**CentOS 操作系统安装 Bernese**

[2018-07-09](http://www.gnss.help/2018/07/09/centos-install-bernese/)

[BERNESE](http://www.gnss.help/categories/Bernese/)

[BERNESE安装](http://www.gnss.help/tags/Bernese%E5%AE%89%E8%A3%85/)

[之前的文章](http://www.gnss.help/2018/07/02/ubuntu-install-bernese/)介绍过了 [Bernese](http://www.bernese.unibe.ch/) 软件在 Ubuntu 操作系统下的安装过程。考虑到 [CentOS](https://www.centos.org/) 是在服务器和工作站领域用户量非常大的 Linux 发行版，并且其依赖环境的配置过程与 Ubuntu OS 有较多的不同，因此本文介绍在 CentOS 下安装 Bernese 软件的过程。

我实际使用的操作系统为 CentOS 7，Bernese 软件版本为 5.2。鉴于 [RHEL（Red Hat Enterprise Linux）](https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux)与 CentOS 有非常密切的关系，因此该安装过程同样适用于对应的 RHEL 7 版本。

**软件包内容**

拿到 Bernese 软件的安装包后，你应该首先检查安装文件是否齐全，一份完整的软件包至少包含以下文件：

* BERN52.tgz：主程序包；
* CAMPAIGN52.tgz：示例工程；
* DATAPOOL.tgz：输入数据；
* ICONS.tgz：程序图标；
* README：程序说明文档；
* SAVEDISK.tgz：输出数据；
* setup.sh：程序安装脚本。

**配置环境依赖**

Bernese 软件使用 Fortran 90、基于 [Qt](https://www.qt.io/) 的 C++、Perl 5 以及 Shell 等语言编写。使用该程序需要安装相应的编译器和解释器。要成功安装 Bernese，必须先安装以下程序：

* C++ 编译器；
* Fortran 90 编译器;
* Perl 5；
* Qt 4；
* 其他依赖库。

CentOS 7 已经自带了 Perl 5，因此还需要安装 C++ 和 Fortran 语言编译器、Qt 4 和依赖环境。

**安装依赖库**

安装 Qt 4 和 Bernese 软件需要使用 X11 和 zlib 两个重要的库。在终端中执行如下命令安装 X11 和 zlib：

|  |  |
| --- | --- |
| 1  2  3  4 | $ sudo yum install libX11-devel  $ sudo yum install libXext-devel  $ sudo yum install libXtst-devel  $ sudo yum install zlib-devel |

**安装 GNU-C++**

取决于你选择的操作系统安装方式，CentOS 7 可能不会自带 GNU-C++ 编译器，为保证 Bernese 能够安装成功，使用下面的命令安装它：

|  |  |
| --- | --- |
| 1 | $ sudo yum install gcc-c++ |

**安装 GNU-Fortran**

Fortran 语言编译器用于编译 Bernese 软件中使用 Fortran 90 开发的主程序。在终端中执行以下命令安装 GNU-Fortran（gfortran）：

|  |  |
| --- | --- |
| 1 | $ sudo yum install gcc-gfortran |

**安装 Qt 4**

Bernese 软件的程序主窗口采用 Qt 4 编写，编译 Bernese 菜单模块需要 Qt 4。从 Qt 4 的下载页面，下载 [Qt 4 的源代码](http://download.qt.io/archive/qt/4.8/4.8.7/qt-everywhere-opensource-src-4.8.7.tar.gz)进行编译安装。以 qt-everywhere-opensource-src-4.8.7.tar.gz 为例，首先解压源代码文件：

|  |  |
| --- | --- |
| 1 | $ tar -zxvf qt-everywhere-opensource-src-4.8.7.tar.gz |

使用 cd 命令进入解压得到的文件夹，依次执行如下命令将 Qt 4 安装到 /usr/local/qt4 目录内：

|  |  |
| --- | --- |
| 1  2  3 | $ sudo ./configure -release -opensource -static -prefix /usr/local/qt4  $ sudo make  $ sudo make install |

待 Qt 4 编译完成后，还需要为其配置环境变量 QTDIR，以指向 Qt 4 的安装目录。环境变量根据实际使用的 Shell 有所不同，对于 Bash 的用户，使用如下方式设置：

|  |  |
| --- | --- |
| 1 | $ export QTDIR="/usr/local/qt4" |

C Shell 或 TC Shell 用户，请使用以下命令设置：

|  |  |
| --- | --- |
| 1 | $ setenv QTDIR "/usr/local/qt4" |

至此，依赖环境已经配置完毕，可以开始安装 Bernese 了。

**安装 Bernese**

Bernese 为用户提供了使用方便的安装脚本 setup.sh。该脚本默认安装将 Bernese 安装到用户的 Home 文件夹，但也可以安装到其他文件夹下。如果你希望跟计算机上的其他用户也能使用 Bernese，需要共享 BERN52 和 MENU.INP 的执行和修改权限。完整安装 Bernese 需要约 500 MB 的硬盘空间。我习惯将外部编译的大型程序安装至 /opt 文件夹内，本文即以 /opt 作为安装目录示例。

使用 cd 命令进入 Bernese 源码文件夹，使用 root 用户执行其中的安装设置脚本：

|  |  |
| --- | --- |
| 1 | $ sh setup.sh |

之后，安装脚本提示输入安装目录：

Full path where the BERN52 software tree will be installed  
[ /root ]:

这里的 /root 表示默认安装路径。我希望将其安装到 /opt/BERN52，因此输入 /opt。随后脚本打印出找到的 Perl 解释器的位置：

Perl program to be used [ /usr/bin/perl ]:

使用系统自带的 Perl 解释器就足够了，此处直接回车。之后，设置脚本 configure.pm 将被启动，并打印出以下提示：

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝  
CONFIGURATION OF THE BERNESE GNSS SOFTWARE  
＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝  
0 … Complete installation (Steps 1 to 5)  
1 … Update LOADGPS.setvar  
2 … Install online updates  
3 … Add a new user environment  
4 … Compile the menu  
5 … Compile the programs  
6 … Install the Example campaign  
7 …　－－－

x … Exit

此处第 1 项为更新环境变量设置，第 2 项为安装在线更新包，第 3 项为添加新用户环境，第 4 项为编译图形界面菜单，第 5 项为编辑程序，第 6 项为安装解算工程示例；而第 0 项为完整安装，即执行从第 1 项至第 5 项的全部过程；键入 x 为退出安装程序。这里当然要选择完整安装，键入 0。

程序继续打印出 Bernese 软件支持的编译器，其中 GNU 一项指示到 gfortran 的路径，表明程序找到的了该编译器：

Select Compiler for OS LINUX:  
1: G95  
2: GNU -> /bin/gfortran  
3: IFC  
4: PG\_F90  
5: SUNF90  
6: OTHER

这里系统已经检测到了 GNU Fortran。选择编译器，这里使用我之前安装的 gfortran。输入 2。随后将打印出当前应用的环境变量：

Current Values:  
－－－－－－－  
VARIABLE DESCRIPTION 　 　 　VARIABLE NAME 　 　　　　VARIABLE VALUE  
1: Path to the software　　　　　　C 　 　　　　　　　　　=>　/opt/BERN52  
2: Path to Qt lib for Bernese 　　　 QTBERN　　　　　　　=>　/usr/local/qt\_4.7.4  
3: Operating system group　　　　OS 　 　 　 　　　　　　=>　UNIX  
4: Name of the operating system 　 OS\_NAME 　　　　　　=>　LINUX  
5: Fortran compiler name　　　 　 F\_VERS 　　　　　　　=>　 GNU  
6: List of additional compilers　 　 F\_VERS\_LIST　 　　　 =>  
7: Host of the BPE server　　 　 　 BPE\_SERVER\_HOST 　=>　myComputer  
8: Path to user environment 　　 　U 　 　 　　　　　　 　 =>　${HOME}/GPSUSER52  
9: Path to temp. user environment　T 　 　 　 　　　　　　=>　${HOME}/GPSTEMP  
10: Path to campaign directory 　 　P 　 　 　　　　　 　　 =>　${HOME}/GPSDATA/CAMPAIGN52  
11: Path to datapool area　　　 　　D 　 　 　 　　　　　　=>　${HOME}/GPSDATA/DATAPOOL  
12: Path to savedisk area　　　　 　S 　 　 　 　　　　 　　=>　${HOME}/GPSDATA/SAVEDISK

Accept the values (y/n):

这里的第 1 项为 Bernese 软件的安装目录，第 2 项为所使用的 Qt 4 所在目录，第 3 项为操作系统类型，第 4 项为操作系统名称，第 5 项为使用的 Fortran 编译器，第 6 项为其他可用的编译器，第 7 项为 BPE 服务器，第 8 项为用户环境目录，第 9 项为用户临时文件缓存目录，第 10 项为用户解算工程文件夹，第 11 项为用户数据文件夹目录，第 12 项为解算成果目录。若接受其默认的变量设置，输入 y。要修改其中的配置，输入 n，然后根据其提示，输入变量编号来配置变量。这里使用默认设置，输入 y。

随后安装程序将检测是否存在所需的文件夹，当提示某个文件夹不存在时，程序提示是否创建，键入 y 创建即可。之后将开始 Bernese 的安装过程，泡杯茶等待程序编译完成。

编译成功后提示：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* Bernese menu compiled successfully.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

和

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* Fortran programs compiled successfully.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

如果你看到了以上两个提示，那么恭喜你：Bernese 软件已经编译成功了！

**添加用户**

使用 root 用户进行日常操作既不方便也不安全。为了让非 root 用户也能使用 Bernese 软件，需要使用新用户加载环境变量设置文件，再启动配置程序。在终端中执行如下命令：

|  |  |
| --- | --- |
| 1  2 | $ source /opt/BERN52/GPS/EXE/LOADGPS.setvar  $ perl /opt/BERN52/GPS/EXE/configure.pm |

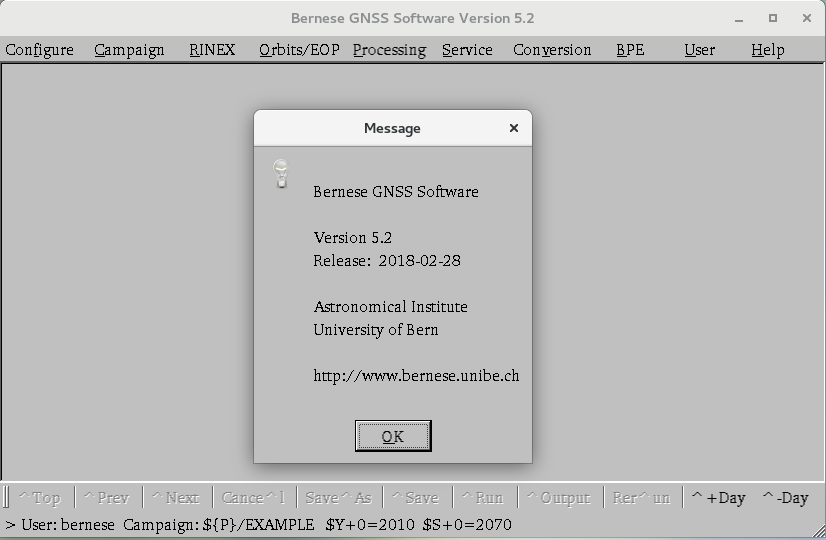
程序将再次打印出安装提示，此时键入 3，根据提示添加新用户的环境变量和 Bernese 运行所需的文件夹即可。

**配置环境变量**

为了在每次开机后都能直接使用 Bernese 软件，需要编辑 Shell 配置文件，让系统在启动时自动加载环境变量。Bash 用户打开 ~/.bashrc，C Shell 或 TC Shell 用户打开 ~/.cshrc，为其添加以下内容：

|  |  |
| --- | --- |
| 1  2 | # Load environment variables for Bernese GNSS Software  source /opt/BERN52/GPS/EXE/LOADGPS.setvar |

之后将可以启动 Bernese 了，你可以在终端中键入 G 试一下。若程序安装成功，将弹出类似下面这样的 Bernese 软件窗口：

[](http://www.gnss.help/images/bernese-centos.png)Bernese GNSS Software

**Ubuntu 操作系统安装 Bernese**

[2018-07-02](http://www.gnss.help/2018/07/02/ubuntu-install-bernese/)

[BERNESE](http://www.gnss.help/categories/Bernese/)

[BERNESE](http://www.gnss.help/tags/Bernese/)

[Bernese software](http://www.bernese.unibe.ch/) 是另一个著名的高精度 GNSS 数据处理软件。由瑞士伯尔尼大学（University of Berne）天文研究所开发，能够处理包括 GPS、GLONASS 和卫星激光测距（SLR）等多种观测数据。该程序适用于 Windows 或 UNIX/Linux 操作系统，内嵌图形界面（虽然比较丑），操作友好，模型准确，解算精度高，并且具有强大的数据批处理能力。

当前 Bernese 软件可以购买的最新版本为 5.2，获得授权后，你可以使用本文介绍的方法将其安装到运行 Linux 操作系统的计算机上。我实际使用的 Linux 发行版为 [elementary OS](https://elementary.io/zh_CN/) Loki 0.4.1，但本安装过程应该也适用于 Ubuntu 16.04 LTS。

**软件包内容**

着手开始安装前，你应该首先检查安装文件是否齐全，一份完整的软件包至少包含以下文件：

* BERN52.tgz：主程序包；
* CAMPAIGN52.tgz：示例工程；
* DATAPOOL.tgz：输入数据；
* ICONS.tgz：程序图标；
* README：程序说明文档；
* SAVEDISK.tgz：输出数据；
* setup.sh：程序安装脚本。

**安装依赖程序**

Bernese 软件代码量庞大，计算部分主要采用 Fortran 语言编写，界面部分使用基于 [Qt](https://www.qt.io/) 的 C++ 实现，但也有一些 Perl 或 Shell 语言实现的脚本。在编译安装 Bernese 之前，你需要安装以下程序：

* 依赖库文件；
* C++ 编译器；
* Fortran 编译器;
* Perl 5；
* Qt 4。

目前，绝大部分的 Linux 发行版都已经自带了 Perl 5 解释器和 GNU-C++（g++）编译器，当然也包括 Ubuntu OS。因此实际上只安装所需依赖库、Fortran 编译器和 Qt 4 即可。

**安装依赖库**

Bernese 软件依赖于 X11 和 zlib，编译软件之前需要首先安装这些依赖。使用如下命令安装 X11 和 zlib：

|  |  |
| --- | --- |
| 1  2  3  4 | $ sudo apt-get install libx11-dev  $ sudo apt-get install libxext-dev  $ sudo apt-get install libxtst-dev  $ sudo apt-get install zlib1g-dev |

**安装 GNU-Fortran**

Fortran 编译器用于编译 Bernese 软件中使用 Fortran 语言编写的计算程序，在终端中使用如下命令安装 GNU-Fortran（gfortran）编译器：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install gfortran |

**安装 Qt 4**

Bernese 的图形操作界面部分基于 Qt 开发，编译程序菜单时需要引用 Qt 4。从 Qt 4 的下载页面，下载 [Qt 4 的源代码](http://download.qt.io/archive/qt/4.8/4.8.7/qt-everywhere-opensource-src-4.8.7.tar.gz) 进行编译。这里以 qt-everywhere-opensource-src-4.8.7.tar.gz 为例。首先解压源代码文件：

|  |  |
| --- | --- |
| 1 | $ tar -zxvf qt-everywhere-opensource-src-4.8.7.tar.gz |

使用 cd 命令进入解压得到的文件夹，执行如下命令将 Qt 4 安装到 /usr/local/qt4 目录内：

|  |  |
| --- | --- |
| 1  2  3 | $ sudo ./configure -release -opensource -static -prefix /usr/local/qt4  $ sudo make  $ sudo make install |

Qt 4 的编译安装过程需要耗费较长的时间，具体时长取决于你的计算机硬件配置，耐心等待其编译完成。然后设置环境变量 QTDIR，将其指向 Qt 4 的安装目录。如果你使用 Bash，请使用以下命令：

|  |  |
| --- | --- |
| 1 | $ export QTDIR="/usr/local/qt4" |

如果你是 C Shell 或 TC Shell 用户，使用以下命令：

|  |  |
| --- | --- |
| 1 | $ setenv QTDIR "/usr/local/qt4" |

至此，依赖环境已经配置完毕，可以开始安装 Bernese 了。

**安装 Bernese**

Bernese 程序为用户提供了安装引导脚本 setup.sh，执行该脚本并根据提示操作即可完成安装过程。Bernese 默认安装到当前用户的 Home 文件夹，但也可以安装到其他文件夹，完整安装需要约 500 MB 的磁盘空间。为了供多个用户使用，需要共享 BERN52 和 MENU.INP 的执行和修改权限。本示例将 Bernese 安装至 /opt 目录内。

使用 cd 命令进入 Bernese 源码文件夹，通过 root 用户执行其中的安装引导脚本：

|  |  |
| --- | --- |
| 1 | $ sh setup.sh |

之后程序提示输入 Bernese 软件的安装目录：

Full path where the BERN52 software tree will be installed  
[ /root ]:

这里的 /root 表示默认安装路径。我希望将其安装到 /opt/BERN52 文件夹，因此输入：/opt。之后程序提示找到的 Perl 语言解释器位置：

Perl program to be used [ /usr/bin/perl ]:

一般使用系统自带的 Perl 语言解释器就可以了，此处直接回车。之后，设置安装脚本 configure.pm 被启动，并打印以下提示：

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝  
CONFIGURATION OF THE BERNESE GNSS SOFTWARE  
＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝  
0 … Complete installation (Steps 1 to 5)  
1 … Update LOADGPS.setvar  
2 … Install online updates  
3 … Add a new user environment  
4 … Compile the menu  
5 … Compile the programs  
6 … Install the Example campaign  
7 …　－－－

x … Exit

此处第 1 项为更新环境变量设置，第 2 项为安装在线更新包，第 3 项为添加新用户环境，第 4 项为编译图形界面菜单，第 5 项为编辑程序，第 6 项为安装解算工程示例；而第 0 项为完整安装，即执行从第 1 项至第 5 项的全部过程；键入 x 为退出安装程序。这里当然要选择完整安装，键入 0，此后安装程序将自动执行步骤 1 到步骤 5。

随后安装程序打印出支持的编译器，其中 GNU 指示到 gfortran 的路径，表明安装程序找到的了该编译器：

Select Compiler for OS LINUX:  
1: G95  
2: GNU -> /bin/gfortran  
3: IFC  
4: PG\_F90  
5: SUNF90  
6: OTHER

Select [2]:

这里系统已经检测到了 GNU-Fortran，键入 2 选择 gfortran 编译器。然后安装程序继续打印出当前应用的环境变量：

Current Values:  
－－－－－－－  
VARIABLE DESCRIPTION 　 　 　VARIABLE NAME 　 　　　　VARIABLE VALUE  
1: Path to the software　　　　　　C 　 　　　　　　　　　=>　/opt/BERN52  
2: Path to Qt lib for Bernese 　　　 QTBERN　　　　　　　=>　/usr/local/qt\_4.7.4  
3: Operating system group　　　　OS 　 　 　 　　　　　　=>　UNIX  
4: Name of the operating system 　 OS\_NAME 　　　　　　=>　LINUX  
5: Fortran compiler name　　　 　 F\_VERS 　　　　　　　=>　 GNU  
6: List of additional compilers　 　 F\_VERS\_LIST　 　　　 =>  
7: Host of the BPE server　　 　 　 BPE\_SERVER\_HOST 　=>　myComputer  
8: Path to user environment 　　 　U 　 　 　　　　　　 　 =>　${HOME}/GPSUSER52  
9: Path to temp. user environment　T 　 　 　 　　　　　　=>　${HOME}/GPSTEMP  
10: Path to campaign directory 　 　P 　 　 　　　　　 　　 =>　${HOME}/GPSDATA/CAMPAIGN52  
11: Path to datapool area　　　 　　D 　 　 　 　　　　　　=>　${HOME}/GPSDATA/DATAPOOL  
12: Path to savedisk area　　　　 　S 　 　 　 　　　　 　　=>　${HOME}/GPSDATA/SAVEDISK

Accept the values (y/n):

这里的第 1 项为 Bernese 软件的安装目录，第 2 项为所使用的 Qt 4 所在目录，第 3 项为操作系统类型，第 4 项为操作系统名称，第 5 项为使用的 Fortran 编译器，第 6 项为其他可用的编译器，第 7 项为 BPE 服务器，第 8 项为用户环境目录，第 9 项为用户临时文件缓存目录，第 10 项为用户解算工程文件夹，第 11 项为用户数据文件夹目录，第 12 项为解算成果目录。若接受其默认的变量设置，输入 y。要修改其中的配置，输入 n，然后根据其提示，输入变量编号来配置变量。这里使用默认设置，输入 y。

随后安装程序将检测是否存在所需的文件夹，当提示某个文件夹不存在时，程序提示是否创建，键入 y 创建即可。之后将开始 Bernese 的安装过程，泡杯茶等待程序编译完成。

编译成功后提示：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* Bernese menu compiled successfully.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

和

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* Fortran programs compiled successfully.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

如果你看到了以上两个提示，那么恭喜你，Bernese 软件已经编译成功了！

**添加用户**

使用 root 用户进行日常的数据解算操作既不方便也不安全。为了让除 root 以外的其他用户也能使用 Bernese，需要使用新用户加载环境变量设置文件，并启动配置程序。在终端中执行如下命令：

|  |  |
| --- | --- |
| 1  2 | $ source /opt/BERN52/GPS/EXE/LOADGPS.setvar  $ perl /opt/BERN52/GPS/EXE/configure.pm |

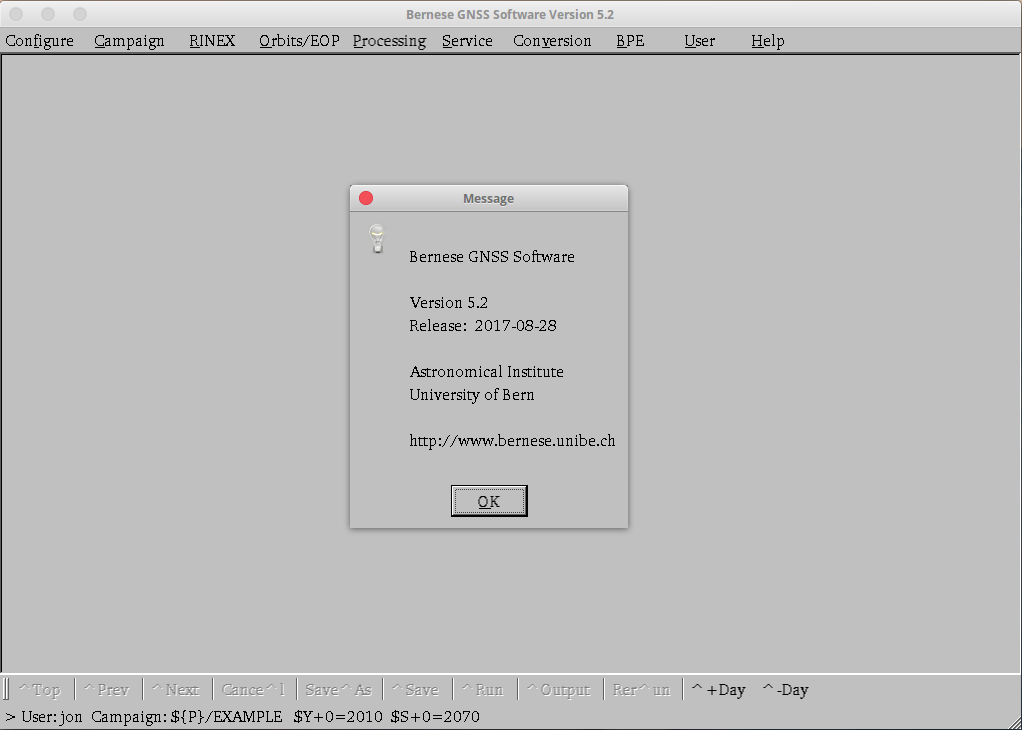
程序再次打印出安装提示，此时键入 3，根据程序提示添加新用户的环境变量即可。

**配置环境变量**

为了在每次开机后都能直接使用 Bernese 软件，需要配置在系统启动时自动加载环境变量。编辑用户主文件夹内的配置文件，Bash 用户打开 ~/.bashrc，C Shell 或 TC Shell 用户打开 ~/.cshrc，为其添加以下内容：

|  |  |
| --- | --- |
| 1  2 | # Load environment variables for Bernese GNSS Software  source /opt/BERN52/GPS/EXE/LOADGPS.setvar |

之后将可以启动 Bernese 了，你可以在终端中键入 G 试一下。若程序安装成功，将弹出类似下面的 Bernese 软件窗口：

[](http://www.gnss.help/images/bernese-elementaryos.png)Bernese GNSS Software

# 基于 elementary OS 配置数据处理与分析环境

[2018-05-07](http://www.gnss.help/2018/05/07/elementary-os-install-config/)

[LINUX](http://www.gnss.help/categories/Linux/)

[LINUX](http://www.gnss.help/tags/Linux/)

GNU/Linux 操作系统的发行版众多，各有其特色和侧重。对于使用者而言，需要明确自己的需求，才能选择适合自己的发行版。此外，安装 Linux 系统之后，配置应用程序也是必要的过程。本文以 [elementary OS](https://elementary.io/) 为例，记录我配置用于 GNSS 数据处理与分析工作环境的过程。首先是对自己经验的总结，便于再次安装时查看；对于其他 Linux 的使用者而言，或许也能有一点儿借鉴意义。

需要说明的是：这里安装的某些软件可能基于个人偏好，并不一定是唯一的选择。

## 操作系统

GNU/Linux 操作系统的发行版不计其数，我选择 [elementary OS](https://elementary.io/)，主要基于以下理由：

* 它很漂亮！elementary OS 由来自英国的设计师主导设计，界面类似于 macOS 但更轻盈；
* 基于 [Ubuntu](http://www.ubuntu.com/)，兼容 Ubuntu 的软件仓库和包，运行较稳定，用户多，有详细的文档参考；
* 自带 [NTFS-3G](https://www.tuxera.com/community/open-source-ntfs-3g/) 等软件，对由 Windows 格式化的移动硬盘可实现即插即用，方便数据交换；
* 拥有一些创新的小特性，例如终端程序的任务完成通知等，对于运行终端程序来说很方便。

本次安装的 elementary OS 操作系统版本为 Loki 0.4.1，基于 Ubuntu 16.04.3 LTS。在 [elementary OS](https://elementary.io/) 的官方网站上，你可以下载到最新版本。如果有能力，请尽量为其捐赠。但其捐赠系统目前只能接收美元，你可以通过点击 “其他” 输入框，输入 “0” 来获取免费下载链接。

下载完成后，将其刻录到光盘，也可以使用 [Rufus](https://rufus.akeo.ie/) 或者 [Universal USB Installer](https://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/) 软件烧录成安装 U 盘。然后从光盘或烧录的 U 盘启动计算机，进入 “Install elementary OS” 项目，根据安装程序提示，设置操作系统的时间和语言、安装位置、时区等。整个安装过程非常简单。

## 软件源

elementary OS 使用 Ubuntu 的软件源。鉴于国内对默认的 Ubuntu 软件源访问速度较慢，因此，添加国内的软件源镜像可以显著地提高软件包的下载和更新速度。国内著名的开源镜像有[清华大学](https://mirrors.tuna.tsinghua.edu.cn/)、[阿里云](https://opsx.alibaba.com/mirror)、[网易](http://mirrors.163.com/)等维护的站点。elementary OS 的软件源配置文件为 /etc/apt/sources.list，这里以配置使用[清华大学镜像](https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu/)为例，首先备份原文件：

|  |  |
| --- | --- |
| 1 | $ sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak |

然后编辑软件源文件，将其替换为以下内容：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | # 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释  deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial main restricted universe multiverse  # deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial main restricted universe multiverse  deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-updates main restricted universe multiverse  # deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-updates main restricted universe multiverse  deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-backports main restricted universe multiverse  # deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-backports main restricted universe multiverse  deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-security main restricted universe multiverse  # deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-security main restricted universe multiverse  # 预发布软件源，不建议启用  # deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-proposed main restricted universe multiverse  # deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-proposed main restricted universe multiverse |

最后执行以下命令刷新软件源缓存：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get update |

离线情况下，你也可以参照 [Linux 系统搭建离线软件源](http://www.gnss.help/2016/09/02/build-offline-software-source/)一文来创建一个本地的软件源，方便后续软件包的安装。

## 应用程序

操作系统配置完毕后，下一步就是安装用于数据处理和分析的应用程序了。

### Anaconda

[Anaconda](https://anaconda.org/) 是 Python 专注于科学计算和数据分析的发行版，相比官方的 CPython 增加了许多科学计算库。如果需要使用 Python 进行数据分析等工作，特别是在离线环境下，直接安装 Anaconda 可以节约很多时间。

在 [Download Anaconda](https://www.anaconda.com/downloads) 页面获取适用于 Linux 的安装包，然后使用 Bash 执行它。以 Anaconda3-5.1.0-Linux-x86\_64.sh 为例：

|  |  |
| --- | --- |
| 1 | $ sudo bash ~/Downloads/Anaconda3-5.1.0-Linux-x86\_64.sh |

根据提示，输入 Anaconda 的安装目录，等待程序执行完毕。程序安装完成后，会询问是否要把 Anaconda 程序目录加入到 PATH 中。这里不建议将其加入到 PATH：elementary OS Loki 已经安装了 Python 3.5，若将 Anaconda 加入到 PATH 中，可能会造成一些冲突（或许会造成 ibus 输入法不可用）。使用以下命令将必要的程序链接到系统的 /usr/bin 目录即可：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $ cd /usr/bin  $ ln -s /usr/local/anaconda/bin/python3.6 ./py  $ ln -s /usr/local/anaconda/bin/pip ./  $ ln -s /usr/local/anaconda/bin/conda ./  $ ln -s /usr/local/anaconda/bin/ipython ./  $ ln -s /usr/local/anaconda/bin/jupyter ./  $ ln -s /usr/local/anaconda/bin/pandoc ./ |

如此，需要使用 Anaconda 中的 Python 时，运行 py，而运行 python3 则会运行系统自带的 Python 3.5。

### Git

[Git](https://git-scm.com/) 是最流行的开源分布式版本控制工具，该程序已经被收录到软件源里。使用以下命令即可完成安装：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install git |

### Visual Studio Code

[Visual Studio Code](https://code.visualstudio.com/) 是微软出品的文本编辑器，亦是一个轻量级的开发工具，功能强大，美观易用。从 [Download Visual Studio Code](https://code.visualstudio.com/Download) 页面下载 .deb 后缀名的安装包，此处以 code\_1.23.0-1525361119\_amd64.deb 为例。依次运行以下命令安装它：

|  |  |
| --- | --- |
| 1  2  3 | $ sudo apt-get install gconf2-common  $ sudo apt-get install libgconf-2-4  $ sudo dpkg -i code\_1.23.0-1525361119\_amd64.deb |

安装 Visual Studio Code 后，你可以通过为其添加扩展来增强功能。若需要在离线环境下安装 Visual Studio Code 扩展，可以从[扩展商店](https://marketplace.visualstudio.com/VSCode)进入特定扩展页面，点击 “Download Extension” 链接下载 .vsix 格式的程序包，然后使用命令安装：

|  |  |
| --- | --- |
| 1 | $ code --install-extension <package.vsix> |

这里的 <package.vsix> 代表要安装的扩展文件路径。

### GAMIT/GLOBK

GAMIT/GLOBK 是由 MIT 和 SIO 联合开发的高精度 GNSS 数据处理软件。elementary OS 基于 Ubuntu，因此你可以参照本博客 [Ubuntu 操作系统安装 GAMIT](http://www.gnss.help/2016/08/26/ubuntu-install-gamit106/) 一文来安装 GAMIT/GLOBK 程序，安装及配置过程完全一致。

### 其他程序

#### Cartopy

Cartopy 是为 Python 提供地图制图功能的扩展工具包，其安装可参考本博客 [Cartopy 简介与安装](http://www.gnss.help/2018/01/30/cartopy-install-introduce/)一文。

#### dos2unix

dos2unix 是一个用于将文本文件字符和换行符编码在 Windows、macOS 和 Linux 系统之间转换的工具。该程序的安装和使用可参考本博客 [DOS2UNIX 安装与使用简介](http://www.gnss.help/2017/07/24/dos2unix-install-usage/)。

#### GFZRNX & TEQC

GFZRNX 和 TEQC 都是用于 GNSS 数据预处理的工具，这两个程序的配置和使用可参考本博客：[TEQC 配置及其数据转换功能的使用](http://www.gnss.help/2016/09/17/teqc-config-translate/) 和 [GFZRNX 配置及格式相关功能的使用](http://www.gnss.help/2016/10/30/gfzrnx-config-translate/)。

#### Graphviz

Graphviz 是一个命令行程序，它可以根据一种名为 dot 的图形结构描述语言，生成简洁美观的示意图。如结构图、流程图、UML 图等。该程序的安装参考本博客：[Graphviz 安装与使用简介](http://www.gnss.help/2017/06/30/graphviz-install-usage/)。

#### Pandoc

Pandoc 能够将文档在 Markdown、LaTeX、reStructuredText、HTML 和 Word docx 等多种标记格式之间相互转换，并支持输出 PDF、EPUB、HTML 幻灯片等多种格式，是编写文档的利器。该程序的安装参考本博客：[Pandoc 安装与使用心得](http://www.gnss.help/2017/06/12/pandoc-install-usage/)。

## 补充

若你需要一个运行在服务器或者工作站上的、非常稳定的 Linux 操作系统，elementary OS 可能不太适合。在我的计算机上，elementary OS Loki 出现了重新点亮屏幕后无法正常显示登录界面或桌面的 Bug（经检查，该问题是因为显示器同时通过 DP 和 DVI 接口连接至主机所致，断开 DVI 接口之后恢复正常）。随着 Ubuntu 18.04 LTS 的发布，elementary OS 的下一个版本 Juno 也将在几个月之后释出，希望新版本可以带来更好的稳定性和更强大的功能。

# Cartopy 绘图示例库

[2018-04-24](http://www.gnss.help/2018/04/24/cartopy-gallery/)

[PYTHON](http://www.gnss.help/categories/Python/)

[CARTOPY](http://www.gnss.help/tags/Cartopy/), [PYTHON](http://www.gnss.help/tags/Python/)

[Cartopy](http://scitools.org.uk/cartopy/) 是为了向 Python 添加地图制图功能而开发的扩展库。该项目致力于以 [matplotlib](https://matplotlib.org/) 包为基础，用简单直观的方式操作各类地理要素的成图。[Cartopy 官网的画廊](http://scitools.org.uk/cartopy/docs/latest/gallery/)页面已经提供了很多绘图的例子，它们和官方文档一起，是学习该工具的主要材料。

本文亦提供一些例子，演示 Cartopy 在测量学等领域的应用，包括绘制中国政区图、IGS 站点分布图、GNSS 控制网、电子含量分布图以及突出显示某些地理要素等，旨在提供大地测量学方面的补充。

## 绘制中国政区图

本图使用兰伯特等积投影。Cartopy 默认使用的由 [Natural Earth](http://www.naturalearthdata.com/) 提供的国界数据不符合我国的领土主张，本文的中国政区边界数据来自 [GMT 中文社区](http://gmt-china.org/example/ex003/)，包含中国国界、省界、十段线以及南海诸岛。你也可以在 [GMT 中文社区](http://gmt-china.org/example/ex003/)网站上下载单独的国界和十段线数据。在此向 GMT 中文社区的维护和贡献者表示感谢！

[](http://www.gnss.help/images/china.png)中国政区图

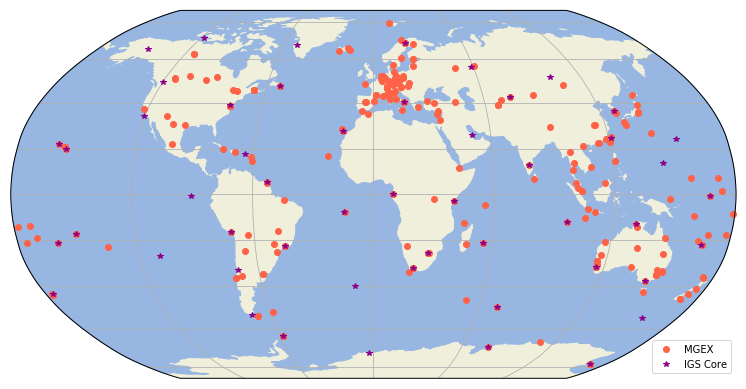
本图使用的中国国界、省界、十段线数据中，边界线数据块使用 “>” 号分隔。因此首先将其内容按照 “>” 切块，然后加载到 NumPy 中。绘图使用的代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | import numpy as np  import matplotlib.pyplot as plt  import cartopy.crs as ccrs  import cartopy.feature as cfeature  # Load the border data, CN-border-La.dat is downloaded from  # http://gmt-china.org/datas/CN-border-La.dat  with open('CN-border-La.dat') as src:  context = src.read()  blocks = [cnt for cnt in context.split('>') if len(cnt) > 0]  borders = [np.fromstring(block, dtype=float, sep=' ') for block in blocks]  # Set figure size  fig = plt.figure(figsize=[10, 8])  # Set projection and plot the main figure  ax = plt.axes(projection=ccrs.LambertConformal(central\_latitude=90,  central\_longitude=105))  # Add ocean, land, rivers and lakes  ax.add\_feature(cfeature.OCEAN.with\_scale('50m'))  ax.add\_feature(cfeature.LAND.with\_scale('50m'))  ax.add\_feature(cfeature.RIVERS.with\_scale('50m'))  ax.add\_feature(cfeature.LAKES.with\_scale('50m'))  # Plot border lines  for line in borders:  ax.plot(line[0::2], line[1::2], '-', color='gray',  transform=ccrs.Geodetic())  # Plot gridlines  ax.gridlines(linestyle='--')  # Set figure extent  ax.set\_extent([80, 130, 13, 55])  # Plot South China Sea as a subfigure  sub\_ax = fig.add\_axes([0.741, 0.11, 0.14, 0.155],  projection=ccrs.LambertConformal(central\_latitude=90,  central\_longitude=115))  # Add ocean, land, rivers and lakes  sub\_ax.add\_feature(cfeature.OCEAN.with\_scale('50m'))  sub\_ax.add\_feature(cfeature.LAND.with\_scale('50m'))  sub\_ax.add\_feature(cfeature.RIVERS.with\_scale('50m'))  sub\_ax.add\_feature(cfeature.LAKES.with\_scale('50m'))  # Plot border lines  for line in borders:  sub\_ax.plot(line[0::2], line[1::2], '-', color='gray',  transform=ccrs.Geodetic())  # Set figure extent  sub\_ax.set\_extent([105, 125, 0, 25])  # Show figure  plt.show() |

下载对应的 [Jupyter Notebook](http://www.gnss.help/images/Plot-China.ipynb)。

## 绘制 IGS 站点分布图

本图使用的 IGS 核心站与 MGEX 项目站点，及其坐标均来自 [IGS 网站](http://www.igs.org/network)。我已经将其整理成为 [igs-core](http://www.gnss.help/images/igs-core.csv) 和 [mgex](http://www.gnss.help/images/mgex.csv) 两个 CSV 文件，你可以直接下载。

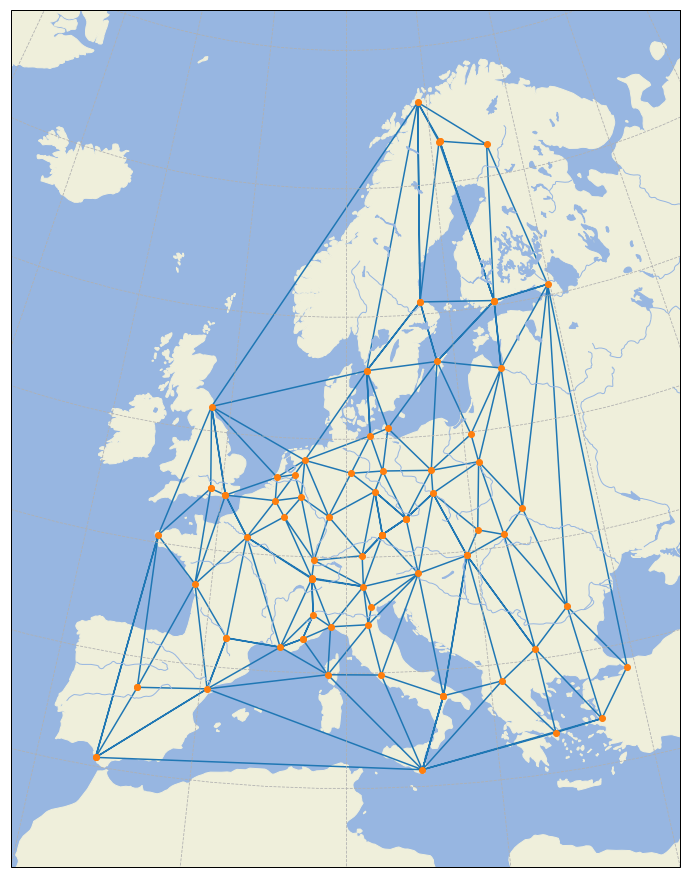
[](http://www.gnss.help/images/igs-sites.png)IGS 核心站与 MGEX 站点分布图

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | import numpy as np  import matplotlib.pyplot as plt  import cartopy.crs as ccrs  import cartopy.feature as cfeature  # Load the coordinate of IGS Core & MGEX sites, The CSV files are  # exported from: http://www.igs.org/network  # Download igs-core.csv & mgex.csv from  # http://ocijpaoi3.bkt.clouddn.com/igs-core.csv  # http://ocijpaoi3.bkt.clouddn.com/mgex.csv  igs\_core = np.recfromcsv('igs-core.csv', names=True, encoding='utf-8')  mgex = np.recfromcsv('mgex.csv', names=True, encoding='utf-8')  fig = plt.figure(figsize=[9, 6])  # Set projection  ax = plt.axes(projection=ccrs.Robinson())  # Add ocean and land  ax.add\_feature(cfeature.LAND)  ax.add\_feature(cfeature.OCEAN)  # Add MGEX & IGS core sites  ax.plot(mgex['longitude'], mgex['latitude'], 'o', color='tomato',  label='MGEX', transform=ccrs.Geodetic())  ax.plot(igs\_core['longitude'], igs\_core['latitude'], '\*', color='darkmagenta',  label='IGS Core', transform=ccrs.Geodetic())  # Plot gridlines  ax.gridlines(linestyle='--')  # Set figure extent  ax.set\_global()  # Set legend location  plt.legend(loc='lower right')  # Show figure  plt.show() |

下载对应的 [Jupyter Notebook](http://www.gnss.help/images/Plot-IGS-network.ipynb)。

## 绘制 GNSS 控制网

这里使用的 IGS 站点坐标数据同样来自 [IGS 网站](http://www.igs.org/network)。我已将其整理成一个 CSV 格式的文件：[euro-igs](http://www.gnss.help/images/euro-igs.csv)，你可以直接下载使用。这里使用 matplotlib.tri 中的 Triangulation 来根据输入的点位坐标来创建 Delaunay 三角网，然后使用 plt.triplot() 方法绘制。

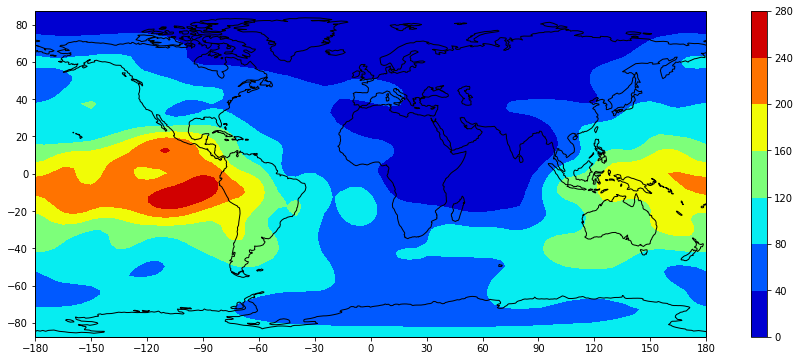
[](http://www.gnss.help/images/gnss-net.png)GNSS 控制网

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | import numpy as np  import matplotlib.pyplot as plt  import matplotlib.tri as tri  import cartopy.crs as ccrs  import cartopy.feature as cfeature  # Load coordinate of the IGS sites in Europe, this CSV file is  # exported from: http://www.igs.org/network  # Download euro-igs.csv from:  # http://ocijpaoi3.bkt.clouddn.com/euro-igs.csv  igs\_sites = np.recfromcsv('euro-igs.csv', names=True, encoding='utf-8')  # Generate Delaunay triangles  triangles = tri.Triangulation(igs\_sites['longitude'], igs\_sites['latitude'])  fig = plt.figure(figsize=[6, 8])  # Set projection  ax = plt.axes(projection=ccrs.LambertConformal(central\_latitude=90,  central\_longitude=10))  # Add ocean, land, rivers and lakes  ax.add\_feature(cfeature.OCEAN.with\_scale('50m'))  ax.add\_feature(cfeature.LAND.with\_scale('50m'))  ax.add\_feature(cfeature.RIVERS.with\_scale('50m'))  ax.add\_feature(cfeature.LAKES.with\_scale('50m'))  # Plot triangles  plt.triplot(triangles, transform=ccrs.Geodetic(), marker='o', linestyle='-')  # Plot gridlines  ax.gridlines(linestyle='--')  # Set figure extent  ax.set\_extent([-10, 30, 30, 73])  # Show figure  plt.show() |

下载对应的 [Jupyter Notebook](http://www.gnss.help/images/Plot-GNSS-Network.ipynb)。

## 绘制电子含量分布图

Cartopy 以 matplotlib 包作为基础，可以使用 matplotlib 中的方法来绘制等值线图，只需在绘制时使用 Cartopy 处理地图投影变形。这里以绘制全球电离层电子含量图为例，模型来自[北京航空航天大学·前沿电离层实验室](http://ionosphere.cn/)，但只截取其产品文件 [whug1420.18i](ftp://ftp.ionosphere.cn/product/2018/142/whug1420.18i.Z) 中第一个时段的数据，即使用的 [1st-tecmap.dat](http://www.gnss.help/images/1st-tecmap.dat) 文件。

[](http://www.gnss.help/images/gim.png)全球电子含量分布图

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | import numpy as np  import matplotlib.pyplot as plt  import cartopy.crs as ccrs  # Read VTEC data from a file, download 1st-tecmap.dat from:  # http://ocijpaoi3.bkt.clouddn.com/1st-tecmap.dat  with open('1st-tecmap.dat') as src:  data = [line for line in src if not line.endswith('LAT/LON1/LON2/DLON/H\n')]  tec = np.fromstring(''.join(data), dtype=float, sep=' ')  # Generate a geodetic grid  nlats, nlons = 71, 73  lats = np.linspace(-87.5, 87.5, nlats)  lons = np.linspace(-180, 180, nlons)  lons, lats = np.meshgrid(lons, lats)  tec.shape = nlats, nlons  fig = plt.figure(figsize=(8, 3))  # Set projection  ax = plt.axes(projection=ccrs.PlateCarree())  # Plot contour  cp = plt.contourf(lons, lats, tec, transform=ccrs.PlateCarree(), cmap='jet')  ax.coastlines()  # Add a color bar  plt.colorbar(cp)  # Set figure extent & ticks  ax.set\_extent([-180, 180, -87.5, 87.5])  ax.set\_xticks([-180, -150, -120, -90, -60, -30, 0, 30, 60, 90, 120, 150, 180])  ax.set\_yticks([-80, -60, -40, -20, 0, 20, 40, 60, 80])  plt.show() |

下载对应的 [Jupyter Notebook](http://www.gnss.help/images/Plot-TEC.ipynb)。

## 突出显示某区域

Cartopy 默认对所有地物使用相同的外观，如果需要突出显示某些地物，就必须进行筛选。这里从 Natural Earth 提供的[小比例尺国界](http://www.naturalearthdata.com/downloads/110m-cultural-vectors/110m-admin-0-countries/)数据中，提取出欧盟国家，然后使用 ax.add\_geometries() 方法将它们加入到绘图元素中。

[](http://www.gnss.help/images/eu.png)欧洲联盟

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | import matplotlib.pyplot as plt  import cartopy.crs as ccrs  import cartopy.feature as cfeature  import cartopy.io.shapereader as shpreader  # Country names in Europe Union, exported from Wikipedia:  # https://en.wikipedia.org/wiki/European\_Union  eu\_country\_names = ('Austria', 'Belgium', 'Bulgaria', 'Croatia', 'Cyprus',  'Czechia', 'Denmark', 'Estonia', 'Finland', 'France',  'Germany', 'Greece', 'Hungary', 'Ireland', 'Italy',  'Latvia', 'Lithuania', 'Luxembourg', 'Malta',  'Netherlands', 'Poland', 'Portugal', 'Romania', 'Slovakia',  'Slovenia', 'Spain', 'Sweden', 'United Kingdom')  # Create a .shp file reader  shp\_file = shpreader.natural\_earth(resolution='110m', category='cultural',  name='admin\_0\_countries')  shp\_reader = shpreader.Reader(shp\_file)  # Reader the shape file  records = shp\_reader.records()  # Collect the European Union countries  eu\_countries = []  for rec in records:  if rec.attributes['NAME'] in eu\_country\_names:  eu\_countries.append(rec)  # Start ploting  fig = plt.figure(figsize=[6, 6])  # Set projection  ax = plt.axes(projection=ccrs.Orthographic(central\_latitude=50,  central\_longitude=10))  # Add land  ax.add\_feature(cfeature.LAND, facecolor='lightgray')  # Plot the European Union countries  for country in eu\_countries:  ax.add\_geometries(country.geometry, crs=ccrs.PlateCarree(),  facecolor='darkgreen')  # Plot gridlines  ax.gridlines(linestyle='--')  # Show figure  plt.show() |

下载对应的 [Jupyter Notebook](http://www.gnss.help/images/Plot-EU.ipynb)。

**RTKLIB 中 convbin 模块的使用**

[2018-03-26](http://www.gnss.help/2018/03/26/rtklib-convbin-usage/)

[RTKLIB](http://www.gnss.help/categories/RTKLIB/)

[RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/), [RTKLIB](http://www.gnss.help/tags/RTKLIB/)

[RTKLIB](http://www.rtklib.com/) 是著名的 GNSS 数据处理软件，提供标准单点定位（SPP）和精密单点定位（PPP）等功能。它由东京海洋大学的 Tomoji Takasu 编写，并开源在 [Github](https://github.com/tomojitakasu/RTKLIB) 网站上。但本文的关注点并不是该程序如何使用，只是介绍其中用于数据转换的一个小模块：convbin。

convbin 是包含在 RTKLIB 中的一个命令行程序，它可以将 RTCM2、RTCM3、u-blox 或 BINEX 等格式的 GNSS 数据转换到 RINEX 2.11。你可以使用该模块完成从原始数据文件到 RINEX 格式的转换。

