : (TREE CLEARING/CONSTRUCTION/etc)


11.  On-Site, Point of Contact Agency Information

     Agency                   : Instituto Geografico Nacional
     Preferred Abbreviation   : IGN-E
     Mailing Address          : C/ General Ibanez Ibero, 3
                              : E-28003 MADRID
```

```
                              : SPAIN
     Primary Contact
       Contact Name           : Pedro Gonzalo Lopez
       Telephone (primary)    : +34 91 597 9625
       Telephone (secondary)  : +34 91 597 9562
       Fax                    : +34 91 597 9616
       E-mail                 : pgonzalo@fomento.es
     Secondary Contact
       Contact Name           : Jose A. Sanchez Sobrino
       Telephone (primary)    : +34 91 597 9428
       Telephone (secondary)  : +34 91 597 9562
       Fax                    : +34 91 597 9616
       E-mail                 : jassobrino@fomento.es
     Additional Information   : (multiple lines)

12.  Responsible Agency (if different from 11.)

     Agency                   : (multiple lines)
     Preferred Abbreviation   : (A10)
     Mailing Address          : (multiple lines)
     Primary Contact
       Contact Name           :
       Telephone (primary)    :
       Telephone (secondary)  :
       Fax                    :
       E-mail                 :
     Secondary Contact
       Contact Name           :
       Telephone (primary)    :
       Telephone (secondary)  :
       Fax                    :
       E-mail                 :
     Additional Information   : (multiple lines)

13.  More Information

     Primary Data Center      : BKGE
     Secondary Data Center    : OLG
     URL for More Information : http://www.epncb.oma.be/info/ACOR.html
     Hardcopy on File
       Site Map               : (Y or URL)
       Site Diagram           : (Y or URL)
       Horizon Mask           : (Y or URL)
       Monument Description   : (Y or URL)
       Site Pictures          : (Y or URL)
     Additional Information   : (multiple lines)
     Antenna Graphics with Dimensions

LEIAT504
```




```
   ARP: Antenna Reference Point
   L1 : L1 Phase Center                  L2 : L2 Phase Center
   TCR: Top of Chokering                 BCR: Bottom of Chokering
```

Appendix B: the main python code to run the processes and daemons

The strength of this software is its ability to run several independent processes, using a set of common parameters which, together with the specific one, lead to the data elaboration. Here is shown the structure of the main code that could be useful for further applications.

As shown below the first process is named “CreaProcessi(Avvio)”: it initializes with the common configurations and then iterates through the list of networks to run different processes. Each process creates a daemon by running the procedure “CreaDemone(nNET, CGPSconf, Avvio)”, which uses the “Daemon” class to fork the parent process. The new created process overrides the “run(self)” procedure in the class “daemEPOS(Daemon)”, which performs all the elaboration as child process, with the specific parameters retrieved by the number “nNET” given by the “for loop” iteration. Follows a schema of the structure of the source code:

```
#-----------------------------------------------------
if __name__ == "__main__":
if len(sys.argv) == 2:
    if 'start' == sys.argv[1]:
        CreaProcessi(True)
    elif 'stop' == sys.argv[1]:
        CreaProcessi(False)
#---------------------------
    def CreaProcessi(Avvio):                          # Create processes associated to daemons
        CGPSconf = CGPSconfigurazioni()               # Initialization of Common Configuration
        for nNET, NET in enumerate(CGPSconf.NETar):   # Iterate to create the processes
            p = Process(target=CreaDemone, args=(nNET, CGPSconf, Avvio))     # Each process
            p.start()                                                        # runs a daemon
            p.join()

    def CreaDemone(nNET, CGPSconf, Avvio):            # Daemon creation
        daemon = daemEPOS(nNET, CGPSconf)
#---------------------------------------------------------------------
class daemEPOS(Daemon):                         # The class “Daemon” is shown in Appendix “C”
    def __init__(self, nNET, CGPSconf):         # it creates processes using common parameters
        self.nNET = nNET
        self.CGPSconf = CGPSconf
        self.pidfile = '/tmp/daemon_' + self.NET + '.pid'

    def run(self):                                    # procedure “Overridden” by the daemon
        [Initialization]
        while True:                                   # Loop to run according the time table
            [elaboration]                             # Procedure to check GPS Log-Files and
                                                      # to create station-info files
```

Appendix C: procedure to fork the parent process into a child process

In order to “demonize” a process the main class uses a second class named "Daemon" included in the "daemon.py" file: this code is responsible for twinning the parent process into child processes on which to run the "run(self)" procedure; this procedure is then overridden in the main class, where it runs in background creating the station-info files. The code used to create daemons was inspired from internet documentations, follows a portion of the code:

```
#!/usr/bin/env python
# http://www.jejik.com/articles/2007/02/a_simple_unix_linux_daemon_in_python/
# http://www.erlenstar.demon.co.uk/unix/faq_2.html#SEC16

import sys, os, time, atexit
from signal import SIGTERM

class Daemon:
# A generic daemon class.
# Usage: subclass the Daemon class and override the run() method
#-----------------------------
   def __init__(self, nNET, CGPSconf):
       pass    #   take the parameters from class daemEPOS(Daemon):

   def daemonize(self):
      try:
           pid = os.fork()
           if pid > 0:
               # exit first parent
               sys.exit(0)
       except OSError, e:
           sys.stderr.write("fork #1 failed: %d (%s)\n" % (e.errno, e.strerror))
           sys.exit(1)
       # decouple from parent environment
       os.setsid()
       os.umask(0)

       # do second fork
       try:
           pid = os.fork()
           if pid > 0:
               # exit from second parent
               sys.exit(0)
       except OSError, e:
           sys.stderr.write("fork #2 failed: %d (%s)\n" % (e.errno, e.strerror))
           sys.exit(1)
       #----------------------------------------
       # write pidfile
       atexit.register(self.delpid)
       pid = str(os.getpid())
       try:
           file(self.pidfile, 'w+').write("%s\t%s\n" %(pid, self.NET))
       except IOError, e:
           print ('Errore pidfile: %d (%s)' % (IOError,e))

   def delpid(self):
       os.remove(self.pidfile)

   def start(self):                        # Start the daemon
       # Check for a pidfile to see if the daemon already runs
       try:
           pf = file(self.pidfile, 'r')
           pid = int(pf.read().split('\t')[0])
           pf.close()
       except IOError:
           pid = None

       if pid:
           message = "pidfile %s already exist. Daemon already running.\n"
```

```
            sys.stderr.write(message % self.pidfile)
            sys.exit(1)

        # Start the daemon
        self.daemonize()
        self.run()

    def stop(self):                         # Stop the daemon
        # Get the pid from the pidfile
        try:
            pf = file(self.pidfile, 'r')
            pid = int(pf.read().split('\t')[0])
            pf.close()
        except IOError:
            pid = None

        if not pid:
            message = "pidfile %s does not exist. Daemon not running.\n"
            sys.stderr.write(message % self.pidfile)
            return  # not an error in a restart

        # Try killing the daemon process
        try:
            while 1:
                os.kill(pid, SIGTERM)
                time.sleep(0.1)
        except OSError, err:
            err = str(err)
            if err.find("No such process") > 0:
                if os.path.exists(self.pidfile):
                    os.remove(self.pidfile)
            else:
                print str(err)
                sys.exit(1)

    def restart(self):              # Restart the daemon
        self.stop()
        self.start()

    def run(self):
        # override this method when you subclass Daemon.
        # It will be called after the process has been daemonized by start() or
restart().
```

ISSN 1590-2595

Quaderni di Geofisica

http://istituto.ingv.it/it/le-collane-editoriali-ingv/quaderni-di-geofisica.html

I Quaderni di Geofisica coprono tutti i campi disciplinari sviluppati all'interno dell'INGV, dando particolare risalto alla pubblicazione di dati, misure, osservazioni e loro elaborazioni anche preliminari, che per tipologia e dettaglio necessitano di una rapida diffusione nella comunità scientifica nazionale ed internazionale. La pubblicazione on-line fornisce accesso immediato a tutti i possibili utenti. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-7941

rapporti tecnici INGV

http://istituto.ingv.it/it/le-collane-editoriali-ingv/rapporti-tecnici-ingv.html

I Rapporti Tecnici INGV pubblicano contributi, sia in italiano che in inglese, di tipo tecnologico e di rilevante interesse tecnico-scientifico per gli ambiti disciplinari propri dell'INGV. La collana Rapporti Tecnici INGV pubblica esclusivamente on-line per garantire agli autori rapidità di diffusione e agli utenti accesso immediato ai dati pubblicati. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-6651

Miscellanea INGV

http://istituto.ingv.it/it/le-collane-editoriali-ingv/miscellanea-ingv.html

La collana Miscellanea INGV nasce con l'intento di favorire la pubblicazione di contributi scientifici riguardanti le attività svolte dall'INGV (sismologia, vulcanologia, geologia, geomagnetismo, geochimica, aeronomia e innovazione tecnologica). In particolare, la collana Miscellanea INGV raccoglie reports di progetti scientifici, proceedings di convegni, manuali, monografie di rilevante interesse, raccolte di articoli, ecc.

Coordinamento editoriale e impaginazione
Centro Editoriale Nazionale | INGV

Progetto grafico e redazionale
Daniela Riposati | Laboratorio Grafica e Immagini | INGV

© 2018 INGV Istituto Nazionale di Geofisica e Vulcanologia
Via di Vigna Murata, 605
00143 Roma
Tel. +39 06518601 Fax +39 065041181

http://www.ingv.it

Istituto Nazionale di Geofisica e Vulcanologia