**安装与简介**

对于 Windows 操作系统，RTKLIB 提供了[预编译好的可执行程序](https://github.com/tomojitakasu/RTKLIB_bin)，你可以直接下载并执行它们。对于 UNIX/Linux 操作系统，则需要下载源代码在本地编译来获得可用的 RTKLIB 软件。作为 RTKLIB 的一部分，convbin 随 RTKLIB 一同发布。但该程序并不依赖其他程序或模块，因此你也可以单独下载使用它。convbin 目前支持的格式或协议有：

* RTCM 2/3；
* NovAtel OEMV/4,OEMStar；
* NovAtel OEM3；
* u-blox LEA-4T/5T/6T；
* NovAtel Superstar II；
* Hemisphere；
* SkyTraq S1315F；
* GW10；
* Javad；
* NVS；
* BINEX；
* Trimble RT17；
* RINEX。

**参数说明**

上文已经说过，convbin 是一个命令行程序，它的使用方式为：

|  |  |
| --- | --- |
| 1 | $ convbin [options] <file> |

这里的 <file> 为输入的文件，[options] 为参数选项。为保持行文流畅，只列出最常用的参数选项，完整的参数选项请参考文末的附表：

* -ts <y/m/d h:m:s>：输出文件的开始时刻，默认为开始记录时刻；
* -te <y/m/d h:m:s>：输出文件的结束时刻，默认为结束记录时刻；
* -tr <y/m/d h:m:s>：RTCM 数据的近似观测时刻；
* -span <span>：观测时段长（h），默认为所有时长；
* -r <format>：输入文件格式，多数时候 convbin 能根据扩展名识别文件类型，但某些时候需要指定；
* -c <station>：使用该参数设定的站名构造输出的 RINEX 文件名；
* -hm <marker>：测量标志名；
* -ho <observer/agency>：观测者与观测机构，以 / 分隔；
* -hr <rec\_id/type/version>：接收机编号、类型与版本，以 / 分隔；
* -ha <antenna\_id/type>：天线编号与天线类型，以 / 分隔；
* -hp <position>：站点概略坐标（XYZ），以 / 分隔；
* -hd <delta\_H/E/N>：站点天线偏移量（HEN），以 / 分隔；
* -v <version>：RINEX 版本号，默认为 RINEX 2.11；
* -d <directory>：数据的输出文件夹，默认为输入数据所在的文件夹。

**使用示例**

**处理单个文件**

convbin 最常见的使用场景就是将某个 RTCM 文件转换为 RINEX 格式。对于单个输入文件，可以如下命令：

|  |  |
| --- | --- |
| 1 | $ convbin rtcm/demo-180311-01.rtcm3 -tr 2018/03/11 1:0:0 -hm DEMO -d rinex/ |

以上命令将 rtcm/ 文件夹内的 demo-180311-01.rtcm3 转换为 RINEX 格式。需注意的是：此处必须使用 -tr 参数来指定数据的近似观测时刻，否则生成的文件中观测时刻都是错误的；并且，由于 RTCM 数据不包含 RINEX 文件头中的元信息，因此这里使用 -hm 指定观测站名。执行完毕后，你可以在 rinex/ 文件夹内找到输出观测文件：demo-180311-01.obs。

**使用 RINEX 命名风格**

你已经看到，上一个命令输出的文件并非 RINEX 格式的命名风格，这一点可能让人不快。实际上，你可以使用 -c 参数来指定站点名，这样程序就可以使用它来构造正确的 RINEX 输出文件名了。示例：

|  |  |
| --- | --- |
| 1 | $ convbin rtcm/demo-180311-01.rtcm3 -tr 2018/03/11 1:0:0 -hm DEMO -d rinex/ -c DEMO |

执行该命令后，将在 rinex/ 目录内找到输出文件 DEMO0700.18O，这里的年积日是根据输入的近似观测时间来计算出来的。

**处理时切割**

有时候 RTCM 文件中包含的数据很多，而我们只需要其中的一段。你可以使用 convbin 指定输出数据的开始和结束观测时刻，实现数据转换时的时间切割。示例：

|  |  |
| --- | --- |
| 1 | $ convbin rtcm/demo-180217-01.rtcm3 -tr 2018/02/17 1:0:0 -ts 2018/02/17 1:0:0 -te 2018/02/17 1:30:0 -hm DEMO -d rinex/ |

在这个命令中，使用 -ts 和 -te 参数，以 “年/月/日 时:分:秒” 的形式指定输出数据的开始和结束时刻。执行完成后，得到文件 DEMO0480.18O。

**处理时合并**

前面的使用方式都只能处理一个文件，但 convbin 亦支持在数据转换的同时进行文件合并。对于大量的数据，你可以这样将输出数据拼接在一起：

|  |  |
| --- | --- |
| 1 | $ convbin rtcm/demo-180217-\*.rtcm3 -tr 2018/02/17 0:0:0 -hm DEMO -d rinex/ -c DEMO -scan |

这里的输入文件是一个通配符，然后使用 -scan 参数告诉程序按照这个规则查找所有匹配的文件名作为输入。因此这个命令可以处理 DEMO 站于 2018 年 2 月 17 日观测的所有数据。执行完成后，将得到 DEMO0480.18O 等。

**元信息指定**

以上所有的命令生成的 RINEX 数据，其文件头中都没有观测者、观测机构等信息。这可以说是一种缺陷，因为完善的此类信息更有利于处理和使用。为了后续过程的便利，你可以在转换时将接收机等信息一并写入到输出文件内。示例：

|  |  |
| --- | --- |
| 1 | $ convbin rtcm/demo-180217-01.rtcm3 -tr 2018/02/17 1:0:0 -hm DEMO -hr 1234567/TPS NETG3/1.00 -ha 7654321/TRM29659.00 GSI -d rinex/ |

在这个命令中，分别使用 -hr 和 -ha 参数来指定输入数据的接收机和天线的信息。等待命令运行完后，将得到输出文件 DEMO0480.18O。

**完整的参数表**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | -ts <y/m/d h:m:s> # 开始时刻，默认为开始记录时刻  -te <y/m/d h:m:s> # 结束时刻，默认为结束记录时刻  -tr <y/m/d h:m:s> # RTCM 数据的近似观测时刻  -ti <interval> # 观测文件采样间隔 (s)，默认为原始采样间隔  -span <span> # 观测时段长 (h)。默认为所有时长  -r <format> # 输入文件格式，该项的可选参数有：  rtcm2 # RTCM 2  rtcm3 # RTCM 3  nov # NovAtel OEMV/4/6,OEMStar  oem3 # NovAtel OEM3  ubx # ublox LEA-4T/5T/6T  ss2 # NovAtel Superstar II  hemis # Hemisphere Eclipse/Crescent  stq # SkyTraq S1315F  javad # Javad  nvs # NVS NV08C BINR  binex # BINEX  rt17 # Trimble RT17  rinex # RINEX  -ro <options> # 在输出文件中添加一个接收机注释  -f <freqency> # 观测频段数，默认为 2  -hc <comment> # RINEX 文件中的注释行  -hm <marker> # 测量标志名  -hn <mark\_number> # 测量标志号  -ht <mark\_type> # 测量标志类型  -ho <observer/agency> # 观测者与观测机构，以 / 分隔  -hr <rec\_id/type/version> # 接收机编号、类型与版本，以 / 分隔  -ha <antenna\_id/type> # 天线编号与天线类型，以 / 分隔  -hp <position> # 站点概略坐标，以 / 分隔  -hd <delta\_H/E/N> # 站点天线偏移量（HEN），以 / 分隔  -v <version> # RINEX 版本号，默认为 RINEX 2.11  -od # 在 RINEX 观测文件中包含多普勒观测量  -os # 在 RINEX 观测文件中包含信噪比信息  -oi # 在 RINEX 导航文件中包含电离层改正数  -ot # 在 RINEX 导航文件中包含时钟改正数  -ol # 在 RINEX 导航文件中包含调秒  -scan # 根据通配符搜索输入文件  -mask [sig[,...]] # 要排除的观测量 (sig={G|R|E|J|S|C}L{1C|1P|1W|...})  -x <satellite> # 要排除的卫星 PRN 号  -y <remove\_system> # 要排除的卫星系统，可选：G R E C J I S  -d <directory> # 数据的输出文件夹，默认为输入数据所在文件夹  -c <station> # 使用该参数设定的站名构造输出的 RINEX 文件名  -o <ofile> # 输出的 RINEX 观测文件  -n <nfile> # 输出的 RINEX 导航文件，GPS  -g <gfile> # 输出的 RINEX 导航文件，GLONASS  -h <hfile> # 输出的 RINEX 导航文件，Geostationary  -q <qfile> # 输出的 RINEX 导航文件，QZSS  -l <lfile> # 输出的 RINEX 导航文件，Galileo  -s <sfile> # 输出的 RINEX 导航文件，SBAS  -trace <level> # 输出 trace level |

**补充**

除了使用本文介绍的 convbin 模块外，在 Windows 操作系统下，你还可以使用相同文件夹内的 rtkconv 这个模块。它是一个具有图形用户界面的 GNSS 数据格式转换程序。相比 convbin，其操作方式更友好，但它只能对单个文件进行操作，在处理大量数据时的效率较低。

# 使用 PyNMEA2 解析 NMEA 0183 协议

[2018-03-01](http://www.gnss.help/2018/03/01/pynmea2-readme/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PYNMEA2](http://www.gnss.help/tags/PyNMEA2/), [PYTHON](http://www.gnss.help/tags/Python/)

[NMEA 0183](http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp) 是一套定义接收机输出的标准协议，有几种不同的格式，每种都是独立的、逗点隔开文本数据。它们包含了可见的卫星、卫星状态、定位状态以及接收机速度等信息。NMEA 0183 实际上已成为所有的 GPS 接收机最通用的数据输出格式，同时它也被用于与 GPS 接收机接口的大多数的软件包里。

[pynmea2](https://pypi.python.org/pypi/pynmea2) 是一个用来处理 NMEA 0183 协议的第三方模块，本文将介绍该模块的安装与使用方法。

## 简介

pynmea2 模块兼容 Python 2 和 Python 3，能够解析 GSA、GGA、GSV、RMC、VTG、GLL 等 NMEA 0183 协议定义的各类数据，功能强大。该模块目前以 MIT 协议开源并托管在 [Github](https://github.com/Knio/pynmea2) 网站上。

## 安装与导入

pynmea2 包已被 PyPI 索引，你只需执行以下命令即可安装 pynmea2：

|  |  |
| --- | --- |
| 1 | $ pip install pynmea2 |

安装成功后，你就可以导入和使用 pynmea2，本文之后的示例代码都假设你已通过如下代码导入了该模块：

|  |  |
| --- | --- |
| 1 | >>> import pynmea2 |

## 从字符串解析

解析字符串中 NMEA 0183 协议的数据，可以使用 pynmea2.parse(data, check=False) 方法，其中的 check 参数指定是否对消息中的检校字段进行检查。

示例代码演示解析 GGA 数据（数据来自[维基百科](https://en.wikipedia.org/wiki/NMEA_0183)）：

|  |  |
| --- | --- |
| 1  2  3  4 | >>> line = '$GPGGA,092750.000,5321.6802,N,00630.3372,W,1,8,1.03,61.7,M,55.2,M,,\*76'  >>> record = pynmea2.parse(line)  >>> record  <GGA(timestamp=datetime.time(9, 27, 50), lat='5321.6802', lat\_dir='N', lon='00630.3372', lon\_dir='W', gps\_qual=1, num\_sats='8', horizontal\_dil='1.03', altitude=61.7, altitude\_units='M', geo\_sep='55.2', geo\_sep\_units='M', age\_gps\_data='', ref\_station\_id='')> |

解析完成后，你就可以通过属性来访问记录中的各个字段了：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | >>> print('GPS Quality Indicator:', record.gps\_qual)  GPS Quality Indicator: 1  >>> print('Horizontal Dilution of Precision:', record.horizontal\_dil)  Horizontal Dilution of Precision: 1.03  >>> print('Latitude:', record.lat)  Latitude: 5321.6802  >>> print('Latitude Direction:', record.lat\_dir)  Latitude Direction: N  >>> print('Number of Satellites:', record.num\_sats)  Number of Satellites: 8  >>> record.timestamp  datetime.time(9, 27, 50) |

除了按照协议约定格式对数据进行解析之外，pynmea2.parse() 函数还做了一些必要的数据转换工作，将经纬度坐标转换为 Python 中的 float 类型：

|  |  |
| --- | --- |
| 1  2  3  4 | >>> print('Latitude:', record.latitude)  Latitude: 53.361336666666666  >>> print('Longitude:', record.longitude)  Longitude: -6.50562 |

此外，解析方法还为输出结果添加了额外的属性：latitude\_minutes，latitude\_seconds，longitude\_minutes和 longitude\_seconds，它们存储了大地坐标的分、秒各部分。因此你可以方便地对坐标进行格式化输出：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | >>> print('Latitude: {:02d}°{:02d}′{:07.4f}″'.format(int(record.latitude), int(record.latitude\_minutes), record.latitude\_seconds))  Latitude: 53°21′40.8120″  >>> print('Longitude: {:02d}°{:02d}′{:07.4f}″'.format(int(record.longitude), int(record.longitude\_minutes), record.longitude\_seconds))  Longitude: -6°30′20.2320″  >>> print('Altitude: {:.3f}'.format(record.altitude))  Altitude: 61.700 |

## 从文件中解析

NMEA 0183 协议数据经常存储在文件中，对于这种应用场景，pynmea2 创建了 pynmea2.NMEAFile 类。你可以使用这个类对遵守 NMEA 0183 协议的文件进行处理，只需传入目标文件的路径：

|  |  |
| --- | --- |
| 1  2  3 | >>> nmea\_file = pynmea2.NMEAFile('data/demo.gga')  >>> nmea\_file.readline()  <GSA(mode='A', mode\_fix\_type='3', sv\_id01='01', sv\_id02='20', sv\_id03='19', sv\_id04='13', sv\_id05='', sv\_id06='', sv\_id07='', sv\_id08='', sv\_id09='', sv\_id10='', sv\_id11='', sv\_id12='', pdop='40.4', hdop='24.4', vdop='32.2')> |

以上代码只是为了演示 pynmea2.NMEAFile 类最基本的使用方式。实际使用中不需要这样，该类已经实现了迭代器和上下文管理器接口。上下文管理器可以帮你打理好文件的打开与关闭，迭代器则可以让循环操作的代码更清晰易读。因此更 Pythonic 的使用姿势为：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | >>> records = []  >>> with pynmea2.NMEAFile('data/demo.gga') as nmea\_file:  ... for record in nmea\_file:  ... records.append(record)  ...  >>> print('Count of records:', len(records))  Counts of records: 12  >>> print('The last record:', repr(records[-1]))  The last record: <RMC(timestamp=datetime.time(9, 27, 51), status='A', lat='5321.6802', lat\_dir='N', lon='00630.3371', lon\_dir='W', spd\_over\_grnd=0.06, true\_course=31.66, datestamp=datetime.date(2011, 5, 28), mag\_variation='', mag\_var\_dir='') data=['A']> |

## 从数据流解析

pynmea2 还能够直接处理 NMEA 0183 协议的数据流，你可以使用 pynmea2.NMEAStreamReader 类来解析数据流：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | >>> streamreader = pynmea2.NMEAStreamReader(input\_stream)  >>> while 1:  ... for record in streamreader.next():  ... print(repr(record))  ...  <GSV(num\_messages='3', msg\_num='1', num\_sv\_in\_view='11', sv\_prn\_num\_1='10', elevation\_deg\_1='63', azimuth\_1='137', snr\_1='17', sv\_prn\_num\_2='07', elevation\_deg\_2='61', azimuth\_2='098', snr\_2='15', sv\_prn\_num\_3='05', elevation\_deg\_3='59', azimuth\_3='290', snr\_3='20', sv\_prn\_num\_4='08', elevation\_deg\_4='54', azimuth\_4='157', snr\_4='30')>  <VTG(true\_track=89.68, true\_track\_sym='T', mag\_track=None, mag\_track\_sym='M', spd\_over\_grnd\_kts=Decimal('0.00'), spd\_over\_grnd\_kts\_sym='N', spd\_over\_grnd\_kmph=0.0, spd\_over\_grnd\_kmph\_sym='K')>  <GLL(lat='4250.5589', lat\_dir='S', lon='14718.5084', lon\_dir='E', timestamp=datetime.time(9, 22, 4, 999000), status='A')>  ... |

## 生成 NMEA 0183 数据

上文介绍过了数据解析，你还可以使用 pynmea2 包来编码生成符合 NMEA 0183 协议的数据，只需使用合适的数据来实例化 pynmea2 中对应的类。

示例，使用下面的代码生成 GGA 数据（使用 pynmea2.GGA 类），传入的参数依次为数据源（talker）、类型（type）和字段记录（fields）：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | >>> record = pynmea2.parse('$GPGGA,020249.20,3511.49986189,N,11922.42306087,E,0,04,4.1,-2.285,M,4.514,M,,\*7E')  >>> print(record.data)  ['020249.20', '3511.49986189', 'N', '11922.42306087', 'E', '0', '04', '4.1', '-2.285', 'M', '4.514', 'M', '', '']  >>> sentence = pynmea2.GGA('GP', 'GGA', ('020249.20', '3511.49986189', 'N', '11922.42306087', 'E', '0', '04', '4.1', '-2.285', 'M', '4.514', 'M', '', ''))  >>> str(sentence)  '$GPGGA,020249.20,3511.49986189,N,11922.42306087,E,0,04,4.1,-2.285,M,4.514,M,,\*7E' |

## 异常处理

pynmea2 在数据解析失败时会抛出 pynmea2.nmea.ParseError 异常，在需要的时候，你可以捕获并处理它。示例：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | >>> try:  ... gga\_msg = pynmea2.parse('at^sisw=1,1500,0,0')  ... except pynmea2.nmea.ParseError:  ... print('Warning: a line parsing failed!')  ...  Warning: a line parsing failed! |

**TRACK 模块的配置与使用**

[2018-02-16](http://www.gnss.help/2018/02/16/track-usage/)

[TRACK](http://www.gnss.help/categories/TRACK/)

[TRACK](http://www.gnss.help/tags/TRACK/)

TRACK 是 GAMIT/GLOBK 程序的重要组成部分，该模块使用卡尔曼滤波算法，提供基于差分相位观测量的动态后处理定位功能。随着针对如飞行器、车辆、船舶或浮标等运动设备的 GPS 动态定位技术需求的不断增加，TRACK 模块凭借较高的数据处理精度，已成为该领域最受重视的程序工具之一。

本文将介绍 TRACK 模块的使用和配置过程。

**使用方法**

TRACK 模块目前还只能处理针对 GPS 卫星的观测数据。该模块进行数据处理时，其所有的配置参数都是通过控制文件来输入的，因此控制文件的编辑是该模块的使用过程中最重要的部分。编写好配置文件之后，你可以用以下方式启动 TRACK 程序：

|  |  |
| --- | --- |
| 1 | $ track -f <cmd-file> -a <amb-file> -d <day> -w <week> -s <strings> |

参数释义：

* -f <cmd-file>：输入的控制文件
* -a <amb-file>：输入的模糊度文件；
* -d <day>：数据观测的年积日；
* -w <week>：数据观测的 GPS 周；
* -s <strings>：字符串组。

在这些参数中，只有控制文件是必要的。这里的 -d <day>、-w <week>、-s <strings> 实际上是为了编写控制文件模板而设置的变量。在编辑控制文件时，你可以使用 <day> 来代替年积日，使用 <week> 来代替 GPS 周，然后在运行 TRACK 时通过命令行参数来设置它们的值。

**控制文件**

TRACK 模块的控制文件通常是一个 .cmd 文件，MIT 在其网站上给出了一个[控制文件示例](http://geoweb.mit.edu/~tah/track_example/track.cmd)。在控制文件中，每一行作为一个控制命令，有效的控制命令必须以至少一个空白字符开始，否则将被当作注释。此外，有些控制命令可能需要多个输入项（例如观测数据），对于这样的命令，其后紧跟的非注释行都将被作为其子项，直到遇到一个空行。

在控制文件中，第一个控制命令必须是输入的观测数据，其后控制命令的顺序可以是任意的。接下来将对最主要的几个控制命令做详细介绍。

**观测数据**

观测数据的输入使用 obs\_file 命令，其后的每一行输入一个观测文件。因此其格式为：

|  |  |
| --- | --- |
| 1  2  3 | obs\_file  <site> <RINEX\_file> <type>  ... |

其中的 <site> 表示站名，<RINEX\_file> 为站点对应的观测文件，最后的 <type> 代表站点的活动状态（即静止或运动）。静止的站点使用 “F” 来标记，运动的站点使用 “K” 标记。

示例，输入两个站于 2018 年第 045 日的观测文件，其中 STAT 站为静态站，FLOT 为动态站。你可以这样设置：

|  |  |
| --- | --- |
| 1  2  3 | obs\_file  stat stat0450.18o F  flot flot0450.18o K |

**星历数据**

导航卫星的星历数据使用 nav\_file 输入。TRACK 程序既可以接收 RINEX 格式的广播星历数据，也可以接收 SP3 格式的精密星历数据。其输入格式为：

|  |  |
| --- | --- |
| 1 | nav\_file <NAVI\_file> <type> |

这里的 <NAVI\_file> 表示星历文件名，<type> 星历的数据类型标记。广播星历使用 NAV 标记，精密星历使用 <SP3> 标记。需注意的是，对于一个时段的解算，目前 TRACK 只能接受一个星历文件。因此如果要处理跨天的数据，你需要手动将星历数据拼接成一个文件。

示例，对应于上文的观测文件，以下命令输入精密星历数据：

|  |  |
| --- | --- |
| 1 | nav\_file igr19883.sp3 SP3 |

**解算模式**

实际的数据处理中，根据观测环境和状态的不同，通常需要采取不同的解算策略。TRACK 提供了许多参数，给予你对解算过程最自由的控制权。为了简化配置过程，TRACK 同样提供了针对不同情况的快速设置。根据动态站载体或动态站和静态站之间的距离，你可以通过 mode 参数来对解算过程进行快速设置。解算模式的控制命令格式为：

|  |  |
| --- | --- |
| 1 | mode <type> |

可用的解算模式有：

* AIR：动态站位于飞行器上；
* SHORT：站点之间的基线长度在 300km 内；
* LONG：站点之间的基线长度大于 300km。

示例，要使用 SHORT 模式，只需进行如下配置：

|  |  |
| --- | --- |
| 1 | mode SHORT |

**概略坐标**

在默认情况下，TRACK 使用 RINEX 观测数据文件头中的概略坐标。但某些时候，观测文件中的概略坐标可能不正确，这将导致解算出错。此时你可以将正确的站点概略坐标写入到控制文件中。输入概略坐标的控制命令格式为：

|  |  |
| --- | --- |
| 1  2 | site\_pos  <site> <X> <Y> <Z> |

这里，<site> 为站名，<X>、<Y>、<Z> 分别为站点在空间直角坐标系下的坐标分量。

示例，对应于上文的观测站，输入其概略坐标：

|  |  |
| --- | --- |
| 1  2  3 | site\_pos  stat -2592743.6008 4468222.8317 3728304.5354  flot -2605480.4172 4455888.3716 3734143.3193 |

**参考点坐标**

在默认情况下，TRACK 的解算结果将以所输入的第一个参考站的站心坐标系（即 NEU）的形式输出。当然，你也可以手动配置输出成果的参考点。参考点坐标的输入方式为：

|  |  |
| --- | --- |
| 1 | ref\_neu <X> <Y> <Z> |

这里的 <X>、<Y>、<Z> 分别为站点在空间直角坐标系下的坐标分量。

示例，配置参考点坐标：

|  |  |
| --- | --- |
| 1 | ref\_neu -2606241.4364 4454156.4228 3735667.9585 |

**成果输出形式**

TRACK 程序的数据处理成果默认以 NEU 坐标输出，但你也可以添加其他的成果输出形式。配置成果输出的控制命令为：

|  |  |
| --- | --- |
| 1 | out\_type <type> |

这里的 <type> 即成果的输出形式，它可以是以下一种或几种：

* GEOD：大地坐标系；
* NEU：站心坐标系；
* DHU：解算结果与初始坐标的差异；
* XYZ：空间直角（笛卡尔）坐标系；
* DUMP：输出 L1/L2 伪距和相位组合的残差，如果输入了 IONEX 文件，还可以同时输出 TEC（总电子含量）。

以上选项可以自由组合，例如，配置同时输出 NEU 和大地坐标系（BLH）的坐标：

|  |  |
| --- | --- |
| 1 | out\_type GEOD+NEU |

**解算间隔**

数据处理时选择合适的解算间隔也是很重要的，间隔太稠密将大大延长数据处理所需的时间。默认的数据处理间隔同观测文件一致，但你也可以用以下形式来配置：

|  |  |
| --- | --- |
| 1 | interval <num> |

这里的 <num> 以秒作为单位。因此要将数据解算的间隔设置为 30 秒，你可以如下配置：

|  |  |
| --- | --- |
| 1 | interval 30 |

**成果文件**

TRACK 程序的解算结果主要保存在 SUM 文件中，你可以使用 sum\_file 命令来配置结果输出文件：

|  |  |
| --- | --- |
| 1 | sum\_file <filename> |

这里的 <filename> 即 SUM 文件的文件名。例如，对于上文的解算任务，可以如下配置：

|  |  |
| --- | --- |
| 1 | sum\_file TRACK18\_045.sum |

**任务示例**

MIT 在其网站上给出了一个使用 TRACK 进行[数据处理的示例](http://geoweb.mit.edu/~tah/track_example)，你可以在页面上查看其数据处理成果，也可以下载其[完整的输入文件](http://geoweb.mit.edu/~tah/track_example/track_example.tar.Z)在自己的计算机上重新运行。

下载并解压输入文件包之后，使用 cd 命令进入输入文件目录，然后使用如下命令启动 TRACK:

|  |  |
| --- | --- |
| 1 | $ track -f track.cmd -d 356\_19 -w 12501 | tee TRAK\_356\_19.out |

TRACK 程序的帮助信息非常详尽，可说是巨细无遗，不加任何参数运行 track 即可看到。MIT 还准备了一份[在线的帮助信息](http://geoweb.mit.edu/~simon/gtgk/help/track.hlp.htm)供你随时随地查看。

**Cartopy 中的地图投影**

[2018-02-11](http://www.gnss.help/2018/02/11/projection-in-cartopy/)

[PYTHON](http://www.gnss.help/categories/Python/)

[CARTOPY](http://www.gnss.help/tags/Cartopy/), [PYTHON](http://www.gnss.help/tags/Python/)

经典地图制图中使用的投影面有圆柱面、圆锥面或平面等。根据所用投影面，地图投影相对应地可分为圆柱投影、圆锥投影和方位投影。后来又发展出了很多现代地图投影，如伪圆柱、多圆锥等投影办法。在制图时选择合适的地图投影是一件重要的工作。

本文将对 Cartopy 中支持的地图投影进行逐一进行简要介绍，这些投影都是 cartopy.crs 中的类。若要绘制某一类投影的地图，只需将其实例化之后传入 plt.axes() 方法的 projection 参数即可。本文不涉及投影的原理，旨在展示每种地图投影的效果和用法，以便在选择时参考。

**圆柱投影**

**PlateCarree**

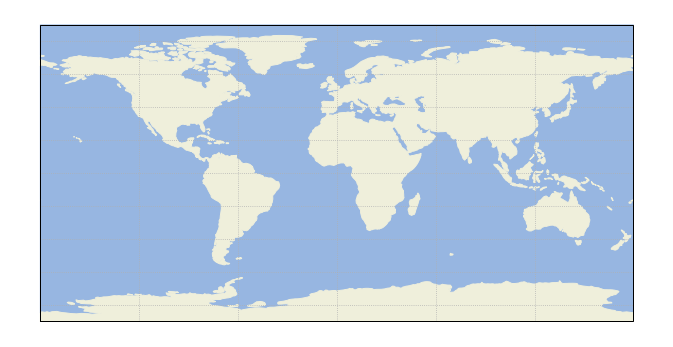
PlateCarree（可利）投影将地物投影到圆柱面上再展开，常用来绘制世界地图。该类的定义为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.PlateCarree(central\_longitude=0.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-platecarree.png)

**LambertCylindrical**

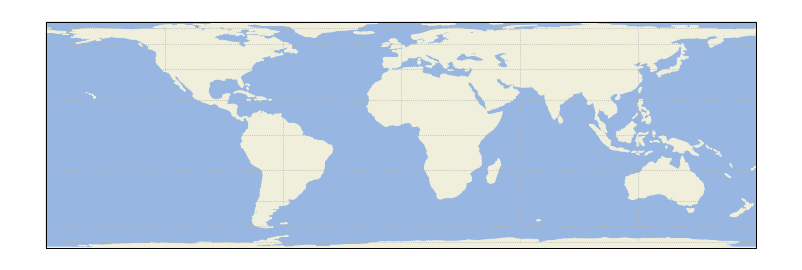
Lambert cylindrical projection（兰伯特等积投影）也是绘制世界地图的一种常用方案。该类的定义为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.LambertCylindrical(central\_longitude=0.0) |

参数说明：

* central\_longitude：中央经线，默认为 0。

示例：

[](http://www.gnss.help/images/projections-lambertcylindrical.png)

**Mercator**

Mercator（墨卡托）投影，属于正轴等角割圆柱投影。经纬线为相互垂直的平行直线，图上无角度变形，但面积变形较大。该投影常用于制作航海图或赤道附近区域的地图。其类定义为：

|  |  |
| --- | --- |
| 1  2 | class cartopy.crs.Mercator(central\_longitude=0.0, min\_latitude=-80.0, max\_latitude=84.0,  latitude\_true\_scale=0.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* min\_latitude：显示的最小纬线（最南端），默认为 -80；
* max\_latitude：显示的最大纬线（最北端），默认为 84；
* latitude\_true\_scale：长度不变形的纬线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-mercator.png)

**Miller**

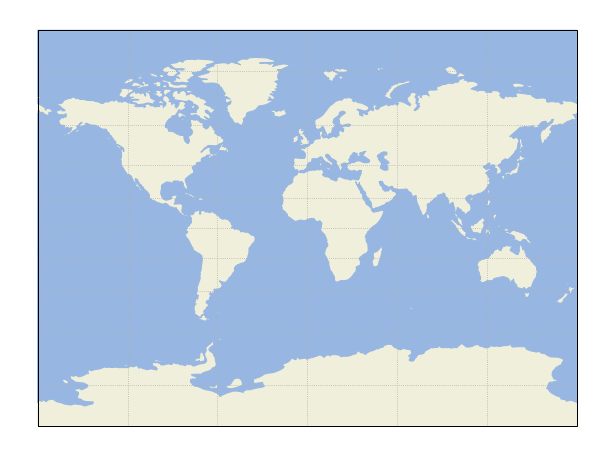
Miller（米勒）投影。其类定位为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.Miller(central\_longitude=0.0) |

参数说明：

* central\_longitude：中央经线，默认为 0。

示例：

[](http://www.gnss.help/images/projections-miller.png)

**TransverseMercator**

TransverseMercator（横轴墨卡托）投影是对标准墨卡托投影的一种变换。其类定义为：

|  |  |
| --- | --- |
| 1  2  3 | class cartopy.crs.TransverseMercator(central\_longitude=0.0, central\_latitude=0.0,  false\_easting=0.0, false\_northing=0.0,  scale\_factor=1.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* central\_latitude：中央纬线，默认为 0；
* false\_easting：平面上 X 轴坐标（东方向）的偏移量，以米为单位；
* false\_northing：平面上 Y 轴坐标（北方向）的偏移量，以米为单位；
* scale\_factor：中央经线处的长度变形因子，默认为 1；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-transversemercator.png)

**UTM**

UTM（通用横轴墨卡托投影），制图中最常用的一种投影方法之一。在 UTM 中，地球上的主要陆地被划分为一个个的网格区间。对于纬度，从南纬 80° 开始，每 8° 被划分为一个区间，自南向北由从 “C” 至 “X” 的字母表示（为包含北极附近的陆地，最北的区间包含 12°）；对于经度，从零度子午线开始每隔 6° 向东编排，其带好以数字 01-60 表示。该类的定义为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.UTM(zone, southern\_hemisphere=False, globe=None) |

参数说明：

* zone：所绘地区的 6° 经度区间带号；
* southern\_hemisphere：投影带是否在南半球；
* globe：椭球定义，默认为 WGS84。

示例：

[http://www.gnss.help/images/projections-utm.png](http://www.gnss.help/images/projections-utm.png)

**RotatedPole**

RotatedPole（旋转极）投影，常被用于天气预测模型中。该类的定义为：

|  |  |
| --- | --- |
| 1  2 | class cartopy.crs.RotatedPole(pole\_longitude=0.0, pole\_latitude=90.0,  central\_rotated\_longitude=0.0, globe=None) |

参数说明：

* pole\_longitude：旋转极的经度；
* pole\_latitude：旋转极的纬度；
* central\_rotated\_longitude：旋转的经度；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-rotatedpole.png)

**OSGB**

OSGB（Ordnance Survey of Great Britain），英国军用测量大地网，用于测量大不列颠岛的一种区域等距圆锥投影。

示例：

[](http://www.gnss.help/images/projections-osgb.png)

**EuroPP**

EuroPP，用于绘制欧洲地图的一种等距圆锥投影。

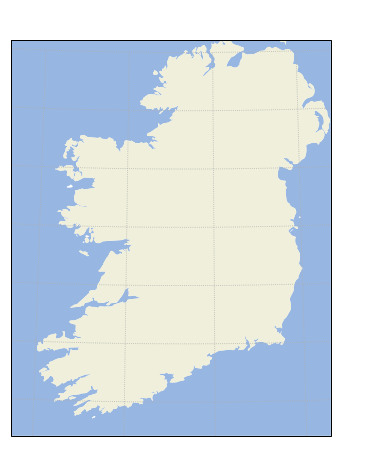
示例：

[](http://www.gnss.help/images/projections-europp.png)

**OSNI**

OSNI（Ordnance Survey of Northern Ireland），一种区域性质的等距圆锥投影，用于测绘北爱尔兰岛。

示例：

[](http://www.gnss.help/images/projections-osni.png)

**圆锥投影**

**AlbersEqualArea**

AlbersEqualArea（阿尔伯斯等面积）投影。其类定义为：

|  |  |
| --- | --- |
| 1  2  3 | class cartopy.crs.AlbersEqualArea(central\_longitude=0.0, central\_latitude=0.0,  false\_easting=0.0, false\_northing=0.0,  standard\_parallels=(20.0, 50.0), globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* central\_latitude：中央纬线，默认为 0；
* false\_easting：平面上 X 轴坐标（东方向）的偏移量，以米为单位；
* false\_northing：平面上 Y 轴坐标（北方向）的偏移量，以米为单位
* standard\_parallels：长度不变形的纬线，默认为 (20, 50)；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-albersequalarea.png)

**LambertConformal**

LambertConformal（兰伯特正形）投影。其类定义为：

|  |  |
| --- | --- |
| 1  2  3  4 | class cartopy.crs.LambertConformal(central\_longitude=-96.0, central\_latitude=39.0,  false\_easting=0.0, false\_northing=0.0,  secant\_latitudes=None, standard\_parallels=None,  globe=None, cutoff=-30) |

参数说明：

* central\_longitude：中央经线，默认为 -96；
* central\_latitude：中央纬线，默认为 39；
* false\_easting：平面上 X 轴坐标（东方向）的偏移量，以米为单位；
* false\_northing：平面上 Y 轴坐标（北方向）的偏移量，以米为单位
* secant\_latitudes：割线的纬度；
* standard\_parallels：长度不变形的纬线，默认为 (33, 45)；
* cutoff：显示的纬度边界，默认为 -30；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-lambertconformal.png)

**方位投影**

**Orthographic**

Orthographic（正射）投影，视点在无穷远处，常用于表现某个半球。该类的定义为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.Orthographic(central\_longitude=0.0, central\_latitude=0.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* central\_latitude：中央纬线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-orthographic.png)

**AzimuthalEquidistant**

AzimuthalEquidistant（正方位等距）投影，在切点辐射的方向上没有长度变形，有角度和面积变形。该类的定义为：

|  |  |
| --- | --- |
| 1  2 | class cartopy.crs.AzimuthalEquidistant(central\_longitude=0.0, central\_latitude=0.0,  false\_easting=0.0, false\_northing=0.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* central\_latitude：中央纬线，默认为 0；
* false\_easting：平面上 X 轴坐标（东方向）的偏移量，以米为单位；
* false\_northing：平面上 Y 轴坐标（北方向）的偏移量，以米为单位
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-azimuthalequidistant.png)

**Stereographic**

Stereographic（球极平面）投影，这也是一种等角投影，其视点在球面上。但这种投影会造成较大的长度和面积变形，且离切点愈远，其变形幅度愈大。其类定义为：

|  |  |
| --- | --- |
| 1  2  3 | class cartopy.crs.Stereographic(central\_latitude=0.0, central\_longitude=0.0,  false\_easting=0.0, false\_northing=0.0,  true\_scale\_latitude=None, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* central\_latitude：中央纬线，默认为 0；
* false\_easting：平面上 X 轴坐标（东方向）的偏移量，以米为单位；
* false\_northing：平面上 Y 轴坐标（北方向）的偏移量，以米为单位
* true\_scale\_latitude：长度不变形的纬线；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-stereographic.png)

**Geostationary**

Geostationary（地球同步卫星）视角，视点在赤道上空某一点处。这种投影好像在地球同步（GEO）卫星上观察到的地球。其类定义为：

|  |  |
| --- | --- |
| 1  2 | class cartopy.crs.Geostationary(central\_longitude=0.0, satellite\_height=35785831,  false\_easting=0, false\_northing=0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* satellite\_height：卫星高度，默认为 35785831 米，这是 GEO 卫星的高度；
* false\_easting：平面上 X 轴坐标（东方向）的偏移量，以米为单位；
* false\_northing：平面上 Y 轴坐标（北方向）的偏移量，以米为单位
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-geostationary.png)

**NearsidePerspective**

NearsidePerspective（驾驶员）视角，视点在地球外某一点处。这种投影好像在高空中某处宇宙飞船驾驶员的视角。其类定义为：

|  |  |
| --- | --- |
| 1  2  3 | class cartopy.crs.NearsidePerspective(central\_longitude=0.0, central\_latitude=0.0,  satellite\_height=35785831, false\_easting=0,  false\_northing=0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* central\_latitude：中央纬线，默认为 0；
* satellite\_height：卫星高度，默认为 35785831 米，即 GEO 卫星的高度；
* false\_easting：平面上 X 轴坐标（东方向）的偏移量，以米为单位；
* false\_northing：平面上 Y 轴坐标（北方向）的偏移量，以米为单位
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-nearsideperspective.png)

**LambertAzimuthalEqualArea**

LambertAzimuthalEqualArea（兰伯特方位等积）投影，具有等积性质，但在靠近边缘的地方长度和角度变形较大。其类定义为：

|  |  |
| --- | --- |
| 1  2 | class cartopy.crs.LambertAzimuthalEqualArea(central\_longitude=0.0, central\_latitude=0.0,  false\_easting=0.0, false\_northing=0.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* central\_latitude：中央纬线，默认为 0；
* satellite\_height：卫星高度，默认为 35785831 米，即 GEO 卫星的高度；
* false\_easting：平面上 X 轴坐标（东方向）的偏移量，以米为单位；
* false\_northing：平面上 Y 轴坐标（北方向）的偏移量，以米为单位
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-lambertazimuthalequalarea.png)

**Gnomonic**

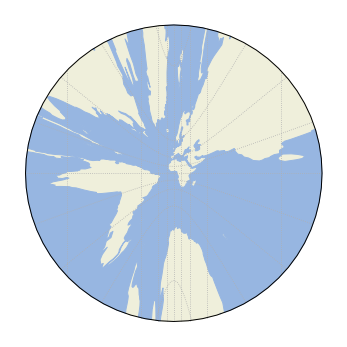
Gnomonic（球心）投影，将地球想象为一个水面透明的玻璃球，这种投影好像从地心光源将陆地的影子投射到墙上的景象。该类定义为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.Gnomonic(central\_longitude=0.0, central\_latitude=0.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* central\_latitude：中央纬线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-gnomonic.png)

**NorthPolarStereo**

NorthPolarStereo（北极极射）投影。该类的定义为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.NorthPolarStereo(central\_longitude=0.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-northpolarstereo.png)

**SouthPolarStereo**

SouthPolarStereo（南极极射）投影。该类的定义为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.SouthPolarStereo(central\_longitude=0.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-southpolarstereo.png)

**其他投影方式**

**Mollweide**

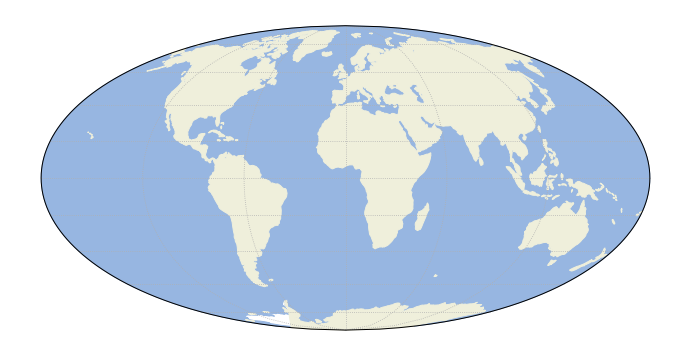
Mollweide（摩尔威特）投影，一种正轴等积性质的伪圆柱投影，由德国人摩尔威特（K.B.Mollweide）所创，常用于绘制世界地图或东西半球图。中央经线为直线，距离中央经线东西相差 90° 的经线构成一个大圆，其面积等于地球表面积的1/2，其余经线为椭圆。Cartopy 中对应的类为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.Mollweide(central\_longitude=0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-mollweide.png)

**Robinson**

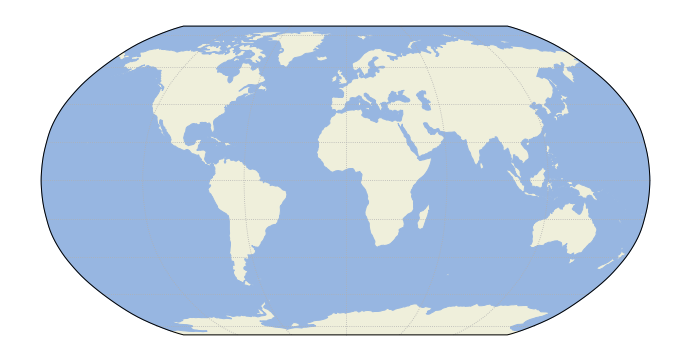
Robinson（罗宾逊）投影也是一种伪圆柱投影，主要为了克服圆柱投影形状拉伸过大，希望获得角度和面积变形均最小的世界地图投影。Cartopy 中对应的类为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.Robinson(central\_longitude=0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-robinson.png)

**Sinusoidal**

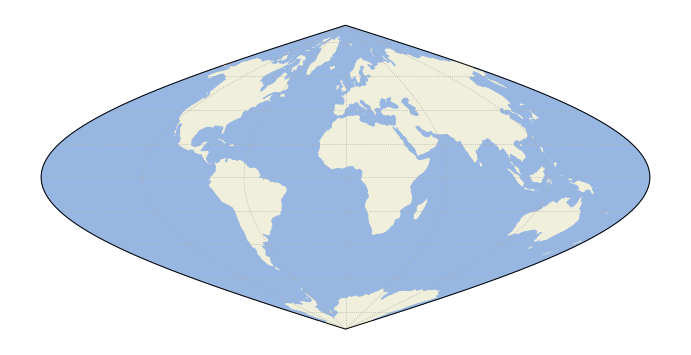
Sinusoidal（正弦）投影是一种伪圆柱等积投影，由法国人桑逊（N.Sanson）所创，有时候也被称为桑逊投影或墨卡托等积投影。这是一种等积性质的伪圆柱投影，纬线为间隔相等的平行直线，经线为对称于中央经线的正弦曲线。Cartopy 中对应的类为：

|  |  |
| --- | --- |
| 1  2 | class cartopy.crs.Sinusoidal(central\_longitude=0.0, false\_easting=0.0,  false\_northing=0.0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-sinusoidal.png)

**InterruptedGoodeHomolosine**

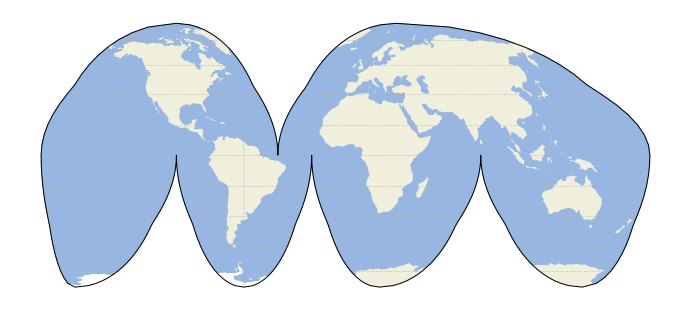
InterruptedGoodeHomolosine（分瓣正弦古德）投影，一种伪圆柱、等积投影。该投影中，全图被分为几瓣，各瓣之间通过赤道连接，这样可使得各大洲陆地部分表现得更完好。其缺点也显而易见：地图的连续性被破坏殆尽。Cartopy 中对应的类为：

|  |  |
| --- | --- |
| 1 | class cartopy.crs.InterruptedGoodeHomolosine(central\_longitude=0, globe=None) |

参数说明：

* central\_longitude：中央经线，默认为 0；
* globe：椭球定义，默认为 WGS84。

示例：

[](http://www.gnss.help/images/projections-interruptedgoodehomolosine.png)

**总结**

对于上文提到的 false\_easting 和 false\_northing 两个变量的含义与用途，有些读者可能会有困惑。这里说一下我的理解。在某些局部地区的坐标系中，为了避免坐标值出现负数，通常会在可能出现负数的坐标上加上一个较大的常数。例如，我国普遍采用高斯投影的平面直角坐标系，以赤道和中央子午线作为坐标轴，由于我国位于北半球，因此南北方向不会出现负数，但是东西方向上会出现负数（中央子午线以西的点）。此时通常的解决办法就是为所有的东西方向的坐标加上 500 km。这里的 500 km 就是 false\_easting 的值。

当制图范围较小时，无论什么投影方式都无太大变形；对于范围广大的世界地图、半球地图、大洲地图、大国地图等，则需要慎重考虑。对表现大块区域常用的投影方式可总结为：

* 世界地图：正圆柱、伪圆柱和多圆锥投影；
* 东、西半球：常选用横轴方位投影；
* 南、北半球：常采用正轴方位投影；
* 水、陆半球：一般选用斜轴方位投影；
* 极地——正轴方位投影；
* 赤道附近——横轴方位投影或正轴圆柱投影；
* 中纬地区——正轴圆锥投影或斜轴方位投影。

最后，生成本文示例图片的代码开放下载，你可以自由选择[独立脚本](http://www.gnss.help/images/projection-scripts.zip)或 [Jupyter Notebook](http://www.gnss.help/images/projections.ipynb)笔记本。

**Cartopy 简介与安装**

[2018-01-30](http://www.gnss.help/2018/01/30/cartopy-install-introduce/)

[PYTHON](http://www.gnss.help/categories/Python/)

[CARTOPY](http://www.gnss.help/tags/Cartopy/), [PYTHON](http://www.gnss.help/tags/Python/)

将空间相关的数据展绘到地图上是常见的需求，实际上也早有一些软件或工具包来完成这样的工作。比较专业的程序如支持 Windows、UNIX/Linux 等多种操作系统的命令行工具 [GMT（Generic Mapping Tools）](https://gmt.soest.hawaii.edu/)，Python 绘图工具 matplotlib 的扩展包 [Basemap](https://matplotlib.org/basemap/)，地理信息系统使用的专业软件 [ArcGIS](https://www.arcgis.com/) 等。但在我看来，这些工具也都有些美中不足之处。

本文将介绍 Python 的制图工具包 [Cartopy](http://scitools.org.uk/cartopy/)，以及它在各平台的安装过程。

**Cartopy 简介**

对于制作用于演示说明的地图而言，ArcGIS 毫无疑问是在大材小用，GMT 和 Basemap 相对而言比较合适。但 GMT 大多数时间还是在命令提示符界面（或终端）中使用，要集成到程序中比较困难；而作为 Python 2 的殉葬品，[Basemap 将在 2020 年停止维护](https://matplotlib.org/basemap/users/intro.html#cartopy-new-management-and-eol-announcement)。因此，如果你在寻找一个在 Python 中使用的制图工具包，作为 Basemap 钦定的继承人，Cartopy 几乎就是最好的选择了。

Cartopy 是一个开源免费的第三方 Python 扩展包，由英国气象办公室的科学家们开发，支持 Python 2.7 和 Python 3，致力于使用最简单直观的方式生成地图，并提供对 matplotlib 友好的协作接口。该工具包使用 LGPLv3 协议，代码托管在 [Github](https://github.com/SciTools/cartopy) 网站上。

**安装**

**使用 Anaconda**

如果你正在使用 Python 的科学计算发行版 Anaconda，安装 Cartopy 非常容易。你只需要运行：

|  |  |
| --- | --- |
| 1 | $ conda install -c conda-forge cartopy |

等待这个命令运行完毕，就能使用 Cartopy 了！

**Linux 操作系统**

在 Linux 上，安装 Cartopy 也很容易。Cartopy 依赖于 [GEOS](https://trac.osgeo.org/geos/) 和 [PROJ.4](https://proj4.org/) 这两个包，所以需要首先安装所需的依赖：

|  |  |
| --- | --- |
| 1  2  3  4 | $ sudo apt-get install libgeos-dev  $ sudo apt-get install libgeos++-dev  $ sudo apt-get install proj-bin  $ sudo apt-get install libproj-dev |

然后使用 PIP 安装 Cartopy：

|  |  |
| --- | --- |
| 1 | $ pip install cartopy |

**Windows 操作系统**

在 Windows 系统的计算机上，你还可以使用 PIP 来安装预编译包的 Wheel 包。安装 Cartopy 所需的包有：

* pyproj：处理地图投影变形，Cartopy 基于它定义了丰富的地图投影；
* pillow：Python 的图像处理包，读写和操作栅格图像；
* pyshp：Python 实现的 ESRI Shapefile 读写包；
* shapely：操作和分析空间地理对象；
* cartopy：Cartopy 安装包，地图制图工具。

你可以在 [UC Irvine](https://www.lfd.uci.edu/~gohlke/pythonlibs/) 的页面上找到这些工具包针对 Windows 操作系统的预编译版本。下载适合你的安装包，然后按照以上清单的顺序将所需要的程序包安装到计算机中。例如，我使用的是 64 位的 Python 3.5，因此安装过程如下：

|  |  |
| --- | --- |
| 1  2  3  4  5 | $ pip install pyproj-1.9.5.1-cp35-cp35m-win\_amd64.whl  $ pip install Pillow-5.0.0-cp35-cp35m-win\_amd64.whl  $ pip install pyshp‑1.2.12‑py2.py3‑none‑any.whl  $ pip install Shapely-1.6.4.post1-cp35-cp35m-win\_amd64.whl  $ pip install Cartopy-0.16.0-cp35-cp35m-win\_amd64.whl |

**数据源**

Cartopy 可以读取 Shapefile 作为输入的数据源，并且原生支持由 [Natural Earth](http://www.naturalearthdata.com/) 或 [GSHHS](https://www.ngdc.noaa.gov/mgg/shorelines/gshhs.html) 发布的开源地理数据。但为了保持 Cartopy 的精致小巧，该包并未附带任何地理数据。只有当你首次使用某个数据源时，它才会进行下载，并保存在数据目录下以备以后使用。

为了在离线的情况下也能使用 Cartopy，你可以将预下载的地图数据放在其指定的文件夹内。Cartopy 的配置信息保存在 cartopy.config 变量中，要查看数据文件夹位置，你可以在 Python 终端中运行以下命令：

|  |  |
| --- | --- |
| 1  2 | >>> import cartopy  >>> print(cartopy.config['data\_dir']) |

你可以手动下载所需的数据并分门别类地放入 Cartopy 数据文件夹中。我已经下载并整理了常用的数据文件并打包为一个[压缩文件](http://www.gnss.help/images/natural_earth.zip)，你可以解压后直接使用。它们包括：

* 1:10m、1:50m、1:110m 比例尺的海岸线；
* 1:10m、1:50m、1:110m 比例尺的海洋、陆地和岛屿；
* 1:10m、1:50m、1:110m 比例尺的湖泊、河流；
* 1:10m、1:50m、1:110m 比例尺的地理界线（赤道、回归线和极圈）；
* 1:10m、1:50m、1:110m 比例尺的国家界线；
* 1:10m、1:50m、1:110m 比例尺的省、州界线；
* 1:10m、1:50m、1:110m 比例尺的主要城市。

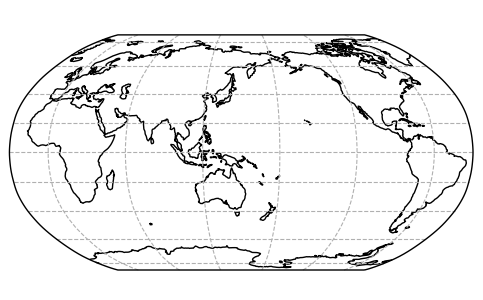
**Hello World**

Cartopy 的工作流非常简单：设置地图投影，添加地图要素，最后显示地图。Cartopy 与 matplotlib 协作得很好，对于高阶的功能，你可以直接使用 matplotlib。

下面是一个 Cartopy 版本的 “Hello world”：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | import matplotlib.pyplot as plt  import cartopy.crs as ccrs  # set projection  ax = plt.axes(projection=ccrs.Robinson(central\_longitude=150))  # plot coastlines & gridlines  ax.coastlines()  ax.gridlines(linestyle='--')  # show figure  plt.show() |

复制并运行以上命令，你将看到一副熟悉的世界地图。Cartopy 的使用就是这样，简单而直观！

[](http://www.gnss.help/images/Robinson_world.png)世界地图

**使用 LaTex 撰写数学公式**

[2018-01-23](http://www.gnss.help/2018/01/23/formula-using-latex2/)

[LATEX](http://www.gnss.help/categories/Latex/)

[LATEX](http://www.gnss.help/tags/Latex/)

[LaTex](https://www.latex-project.org/) 是一种是一种基于 TeX 的排版系统。该系统由美国计算机科学家莱斯利·兰伯特在20世纪80年代初期开发。对于生成复杂的表格、公式、化学方程式等，该系统的优势尤为突出。因此在学术论文或书籍等的出版行业，LaTex 是最常用的排版方案之一。

数学公式的排版是 LaTex 的拿手好戏，限于篇幅，本文只总结使用 LaTex 语言撰写数据公式的部分。如果你希望从头开始学习 LaTex，建议查看其[文档](https://www.latex-project.org/help/documentation/)或[其他组织撰写的教程](http://www.mohu.org/info/lshort-cn.pdf)。

**基础知识**

使用 LaTex 排版的公式大体分为两种：随段落排版的，夹在行内的公式；和单独成行的，独立的展开公式。类似与 HTML 或 XML，这两种方式都使用开始、结束的标记来声明。而对于其内容，将忽略其中的空白并进行自动编排。

行内数学公式的标记方法有三种：

* $Formula$：这是最常用的标记方式，某些较高级的 Markdown 解析器也支持这种标记方法，例如前文曾提到过的 [Pandoc](https://www.latex-project.org/help/documentation/)。
* \begin{math}Formula\end{math}：对待非常长的数学公式，这种显式声明的方式更清晰。
* \(Formula\)：该方式是上一种方式的省略写法。

展开公式的标记方法为：

* \[Formula\]：最常用的方式，该方式不会为公式编号。
* \begin{displaymath}Formula\end{displaymath}：显示声明的方式，在编写长公式时较清晰，该方式也不会为公式编号。
* \begin{equation}Formula\end{equation}：另一种显示声明的方式，该方式将自动为公式编号。

需要注意的是，在 LaTex 中，并没有完整的定义使用 “\$\$” 来标记展开公式的用法。但很多编辑器亦支持这种写法。使用该方法之前需记得首先对编辑器进行测试。

**符号规则**

**指数和下标**

指数和下标分别可以用符号 ^ 或 \_ 后加相应字符来实现。比如，对于勾股定理，可以写作为：

|  |  |
| --- | --- |
| 1 | a^2 + b^2 = c^2 |

**开方符号**

平方根符号使用 \sqrt{a} 来表示，对于 n 次方根，其对应的表示方式为：\sqrt[n]{a}。示例，使用如下方式表示的向量 (x, y) 至原点的距离：

|  |  |
| --- | --- |
| 1 | f(x) = \sqrt{x^2 + y^2} |

**分式**

如果要用分子分母上下排版的方式表示分数，可以使用 \frac{a}{b} 的形式。因此可以用如下方式显示一元二次方程的求根公式：

|  |  |
| --- | --- |
| 1 | x\_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} |

**水平线，向量**

有时候你可能希望在公式上方或下方添加水平线，或者添加用于表示矢量的箭头标志。它们分别可以使用 \overline、\underline 或 \vec 表示。因此你可以用如下的方式来表示牛顿第二定律：

|  |  |
| --- | --- |
| 1 | \vec{F} = m \cdot \vec{a} |

**积分、求和与乘积**

积分、求和与乘积运算符也是常用的数学符号，它们分别使用 \inf、\sum 和 \prod 表示。其中积分的上下限使用 ^ 和 \_ 来表示，类似于之前提到的上下标。示例，你可以以如下方式表示一个定积分：

|  |  |
| --- | --- |
| 1 | \int\_{0}^{\pi} \sin(x) dx = 2 |

**字体与字号**

**字体**

在某些情况下，你可能希望自定义公式的字体，以表达某些特殊含义。你可以用下面的几种标记来更改公式的字体：

* \mathrm：标准字体；
* \mathtt：较纤细的字体；
* \mathbf：粗体；
* \mathsf：无衬线体；
* \mathit：斜体；
* \mathcal：花体字。

**字号**

在某些分式中，默认的字号可能很小以至于难以阅读，你可以使用如下的几个标记来调节公式的字号：

* \displaystyle：展开公式的标准字号大小；
* \textstyle：行内公式的标准字号大小；
* \scriptstyle：第一层上下标字体的大小；
* \scriptscriptstyle：第二层上下标的字号大小。

**附录**

**特殊函数**

我们还可能需要用到一些常见的函数，LaTex 内置了常用的 32 种函数的表示方法：

| **表示** | **效果** |
| --- | --- |
| \arccos(x) | $\arccos(x)$ |
| \arcsin(x) | $\arcsin(x)$ |
| \arctan(x) | $\arctan(x)$ |
| \arg(x) | $\arg(x)$ |
| \cos(x) | $\cos(x)$ |
| \cosh(x) | $\cosh(x)$ |
| \cot(x) | $\cot(x)$ |
| \coth(x) | $\coth(x)$ |
| \csc(x) | $\csc(x)$ |
| \deg(x) | $\deg(x)$ |
| \det(x) | $\det(x)$ |
| \dim(x) | $\dim(x)$ |
| \exp(x) | $\exp(x)$ |
| \gcd(x) | $\gcd(x)$ |
| \hom(x) | $\hom(x)$ |
| \inf(x) | $\inf(x)$ |
| \ker(x) | $\ker(x)$ |
| \lg(x) | $\lg(x)$ |
| \lim(x) | $\lim(x)$ |
| \liminf(x) | $\liminf(x)$ |
| \limsup(x) | $\limsup(x)$ |
| \ln(x) | $\ln(x)$ |
| \log(x) | $\log(x)$ |
| \max(x, y) | $\max(x, y)$ |
| \min(x, y) | $\min(x, y)$ |
| \Pr(x) | $\Pr(x)$ |
| \sec(x) | $\sec(x)$ |
| \sin(x) | $\sin(x)$ |
| \sinh(x) | $\sinh(x)$ |
| \sup(x) | $\sup(x)$ |
| \tan(x) | $\tan(x)$ |
| \tanh(x) | $\tanh(x)$ |

**小写希腊字母**

| **表示** | **效果** |
| --- | --- |
| \alpha | $\alpha$ |
| \beta | $\beta$ |
| \gamma | $\gamma$ |
| \delta | $\delta$ |
| \epsilon | $\epsilon$ |
| \varepsilon | $\varepsilon$ |
| \zeta | $\zeta$ |
| \eta | $\eta$ |
| \theta | $\theta$ |
| \vartheta | $\vartheta$ |
| \iota | $\iota$ |
| \kappa | $\kappa$ |
| \lambda | $\lambda$ |
| \mu | $\mu$ |
| \nu | $\nu$ |
| \xi | $\xi$ |
| o | $o$ |
| \pi | $\pi$ |
| \varpi | $\varpi$ |
| \rho | $\rho$ |
| \varrho | $\varrho$ |
| \sigma | $\sigma$ |
| \varsigma | $\varsigma$ |
| \tau | $\tau$ |
| \upsilon | $\upsilon$ |
| \phi | $\phi$ |
| \varphi | $\varphi$ |
| \chi | $\chi$ |
| \psi | $\psi$ |
| \omega | $\omega$ |

**大写希腊字母**

| **表示** | **效果** |
| --- | --- |
| \Gamma | $\Gamma$ |
| \Delta | $\Delta$ |
| \Theta | $\Theta$ |
| \Lambda | $\Lambda$ |
| \Xi | $\Xi$ |
| \Pi | $\Pi$ |
| \Sigma | $\Sigma$ |
| \Upsilon | $\Upsilon$ |
| \Phi | $\Phi$ |
| \Psi | $\Psi$ |
| \Omega | $\Omega$ |

**关系运算符**

| **表示** | **效果** |
| --- | --- |
| < | $<$ |
| > | $>$ |
| = | $=$ |
| \le | $\le$ |
| \ge | $\ge$ |
| \equiv | $\equiv$ |
| \ll | $\ll$ |
| \gg | $\gg$ |
| \subset | $\subset$ |
| \supset | $\supset$ |
| \approx | $\approx$ |
| \subseteq | $\subseteq$ |
| \supseteq | $\supseteq$ |
| \in | $\in$ |
| \ni | $\ni$ |
| \propto | $\propto$ |
| \mid | $\mid$ |
| \parallel | $\parallel$ |
| \perp | $\perp$ |
| \notin | $\notin$ |
| \ne | $\ne$ |
| : | $:$ |

**数学运算符**

| **表示** | **效果** |
| --- | --- |
| + | $+$ |
| - | $-$ |
| \pm | $\pm$ |
| \mp | $\mp$ |
| \times | $\times$ |
| \cdot | $\cdot$ |
| \ast | $\ast$ |
| \div | $\div$ |
| \cap | $\cap$ |
| \cup | $\cup$ |
| \setminus | $\setminus$ |
| \star | $\star$ |
| \circ | $\circ$ |
| \diamond | $\diamond$ |
| \bullet | $\bullet$ |
| \oplus | $\oplus$ |
| \ominus | $\ominus$ |
| \odot | $\odot$ |
| \otimes | $\otimes$ |

**补充**

实际上 LaTex 内置的符号远不止这些，但这里不再给出列表。有需求的读者请查阅其[文档](https://www.latex-project.org/help/documentation/)。

**JDCAL 使用文档**

[2018-01-03](http://www.gnss.help/2018/01/03/jdcal-readme/)

[PYTHON](http://www.gnss.help/categories/Python/)

[JDCAL](http://www.gnss.help/tags/JDCAL/), [PYTHON](http://www.gnss.help/tags/Python/)

[儒略日（Julian Day，JD）](https://zh.wikipedia.org/wiki/%E5%84%92%E7%95%A5%E6%97%A5)是在儒略周期内以天数计算时间的方法，主要在天文学等领域使用，是描述地球、天体等运动的时间衡量方式。儒略日的起点为儒略历的公元前 4713 年 1 月 1 日中午 12 点。由于儒略历的数字位数太多，在实际使用时多有不便，国际天文学联合会又定义了简化儒略日（MJD）。简化儒略日的起点是公历 1858 年 11 月 17 日 0 时。

在儒略日与日常使用的[公历（格里高利历）](https://zh.wikipedia.org/wiki/%E5%85%AC%E5%8E%86)之间转换是经常需要做的事情。本文介绍一个 Python 模块：jdcal，使用该模块可以避免重新发明轮子。

**简介**

[jdcal](https://pypi.python.org/pypi/jdcal) 是由 Prasanth Nair 实现的一个 Python 模块，该模块提供了处理儒略日、儒略历和公历等日期转换的一系列常量和方法。该模块支持 Python 2 和 Python 3，基于 BSD 许可证开源，其代码托管在 [Github](https://github.com/phn/jdcal) 网站上。

**安装与更新**

jdcal 支持使用 pip 安装，在终端或命令提示符界面执行以下命令进行安装：

|  |  |
| --- | --- |
| 1 | $ pip install jdcal |

上面的命令将从 [PyPI](https://pypi.python.org/pypi/jdcal) 仓库下载 jdcal 的程序包，并将其加入到你的程序库中。要更新 jdcal，可运行命令：

|  |  |
| --- | --- |
| 1 | $ pip install --upgrade jdcal |

**模块内容**

**常量**

模块中定义了两个常量：

* **MJD\_0**  
  简化儒略日的起点。
* **MJD\_JD2000**  
  天文学中常用的 JD2000.0（2000/1/1 12:00:00）历元对应的简化儒略日。

作为示例的交互式控制台代码：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> from jdcal import \*  >>> MJD\_0  2400000.5  >>> MJD\_JD2000  51544.5 |

**儒略日与公历**

* **gcal2jd(year, month, day)**  
  将公历的年月日转换为儒略日，返回一个包含两个浮点数的元组。其中第一个数字为简化儒略日的起点，第二个数字为公历对应的简化儒略日（MJD）。因此，要得到公历对应的儒略日（JD），你需要将两个浮点数加起来。还需要注意的是：该函数的输出是按照输入日期的 0 时（不是 12 时）计算的。
* **jd2gcal(jd)**  
  将儒略日或简化儒略日转换为公历中的年月日，返回一个包含 4 个数字的元组。其中前三个为整数的年月日，最后的浮点数为不足一天的小数部分。该函数将两个输入参数的和作为儒略日使用。因此，若你输入的是简化儒略日，则另一个参数应为简化儒略日的起点；若你输入的是儒略日，保持另一个参数为零即可。

示例，使用 gcal2jd(year, month, day) 函数获得公历对应的简化儒略日，然后对输出的元组求和得到儒略日：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> from jdcal import \*  >>> gcal2jd(2000, 1, 1)  (2400000.5, 51544.0)  >>> sum(gcal2jd(2000, 1, 1))  2451544.5 |

示例，使用 jd2gcal(jd) 函数获得儒略日或简化儒略历所对应的公历：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> from jdcal import \*  >>> jd2gcal(0, 2451545)  (2000, 1, 1, 0.5)  >>> jd2gcal(MJD\_0, 51544.5)  (2000, 1, 1, 0.5) |

**儒略日与儒略历**

* **jcal2jd(year, month, day)**  
  将儒略历中的年月日转换为儒略日，返回一个包含两个浮点数的元组。其中第一个数字为简化儒略日的起点，第二个数字为儒略历对应的简化儒略日（MJD）。因此，要得到儒略历对应的儒略日（JD），你亦需要将两个浮点数加起来。另外，该函数的输出也是按照输入日期的 0 时（不是 12 时）计算的。
* **jd2jcal(jd)**  
  将儒略日转换为儒略历中的年月日，返回一个包含 4 个数字的元组。其中前三个为整数的年月日，最后的浮点数为不足一天的小数部分。与 jd2gcal(jd) 类似，该函数同样将两个输入参数的和作为儒略日使用。因此，若你输入的是简化儒略日，则另一个参数应为简化儒略日的起点；若你输入的是儒略日，保持另一个参数为零。

示例，使用 jcal2jd(year, month, day) 函数获得儒略历对应的简化儒略日，然后对输出的元组求和得到儒略日：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> from jdcal import \*  >>> jcal2jd(1999, 12, 19)  (2400000.5, 51544.0)  >>> sum(jcal2jd(1999, 12, 19))  2451544.5 |

示例，使用 jd2jcal(jd) 函数获得儒略日或简化儒略历所对应的儒略历：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> from jdcal import \*  >>> jd2jcal(0, 2451544.5)  (1999, 12, 19, 0.0)  >>> jd2jcal(MJD\_0, 51544.0)  (1999, 12, 19, 0.0) |

**其他方法**

* **is\_leap(year)**  
  判断输入的年是否为格里高利历中的闰年。即对于非世纪年（不能被 100 整除），能被 4 整除；对于世纪年，能被 400 整除。

作为示例的交互式控制台代码：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> from jdcal import \*  >>> is\_leap(1900)  False  >>> is\_leap(2000)  True |

**测试**

如果你需要测试 jdcal，其测试代码放在 test\_jdcal.py 中。你可以通过执行该脚本来测试位于相同目录下的 jdcal.py 程序：

|  |  |
| --- | --- |
| 1 | $ python test\_jdcal.py |

**搭建基于 Python 的科学计算和数据分析环境**

[2017-12-29](http://www.gnss.help/2017/12/29/python-scientific-packages/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PYTHON](http://www.gnss.help/tags/Python/)

科学领域的应用是 Python 语言的专长之一。相比 Matlab，作为一个完备的编程语言，Python 具备很多优势：该语言有面向过程、面向对象、函数式等多种编程范式。并且，Python 的开发和运行环境更加轻量级，开发工具也更加多样和自由，程序集成和效率优化方案更成熟。借力数据科学和机器学习等热门应用的东风，Python 成为 [Stackoverflow 网站 2017 年度上升最快的编程语言](https://stackoverflow.blog/2017/09/06/incredible-growth-python/)。

本文介绍 Python 科学计算和数据分析模块的安装和配置过程。

**Python 版本**

当前的 Python 语言分为 Python 2 和 Python 3 两大版本。如果你参考比较老旧的资料，它可能会以兼容性问题等缘由向你推荐 Python 2。但这些已经是过去式了，除非已停止维护，目前绝大部分的模块包都支持 Python 3。Python 3 具有许多新特性，幻灯片 [10 awesome features of Python that you can’t use because you refuse to upgrade to Python 3](http://www.asmeurer.com/python3-presentation/slides.html#1) 在这方面做了精准的概括。

另外，[Python 2 将在 2020 年被停止支持和维护](https://www.python.org/dev/peps/pep-0373/#update)，与此同时[很多第三方扩展库也将同时停止对 Python 2 的支持](http://python3statement.org/)，这包括本文提到的所有工具包。一个方法是：如果你找不到必须使用 Python 2 的理由，那么应该果断投向 Python 3 的怀抱。

**所需模块**

Python 的科学计算和数据分析功能主要依赖于几个著名的模块实现，因此本文也将介绍这几个模块的安装过程。这些模块主要有：

* NumPy，提供 N 维数组类型，该库是科学计算的基础；
* SciPy，科学计算包，Matlab 工具箱的 Python 实现；
* SymPy，符号计算库，求解方程和公式推导；
* matplotlib，绘制各种二维图形（统计图等）；
* Pandas，数据分析库，处理大量数据，用于分析、建模和可视化；
* Jupyter Notebook，提供一个浏览器中使用的交互式 Python 环境。

**安装 Anaconda**

[Anaconda](https://www.anaconda.com/) 是一个 Python 发行版，相比官方发布的 CPython 解释器，Anaconda 集成了大量的第三方库，致力于简化包的管理和部署，专注于大规模的数据处理、科学计算、预测分析和数据挖掘等任务。Anaconda 现提供适用于 Windows、Linux 和 macOS 等操作系统的安装包，整个安装过程简单友好。若你不想在 Python 语言底层耗费太多的时间，只需要一个简单快捷的部署方案，Anaconda 是一个非常优秀的选择。但其缺点是安装包较大，约 400 至 600 MB。

在 Windows 操作系统上安装 Anaconda 时，需避免安装路径中包含中文字符或者空格，以免造成兼容性问题。安装完成后，还要配置环境变量，将 Anaconda 安装路径及其中的 Scripts 文件夹路径加入到系统的 Path 变量中。

为 Linux 操作系统安装 Anaconda 也非常简单，其安装文件已经被封装成 SHELL 程序脚本。你只需要下载对应的安装文件，然后使用 Bash 执行它。安装时程序将提示你输入安装路径等，按照其提示操作即可顺利完成安装过程。

**自定义安装**

如果你只需要为数不多的几个 Python 库，并且希望节约计算机磁盘空间，那么可以手动来安装自己需要的库。[加州大学尔湾分校（UC Irvine）](https://www.lfd.uci.edu/~gohlke/pythonlibs/)的网站上提供了预编译的可用于 Windows 操作系统的许多 Python 扩展库，其中包括了 NumPy、SciPy 等多个扩展库的程序包。你可以根据自己的 Python 版本下载对应的程序包文件。

我使用的解释器版本是 64 位的 Python 3.5，因此下载了如下程序包文件：

* numpy‑1.14.3+mkl‑cp35‑cp35m‑win\_amd64.whl；
* scipy‑1.0.1‑cp35‑cp35m‑win\_amd64.whl；
* sympy‑1.1.1‑py2.py3‑none‑any.whl；
* matplotlib‑2.2.2‑cp35‑cp35m‑win\_amd64.whl；
* pandas‑0.22.3‑cp35‑cp35m‑win\_amd64.whl。

由于该页面提供的程序包使用 Wheel 打包，因此你需要首先安装 Wheel。在 “命令提示符” 窗口执行：

|  |  |
| --- | --- |
| 1 | $ pip install wheel |

使用 cd 命令进入程序包文件所在目录，依次运行以下命令安装所下载的 .whl 包文件：

|  |  |
| --- | --- |
| 1  2  3  4  5 | $ pip install numpy‑1.14.3+mkl‑cp35‑cp35m‑win\_amd64.whl  $ pip install scipy‑1.0.1‑cp35‑cp35m‑win\_amd64.whl  $ pip install sympy‑1.1.1‑py2.py3‑none‑any.whl  $ pip install matplotlib‑2.2.2‑cp35‑cp35m‑win\_amd64.whl  $ pip install pandas‑0.22.3‑cp35‑cp35m‑win\_amd64.whl |

若安装中出现类似 “numpy-1.13.3+mkl-cp35-cp35m-win\_amd64.whl is not a supported wheel on this platform.” 的错误，那么将其文件名中的 “cp35m” 改为 “none”，然后重新安装即可。该问题的详细解答可参考 [Stackoverflow 网站的相关条目](https://stackoverflow.com/questions/28107123/cannot-install-numpy-from-wheel-format)。

在联网的情况下，之后的 Jupyter Notebook 可以通过 pip 在线安装：

|  |  |
| --- | --- |
| 1 | $ pip install jupyter |

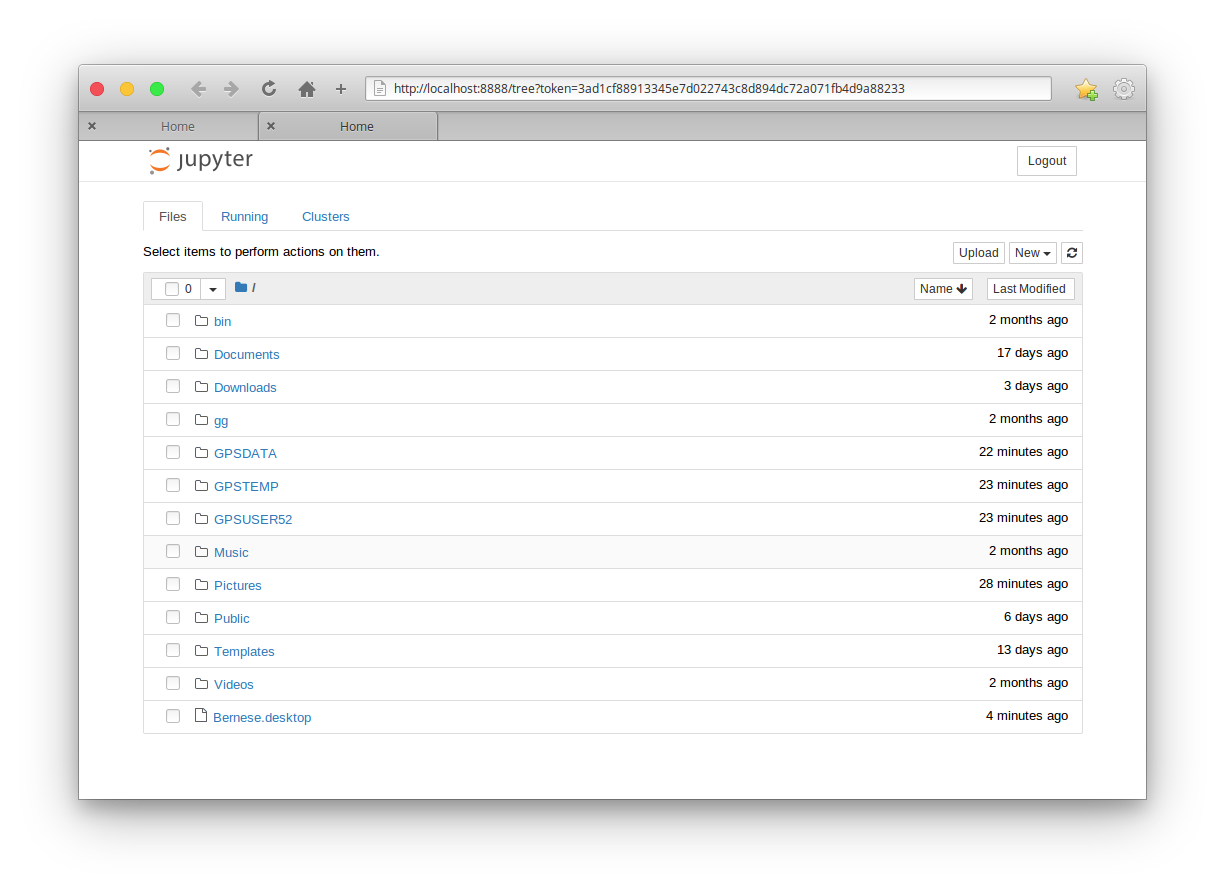
如果你希望在离线情况下完成部署，那么可以在 [UC Irvine](https://www.lfd.uci.edu/~gohlke/pythonlibs/) 的网站上下载所有模块及其依赖的程序包进行安装。例如，数据分析工具包 Pandas 依赖于 numpy、dateutil、pytz、setuptools 这些模块，那么在安装 Pandas 之前，你需要首先下载并安装 numpy、dateutil、pytz、setuptools 这些包。此时为了简化配置过程，强烈建议你直接安装 Anaconda。

**补充**

Jupyter Notebook（原 IPython Notebook）是一个非常优秀的数据分析工具，它提供的基于浏览器的交互式计算界面使用方便、功能强大，使用过程如行云流水。既可以在本地计算机进行计算，也能作为客户端操作远程工作站或者服务器。更棒的是它支持 Markdown 和 Latex 语法，可以在计算过程中插入文字描述或公式，然后轻松地将计算过程导出为文档或演示文稿使用。要体验 Jupyter Notebook 的魅力，可以在命令提示符中执行：

|  |  |
| --- | --- |
| 1 | $ jupyter notebook |

执行该命令后，程序将在浏览器中打开如下图所示的页面，该页面就是 Jupyter Notebook 提供的交互式计算界面。

[](http://www.gnss.help/images/try-jupyter-notebook.png)Jupyter Notebook

你可以在其中选择要进入的文件夹，然后点击 “New” 按钮创建一个基于 Python 3 环境的笔记本，进行交互式计算的体验。

**TPS2RIN 程序配置与使用方法**

[2017-12-03](http://www.gnss.help/2017/12/03/tps2rin-config-usage/)

[RINEX](http://www.gnss.help/categories/RINEX/)

[RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/), [TPS2RIN](http://www.gnss.help/tags/TPS2RIN/)

[TPS2RIN](http://positioning.topcon.com.cn/index.php?optionid=843&auto_id=700) 是由拓普康（Topcon）公司开发的一个命令行程序，该程序目前仅支持 Windows 操作系统，可以将拓普康旗下 GNSS 接收机输出的 .jps 或 .tps 格式观测数据转换为标准的 RINEX 2.10 格式。在格式转换过程中，还支持一些简单的数据编辑功能，如数据截取、卫星和观测量筛选、采样率变更以及文件头信息编辑等。

本文将介绍该程序的配置及使用方法。

**程序配置**

TPS2RIN 程序目前的最新版本为 9.5p1，从[拓普康官网的下载页面](http://positioning.topcon.com.cn/index.php?optionid=843&auto_id=700)可以获得该程序。下载并解压之后，你将得到一个可执行文件和一份软件使用说明书。打开 “命令提示符” 窗口，使用 cd 命令进入程序所在目录，执行 tps2rin 命令即可使用该程序。

当然，如果你希望在任何目录都能使用该程序，可以考虑将程序所在目录添加到系统的环境变量，或直接将 tps2rin.exe 文件移动至 “C:\Windows\System32” 文件夹内。

**操作模式**

为了便于后续操作的说明，首先介绍 TPS2RIN 程序的操作模式为：

|  |  |
| --- | --- |
| 1 | $ tps2rin <options> [file] |

其中的 [file] 即输入的 .jps 或 .tps 格式的 GNSS 接收机采集到的观测数据文件，<options> 为附加的参数选项。下文将按功能介绍所支持的 <options> 参数，你也可以查看文后的参数汇总表。

**格式转换**

使用 TPS2RIN 最简单的操作就是将 .jps 或 .tps 文件转换为 RINEX 2.10 标准格式，转换时使用 -o 参数指定文件的输出目录。若转换时未指定输出目录，则默认输出到运行程序时的目录。

示例，将 2017 年第 170 日观测的 test170a.jps 文件转换为 RINEX，并输出到 rinex/ 文件夹内：

|  |  |
| --- | --- |
| 1 | $ tps2rin -o rinex/ test170a.jps |

运行该命令后，将在 rinex/ 文件夹得到输出的观测文件 test170a.17o，星历文件 test170a.17n 和 test170a.17g。同时，程序在终端中也会打印相应的提示信息。你还可以通过参数来控制打印的提示信息：使用 -l 指定输出日志文件，使用 -e 显示转换过程中的统计信息（观测历元数、丢失历元数等），或者使用 -q 启动静默模式关闭提示。

示例，转换 test170a.jps 文件，输出日志文件并打印统计信息，将生成的 RINEX 文件和日志输出到当前目录：

|  |  |
| --- | --- |
| 1 | $ tps2rin -l -e test170a.jps |

使用静默模式，将 test170a.jps 转换为 RINEX 格式：

|  |  |
| --- | --- |
| 1 | $ tps2rin -q test170a.jps |

**信息编辑**

在进行数据转换时，你还可以进行一些简单的信息替换工作，将天线类型和天线高等添加到输出的 RINEX 观测文件中。处理大量数据时，最方便的办法是编写一个包含要写入的信息的配置文件。若使用 “<>” 包裹的内容代表变量，“//” 标识注释，则配置文件的格式可表示为：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | [TPS2RIN]  RUN BY = <name> // 运行程序人  COMMENT = <comment> // 注释  OBSERVER = <observer> // 观测者  AGENCY = <agency> // 观测机构  ANT # = <antenna number> // 天线号  ANT TYPE = <antenna type> // 天线类型  ANT DELTA H = <eccentricity height> // 天线高  ANT DELTA E = <eccentricity east> // 天线东偏  ANT DELTA N = <eccentricity north> // 天线北偏  MARKER NAME = <marker name> // 测量标志名  MARKER NUMBER = <marker number> // 测量标志号  POSITION X = <approx x> // 概略坐标 X  POSITION Y = <approx y> // 概略坐标 Y  POSITION Z = <approx z> // 概略坐标 Z |

撰写的配置文件一般以“.ini”作为扩展名，并在格式转换时使用 -p 参数引入。示例，在对 test170a.jps 做格式转换时，将配置文件 demo.ini 中的信息写入到输出的 RINEX 观测文件中：

|  |  |
| --- | --- |
| 1 | $ tps2rin -o rinex/ -p demo.ini test170a.jps |

当然，有些信息可能在以后略有变化，你可以使用参数 -M、-m、-A、-L 在运行时指定测量标志名、测量标志编号、天线类型或闰秒。如果你还同时指定了配置文件，那么最终的输出以运行时指定的信息为准。示例，将 test170a.jps 转换为 RINEX 格式，指定配置文件为 demo.ini，并使用 -A 参数覆盖输出观测文件中的天线类型：

|  |  |
| --- | --- |
| 1 | $ tps2rin -p demo.ini -p "TPSCR.G3 NONE" test170a.jps |

**数据编辑**

TPS2RIN 在数据转换时支持的数据编辑操作主要有数据时间窗提取、采样率抽取、观测量和观测卫星编辑。

**数据提取**

数据转换时使用 -s 和 -f 指定时间窗口边界，由此实现数据的提取。对于时间窗口边界的指定有两种方式（其中 “[]” 包裹的部分可以省略）：

* DD[HH[MM[SS]]]：使用日、时、分、秒的方式指定时间窗边界，从左到右每 2 个数字一节；
* DD[mm[YY]]d[HH[MM[SS[.SS]]]]：使用日、月、年、时、分、秒的方式指定时间窗边界，这里使用 “d” 来作为日期和时间的界限，各自从左到右每 2 个数字一节。

示例，源文件 demo170a.tps 的观测时间为 2017 年 6 月 19 日 0 点至 1 点。将其中 0:00 点至 0:30 之间观测的部分作为 RINEX 格式输出，使用第 1 种方式为：

|  |  |
| --- | --- |
| 1 | $ tps2rin -s 100000 -f 100030 demo170a.tps |

使用第 2 种方式为：

|  |  |
| --- | --- |
| 1 | $ tps2rin -s 19d0000 -f 19d0030 demo170a.tps |

**采样率抽取**

采样率较高的数据会占据更大的磁盘容量，在精度允许的情况下减小采样率可以节约磁盘的使用。TPS2RIN 程序使用 -I或 -d 参数来更改输出文件的采样间隔。两个参数的区别在于输出文件的采样率的计算方式：对于 -I 参数，输出文件的采样率为源文件采样率和该参数设定值的最小公倍数；而 -d 参数指定的则是源文件每隔多少个观测历元输出一个观测值。

示例，对于采样间隔为 10 秒的源文件 demo170a.tps，执行下面的命令将得到采样间隔为 30 秒的观测文件：

|  |  |
| --- | --- |
| 1 | $ tps2rin -I 6 demo170a.tps # 6 与 10 的最小公倍数为 30 |

执行下面的命令则会得到采样间隔为 60 秒的观测文件：

|  |  |
| --- | --- |
| 1 | $ tps2rin -d 6 demo170a.tps # 10 \* 6 = 60 |

**观测量编辑**

某些观测量可能不是你需要的，你可以在格式转换时对数据进行编辑以去掉某些观测量。可用于删除观测量的参数有：

* -1：去掉 L1 波段的所有观测；
* -2：去掉 L2 波段的所有观测；
* -C2：去掉 L2 波段上的 L2C 观测量；
* -D：去掉多普勒观测信息。

示例，在数据转换时删除多普勒观测信息和 L2C 观测量：

|  |  |
| --- | --- |
| 1 | $ tps2rin -C2 -D demo170a.tps |

另外，你还可以选择在数据转换时输出某些信息（主要是多普勒观测量和接收机钟差），或者对伪距观测量进行平滑以削弱多路径效应的影响。使用这些功能所用的参数为：

* -O：在输出的 O-文件中包含接收机钟差；
* -S：输出多普勒观测量，单位为 dB\*Hz；
* -~：实行伪距平滑。

示例，将 demo170a.tps 转换为 RINEX 格式，转换时输出接收机钟差并进行伪距平滑：

|  |  |
| --- | --- |
| 1 | $ tps2rin -O -~ demo170a.tps |

**观测卫星筛选**

在执行数据格式转换操作时，你也可以对观测卫星进行筛选以便去掉不需要的导航卫星或卫星系统。与筛选观测卫星相关的参数有 6 个，你可以直接删除某个卫星系统的所有卫星，也可以删除指定 PRN 的某个卫星的观测：

* -G：去掉所有 GPS 卫星的观测；
* -g <prn>：去掉 PRN 为 n 的某颗 GPS 卫星的观测；
* -R：去掉所有 GLONASS 卫星的观测；
* -r <prn>：去掉 PRN 为 n 的某颗 GLONASS 卫星的观测；
* -W：去掉所有 WAAS（广域增强系统）卫星的观测；
* -w <prn>：去掉 PRN 为 n 的某颗 WAAS 卫星的观测；

示例，将 demo170a.tps 转换为 RINEX 格式，转换时去掉所有 WAAS、GLONASS 卫星以及 GPS 卫星系统中的 4 号星的观测量：

|  |  |
| --- | --- |
| 1 | $ tps2rin -R -W -g 4 demo170a.tps |

**完整参数表**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | -h # 显示帮助信息  -o <directory> # 文件输出目录，缺省为当前路径  -p <profile> # 配置文件路径，键入 TPS2RIN –p ? 显示更多帮助  -s <datetime> # 开始历元时刻，键入 TPS2RIN –s ? 显示更多帮助  -f <datetime> # 结束历元时刻，键入 TPS2RIN –f ? 显示更多帮助  -G # 删除 GPS 卫星  -g <prn> # 删除某颗 GPS 卫星，prn 为卫星号  -R # 删除 GLONASS 卫星  -r <prn> # 删除某颗 GLONASS 卫星，prn 为卫星号  -W # 删除 WAAS 卫星  -w <prn> # 删除某颗 WAAS 卫星，prn 为卫星号  -1 # 删除 L1 载波频率  -2 # 删除 L2 载波频率  -C2 # 删除 L2C 信号  -l # 写日志文件  -q # 静默模式  -e # 观测历元信息统计  -I <ss.ss> # 间隔多少秒输出一个观测值，键入 TPS2RIN -I ? 显示更多帮助  -d <number> # 间隔多少历元输出一个观测值，键入 TPS2RIN -d ? 显示更多帮助  -L # 闰秒信息，缺省值为 15s  -D # 删除多普勒信息  -O # 输出接收机钟差  -S # 输出观测值 S1，S2。 单位：dB\*Hz  -~ # 执行伪距平滑  -M <name> # 覆盖配置文件或其它输入文件中的点名  -m <number> # 覆盖配置文件或其它输入文件中的点号  -A # 覆盖天线类型名；键入 TPS2RIN -A ? 显示更多帮助  -a # 覆盖配置文件或其它输入文件中的天线序列号  -= # 忽略用户事件，键入 TPS2RIN -= ? 显示更多帮助 |

**补充**

除了使用 TPS2RIN 程序外，TEQC 程序也是数据转换的一种选择，并且 TEQC 所支持的接收机数据格式远比 TPS2RIN 丰富。要查看 TEQC 数据格式转换功能的使用方法，可以移步本站 [TEQC 配置及其数据转换功能的使用](http://www.gnss.help/2016/09/17/teqc-config-translate/)一文。

**有关购买 Bernese 软件你可能想知道的**

[2017-10-23](http://www.gnss.help/2017/10/23/buy-bernese-authorization/)

[BERNESE](http://www.gnss.help/categories/Bernese/)

[BERNESE](http://www.gnss.help/tags/Bernese/)

[Bernese](http://www.bernese.unibe.ch/) 是著名的高精度 GNSS 数据处理软件之一，该程序由[瑞士伯尔尼大学天文学院](http://www.aiub.unibe.ch/)开发，能处理 GPS、GLONASS 或 Galileo 系统的观测数据。既可用于地面点的定位，也可用于 GNSS 或低轨卫星的定轨。其处理方式包含非差精密单点定位（PPP）和双差相对定位两种方式，功能强大。

要使用未开放的 Bernese 软件，需要向伯尔尼大学支付费用获得授权。本文将介绍购买 Bernese 软件的相关信息，希望可以向对该软件感兴趣的用户提供一些帮助。

**产品形式**

Bernese 软件主要使用 Fortran 和 Perl 语言开发，并兼容 Windows、UNIX/Linux 等操作系统，软件的[支持页面](http://www.bernese.unibe.ch/support/#TEST)列出了目前经过测试的操作系统平台（目前最新版本已支持 Windows 10）。

购买该软件后，你将获得源代码和预编译的用于 Windows 平台的可执行程序。如果你希望在 UNIX/Linux 系统运行该程序，你可以使用支持的编译器将程序编译安装到计算机上。当然，如果你愿意，在 Windows 系统上也可以编译该程序。但在 UNIX/Linux 系统上运行有更多优势，比如可以在多台计算机之间分配任务。

**软件版本**

目前可以购买的软件主版本号为 Bernese 5.2。购买该程序后，伯尔尼大学天文学院将为你注册一个用户，你可以从其网站上免费下载针对该版本（例如 5.2）的更新和支持。

如果在购买 Bernese 软件 1 年半的时间内，Bernese 发布了新的主版本（例如 Bernese 5.3），你可以获得免费升级。但如果购买日期在 1 年半之前，则需要额外地支付 1500 瑞士法郎才能获得最新主版本的升级。

**售价**

**培训课程**

针对已购买 Bernese 软件的用户或对软件有兴趣的潜在用户，伯尔尼大学天文学院每年在伯尔尼举办两次培训。培训时长为 5 天，在此期间，你可以了解程序的功能和特性。培训费为每人 1200 瑞士法郎（CHF, Swiss Francs）。

**科研用途**

对于科研机构，Bernese 软件的授权费用为 12000 瑞士法郎，这笔费用还包含了两人的培训（价值 2400 瑞士法郎）。若不参加培训，则只需支付 9600 瑞士法郎。使用科研授权的软件可安装到多台计算机上。

**商业用途**

对于商业授权，授权费用依安装的计算机数不同：

* 1 台计算机：24000 瑞士法郎，含 1 人的培训费用；
* 2 台计算机：32000 瑞士法郎，含 2 人的培训费用；
* 多台计算机：36000 瑞士法郎，含 2 人的培训费用。

同样，如果不参加培训，也可以节约相应的培训费。

**支付方式**

伯尔尼大学只接受电子汇款，不接受其他的付款方式。为了避税，你可以选择由接收者进行货币兑换，否则还需额外缴纳约 10% 左右的税费。

**试用版**

Bernese 软件由伯尔尼大学天文学院开发，维护一份额外的试用版超出了其资源和精力。因此，Bernese 软件没有试用版本。对于有兴趣购买 Bernese 的用户，可以前往伯尔尼参加培训（上文已提到），并在此期间了解程序的功能和特性。

**传递方式**

如果你购买了 Bernese 软件，其传递方式较为灵活。你可以选择以下任意一种方式：

* 邮寄包含该程序的 CD/DVD 光盘；
* 通过网络传递，例如使用 FTP 等媒介接收或下载。

毫无疑问，第二种方式更快速，同时也能免去海关等复杂的手续，这也是伯尔尼大学推荐的方式。

**联系人**

任何有关获取和购买 Bernese 软件授权方面的疑问，你都可以通过电子邮件的方式向伯尔尼大学天文学院咨询。受理有关 Bernese 软件授权事宜的电子信箱为 [bernese@aiub.unibe.ch](mailto:bernese@aiub.unibe.ch)。

# 使用 GAMIT 解算北斗或 Galileo 数据

[2017-10-01](http://www.gnss.help/2017/10/01/gamit-solve-bds-galileo/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/)

GAMIT/GLOBK 程序自 10.5 版本开始，就逐步加入对其他各大 GNSS 系统（北斗、Galileo）的支持。随着 10.61 版本的发布，该程序终于能够支持北斗观测数据的解算，并且附带的共用表文件也包含了多个 GNSS 系统的信息。经过实测，使用新版本的 GAMIT/GLOBK 程序处理北斗或 Galileo 观测数据的过程比较顺畅。

本文介绍使用 GAMIT/GLOBK 程序处理北斗或 Galileo 观测数据的过程。使用其最新的 10.61 版本，并安装了于 2017 年 8 月 8 日发布的更新包：incremental\_updates.170808.tar.gz。

## 文件组织

本次解算示例使用了 MGEX 项目于 2017 年 7 月 30 日观测的多系统 GNSS 数据，包括：多系统的混合广播星历文件 brdm2110.17p，武汉大学发布的混合多系统精密星历 wum19600.sp3 以及来自 ABMF、AGGO 等多个 MGEX 连续观测站的 RINEX 3 格式观测文件。整个数据处理项目的文件结构如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | · demo/  |—— 211/  |—— brdc/  |—— brdm2110.17p  |—— igs/  |—— wum19600.sp3  |—— rinex/  |—— ABMF00GLP\_R\_20172110000\_01D\_30S\_MO.rnx  |—— AGGO00ARG\_R\_20172110000\_01D\_30S\_MO.rnx  |—— CAS100ATA\_R\_20172110000\_01D\_30S\_MO.rnx  |—— DAEJ00KOR\_R\_20172110000\_01D\_30S\_MO.rnx  |—— DUND00NZL\_R\_20172110000\_01D\_30S\_MO.rnx  |—— HKWS00HKG\_R\_20172110000\_01D\_30S\_MO.rnx  |—— LHAZ00CHN\_R\_20172110000\_01D\_30S\_MO.rnx  |—— tables/ |

在解算文件夹 demo/ 内，平行创建 5 个子文件夹：211/ 和 tables/ 文件夹为空；brdc/ 文件夹包含多系统广播星历文件 brdm2110.17p；igs/ 文件夹包含多系统精密星历文件 wum19600.sp3；rinex/ 文件夹包含 7 个 MGEX 站的观测文件。从 IGS 服务器上获取到的原始观测文件使用了 RINEX 3 格式的文件名，需要使用 sh\_rename\_rinex3 命令将其重命名为 RINEX 2 格式的文件名：

|  |  |
| --- | --- |
| 1 | $ sh\_rename\_rinex3 -f rinex/\*.rnx -r |

要详细了解 sh\_rename\_rinex3 命令，请移步至 [GAMIT 中的 RINEX 3 文件重命名脚本](http://www.gnss.help/2017/09/13/sh-rename-rinex3/)。

## 参数配置

### 链接表文件

首先链接所需的共用表文件，该命令已介绍过多次，不在赘述。使用的命令为：

|  |  |
| --- | --- |
| 1 | $ sh\_setup -yr 2017 -doy 211 |

在解算目录（demo/）内执行该命令，将对应日期的共用表文件链接至 tables/ 文件夹内。

### 参数配置

在 sestbl. 文件中：

Use otl.list = Y           ; Ocean tidal loading list file from OSO  
Use otl.grid = N         ; Ocean tidal loading grid file, GAMIT-format converted from OSO

这两项是 GAMIT 基线解算中所需的海潮改正文件的配置，详情可参考 [GAMIT 中的 GRID 文件简介](http://www.gnss.help/2016/10/02/grids-in-gamit/)。

## 开始解算

至此，所有的准备工作都已完成，我们可以开始数据解算操作了。以下的所有操作都将在年积日文件夹（211/）执行。

### 链接文件

使用以下的命令将所需的广播星历、精密星历、观测数据和共用表文件链接至年积日文件夹内：

|  |  |
| --- | --- |
| 1  2  3  4 | $ ln -s ../brdc/brdm2110.17p .  $ ln -s ../igs/wum19600.sp3 .  $ ln -s ../rinex/\*.17o .  $ ln -s ../tables/\* . |

### 执行解算

首先使用 sh\_makexp 命令生成解算所需的部分文件。相比处理 GPS 数据时的操作，这里使用了 -gnss C 参数指定解算的 GNSS 系统为北斗：

|  |  |
| --- | --- |
| 1 | $ sh\_makexp -expt bdst -orbt wumf -yr 2017 -doy 211 -gnss C -sess 0 -nav brdm2110.17p -jclock sp3 -apr lfile. -sinfo 30 00 00 2880 |

然后，使用 sh\_sp3fit 命令进行轨道拟合，生成可用的 G-文件。这里同样使用 -gnss C 参数将解算的 GNSS 系统指定为北斗：

|  |  |
| --- | --- |
| 1 | $ sh\_sp3fit -f wum19600.sp3 -o wumf -d 2017 211 -gnss C |

下面将检查生成的 G-文件，重新生成任务配置文件 session.info。这一步将会从解算任务中剔除精密星历未包含的卫星，对于北斗和 Galileo 来说都是必要的：

|  |  |
| --- | --- |
| 1 | $ sh\_check\_sess -sess 211 -type gfile -file gwumf7.211 |

上面的命令中，通过 -sess 参数指定数据观测的年积日，-type 参数指定要检查的文件类型，-file 命令指定要检查的文件。

下一步为生成卫星钟文件，这里的最后一个字符 C 将 GNSS 系统指定为北斗：

|  |  |
| --- | --- |
| 1 | $ makej brdm2110.17p jwumf7.211 wum19600.sp3 C |

检查生成的 J-文件：

|  |  |
| --- | --- |
| 1 | $ sh\_check\_sess -sess 211 -type jfile -file jwumf7.211 |

后面的步骤都很熟悉了，生成双差观测数据：

|  |  |
| --- | --- |
| 1 | $ makex bdst.makex.batch |

生成解算输入参数和批处理命令：

|  |  |
| --- | --- |
| 1 | $ fixdrv dbdst7.211 |

启动数据处理脚本：

|  |  |
| --- | --- |
| 1 | $ csh bbdst7.bat |

## 补充

解算 Galileo 系统的观测数据与处理北斗数据大同小异，也非常简单：直接将本文用到的指定北斗系统的字符 C 替换为 E 即可。

目前 GAMIT/GLOBK 程序尚不能同时解算多个 GNSS 系统的数据。如果你指定了多个卫星系统，实际上只会处理第一个卫星系统的数据。另外，若使用间隔 5 分钟的精密星历（GBM，TUM 等），执行 sh\_sp3fit 时很可能会出错。建议使用 COM、WUM 等机构发布的间隔 15 分钟的精密星历。

**GNSSCAL 使用文档**

[2017-09-25](http://www.gnss.help/2017/09/25/gnsscal-readme/)

[PYTHON](http://www.gnss.help/categories/Python/)

[GNSSCAL](http://www.gnss.help/tags/GNSSCAL/), [PYTHON](http://www.gnss.help/tags/Python/)

公历（格里高利历）是我们实际生活中使用的历法。但对于 GNSS 数据处理而言，其数据的观测日期通常以年、年积日的方式表示；而产品所对应的日期则常常以 GPS 周、周内日的方式表示。我们经常需要将日期在这几种表示方法之间切换。

本文将介绍一个 Python 库：gnsscal，它既可以作为模块导入，亦提供命令行操作方式。使用这个库可以方便快捷地查询公历、年积日和 GPS 周、BDS 周等对应的信息。

**简介**

[gnsscal](https://pypi.python.org/pypi/gnsscal/1.1.0) 是一个 Python 模块，该模块提供处理 GNSS 数据所需的日期的一些常量和函数。模块支持 Python 2 和 Python 3，代码托管在 [Github](https://github.com/purpleskyfall/gnsscal) 网站上。

**安装与更新**

gnsscal 支持使用 pip 进行安装，安装命令为：

|  |  |
| --- | --- |
| 1 | $ pip install gnsscal |

上面的命令将从 [PyPI 仓库](https://pypi.python.org/pypi/gnsscal/1.1.0)下载程序包，并将其加入到你的程序库中。要更新 gnsscal，可使用命令：

|  |  |
| --- | --- |
| 1 | $ pip install --upgrade gnsscal |

**命令行模式**

安装 gnsscal 之后，你可以通过命令行模式来使用它：

|  |  |
| --- | --- |
| 1 | $ python -m gnsscal [options] |

这里的 [options] 主要用于指定输入的日期，你可以使用如下参数：

* -date <year> <month> <day>：通过年、月、日的方式指定输入的日期；
* -ydoy <year> <doy>：通过年、年积日的方式指定输入的日期；
* -gpswd <week> <dow>：通过 GPS 周、周内日的方式指定日期；
* -bdswd <week> <dow>：通过 BDS 周、周内日的方式指定日期。
* -v、--version：显示程序版本信息；
* -h、--help：显示帮助信息。

示例，输入年、月、日，查看该日对应的年积日、GPS 周、BDS 周等信息：

|  |  |
| --- | --- |
| 1 | $ python -m gnsscal -date 2017 9 25 |

输入年和年积日，查看对应的公历和 GPS 周、BDS 周等信息：

|  |  |
| --- | --- |
| 1 | $ python -m gnsscal -ydoy 2017 268 |

输入 GPS 周和周内日，查看对应的信息：

|  |  |
| --- | --- |
| 1 | $ python -m gnsscal -gpswd 1968 1 |

输入 BDS 周和周内日，查看对应的信息：

|  |  |
| --- | --- |
| 1 | $ python -m gnsscal -bdswd 612 1 |

以上示例的输出都是相同的：

Gregorian date: 2017-09-25  
year, doy: 2017, 268  
GPS week: 1968, 1  
BDS week: 0612, 1

**作为模块**

gnsscal 亦可以作为模块导入，这样你就可以使用其提供的函数库。目前，gnsscal 模块提供了两个常量和一系列的函数。你可以直接调用它们而不需要重新发明轮子。

**常量**

模块中定义的两个常量为：

* **GPS\_START\_DATE**  
  GPS 周计算的起点，该值是一个 datetime.date 型的常量，定义为 1980 年 1 月 6 日。
* **BDS\_START\_DATE**  
  BDS 周计算的起点，该值是一个 datetime.date 型的常量，定义为 2006 年 1 月 1 日。

作为示例的交互式控制台代码：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> from gnsscal import \*  >>> GPS\_START\_DATE  datetime.date(1980, 1, 6)  >>> BDS\_START\_DATE  datetime.date(2006, 1, 1) |

**公历与年积日转换**

* **date2doy(date)**  
  将公历转换为年积日，返回一个整数。
* **date2yrdoy(date)**  
  将公历转换为年和年积日，返回一个包含两个整数的元祖。
* **yrdoy2date(year, doy)**  
  将年和年积日转换为公历，返回一个 datetime.date 型的实例。

作为示例的交互式控制台代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | >>> from datetime import date  >>> from gnsscal import \*  >>> date2doy(date(2017, 5, 17))  137  >>> date2yrdoy(date(2017, 5, 17))  (2017, 137)  >>> yrdoy2date(2017, 137)  datetime.date(2017, 5, 17) |

**公历与 GPS/BDS 周转换**

* **date2bdswd(date)**  
  将公历转换为 BDS 周和周内日，返回一个包含两个整数的元组。
* **date2gpswd(date)**  
  将公历转换为 GPS 周和周内日，返回一个包含两个整数的元组。
* **bdswd2date(bdsweek, dayofweek)**  
  将 BDS 周和周内日转换为公历，返回一个 datetime.date 型的实例。
* **gpswd2date(gpsweek, dayofweek)**  
  将 GPS 周和周内日转换为公历，返回一个 datetime.date 型的实例。

作为示例的交互式控制台代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | >>> from datetime import date  >>> from gnsscal import \*  >>> date2bdswd(date(2017, 5, 17))  (593, 3)  >>> date2gpswd(date(2017, 5, 17))  (1949, 3)  >>> bdswd2date(593, 3)  datetime.date(2017, 5, 17)  >>> gpswd2date(1949, 3)  datetime.date(2017, 5, 17) |

**年积日与 GPS/BDS 周转换**

* **bdswd2yrdoy(bdsweek, dayofweek)**  
  将 BDS 周和周内日转换为年和年积日，返回一个包含两个整数的元组。
* **gpswd2yrdoy(gpsweek, dayofweek)**  
  将 GPS 周和周内日转换为年和年积日，返回一个包含两个整数的元组。
* **yrdoy2gpswd(year, doy)**  
  将年和年积日转换为 GPS 周和周内日，返回一个包含两个整数的元组。
* **yrdoy2bdswd(year, doy)**  
  将年和年积日转换为 BDS 周和周内日，返回一个包含两个整数的元组。

作为示例的交互式控制台代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | >>> from gnsscal import \*  >>> bdswd2yrdoy(593, 3)  (2017, 137)  >>> gpswd2yrdoy(1949, 3)  (2017, 137)  >>> yrdoy2gpswd(2017, 137)  (1949, 3)  >>> yrdoy2bdswd(2017, 137)  (593, 3) |

**BDS/GPS 周相互转换**

* **bdsw2gpsw(bdsweek)**  
  将 BDS 周转换为 GPS 周，返回一个整数。
* **gpsw2bdsw(gpsweek)**  
  将 GPS 周转换为 BDS 周，返回一个整数。

作为示例的交互式控制台代码：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> from gnsscal import \*  >>> bdsw2gpsw(593)  1949  >>> gpsw2bdsw(1949)  593 |

**测试**

如果你需要测试 gnsscal，其测试代码放在 test\_gnsscal.py 中。你可以通过执行它来测试位于相同目录下的 gnsscal.py 程序：

|  |  |
| --- | --- |
| 1 | $ python test\_gnsscal.py |

**补充**

如果你仅仅想要查看公历和年积日、GPS 周之间的相对应信息，访问 [GNSS Calendar](http://gnsscalendar.com/) 网站也是一个不错的选择。从该网站还可以方便地下载 IGS 提供的产品。另外，如果你需要一个离线可用的 GUI 应用，也可以尝试一下本站[下载](http://www.gnss.help/download/)页面所提供的 “GPS 日历”程序。

**GAMIT 的 RINEX 3 文件重命名脚本**

[2017-09-13](http://www.gnss.help/2017/09/13/sh-rename-rinex3/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/)

最新的 GAMIT/GLOBK 程序已经能够处理 RINEX 3 格式的文件输入。我们可以在不进行格式转换的情况下，直接对 RINEX 3 的观测数据或卫星星历进行处理。但美中不足的是：目前的程序尚不能支持 RINEX 3 格式的文件名。为了处理这个问题，GAMIT/GLOBK 10.61 提供了 sh\_rename\_rinex3 脚本。该脚本可以将 RINEX 3 格式的文件名重命名为 RINEX 2 格式的文件名。

本文将介绍 sh\_rename\_rinex3 脚本的使用方法。

**程序使用**

脚本命令 sh\_rename\_rinex3 的使用方法为：

|  |  |
| --- | --- |
| 1 | $ sh\_rename\_rinex3 -f <files> [-d <dir>] [-c -l -r] |

参数释义：

* -f <files>：要处理的文件名；
* -d <dir>：文件的输出路径；
* -c：以拷贝的方式重命名；
* -l：以链接的方式重命名（默认）；
* -r：以剪切的方式重命名。

该脚本的前两个参数指定输入输出的路径，后三个参数指定文件重命名的方式。默认的情况下，会输出到当前目录并以创建链接的方式进行重命名。

示例，在 rinex3/ 文件夹内保存如下文件：

* AGGO00ARG\_R\_20172110000\_01D\_30S\_MO.rnx
* ALGO00CAN\_R\_20172110000\_01D\_30S\_MO.rnx
* BAKE00CAN\_R\_20172110000\_01D\_30S\_MO.rnx
* HKWS00HKG\_R\_20172110000\_01D\_30S\_MO.rnx
* LHAZ00CHN\_R\_20172110000\_01D\_30S\_MO.rnx

将 rinex3/ 文件夹内的文件重命名为 RINEX 2 的文件名，并输出到 rinex2/ 目录内，重命名的方式为创建链接：

|  |  |
| --- | --- |
| 1 | $ sh\_rename\_rinex3 -f rinex3/\*.rnx -d rinex2/ |

运行上面的命令之后，rinex2/ 文件夹内的内容如下所示：

* aggo2110.17o；
* algo2110.17o；
* bake2110.17o；
* hkws2110.17o；
* lhaz2110.17o。

类似的，将 rinex3/ 目录内的文件重命名为 RINEX 2 的文件名，并输出到当前目录内，重命名的方式为拷贝：

|  |  |
| --- | --- |
| 1 | $ sh\_rename\_rinex3 -f rinex3/ -c |

**补充**

需说明的是：拷贝方式的重命名是最耗时间的，除非必须要这么做，否则应将链接或剪切的方式作为首选。另外，该脚本目前不能很好地对 RINEX 3 的广播星历文件（Nav）重命名：其输出的文件名总是以 “o” 结尾。因此在处理 RINEX 3 格式的数据时，你需要手动将 RINEX 3 格式的广播星历文件重命名为 RINEX 2 的形式。

将所有 RINEX 3 形式的文件名重命名之后，你就可以开始数据处理工作了。操作流程和处理 RINEX 2 格式的数据时完全一致。

**从 GAMIT/GLOBK 解算结果导出产品**

[2017-09-05](http://www.gnss.help/2017/09/05/export-products-from-gamit/)

[GLOBK](http://www.gnss.help/categories/GLOBK/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/), [GLOBK使用](http://www.gnss.help/tags/GLOBK%E4%BD%BF%E7%94%A8/)

熟悉 GAMIT/GLBOK 的读者可能知道：该软件不仅可以用来解算地面 GNSS 测站的位置，还能用来解算 GNSS 卫星的轨道。如果你有足够多且遍布全球的 GNSS 数据接收站，就可以使用它来做 GNSS 卫星定轨。并且在解算完成后将你的解算成果保存为通用的 IGS 产品格式。

本文将简介将 GAMIT/GLOBK 程序的解算结果导出为 IGS 产品的命令。主要包括两种操作：导出 SP3 精密星历文件和 SINEX 解算结果文件。

**导出 SP3 文件**

GAMIT/GLBOK 程序内部使用的星历主要为二进制的 T-文件，我们可以将其解算输出的 T-文件导出使用。该过程涉及的命令为 ttongs，该程序将 GAMIT 输出的星历表文件（T-文件）转化为标准的 SP3 精密轨道文件。

ttongs 命令接受的参数为：

|  |  |
| --- | --- |
| 1 | $ ttongs <tfile> <fmt> <org> <type> <frame> <otlmod> <pcvmod> <filename> [<clkfile> <start> <stop>] |

参数释义：

* <tfile>：作为输入的 T-文件；
* <fmt>：输出文件格式（SP1 或 SP3）；
* <org>：产品解算的机构；
* <type>：轨道类型（FIT、EXT 或 BRD）；
* <frame>：5 字符的轨道坐标系；
* <otlmod>：所使用的海潮文件模型；
* <pcvmod>：所使用的天线相位中心改正模型；
* <filename>：输出文件的文件名；
* <clkfile>：输入的精密钟差文件（可选）；
* <start>：输出文件的开始时刻（可选）；
* <stop>：输出文件的结束时刻（可选）。

示例，将 tdemo7.211 文件转化为 SP3 文件 dem19600.sp3。其中坐标系为地固坐标系 IGS14，该坐标系为地心地固系（FIT），海潮模型为 FES2004，天线相位中心改正模型为 IGS14\_1958：

|  |  |
| --- | --- |
| 1 | $ ttongs tdemo7.211 sp3 institute fit IGS14 FES2004 IGS14\_1958 dem19600.sp3 |

以上的命令输出的 SP3 文件的时长可能不止一天，这或许不是你想要的。你可以通过后面的参数来指定文件的起止时刻。但请注意：该程序接收的所有参数都是有顺序的，如果你希望忽略中间的项，必须其位置处使用 ' ' 代替。例如，指定输出的起止时刻，但不输入精密钟差文件：

|  |  |
| --- | --- |
| 1 | $ ttongs tdemo7.211 sp3 institute fit ITR08 FES2004 IGS14\_1958 dem19600.sp3 ' ' 2017 211 0 0 2017 211 23 45 |

**导出 CLK 文件**

GAMIT/GLOBK 程序默认不解算卫星钟差，解算卫星钟差的设置在 autcln.cmd 文件中。该文件中有以下两个配置项目：

* apply\_chs\_clk：解算卫星钟差时的配置；
* igs\_clk：输出钟差解算结果的配置。

这两个项目的配置方式如下：

|  |  |
| --- | --- |
| 1  2 | apply\_phs\_clk [MAX Iter] [Non-int] [Converged] [Over shoot]  igs\_clk [clock-file name] [min rms] [max rms] [List of reference clocks] |

在 apply\_phs\_clk 中，所有配置项都是可选的：MAX Iter 为解算时的最大迭代次数，该项的默认值为 30；Non-int 为估计钟差时应用非整数参数前的迭代次数，该项默认为 3；Converged 为百分比形式的所允许的每次迭代时单向残差 RMS 变化量；Over shoot 用于控制收敛速度，其默认值为 1.5，大于 2 将可能导致估计时出错。

在 igs\_clk 配置项中，clock-file name 为所输出的 CLK 文件的 3 个前缀字符，程序将根据这 3 个字符加上 GPS 周等生成最终的文件名；在没有给出参考钟时，min rms 用于设置所使用的参考钟的上限，其单位为周期数，对于氢钟的推荐值为 20；max rms 用于设置在输出测站钟差时的 RMS 值的上限，高于该值的钟差不会输出（一般设置为 200 即可输出大部分的测站钟差）；最后的 List of reference clocks 用于自定义作为参考钟的测站列表，该项是可选的。

配置示例：

apply\_phs\_clk 30 3 0.1 1.5  
igs\_clk mit 20 200

**导出 SINEX 文件**

**单个导出**

将 GAMIT/GLOBK 解算输出的二进制 H-文件导出为 SINEX，可使用 glbtosnx 命令，该程序以 .glb 或 .glx 文件作为输入数据。glbtosnx 程序接受的参数为：

|  |  |
| --- | --- |
| 1 | $ glbtosnx <dir> <comments> <input> <output> [-h -s] |

参数释义：

* <dir>：SINEX 文件的输出文件夹；
* <comments>：添加到 SINEX 中的注释文件；
* <input>：输入的 GLOBK 二进制 H-文件；
* <output>：输出的 SINEX 文件名。

同上文一样，如果你希望忽略中间的项，也可以使用 ' ' 代替。示例，将当前目录下的 h1707301200\_demo.glx 导出为 dem19600.snx：

|  |  |
| --- | --- |
| 1 | $ glbtosnx . ' ' h1707301200\_demo.glx dem19600.snx |

**批量导出**

如果你有多个二进制 H-文件需要转化，你可以试试 sh\_glbtosnx 命令。该程序可以查找所有待转化的二进制 H-文件，然后生成一个可执行脚本。我们之后执行这个脚本即可实现批量操作。该命令的使用方式为：

|  |  |
| --- | --- |
| 1 | $ sh\_glbtosnx -f <files> -ext <extent> |

参数释义：

* -f <files>：待转化的 GLOBK 二进制 H-文件名；
* -ext <extent>：H-文件的扩展名。

示例，首先执行 sh\_glbtosnx 命令来生成一个批量转化脚本：

|  |  |
| --- | --- |
| 1 | $ sh\_glbtosnx -f h\* -ext glx > glbtosnx.bat |

上面的命令将搜索所有以 h 开头并以 glx 为扩展名的文件，生成一个用来转化它们的可执行脚本：glbtosnx.bat。接下来，我们就可以执行这个批量转化脚本将所有符合条件的二进制 H-文件转化为 SINEX 解文件：

|  |  |
| --- | --- |
| 1 | $ csh glbtosnx.bat |

**使用广播星历进行 GAMIT 解算**

[2017-08-28](http://www.gnss.help/2017/08/28/gamit-only-brdc/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/)

GAMIT/GLOBK 作为一个高精度的 GNSS 数据处理程序，在数据处理时通常需要将 IGS 发布的精密轨道产品作为输入。但有时我们可能并没有相应的精密轨道产品可供使用，只能退而求其次，采用广播星历。GAMIT 软件可以满足这一需求，但是在操作上与通常情况有些差异。

这一篇博文将以操作演示的形式，介绍只使用广播星历进行数据解算的过程。操作时所用的 GAMIT/GLOBK 程序的版本号为 10.61。

**数据准备**

开始数据解算前我们依然要首先进行数据准备工作。我在本次解算时采用了 BJFS、CHAN、DAEJ、SHAO、SUWN 和 WUH2 共 6 个 IGS 站在 2017 年第 192 日的观测文件，并采用对应日期的广播星历。文件的组织方式为：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | · demo/  |—— 192/  |—— brdc/  |—— brdc1920.17n  |—— rinex/  |—— bjfs1920.17o  |—— chan1920.17o  |—— daej1920.17o  |—— shao1920.17o  |—— suwn1920.17o  |—— wuh21920.17o  |—— tables/ |

在解算目录 demo/ 内平行地创建 4 个子文件夹：其中 brdc/ 文件夹存放广播星历文件；rinex/ 文件夹存放各站点的观测文件；tables/ 文件夹用于存放解算所需的共用表文件；而年积日文件夹（192/）用于解算操作，所有结果文件和中间本文都保存在该文件夹内。

**参数设置**

**链接表文件**

无论做哪一种数据处理任务，执行操作之前，首先需要做的就是链接数据解算所需的共用表文件。在解算目录下运行 sh\_setup 命令，将共用表文件链接至解算目录的 tables/ 文件夹内：

|  |  |
| --- | --- |
| 1 | $ sh\_setup -yr 2017 -doy 192 |

**参数设置**

执行上面的 sh\_setup 命令之后，解算所需的配置文件也已经被拷贝至 tables/ 文件内，你可以通过编辑它们对解算进行设置。这里我只是用于演示目的，因此采用默认配置，不做任何修改。

**解算操作**

需要说明的一点是：虽然在 sh\_gamit 命令中可以通过 -orbt 选项来指定解算所需的轨道文件，但我测试后发现，该命令此时不太好用。如果使用该命令解算，将遭遇很多错误。并且，使用广播星历解算时使用的命令与通常情况并不十分一致。因此，我在实际操作中使用分步运行的方式。

**链接文件**

首先，使用 UNIX/Linux 系统的 ln -s 命令，将准备好的文件链接至年积日文件夹（192/）内。在年积日文件夹执行：

|  |  |
| --- | --- |
| 1  2  3 | $ ln -s ../brdc/\*n .  $ ln -s ../rinex/\*o .  $ ln -s ../tables/\* . |

**开始解算**

现在，就可以执行任务的初始化等准备工作了。这一步骤可以使用 makexp 或 sh\_makexp 完成。此处我将使用 sh\_makexp，它是一个非交互性的脚本指令，接受的参数为：

|  |  |
| --- | --- |
| 1 | $ sh\_makexp -expt <expt> -yr <year> -doy <doy> -nav <file> -orbt <orbt> -apr <file.> -sess <no.> -sinfo <session> |

参数释义：

* -expt <expt>：4 字符的项目名；
* -yr <year>：观测数据的年；
* -doy <doy>：观测数据的年积日；
* -orbt <orbt>：4 字符的轨道名；
* -gnss <code>：GNSS 系统代号（默认 G）；
* -nav <file>：广播星历的文件名；
* -apr <file.>：站点概略坐标文件名；
* -sess <no.>：解算任务的 session 号；
* -jclock <>：生成钟文件（J-文件）的来源文件（brdc 或 sp3），默认 brdc；
* -sinfo <session>：数据解算的间隔、开始时刻（时、分）和总历元数。

具体到本次任务，实际运行的命令为：

|  |  |
| --- | --- |
| 1 | $ sh\_makexp -expt demo -yr 2017 -doy 192 -nav brdc1920.17n -orbt brdc -apr lfile. -sess 99 -sinfo 30 00 00 2880 |

然后就是进行轨道拟合，使用精密星历时我们通过运行 sh\_sp3fit 来完成这一过程。但此时我们使用广播星历，其对应的命令为 sh\_bcfit，这一步几乎就是整个任务成败的关键。它接受的参数为：

|  |  |
| --- | --- |
| 1 | $ sh\_bcfit -f <file> -d <year> <doy> |

此处的 <file> 代表广播星历文件名，而 <year> 和 <doy> 分别为数据的年和年积日。即：

|  |  |
| --- | --- |
| 1 | $ sh\_bcfit -f brdc1920.17n -d 2017 192 |

之后就进入了熟悉的流程。生成卫星钟文件（J-文件）：

|  |  |
| --- | --- |
| 1 | $ makej brdc1920.17n jbrdc7.192 |

生成双差观测文件（X-文件）：

|  |  |
| --- | --- |
| 1 | $ makex demo.makex.batch |

启动数据驱动程序：

|  |  |
| --- | --- |
| 1 | $ fixdrv ddemo7.192 |

执行数据解算解算脚本：

|  |  |
| --- | --- |
| 1 | $ csh bdemo7.bat |

等待程序运行完成。若没有提示任何错误，就得到了数据处理的结果。

**补充**

有读者可能会质疑本文操作的必要性，毕竟 IGS 的超快速轨道产品在观测前的几个小时就能拿到。此时可以试想一下：IGS 等机构是如何生产精密轨道产品的？

**GAMIT 常见错误及解决方案**

[2017-08-01](http://www.gnss.help/2017/08/01/gamit-fatal-list/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/)

自使用 GAMIT 程序以来，一直希望做一个包含常见的错误及其解决方案的列表。有 GAMIT/GLOBK、Bernese、Panda 等科研软件使用经验的人都知道：使用这些程序做数据处理，其实就是不断解决错误的过程。因此一个包含常见错误及解决方案的列表必定能为使用者带来很多遍历。遗憾的是，平时对遇到的错误却常常无意识去做完整的整理。所以一直担心的是：此常见错误列表总是无法包含“常见错误”。

这几日像当初搭博客时一样，终于下定决心：不管最初版本能包含多少错误，先做出来再说。因此才有了这一篇博文。本篇博文可能永远不能包含所有错误的解决方案，但好在将随时保持更新。

**检索说明**

根据我个人的使用经验，暂且将错误类型分为三类：

* 与观测信息有关的错误；
* 与设置或表文件有关的错误；
* 与数据相关的错误。

本文列出的错误信息，仅仅是出现错误时提示的第一条“Fatal”。解决错误时，第一条错误信息通常最有用，因为后续的错误大多都是由之前的错误所引发。并且，这里给出的错误信息中，使用了具体的站点名和日期等，你遇到的错误在这些部分可能略有不同。

正如前面所说，限于个人的能力和精力，该错误列表可能永远不能包含所有可能遇到的错误。因此如果你有任何补充或建议，欢迎通过[邮件](mailto:jiangyingming@live.com)联系。补充时请提供完整的错误提示、错误原因和解决方案。

**与观测信息相关的错误**

**Error opening reference station.info file**

当提示该信息时，通常是因为解算目录的 tables 文件夹内缺少 station.info 文件。

解决方案：将 GAMIT/GLOBK 安装目录的 station.info 文件或从网上下载的 station.info 文件拷贝到解算目录的 tables 文件夹内。

**GDBTAB/geoc\_to\_geod: Failure to converge**

当提示该条提示信息时，通常是因为提供的某些站点概略坐标错误。

解决方案：在 GAMIT 输出信息中从该条错误信息开始往前查找最近的一个站名，该站点即为概略坐标错误的站点。修改 lfile. 文件中该站点的概略坐标为正确值然后重新进行解算。

**Geodetic height unreasonable: check p- and l-files**

与上一条错误类似，该错误也是由站点概略坐标错误引起。

解决方案：在 GAMIT 输出信息中从该条错误信息开始往前查找最近的一个站名，该站点即为概略坐标错误的站点。修改 lfile. 文件中该站点的概略坐标为正确值然后重新进行解算。

**No match SITE 2016 169 0 0 0 in station.info**

当提示类似这样的信息时，说明 station.info 文件里没有 SITE 站点在某时段的站点信息。

解决方案：向 station.info 文件中添加缺失的信息即可，具体命令可参考[站点信息文件的创建与更新](http://www.gnss.help/2017/03/05/create-station-info/)。

**SITE PCN-code missing for receiver type in rcvant.dat**

当出现类似这样的提示信息时，通常是因为某个站点（SITE）的接收机类型不包含在 rcvant.dat 文件内，即该站点的接收机类型不受支持。

解决方案：查看对应站点的观测文件和 rcvant.dat 文件，核对其中的接收机类型。若观测文件中的接收机类型与 rcvant.dat 中的相似但不完全一致，则可能是观测文件中的接收机类型不规范，此时可以将其中的接收机类型改为 rcvant.dat 中的规范名；若观测数据的接收机类型在 rcvant.dat 中不存在，可以考虑将观测数据的接收机类型用相似的接收机代替。

**Station.info missing antenna for SITE 2017 211**

当出现类似这样的错误提示时，说明 station.info 文件中缺少某个站点（SITE）的天线类型。

解决方案：编辑站点在对应年积日的观测文件，加入或修改其中的天线类型信息。然后使用 sh\_upd\_stnfo 命令更新站点信息文件 station.info。具体可参考[站点信息文件的创建与更新](http://www.gnss.help/2017/03/05/create-station-info/)。

**与设置或表文件相关的错误**

**components on grid file record not equal 44 or 84 (Name otl.grid)**

该错误是由海潮模型引起的，原因是没有正确设置海潮模型。

解决方案：方案之一是在 sestbl. 文件中禁用海潮模型，在其中进行如下设置：

Use otl.grid = N  
Tides applied = 23

或者下载所需的海潮模型文件，并将 otl.grid 链接到模型文件。然后设置：

Use otl.grid = Y  
Tides applied = 31

之后重新解算即可。该部分的详细介绍可移步至：[GAMIT 中的 GRID 文件简介](http://www.gnss.help/2016/10/02/grids-in-gamit/)。

**Date for TAI-UTC(2457936) after stop date in leap.sec**

出现类似错误的原因是参与解算的数据的观测日期晚于 leap.sec 文件的更新日期。

解决方案：下载最新的 leap.sec 文件，替换解算目录 tables/ 文件夹下的同名文件，然后重新进行解算。程序详细的更新策略可参考 [GAMIT/GLOBK 更新方法](http://www.gnss.help/2016/08/29/UpdateGAMIT/)。

**JD= 2458017 out of range of pole/ut1 table, JDT1= …**

出现类似错误的原因是参与解算的数据的观测日期晚于 pole.usno 或 ut1.usno 文件的更新日期。

解决方案：下载最新的 pole.usno 和 ut1.usno 文件，替换解算目录 tables/ 文件夹下的同名文件，然后重新进行解算。程序详细的更新策略可参考 [GAMIT/GLOBK 更新方法](http://www.gnss.help/2016/08/29/UpdateGAMIT/)。

**Ocean loading requested no list or grid file**

出现该问题的原因是没有正确链接海潮模型文件。

解决方案：下载所需的海潮模型文件，并将 otl.grid 链接到模型文件。然后设置：

Use otl.list = N  
Use otl.grid = Y  
Tides applied = 31

之后重新解算即可。该部分的详细介绍可移步至：[GAMIT 中的 GRID 文件简介](http://www.gnss.help/2016/10/02/grids-in-gamit/)。

**Site-dependent mapping function requested but no list or grid file**

出现该错误的原因是缺少与设置相对应的映射函数（VMF1）。

解决方案：基线解算过程中使用的映射函数模型在 sestbl. 文件中设置。GAMIT 目前支持三种映射函数，但只有 VMF1 需要引入模型文件。所以方案一是将解算使用的映射函数修改为 GMF 或 NMF：

DMap = GMF 　　 ; GMF(default)/VMF1/NMFH; GMF now invokes GPT2 if gpt.grid is available (default)  
WMap = GMF 　　 ; GMF(default)/VMF1/NMFW; GMF now invokes GPT2 if gpt.grid is available (default)  
Use map.list = N 　　 ; VMF1 list file with mapping functions, ZHD, ZWD, P, Pw, T, Ht  
Use map.grid = N 　　 ; VMF1 grid file with mapping functions and ZHD

如果你执意使用 VMF1 映射函数，方案二是下载 VMF1 的格网模型。将 tables/ 目录内的 map.grid 链接到模型文件，然后设置：

DMap = VMF1 　　 ; GMF(default)/VMF1/NMFH; GMF now invokes GPT2 if gpt.grid is available (default)  
WMap = VMF1 　　 ; GMF(default)/VMF1/NMFW; GMF now invokes GPT2 if gpt.grid is available (default)  
Use map.list = N 　　 ; VMF1 list file with mapping functions, ZHD, ZWD, P, Pw, T, Ht  
Use map.grid = Y 　　 ; VMF1 grid file with mapping functions and ZHD

与格网模型文件相关的详细内容可移步至：[GAMIT 中的 GRID 文件简介](http://www.gnss.help/2016/10/02/grids-in-gamit/)。

**与数据相关的错误**

**Beidou frequencies on x-/c-file file not C2 and C7**

处理北斗数据时可能会出现此错误。错误出现的原因是：某些观测文件不包含北斗观测数据。

解决方案：如果你想查看有问题的观测文件，在 GAMIT 输出信息中从该条错误信息开始往前查找最近的一个站名，该站即为观测数据有问题的站点。但为了彻底解决该问题，你应该逐个检查输入的 RINEX 观测数据，剔除所有不含北斗观测值的文件，然后重新进行数据处理。

**Error in input data file at epoch 2341**

出现此类错误是因为观测数据中某个历元下的数据出现问题。

解决方案：从该错误提示开始向上查找到的第一个站名就是数据有问题的测站，使用 GFZRNX 程序对数据文件进行修复，或直接将该站点的观测文件删除，然后清空年积日文件夹后重新解算。数据文件修复方法参见本站博文：[GFZRNX 配置及格式相关功能的使用](http://www.gnss.help/2016/10/30/gfzrnx-config-translate/)。

**Only one or no existing X-files**

此错误一般不会单独出现，若该错误作为第一条 Fatal，那么原因可能是观测数据不合格而被程序忽略。观测数据不合格的原因可能有：

* 观测文件未采用小写字母命名；
* 观测文件不是 GAMIT/GLOBK 支持的格式；
* 观测文件的文件太小而被忽略。

解决方案：对于第三种情况，GAMIT 解算时的控制文件 process.defaults 中，将所使用的观测文件的最小值设置为 300 KB。若你的观测文件小于该值，你可以修改 tables 文件夹内的该文件然后重新解算。

**sh\_chksolve: Solve failed to complete normally**

这个错误只遇到过一次，但从提示信息中看不出错误原因。

解决方案：清空年积日文件夹，重新运行 GAMIT 批处理命令。有时候这样可以解决该问题，推测该错误可能是由程序的 Bug 造成的。

**Word to long**

该错误通常在执行 sh\_upd\_stnfo 命令时出现。由于该命令指定 RINEX 观测文件时采用通配符，若通配符匹配到的文件太多，或文件路径太长，易触发该错误。

解决方案：调整匹配 RINEX 文件的通配符，每次使用更少的 RINEX 文件对站点信息文件进行更新。

**DOS2UNIX 安装与使用简介**

[2017-07-24](http://www.gnss.help/2017/07/24/dos2unix-install-usage/)

[LINUX](http://www.gnss.help/categories/Linux/)

[DOS2UNIX](http://www.gnss.help/tags/Dos2unix/), [LINUX](http://www.gnss.help/tags/Linux/)

目前很多脚本语言都能够跨平台：在 Windows 系统的计算机上开发调试，然后上传到远程的 Linux 计算机去执行的情况非常普遍。但是如果你只是简单地将 Windows 系统下编辑过的脚本拿到 Linux 系统执行，会遭遇奇怪的错误。这通常是由换行符编码的问题造成的：Windows、UNIX/Linux 和早期的 macOS 操作系统对换行符使用了各自的编码方案。

dos2unix 是一个开源免费的可执行程序包，支持多个平台，不仅可以将文本文件的换行符编码在 Windows、macOS 和 UNIX/Linux 操作系统之间进行转换，还能转换文件的编码方式。本文介绍该程序包的配置和使用过程。

**安装**

**Windows 操作系统**

对于 Windows 操作系统的用户，从 [SourceForge](http://sourceforge.net/projects/Dos2unix/) 网站下载程序压缩文件，解压后即可看到 bin/ 文件夹内的可执行程序。打开 “命令提示符” 窗口，使用 cd 命令进入程序所在目录，键入程序名（如 dos2unix）即可运行相应的程序。

如果你希望在任何目录都能使用该程序，可以考虑将程序所在目录添加到系统的环境变量，或直接将软件包内的可执行文件移动至 “C:\Windows\System32” 文件夹内。

**Linux 操作系统**

对于 Linux 操作系统，dos2unix 程序已经被添加到系统的软件源内，可以直接从软件源进行安装。

Fedora、CentOS 等 Linux 发行版的安装命令为：

|  |  |
| --- | --- |
| 1 | $ sudo yum install dos2unix |

Debian、Ubuntu 等发行版的安装命令为：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install dos2unix |

安装之后即可使用其包含的几个命令，如 dos2unix、unix2dos 等。

**Node.js**

使用 Node.js 的用户，还可以直接使用 Node.js 中的对应模块：

|  |  |
| --- | --- |
| 1 | $ npm install dos2unix |

遗憾的是，该模块不可以作为命令行程序直接使用，只能作为 Node.js 的编程模块。

**程序使用**

**程序功能**

Dos2unix 程序包由 4 个子程序组成，它们分别是：

* dos2unix：将 MS-DOS 编码的文件转换为 UNIX/Linux 的格式；
* mac2unix：将 macOS 系统编码文件转换为 UNIX/Linux 的格式；
* unix2dos：将 UNIX/Linux 系统编码文件转换为 MS-DOS 的格式；
* unix2mac：将 UNIX/Linux 系统编码文件转换为 macOS 的格式。

**参数使用**

由于该程序包 4 个子程序接收的输入参数是一致的，此处仅以 dos2unix 为例。它的参数输入方式为：

|  |  |
| --- | --- |
| 1 | $ dos2unix [options] <files> |

这里的 <files> 为待处理的文件列表，而 [options] 为可选的控制参数。其中可选参数非常多，这里只择其概要：

* -o <files>：直接操作输入文件进行编码转换，此处的 -o 可以省略；
* -n <input> <output>：转换输入文件，将操作结果输出至新的输出文件；
* -i <files>：仅查看文件的格式信息，不对文件进行转换操作；
* -f、--force：强制转换二进制文件，默认为跳过二进制文件；
* -k、--keep-date：保持新文件的时间戳（修改时间）不变；
* -b、--keep-bom：保持 BOM 不变；
* -m、--add-bom：添加 BOM（在 unix2dos 中为默认项）；
* -r、--remove-bom：移除 BOM（在 dos2unix 中为默认项）；
* -q、--quiet：安静模式，转换时不输出提示，仅以返回值表示操作成败；
* -v、--verbose：“啰嗦”模式，输出详细的提示信息；用于 Debug；
* -V、--version：显示程序版本号；
* -h、--help：显示程序帮助信息。

**使用示例**

不对文件进行转换操作，仅查看文件 demo.txt 的编码方式，：

|  |  |
| --- | --- |
| 1 | $ dos2unix -i demo.txt |

执行上面的命令将得到类似这样的一行输出信息：

  100   0   0   no\_bom   text   demo.txt

以上信息依次为：MS-DOS 方式的换行符个数、UNIX/Linux 方式的换行符个数、Mac OS 方式的换行符个数、BOM、文件类型和文件名。

直接操作原有文件，将 demo.txt 由 MS-DOS 编码模式转换到 UNIX/Linux 编码模式：

|  |  |
| --- | --- |
| 1 | $ dos2unix demo.txt |

以上命令等效于：

|  |  |
| --- | --- |
| 1 | $ dos2unix -o demo.txt |

以详细模式运行程序，将 dos.txt 由 MS-DOS 编码模式转换到 UNIX/Linux 编码模式，生成新的文件 unix.txt：

|  |  |
| --- | --- |
| 1 | $ dos2unix -v -n dos.txt unix.txt |

将采用 UNIX/Linux 编码方案的 unix.txt 转换为 MS-DOS 编码模式，并输出到 dos.txt，在转换时保持文件的时间戳不变：

|  |  |
| --- | --- |
| 1 | $ unix2dos -k -n unix.txt dos.txt |

如果你希望将 Dos2unix 程序包集成到程序中，并且不希望其显示输出信息，可以使用安静模式。即在运行是添加 -q 参数。在这种模式下，只要不是程序输入参数错误，返回值将总是 0。示例，为上一个命令应用安静模式：

|  |  |
| --- | --- |
| 1 | $ unix2dos -k -q -n unix.txt dos.txt |

# Linux 系统搭建 FTP 服务器

[2017-07-17](http://www.gnss.help/2017/07/17/build-vsftpd-server/)

[LINUX](http://www.gnss.help/categories/Linux/)

[VSFTPD](http://www.gnss.help/tags/vsFTPd/)

FTP 服务目前依然是最简单有效的文件传输和分享方案之一，在 Windows 操作系统上搭建 FTP 服务器非常简单。我在 Linux 系统上搭建 FTP 时，本以为也会一路畅通，却最终走了很多弯路。特别是中文教程良莠不齐，作者的表述也不够细致准确。因此我决定将我的配置过程写下来，并尽量保证清晰周全，希望能为有需要的读者提供参考。

我的 FTP 服务器仅供极少的用户使用，因此采用本地用户的方式进行登录。为了同时兼容 FTP 客户端和浏览器等多种接入方式，还同时设置了对主动模式和被动模式两种访问方式的支持。

## 环境说明

我所搭建的 FTP 服务器基于 CentOS 7 操作系统，使用的 FTP 服务器软件为 vsFTPd。但在配置时并不涉及对操作系统的设置，因此实际上，其他 Linux 发行版的用户也可以作为参考。

### 程序安装

vsFTPd 程序体积小巧而功能强大，往往被作为 FTP 服务器的首选。目前许多的 Linux 发行版都已将该程序添加到系统的软件源中，从软件源安装自然是最方便的。对于 CentOS 等发行版，安装 vsFTPd 的命令为：

|  |  |
| --- | --- |
| 1 | $ sudo yum install vsftpd |

对于 Ubuntu 等发行版，对应的安装命令为：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install vsftpd |

### 主动模式设置

程序安装完成之后，下一步就是进行程序配置。如果你使用本文所推荐的方式进行程序安装，vsFTPd 程序的配置文件通常是在 “/etc/vsftpd/” 目录内。

谨慎起见，在修改配置文件之前，我们最好将默认配置备份：

|  |  |
| --- | --- |
| 1 | $ cp vsftpd.conf vsftpd.conf.bak |

之后就是对 vsftpd.conf 进行编辑设置了。在此文件中，以 “#” 开头的部分为注释。我对其中的改动如下：

anonymous\_enable=NO  
local\_enable=YES  
chroot\_local\_enable=YES  
chroot\_list\_file=/etc/vsftpd/chroot\_list  
listen=YES  
#listen\_ipv6=YES

以上内容中的第一行关闭匿名用户的访问，第二行开启本地用户的访问；第三行，允许本地用户在使用 FTP 服务时，能访问自己主目录的内容；为了安全起见，第四行指定了不允许使用 FTP 方式登录的用户名列表；最后两行将 FTP 监听的 IP 地址改为 IPv4，并关闭了对 IPv6 的支持。因为目前的 vsFTPd 还不能同时使用 IPv4 和 IPv6。

在很多教程中，设置过程到此已经结束了。但实际上如果你现在启动 FTP 服务的话，将遭遇类似的报错信息：

refusing to run with writable root inside chroot()

配置文件中还需要添加一行：

allow\_writeable\_chroot=YES

现在就可以开启防火墙相关的端口了。对于 CentOS 7 操作系统，系统默认的防火墙程序为 firewalld，开启 FTP 端口并重启防火墙的命令为：

|  |  |
| --- | --- |
| 1  2 | $ firewall-cmd --add-service ftp --permanent  $ firewall-cmd --reload |

如果你使用的是较早的 Linux 系统，默认的防火墙程序为 iptables。开启 FTP 端口的命令为：

|  |  |
| --- | --- |
| 1  2 | $ iptables -I INPUT -p tcp --destination-port 20:21 -j ACCEPT  $ service iptables save |

最后启动 vsFTPd 程序，主动模式下的 FTP 服务器配置就已经完成了：

|  |  |
| --- | --- |
| 1 | $ service vsftpd start |

如此设置后，你可以通过 Windows 操作系统 “命令提示符” 窗口的 ftp 命令，成功登录并使用 FTP 服务。但使用资源管理器或浏览器则无法访问，因为它们使用被动模式的 FTP。要想使用资源管理器或浏览器访问 FTP 服务器，还需要为 vsFTPd 设置被动模式。

### 被动模式设置

继续编辑 vsFTPd 的配置文件，在其中添加以下项目：

port\_enable=YES  
pasv\_enable=YES  
pasv\_min\_port=65500  
pasv\_max\_port=65535  
pasv\_address=12.34.56.78

在以上的改动项中，第一项设置开启 FTP 主动模式，第二行开启 FTP 被动模式；之后两行，设置了被动模式下数据传输端口的窗口范围；最后一行设置被动模式下，服务器的公网 IP 地址，注意将此处的地址改为服务器实际的公网 IP。

同样的，还需要开启防火墙的相应端口。对于使用 firewalld 作为防火墙的系统，运行命令为：

|  |  |
| --- | --- |
| 1  2 | $ firewall-cmd --add-port=65500-65535/tcp --permanent  $ firewall-cmd --reload |

对于采用 iptables 作为防火墙的系统，运行命令为：

|  |  |
| --- | --- |
| 1  2 | $ iptables -I INPUT -p tcp --destination-port 65500:65535 -j ACCEPT  $ service iptables save |

最后，重启 vsFTPd 程序以应用配置：

|  |  |
| --- | --- |
| 1 | $ service vsftpd restart |

现在就可以使用 FTP 的被动模式了，你可以在系统的资源管理器中输入 FTP 服务的地址试一下。

**网站迁移笔记**

[2017-07-11](http://www.gnss.help/2017/07/11/notes-of-site-migration/)

[网站](http://www.gnss.help/categories/%E7%BD%91%E7%AB%99/)

[网站](http://www.gnss.help/tags/%E7%BD%91%E7%AB%99/)

7 月 5 日至 8 日，本站自上线以来遭遇了第一次长时间无法访问。原因有些用户可能已经看到了当时的提示：网站没有按照要求在工信部备案。此事说来甚委屈，毕竟博客仅仅上线一年，只是用来总结一下自己从事基础 GNSS 数据处理工作的一点人生经验。况且，网站并没有太多用户，我一个小博客，怎么就无法访问了呢？

值得庆幸的是，网站终于在 8 日下午得到恢复。本文简单总结这次搬家的过程，既可以说是对中断原因的解释，也算是一次教训检讨。若今后再出现网站不能访问的问题时，请大家及时反馈，并耐心等待解决方案。

**中断原因**

本网站的域名通过阿里云旗下的万网购买，自上线之日起，域名解析和网站一直托管在阿里云服务器上。之前也听说过托管在境内服务器的网站需要备案的事情，但在尝试备案时，系统提示该域名后缀无法备案。但网站一直能够访问，就想当然的以为该类后缀无需备案了。直至网站突然无法访问才领悟到，当时的想法太想当然了。

出现网站不能访问的情况后，第一时间联系了阿里云的工作人员。经过客服的解释，才终于理清了思路：

1. 凡是托管在在境内（除港澳台）的网站，都需要在上线之前在工信部备案；
2. 工信部目前已经批准了包括 .help、.info、.me 等后缀的域名注册；
3. 工信部当前仅接受 .com、.cn、.net 等传统后缀域名备案的申请。

是的，有些类似于第二十二条军规：你可以在国内注册，我也很乐意批准，但你不可以在国内服务器上线。

**迁移过程**

**ECS 购买**

好的，要恢复网站有两条路可以走：要么将网站更换成能备案的域名；要么将其迁移到港澳台或国外的机房，绕过备案的限制。经过权衡之后，忍痛租了一台位于香港的 ECS 云服务器。价格虽然贵了一些，但考虑到以后还能用来扩展一些其他的功能，还是值得的。

这次购买的是阿里 ECS 最低配置的版本：1 个内核，1 GB 内存，40 GB 磁盘空间，运行 CentOS 7 操作系统。

**安全设置**

初始化之后的服务器环境，基本禁用了除 SSH 或 RDP 远程控制外所有的功能端口。因此，设置服务器的第一步就是先开放服务器的 HTTP 端口。

进入阿里云的 ECS 管理控制台，找到“安全组”的“配置规则”页面，在“入方向”上添加 HTTP 的入站规则。

**程序安装**

下一步是安装 nginx 程序，它是一个轻量级的 Web 服务器程序，可用来托管网站和电子邮件服务。CentOS 7 操作系统安装该程序的命令为：

|  |  |
| --- | --- |
| 1  2 | $ sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm  $ sudo yum install nginx |

之后还需要开启操作系统防火墙的 80 端口，nginx 将通过该端口对外提供服务。CentOS 7 系统默认的防火墙程序已经由 iptables 换成了 firewalld，并且默认没有启动。可使用下面的命令来启动防火墙，并将其添加到开机启动项：

|  |  |
| --- | --- |
| 1  2 | $ systemctl start firewalld  $ systemctl enable firewalld.service |

然后使用下面的指令来永久开放 80 端口，并重启防火墙：

|  |  |
| --- | --- |
| 1  2 | $ firewall-cmd --add-port=80/tcp --permanent  $ firewall-cmd --reload |

**服务器设置**

最后就是对网站运行环境的设置了。首先为系统创建一个新用户，用于保管网站的文件。这样既方便以后的网站部署，也有利于保护服务器的安全。使用新用户将 hexo 生成的 public/ 文件夹上传至该用户的主文件夹下。作为一个静态站，该文件夹已经包含了网站的全部资源。

修改 nginx 程序的配置文件：/etc/nginx/nginx.conf，在其中添加网站配置。主要修改的部分如下：

server {  
listen　　　　　80 default\_server;  
listen　　　　　[::]:80 default\_server;  
server\_name　　www.gnss.help gnss.help;  
root　　　　　　/home/writer/public;  
index 　　　　　index.html;  
}

在该文件中，listen 为服务器监听的端口，这里保持默认即可；server\_name 项为网站域名；root 项代表文件资源所在的目录；index 项为网站主页文件的文件名。

配置完成后，开启 nginx 就可以通过服务器的公网 IP 访问网站了：

|  |  |
| --- | --- |
| 1 | $ nginx & |

如果访问时提示 “403 Forbiden” 的错误，说明启动 nginx 程序的用户没有访问网站资源文件夹的权限。更改网站资源文件夹及其上层文件夹的权限为 755 即可解决。

完成以上操作之后，网站就已经能够访问了。以后每次发布新的文章时，只要通过 SFTP 用本地的 public/ 文件夹覆盖服务器上的同名文件夹即可。

**Graphviz 安装与使用简介**

[2017-06-30](http://www.gnss.help/2017/06/30/graphviz-install-usage/)

[GRAPHVIZ](http://www.gnss.help/categories/Graphviz/)

[GRAPHVIZ](http://www.gnss.help/tags/Graphviz/)

之前绘制各种流程图和示意图时，使用过微软的 Visio，也用过 PowerPoint、画图等程序。总觉得这些程序效率不高，其中的很多时间浪费在对格式的调整和修饰上。如果专注于定义图形的逻辑结构，而不参与具体实现过程，更有利于集中精力。这也是我一直追求纯文本编辑方案、偏爱命令行程序的主要原因。

[Graphviz](http://graphviz.org/) 就是这么一个命令行程序：它可以根据一种名为 [dot](http://graphviz.org/content/dot-language) 的图形结构描述语言，生成简洁美观的示意图。如结构图、流程图、UML 图等。本文包含对程序的安装和使用该程序绘制流程图的简介。

**程序安装**

作为一款开源软件，Graphviz 提供了可用于 Windows、macOS、Linux 等常见操作系统的程序包。这些程序包都可以在其[官网下载页面](http://graphviz.org/Download.php)找到。

**Windows 操作系统**

要在 Windows 操作系统中使用 Graphviz，下载对应的安装包后，双击运行，即可根据提示将其安装到计算机中。

但安装程序并未帮我们配置环境变量，因此还不能在任意目录下使用 Graphviz。你可以通过控制面板中的 “环境变量” 配置窗口，将 Graphviz 程序安装目录的 bin/ 文件夹添加到系统的 Path 变量中。

**Linux 操作系统**

对于 Ubuntu 等 Linux 发行版，Graphviz 已经被添加到官方的软件源中。因此可以通过下面的命令进行安装：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install graphviz |

对于 CentOS 等操作系统，Graphviz 已经被添加到其企业软件附加包（EPEL）中，可以在安装该附加包后，通过下面的命令安装：

|  |  |
| --- | --- |
| 1 | $ sudo yum install graphviz |

当然，你也可以从其官网下载对应的程序文件包进行安装。

**程序简介**

**各子程序**

如果你已经迫不及待地尝试过运行 graphviz 命令，一定会感到失望。因为系统提示没有这个程序。实际上，Graphviz 程序是由多个子程序组成的，它们适用于不同的构图需求。主要有：

* dot：绘制有向图；
* neato：绘制无向图；
* twopi：绘制辐射状布局的图形；
* circo：绘制圆环状布局的图形；
* fdp：绘制无向图，使用了不同的算法；
* sfdp：绘制大图幅的无向图；
* patchwork：绘制树状图；
* osage：绘制基于阵列的图形。

**dot 语言简介**

在介绍 Graphviz 程序的使用之前，需要先简单介绍 [dot](http://graphviz.org/content/dot-language) 语言。

dot 是 Graphviz 定义的一种图形结构描述语言，用于对图形的结构和样式进行定义。dot 首先将图形分为两类，即无向图（graph）和有向图（digraph）；然后进一步将图形的组成元素分为三类：节点（node）、边（edge）和子图（subgraph）。图形的样式可以通过各部分的属性进行定义。使用 dot 语言描述一副图形的结构时，首先定义该图是无向图还是有向图，然后定义其包含的节点、边和子图，并分别定义各组成部分的属性。

**流程图**

该语言一个简单的流程图演示如下（感谢知乎[轮子哥](https://www.zhihu.com/question/61624754/answer/189819014)提供如此生动的例子）：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | digraph coder {  // set graph attributes  resolution = 320  // set nodes' attributes  node [shape=rect] // set default shape  start, end [shape=Mrecord]  dead [shape=diamond, label="dead ?"]  // set edges' attributes  start -> work -> eat -> poop -> dead  dead -> burial [label=" Yes"]  dead -> work [label=" No", constraint=false]  burial -> end  } |

在该示例中，digraph 定义该图形是一个有向图，之后的 coder 为图形的标识名，大括号内的部分包含图形结构的定义。resolution 项定义了该图的分辨率。之后的区块中，定义了各节点和它们的属性。其中 shape 属性为节点的轮廓形状，而 label 属性为节点内的文字标签。最后的区块定义了各节点之间的连接关系，即它们之间的边。该图中的边皆为有向边，它们使用 -> 表示。部分边还有文字标签等属性，用于对路径进行说明。

与有向图相对应的无向图使用 graph 关键字定义，其中的无向边用 -- 表示。此外，还可以使用 subgraph 定义子图等。想要详细了解 dot 语言，可以参考其[官方文档](http://graphviz.org/content/dot-language)。

**关系图**

类似的，一个简单的社交网络图的 dot 语言代码如下所示，该图展现了初唐诗人的社交关系：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | digraph poets {  resolution = 320  node [fontname="FangSong"]  // poets network by https://github.com/MrQianJinSi/poetry\_analyzer  王勃 -> 骆宾王 -> 李峤 -> 杜审言 -> 苏味道  骆宾王 -> 宋之问 -> 杜审言 -> 韦承庆  骆宾王 -> 沈佺期 -> 李贤  李峤 -> 骆宾王 -> 卢照邻  李峤 -> 宋之问  宋之问 -> 李世民 -> 魏征  宋之问 -> 沈佺期 -> 乔知之  宋之问 -> 陈子昂 -> 乔知之  沈佺期 -> 宋之问  沈佺期 -> 陈子昂  乔知之 -> 陈子昂  } |

**输入参数**

Graphviz 各子程序的输入参数大同小异，其形式为：

|  |  |
| --- | --- |
| 1 | $ command <input> <options> |

这里的 command 代表具体的程序名；<input> 代表输入的内容，其输入即可以来自文件，也可以来自标准输入；而 <options> 为参数选项。主要的参数选项有：

* -T <format>：指定输出文件的格式，如 FIG、PNG、PS 等；
* -o <file>：指定输出文件；
* -v：开启用于 Debug 的 Verbose 模式；
* -V：显示程序版本号；
* -?：显示程序帮助信息。

**使用示例**

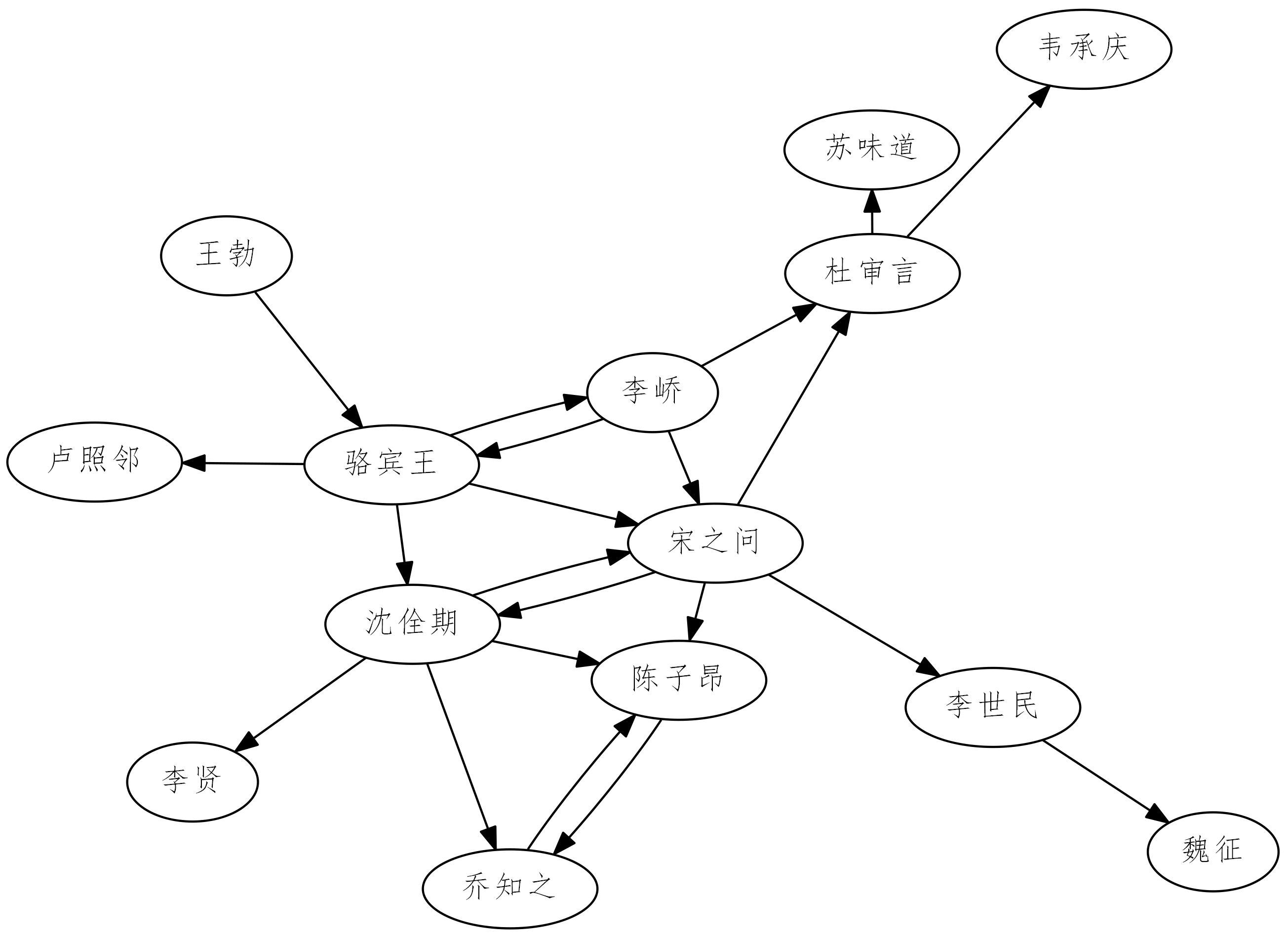
下面渲染上文的演示代码作为程序的使用示例。将上文流程图的代码保存为 demo.gv，使用 cd 命令进入文件所在目录，然后执行：

|  |  |
| --- | --- |
| 1 | $ dot demo.gv -T png -o demo.png |

对于社交关系图，可将其保存为 poets.gv，然后执行：

|  |  |
| --- | --- |
| 1 | $ fdp poets.gv -T png -o poets.png |

运行该命令后，将得到输出的图片，它们分别是是一个简单的流程图和社交关系图。其中的社交关系图如下：

[](http://www.gnss.help/images/poets_fdp.png)初唐诗人社交网络

如果你不确定该使用 Graphviz 中的哪个命令，那么你可以都试试，然后挑选出自己最满意的一个。此外，[Graphviz 网站](http://graphviz.org/Gallery.php)上还提供了更多的示例，方便使用者查看和学习。

**Pandoc 安装与使用心得**

[2017-06-12](http://www.gnss.help/2017/06/12/pandoc-install-usage/)

[PANDOC](http://www.gnss.help/categories/Pandoc/)

[PANDOC](http://www.gnss.help/tags/Pandoc/), [网站](http://www.gnss.help/tags/%E7%BD%91%E7%AB%99/)

[Markdown](https://zh.wikipedia.org/zh-cn/Markdown) 是我平时编写各类材料和文章时最喜欢的工具之一，但将其作为文档分发时却略有些不便。毕竟不是所有人都了解 Markdown 的语法，而且它亦不如富文本文档易读。直至我发现了 [Pandoc](http://pandoc.org/)，它可以将文档在 Markdown、LaTeX、reStructuredText、HTML、Word docx 等多种标记格式之间相互转换，并支持输出 PDF、EPUB、HTML 幻灯片等多种格式。该程序被称为格式转换界的 “瑞士军刀”。

我平时大量使用该程序生成 HTML 和 PDF 文档，希望本文可以总结一些自己使用的心得。

**程序简介**

Pandoc 的作者是 [John MacFarlane](http://johnmacfarlane.net/)，他是加州大学伯克利分校的哲学系教授。Pandoc 使用 [Haskell](http://www.haskell.org/) 语言编写，被作者用来生成讲义、课件和网站等。该程序开源免费，目前以 GPL 协议托管在 [Github 网站](https://github.com/jgm/pandoc)上。

**程序安装**

Pandoc 的安装有许多方式，但本文只介绍最简单的方法。Pandoc 的作者已经为 Windows、macOS、Linux 等操作系统分别制作了对应的[程序安装包](http://pandoc.org/installing.html)。要使用该程序，只需下载对应的程序安装包进行安装即可。

对于 Ubuntu 等 Linux 发行版，Pandoc 已经被集成到系统的软件源内，因此还可以直接从软件源安装：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install pandoc |

或者，如果你已经安装了 Anaconda，那么你可以直接使用 Pandoc 了。该程序已经被集成到 Anaconda 中。

**参数说明**

Pandoc 程序的命令使用方式为：

|  |  |
| --- | --- |
| 1 | $ pandoc <files> <options> |

其中 <files> 为输入的内容，其输入即可以来自文件，也可以来自标准输入甚至网页链接。而 <options> 为参数选项。主要的参数选项有：

* -f <format>、-r <format>：指定输入文件格式，默认为 Markdown；
* -t <format>、-w <format>：指定输出文件格式，默认为 HTML；
* -o <file>：指定输出文件，该项缺省时，将输出到标准输出；
* --highlight-style <style>：设置代码高亮主题，默认为 pygments；
* -s：生成有头尾的独立文件（HTML，LaTeX，TEI 或 RTF）；
* -S：聪明模式，根据文件判断其格式；
* --self-contained：生成自包含的文件，仅在输出 HTML 文档时有效；
* --verbose：开启 Verbose 模式，用于 Debug；
* --list-input-formats：列出支持的输入格式；
* --list-output-formats：列出支持的输出格式；
* --list-extensions：列出支持的 Markdown 扩展方案；
* --list-highlight-languages：列出支持代码高亮的编程语言；
* --list-highlight-styles：列出支持的代码高亮主题；
* -v、--version：显示程序的版本号；
* -h、--help：显示程序的帮助信息。

虽然 Pandoc 提供了用于指定输入输出格式的参数，但是很多时候该参数不必使用。Pandoc 已经足够聪明到可以根据文件名判断输入输出格式，所以除非文件名可能造成歧义，否则这两个参数都可以省略。

**使用示例**

**信息查看**

查看程序支持的输入文件格式：

|  |  |
| --- | --- |
| 1 | $ pandoc --list-input-formats |

查看程序支持代码高亮的编程语言：

|  |  |
| --- | --- |
| 1 | $ pandoc --list-highlight-languages |

查看程序帮助：

|  |  |
| --- | --- |
| 1 | $ pandoc --help |

**生成 HTML 文档**

使用 Pandoc 可以很容易地将 Markdown 文档渲染为 HTML 网页：

|  |  |
| --- | --- |
| 1 | $ pandoc demo.md -o demo.html |

上面的命令将输出一个 HTML 文档，但该文档不包含任何样式，它的显示效果依赖于你使用的浏览器。我们当然希望可以得到排版更精美的文档，只要在转换时引入自己的层叠样式表 CSS 文件。输入的 CSS 文件可使用 -c 命令来指定：

|  |  |
| --- | --- |
| 1 | $ pandoc demo.md -c style.css -o demo.html |

如此输出的 HTML 文档已经包含样式文档了，平时自己查看时，效果很不错。但该方式依然存在部分问题。我们发布或共享文档时，需要传送至少两个文件：1 个 HTML 文件和 1 个 CSS 文件，略有些不便。而如果文档中还包含多个本地图片等文件，共享文档几乎成了不可能的事情。好在 Pandoc 可以将外部文件嵌入到 HTML 文档中，生成一个自包含的独立文件：

|  |  |
| --- | --- |
| 1 | $ pandoc demo.md --self-contained -c style.css -o demo.html |

在该命令中，--self-contained 参数指定：将任何的外部文件嵌入至输出的文件中，形成一个独立的 HTML 文档。这样传送资料时只传送一个文件就可以了，就像分享 PDF 文档一样方便。

**生成 docx 文档**

虽然我很喜欢使用 HTML 作为文档交换格式，但某些情况下你可能还是需要传送 Word docx 文件。这也不是问题，Pandoc 能够将所支持的输入文件一键转换为 Word docx 格式。

下面的命令将一份 Markdown 文件转换为 docx 格式：

|  |  |
| --- | --- |
| 1 | $ pandoc demo.md -o demo.docx |

下面的命令将 HTML 网页转换为 docx 格式：

|  |  |
| --- | --- |
| 1 | $ pandoc http://gnss.help/2017/06/12/pandoc-install-usage/ -o this\_page.docx |

需补充的是：Pandoc 无法为生成的 Word docx 文档指定排版方式。你可能需要二次编辑输出的文件，将标题、正文等调整为满意的样式。

**生成 PDF 文档**

使用 Pandoc 直接生成 PDF 文件时，需要安装 LaTeX。并且，Pandoc 自带的 PDF 引擎不支持中文，必须为中文配置额外的引擎和模板。Pandoc 程序生成 PDF 文件的命令为：

|  |  |
| --- | --- |
| 1 | $ pandoc demo.md -o demo.pdf |

我生成 PDF 文档时，未使用以上的方法。而是采用 HTML 文件作为中间文件过渡，使用 Windows 系统的 “打印到 PDF” 功能，将 HTML 文档进一步转换为所需的 PDF 文档。

**生成 Markdown 文档**

别忘了 Markdown 也是 Pandoc 支持的输出格式之一，我们可以将任何支持的输入格式转换为 Markdown。这对于我们将之前的文档也切换到 Markdown 格式来说，实在是太方便了。

下面的命令由 Word docx 文档生成 Markdown 文件：

|  |  |
| --- | --- |
| 1 | $ pandoc demo.docx -o demo.md |

下面的命令由 HTML 网页生成 Markdown 文档：

|  |  |
| --- | --- |
| 1 | $ pandoc http://gnss.help/2017/06/12/pandoc-install-usage/ -o this\_page.md |

**样式表文件**

使用 Markdown 文件生成 HTML 文档时，CSS 层叠样式表文件决定最终的显示样式，因此有一个漂亮的 CSS 样式表文件非常重要。在此推荐两个 CSS 文件，首先是由 Alberto Leal 制作的 [Github 风格的样式表文件](https://gist.github.com/Dashed/6714393)，它的显示效果类似于 Github 网站的 README 文档。另一个是我制作的，类似本站曾采用的 [Minos](http://blog.zhangruipeng.me/hexo-theme-minos/) 主题（[Minos-style](https://gist.github.com/purpleskyfall/98ecbccf4f2184aa0f365fbbae36ebdd)）的样式表文件。该文件还未完全稳定，尚需部分完善，不过对付一般的文字排版已经没有问题。

**Anubis 配置文件编辑**

[2017-05-22](http://www.gnss.help/2017/05/22/anubis-config/)

[ANUBIS](http://www.gnss.help/categories/Anubis/)

[ANUBIS](http://www.gnss.help/tags/Anubis/), [RINEX质量分析](http://www.gnss.help/tags/RINEX%E8%B4%A8%E9%87%8F%E5%88%86%E6%9E%90/)

[上一篇文章](http://www.gnss.help/2017/05/12/anubis-install-usage/)已经简要介绍过 [G-Nut/Anubis](http://www.pecny.cz/gop/index.php/gnss/sw/anubis) 程序的配置和使用方式。但如果仅阅读过前文，你肯定还不能愉快顺畅地使用它。因为该程序的输入参数都集中配置文件中，我们需要根据自己的使用意图来编辑配置文件。

本文介绍 Anubis 程序配置文件的设置方法。该文件使用可扩展标记语言 XML（Extensible Markup Language）编写，因此在阅读本文前，你可能需要首先了解一下 XML 格式。

**配置文件结构**

[前文](http://www.gnss.help/2017/05/12/anubis-install-usage/)已经提到过一个默认的 Anubis 程序配置文件的获得方法，这里不在赘述。记住完整的配置文件的内容和格式不仅非常困难，也没有必要。最好的办法是以默认的配置文件作为模板，对其进行必要的修改，产生我们需要的效果。

配置文件中包含一个 config 元素，其中包含各配置项目的设置，下文将逐个介绍。

**全局配置项**

全局配置项指的是 gen 元素中的项目。其包含的子元素有：

* beg：开始处理的时刻；
* end：结束处理的时刻；
* int：数据的采样间隔；
* sys：要处理的卫星系统；
* rec：要处理的站点列表。

一个 gen 元素的示例如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <gen>  <beg> "2017-04-19 00:00:00" </beg>  <end> "2017-04-19 23:59:59" </end>  <int> 30 </int>  <sys> GPS GAL GLO BDS -QZS -SBS </sys>  <rec> BJFS CHAN DAEJ </rec>  </gen> |

在上例中，将要处理的时间间隔设置为 2017 年 4 月 19 日 0 点 0 分 0 秒至 23 点 59 分 59 秒，数据的采样间隔为 30 秒，要处理的观测站的列表为 BJFS、CHAN 和 DAEJ；并指定了要处理的和不要处理的卫星系统，不处理的卫星系统前添加“-”号作为标记。

**质量检查配置项**

质量检查配置项指的是 qc 元素中的属性，其属性有：

* sec\_sum：总结报告信息的详细度；
* sec\_hdr：观测信息检查报告的详细度；
* sec\_obs：观测量统计报告的详细度；
* sec\_est：定位解算（SPP）报告的详细度；
* sec\_gap：观测值跳变报告的详细度；
* sec\_bnd：观测频段统计报告的详细度；
* sec\_pre：周跳和失锁等预处理报告的详细度；
* sec\_ele：卫星方位角和高度角报告的详细度；
* sec\_mpx：多路径效应影响报告的详细度；
* sec\_snr：信噪比统计报告的详细度；
* int\_stp：预处理时的数据取样间隔；
* int\_gap：探测观测跳变时的阈值；
* int\_pcs：探测观测缺失时的阈值；
* col\_sat：与卫星相关的报告的列数；
* mpx\_nep：探测多路径系统误差时使用的历元数；
* mpx\_lim：探测多路径误差时使用的阈值；
* ele\_cut：设置的截至高度角；
* pos\_kin：接收机是否为运动状态。

在以上的属性中，以“sec”作为前缀的项，其取值范围为 0 到 9。以“int”作为前缀的项，以秒作为单位，截至高度角以度作为单位。而 pos\_kin 的取值范围只有 true 或 false。

一个 qc 元素的示例如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <qc  sec\_sum="1"  sec\_hdr="1"  sec\_obs="1"  sec\_est="1"  sec\_gap="1"  sec\_bnd="1"  sec\_pre="1"  sec\_ele="1"  sec\_mpx="1"  int\_stp="900"  int\_gap="600"  int\_pcs="1800"  col\_sat="35"  mpx\_nep="15"  mpx\_lim="5"  ele\_cut="10"  pos\_kin="false"  /> |

**输入数据配置项**

输入数据配置项指的是 input 元素中的项目，其中的元素有：

* rinexo：输入的 RINEX 观测数据文件，多个文件使用空白分隔；
* rinexn：输入的 RINEX 广播星历文件，多个文件使用空白分隔；
* sp3：输入的精密星历文件，多个文件使用空白分隔。

一个示例的 input 元素如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <inputs>  <rinexo>  rinex/bjfs1090.17o  rinex/chan1090.17o  rinex/daej1090.17o  </rinexo>  <rinexn> brdc/brdm1090.17p </rinexn>  <sp3> igs/gbm19453.sp3 </sp3>  </inputs> |

在该元素中，指定了输入的观测文件为 rinex/ 文件夹内的 bjfs1090.17o，chan1090.17o 和 daej1090.17o；输入的广播星历文件为 brdc/ 文件夹内的 brdm1090.17p；输入的精密星历文件为 igs/ 文件夹内的 gbm19453.sp3。

**输出数据配置项**

输出数据配置项指的是 output 元素中的项目，其中的元素有：

* xtr：文本格式的处理报告文件，较繁复；
* xml：XML 格式的处理报告文件，较简洁；
* log：处理日志文件，文本格式。

一个 output 元素的示例如下：

|  |  |
| --- | --- |
| 1  2  3  4 | <outputs append="0" verb="0" >  <xtr> $(rec)2017109.xtr </xtr>  <xml> $(rec)2017109.xml </xml>  </outputs> |

Anubis 将为每个输入的观测数据输出一个报告文件。示例在配置报告文件的输出路径时，使用了 $(rec) 这个变量，它代表输入的观测数据的测站名。

**卫星系统配置项**

卫星系统配置项可以对每个卫星系统做单独的设置，通常用于过滤卫星、频段和观测类型等。其中的卫星系统名采用[上文](http://www.gnss.help/2017/05/12/anubis-install-usage/)提到的 3 个字符的简写方式。每个卫星系统可设置如下参数：

* sat：要处理的卫星编号（默认为所有卫星）；
* type：要处理的观测类型（默认为所有类型）；
* band：要处理的频段（默认为所有频段）；
* attr：要处理的观测值属性（默认为所有属性）。

其中的 attr 项目针对 RINEX 3 标准。在 RINEX 3 中，观测类型由以前的 2 个字符改为 3 个字符。新增的第 3 个字符表示观测值属性，该属性与产生观测值的接收机通道有关。

一个对 GLONASS 系统的示例配置如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <glo>  <sat> R01 R02 R04 </sat>  <type> C L D S P </type>  <band> 1 2 3 </band>  <attr> A B C D I L M N P Q S W X Y Z</attr>  </glo> |

**配置文件示例**

最终形成的文件如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  <!DOCTYPE config>  <config>  <gen>  <beg> "2017-04-19 00:00:00" </beg>  <end> "2017-04-19 23:59:59" </end>  <sys> GPS GAL GLO BDS QZS SBS </sys>  <rec> BJFS CHAN DAEJ </rec>  <int> 30 </int>  </gen>  <qc  sec\_sum="1"  sec\_hdr="1"  sec\_obs="1"  sec\_est="1"  sec\_gap="1"  sec\_bnd="1"  sec\_pre="1"  sec\_ele="1"  sec\_mpx="1"  int\_stp="900"  int\_gap="600"  int\_pcs="1800"  col\_sat="35"  mpx\_nep="15"  mpx\_lim="5"  ele\_cut="10"  pos\_kin="false"  />  <inputs>  <rinexo>  rinex/bjfs1090.17o  rinex/chan1090.17o  rinex/daej1090.17o  </rinexo>  <rinexn> rinex/brdm1090.17p </rinexn>  <sp3> rinex/gbm19453.sp3 </sp3>  </inputs>  <outputs append="0" verb="0" >  <xtr> $(rec)2017109.xtr </xtr>  <xml> $(rec)2017109.xml </xml>  </outputs>  <glo>  <sat> R01 R02 R04 </sat>  <type> C L D S P </type>  <band> 1 2 3 </band>  <attr> A B C D I L M N P Q S W X Y Z</attr>  </glo>  </config> |

**Anubis 配置与使用简介**

[2017-05-12](http://www.gnss.help/2017/05/12/anubis-install-usage/)

[ANUBIS](http://www.gnss.help/categories/Anubis/)

[ANUBIS](http://www.gnss.help/tags/Anubis/), [RINEX质量分析](http://www.gnss.help/tags/RINEX%E8%B4%A8%E9%87%8F%E5%88%86%E6%9E%90/)

[G-Nut/Anubis](http://www.pecny.cz/GOP/index.php/gnss/sw/anubis) 是一个 GNSS 数据质量检核软件，由捷克的国家大地测量、地形与地图制图研究所开发。该软件支持 RINEX 3 文件格式，可以对几乎所有导航卫星系统的观测量进行质量检核与分析操作，并将检核和分析成果绘制成图。

G-Nut/Anubis 软件开源免费，支持 Windows、Linux 和 macOS 等常见的操作系统。类似于 TEQC，该程序也在命令行窗口运行。本文简介该程序的安装配置和使用方法。

**程序简介**

Anubis 的优点在于支持几乎所有导航卫星系统卫星的观测量，并可接受 RINEX 广播星历或精密星历等作为输入的卫星轨道文件。该程序提供的功能包括：

* 多系统卫星的观测量统计；
* 多系统定位计算（SPP）；
* 对观测数据进行预处理；
* 计算卫星方位角与高度；
* 绘制卫星可见性示意图；
* 计算信噪比、多路径效应等的影响；
* 观测数据的信息检查和星历文件合并；
* 以 XML 等的方式输出报告文件。

**安装配置**

**Windows 操作系统**

首先从程序的[下载页面](http://www.pecny.cz/sw/anubis/)获取文件名中带有 “win” 字样的适用于 Windows 操作系统的程序文件。例如，我下载的文件为：anubis-2.1.3-win-static-32b，下载完成后将该文件重命名为 “anubis.exe”。打开“命令提示符”窗口，使用 cd 命令进入程序所在目录，执行 anubis 命令即可使用该程序。

如果你希望在任何目录都能使用该程序，可以考虑将程序所在目录添加到系统的 “Path” 环境变量，或直接将 anubis.exe 移动至 “C:\Windows\System32” 文件夹内。

需要注意的是，如果你使用的是 Windows 7 或者更早的操作系统，可能会遭遇缺失某些动态链接库（.dll文件）的错误。这种情况下，对于能够连接互联网的计算机，可以为系统安装最新的更新；对于无法连接网络的计算机，可以尝试安装微软 VC++ 2015 套件，或者将所缺失的 dll 文件放置到 anubis 程序所在的文件夹来解决。

**Linux 操作系统**

对于 UNIX/Linux 操作系统，下载文件名中含有 “lin” 字样的程序文件，注意程序的适用环境（32 或 64 位系统）。下载完成后将其重命名为 “anubis”，在终端中进入程序所在目录，使用如下命令为其添加执行权限：

|  |  |
| --- | --- |
| 1 | $ chmod +x anubis |

之后就可以通过命令 anubis 使用程序了，但是每次使用前需要切换工作目录。如果你希望在任何目录下都可以使用此程序，可以将其移动到 “/usr/bin” 目录下。

**程序使用**

**接收参数**

Anubis 程序可接收的命令行参数有：

|  |  |
| --- | --- |
| 1 | $ anubis -x <xml> -l <log> -v <int> -X -V -h |

参数释义：

* -x <xml>：输入的配置文件；
* -l <log>：输出的日志文件；
* -v <int>；日志文件详细度；
* -X：输出默认的配置文件；
* -V：显示程序的版本信息；
* -h -help：显示帮助信息。

其中的配置文件以 XML 格式编写。

**配置文件**

使用下面的命令可以输出一个包含默认项的配置文件：

|  |  |
| --- | --- |
| 1 | $ anubis -X 2> anubis\_defaults.xml |

其中，程序的参数配置放在配置文件的 config 元素中，该元素包含以下子项：

* gen：全局配置项，包含采样间隔、卫星系统、站点等；
* gnss：卫星系统配置项，如对卫星和观测量、频段等的筛选；
* qc：数据质量分析检查的配置项；
* input：输入文件配置，包括观测文件与星历文件；
* output：输出文件配置，包括输出数据、质量分析结果、操作日志等。

对于和 GNSS 卫星系统相关的配置，使用卫星系统的 3 个字符简称，它们的对应关系为：

* GPS：GPS；
* GLONASS：GLO；
* Galileo：GAL；
* BeiDou：BDS；
* SBAS：SBS；
* QZSS：QZS。

一份具体的配置文件附在文末，对该文件的介绍请移步本博客 [Anubis 配置文件编辑](http://www.gnss.help/2017/05/22/anubis-config/)。

**使用示例**

Anubis 程序的一个使用示例如下：

|  |  |
| --- | --- |
| 1 | $ anubis -x config.xml -l process.log |

上面的命令从当前目录的 config.xml 文件加载配置信息，然后进行处理，数据处理日志将记录在 process.log 文件中。从该命令中我们并不能得到更多的信息，因为其中的配置项都在 config.xml 文件中。

如果你希望在数据处理时看到更多的信息，可以在使用程序时添加一个 -v 参数，其后跟一个 0 到 9 的整数，此数字越大则输出程度越详细：

|  |  |
| --- | --- |
| 1 | $ anubis -x config.xml -l process.log -v 5 |

查看程序版本号：

|  |  |
| --- | --- |
| 1 | $ anubis -V |

查看程序帮助：

|  |  |
| --- | --- |
| 1 | $ anubis -h |

**配置文件示例**

一份默认的配置文件如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61 | <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  <!DOCTYPE config>  <config>  <gen>  <beg> "2017-05-10 00:00:00" </beg>  <end> "2017-05-10 23:59:59" </end>  <sys> GPS GAL GLO BDS QZS SBS </sys>  <rec> GOPE WTZR POTS </rec>  <int> 30 </int>  </gen>  <qc  sec\_sum="1"  sec\_hdr="1"  sec\_obs="1"  sec\_est="1"  sec\_gap="1"  sec\_bnd="1"  sec\_pre="1"  sec\_ele="1"  sec\_mpx="1"  int\_stp="900"  int\_gap="600"  int\_pcs="1800"  col\_sat="35"  mpx\_nep="15"  mpx\_lim="5"  pos\_kin="0"  ele\_cut="10"  />  <inputs>  <rinexo> file://dir/name </rinexo> <!-- obs RINEX decoder -->  <rinexn> file://dir/name </rinexn> <!-- nav RINEX decoder -->  </inputs>  <outputs append="0" verb="0" >  <flt> file://dir/name </flt> <!-- filter output encoder -->  </outputs>  <gps> <!-- any GNSS constellation: GPS GLO GAL BDS SBS QZS -->  <sat> </sat> <!-- list of GPS satellites: G01 G02 .. or empty(ALL) -->  <type> </type> <!-- list of GPS obs types: C L D S P or empty(ALL) -->  <band> </band> <!-- list of GPS obs bands: 1 2 5 or empty(ALL) -->  <attr> </attr> <!-- list of PGS attributes: A B C D I L M N P Q S W X Y Z or empty(ALL) -->  </gps>  <glo> <!-- any GNSS constellation: GPS GLO GAL BDS SBS QZS -->  <sat> </sat> <!-- list of GPS satellites: R01 R02 .. or empty(ALL) -->  <type> </type> <!-- list of GPS obs types: C L D S P or empty(ALL) -->  <band> </band> <!-- list of GPS obs bands: 1 2 3 or empty(ALL) -->  <attr> </attr> <!-- list of PGS attributes: A B C D I L M N P Q S W X Y Z or empty(ALL) -->  </glo>  <rec id="GOPE" name="GOPE 11502M002" desc="Geodetic Observatory Pecny, Czech Republic" >  <set beg="1995 05 13 00 00 00" end="1997 06 11 00 00 00" rec="TRIMBLE 4000SSE" ant="TRM14532.00 NONE"  X="3979316.0" Y="1050312.0" Z="4857067.0" dX="0.0" dY="0.0" dZ="0.0" dN="0.0" dE="0.0" dU="0.0" />  <set beg="1997 06 11 00 00 00" end="1997 06 20 00 00 00" rec="SPP GEOTRACER100" ant="TRM14532.00 NONE" />  <set beg="1997 06 20 00 00 00" end="1999 11 04 00 00 00" rec="TRIMBLE 4000SSE" ant="TRM14532.00 NONE" />  <set beg="1999 11 04 00 00 00" end="2000 07 24 00 00 00" rec="ASHTECH Z18" ant="ASH701073.3 SNOW" />  <set beg="2000 07 24 00 00 00" end="2000 10 04 00 00 00" rec="TRIMBLE 4000SSE" ant="TRM14532.00 NONE" />  <set beg="2000 10 04 00 00 00" end="2001 07 18 00 00 00" rec="ASHTECH Z18" ant="ASH701073.3 SNOW" />  <set beg="2006 07 14 00 00 00" end="2009 12 14 00 00 00" rec="ASHTECH Z18" ant="TPSCR3\_GGD CONE" />  <set beg="2009 12 14 00 00 00" end="2013 03 19 00 00 00" rec="TPS NETG3" ant="TPSCR.G3 TPSH" />  </rec>  </config> |

**RINEX 3 格式简介**

[2017-04-22](http://www.gnss.help/2017/04/22/rinex3-introduction/)

[RINEX](http://www.gnss.help/categories/RINEX/)

[RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/)

RINEX 3 是最新的 RINEX 格式标准。与之前的版本相比，新的标准对之前的文件类型做了较大幅度的修改，将文件格式精简为观测文件、导航文件和气象文件三种，并能够更好地提供对多卫星系统的支持。目前 IGS 的 [MGEX](http://mgex.igs.org/) 项目已经大量使用该格式。

尽管新的 RINEX 3 标准目前还未得到广泛应用，但可想而知，未来必属于新生事物。RINEX 3.03 发布于 2015 年 7 月，本文修改自我在阅读 [RINEX 3.03 格式说明](http://igscb.jpl.nasa.gov/igscb/data/format/rinex303.pdf)时所做的笔记，总结了 RINEX 3 标准最主要的修改与特色。

**文件类型**

与以前相同的是，RINEX 文件编码依旧只采用了 ASCII 字符。但新的 RINEX 格式标准精简了文件类型，对诸如 GPS 系统星历、GLONASS 系统星历等相似的文件类型进行了合并。合并后，只剩下 3 种文件类型：

* 观测数据文件；
* 导航（星历）文件；
* 气象数据文件。

在新标准的观测文件和导航文件中，既可以包含单一卫星系统的数据，也可以包含来自不同卫星系统的混合数据。文件所包含的卫星系统依然可以通过文件名进行区分。

**命名方式**

新的 RINEX 格式抛弃了以往在文件扩展名中加入观测年份的特点，只包含两种扩展名：.rnx 表示标准的 RINEX 文件；.crx 表示压缩过的 Compact RINEX 格式。个人认为这个改动非常好，统一的后缀名更易于被操作系统、文本编辑器和人类识别。新的 RINEX 文件命名方式为：

|  |  |
| --- | --- |
| 1 | <SITE><RN><CRC>\_<S>\_<YEARDOYHRMN>\_<LEN>\_<FRQ>\_<ST>.<FMT> |

文件名各部分释义：

* <SITE> 为四个字符的观测站点名；
* <RN> 为接收机的编号；
* <CRC> 为三位 [ISO 3166-1 标准](https://zh.wikipedia.org/zh-cn/ISO_3166-1)的国家和地区代码，标识站点位置；
* <S> 为数据源，即数据来源于接收机（R）还是数据流（S）；
* <YEARDOYHRMN> 为观测开始时刻：年、年积日、时、分；
* <LEN> 为观测时段的长度；
* <FRQ> 为观测时的采样间隔或采样频率（星历文件无此项）；
* <ST> 为包含的卫星系统和数据类型，第一位表示卫星系统（M、G、R、C、E、J、I）；第二位为数据类型，即观测文件（O）、导航文件（N）或气象文件（M）；
* <FMT> 为扩展名，扩展名只有两种：rnx 或 crx。

文件名示例：

* ALGO00CAN\_R\_20170420000\_01D\_30S\_MO.rnx 表示数据是来自加拿大的 ALGO 站 0 号接收机，于 2017 年第 42 日 0 点开始观测的，时长 1 天，采样间隔为 30 秒的多系统混合观测数据；
* BJFS00CHN\_S\_20170420100\_15M\_01S\_GO.rnx 表示数据是来自中国的 BJFS 站 0 号接收机的实时数据流，观测开始于 2017 年第 42 日 1 点，时长为 15 分钟，采样间隔 1 秒的 GPS 观测数据；
* ALGO00CAN\_R\_20170420100\_01H\_05Z\_MO.crx 表示数据是来自加拿大的 ALGO 站 0 号接收机，于 2017 年第 42 日 1 点开始观测的，时长 1 小时，采样间隔为 5 Hz 的多系统混合 Compact RINEX 观测数据；
* ALGO00CAN\_R\_20170420000\_01D\_MN.rnx 表示数据是来自加拿大的 ALGO 站 0 号接收机，于 2017 年第 42 日 0 点开始观测，时长 1 天的多系统混合的导航数据；
* BRDC00IGS\_R\_20170420000\_01D\_MN.rnx 表示数据来自 IGS，由 IGS 合并生成的包含多系统所有可用卫星的混合导航数据；
* ALGO00CAN\_R\_20170420000\_01D\_RN.rnx 表示数据是来自加拿大的 ALGO 站 0 号接收机，于 2017 年第 42 日 0 点开始观测，时长 1 天的 GLONASS 系统的导航数据；
* DAVS00ATA\_R\_20170420000\_01D\_30S\_MM.rnx 表示数据来自南极洲 DAVS 站 0 号接收机，于 2017 年第 42 日 0 点开始观测，时长为 1 天的混合气象数据。

**观测文件结构**

在新定义的观测文件中，每个文件只能包含一个站点某一时段的数据。该文件依旧由文件头部分和数据部分组成，文件头部分每行依然只包含 80 列，其中第 61 至 80 列部分为信息标识。但对于观测数据文件的数据部分，已经不在对每行的字符数进行限制。

**文件头信息部分**

文件头信息区无甚大变化，其特点为：

* 每行依然为 80 列；
* 从第一行开始，至包含 “END OF HEADER” 标志行结束；
* 每行前 60 列包含信息，第 61 至 80 列标志信息类型；
* 包含 “COMMENT” 标志的行表示注释；
* 注释可以出现在信息区或数据区；
* 观测类型标识由 “TYPE OF OBSERV” 变为 “OBS / TYPES”；
* 观测类型由两个字符增加为三个字符。

**数据部分**

最大的变化在数据区，其特点为：

* 每行列数不再限制；
* 第一部分为历元/卫星数标识，以 “>” 开头；
* 历元/卫星部分指明观测的时刻、卫星数量；
* 第二部分为观测值；
* 观测值部分首先标识卫星号，然后给出该卫星的各项观测值。观测值的顺序以文件头的信息为准；
* 若某一项观测值缺失，则数据部分为空白。

**导航文件结构**

导航文件亦经过了较大的改变。以往的导航数据文件只能包含单个卫星系统的轨道信息，但新的导航文件可以包含混合的卫星系统的轨道数据。

**文件头信息部分**

文件头信息特点：

* 每行依然为 80 列；
* 文件第一行增加了卫星系统标识；
* 从第一行开始，至包含 “END OF HEADER” 标志行结束；
* 每行前 60 列包含信息，第 61 至 80 列标志信息类型；
* 包含 “COMMENT” 标志的行表示注释；
* 注释可以出现在信息区或数据区。

**数据部分**

数据区特点：

* 每行依然为 80 列；
* 可以从接收机中导出，亦可以从互联网下载；
* 第一部分为卫星号、发布时刻、卫星钟参数；
* 之后以广播轨道 1 到 7 的方式给出卫星的轨道根数；
* 广播轨道每 2h 更新一次。

**格式转化程序**

RINEX 3 数据格式与较旧的 RINEX 2 数据格式之间的格式转化程序主要有 RINEX Converter、GNSS Converter 和 GFZRNX 等。详情可移步之前的文章：[常用的 RINEX 版本格式转换程序整理](http://www.gnss.help/2016/11/27/rinex-version-switch/)。

# INX Editor 程序使用说明

[2017-03-29](http://www.gnss.help/2017/03/29/inx-editor-usage/)

[IONEX](http://www.gnss.help/categories/IONEX/)

[IONEX](http://www.gnss.help/tags/IONEX/), [INX](http://www.gnss.help/tags/Inx/)

IONEX（IONosphere Map EXchange）是最常用的空间电离层数据交换格式。使用者经常需要对该类文件进行操作，例如 IONEX 数据编辑、数据提取以及将文件在二进制格式和文本格式之间相互转换等，但对于此类文件目前尚缺一个好用的操作软件。

[INX Editor](http://ionosphere.cn/inx.html) 是由[武汉大学 IGS 电离层分析中心](http://ionosphere.cn/)开发的一款 IONEX 文件编辑工具，支持平台广，使用方便。本文是对该程序的配置及使用方法简介，并随程序新功能的加入持续更新。

## 程序简介

INX Editor 是一款 IONEX 格式电离层数据编辑软件，它支持 Windows、Linux 以及苹果 macOS 等常见操作系统。该程序于 2018 年 3 月 19 日发布了版本 1.2，相比旧版本，新版本统一了程序的返回值和 TEC 数值的精度。INX Editor 目前支持的操作有：

* IONEX 数据的加密、抽稀；
* IONEX 文件自定义范围截取；
* 计算格网内任意位置的电子含量；
* IONEX 文本文件与二进制格式转换。

## 环境配置

要获取 INX Editor 程序，可直接到相应的 [Github](https://github.com/1acheng/INX_Editor) 页面或者[武汉大学 IGS 电离层分析中心的 FTP 服务器](ftp://ftp.ionosphere.cn/software/inx_editor/)获取对应操作系统的可执行文件。

### Windows 操作系统

对于 Windows 操作系统，下载对应的可执行文件后，打开 “命令提示符” 窗口，使用 cd 命令进入程序所在目录，键入程序名 inx 即可运行程序。

当然，如果你希望在任何目录都能运行该程序，可以考虑将程序所在目录加入到系统的 Path 环境变量，或直接将 inx.exe 文件移动至 “C:\Windows\System32” 文件夹内。

### Linux 或 macOS 操作系统

对于 Linux 或苹果 macOS 操作系统，下载对应的可执行文件后，用 cd 命令进入程序所在文件夹，首先为程序分配可执行权限：

|  |  |
| --- | --- |
| 1 | $ chmod +x inx |

然后就可以通过命令 inx 使用程序了，但是每次使用前需要切换工作目录。如果你希望在任何目录下都可以使用此程序，可以将其移动到 “/usr/bin” 目录下。

## 参数说明

INX Editor 程序的参数输入方式为：

|  |  |
| --- | --- |
| 1 | $ inx <mode> <options> |

其中的 <mode> 参数表示操作模式。该程序目前有 4 种操作模式：edit、tec、i2b 以及 b2i。这里的 edit模式用于对输入文件的数据进行编辑，如数据截取、数据采样率的加密与抽稀；tec 模式用来内插计算格网内任意位置的电子含量（TEC）；而 i2b 和 b2i 用于数据在二进制格式与文本格式之间进行转换。

<options> 表示模式的操作选项，取决于模式的不同，可接受的参数略有差别。edit 模式可接受的参数选项如下：

* -i <file>、--input <file> 指定输出的文件；
* -o <file>、--output <file> 指定输出的文件；
* -s <time>、--start <time> 指定文件截取时的开始时间；
* -e <time>、--end <time> 指定文件截取时的结束时间；
* -n <int>、--interval <int> 指定数据的采样间隔；
* --lat0 <lat> 指定文件截取时的起始纬度；
* --lat1 <lat> 指定文件截取时的终止纬度；
* --dlat <int> 指定文件截取时的纬度间隔；
* --lon0 <lon> 指定文件截取时的起始经度；
* --lon1 <lon> 指定文件截取时的终止经度；
* --dlon <int> 指定文件截取时的经度间隔。

tec 模式可接受的参数为：

* -i <file>、--input <file> 指定输出的文件；
* -t <time>、--time <time> 指定内插的时间；
* -b <lat>、--lat <lat> 指定内插的纬度；
* -l <lon>、--lon <lon> 指定内插的经度。

i2b 和 b2i 模式可接受的参数为：

* -i <file>、--input <file> 指定输出的文件；
* -o <file>、--output <file> 指定输出的文件。

当然，少不了用于显示版本号和帮助信息的参数：

* -v、--version 显示版本号；
* -h、--help 显示帮助。

## 使用示例

### 文件编辑

#### 数据加密与抽稀

通过改变 IONEX 文件的采样率可以实现对数据的加密或抽稀。使用 INX Editor 程序进行此操作非常简单，只需指定输入文件、输出文件和输出文件的采样间隔，注意这里的采样率以秒为单位。

下面的命令将文件 whub0420.17i 的采样间隔加密为 1 小时并输出到 whub0420\_hour.17i：

|  |  |
| --- | --- |
| 1 | $ inx edit -i whub0420.17i -n 3600 -o whub042\_hour.17i |

下面的命令将文件 whub0420.17i 的采样间隔更改为 3 小时并输出到 whub0420\_3hrs.17i：

|  |  |
| --- | --- |
| 1 | $ inx edit --input whub0420.17i --interval 10800 --output whub0420\_3hrs.17i |

#### 数据截取

我们可以通过自定义范围的方式来截取 IONEX 文件，这里的范围既可以是时间范围，也可以是经纬度指定的空间范围。

在时间范围上截取文件时，需要指定输入文件、输出文件、输出文件的起止时刻以及时间间隔。示例，截取 whub0420.17i 文件中从 3 点开始至 9 点结束，时间间隔为 1 小时的数据，并输出到 whub042c.17i：

|  |  |
| --- | --- |
| 1 | $ inx edit --input whub0420.17i --start 3 --end 9 --interval 3600 --output whub042c.17i |

在空间范围上截取文件时，需要指定输入文件、输出文件、输出文件的经纬度范围和经纬度间隔。例如，截取纬度范围从 30° 至 80°，间隔为 10°，经度范围从 80° 至 160°，间隔为 20° 的数据，并输出到 whub0420\_part.17i：

|  |  |
| --- | --- |
| 1 | $ inx edit -i whub0420.17i --lat0 30 --lat1 80 --dlat 10 --lon0 80 --lon1 160 --dlon 20 -o whub0420\_part.17i |

如果你已经足够熟练，edit 模式的操作还可以合并执行。例如，从文件 whub0420.17i 中截取 5 点至 18 点、采样间隔为 1 小时，纬度范围从 20° 至 75° 间隔为 5°、经度范围从 10° 至 160° 间隔为 5° 的数据，将最终的处理结果输出到 whub0420\_part.17i：

|  |  |
| --- | --- |
| 1 | $ inx edit -i whub0420.17i -s 5 -e 18 -n 3600 --lat0 20 --lat1 75 --dlat 5 --lon0 10 --lon1 160 --dlon 5 -o whub0420\_part.17i |

### 电子含量计算

由于 IONEX 文件是一个格网模型，因此当需要某个位置的 TEC 时，通常使用内插法计算。INX Editor 的 tec 模式可以内插计算格网内任意位置的 TEC，只要指定输入文件、内插的时刻和内插点的位置。

武汉市的经纬度约为东经 114.3°，北纬 30.5°，下面的命令将输出武汉在 GMT 时间为 2017 年第 45 日 6 点时的电离层电子含量：

|  |  |
| --- | --- |
| 1 | $ inx tec --input whub0420.17i --time 6 --lat 30.5 --lon 114.3 |

### 格式转换

INX Editor 还可以将 IONEX 文件在文本文件与二进制文件之间转换。其中 i2b 模式将文本文件转换为二进制文件；与之相对的，b2i 模式将二进制文件转换为文本文件，使用时只需额外指定输入文件和输出文件。

示例，将文本格式的 whub0420.17i 转化为二进制文件 whub0420.dat：

|  |  |
| --- | --- |
| 1 | $ inx i2b --input whub0420.17i --output whub0420.dat |

将二进制文件 whub0420.dat 转化为 IONEX 格式的电离层数据 whub0420.17i：

|  |  |
| --- | --- |
| 1 | $ inx b2i --input whub0420.dat --output whub0420.17i |

### 其他操作

查看程序版本号与开发者信息：

|  |  |
| --- | --- |
| 1 | $ inx --version |

查看程序使用帮助：

|  |  |
| --- | --- |
| 1 | $ inx --help |

## 开发者信息

INX Editor 由[武汉大学 IGS 电离层分析中心](http://ionosphere.cn/)的王成博士等开发和维护，在使用中有任何意见、建议或发现 Bug，欢迎通过在 [Github](https://github.com/1acheng/INX_Editor) 上提交 Issue 或[发送 Email](mailto:ac@ionosphere.cn) 向开发者反馈。本文将随着程序新功能的加入持续更新。

**RINEX 与 Compact RINEX 批量转化脚本**

[2017-03-27](http://www.gnss.help/2017/03/27/pinot-crx-rnx-switch/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PINOT](http://www.gnss.help/tags/Pinot/), [PYTHON](http://www.gnss.help/tags/Python/), [RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/)

前文 [RINEX 与 Compact RINEX 格式的转换](http://www.gnss.help/2016/08/30/Rnx-Crx-Switch/)曾经介绍过使用 RNXCMP 程序包或者 GAMIT/GLOBK 软件的内置脚本将 GNSS 观测数据在标准 RINEX O-文件格式和 Compact RINEX （D-文件）之间转化的方法。但问题是，RNXCMP 程序包不善于对大量数据批量操作，而很多时候又无法使用 GAMIT/GLOBK 内置脚本。

本文介绍 [Pinot](http://www.gnss.help/tags/Pinot/) 中的新增脚本：crnx2rnx.py 和 rnx2crnx.py，可实现对该情况的批量处理。

**脚本功能**

本文所涉及的两个脚本及其功能如下：

* crnx2rnx.py：将 Compact RINEX 转化为标准 RINEX；
* rnx2crnx.py：将标准 RINEX 转化为 Compact RINEX。

**运行环境**

两个脚本使用 Python 语言（Python 3.5）编写，依赖于 RNXCMP 程序包中的两个程序：crx2rnx 和 rnx2crx。要使用此脚本，需将这两个程序放入脚本执行目录或系统搜索目录内。

需注意的是：对于某些 Windows 操作系统中（如 Windows Server 2008 R2），可能会出现 “’crx2rnx/rnx2crx’ 不是内部或外部命令，也不是可运行的程序或批处理文件。” 的错误，此时需要保证运行脚本的文件夹内有 crx2rnx 或 rnx2crx 程序。

此外，最新的程序实现了进度条显示功能。该功能使用了 [tqdm](https://pypi.python.org/pypi/tqdm) 模块，要安装该模块，在 “命令提示符” 窗口执行：

|  |  |
| --- | --- |
| 1 | $ pip install tqdm |

**参数说明**

此两个脚本的所接受的参数一致，下面以 crnx2rnx.py 为例。该脚本接受的参数如下：

|  |  |
| --- | --- |
| 1 | $ python crnx2rnx.py <file> [<file> ...] [-out <directory>] [-k] [-r] [-v] [-h] |

参数释义：

* <file>：输入的待处理文件，你可以使用文件名，也可以使用通配符；
* -out <directory>：输出转化结果的文件夹。该项默认为当前目录的 rinex（或 crinex） 文件夹；
* -k、--keep：转化成功后是否保留源文件，不加该参数时默认为不保留；
* -r、--recursive：指定 “\*\*/” 的通配符前缀表示递归搜索子文件夹的内容，不加该参数默认为不递归；
* -v、--version：显示版本信息；
* -h、--help：显示帮助。

**使用示例**

将 crinex/ 文件夹内 BJFS 站和 SHAO 站的所有观测于 2016 年的 Compact RINEX 文件转化为标准 RINEX 格式，将输出保存至 rinex/ 文件夹内，转化成功后删除源文件：

|  |  |
| --- | --- |
| 1 | $ python crnx2rnx.py crinex/bjfs\*.16d crinex/shao\*.16d -out rinex |

将 crinex/ 文件夹及其子文件夹内所有以 .crx 为后缀名的 Compact RINEX 文件转化为标准 RINEX 格式，将输出保存至 rinex/ 文件夹内并保留源文件：

|  |  |
| --- | --- |
| 1 | $ python crnx2rnx.py crinex/\*\*/\*.crx -out rinex -r -k |

将 rinex/ 文件夹内的所有观测于 2016 年第 161 至 168 日的标准 RINEX O-文件压缩为 Compact RINEX 格式，将输出保存至 crinex/ 文件夹并保留源文件：

|  |  |
| --- | --- |
| 1 | $ python rnx2crnx.py rinex/\*16[1-8]?.16o -out crinex -k |

**下载地址**

该脚本及同一工具包的其他脚本均放在 Github 网站上：[purpleskyfall/pinot](https://github.com/purpleskyfall/pinot)。

**修改记录**

* 2017-09-10，版本 0.2.0：
  + 去掉 -dir 和 -glob 参数，使用位置参数指定输入文件；
  + 支持多个输入文件或通配符；
  + 支持 RINEX 3 文件名；
  + 添加进度条显示；
  + 添加多线程支持。
* 2018-03-14，版本 0.2.2：
  + 优化在 UNIX/Linux 中的输出信息显示。
* 2018-03-16，版本 0.2.3：
  + 适配 UNIX/Linux 操作系统。

# 已知点坐标文件的创建与使用

[2017-03-24](http://www.gnss.help/2017/03/24/create-apr/)

[GLOBK](http://www.gnss.help/categories/GLOBK/)

[GLOBK使用](http://www.gnss.help/tags/GLOBK%E4%BD%BF%E7%94%A8/)

已知点坐标文件（.apr 文件）是使用 GLOBK 进行网平差时，用于输入已知点坐标的文件。GAMIT/GLOBK 程序所附带的 APR 文件，一般只有各 IGS 核心站在 ITRF 框架下的坐标和速度。因此，使用自带的 APR 文件，可以将解算点的坐标归化到某一 ITRF 框架内。但是，如果想获得站点在 CGCS2000 国家坐标系的位置，或已知点坐标不在自带的 APR 文件中时，我们就需要制作自己的 APR 文件。

本文将介绍 APR 文件的制作和使用方法。

## 文件格式

打开任何一个 APR 文件，都可以预览其文件的格式。APR 文件中每行包含一个已知点的信息。它一般长这样：

BJFS\_2PS -2148744.08805 4426641.26603 4044655.92200 -0.03124 -0.00577 -0.00668 2005.0

其中的信息依次为站点名、站点在某历元下的三维坐标、站点三维速度以及坐标历元。有了这些信息，就可以很容易地计算出该站点在任何历元下的站坐标。

## 创建方法

尽管可以按照 APR 文件的格式来手动创建自己的 APR 文件，但一般不推荐这么做，毕竟手动编辑的方式很难保证文件格式的正确性。GAMIT/GLOBK 程序中提供的 rx2apr 命令可以提取 RINEX 观测文件中的概略坐标来创建 APR 文件，这是获得 APR 文件模板的好办法。虽然概略坐标的精度很难达到作为已知点坐标的要求，但我们在获得 APR 文件后只需对坐标值做一些校正即可。

实际上，使用 rx2apr 命令来创建 APR 文件的方法在前文 [GAMIT 分步进行基线解算](http://www.gnss.help/2016/08/27/GAMIT-StepbyStep/)早已经提到过，制作概略坐标文件时曾经将产生的 APR 文件作为一个中间文件。

grep 是 UNIX/Linux 操作系统自带的一个程序，它从文本文件中查找并输出符合条件的行，用于匹配文本的模式通过正则表达式指定。该程序的使用教程网络上比比皆是，这里不深究，只提及一下其操作模式。grep 程序的基本操作模式为：

|  |  |
| --- | --- |
| 1 | $ grep <pattern> <input\_file> |

其中 <pattern> 为用于搜索的正则表达式，<input\_file> 为要处理的文件。

假定现在位于解算项目文件夹，首先使用 grep 提取所有观测文件中的概略坐标，并将输出重定向到文件中：

|  |  |
| --- | --- |
| 1 | $ grep "APPROX POSITION XYZ"$ rinex/\*o > lfile.rnx |

然后使用 rx2apr 程序将 lfile.rnx 转化为 APR 文件：

|  |  |
| --- | --- |
| 1 | $ rx2apr lfile.rnx <year> <doy> |

这里的参数 <year> 和 <doy> 要求你输入以年和年积日表示的坐标历元。例如，我使用的已知点坐标历元处于 2016 年第 256 日，可以这样执行：

|  |  |
| --- | --- |
| 1 | $ rx2apr lfile.rnx 2016 256 |

运行命令后，将在操作目录生成文件 lfile.rnx.apr。这样得到的 APR 文件中的站速度一般都是零。你当然可以将其编辑为满意的值，但实际上如果已知点坐标是近期的，任其站速度为零也无不可。

## 应用配置

成功制作 APR 文件后，下一步就是使用它。如你所知，GLOBK 程序的控制文件是两个 .cmd 文件，我们需要修改它们来设置网平差时使用的已知点坐标文件。我制作的 APR 文件名为 my\_regional.apr，位于项目文件夹的 tables 目录内，下面将以它为例。

分别打开 globk\_comb.cmd 和 glorg\_comb.cmd 文件，找到类似这样的一行：

apr\_file ~/gg/tables/itrf08\_comb.apr

这一项就是对网平差时使用的 APR 文件的配置。将其中的 APR 文件路径修改为实际要使用的项：

apr\_file ../tables/my\_regional.apr

然后在 glorg\_comb.cmd 文件中设置固定站列表，在该文件最后找到类似这样的内容：

# This should match the well-determined sites in the apr\_file  
stab\_site clear  
source ~/gg/tables/igb08\_hierarchy.stab\_site

这里的第一行为注释，第二行将之前配置的固定站列表清空，第三行从 GAMIT 安装目录的 tables 文件夹中加载包含固定站列表的“igb08\_hierarchy.stab\_site”文件。

我们直接将第三行注释掉，然后在其之前添加一个固定站列表即可。最终这一部分配置如下（注意保持非注释行第一列为空格）：

# This should match the well-determined sites in the apr\_file  
stab\_site clear  
stab\_site +well  
stab\_site +demo  
stab\_site +work  
x source ~/gg/tables/igb08\_hierarchy.stab\_site

将这两个控制文件应用到 GLOBK 网平差操作，得到的就是使用自定义固定站的解。如果设置的已知点未能成功固定，可以适当调整联测的网形或变更所使用的固定点。

**站点信息文件的创建与更新**

[2017-03-05](http://www.gnss.help/2017/03/05/create-station-info/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/)

站点信息文件 station.info 是使用 GAMIT 程序进行数据处理前必须准备的文件之一，该文件包含了各站点的接收机/天线类型、天线高和观测时段等信息。

使用 GAMIT/GLOBK 程序进行数据解算时，通常需要创建或者更新站点信息文件。本文将详细介绍该文件的文件格式以及创建和更新方法。

**文件结构**

站点信息文件 station.info 于基线解算阶段被使用。在 GAMIT/GLOBK 程序的安装目录的 tables/ 文件夹内，就存在一个 station.info 文件示例。该文件包含了所有 IGS 站的站点信息，并随着 IGS 站点的变化（如新站点的加入、接收机或天线的更换等）进行更新。最新的 station.info 文件可以从[SOPAC 网站的相关页面](http://sopac.ucsd.edu/GAMIT.shtml)下载。

一个作为示例的 station.info 文件内容如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | \* Gamit station.info  \*  \* Generated by SOPAC on 02-Mar-2017 @ 23:00:02 UTC  \* Send questions, comments or concerns to devel@gpsmail.ucsd.edu  \*  \*SITE Station Name Session Start Session Stop Ant Ht HtCod Ant N Ant E Receiver Type Vers SwVer Receiver SN Antenna Type Dome Antenna SN  BJFS Beijing Fangshan 1995 263 00 00 00 2010 142 00 00 00 0.0460 DHPAB 0.0000 0.0000 ASHTECH Z-XII3 CD00-1D02 9.20 LP3332 ASH700936B\_M SNOW 1126  BJFS Beijing Fangshan 2010 142 00 00 00 2010 218 00 00 00 0.0465 DHPAB 0.0000 0.0000 TRIMBLE NETR8 4.01 4.01 4922K35365 TRM59800.00 SCIS 4921353141  BJFS Beijing Fangshan 2010 218 00 00 00 2015 330 00 00 00 0.0465 DHPAB 0.0000 0.0000 TRIMBLE NETR8 4.17 4.17 4922K35365 TRM59800.00 SCIS 4921353141  BJFS Beijing Fangshan 2015 330 00 00 00 2016 086 00 00 00 0.0465 DHPAB 0.0000 0.0000 TRIMBLE NETR9 4.81 4.81 5413K48204 TRM59900.00 SCIS 5317361110  BJFS Beijing Fangshan 2016 086 00 00 00 9999 999 00 00 00 0.0465 DHPAB 0.0000 0.0000 TRIMBLE NETR8 4.17 4.17 4922K35365 TRM59900.00 SCIS 5317361110 |

在该文件中，以“\*”号开头的行为注释。其中注明了文件用途、创建时间和反馈邮箱等。最后一行注释为表头，标明了每一列的意义。从左到右依次为：站点名、详细站名、观测开始时刻、观测结束时刻、天线高、天线高量测方式、天线北方向偏移、天线东方向偏移、接收机类型、接收机版本、接收机版本号、接收机编号、天线类型、天线罩类型、天线编号。

其中最重要的部分是：站点名、开始观测时刻、结束观测时刻、天线高、天线高量测方式、天线北方向偏移、天线东方向偏移、接收机类型和天线类型。这些信息不能为空，否则解算时将会报错。

该文件注释之后的部分逐行给出每个测站在一定时期的观测信息。譬如，第一行是 BJFS 站从 1995 年第 263 日到 2010 年第 142 天的观测信息，而最后一行的观测结束日期为“9999 999”，说明该站点信息至今仍然有效。

**创建与更新**

知道了站点信息文件 station.info 的文件结构之后，我们就可以按照其格式对该文件进行创建与编辑了。但在实际应用中，我们通常不会完全手动创建或编辑它，大多时候使用脚本进行操作，然后在必要时做小幅修改。

**参数说明**

GAMIT/GLOBK 程序包中提供了用于操作站点信息文件的脚本：sh\_upd\_stnfo，该脚本使用 C Shell 语言编写并保存在 GAMIT/GLOBK 安装目录的 com/ 文件夹内。使用该脚本可方便地对站点信息文件进行创建和更新，其接受的主要参数有：

|  |  |
| --- | --- |
| 1 | $ sh\_upd\_stnfo -ref <rsfile> -merge <sfiles> -files <rfiles> -i <ifiles> -x <snxfiles> -replace <replace\_option> -w <outfile> -h <slant\_hgt> -t <dup\_tol> -apr -c -nodup -nosort |

参数释义：

* -ref <rsfile> 定义输出信息格式的参考 station.info 文件，该项默认为当前目录的 station.info 文件；
* -merge <sfiles> 需要与参考文件进行合并的 station.info 文件；
* -files <rfiles> 用来更新 station.info 的 RINEX 观测文件；
* -i <ifiles> 用来更新 station.info 的 IGS log 文件；
* -x <snxfiles> 用来更新 station.info 的 SINEX 文件；
* -replace <replace\_option>，当输入文件既有 SINEX 文件，又有 RINEX 观测文件时，该项可用于对输入的站点信息进行检查。可接受三个值之一：all，diff，none。其中 none 意为只检查信息但不添加新站点；diff 意为只添加不同的站点；all 意为不管站点是否相同，都将其添加到输出文件；
* -w <outfile> 合并操作之后的输出文件，默认为 station.info.new；
* -h <slant\_hgt> 天线高阈值，在 RINEX 文件中的天线高若大于该值，将被认为是斜高；
* -t <dup\_tol> 判定为重复项的阈值，默认为 120 秒，即若两个同名站点观测文件的观测时间相隔 120 秒以上，即认为不是重复项;
* -apr 操作时将站点概略坐标输出到一个 .apr 文件中；
* -c 输出时将原有的注释拷贝至新的输出文件中;
* -nodup 在更新文件时不要添加重复的项目；
* -nosort 不要对输出文件中的站点进行排序。

**使用示例**

本文并没有列出该脚本的所有参数，但已经如此冗长。根据帕列托法则，我们实际使用中通常只会用到有限的几个参数。所有在此只举几个平时操作中使用地命令作为例子。

示例，在解算目录的 tables 文件夹内执行下面的命令来更新其中的 station.info 文件：

|  |  |
| --- | --- |
| 1 | $ sh\_upd\_stnfo -files ../rinex/\*o |

该命令直接将 rinex 文件夹内的所有 O-文件的站点信息添加到当前目录的 station.info 文件中，即参考文件和输出文件使用同一默认值。正因如此，若 tables 文件夹内不存在 station.info 的文件，程序将报错并退出。

使用 SINEX 文件更新 station.info，其中 SINEX 文件为 cod19355.snx，参考文件使用 station.info，并将更新后的信息输出至 station.info.cod：

|  |  |
| --- | --- |
| 1 | $ sh\_upd\_stnfo -x cod19355.snx -ref station.info -w station.info.cod |

通过 IGS log 文件将一个站点的信息加入到站点信息文件，并输出至 station.info.drao：

|  |  |
| --- | --- |
| 1 | $ sh\_upd\_stnfo -ref station.info -i drao.log -w station.info.drao |

合并站点信息文件 station.info.sopac 和 station.info.regional，并将合并后的信息输出至 station.info.new：

|  |  |
| --- | --- |
| 1 | $ sh\_upd\_stnfo -ref station.info.sopac -merge station.info.regional -w station.info.new |

**自动更新**

你或许已经注意到，使用 GAMIT 批量处理多天的观测数据时，程序每次解算前都会自动更新 station.info 文件。因为解算控制文件 sites.defaults 中默认就是如此设置的。如果不希望它自动更新（这可能会耗费很长的时间），也可以通过设置关闭解算此功能。打开项目 tables 文件夹内的 sites.defaults 文件，找到类似的这样一行：

all\_sites expt xstinfo

该配置意为：对于解算项目 expt，对所有的站点都不自动更新站点信息文件。我们只需将此处的“expt”替换为自己实际的项目名，就能关闭站点信息文件自动更新功能。当然，若日后希望重新将开启此功能，只需在本行行首添加一个“#”将其注释掉即可。

**补充**

脚本 sh\_upd\_stnfo 更新 station.info 文件的操作是很慢的，尤其是原本的 station.info 文件就比较大时。所以，为节约数据处理时间，原来的 station.info 文件应该尽可能得小，甚至可以是一个只包含表头注释的空文件。

在解算长时间的大量观测数据时，若期间站点的观测信息没有变化，如果希望节约时间，与其设置每次解算前自动更新站点信息文件，不如在其第一次更新后直接手动编辑该文件中每个站点信息的观测结束时间，一劳永逸。此外，如果你需要长时间维护某一固定的测站网络，使用 station.info 作为测站信息变更的记录文件也很好用。

赏

# 获取站点概略坐标的几种方法

[2017-02-26](http://www.gnss.help/2017/02/26/get-approx-position/)

[RINEX](http://www.gnss.help/categories/RINEX/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/), [TEQC](http://www.gnss.help/tags/TEQC/)

在 RINEX 观测数据（O-文件）的文件头信息中，包含观测点的概略坐标值。并且，在大多数的数据处理时，这一项都是必要的。例如，对于 GAMIT 或 PANDA 等 GNSS 数据处理程序而言，若观测文件中缺失概略坐标或概略坐标值有误，将会造成解算失败。

然而，有些观测点由于操作人员的配置或操作失误，常常会造成观测文件中概略坐标缺失或错误的现象。本文试总结了一些获取站点概略坐标的方法。

## 使用 TEQC

前文 [TEQC 程序观测质量检查功能简介](http://www.gnss.help/2017/01/06/teqc-quality-check/)中曾介绍过 TEQC 程序的质量检查功能。该程序在完整模式（full）的质量检查输出结果文件中，包含“antenna WGS 84”一项。此处即是 TEQC 程序计算出的观测站点概略坐标。

当然，要使用完整模式的质量检查，必须引入同时段的广播星历文件。例如，使用 TEQC 程序计算 WUHN 站点在 2016 年第 151 日的概略坐标，可以使用如下命令：

|  |  |
| --- | --- |
| 1 | $ teqc +qc -rep -nav brdc1510.16n wuhn1510.16o |

运行上述命令后，在输出信息中找到 “antenna WGS 84” 项，即可获得该站点在 WGS84 坐标框架内分别以大地坐标系（BLH）和空间直角坐标系（XYZ）表示的概略坐标值。该命令在我的计算机上运行结果为：

antenna WGS 84 (xyz) : -2267782.6195 5009151.3297 3221297.1568 (m)  
antenna WGS 84 (geo) : N 30 deg 31’ 53.95” E 114 deg 21’ 27.32”  
antenna WGS 84 (geo) : 30.531653 deg 114.357590 deg  
WGS 84 height : 38.6097 m

## 使用 GAMIT 内置脚本

你或许还不知道，GAMIT/GLOBK 程序已经包含了计算站点概略坐标的脚本：sh\_rx2apr。该脚本可输出站点在 WGS84 坐标框架内分别以大地坐标系（BLH）和空间直角坐标系（XYZ）表示的概略坐标值。脚本可接受的完整参数表如下：

|  |  |
| --- | --- |
| 1 | $ sh\_rx2apr -site <site> -nav <nav> -ref <ref> -apr <apr> -chi <val> |

其中：<site> 表示待求站点的观测数据文件，<nav> 表示广播星历文件，<ref> 为相对定位时使用的参考站点观测文件，apr 为包含参考站点坐标的 .apr 文件，<chi> 为迭代计算时的阈值（该项默认为 3 m）。

该脚本输出概略坐标有三种模式：直接从观测文件头信息中获取、使用伪距单点定位方式解算或以一个已知点为参考站使用相对定位的方式解算。使用哪一种模式取决于你的输入：若你只输入了观测文件，则使用直接从观测文件头信息中获取的方式；若你还同时引入了一个广播星历文件，则使用伪距单点定位的方式；若你在输入待求测站观测文件的同时还引入了广播星历文件、已知站点的观测数据以及包含已知点坐标的 .apr 文件，则采用相对定位的方式。

值得说明的是：尽管从 GAMIT 10.61 开始，其基线解算功能已经支持 RINEX 3 格式的数据，但该 sh\_rx2apr 脚本尚未兼容 RINEX 3。在使用该脚本前，需要首先将 RINEX 3 的观测文件转换到 RINEX 2。

示例，从观测文件头信息中获取 WUHN 站的先验坐标：

|  |  |
| --- | --- |
| 1 | $ sh\_rx2apr -site wuhn1510.16o |

使用伪距单点定位的方式获取 WUHN 站的先验坐标：

|  |  |
| --- | --- |
| 1 | $ sh\_rx2apr -site wuhn1510.16o -nav brdc1510.16n |

使用相对定位的方式获取 WUHN 站的先验坐标，其中参考站为 SHAO，参考站坐标文件为 itrf08.apr：

|  |  |
| --- | --- |
| 1 | $ sh\_rx2apr -site wuhn1510.16o -nav brdc1510.16n -ref shao1510.16o -apr ../tables/itrf08.apr |

执行以上命令后，将在脚本执行目录下看到输出的两个文件：lfile.wuhn 和 wuhn.apr。其中 lfile.wuhn 中的概略坐标以大地坐标表示，而 wuhn.apr 中的概略坐标以空间直角坐标表示，它们分别是老版本和新版本的 L-文件。

## 使用 G-Nut/Anubis

如果你的计算机中安装了 [G-Nut/Anubis](http://www.gnss.help/2017/05/12/anubis-install-usage/) 程序，使用 G-Nut/Anubis 获得站点的概略坐标也非常方便。该程序使用标准单点定位（SPP）的方法计算站点的位置，支持 RINEX 3 文件格式，也支持以精密星历作为输入的轨道数据。

因为 Anubis 程序的输入信息使用 XML 文件进行配置，因此你需要根据自己的需求修改其配置文件。对该程序配置文件的详细介绍可移步：[Anubis 配置文件编辑](http://www.gnss.help/2017/05/22/anubis-config/)，这里只介绍获取概略坐标时的必要操作。

首先使用下面的命令获取一个默认的配置文件：

|  |  |
| --- | --- |
| 1 | $ anubis -X 2> anubis\_defaults.xml |

编辑配置文件 anubis\_defaults.xml：在 gen 节点中，将其中的 beg 和 end 节点的时刻修改为观测数据的开始和结束时刻，将 rec 节点中的站点名替换为你的站点名；在 inputs 节点中，将 rinexo 节点中的内容替换为你的 RINEX 观测文件的实际路径，将 rinexn 节点中的内容替换为你的 RINEX 广播星历文件的路径；在 outputs 节点中，将其中的 flt 节点名替换为 xtr，并将其内容替换为你希望的文件输出路径。

上述修改完成后，执行 Anubis 程序的质量检查命令，即可得到一个质量检查输出的结果文件：

|  |  |
| --- | --- |
| 1 | $ anubis -x anubis\_defaults.xml -l process.log |

查看程序输出的 .xtr 报告文件，从 “Estimated values” 部分即可找到站点在大地坐标系和空间直角坐标系下的概略坐标信息。使用我的计算机处理 HERT 站在 2017 年 7 月 1 日的观测数据，所得的这部分结果为：

#====== Estimated values (v.1)  
=PERIOD 2017-07-01 00:00:00 2017-07-01 00:00:00 2017-07-01 23:59:30  
=SAMPLE 2017-07-01 00:00:00 30.000  
=XYZGPS 2017-07-01 00:00:00 4033461.4596 23537.7981 4924318.9564 0.8 0.6 1.3 96 0  
=BLHGPS 2017-07-01 00:00:00 50.867479690 0.334353309 84.2165 0.7 0.9 1.3 96 0

## 补充

当然，计算站点概略坐标的方法还有很多。比如：你也可以使用 Bernese 程序的精密单点定位（PPP）解算模式或 RTKLIB 等程序对站点观测数据进行处理来得到更精确的站点坐标。本文介绍的方式仅限为较简单的几种。如果你有本文未提及的简单方法，欢迎发送邮件进行补充。

**Pinot 脚本文档索引目录**

[2017-02-16](http://www.gnss.help/2017/02/16/pinot-content/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PINOT](http://www.gnss.help/tags/Pinot/), [PYTHON](http://www.gnss.help/tags/Python/), [RINEX编辑](http://www.gnss.help/tags/RINEX%E7%BC%96%E8%BE%91/)

[Pinot（Pinot is not only TEQC）](https://github.com/purpleskyfall/pinot) 是一系列使用 Python 语言开发的脚本，主要用于对大量的 GNSS 静态数据进行批量处理，特别适合处理 CORS 站等静态数据。该项目目前以开源的形式托管在 [Github](https://github.com/purpleskyfall/pinot) 网站上，也算做了一点微小的工作。

本站多篇文章与该软件包中的程序有关。本文是该软件包的简介及文档索引目录，以方便进行查找。

**Pinot 简介**

**运行环境**

Pinot 部分脚本依赖于 [TEQC](https://www.unavco.org/software/data-processing/teqc/teqc.html)、[RNXCMP](http://terras.gsi.go.jp/ja/crx2rnx.html) 或 [runpkr00](http://kb.unavco.org/kb/article/trimble-runpkr00-v5-40-latest-version-mac-osx-10-7-windows-xp-7-linux-solaris-744.html) 程序，并且需要 tqdm 和 PyYAML 模块的支持。要使用此套脚本的完整功能，需保证系统已安装或配置这些程序。

tqdm 模块的安装命令为：

|  |  |
| --- | --- |
| 1 | $ pip install tqdm |

TEQC 程序配置请参考本站文章 [TEQC 程序配置及其数据转换功能的使用](http://www.gnss.help/2016/09/17/teqc-config-translate/)，RNXCMP 配置请参考本站 [RINEX 与 Compact RINEX 格式的转换](http://www.gnss.help/2016/08/30/Rnx-Crx-Switch/)，runpkr00 程序配置请参考本站[runpkr00 程序配置及使用](http://www.gnss.help/2017/01/19/runpkr00-config-usage/)，PyYAML 安装请参考本站 [PyYAML 安装与使用演示](http://www.gnss.help/2016/12/01/install-pyyaml/)。

Pinot 系列脚本在运行 Python 3.5 及更高版本的 RHEL 7、CentOS 7、Windows 7 SP1、Windows Server 2008 R2 SP1 以及 Windows 10 等操作系统下测试通过，其他操作系统使用者请自行测试。对于 UNIX/Linux、macOS 等操作系统，通过 dos2unix 命令转换程序的字符编码，然后配置执行权限之后即可使用。dos2unix 程序的使用方法请参考 [DOS2UNIX 安装与使用简介](http://www.gnss.help/2017/07/24/dos2unix-install-usage/)。

需注意的是：对于 Windows 10 或 Windows 7 操作系统，只需将 TEQC 或 runpkr00 程序放入系统能搜索到的路径下，即在任意目录中启动命令提示符窗口执行 teqc 命令能显示 TEQC 程序的提示信息；但对于更低版本的 Windows 操作系统，需保证运行脚本的文件夹内有 TEQC、RNXCMP 或 runpkr00 程序，否则可能出现类似 “’teqc’ 不是内部或外部命令，也不是可运行的程序或批处理文件。” 的错误。

**程序功能**

Pinot 具备以下功能：

* 批量将原始观测数据（m00、t02）转化为 RINEX 格式；
* 批量将数据在 RINEX 与 Compact RINEX 之间转化；
* 批量检查站点的数据缺失情况；
* 批量对站点观测数据进行重命名；
* 批量检查站点观测信息的正确性；
* 批量对 RINEX 观测数据做观测质量分析；
* 批量进行 RINEX 观测数据文件的数据标准化；
* 批量将 RINEX 文件整理为 IGS 的组织方式；
* 批量进行用于数据解算前的子网划分。

**索引目录**

该软件包所包含的脚本及功能文档链接如下表：

| **脚本** | **功能及文档链接** |
| --- | --- |
| crnx2rnx.py | [将 Compact RINEX 转化为 RINEX](http://www.gnss.help/2017/03/27/pinot-crx-rnx-switch/) |
| leica2rnx.py | [转化徕卡接收机原始数据](http://www.gnss.help/2017/02/09/pinot-convert-leica-trimble/) |
| low2upper.py | [文件重命名工具](http://www.gnss.help/2017/01/31/pinot-rename/) |
| metacheck.py | [文件头信息检查](http://www.gnss.help/2017/02/06/pinot-site-check/) |
| orderfiles.py | [RINRX 文件整理脚本](http://www.gnss.help/2017/02/12/pinot-order-file/) |
| qualitycheck.py | [观测质量分析脚本](http://www.gnss.help/2017/02/03/pinot-qualitycheck/) |
| renamesite.py | [文件重命名工具](http://www.gnss.help/2017/01/31/pinot-rename/) |
| rnx2crnx.py | [将 RINEX 转化为 Compact RINEX](http://www.gnss.help/2017/03/27/pinot-crx-rnx-switch/) |
| sitecheck.py | [站点数据缺失检查](http://www.gnss.help/2017/02/06/pinot-site-check/) |
| subnet.py | [子网划分工具](http://www.gnss.help/2017/02/12/pinot-order-file/) |
| trimble2rnx.py | [转化天宝接收机原始数据](http://www.gnss.help/2017/02/09/pinot-convert-leica-trimble/) |
| unificate.py | [GNSS 数据标准化](http://www.gnss.help/2017/01/27/pinot-unificate/) |
| up2lower.py | [文件重命名工具](http://www.gnss.help/2017/01/31/pinot-rename/) |

**作者信息**

开发者: Jon Jiang & Maosheng Zhou

首次上传时间: 2017-01-20

**修改记录**

* 2017-09-30，删除 copyresult.py 脚本。

**GNSS 数据批量整理脚本**

[2017-02-12](http://www.gnss.help/2017/02/12/pinot-order-file/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PINOT](http://www.gnss.help/tags/Pinot/), [PYTHON](http://www.gnss.help/tags/Python/), [文件整理](http://www.gnss.help/tags/%E6%96%87%E4%BB%B6%E6%95%B4%E7%90%86/)

组织保存大量的 GNSS 数据文件时，IGS 的数据存放方式是一个不错的借鉴。但数据的分类整理亦是非常枯燥繁琐的工作。并且，限于目前的 GAMIT/GLOBK 程序所能处理的数据在 99 个站点以内，因此在处理大量的数据时常常需要对观测网进行子网划分操作。这也同样需要按照子网划分方案对观测数据进行整理。

本文将介绍 [Pinot](http://www.gnss.help/tags/Pinot/) 中所涉及的两个 GNSS 数据整理脚本：orderfile.py 和 subnet.py。

**脚本功能**

本文所涉及的两个脚本及其功能如下：

* orderfile.py：以 IGS 的方式整理 GNSS 观测文件；
* subnet.py：以配置文件的指定方式整理文件。

**参数说明**

**orderfile.py**

该脚本的使用方式为：

|  |  |
| --- | --- |
| 1 | $ python orderfile.py <file> [<file> ...] -out <directory> [-k] [-r] [-v] [-h] |

参数释义：

* <file>：要处理的文件名，可由通配符指定；
* -out <directory>：文件的输出路径，若该文件夹不存在将自动创建。该项默认为当前路径的 daily 文件夹；
* -k、--keep：整理文件时是否保留源文件。即以拷贝的方式整理还是以剪切的方式整理，默认为剪切方式；
* -r、--recursive：指定用 “\*\*/” 的通配符前缀表示递归搜索子文件夹的内容，不加该参数默认为不递归；
* -v、--version：显示版本信息；
* -h、--help：显示帮助。

**subnet.py**

subnet.py 脚本可以按照配置对数据进行整理，这主要用于数据的子网划分操作。该脚本可接受的参数为：

|  |  |
| --- | --- |
| 1 | $ python subnet.py <file> [<file> ...] -out <directory> -cfg <config> [-r] [-v] [-h] |

参数释义：

* <file>：要处理的文件名，可由通配符指定；
* -out <directory>：子网划分结果的输出路径，默认为当前路径下的 subnets 文件夹；
* -cfg <config>：子网划分配置文件路径。该配置文件使用 YAML 语言编写，默认为当前目录中的 \_subnet.yml 文件；
* -r、--recursive：指定用 “\*\*/” 的通配符前缀表示递归搜索子文件夹的内容，不加该参数默认为不递归；
* -v、--version：显示版本信息；
* -h、--help：显示帮助。

**配置文件**

只有 subnet.py 脚本需要配置文件，该配置文件使用 YAML 格式编写。

**配置文件示例**

配置文件中的站点名，需使用小写字母。下面是一个配置文件示例：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | net1:  - bjfs  - chan  - daej  net2:  - bjfs  - wuhn  - kunm  net3:  - bjfs  - urum  - lhaz |

**使用示例**

以剪切的方式整理 data/ 目录内的所有 RINEX 文件并递归搜索其子文件夹的内容，将处理成果输出到当前目录的 daily 文件夹：

|  |  |
| --- | --- |
| 1 | $ python orderfile.py data\\*\*\\* -out daily -r |

以拷贝的方式整理 2016、2017 两个文件夹并递归搜索其子文件夹的内容，将处理成果输出到当前目录的 daily 文件夹：

|  |  |
| --- | --- |
| 1 | $ python orderfile.py 201[67]\\*\*\\* -out daily -r -k |

处理 rinex/ 文件夹中观测年为 2016 年的观测文件，其中配置文件使用当前目录中的 \_subnet.yml，将最终的划分成果输入到当前路径的 nets 文件夹：

|  |  |
| --- | --- |
| 1 | $ python subnet.py rinex\\*.16[odOD] rinex\\*.crx rinex\\*.rnx -out nets -cfg \_subnet.yml |

处理 rinex/ 文件夹及其子文件夹中观测年为 2016 年，年积日为 042 至 045 的 RINEX O-文件，其中配置文件使用当年目录中的 \_subnet.yml，将最终的划分成果输入到当前路径的 nets 文件夹：

|  |  |
| --- | --- |
| 1 | $ python subnet.py rinex\\*\*\\*04[2-5]0.16[oO] -out nets -cfg \_subnet.yml -r |

**下载地址**

该脚本及同一工具包的其他脚本均放在 Github 网站上：[purpleskyfall/pinot](https://github.com/purpleskyfall/pinot)。

**修改记录**

* 2017-09-17，orderfile.py 0.2.0，subnet.py 0.4.0：
  + 去掉 -dir 和 -glob 参数，使用位置参数指定输入文件；
  + 支持多个输入文件或通配符；
  + 支持 RINEX 3 格式的文件。
* 2017-09-17，orderfile.py 0.2.1，subnet.py 0.4.1：
  + 优化在 UNIX/Linux 中的输出信息显示。

**接收机原始数据批量转化脚本**

[2017-02-09](http://www.gnss.help/2017/02/09/pinot-convert-leica-trimble/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PINOT](http://www.gnss.help/tags/Pinot/), [PYTHON](http://www.gnss.help/tags/Python/), [RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/)

将 GNSS 接收机中导出的原始观测数据转化为 RINEX 格式也是数据预处理的一部分。我们通常采用接收机厂商提供的随机软件或 TEQC 程序完成这部分工作。但我在使用随机软件进行数据转化时，发现其操作略有不便。因此在 [Pinot](http://www.gnss.help/tags/Pinot/) 中实现了两个原始观测数据转化脚本：leica2rnx.py 和 trimble2rnx.py。

本文是脚本 leica2rnx.py 和 trimble2rnx.py 的说明文档。

**脚本功能**

本文所涉及的两个脚本及其功能为：

* leica2rnx.py：批量将徕卡 m00 文件转化为 RINEX 2.11；
* trimble2rnx.py：批量将天宝 T02 文件转化为 RINEX 2.11。

**运行环境**

两个脚本的数据转化都依赖于 TEQC 程序。除此之外，trimble2rnx.py 脚本还依赖于前文提到的 [runpkr00 程序](http://www.gnss.help/2017/01/19/runpkr00-config-usage/)。leica2rnx.py 脚本使用 TEQC 将 m00 文件转化为 RINEX 文件；而 trimble2rnx.py 脚本首先使用 runpkr00 程序将 T02 文件转化为天宝 DAT 文件，再将 DAT 文件转化为 RINEX。

需要注意的是，因为 trimble2rnx.py 脚本使用了 runpkr00 程序，因此转化出的观测文件中不可避免地将丢失除 GPS 卫星以外的其他卫星观测量。

要完整使用这两个脚本，需保证 TEQC 和 runpkr00 程序在系统搜索目录或当前脚本的执行目录下。

**参数说明**

**leica2rnx.py**

leica2rnx.py 将徕卡 m00 文件转化为 RINEX 格式。该脚本可接受的参数有：

|  |  |
| --- | --- |
| 1 | $ python leica2rnx.py -dir <input\_dir> -glob <mode> -out <output\_dir> -yr <year> [-r] [-v] [-h] |

释义：

* -dir <input\_dir> 参数代表输入文件夹。该项默认为程序运行的当前路径；
* -glob <mode> 参数代表要处理的文件类型，可以由通配符指定；该项默认为 \*.[Mm]00；
* -out <input\_dir> 参数代表信息输出的文件夹。该项默认为当前目录的 rinex 文件夹；
* -yr <year> 参数代表数据的年份，此项不可空缺，否则报错；
* -r、--recursive 参数指定在搜索文件时是否搜索子文件夹的内容，不加该参数时默认为不搜索；
* -v、--version 参数显示版本信息；
* -h、--help 参数用于显示帮助。

**trimble2rnx.py**

trimble2rnx.py 将天宝 T02 文件转化为 RINEX 格式。该脚本可接受的参数有：

|  |  |
| --- | --- |
| 1 | $ python trimble2rnx.py -dir <input\_dir> -glob <mode> -out <output\_dir> -yr <year> [-r] [-v] [-h] |

释义：

* -dir <input\_dir> 参数代表输入文件夹。该项默认为程序运行的当前路径；
* -glob <mode> 参数代表要处理的文件类型，可以由通配符指定；该项默认为 \*.[Tt]02；
* -out <input\_dir> 参数代表信息输出的文件夹。该项默认为程序运行的当前目录的 rinex 文件夹；
* -yr <year> 参数代表数据的年份，此项不可空缺，否则报错；
* -r、--recursive 参数指定在搜索文件时是否搜索子文件夹的内容，不加该参数时默认为不搜索；
* -v、--version 参数显示版本信息；
* -h、--help 参数用于显示帮助。

**使用示例**

处理 raw/ 文件夹下的 2016 年，年积日为 001 的所有 m00 文件，将处理成果输出到当前目录下的 rinex 目录：

|  |  |
| --- | --- |
| 1 | $ python leica2rnx.py -dir raw -glob \*001?.m00 -out rinex -yr 2016 |

处理 raw/ 文件夹及其子文件夹中观测年为 2016 年测站名为 bosh 的 m00 文件，将最终的处理成果输出到当前路径下的 rinex 文件夹中：

|  |  |
| --- | --- |
| 1 | $ python leica2rnx.py -dir raw -glob bosh\*.m00 -out rinex -yr 16 -r |

处理 raw/ 文件夹下的 2016 年，年积日为 001 的所有 T02 文件，将处理成果输出到当前目录下的 rinex 目录：

|  |  |
| --- | --- |
| 1 | $ python trimble2rnx.py -dir raw -glob \*001?.T02 -out rinex -yr 16 |

处理 raw/ 文件夹及其子文件夹中观测年为 2016 年测站名为 bosh 的 T02 文件，将最终的处理成果输出到当前路径下的 rinex 文件夹中：

|  |  |
| --- | --- |
| 1 | $ python trimble2rnx.py -dir raw -glob bosh\*.T02 -out rinex -yr 2016 -r |

**下载地址**

该脚本及同一工具包的其他脚本均放在 Github 网站上：[purpleskyfall/pinot](https://github.com/purpleskyfall/pinot)。

# GNSS 数据信息批量检查脚本

[2017-02-06](http://www.gnss.help/2017/02/06/pinot-site-check/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PINOT](http://www.gnss.help/tags/Pinot/), [PYTHON](http://www.gnss.help/tags/Python/)

在进行大量的 GNSS 数据处理之前（尤其不是自己观测的数据），我们通常首先需要检查各站点的数据完备性和核对观测信息的正确性等。从而得知缺失数据的站点，并及时掌握观测条件（如所使用的接收机类型、天线类型、先验坐标、天线高等）的变更情况等。

本文介绍 [Pinot](http://www.gnss.help/tags/Pinot/) 中相关的两个脚本：sitecheck.py 与 metacheck.py。

## 脚本功能

本文所涉及的三个脚本及其功能如下：

* sitecheck.py：检查站点的数据缺失状况；
* metacheck.py：检查站点信息的变更状况。

## 参数说明

### sitecheck.py

脚本 sitecheck.py 用于检查站点的观测数据（RINEX O-文件或 D-文件）的缺失状况，其中待检查的站点列表需要通过一个配置文件指定。该脚本可接受以下参数：

|  |  |
| --- | --- |
| 1 | $ python sitecheck.py <directory> [<directory> ...] -yr <year> -doy <doy> -cfg <config.yml> [-r] [-v] [-h] |

参数释义：

* <directory>：输入文件夹的路径，该项默认为程序运行的当前路径；
* -yr <year>：数据观测日期的年，该项是必要的；
* -doy <doy>：数据观测日期的年积日，该项是必要的；
* -cfg <config.yml>：包含站点名的配置文件路径。该配置文件使用 YAML 语言编写，默认为当前目录中的 \_sites.yml 文件；
* -r、--recursive：搜索文件时是否搜索子文件夹的内容，不加该参数时默认为不搜索；
* -v、--version：显示版本信息；
* -h、--help：显示帮助。

### metacheck.py

脚本 metacheck.py 用于检查站点观测信息的变更或缺失状况，如接收机类型、天线类型、先验坐标等。该脚本可接受的参数如下：

|  |  |
| --- | --- |
| 1 | $ python metacheck.py <file> [<file> ...] -cfg <config.yml> -thd <threshold> -out <type> [-r] [-v] [-h] |

参数释义：

* <file>：要处理的文件名，可由通配符指定；
* -cfg <config.yml>：包含站点信息的配置文件路径。该配置文件使用 YAML 语言编写，默认为当前目录中的 \_sitesinfo.yml 文件；
* -thd <threshold>：先验坐标允许变化的阈值，默认值为10m；
* -out <type>：信息输出的方式。该参数有两个选项：列表（默认）或表格。列表输出方式以 list 或 l 指定，表格方式以 table 或 t 指定；
* -r、--recursive：指定用 “\*\*/” 的通配符前缀表示递归搜索子文件夹的内容，不加该参数默认为不递归；
* -v、--version：显示版本信息；
* -h、--help：显示帮助。

## 输出示例

### sitecheck.py

脚本 sitecheck.py 在检查完成后，将输出缺失数据的站点名：

Start processing: ../rinex2, ../rinex3 …  
Observation files not found at 2017, 42 for: anqi, wuch, wuhn, zqrs

### metacheck.py

前文已经提到，metacheck.py 脚本的输出信息有两种形式：列表和表格，采用 -out 参数指定。该参数为 l、list 或未指定时为列表；为 t 或 table 时为表格形式。

#### 列表模式

列表模式将不同的项目上下对齐，突出不同点，适合在检查观测信息变化时使用。一个列表模式的输出样例如下：

15d\chan0270.15d has differences:  
delta in cfg file: 0.0000　0.0000　0.0000  
delta in obs file: 0.2500　0.0000　0.0000

16o\bjfs0010.16o has differences:  
receiver in cfg file: TRIMBLE　NETR8  
receiver in obs file: TRIMBLE　NETR9

#### 表格模式

表格形式的输出信息便于使用 Excel 等程序做进一步处理。一个表格形式的输出消息如下：

|  |  |
| --- | --- |
| 1  2  3 | file type in cfgfile in obsfile  bjfs2220.17d receiver TRIMBLE NETR8 TRIMBLE NETR9  chan2220.17d delta 0.0000 0.0000 0.0000 0.2500 0.0000 0.0000 |

## 配置文件

### sitecheck.py

sitecheck.py 脚本的配置文件非常简单，只需以数组的形式列出要进行检查的站点列表。一个配置文件示例为：

|  |  |
| --- | --- |
| 1  2  3  4  5 | - bjfs  - chan  - shao  - suwn  - wuhn |

### metacheck.py

metacheck.py 的配置文件中保存各站点的观测信息，配置文件中的站点名，需使用小写字母。该脚本亦可使用 unificate.py 脚本的配置文件。下面是一个配置文件示例：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | bjfs:  receiver: "TRIMBLE NETR8"  antenna: "TRM59900.00 SCIS"  position: "-2148744.8407 4426642.9605 4044657.8518"  delta: "0.0465 0.0000 0.0000"  chan:  receiver: "ASHTECH UZ-12"  antenna: "ASH701945E\_M NONE"  position: "-2674431.9143 3757145.2969 4391528.8732"  delta: "0.0000 0.0000 0.0000" |

## 使用示例

检查 rinex/ 文件夹及其下的子文件夹，其中配置文件为当年目录中的 \_sites.yml，观测日期为 2016 年第 36 年积日：

|  |  |
| --- | --- |
| 1 | $ python sitecheck.py rinex/ -yr 2016 -doy 36 -cfg \_sites.yml -r |

处理 rinex/ 文件夹中观测年为 2016 年的观测文件，其中配置文件为当前目录中的 \_sitesinfo.yml，将最终的检查结果以表格方式输出到屏幕：

|  |  |
| --- | --- |
| 1 | $ python metacheck.py rinex/\*.16[odOD] -cfg \_sitesinfo.yml -out t |

处理 rinex/ 文件夹及其子文件夹中观测年为 2016 年，年积日为 042 至 045 的 RINEX O-文件，并将先验坐标允许变化的阈值设置为15m，其中配置文件为当前目录中的 \_sitesinfo.yml，将最终的检查成果以列表方式输出到当前路径下的 result.txt 文件中：

|  |  |
| --- | --- |
| 1 | $ python metacheck.py rinex/\*\*/\*04[2-5]0.16[oO] -thd 15 -cfg \_sitesinfo.yml -r > result.txt |

## 下载地址

该脚本及同一工具包的其他脚本均放在 Github 网站上：[purpleskyfall/pinot](https://github.com/purpleskyfall/pinot)。

## 修改记录

* 2017-09-19，sitecheck.py 版本 0.2.0，metacheck.py 版本 0.4.0：
  + 去掉 -dir 和 -glob 参数，使用位置参数指定输入文件；
  + 支持多个输入文件或通配符；
  + 支持 RINEX 3 格式的文件。
* 2018-03-14，sitecheck.py 版本 0.2.1，metacheck.py 版本 0.4.1：
  + 优化在 UNIX/Linux 中的输出信息显示。

赏

**GNSS 数据批量质量分析脚本**

[2017-02-03](http://www.gnss.help/2017/02/03/pinot-qualitycheck/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PINOT](http://www.gnss.help/tags/Pinot/), [PYTHON](http://www.gnss.help/tags/Python/), [RINEX质量分析](http://www.gnss.help/tags/RINEX%E8%B4%A8%E9%87%8F%E5%88%86%E6%9E%90/)

在对 GNSS 观测数据进行处理之前，我们一般都需要进行观测质量分析。其中 TEQC、GFZRNX 等都是常用的质量检查工具。但这些程序处理大量的数据时皆显得略有不便。并且，很多时候我们只需记录一些最关键的质量分析指标。

qualitycheck.py 脚本是一个批量进行 RINEX 数据质量分析的脚本，通过调用 TEQC 程序对输入数据处理，输出观测时长、信噪比、多路径效应、周跳等质量检查成果。

**运行环境**

由于本程序的质量分析操作依赖于 TEQC 程序，但是因不同版本的 TEQC 的输出信息格式有略微不同，本脚本保证在使用 2016Nov7 及以后版本的 TEQC 时测试通过。

需注意的是：对于 Windows 10、Windows 7 等操作系统，只需将 TEQC 程序放入系统能搜索到的路径下，即在任意目录中启动命令提示符执行 teqc 命令能显示 TEQC 程序的提示信息；但对于 Windows XP 或 Windows server 2008 等操作系统，需保证运行脚本的文件夹内有 TEQC 程序，否则可能出现 “’teqc’ 不是内部或外部命令，也不是可运行的程序或批处理文件。” 的错误。

**参数说明**

该脚本可接受的参数有：

|  |  |
| --- | --- |
| 1 | $ python qualitycheck.py <file> [<file> ...] -out <format> [-r] [-v] [-h] |

参数释义：

* <file>：要处理的文件名，可由通配符指定；
* -out <format>：信息输出的方式，该参数有两个选项：列表或表格（默认）。列表输出方式以 list 或 l 指定，表格输出方式以 table 或 t 指定；
* -r、--recursive：指定用 “\*\*/” 的通配符前缀表示递归搜索子文件夹的内容，不加该参数默认为不递归；
* -v、--version：显示版本信息；
* -h、--help：显示帮助。

**输出示例**

前文已经提到，该脚本的输出信息有两种形式：列表和表格，采用 -out 参数指定。该参数为 l、list 或未指定时为列表；为 t 或 table 时为表格形式。

**列表形式**

一个列表形式的输出消息如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | test\2016\001\16o\bjfs0010.16o quality marks:  date: 2016-01-01  start: 00:00:00.000  end: 23:59:30.000  hours: 23.99  SN1: 5.22  SN2: 7.21  MP1: 0.42  MP2: 0.31  CSR: 0.20 |

**表格形式**

表格形式的输出信息方便使用 Excel 或 Pandas 等程序做进一步处理。一个表格形式的输出消息如下：

|  |  |
| --- | --- |
| 1  2  3 | file date start end hours SN1 SN2 MP1 MP2 CSR  bjfs0010.16o 2016-01-01 00:00:00.000 23:59:30.000 23.99 5.22 7.21 0.42 0.31 0.20  bjfs0200.16o 2016-01-20 00:00:00.000 23:59:30.000 23.99 5.42 7.25 0.41 0.35 0.21 |

**使用示例**

处理 rinex/ 文件夹中观测年为 2016 年的观测文件，将最终成果以表格形式输出到屏幕：

|  |  |
| --- | --- |
| 1 | $ python qualitycheck.py rinex/\*.16[oO] -out table |

处理 rinex/ 内 001、009 两个子文件夹内的观测文件，将最终成果以表格形式输出到当前路径下的 result.txt 文件：

|  |  |
| --- | --- |
| 1 | $ python qualitycheck.py rinex/00[19]/\*[oO] -out t > result.txt |

处理 rinex/ 文件夹及其子文件夹中观测年为 2016 年，年积日为 042 至 045 的 RINEX O-文件，将最终成果以列表形式输出到当前路径下的 result.txt 文件：

|  |  |
| --- | --- |
| 1 | $ python qualitycheck.py rinex/\*\*/\*04[2-5]0.16[oO] -out list -r > result.txt |

**下载地址**

该脚本及同一工具包的其他脚本均放在 Github 网站上：[purpleskyfall/pinot](https://github.com/purpleskyfall/pinot)。

**修改记录**

* 2017-09-22，版本 0.4.0：
  + 去掉 -dir 和 -glob 参数，使用位置参数指定输入文件；
  + 支持多个输入文件或通配符；
  + 添加多线程支持；
  + 将默认输出形式更改为表格。
* 2018-03-14，版本 0.4.3：
  + 优化在 UNIX/Linux 中的输出信息显示。
* 2018-03-14，版本 0.4.4：
  + 将输出信息中的时间窗长度替换为数据累计观测长度（即 length 被替换为 hrs），以便分析实际的数据可用率。
* 2018-05-17，版本 0.4.5：
  + 修复处理质量差数据时出现的 Bug，提升稳定性。

**站点数据批量重命名脚本**

[2017-01-31](http://www.gnss.help/2017/01/31/pinot-rename/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PINOT](http://www.gnss.help/tags/Pinot/), [PYTHON](http://www.gnss.help/tags/Python/), [RINEX重命名](http://www.gnss.help/tags/RINEX%E9%87%8D%E5%91%BD%E5%90%8D/), [大小写转换](http://www.gnss.help/tags/%E5%A4%A7%E5%B0%8F%E5%86%99%E8%BD%AC%E6%8D%A2/)

在处理 GNSS 观测数据时，我们经常需要涉及到对数据文件的重命名处理。比如，变更观测站点名、文件名大小写转换等。虽然之前的博文[文件名大小写批量转换](http://www.gnss.help/2016/08/31/Letter-Case-Switch/)介绍过两个 SHELL 语言编写的脚本，但这两个脚本不甚灵活且运行时需要 C SHELL 语言环境。

本文将介绍 [Pinot](http://www.gnss.help/tags/Pinot/) 中涉及对观测文件重命名的三个脚本：renamesite.py、up2lower.py 和 low2upper.py。

**脚本功能**

本文所涉及的三个脚本及其功能如下：

* renamesite.py：变更文件名中的站点名；
* up2lower.py：将文件名中包含的大写字母转换为小写；
* low2upper.py：将文件名中包含的小写字母转换为大写。

**参数说明**

**renamesite.py**

renamesite.py 脚本可以将文件名中的站点（即观测文件的前 4 个字符）进行重命名。其中原站名与新站名的对应关系通过 YAML 格式的配置文件指定。该脚本可接受以下参数：

|  |  |
| --- | --- |
| 1 | $ python renamesite.py <file> [<file> ...] -out <directory> -cfg <config> [-k] [-r] [-v] [-h] |

参数释义：

* <file>：要处理的文件名，可由通配符指定；
* -out <directory>：处理结果的输出路径，默认为源文件所在的文件夹；
* -cfg <config>：配置文件路径。该配置文件使用 YAML 语言编写，默认为当前目录中的 \_sitemap.yml 文件；
* -r、--recursive：指定 “\*\*/” 的通配符前缀表示递归搜索子文件夹的内容，不加该参数默认为不递归；
* -k、--keep：对文件进行重命名时是否保留源文件，即以拷贝方式重命名还是以剪切的方式；
* -v、--version：显示版本信息；
* -h、--help：显示帮助信息。

**up2lower.py**

对文件名中的字符进行大小写转换的两个脚本：up2lower.py 和 low2upper.py，此两个脚本的所接受的参数一致。下面以 up2lower.py 为例。该脚本接受的参数如下：

|  |  |
| --- | --- |
| 1 | $ python up2lower.py <file> [<file> ...] -out <directory> [-r] [-v] [-h] |

参数释义：

* <file>：要处理的文件名，可由通配符指定；
* -out <directory>：输出文件路径。若该项目未指定，则直接对输入路径内的文件重命名；若指定了输出路径，则拷贝文件并重命名；
* -r、--recursive：指定 “\*\*/” 的通配符前缀表示递归搜索子文件夹的内容，不加该参数默认为不递归；
* -k、--keep：处理文件时是否保留源文件，即以拷贝方式还是剪切方式，默认为剪切操作；
* -v、--version：显示版本信息；
* -h、--help：显示帮助信息。

**配置文件**

只有 renamesite.py 脚本需要配置文件，其格式非常简单，只需以键值对的方式指定原站名与新站名。其形式为：old: new。

**配置文件示例**

一个配置文件示例为：

|  |  |
| --- | --- |
| 1  2  3 | bjfs: beij  chan: chch  wuhn: wuh2 |

在该配置文件中，配置将 bjfs 站的观测数据更名为 beij，将 chan 站的观测数据更名为 chch，将 wuhn 站的观测数据更名为 wuha。

**使用示例**

更新 old/ 文件夹及其子文件夹内观测于 2017 年的数据的站名，其中配置文件使用当前文件夹下的 \_sitemap.yml，将重命名后的观测文件保存在原文件夹内：

|  |  |
| --- | --- |
| 1 | $ python renamesite.py old/\*\*/\*.17[od] -cfg \_sitemap.yml -r |

更新 old/ 文件夹内所有 RINEX 3 格式的观测数据的站名，其中配置文件使用当前文件夹下的 \_sitemap.yml，将重命名后的观测文件复制到 new/ 文件夹内：

|  |  |
| --- | --- |
| 1 | $ python renamesite.py old/\*.rnx -out new -cfg \_sitemap.yml -k |

将 FILES/ 文件夹及其子文件夹内的所有观测文件中的大写字符转换为小写，将重命名后的文件保留在源文件夹内：

|  |  |
| --- | --- |
| 1 | $ python up2lower.py FILES/\*\*/\*O FILES/\*\*/\*.rnx -r |

将 FILES/ 文件夹及其子文件夹内的所有观测文件中的大写字符转换为小写，将重命名后的文件复制到 new/ 文件夹内：

|  |  |
| --- | --- |
| 1 | $ python up2lower.py FILES/\*\*/\*O FILES/\*\*/\*.rnx -out new -r -k |

将 FILES/ 文件夹内 2016 年第 42 至 47 年积日的观测数据文件名中的大写字符转换为小写，将重命名后的文件复制到 new/ 文件夹内：

|  |  |
| --- | --- |
| 1 | $ python up2lower.py FILES/\*04[2-7]0.16O -out new -r -k |

**下载地址**

该脚本及同一工具包的其他脚本均放在 Github 网站上：[purpleskyfall/pinot](https://github.com/purpleskyfall/pinot)。

**修改记录**

* 2017-09-13，版本 0.2.0：
  + 去掉 -dir 和 -glob 参数，使用位置参数指定输入文件；
  + 支持多个输入文件或通配符；
  + 支持 RINEX 3 格式的文件。
* 2018-03-14，版本 0.2.1：
  + 优化在 UNIX/Linux 中的输出信息显示。

**GNSS 数据批量标准化脚本**

[2017-01-27](http://www.gnss.help/2017/01/27/pinot-unificate/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PINOT](http://www.gnss.help/tags/Pinot/), [PYTHON](http://www.gnss.help/tags/Python/), [RINEX编辑](http://www.gnss.help/tags/RINEX%E7%BC%96%E8%BE%91/)

在处理大量的 GNSS 数据时，我们常常需要编辑大量的 RINEX 观测文件（Obs）。有些测站由于观测人员的设置等原因，导致观测数据的文件头信息不标准或不正确。也有些观测数据的采样间隔、观测卫星系统或观测量等不一致。这就需要对数据进行标准化工作。

TEQC 或 GFZRNX 程序可以做到这一点，但这些程序比较适合处理单个或单个测站的观测文件，对于处理大量的数据时显得略不便。本文将介绍 [Pinot](http://www.gnss.help/tags/Pinot/) 中实现的批量标准化程序 unificate.py。该脚本以多线程的方式执行，运行快速。

**运行环境**

本脚本使用 Python 3 语言编写，依赖于 TEQC 程序和 PyYAML 模块。要使用此脚本，需保证系统已配置 TEQC 程序和 PyYAML 程序。TEQC 程序配置请参考本站 [TEQC 程序配置及其数据转换功能的使用](http://www.gnss.help/2016/09/17/teqc-config-translate/)，PyYAML 配置请参考本站 [PyYAML 安装与使用演示](http://www.gnss.help/2016/12/01/install-pyyaml/)。

脚本在 Windows XP、Windows Server 2008 R2 以及 Windows 10 等操作系统测试通过。由于程序的文件编辑操作依赖于 TEQC 程序，需注意的是：对于 Windows 10 操作系统，只需将 TEQC 程序放入系统能搜索到的路径下，即在任意目录中启动命令提示符执行 teqc 命令能显示 TEQC 程序的提示信息；但对于更低版本的 Windows 系统，若出现 “’teqc’ 不是内部或外部命令，也不是可运行的程序或批处理文件。” 的错误，此时需保证运行脚本的文件夹内有 TEQC 程序。

此外，最新的程序实现了进度条显示功能。该功能使用了 [tqdm](https://pypi.python.org/pypi/tqdm) 模块，要安装该模块，在 “命令提示符” 窗口执行：

|  |  |
| --- | --- |
| 1 | $ pip install tqdm |

**参数说明**

该脚本可接受以下参数：

|  |  |
| --- | --- |
| 1 | $ python unificate.py <file> [<file> ...] -out <directory> -cfg <config> [-r] [-v] [-h] |

参数释义：

* <file>输入的待处理文件，你可以使用文件名，也可以使用通配符 ；
* -out <directory> 处理结果的输出路径，默认为当前路径下的 unificated 文件夹；
* -cfg <config> 配置文件路径，该文件使用 YAML 语言编写，默认为当前目录下的 \_sitesinfo.yml 文件；
* -k、--keep 标准化成功后是否保留源文件，不加该参数时默认为不保留；
* -r、--recursive 指定 “\*\*/” 的通配符前缀表示递归搜索子文件夹的内容，不加该参数默认为不递归；
* -v、--version 显示版本信息；
* -h、--help 显示帮助。

**配置文件**

配置文件使用 [YAML](https://zh.wikipedia.org/wiki/YAML) 语言编写。目前支持的编辑项目有：

* agency: 观测机构；
* antenna: 天线类型；
* delta: 天线相位中心偏移；
* interval: 数据采样间隔；
* obs\_type: 保留的观测类型；
* observer: 观测者；
* position: 站点概略坐标；
* receiver: 接收机类型；
* rm\_sys: 要删掉的卫星系统。

这些配置项既可以写在各站点的配置内，也可以放在 “all” 项目中作为默认配置项目。若 “all” 和站点内都设置了某一项信息，则使用站点中的配置。

**配置文件示例**

配置文件中的站点名，需使用小写字母。下面是一个配置文件示例：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | all:  observer: sdust  agency: sdust  rm\_sys:  - C  - R  obs\_type: ['C1', 'C2', 'P1', 'P2', 'L1', 'L2']  interval: 30  delta: "0.0000 0.0000 0.0000"  bjfs:  receiver: "TRIMBLE NETR8"  antenna: "TRM59900.00 SCIS"  position: "-2148744.8407 4426642.9605 4044657.8518"  delta: "0.0465 0.0000 0.0000"  chan:  receiver: "ASHTECH UZ-12"  antenna: "ASH701945E\_M NONE"  position: "-2674431.9143 3757145.2969 4391528.8732" |

在该配置文件中，通过 “all” 配置了全局的观测者、观测机构、删掉的卫星系统、保留的观测类型、采样间隔和天线中心偏移量。

在 “bjfs” 中，配置了该站点使用的接收机类型、天线类型、概略坐标位置和天线中心偏移。在 “chan” 中，配置了该站点的接收机类型、天线类型和概略坐标。其中未配置的项目，将使用 “all” 中的信息。

**使用示例**

处理 rinex/ 文件夹中所有的 RINEX O-文件，其中使用的配置文件为当前目录中的 \_sitesinfo.yml，将最终的处理成果输入到当前路径的 unificated 文件夹：

|  |  |
| --- | --- |
| 1 | $ python unificate.py rinex/\*[Oo] -out unificated -cfg \_sitesinfo.yml |

处理 rinex/ 文件夹中观测年为 2016，年积日为 042 至 045 的 RINEX O-文件，其中配置文件为当前目录中的 \_sitesinfo.yml，将最终的处理成果输入到当前路径的 unificated 文件夹：

|  |  |
| --- | --- |
| 1 | $ python unificate.py rinex/\*04[2-5]0.16[Oo] -out unificated -cfg \_sitesinfo.yml |

处理 rinex/ 文件夹中 BJFS 站观测于 2016 年的 RINEX O-文件，其中配置文件使用当前目录的 \_sitesinfo.yml，将最终的处理成果输入到当前路径的 unificated 文件夹：

|  |  |
| --- | --- |
| 1 | $ python unificate.py rinex/bjfs\*.16[Oo] -cfg \_sitesinfo.yml -out unificated |

处理 rinex/ 文件夹中的 001，002，003 文件夹中的文件，其中配置文件使用当前目录的 \_sitesinfo.yml，将最终的处理成果保存至 unificated 文件夹：

|  |  |
| --- | --- |
| 1 | $ python unificate.py rinex/00[1-3]/\*[Oo] -cfg \_sitesinfo.yml -out unificated |

**下载地址**

该脚本及同一工具包的其他脚本均放在 Github 网站上：[purpleskyfall/pinot](https://github.com/purpleskyfall/pinot)。

**修改记录**

* 2017-09-10，版本 0.5.0：
  + 去掉 -dir 和 -glob 参数，使用位置参数指定输入文件；
  + 支持多个输入文件或通配符；
  + 添加进度条显示；
  + 基于 concurrent.futures 的多线程支持。
* 2018-03-14，版本 0.5.1：
  + 优化在 UNIX/Linux 中的输出信息显示。

# runpkr00 程序配置及使用

[2017-01-19](http://www.gnss.help/2017/01/19/runpkr00-config-usage/)

[RINEX](http://www.gnss.help/categories/RINEX/)

[RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/), [RUNPKR00](http://www.gnss.help/tags/runpkr00/)

[前文](http://www.gnss.help/2016/09/17/teqc-config-translate/)已经介绍过使用 TEQC 程序将接收机产生的原始观测数据转化为 RINEX 数据的方法。但从天宝接收机中导出的 T00、T01、T02 等数据，无法直接通过 TEQC 程序进行处理。

[上一篇文章](http://www.gnss.help/2017/01/17/install-convert-to-rinex/)已经说过了使用天宝 Convert to RINEX 程序的安装及操作方法，本文将介绍另一个可以完成转化任务的程序：runpkr00。

## 程序简介

[runpkr00](http://kb.unavco.org/kb/article/trimble-runpkr00-v5-40-latest-version-mac-osx-10-7-windows-xp-7-linux-solaris-744.html) 是一个命令行程序，由美国天宝公司（Trimble）开发。该程序仅支持 Windows 操作系统，可以将天宝接收机的原始观测数据转化为 TEQC 程序可处理的天宝 DAT 数据格式。使用 TEQC 进一步处理 DAT 格式的输出文件，即可获得最终的 RINEX 格式数据。

## 环境配置

一如其他的命令行程序，使用 runpkr00 程序的配置亦非常简单。启动 “命令提示符” 窗口，使用 cd 命令进入程序所在文件夹，然后运行 runpkr00 即可。当然，如果你希望在任何目录都能运行该程序，可以考虑将其所在目录添加到系统环境变量，或直接将 runpkr00.exe 移动至 “C:\Windows\System32” 文件夹内。

## 使用说明

runpkr00 程序的操作方式为：

|  |  |
| --- | --- |
| 1 | $ runpkr00 <options> <input\_files> <outdir> |

其中 <input\_files> 代表输入文件列表，<outdir> 代表文件输出路径，而 <options> 代表命令参数信息。程序接受的参数如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | -c # 忽略错误信息  -a # 输出 APP 文件（如果可能的话）  -d # 输出 DAT 文件  -e # 输出 EPH 文件  -i # 输出 ION 文件  -m # 输出 MES 文件  -n[!] # fix NetRS serial number, [!-unconditional]  -s[file] # 输出 Summary 文件  -f # 转化时尝试修复损坏文件  -u[0|1] # 更新文件名 [0-dft, 1-station]  -v # 使用 verbose 模式  -x[ehi\*] # exclude initial: e-eph, h-header, i-ionutc, \*-all  -tfmt # 输入文件类型 (fmt=r00,t00,t01,t02)  -q # 将转化过程中的消息输出到标准输出  -g # use type 27 (if it exists) and allow extended type 17 |

以 DEMO 站于 2016 年第 100 年积日的观测文件为例。下面的命令将 DEMO100aA.T02 转化为 DAT 格式的数据并保存至当前目录：

|  |  |
| --- | --- |
| 1 | $ runpkr00 -d DEMO100aA.T02 |

命令运行完成后，将生成文件：DEMO100aA.dat。

其中的参数还可以结合起来使用。下面的命令将 DEMO100aA.T02 转化为 DAT 格式和 EPH 格式的数据并保存至当前目录的 temp/ 文件夹内：

|  |  |
| --- | --- |
| 1 | $ runpkr00 -de DEMO100bA.T02 temp |

命令运行完成后，将生成两个文件：DEMO100bA.dat 和 DEMO100bA.eph。

得到输出的 DAT 文件后，使用 TEQC 将其转化为 RINEX 格式：

|  |  |
| --- | --- |
| 1 | $ teqc DEMO100aA.dat > demo100a.16o |

当存在连续的多个 DAT 文件时，比如 DEMO100aA.T02、DEMO100bA.T02 ……。还可以在转化的同时进行合并：

|  |  |
| --- | --- |
| 1 | $ teqc DEMO100?A.dat > demo1000.16o |

## 补充

一如之前提到的天宝 [Convert to RINEX](http://www.gnss.help/2017/01/17/install-convert-to-rinex/) 软件，runpkr00 程序也存在一些问题。具体表现为：当某些原始数据文件有问题导致程序无法处理时，会导致程序崩溃。并且，使用该方法得到的 RINEX 文件中将丢失除 GPS 之外的所有其他卫星系统的观测数据。

# Convert to RINEX 程序安装与使用

[2017-01-17](http://www.gnss.help/2017/01/17/install-convert-to-rinex/)

[RINEX](http://www.gnss.help/categories/RINEX/)

[RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/)

[Convert to RINEX](http://www.trimble.com/support_trl.aspx?Nav=Collection-40773&pt=Trimble%20RINEX) 是由美国天宝公司（Trimble）所开发的程序，用于将天宝 GNSS 接收机的原始数据文件（DAT、T00、T01、T02、RT17、RT27 或 .cap ）转化为 RINEX 格式。该程序仅支持 Windows 操作系统，其输出的 RINEX 格式文件支持 2.10、2.11、3.02 等版本。

本文将介绍该程序的安装过程及使用方法。

## 获取所需安装包

要使用该程序，需要首先安装一个依赖程序：[Trimble Configuration Utility](http://www.trimble.com/infrastructure/trimbleconfiguration_ts.aspx)。因此，必须要下载两个程序安装包：[Convert to RINEX](http://www.trimble.com/support_trl.aspx?Nav=Collection-40773&pt=Trimble%20RINEX) 和 [Trimble Configuration Utility](http://www.trimble.com/infrastructure/trimbleconfiguration_ts.aspx)。

## 程序安装

### 安装 Trimble Configuration Utility

下载所需安装包后，首先安装 Trimble Configuration Utility 程序。双击所下载的安装包将弹出安装引导窗口，根据其提示进行安装。

### 安装 Convert to RINEX

完成 Trimble Configuration Utility 的安装之后，再安装 Convert to RINEX 程序。同样双击其安装包，根据弹出的引导提示进行安装。

## 使用说明

Convert to RINEX 程序有两种使用方式：图形界面操作和命令行操作。

### 图形界面操作

使用图形界面操作时，双击该程序的快捷方式或程序安装目录中的可执行文件 convertToRinex.exe 启动程序操作窗口。

点击 “File” 菜单中的 “Open” 选项，选择要进行转化的文件。加载文件完成后程序右侧窗口将显示 RINEX 文件输出路径、接收机天线类型、先验坐标等信息，你可以在此处直接对这些信息进行编辑，并在编辑完成后点击 “File” 菜单中的 “Convert Files” 选项启动数据转化。

### 命令行操作

Convert to RINEX 程序还支持在命令行中进行操作。打开系统的 “命令提示符” 窗口，使用 cd 命令进入程序安装目录，运行 convertToRinex 即可在命令行中启动程序。

该程序的使用方式为：

|  |  |
| --- | --- |
| 1 | $ convertToRinex <file> <options> |

其中 <file> 代表要处理的文件路径，而 <options> 表示输出的参数信息。

使用示例，将 DEMO100aA.T02 转化为 RINEX 2.11 格式，并将结果文件输出到 D 盘的 rinex/ 文件夹：

|  |  |
| --- | --- |
| 1 | $ convertToRinex DEMO100aA.T02 -p D:rinex |

运行上述命令后，将在输出文件夹内看到转化出的 RINEX 观测文件、导航文件等。

若希望得到 RINEX 2.10 或 3.02 版本的输出文件，可以指定版本号：

|  |  |
| --- | --- |
| 1 | $ convertToRinex DEMO100aA.T02 -v 3.02 -p D:rinex |

如上命令将得到 RINEX 3.02 格式的输出文件。

参数信息还可以由一个文件进行输入：

|  |  |
| --- | --- |
| 1 | $ convertToRinex @<ParamFile> # <ParamFile> 代表输入的文件名 |

其中的 <ParamFile> 文件每行以 <key> = <value> 的形式指定各参数的值。对于每个参数对应的键，读者可自行查阅帮助信息，这里不再给出。

例如，输入参数从 ParamFile.txt 文件中导入，可执行：

|  |  |
| --- | --- |
| 1 | $ convertToRinex @ParamFile.txt |

## 完整的参数表

convertToRinex 程序可接受的参数信息如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | -? # 显示帮助信息  -p # 设置输出文件路径  -r # 设置观测者和观测机构  -o # 设置观测者  -ag # 设置观测机构  -v # 设置输出文件的 RINEX 版本号，默认为 2.11  -ac # 设置天线类型  -an # 设置天线号  -h # 设置天线高，以米为单位  -mo # 设置观测标识名  -mn # 设置观测标识号  -rc # 设置接收机类型  -rn # 设置接收机号  -n # 不要设置默认天线高  -k # 动态设定首历元观测时刻  -k- # 禁用所有的动态设定  -ca # Account for millisecond time steps in observations and time of observation  -co # 在输出的 OBS 文件中包含接收机钟漂  -d # 在输出的 OBS 文件中包含多普勒观测量  -s # 在输出的 OBS 文件中包含原始的单频观测量  -g # 在输出的 OBS 文件中只保留 GPS 卫星观测值  -m # 如果可能的话，输出 Met 文件  -t # 如果可能的话，输出 AUX 文件 |

## 补充

本以为使用该程序的命令行操作模式将有利于编制批处理脚本，但在测试过程中发现了一个严重的问题：当某些原始数据文件有问题导致程序无法处理时，程序将一直停在当前执行的命令处，不会停止运行也不会报错。这导致脚本在运行过程中将可能一直卡在有问题的文件上。

赏

**TEQC 观测质量检查功能简介**

[2017-01-06](http://www.gnss.help/2017/01/06/teqc-quality-check/)

[TEQC](http://www.gnss.help/categories/TEQC/)

[RINEX质量分析](http://www.gnss.help/tags/RINEX%E8%B4%A8%E9%87%8F%E5%88%86%E6%9E%90/), [TEQC](http://www.gnss.help/tags/TEQC/)

众所周知，TEQC 程序共有三大功能：数据转换、文件编辑和观测质量检查。前文已经分别介绍过该程序的[环境配置与数据格式转换功能](http://www.gnss.help/2017/01/06/teqc-quality-check/2016/09/17/teqc-config-translate/)、[常用文件编辑功能](http://www.gnss.help/2017/01/06/teqc-quality-check/2016/09/19/teqc-edit/)的使用，本文将完成最后一个任务：介绍 TEQC 程序的数据观测质量检查功能。

TEQC 程序在该质量检查方面的功能强大，但本文只涉及最常用的部分。即质量检查操作和输出文件的结构。

**质量检查命令**

要使用 TEQC 程序的观测质量检查功能，只需在执行命令时添加一个 +qc 参数。但还应该知道的是：TEQC 的质量分析模式分为两种——完整模式（full）和轻量模式（lite）。其区别在于在进行质量检查时是否引入卫星广播星历文件。

要使用完整模式的质量检查，需引入同时段（或相近时段）的卫星星历文件。例如，下面的命令将检查 BJFS 测站在 2016 年 1 月 1 日的观测文件，其中星历文件使用 brdc0010.16n 和 brdc0010.16g：

|  |  |
| --- | --- |
| 1 | $ teqc +qc -nav brdc0010.16n,brdc0010.16g bjfs0010.16o |

如果观测文件所在目录中存在相同测站的星历文件时，-nav 参数可以省略：

|  |  |
| --- | --- |
| 1 | $ teqc +qc bjfs0010.16o |

这时，TEQC 程序将检查当前目录或观测文件所在目录下是否存在与测站同名的导航文件：bjfs0010.16n 或 bjfs0010.16g（当包含 GLONASS 卫星的观测值时）。若可以找到该文件，则采用完整模式进行质量检查，否则采用轻量模式。

**检查结果说明**

**文件类型**

质量检查结束后，程序将在屏幕上打印主要的质量检查结果，并在观测文件所在目录生成质量分析结果文件。其文件名与观测文件名相同，不过拥有不同的后缀：

* .ion：电离层观测值；
* .iod：电离层观测值变率；
* .mp1：L1 频段多路径效应；
* .mp2：L2 频段多路径效应；
* .sn1：L1 频段信噪比；
* .sn2：L2 频段信噪比；
* .azi：卫星方位角；
* .ele：卫星高度角；
* .<yr>S：质量检查摘要，其中 <yr> 表示两位的观测年。

需注意的是，较新版本的 TEQC 程序（2013Nov13 以及更新版本）默认只生成一个后缀为 .<yr>S 的质量检查报告文件，但这对于大多数的应用已经足够了。若需要其它质量输出文件，你可以在执行质量检查时添加一个 +plot 参数。某些情况下，你可能连唯一的质量检查报告文件也不需要，只希望查看一下屏幕上的信息，此时可以在执行质量检查操作时添加一个 -rep 参数。

本文将只详细介绍质量检查报告文件的内容，其他的文件主要用来进行作图。

**摘要文件说明**

TEQC 程序质量检查的两种模式所生成的结果文件不完全相同。相比完整模式，轻量模式的结果文件中只包含部分的检查结果。本文将以完整模式所生成的检查结果文件为例 [注：本节内容主要参考自《GAMIT/GLOBK 软件数据处理手册》, 西安测绘研究所, 2004]。

打开一份质量检查结果摘要文件，首先看到所使用的 TEQC 程序版本，然后就是一副卫星可见性图。该图的横轴为时间，竖轴最后两行分别表示定位（轻量模式下为观测状态）和接收机钟状态，其余为各卫星的状态。该图以字符来表示不同的状态，各字符的含义可参考文末的对照表。

一份卫星可见性图（部分）的示例如下，其中省略部分以 “…” 表示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | SV+---------|--------|--------|--------|--------|--------|--------|--------+ SV  1| \_\_LLLLL\_ \_LooooooooooooooooL\_ | 1  2|oooooooooooo+L \_--ooooooo| 2  3| LLooooooooLLL\_ \_LLooooooooooooL^ | 3  4| \_-----------\_\_ \_\_--------------\_ | 4  ...  R 1| \_\_L\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*++\_ \_\_\_ \_L\*|R 1  R 2| LIIIIIIIIIII\*\*\*\*\*\*\*++\_ \_ |R 2  R 3| LIIIIIIIIIII\*\*\*\*\*++ \_\_LL\_ |R 3  R 4| \_L\*\*\*\*\*\*\*\*\*\*\*\*\*\*+\_ \_\_------\_ |R 4  ...  -dn| ++ ++ 1 22122 ++++ + ++++++ 12222111++++++++++++++++2344433+++11++++|-dn  +dn| 1122332332111 1 1124343233211 1 134455445511111 1|+dn  +10|fdffgfffghghigfggeggghhggffgffhhiigiihhhhffggeefffffeghijjjkiihiiiiiiggf|+10  Pos|ooo oo o oo ooo |Pos  Clk| |Clk  +---------|--------|--------|--------|--------|--------|--------|--------+  00:00:00.000 23:59:30.000  2016 Jan 1 2016 Jan 1 |

之后的第 2 部分为观测数据记录及统计情况。我们用于衡量数据观测质量的多路径效应 MP1、MP2，信噪比 SN1、SN2，接收机钟差参数，观测值与周跳数比值 O/SLPS 等均可在该部分找到。除此之外，还包含开始观测时刻、最后观测时刻、观测时段长、观测历元数、数据完整率和大概的站点坐标等。

该部分的一份样例（部分）如下所示，其中省略部分以 “…” 表示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | 4-character ID : BJFS (# = 21601M001)  Receiver type : TRIMBLE NETR9 (# = 5413K48204) (fw = 4.81)  Antenna type : TRM59900.00 SCIS (# = 17361110)  Time of start of window : 2016 Jan 1 00:00:00.000  Time of end of window : 2016 Jan 1 23:59:30.000  Time line window length : 23.99 hour(s), ticked every 3.0 hour(s)  antenna WGS 84 (xyz) : -2148773.3796 4426646.0890 4044668.7204 (m)  antenna WGS 84 (geo) : N 39 deg 36' 30.93" E 115 deg 53' 33.96"  antenna WGS 84 (geo) : 39.608592 deg 115.892768 deg  WGS 84 height : 108.8053 m  ...  Moving average MP12 : 0.387326 m  Moving average MP21 : 0.343590 m  Points in MP moving avg : 50  Mean S1 : 5.87 (sd=1.58 n=41281)  Mean S2 : 7.82 (sd=1.18 n=40769)  ...  first epoch last epoch hrs dt #expt #have % mp1 mp2 o/slps  SUM 16 1 1 00:00 16 1 1 23:59 24.00 30 43856 40769 93 0.39 0.34 129 |

第 3 部分为参数设置表（由软件缺省设置）。该部分的一份样例如下，其中省略部分以 “…” 表示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | Processing parameters are:  Receiver tracking capability : 44 SVs  Maximum ionospheric rate (L1) : 400.00 cm/min  Report data gap greater than : 10.00 min  Expected rms of MP12 multipath : 65.00 cm  Expected rms of MP21 multipath : 65.00 cm  Multipath slip sigma threshold : 4.00 sigma  % increase in MP rms for C/A | A/S : 100.00 %  Points in MP moving averages : 50  Minimum signal to noise for L1 : 4  Minimum signal to noise for L2 : 4  ...  Observations start : 2016 Jan 1 00:00:00.000  Observations end : 2016 Jan 1 23:59:30.000  Observation interval : 30.0000 second(s) |

第 4 部分为每一颗卫星观测量的统计情况。其中第 1 列为卫星号，第 2 列为高度角在 0 度以上时可观测到的观测量个数，第 3 列为高度角在 0 度以上时的平均高度角，第 4 列为高度角在设定高度角以上时可观测到的观测量个数，第 5 列为高度角在设定高度角以上时的平均高度角，第 6 列为已报告的观测量个数，第 7 列为已报告且每个观测量均完整的观测量个数，其余部分别为各观测类型的实际观测量个数。

该部分的一份样例如下，其中省略部分以 “…” 表示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | SV #+hor <ele> #+mask <ele> #reprt #compl L1 L2 P1 P2 C1 C2  --- ------ ----- ------ ----- ------ ------ ------ ------ ------ ------ ------ ------  G01 1054 29.80 660 44.50 660 660 660 660 0 660 660 0  G02 944 41.52 834 46.34 822 819 822 819 0 819 822 0  G03 1171 23.03 926 27.81 926 908 924 908 0 908 926 0  G05 926 40.80 813 45.78 813 813 813 813 0 813 813 0  ...  R01 1037 30.52 803 38.42 803 803 803 803 803 803 803 0  R02 843 36.11 730 40.96 730 730 730 730 730 730 730 0  R03 927 33.14 644 46.09 644 644 644 644 644 644 644 0  R04 977 29.41 730 37.61 563 563 563 563 563 563 563 0  ...  Obs below mask ( 10.00 deg) : 4602  Obs reported w/ code | phase : 45904  Obs deleted (any reason) : 5135  Obs complete : 40769  ...  No. of Rx clock offsets : 0  Total Rx clock drift : 0.000000 ms  Rate of Rx clock drift : 0.000000 ms/hr |

第 5 部分为电离层延迟观测量统计情况。该部分的左侧为为高度角区间及观测量个数统计情况，右侧为柱壮图形的直观表示。其中柱状图中的 “=” 表示电离层延迟跳变个数占总观测量百分比，“|” 表示电离层延迟均方根差大小（单位为米），“#” 表示 “=” 和 “|” 重叠。

该部分的一份样例如下，其中省略部分以 “…” 表示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | elev (deg) tot slps <ION rms, m> 5=% 1|m 15=% 2|m  85 - 90 160 0 0.000000  80 - 85 532 6 0.000000 ==  ...  60 - 65 2266 9 0.000000  55 - 60 2775 21 0.000000 =  50 - 55 2733 16 0.000000 =  ...  30 - 35 3246 28 0.000000 =  25 - 30 3493 24 0.000000 =  20 - 25 3853 50 0.000000 ==  15 - 20 4044 57 0.000000 ==  10 - 15 3539 40 0.000000 ==  5 - 10 3037 51 0.000000 ===  0 - 5 72 20 0.000000 ==============================================>  < 0 0 0 0.000000 |

第 6 部分为 P1、P2 上的多路径统计。其中第 1 列为卫星号，第 2 列为高度角大于 10 度时的观测量个数，第 3 列为高度角大于 10 度时的没有记录的观测量个数，第 4 列为高度角大于 10 度时的平均高度角，第 5 列为 P1 多路径的 RMS 值，第 6-8 列为高度角在 25 度以下时记录的多路径跳变次数和 L1、L2 上报告的周跳次数，第 9-11 列为高度角在 25 度以上时记录的多路径跳变次数和 L1、L2 上报告的周跳次数。

该部分的一份样例如下，其中省略部分以 “…” 表示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | MP12 RMS summary (per SV):  slips L1 rx L2 rx slips L1 rx L2 rx  SV obs>10 # del <elev> MP12 rms [m] < 25 < 25 < 25 > 25 > 25 > 25  G01 660 0 44.50 0.500379 0 1 1 0 0 0  G02 822 3 47.02 0.320805 0 1 1 0 0 0  G03 926 18 28.13 0.501638 0 11 11 0 0 0  G05 813 0 45.84 0.321881 0 0 0 0 0 0  G06 833 0 45.16 0.326040 0 0 0 0 0 0  G07 803 11 35.56 0.406522 0 2 2 0 0 0  ...  mean MP12 rms : 0.387316 m  total mean elevation : 40.46 degrees  # MP12 obs > 10 : 40769  # qc MP12 slips < 25 : 145  # Rvr L1 slips < 25 : 323  # Rvr L2 slips < 25 : 323  # qc MP12 slips > 25 : 165  # Rvr L1 slips > 25 : 191  # Rvr L2 slips > 25 : 191  ...  elev (deg) tot slps <MP12 rms, m> 5=% 1|m 15=% 2|m  85 - 90 160 0 0.258961 |||||  80 - 85 532 6 0.248227 ##|||  75 - 80 1054 11 0.357657 ##|||||  70 - 75 1813 7 0.266075 |||||  65 - 70 2167 3 0.320790 ||||||  ...  20 - 25 3853 50 0.461703 ##|||||||  15 - 20 4044 55 0.516584 ##||||||||  10 - 15 3539 40 0.630460 ##|||||||||||  5 - 10 3037 62 0.852693 ####|||||||||||||  0 - 5 72 25 0.370823 #######=======================================>  < 0 0 0 0.000000 |

第 7 部分为L1、L2 信噪比统计情况。其左侧为高度角区间、观测量个数、信噪比的 1 倍中误差、信噪比大小，右侧为柱壮图形的直观表示。其中柱状图中的“=”表示信噪比的 1 倍中误差大小，“|”表示信噪比，“#”表示“=”和“|”重叠。

该部分的一份样例如下，其中省略部分以 “…” 表示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | S/N L1 summary (per elevation bin):  elev (deg) tot SN1 sig mean 0|5 1|0  85 - 90 160 0.872 7.194 ###||||||||||||||||||||||||||  80 - 85 532 0.739 6.977 ###|||||||||||||||||||||||||  75 - 80 1054 0.699 7.196 ###||||||||||||||||||||||||||  70 - 75 1813 0.589 7.096 ##||||||||||||||||||||||||||  ...  20 - 25 3854 0.944 4.807 ####|||||||||||||||  15 - 20 4089 1.077 4.222 ####|||||||||||||  10 - 15 4021 1.375 2.948 #####|||||||  5 - 10 4397 0.917 1.759 ####|||  0 - 5 82 1.763 3.598 #######|||||||  < 0 6 3.327 5.333 #############|||||||| |

**观测状态符号对照表**

本节原始的信息可使用以下命令查看：

|  |  |
| --- | --- |
| 1 | $ teqc ++sym |

在卫星与观测状态图中，与卫星有关的符号有：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | C # 钟发生跳变，每一颗卫星在 P1 和 P2 上多路径影响的发生相同的跳变量，且变化量为整毫秒  m # 钟发生跳变，部分观测到的卫星有这种变化或者各颗星的变化量不同，且变化量为整毫秒  I # 相位中的电离层观测值发生跳变  M # P1 和 P2 的多路径影响发生跳变，但变化量不为整毫秒  1 # 只有 P1 的多路径影响发生跳变  2 # 只有 P2 的多路径影响发生跳变  - # 卫星在设定高度角之上，但接收机没有记录数据  + # 卫星在设定高度角之下，但接收机记录了所有码和相位数据  ^ # 卫星在设定高度角之下，接收机只记录了部分码和相位数据  . # A/S 关闭，只有 L1、C/A 数据  : # A/S 关闭，只有 L1、P1 数据  ~ # A/S 关闭，只有 L1、C/A、L2、P2 数据  \* # A/S 关闭，有 L1、P1、L2、P2 数据  , # A/S 开启，有 L1、C/A 数据  ; # A/S 开启，有 L1、P1 数据  o # A/S 开启，有 L1、C/A、L2、P2 数据  y # A/S 开启，有 L1、P1、L2、P2 数据  L # 对 L1、L2 的失锁指示  \_ # 卫星在水平面和设定高度角之间，没有记录数据 |

和定位有关的符号有：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | ^ # 有较大的位置变化  X # 码定位失败  C # 定位比较散乱  H # 水平方向定位不准  V # 高程方向定位不准  T # 各个方向定位均不准  > # 动态测量正常  o # 静态测量正常  O # 观测量不足  E # 星历不足  S # 卫星数不足 |

和接收机钟差有关的符号有：

|  |  |
| --- | --- |
| 1  2  3 | - # 重新设置减了数个毫秒  + # 重新设置加了数个毫秒  ^ # 没有记录下观测历元 |

**GFZRNX 文件头信息编辑功能详解**

[2016-12-10](http://www.gnss.help/2016/12/10/gfzrnx-header-edit/)

[GFZRNX](http://www.gnss.help/categories/GFZRNX/)

[GFZRNX](http://www.gnss.help/tags/GFZRNX/), [RINEX编辑](http://www.gnss.help/tags/RINEX%E7%BC%96%E8%BE%91/)

[前文](http://www.gnss.help/2016/11/15/gfzrnx-edit/)在介绍 GFZRNX 程序的文件编辑命令时，并没有提到其对 RINEX 格式文件的文件头信息的导出和编辑功能。相比 TEQC 程序，该程序的文件头信息编辑的功能更强大，但也更复杂。为协调各文章的篇幅，将其独立成此文。

首先需要说明的是，GFZRNX 支持对 RINEX 格式的观测文件（O-文件）、导航文件（N-文件）和气象文件（M-文件）的操作，但考虑到我们编辑得最多的还是观测文件，因此本文的示例以对 O-文件的操作为主。

**信息导出**

介绍信息编辑功能前，让我们先顺便了解一下 GFZRNX 程序文件头信息导出的功能。该功能可以将文件头中的信息提取出来，便于查看、归档。

信息导出功能使用 -meta 参数，支持 TXT、JSON、XML 等格式，使用起来非常简单。下面的命令以 TXT 格式导出观测信息：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp shao0420.16o -meta basic:txt > shao0420.txt |

下面的命令以 JSON 格式导出观测信息：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp shao0420.16o -meta basic:json > shao0420.json |

下面的命令以 XML 格式导出观测信息：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp shao0420.16o -meta basic:xml > shao0420.xml |

限于篇幅，为保证行文流畅性，将导出的信息附在文末。

**信息编辑**

GFZRNX 程序的文件头信息编辑包括三种模式：更新/插入模式、替换模式和重命名模式。其中更新/插入模式主要用于修改文件头中的信息，替换模式用于编辑文件头标志，而重命名模式常用于重命名观测的卫星号和观测量。

使用该程序的信息编辑功能时，必须通过 -crux 参数引入一个配置文件。一份配置文件模板可以使用下面的命令获得：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -show\_crux > example.txt |

运行该命令后，得到示例文件 example.txt。但该文件几乎空无一物，并没有太大的演示价值，因此本文将以我的配置文件为例。

**更新/插入模式**

一份更新/插入模式的配置文件就像这样：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | update\_insert:  #-------------  O - SHAO.DAEJ:  "REC # / TYPE / VERS": { 1: "TRIMBLE NETG3" }  "ANT # / TYPE": { 1: "TRM59800.00", 2: "NONE" }  O - 2015209:00000 2016365:86399 - SHAO:  "APPROX POSITION XYZ": { 0: -3857167.6484, 1: 3108694.9138, 2: 4004041.6876 }  "ANTENNA: DELTA H/E/N" : { 0: 0.1209, 1: 0.0008, 2: 0.0007 }  O - SHAO:  "OBSERVER / AGENCY": { 0 : "SHAO", 1 : "SHAO CAS" }  O - DAEJ:  "OBSERVER / AGENCY" + 00000000:000000 20130126:235959: { 0: "KASI", 1: "KASI" }  "OBSERVER / AGENCY" + 20130127:000000 00000000:000000: { 0: "KASI", 1: "KASI KOREA" } |

该文件首行的“update\_insert”即声明了该配置为更新/插入模式。第 2 行为注释，以“#”号开头。

之后的三行作为一组。其中第 3 行的首个字符定义了要编辑的数据类型（O、M、N 分别代表 RINEX 格式的观测文件、气象文件和导航文件）。若省略该字符，则表示将设置应用到所有数据类型。连字符之后是要应用这些设置的目的站点列表，多个点名之间使用“.”号分隔。因此这里的配置指示：将这一组设置应用到 SHAO 和 DAEJ 两个站点。如果你希望将配置应用到所有输入的文件，可以在这里使用 “ALL”。

文件的第 4 行是一个配置项。如你所知，RINEX 文件头信息中包含“REC # / TYPE / VERS”的这一行有 3 个数据项，分别表示接收机编号、接收机类型和接收机版本。如果我们将这一行的数据项以 0 开始编号，那么三项对应的是：

* 0 ：接收机编号；
* 1 ：接收机类型；
* 2 ：接收机版本。

因此，这个配置项意为：更新“REC # / TYPE / VERS”这一行的信息，将其中编号为 1 的项目（即接收机类型）修改为“TRIMBLE NETG3”。

文件第 5 行与上一行类似，更新天线类型为“TRM59800.00”，更新天线罩为“NONE”。

第 7 行与第 3 行类似，但在文件类型和站点名之间插入了一段字符“2015209:00000 2016365:86399”。这段字符指明，以下的配置仅应用于观测时间在该时间段之内的数据。其中的起止时刻以年、年积日和秒数的方式指定。之后的两行配置分别更新了观测信息中的先验坐标和天线偏移。

第 10 行和第 11 行配置将 SHAO 站的观测者和观测机构分别修改为“SHAO”和“SHAO CAS”。这里省略了应用配置的起止时刻，因此默认将此配置应用到所有时间段的观测数据。

最后的 3 行对 DAEJ 站的观测者和观测机构进行配置，但在配置项中插入了以“+”号开始的一个起止时刻。这表示，在 2013 年 1 月 26 日 23 点 59 分 59 秒之前的观测数据，其观测者和观测机构修改为“KASI”，在 2013 年 1 月 27 日 0 点 0 分 0 秒之后的观测数据，其观测者和观测机构分别修改为“KASI”和“KASI KOREA”。这里的时刻是以年月日、时分秒的形式指定的。事实上，该配置文件中的日期和时刻的格式分别各有两种，即年积日、年月日和时分秒、日积秒。

要应用该配置文件，可以将配置文件保存为 updist\_crux.txt。然后执行类似如下的命令：

|  |  |
| --- | --- |
| 1  2 | $ gfzrnx -finp daej0420.16o -crux updist\_crux.txt -kv > temp/dnew0420.16o  $ gfzrnx -finp shao0420.16o -crux updist\_crux.txt -kv > temp/snew0420.16o |

以上两个命令将分别得到修改后的文件 dnew0420.16o 和 snew0420.16o。

**替换模式**

替换模式的配置于之前的更新/插入模式的配置方式有许多相同之处。一份替换模式的配置文件如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | replace:  #-------  ALL:  string\_from: "PGM/RUN BY/DATE"  string\_to: "PGM / RUN BY / DATE"  DAEJ.SHAO:  regexp\_from: "^(.{60})PGM\s\*/\s\*RUN\s\*BY\s\*/\s\*DATE\s\*$"  regexp\_to: "$1PGM / RUN BY / DATE" |

首行声明配置为替换模式，之后以“#”号开头的一行为注释。第 3 行和第 6 行各指出了配置项的应用到哪些站点。

对于设置将被替换的项目和替换后的内容，既可以使用字符串指定，如第 4 到 5 行；也可以使用[正则表达式](https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F)指定，如第 7 到 8行。只需分别以“string”和“regexp”作为配置项的前缀。

对于配置项的后缀，“from”指定要被替换的内容，而“to”指定替换后的内容。

与之前类似，要应用该配置文件，可以将配置文件保存为 repl\_crux.txt。然后执行类似如下的命令：

|  |  |
| --- | --- |
| 1  2 | $ gfzrnx -finp daej0420.16o -crux repl\_crux.txt -kv > temp/dnew0420.16o  $ gfzrnx -finp shao0420.16o -crux repl\_crux.txt -kv > temp/snew0420.16o |

**重命名模式**

重命名模式常用来修改观测卫星号和观测类型标志。实际上，该模式也只有这两个功能。并且在声明时需要明确指明要重命名的是卫星号还是观测类型。

一份修改卫星号的配置文件如下：

|  |  |
| --- | --- |
| 1  2  3  4  5 | rename: prn  #----------  ON - 20140105:000000 20150101:000000 - E51 - E01: ALL  ON - 20140105:000000 00000000:000000 - E52 - E02: DAEJ.SHAO  E53 - E03: ALL |

首行声明配置为重命名卫星编号，之后的一行为注释。

第 3 行一开始的“ON”指定将配置应用于观测文件与导航文件，然后的两个时刻指定应用配置的时间。之后指定将卫星“E51”重命名为“E01”，应用的站点为所有站点。第 4 行与上一行类似，但限制将配置应用到 DAEJ 和 SHAO 两个站点。最后一行没有指定设置要应用的文件类型和时段，因此该设置将应用到所有时段的所有文件。

一份重命名观测类型的配置文件如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | rename: obs  #----------  20140105:000000 20150101:000000 - L2X - L2L - G : DAEJ.SHAO  20140105:000000 20150101:000000 - L2L - L2X - G : DAEJ  20140105:000000 20150101:000000 - \*\*X - \*\*L - C : ALL  20140105:000000 20150101:000000 - \*2 - \*1 - G04.G08 : ALL  \*2 - \*1 - C : ALL |

首行声明配置为重命名观测类型，之后的一行为注释。

第 3 行与第 4 行，指定了应用配置的时间段、需重命名的观测类型、重命名后的类型、卫星系统和应用的站点。第 4 行以通配符的方式指定了要重命名的观测类型与重命名后的类型。第 5 行与上一行类似，但限制了应用范围为 G4 和 G8 卫星。最后一行省略了时段，因此该配置将应用于所有观测时段的数据。

应用重命名模式的命令方式与之前的两个模式一致，这里不在赘述。

**导出信息样例**

**TXT 格式导出的观测信息**

以下为上文导出的 shao0420.txt 文件的内容。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | antenna:  height:  x = 0  y = 0  z = 0  name = AOAD/M\_T  number = 429  radome = JPLA  data:  epoch:  first = 2016 02 11 00 00 00.0000000  interval = 30.000  last = 2016 02 11 23 59 30.0000000  file:  md5 = c8ad2534683bf037c7a9e77eab2ef0a3  system = G  type = O  version = 2.11  receiver:  firmware = CQ00  name = ASHTECH UZ-12  number = UC2200524020  site:  agency = SHANGHAI OBSERVATORY  name = SHAO  number = 21605M002  observer = GGN  position:  x = -2831733.5830  y = 4675665.9580  z = 3275369.4100 |

**JSON 格式导出的观测信息**

以下为上文导出的 shao0420.json 文件的内容（为增加可读性，手动添加了缩进）。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | {  "receiver":{  "number":"UC2200524020",  "name":"ASHTECH UZ-12",  "firmware":"CQ00"  },  "site":{  "number":"21605M002",  "position":{  "y":"4675665.9580",  "x":"-2831733.5830",  "z":"3275369.4100"  },  "name":"SHAO",  "agency":"SHANGHAI OBSERVATORY",  "observer":"GGN"  },  "file":{  "system":"G",  "version":"2.11",  "type":"O",  "md5":"c8ad2534683bf037c7a9e77eab2ef0a3"  },  "data":{  "epoch":{  "first":"2016 02 11 00 00 00.0000000",  "last":"2016 02 11 23 59 30.0000000",  "interval":"30.000"  }  },  "antenna":{  "number":"429",  "name":"AOAD/M\_T",  "height":{  "y":0,  "x":0,  "z":0  },  "radome":"JPLA"  }  } |

**XML 格式导出的观测信息**

以下为上文导出的 shao0420.xml 文件的内容。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | <?xml version="1.0" encoding="UTF-8" standalone="yes"?>  <meta\_info>  <antenna>  <height>  <x><![CDATA[0]]></x>  <y><![CDATA[0]]></y>  <z><![CDATA[0]]></z>  </height>  <name><![CDATA[AOAD/M\_T]]></name>  <number><![CDATA[429]]></number>  <radome><![CDATA[JPLA]]></radome>  </antenna>  <data>  <epoch>  <first><![CDATA[2016 02 11 00 00 00.0000000]]></first>  <interval><![CDATA[30.000]]></interval>  <last><![CDATA[2016 02 11 23 59 30.0000000]]></last>  </epoch>  </data>  <file>  <md5><![CDATA[c8ad2534683bf037c7a9e77eab2ef0a3]]></md5>  <system><![CDATA[G]]></system>  <type><![CDATA[O]]></type>  <version><![CDATA[2.11]]></version>  </file>  <receiver>  <firmware><![CDATA[CQ00]]></firmware>  <name><![CDATA[ASHTECH UZ-12]]></name>  <number><![CDATA[UC2200524020]]></number>  </receiver>  <site>  <agency><![CDATA[SHANGHAI OBSERVATORY]]></agency>  <name><![CDATA[SHAO]]></name>  <number><![CDATA[21605M002]]></number>  <observer><![CDATA[GGN]]></observer>  <position>  <x><![CDATA[-2831733.5830]]></x>  <y><![CDATA[4675665.9580]]></y>  <z><![CDATA[3275369.4100]]></z>  </position>  </site>  </meta\_info> |

# PyYAML 安装与使用演示

[2016-12-01](http://www.gnss.help/2016/12/01/install-pyyaml/)

[PYTHON](http://www.gnss.help/categories/Python/)

[PYTHON](http://www.gnss.help/tags/Python/)

[YAML](http://yaml.org/)（YAML Ain’t Markup Language）是一种直观的能够被电脑识别的数据序列化格式，既方便被人类阅读，又容易被程序解析。相比 XML 和 JSON ，其语法更简洁直白。YAML 非常适合用来保存数据和程序配置，但可惜 Python 语言标准库中并没有集成解析 YAML 的库（[Anaconda](https://anaconda.org/) 已经集成了）。我们可以通过安装 [PyYAML](http://pyyaml.org/) 模块来获得操作 YAML 格式文件的功能。

本文将介绍 PyYAML 库的安装过程。

## 简介

YAML 是一种人类友好、跨语言易扩展、基于 Unicode 的数据描述语言。PyYAML 是 Python 用于解析和编码 YAML 文件的程序包，由 Kirill Simonov 编写，兼容 Python 2 和 Python 3，代码以 MIT 协议托管在 [Github](https://github.com/yaml/pyyaml) 网站上。

### Windows 操作系统

为 Windows 操作系统安装 PyYAML 非常简单。首先需保证计算机已安装 Python 运行环境，然后可以使用以下任意一种方式安装。

#### 使用引导程序

最简单直接的方式是在 [PyYAML 的下载页面](http://pyyaml.org/wiki/PyYAML)下载适用于 Windows 系统的对应安装程序，双击安装即可。安装程序将自动检测 Python 环境的安装目录，你只需确认安装目录正确无误。但该方法的缺点是：网站提供的安装程序可能不适用于最新的 Python 版本。比如，截至修改该文章时，该网站还没有提供适配 Python 3.6 的安装引导程序。

#### 使用 Wheel

有些开发者将常用的开源 Python 模块提前编译，并制作成可供 Wheel 程序直接安装的 .whl 文件。你可以从 [加州大学尔湾分校（UC Irvine）的页面](http://www.lfd.uci.edu/~gohlke/pythonlibs/#pyyaml)下载适用的 PyYAML 程序包进行安装。

安装之前必须首先安装 Wheel，在命令行窗口执行：

|  |  |
| --- | --- |
| 1 | $ pip install wheel |

然后就可以使用 pip 安装 .whl 程序包了。以适用于 Python 3.6 的 64 位程序包为例，在该文件所在目录执行：

|  |  |
| --- | --- |
| 1 | $ pip install PyYAML‑3.12‑cp36‑cp36m‑win\_amd64.whl |

### Linux 操作系统

根据 PyYAML 官方给出的安装方法，下载适用于 Linux 系统的程序包。解压后进入软件包目录，执行以下命令即可完成安装：

|  |  |
| --- | --- |
| 1 | $ python setup.py install |

但我在 RHEL7 操作系统按照上述方法编译安装时，并没有成功，错误原因是“找不到文件 yaml.h”。我的解决方案是：首先安装 libYAML，再安装 PyYAML。

#### 安装 libYAML

[libYAML](http://pyyaml.org/wiki/LibYAML) 是使用 C 语言实现的 YAML 解析库，截至目前的最新版本为 0.1.7。可以很容易地将其[源代码包](http://pyyaml.org/download/libyaml/yaml-0.1.7.tar.gz)下载至本地。

使用下面的命令将安装包解压：

|  |  |
| --- | --- |
| 1 | $ tar -zxvf yaml-0.1.7.tar.gz |

解压后，使用 cd 进入解压得到的文件夹，执行安装命令（很可能需要 root 权限）：

|  |  |
| --- | --- |
| 1  2  3 | $ ./configure  $ make  $ make install |

#### 安装 PyYAML

libYAML 安装完成之后，就可以继续安装 PyYAML 了。在 PyYAML 解压得到的文件夹中执行：

|  |  |
| --- | --- |
| 1 | $ python setup.py install |

运行上述命令将为系统默认的 Python 安装 PyYAML。但如果你计算机上同时安装了 Python2 与 Python3，那么需要使用两个版本的 Python 各自执行一遍安装程序：

|  |  |
| --- | --- |
| 1  2 | $ python2 setup.py install  $ python3 setup.py install |

## 演示程序

PyYAML 模块的使用非常简单，使用 yaml.load() 方法解析 YAML 文档，使用 yaml.dump() 方法将数据格式化为 YAML。其中 yaml.load() 方法既可以解析字符串，也可以处理打开的 YAML 文件。以下是采用 Python 3 语法的一个简单示例：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> import yaml  >>> yaml.load('greeting: "Hello world!"')  {'greeting': 'Hello world!'}  >>> yaml.load(StringIO('foo: [bar, baz]'))  {'foo': ['bar', 'baz']} |

同样，yaml.dump() 函数也可以处理两种场景。如果你希望该函数返回符合 YAML 格式的字符串，你可以只输入要编码的数据；如果你希望将数据编码输入到文件，只需多传入一个可写的文件流对象：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | >>> import yaml  >>> data = {'foo':{'bar': 'hello', 'baz': 'world'}}  >>> print(yaml.dump(data))  foo: {bar: hello, baz: world}  >>> pseudo\_file = StringIO()  >>> yaml.dump({'greeting': 'Hello world!'}, pseudo\_file)  >>> print(pseudo\_file.getvalue())  {greeting: Hello world!} |

你或许已经发现了：yaml.dump() 函数的输出比较紧凑，与 YAML 以缩进来表示层次的语法不甚一致。为增强人类可读性，你可以为该函数增加一个 default\_flow\_style=False 参数来强制使用缩进形式的编码风格：

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> data = {'foo':{'bar': 'hello', 'baz': 'world'}}  >>> print(yaml.dump(data, default\_flow\_style=False))  foo:  bar: hello  baz: world |

# 常用的 RINEX 版本格式转换程序整理

[2016-11-27](http://www.gnss.help/2016/11/27/rinex-version-switch/)

[RINEX](http://www.gnss.help/categories/RINEX/)

[RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/)

RINEX 格式是 GNSS 领域最常用的数据存储、交换格式，至今已经发布了多个版本标准。但是受限于 GNSS 数据处理程序的更新频率，现在的数据处理软件尚不能支持所有的 RINEX 格式。因此在进行数据处理时，将待处理数据在 RINEX 各格式之间转换是很常见的工作。

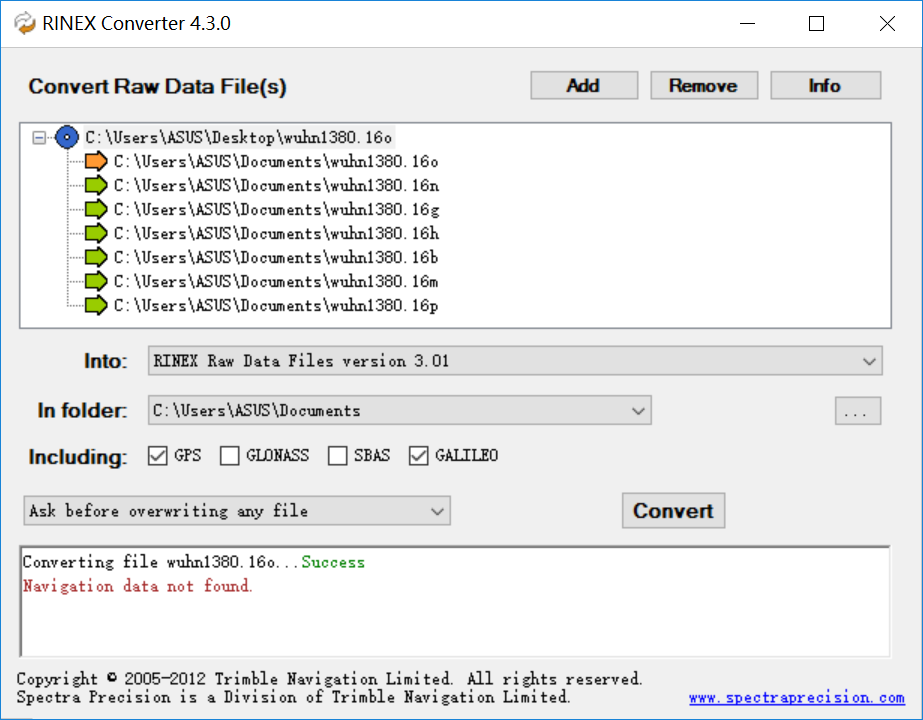
本网站已经介绍过使用得最多的两个 RINEX 格式版本数据转换工具。但都只是在程序使用介绍中提及，至今尚没有将该主题统一整理。本文中，将统一整理具备该功能的程序。它们既有适用于 UNIX/Linux 操作系统的命令行工具，也有界面友好的图形界面软件，还有运行于网页上的在线应用，可以满足你五花八门的各类需求。

## RINEX Converter

美国 Ashtech 公司开发的 [RINEX Converter](ftp://ftp.ashtech.com/OEM_Sensor_ADU/UTILITY%20SOFTWARE/RINEX%20Converter_Spectra%20Precision%20/Setup%20RINEX%20Converter%204.6.11/RINEXSetup.exe)，是一款用于将 RINEX 数据在各格式版本之间进行转换的程序。Ashtech 公司于 2011 年被 Trimble 收购。

RINEX Converter 适用于微软 Windows 操作系统，具备友好的图形操作界面，学习成本低。但该软件很久没有更新过了，当前的最新版本为 4.6.11。其支持的 RINEX 版本号有：2.10、2.11、2.11 for OPUS、2.12、3.00 以及 3.01，可以将 RINEX 格式数据在这些版本号之间任意转换。

程序的使用界面如图：

[](http://www.gnss.help/images/rinex-converter.PNG)RINEX Converter

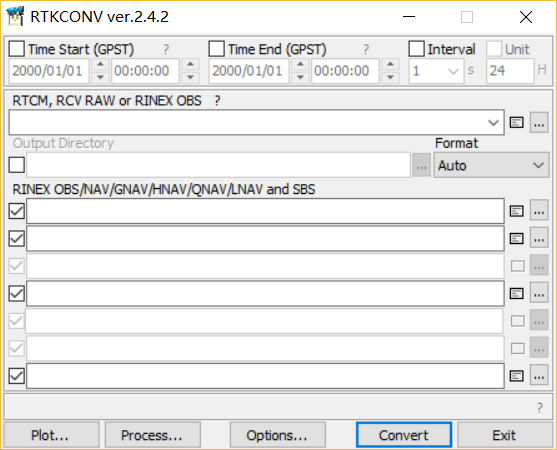
在该程序中，使用右上方的 “Add” 和 “Remove” 按钮对要处理的数据文件进行添加、删除操作。选择一个待处理的文件后，还可以点击 “Info” 按钮对其中的文件头信息进行修改操作。

程序界面中间的 “Into”、“In folder” 及 “Including” 三个选项分别用于配置输出文件的版本号、输出文件夹及所包含的 GNSS 卫星系统。并可以在“Including”选项的下方配置当出现同名文件时的行为。最下方的空白区域为转换过程中的消息的输出面板。

最后，点击其中的 “Convert” 按钮启动数据转换工作。

## RTKLib

rtkconv 是 [RTKLib](http://www.rtklib.com/) 软件中一个 GNSS 数据格式转换程序，支持 RTCM、BINEX 等数据流以及 Javad、u-blox、NovAtel 等接收机厂商的自定义格式数据。当然，其输入输出也支持 RINEX 2.10、2.11、2.12、3.00、3.01、3.02 等，你可以使用该程序实现 RINEX 各版本之间的转换。该程序的主界面如下：

[](http://www.gnss.help/images/rtkconv.png)RTKCONV

该程序使用非常简单，点击 “RTCM, RCV RAW or RINEX OBS” 文本框右侧的 “…” 按钮选择要中转换的文件，或直接在输入文件路径；在 “Output Directory” 文本框中设置输出的文件夹，或者逐个设置每个文件的输出路径。另外，为了能够同时输入多个文件，在设置输入文件时也可以使用 \* 号作为通配符。

如果你的计算机运行的不是 Windows 操作系统，对于 UNIX/Linux，本程序有对应的 CLI（命令行）程序：convbin，了解该程序的使用请移步本博客 [RTKLIB 中 convbin 模块的使用](http://www.gnss.help/2018/03/26/rtklib-convbin-usage/)一文。

## GNSS Converter

[GNSS Converter](https://www.gps-solutions.com/GNSSConverter/index.html) 是由 GPS Solutions 网站提供的一个在线 RINEX 格式数据转换工具。截至本文写作时，其支持的 RINEX 格式版本号有：2.11、2.12 及 3.02。

使用该工具前需要在网站进行注册。提交自己的名字和邮箱地址，系统将为你发送一封验证邮件（可能出现在垃圾信箱中），点击其中的验证链接即可创建一个账户。

按照系统引导登录后将能看到程序使用界面。首先按照提示上传需要处理的 RINEX 文件，然后设置输出文件的版本号、输出文件命名方式、数据开始与结束时间和文件头信息等。之后点击“Generate File”按钮，就可以将程序输出的文件下载到本地。

## GFZRNX

[GFZRNX](http://semisys.gfz-potsdam.de/semisys/scripts/download/) 程序是由德国波兹坦地学研究中心（GFZ）的开发的一款用于 GNSS 数据预处理程序，它也具备一定的 RINEX 格式版本转换功能。目前该程序支持 UNIX/Linux、Windows 等操作系统，可以实现将数据文件格式到 2.11 或 3.03 之间的转换。即不管输入文件的版本号是多少，输出文件版本号只能是 2.11 或 3.03。

要使用 GFZRNX 程序的格式转换功能，只需使用 -vo 参数指定输出文件的大版本号（2 或 3）。该程序的配置及格式转换功能的详细使用说明请移步前文 [GFZRNX 程序配置及其格式相关功能的使用](http://www.gnss.help/2016/10/30/gfzrnx-config-translate/)。

## TEQC

[TEQC](https://www.unavco.org/software/data-processing/teqc/teqc.html) 程序是目前最常用的 GNSS 数据预处理程序。该程序由 UNAVCO 开发，支持 UNIX/Linux、Windows 等常用的操作系统。

与前面介绍的几个 RINEX 格式版本转换工具相比，该程序的数据转换功能最弱。目前仅支持有限的版本作为输入（限于 2.11 及更早的版本），并将 2.11 版本格式作为输出。使用该程序的版本转换功能不需要显式的参数，因为任何的可接受输入都将产生 RINEX 2.11 版本的数据作为输出。该程序的详细配置说明可参考前文 [TEQC 程序配置及其数据转换功能的使用](http://www.gnss.help/2016/09/17/teqc-config-translate/)。

# GFZRNX 常用的文件编辑命令

[2016-11-15](http://www.gnss.help/2016/11/15/gfzrnx-edit/)

[GFZRNX](http://www.gnss.help/categories/GFZRNX/)

[GFZRNX](http://www.gnss.help/tags/GFZRNX/), [RINEX编辑](http://www.gnss.help/tags/RINEX%E7%BC%96%E8%BE%91/)

GFZRNX 是由德国波兹坦地学研究中心（GFZ）开发的一款用于 GNSS 数据预处理（适用于 RINEX 2 与 3 版本格式）的程序，支持对 RINEX 格式的观测数据文件（Obs）、广播星历文件（Nav）和气象数据文件（Met）的操作。

[前文](http://www.gnss.help/2016/10/30/gfzrnx-config-translate/)已经介绍过该程序的配置和文件格式转换功能的使用，本文将以实例的方式介绍其文件编辑功能的使用方法,包括文件分割与拼接、数据提取、采样率抽取、观测卫星筛选和观测量编辑等。

## 文件编辑命令

### 文件分割

要使用程序的文件分割功能，只需在运行时添加 -split 参数，然后输入分割文件的时段长度，其中时长以秒为单位。

示例，将 SHAO 站于 2016 年 2 月 11 日全天的观测数据分割为 24 个时长为 1 小时的观测文件：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp shao0420.16o -fout ::RX2:: -split 3600 |

上述命令中的 -fout ::RX2:: 参数指定输出文件以 RINEX 2 的命名方式自动命名。运行该命令，将得到时长为 1 小时的 24 个观测文件：shao042a.16o、shao042b.16o 至 shao042x.16o。

查看输出的这些文件，你将发现它们已被自动转换为 RINEX 3 格式，别忘了这是程序默认的输出格式。但是这可能不是你想要的，如果希望文件分割前后观测数据的大版本号不变，可以在命令中添加 -kv 参数。即：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp shao0420.16o -fout ::RX2:: -split 3600 -kv |

你也可能希望得到 RINEX 3 格式的输出文件，并且以 RINEX 3 格式的命名方式作为文件名。以下为一个示例：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp shao0420.16o -fout ::RX3:: -split 3600 |

需要补充的是，目前该程序似乎存在一个 Bug。当指定分割后的文件时段长于 1 小时，程序将为输出文件命名为类似 “site0010.16o” 的形式。这样的后果是：当 -split 参数指定的时段长度大于 3600 秒，程序将只输出第一个时段的文件。因为后续输出的文件与第一个文件重名，造成程序终止。当然，如果你在命令中还添加了 -f 参数用于强制覆盖重名文件，那么将只得到最后一个时段的观测文件。因为之前输出的文件被覆盖了。鉴于此，要获得时长超过 1 小时的数据时，建议使用下文介绍的数据提取的操作方式。

### 数据提取

数据提取即从观测文件中提取任意一段时间的数据。在使用该功能时，使用 -epo\_beg 参数来指定首历元开始时刻，使用 -d 参数指定以秒为单位的时长。其中输入的开始时刻可使用简化儒略日、GPS 周、年月日、年积日等多种形式。

依然以上文使用的 SHAO 站的观测数据为例。该天为 2016 年第 42 日，第 1883 GPS 周的星期四，对应的简化儒略日为 57429。示例，从 shao0420.16o 中提取 2 点开始，时长为 2 小时的观测数据：

|  |  |
| --- | --- |
| 1 | gfzrnx -finp shao0420.16o -epo\_beg 2016-02-11\_02:00:00 -d 7200 -kv > shao042c.16o |

运行这个命令后，将得到包含所需数据的文件 shao042c.16o。按照日期指定方式的不同，这个命令还可以如此改写：

|  |  |
| --- | --- |
| 1  2  3 | $ gfzrnx -finp shao0420.16o -epo\_beg 18834\_02:00:00 -d 7200 -kv > shao042c.16o # 日期以 GPS 周指定  $ gfzrnx -finp shao0420.16o -epo\_beg 2016042\_02:00:00 -d 7200 -kv -fout shao042c.16o # 日期以年与年积日指定  $ gfzrnx -finp shao0420.16o -epo\_beg 57429\_02:00:00 -d 7200 -kv -fout shao042c.16o # 日期以简化儒略日指定 |

### 文件拼接

说过文件分割与数据提取，现在介绍其逆操作——文件拼接。使用该功能时不需其它参数，只需以 -finp 参数指定要拼接的文件列表。其顺序可以是任意的，GFZRNX 程序能自动确定拼接的顺序。

下面的命令将前面文件分割时得到的 24 个文件中的前 3 个拼接到一起，并保持拼接前后文件的 RINEX 格式大版本号不变：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp shao042a.16o shao042c.16o shao042b.16o -kv > shao0420.16o |

### 采样率抽取

高采样率的文件体积通常很大，重新进行采样率抽取可以对其瘦身。要应用该功能可以使用 -smp 参数指定输出文件的采样间隔：

下面的命令将采样间隔为 30 秒的源文件重采样为 60 秒：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp shao0420.16o -smp 60 > shao0420\_60s.16o |

运行该命令，得到采样间隔为 60 秒的观测文件 shao0420\_60s.16o。

### 观测量编辑

GFZRNX 程序还支持直接对观测量进行编辑。要使用该功能，可以使用 -obs\_types 参数来指定要保留的观测量列表。其中多个项目之间以逗号分隔。

下面的命令将在输出文件中删去除了 L1、L2、P1、P2、C1、C2 之外的观测量：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp daej0420.16o -obs\_types L1,L2,P1,P2,C1,C2 -kv > temp0420.16o |

如果不关心观测频段而只关心观测类型，还可以使用如下的命令：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp daej0420.16o -obs\_types L,P,C -kv > temp0420.16o |

或者只关心观测频段不关心观测类型，可以使用如下的命令：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp daej0420.16o -obs\_types 1,2 -kv > temp0420.16o |

### 观测卫星筛选

有些 GNSS 数据观测量较差，或者受数据处理程序所限，可能需要从观测数据中删除某些观测数据。GFZRNX 程序支持对卫星或卫星系统进行筛选。

-prn 参数和 -no\_prn 参数用于对卫星进行筛选。其中 -prn 参数用于设置保留的卫星，而 -no\_prn 参数用于设置要去除的卫星。对于多个卫星的操作，可以用逗号进行分隔，亦可使用“-”指定起止卫星号。

下面的命令将 GLONASS 卫星 R1 与 R5 的观测数据删除：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp daej0420.16o -no\_prn R01,R05 -kv > temp0420.16o #卫星号最好使用两位数字，否则易出错 |

运行命令后，检查输出的文件 temp0420.16o，发现其中 R1 与 R5 的观测数据被删去了。

下面的命令则用于删除从 R1 到 R5 之间所有卫星的观测：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp daej0420.16o -no\_prn R01-05 -kv > temp0420.16o |

运行命令后，检查输出的文件 temp0420.16o，发现其中 R1、R2、R3、R4、R5 的观测数据都被删去了。

-prn 参数的使用方式与 -no\_prn 类似。以下的命令将在输出文件中只保留对从 G1 到 G30、从 R1 到 R10 卫星的观测：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp daej0420.16o -prn G01-30,R01-10 -kv > temp0420.16o |

### 卫星系统筛选

除了使用 -prn 或 -no\_prn 对某些卫星的数据进行操作，还可以使用 -satsys 参数直接对卫星系统的筛选，以下的命令将在观测文件中删去除 GPS 和 GLONASS 系统之外的所有卫星：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp daej0420.16o -satsys GR -kv > temp2000.16o |

值得注意的是，受到 RINEX 2 标准的限制（RINEX 2.11 标准未定义北斗观测量），筛选包含北斗卫星的观测数据时可能会出现问题。具体表现为，当输出格式为 RINEX 2 时，虽然设置了保留北斗系统卫星，但输出文件中的北斗卫星观测量被空白取代。不过对星历文件操作是没有问题的。

下面的命令将从混合的星历数据中分离出北斗卫星的信息：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp BRDC00IGS\_R\_20170420000\_01D\_MN.rnx -satsys C > brdc0420.17c |

运行命令后，得到只包含北斗卫星轨道信息的星历文件 brdc0420.17c。

### 星历重排序

上文对卫星系统筛选后输出的星历文件中，卫星轨道信息按卫星的 PRN 编号升序排列，这可能不是你想要的。通过 -ns参数可以对其中的卫星轨道信息重新排序。该参数接受两个选项：prn 或 time，其中 prn 指定输出文件中卫星轨道信息按照卫星 PRN 编号排序，而 time 则将按照发布时间排序。

示例，对输入的广播星历文件按照发布时间重新排序：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp brdc0420.17n -ns time -kv > brdn0420.17n |

对输入的广播星历文件按照卫星的 PRN 编号重新排序：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp brdc0420.17n -ns prn -kv > brdn0420.17n |

# GAMIT 中的文件合并命令

[2016-11-06](http://www.gnss.help/2016/11/06/gamit-merge-script/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/), [RINEX编辑](http://www.gnss.help/tags/RINEX%E7%BC%96%E8%BE%91/)

可用于 RINEX 文件合并的工具实在是太多了，[之前的文章](http://www.gnss.help/2016/09/19/teqc-edit/)中就曾介绍过使用 TEQC 程序达到这一目的的方法。但你可能还不知道，GAMIT/GLOBK 程序包中也包含了实现类似功能的脚本。本文将介绍附在 GAMIT/GLOBK 程序包中的两个脚本：sh\_merge\_rinex 和 sh\_merge\_nav，分别用来合并 RINEX 观测文件（Obs）和卫星星历文件（Nav）。

## 合并观测文件

sh\_merge\_rinex 脚本用于将多个 RINEX 观测数据合并为一个独立文件。其使用方式为：

|  |  |
| --- | --- |
| 1 | $ sh\_merge\_rinex -site <station> -year <yr> -days <doy> [-int <sampling> -obs <max\_obs>] |

其中：<station> 为 4 个字符的站点名；<yr> 表示数据观测的 4 位年；<doy> 为观测数据的 3 位年积日列表。其他的参数都是可选项：<sampling> 为输出数据的采样间隔，默认为 30s；<max\_obs> 为输出数据的观测类型，默认为所有类型。

示例，将 BJFS 站与 2016 年第 235 日的观测数据 bjfs235a.16o、bjfs235g.16o、bjfs235m.16o 和 bjfs235s.16o 合并为一个文件：

|  |  |
| --- | --- |
| 1 | $ sh\_merge\_rinex -site bjfs -yr 2016 -days 235 |

运行此命令后，得到合并完成的文件 bjfs2350.16o。

## 合并导航文件

sh\_merge\_nav 用于将多个 RINEX 星历数据合并为一个独立文件。该命令要求输入的待合并的文件至少有 5 个，否则合并不会成功。其使用方式为：

|  |  |
| --- | --- |
| 1 | $ sh\_merge\_nav -t <yr> <doy> [-c <eph> -p <path> -d] |

其中：<yr> 与 <doy> 分别表示待合并数据的 4 为年与 3 位年积日。其他项皆为可选项：<eph> 为 4 个字符的轨道名，默认为 brdc；-d 指定在转换时运行 [dos2unix](http://www.gnss.help/2017/07/24/dos2unix-install-usage/) 程序。

示例，合并 2016 年第 235 日每个文件时长 4 个小时的广播星历文件：brdc235a.16n、brdc235e.16n 等：

|  |  |
| --- | --- |
| 1 | $ sh\_merge\_nav -t 2016 235 -c brdc |

运行此命令后，得到合并完成的文件 brdc2350.16n。

**GFZRNX 配置及格式相关功能的使用**

[2016-10-30](http://www.gnss.help/2016/10/30/gfzrnx-config-translate/)

[GFZRNX](http://www.gnss.help/categories/GFZRNX/)

[GFZRNX](http://www.gnss.help/tags/GFZRNX/), [RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/)

提到 GNSS 领域的数据预处理工具，你首先想到的肯定是 [TEQC](https://www.unavco.org/software/data-processing/teqc/teqc.html) 程序。该程序因其丰富的数据转换、编辑和质量检查等功能而被人们所熟知。但该程序也并非尽善尽美。比方说，截至目前，该程序在 RINEX 各版本之间（尤其是 RINEX 2 与 3 之间）的格式转换功能尚弱。

本文将介绍另一个 GNSS 数据预处理程序：GFZRNX。该程序也具有丰富的功能，可以帮助你轻松跨越 RINEX 2 与 3 版本格式之间的鸿沟。如果配合 TEQC 程序双剑合璧，更能让你在数据预处理工作方面游刃有余。

**程序简介**

[GFZRNX](http://semisys.gfz-potsdam.de/semisys/scripts/download/) 程序是由德国波兹坦地学研究中心（GFZ）的开发的一款用于 GNSS 数据预处理（主要适用于 RINEX 2 与 3 版本格式）的程序，支持 Windows、Linux、macOS 等常见的操作系统。此程序并不开源，但对于非商业用途的用户，提供免费的使用授权。在程序的[介绍页面](http://semisys.gfz-potsdam.de/semisys/scripts/download/)，点击 “GFZ software” 标签即可看到程序的下载链接。

GFZRNX 程序支持对 RINEX 格式的观测数据文件（Obs）、广播星历文件（Nav）和气象数据文件（Met）的操作。目前提供的功能有：

* RINEX 格式检查与修复；
* RINEX 版本格式转换；
* RINEX 文件分割与拼接；
* RINEX 文件头信息编辑与导出；
* RINEX 数据编辑，如采样抽取、观测卫星筛选等；
* RINEX 文件差异比较；
* RINEX 数据质量分析。

如果你在数据处理时使用了此程序，请在论文后面为它添加如下的一个引用：

Nischan, Thomas (2016): GFZRNX - RINEX GNSS Data Conversion and Manipulation Toolbox. GFZ Data Services. <http://dx.doi.org/10.5880/GFZ.1.1.2016.002>

**环境配置**

**Windows 操作系统**

对于 Windows 操作系统，下载对应的可执行文件后，打开 “命令提示符” 窗口，使用 cd 命令进入程序所在目录，然后键入程序名（gfzrnx\_win32.exe 或 gfzrnx\_win64.exe）即可运行程序。

我可不想每次使用该程序前切换工作目录，也不喜欢命令后面诸如 “win32” 或 “win64” 这样的后缀。因此我将其重命名为 “gfzrnx.exe”，然后移动至 “C:\Windows\System32” 文件夹内。这样每次使用该程序时，只需要在 “命令提示符” 窗口键入 gfzrnx 就行了。当然，你也可以将该程序所在目录添加到系统的 Path 变量。

**Linux 或 macOS 操作系统**

对于 Linux 或 macOS 操作系统，下载对应的可执行文件后，首先将其重命名为 “gfzrnx” 以去除冗长的后缀，然后为程序分配可执行权限：

|  |  |
| --- | --- |
| 1 | $ chmod +x gfzrnx |

这样就可以在程序所在目录中通过命令 gfzrnx 来使用程序了。如果你希望在任何目录下都可以使用此程序，可以将其移动到 “/usr/bin” 目录下。

**操作模式**

使用 GFZRNX 程序前，首先要了解其操作模式。该程序的输入，即可以来自文件，也可以来自标准输入或管道。对于来自于文件的数据，需要使用 -finp 参数指定。对于程序的输出，默认为标准输出，也可以使用重定向或 -fout 参数将输出转到文件。对于出错信息，默认为标准错误输出，也可以使用 -errlog 参数（或重定向）将出错信息转到文件。因此程序常见的操作模式为：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp <input\_files> -errlog <error\_log> [options] > <output\_file> |

这里的 [options] 代表附加的参数选项，用于指定数据处理中使用的功能。使用 GFZRNX 程序进行数据预处理时，任何符合 RINEX 2 或 3 格式标准的数据都可以作为输入，但只输出最新的 RINEX 2（目前为 RINEX 2.11） 或 3（目前为 RINEX 3.03）版本的数据。如果未明确指定输出数据的格式版本，则默认为 RINEX 3.03。

**格式相关操作**

**格式检查与修复**

GFZRNX 程序提供对 RINEX 格式的检查功能，可以检测你输入的数据文件是否合乎规范。对于格式不正确的内容，甚至会自动尝试修复它。经过格式检查，GFZRNX 将会更新文件头中的信息并且移除所有出错的观测数据。在使用诸如 PANDA、GAMIT/GLOBK 等高精度 GNSS 数据处理程序时，该功能可以解决很多因为数据格式造成的错误。要在数据预处理中使用该功能，只需在操作命令中添加 -chk 参数。

以 BJFS 站 2016 年 9 月 12 日的观测数据 bjfs2560.16o 为例，其原始数据格式为 RINEX 2.10。下面的命令将调用 GFZRNX 程序对该文件进行格式检查，并将尝试修复后的数据保存至新的文件 bjfx2560.16o：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp bjfs2560.16o -chk > bjfx2560.16o |

该命令执行完成后，如果你查看生成的 bjfx2560.16o 文件，将发现该文件已经符合最新的 RINEX 3 格式标准。如果你不希望将输出的文件转换为 RINEX 3 格式，可以添加一个 -kv 参数，例如：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp bjfs2560.16o -chk -kv > bjfx2560.16o |

此时查看生成的新文件 bjfx2560.16o，将发现其数据格式没有变成 RINEX 3，但也不再是原始的 RINEX 2.10，而是 RINEX 2.11。因为这是最新的 RINEX 2 格式标准，-kv 参数只指定数据格式大版本不发生变化。

**格式转换**

使用 GFZRNX 程序的数据格式转换功能时，只需在操作命令中使用 -vo 参数。其输入值被限制为 2 或 3，代表输出数据的 RINEX 格式版本。

以 CHAN 站点于 2016 年 9 月 12 日的观测数据为例，其原始文件为 RINEX 2.11 格式。下面的命令将其格式转化为 RINEX 3：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp chan2560.16o -vo 3 > CHAN00CHN\_R\_20162560000\_01D\_30S\_MO.rnx |

转化广播星历文件：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp chan2560.16n -vo 3 > CHAN00CHN\_R\_20162560000\_01D\_GN.rnx |

转化气象数据文件：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp dav12560.16m -vo 3 > DAV100ATA\_R\_20162560000\_01D\_30S\_MM.rnx |

类似的，下面的命令将之前的命令所生成的文件转化回 RINEX 2：

|  |  |
| --- | --- |
| 1 | $ gfzrnx -finp CHAN00CHN\_R\_20162560000\_01D\_30S\_MO.rnx -vo 2 > chan2560.12o |

如果你在 UNIX/Linux 操作系统上进行操作，还可以通过管道使 GFZRNX 程序与 RNXCMP、TEQC 等程序相配合。例如：

|  |  |
| --- | --- |
| 1 | $ crx2rnx CHAN00CHN\_R\_20162560000\_01D\_30S\_MO.crx - | gfzrnx -f -vo 2 > chan2560.16o |

运行该命令后，将从最初的 CHAN00CHN\_R\_20162560000\_01D\_30S\_MO.crx 文件直接得到符合 RINEX 2 格式标准的文件 chan2560.16o。这里的 -f 参数指示强制覆盖可能的已有文件。

至此，GFZRNX 程序的格式检查、修复与转换功能已经介绍完毕。对于此程序其他功能的使用介绍，请查看本站 [#GFZRNX](http://www.gnss.help/tags/GFZRNX/)标签中的文章。

**使用 CSS 实现标题自动编号**

[2016-10-23](http://www.gnss.help/2016/10/23/css-counter/)

[网站](http://www.gnss.help/categories/%E7%BD%91%E7%AB%99/)

[CSS](http://www.gnss.help/tags/CSS/), [网站](http://www.gnss.help/tags/%E7%BD%91%E7%AB%99/)

近几天在学习 HTML5 与 CSS3 时，偶然看到了 CSS 中自动为标题添加编号的特性。仔细阅读过 [W3Schools](http://www.w3schools.com/) 上提供的文档之后，试验了一下，确实非常好用。从此不需再花费精力校对标题编号，可以将时间集中到真正需要做的事情，也避免了手动编号出错的窘态。

这篇博文整理了我搜集的相关知识点，方便希望详细了解的人查看。

**涉及内容**

首先介绍一下本文涉及到的相关 CSS 知识点。其中用到的 CSS 伪元素和属性皆基于 CSS2.1 标准，所以基本不需要担心浏览器的兼容性问题。如果你对此还不够放心，可以在不同浏览器中打开文末给出的样例进行测试。

**before 伪元素选择器**

before 伪元素选择器用于在某个元素之前插入一些内容。所插入的内容由 content 属性指定，可以是字符串、图片、音频甚至视频文件等。

与之对应的，还有 after 伪元素，用于在某个元素的后面插入一些内容。

其用法如：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | /\* insert a word before anchor \*/  a:before {  content: 'goto: '  }  /\* add a icon after an url end by mp4\*/  a[href$=mp4]:after {  content: url(moive.png)  } |

**counter-increment 属性**

counter-increment 属性用于递增计数器，该属性可接受的输入有：

|  |  |
| --- | --- |
| 1 | counter-increment: none|id|initial|inherit; |

其中值的意义为：

* none：不递增计数器；
* id：需递增的计数器ID；
* initial：递增的步长，默认为 1；
* inherit：从父元素继承。

**counter-reset 属性**

counter-reset 属性用于重置计数器，比如在每篇文章开始前将计数器重置为 0。该属性可接受的输入有：

|  |  |
| --- | --- |
| 1 | counter-reset: none|name-number|initial|inherit; |

其中值的意义为：

* none：不重置计数器；
* name-number：需重置的计数器ID；
* initial：重置的值，默认为 0；
* inherit：从父元素继承。

**示例代码**

以下面的网页作为示例：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | <!doctype html>  <html>  <head>  <meta charset="utf-8">  <title>自动编号演示</title>  <link rel="stylesheet" href="counter.css">  </head>  <body>  <h1>文章大标题</h1>  <h2>前言</h2>  <h2>数据准备</h2>  <h3>观测数据准备</h3>  <h3>表文件准备</h3>  <h2>参数设置</h2>  <h3>高度角</h3>  <h3>模型设置</h3>  <h2>开始解算</h2>  <h2>成果查看</h2>  </body>  </html> |

其中引入的 counter.css 中的代码为：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | /\* reset h2 counter to -1 when meet a h1 element \*/  h1 {  counter-reset: h2Counter -1;  }  /\* insert number before h2 element \*/  h2:before {  content: counter(h2Counter)'. '  }  /\* increase h2 counter and reset h3 counter when meet a h2 element \*/  h2 {  counter-increment: h2Counter;  counter-reset: h3Counter;  }  h3:before {  content: counter(h2Counter) '.' counter(h3Counter) ' '  }  h3 {  counter-increment: h3Counter;  } |

将以上两块代码分别保存为 counter.html 和 counter.css，确保这两个文件位于同一目录内，然后使用浏览器打开其中的 counter.html 文件即可预览效果。

# Linux 系统安装 Node.js

[2016-10-16](http://www.gnss.help/2016/10/16/install-nodejs/)

[LINUX](http://www.gnss.help/categories/Linux/)

[LINUX](http://www.gnss.help/tags/Linux/)

在我们的传统印象中，JavaScript 是一种只能运行在浏览器环境中的脚本语言，直到 [Node.js](https://nodejs.org/zh-cn) 的出现终于改变了这一切。Node.js 是一个开源的、跨平台的 JavaScript 运行时，它基于 Google Chrome 浏览器的 V8 JavaScript 引擎，可以让我们在桌面和服务器上运行由 JavaScript 语言编写的脚本。现在的 Node.js 主要用于编写像 Web 服务器等网络应用，其事件驱动的特性使得开发者可以在不使用线程的情况下开发出能承载高并发的服务器。目前 Node.js 已经有数十万模块，通过一个名为 npm 的管理器进行管理。

本文将介绍在 Linux 系统安装 Node.js 的过程。

## 程序包获取

从 [Node.js 官方网站](https://nodejs.org/zh-cn/download/)下载适用于你的 Linux 系统的二进制安装包（Linux Binaries，主要是 32 位与 64 位的区别）。我下载的安装包为 node-v4.6.0-linux-x64.tar.xz，就以这个文件为例。

将所下载的程序包放在任何你希望的位置，然后在该目录内执行以下命令对其进行解压缩：

|  |  |
| --- | --- |
| 1 | $ tar xvf node-v4.6.0-linux-x64.tar.xz |

解压完成后将看到一个命名与压缩包类似的文件夹（如 node-v4.6.0-linux-x64），其中包含 Node.js 的可执行文件。要运行 Node.js 或 npm 包管理器程序，使用 cd 进入解压缩出现的文件夹内的 bin/ 目录，执行 node 或 npm 命令即可。

## 软件配置

你肯定希望能够在任何目录下使用 Node.js。并且，如果你的 Linux 计算机有多个用户，你可能也希望能够与其他用户共享该程序。

以管理员角色，将 Node.js 的文件夹移动到 /usr/local 目录内，并重命名为 node-v4.6.0：

|  |  |
| --- | --- |
| 1 | $ mv node-v4.6.0-linux-x64 /usr/local/node-v4.6.0 |

将 Node.js 所在文件夹访问权限设置为所有用户可读可执行：

|  |  |
| --- | --- |
| 1  2 | $ cd /usr/local  $ chmod 755 -R node-v4.6.0 |

将 node 和 npm 可执行文件链接至 /usr/bin 目录内：

|  |  |
| --- | --- |
| 1  2 | $ cd /usr/bin  $ ln -s /usr/local/node-v4.6.0/bin/\* ./ |

这样所有用户在任何目录下都可以使用 Node.js 了。

## 演示程序

是时候编写简单的 “Hello World” 程序来感受一下 Node.js 如何工作了。

### 本地命令行程序

新建一个文件，在其中输入以下内容：

|  |  |
| --- | --- |
| 1 | console.log("Hello World!"); |

将该文件保存为 localHelloWorld.js，然后运行命令：

|  |  |
| --- | --- |
| 1 | $ node localHelloWorld.js |

程序将输出一行字符串：Hello World!。

### Web 服务器程序

新建一个文件，在其中输入以下内容（源码来自 [Node.js 官网](https://nodejs.org/zh-cn/about/)）：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | const http = require('http');  const hostname = '127.0.0.1';  const port = 3000;  const server = http.createServer((req, res) => {  res.statusCode = 200;  res.setHeader('Content-Type', 'text/plain');  res.end('Hello World!\n');  });  server.listen(port, hostname, () => {  console.log(`Server running at http://${hostname}:${port}/`);  }); |

将该文件保存为 webHelloWorld.js，然后运行命令：

|  |  |
| --- | --- |
| 1 | $ node webHelloWorld.js |

打开本机的浏览器，访问：[http://127.0.0.1:3000](http://127.0.0.1:3000/)，将看到服务器响应的 “Hello World!”。

# 使用 GAMIT 解算可降水汽含量

[2016-10-12](http://www.gnss.help/2016/10/12/gamit-calculate-pwv/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/), [可降水汽含量](http://www.gnss.help/tags/%E5%8F%AF%E9%99%8D%E6%B0%B4%E6%B1%BD%E5%90%AB%E9%87%8F/)

可降水汽含量（Precipitable water vapor, PWV）是进行 GNSS 气象学研究的重要数据，目前对该数值的解算大多依赖于 GAMIT/GLOBK 软件。作为 GAMIT 基线解算的重要副产品，PWV 可用于为气象预报、气象灾害、气候变化等方面的研究提供重要的参考指标。

本文将介绍使用 GAMIT 程序解算 PWV 的过程。

## 数据准备

上面已经提到过，PWV 是 GAMIT 基线解算时的副产品。我们只需要在基线解算前做一些设置，就可以在数据处理的同时输出对测站地区气象参数如干延迟、湿延迟、PWV 等的估计结果。

作为示例，我将选用周边的几个 IGS 测站进行处理。需要注意的是，在你处理自己观测的数据时，最好也引入几个 IGS 站点进行联合解算。因为过短的基线会使得解算出的气象参数有一定的相关性。我选取了 2016 年 9 月 11 日的 7 个 IGS 站点（BJFS，DAEJ，LHAZ，SHAO，SUWN，URUM，WUHN）的观测文件作为示例。这些站点中个别还具有气象数据，我也将其引入进来。最终的文件组织结构如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | · work/  |—— brdc/  |—— brdc2550.16n  |—— igs/  |—— igs19140.sp3  |—— met/  |—— bjfs2550.16m  |—— daej2550.16m  |—— urum2550.16m  |—— wuhn2550.16m  |—— rinex/  |—— bjfs2550.16o  |—— daej2550.16o  |—— lhaz2550.16o  |—— shao2550.16o  |—— suwn2550.16o  |—— urum2550.16o  |—— wuhn2550.16o  |—— tables/ |

## 参数配置

### 表文件准备

共用表文件的更新方法[前文](http://www.gnss.help/2016/08/27/GAMIT-StepbyStep/)早已经提到过，不再赘述。此处仅说明获取映射函数模型的过程。

本次解算任务我将采用 VMF1 映射函数模型，该模型文件需要从 [MIT 的 FTP 服务器](ftp://everest.mit.edu/pub/GRIDS)下载，欲了解详情请移步[之前的文章](http://www.gnss.help/2016/10/02/grids-in-gamit/)。因为撰写本文时 2016 年还未过去，所以下载到的映射函数模型仅适用于对年积日在 284 以前的数据进行处理，其文件名为 vmf1grd.2016\_284（随获取日期不同，获取到的文件名可能略有差异）。下载后将其放入 GAMIT/GLOBK 程序安装目录的 tables/ 文件夹内。

在解算工程目录（work/）内使用 sh\_setup 命令将所需的共用表文件链接至解算目录的 tables/ 文件夹：

|  |  |
| --- | --- |
| 1 | $ sh\_setup -yr 2016 -doy 255 |

然后检查其中的 map.grid 是否正确链接至刚才下载的模型文件 vmf1grd.2016\_284：

|  |  |
| --- | --- |
| 1 | $ ls -l tables/map.grid |

如果系统输出中有类似这样的提示：

tables/map.grid -> /home/jon/gg/tables/map.grid.2016\_284

则说明链接正确，否则需要使用如下命令重新建立链接：

|  |  |
| --- | --- |
| 1  2 | $ rm -f tables/map.grid  $ ln -s ~/gg/tables/vmf1grd.2016\* tables/map.grid |

第一条用于删除原始链接，第二条用于创建新链接。注意：如果你处理的不是 2016 年的观测数据，使用实际的年替换这里的 “2016”。

### 参数配置

对于本文的任务，我打算使用 sh\_gamit 批处理命令进行数据解算。如[之前](http://www.gnss.help/2016/09/01/sh-gamit/)所述，此时的配置文件主要有四个：sestbl.、sittbl.、process.defaults 和 sites.defaults。在这里，我只修改必要的配置，其他项目保持默认。打开 sestbl.，将其中对应的项修改为：

Met obs source = RNX UFL GPT 50 　 ; hierarchical list with humidity value at the end; e.g. RNX UFL GPT 50 ; default GTP 50  
Output met = Y 　　 ; write the a priori met values to a z-file (Y/N)

这里的 “Met obs source = RNX UFL GPT 50” 表示需要测站处的温度和气压等气象观测数据时，首先检查工程目录的 met/ 文件夹中有没有对应测站的 RINEX 气象文件，然后检查 U-文件中有无该站的信息，最后才采用 GPT 模型中的值。末尾的 “50” 表示假设的相对湿度，默认为 50%。你也可以对其进行更改，但实际上该值对解算结果影响很小。

而 “Output met = Y” 一项，表示设置 GAMIT 程序在基线解算的同时输出对气象参数的估计值，这些结果将保存至 Z-文件中。

同样在 sestbl. 文件中，将映射函数修改为使用 VMF1：

DMap = VMF1 　　 ; GMF(default)/VMF1/NMFH; GMF now invokes GPT2 if gpt.grid is available (default)  
WMap = VMF1 　　 ; GMF(default)/VMF1/NMFW; GMF now invokes GPT2 if gpt.grid is available (default)  
Use map.list = N 　 ; VMF1 list file with mapping functions, ZHD, ZWD, P, Pw, T, Ht  
Use map.grid = Y 　 ; VMF1 grid file with mapping functions and ZHD

## 解算操作

完成上文的所有准备工作之后，就可以运行基线解算命令了：

|  |  |
| --- | --- |
| 1 | $ sh\_gamit -expt demo -d 2016 255 -met |

这里的 -met 参数指示在解算时引入 RINEX 格式的气象观测数据。如果你没有此类数据，可以省略该参数。等待解算完成，就可以在年积日文件夹看到输出的 Z-文件，在本示例中为：zbjfs6.255，zdaej6.255 等。

但如果你查看这些生成的 Z-文件，会发现其中并没有 PWV 解算结果而只有对流层总延迟和干延迟等参数信息，还需要执行 sh\_metutil 命令做进一步处理。这个命令的常见用法为：

|  |  |
| --- | --- |
| 1 | $ sh\_metutil -f <o-file> -z <z-file> |

其中 <o-file> 代表基线解算结果的 O-文件，<z-file> 代表输出的 Z-文件。具体到本例，在年积日文件夹执行命令：

|  |  |
| --- | --- |
| 1 | $ sh\_metutil -f odemoa.255 -z z\*.255 |

最终的可降水汽解算成果保存在 “met\_” 开头的文件中。具体到本例，它们是：met\_bjfs.16255，met\_daej.16255 等。

## 补充

如果你希望在基线解算完成后直接生成可降水汽的估计值，可以为 sh\_gamit 命令多加一个 -metutil Z 参数。如：

|  |  |
| --- | --- |
| 1 | $ sh\_gamit -expt demo -d 2016 255 -met -metutil Z |

另外，除了使用 Z-文件估计可降水汽，sh\_metutil 命令还支持由测站的 RINEX 气象观测数据（Met）和基线解算得到的 O-文件生成可降水汽估计值。其命令为：

|  |  |
| --- | --- |
| 1 | $ sh\_metutil -f <o-file> -m <met-file> |

在该命令中，<o-file> 代表基线解算结果的 O-文件，<met-file> 代表站点气象观测数据。

**GAMIT 中的 GRID 文件简介**

[2016-10-02](http://www.gnss.help/2016/10/02/grids-in-gamit/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/), [GRID文件](http://www.gnss.help/tags/GRID%E6%96%87%E4%BB%B6/)

在使用 GAMIT/GLOBK 程序进行数据处理时，你肯定已经注意到了其 tables 文件夹中存在许多 .grid 文件。如果你检查过 GAMIT/GLOBK 安装目录的 tables 文件夹，你会发现更多这样的文件。它们多数体积都很大，一般为数百 MB。这样的文件称为格网模型文件，一般在 [MIT 的 FTP 服务器](ftp://everest.mit.edu/pub/GRIDS/)上提供下载。

本文将试图逐个地介绍其中的格网模型文件，可以算是介绍其格网模型文件的一份文档。

**atl.grid**

全球的大气潮格网模型文件，由澳大利亚国家大学的 Paul Tregoning 创建。该文件包含在 GAMIT/GLOBK 程序的安装包中且不需要更新。也就是说，不管你处理哪一年的观测数据，都可以引入这个文件。

要在数据处理中应用大气潮格网模型，你首先需要保证在进行数据处理的工程文件夹的 tables/ 目录中，其下的 atl.grid 已经正确链接到 GAMIT/GLOBK 安装目录中的 tables/atl.grid 文件。然后在 sestbl. 中修改设置为：

Use atl.list = N 　　 ; Atmospheric tides, list file, not yet available  
Use atl.grid = Y 　　 ; Atmospheric tides, grid file

**atml.grid**

全球的无潮汐大气负载参数格网模型文件，由澳大利亚国家大学的 Paul Tregoning 根据卢森堡大学的 Tonie van Dam 提供的数据创建。该模型一般每年对应一个文件，供 GAMIT/GLOBK 使用的文件命名方式一般为：atmdisp\_cm.<year>。如用于 2015 年的文件名为 atmdisp\_cm.2015。

要在数据处理中应用无潮汐大气负载参数格网模型，首先从 [MIT 的 FTP 服务器](ftp://everest.mit.edu/pub/GRIDS/)上下载适用于当年的模型文件并将其放入至 GAMIT/GLOBK 安装目录的 tables/ 文件夹内，然后确保在进行数据处理的工程文件夹的 tables/ 目录中，其下的 atml.grid 正确链接到当年的模型文件。最后在 sestbl. 中设置：

Use atml.list = N 　　 ; Atmospheric (non-tidal) loading list file from LU  
Use atml.grid = Y 　　 ; Atmospheric (non-tidal) loading grid file from LU, converted to GAMIT format

**gpt.grid**

全球气压和温度模型文件，该模型由 Boehm 和 Schuh 提出。在估计对流层参数时，GAMIT/GLOBK 从该模型中内插获取观测所在地区的气压和温度。该模型一般不会更新，并且已经被包含在 GAMIT/GLOBK 程序的安装包中。

要在数据处理中使用全球气压与温度模型，首先确保你进行数据处理的工程文件夹的 tables 目录中，其下的 gpt.grid 已经正确链接到 GAMIT/GLOBK 安装目录中的 tables/gpt.grid 文件。然后在 sestbl. 中设置：

Met obs source = GPT 50 　　 ; hierarchical list with humidity value at the end; e.g. RNX UFL GPT 50 ; default GTP 50

这样，在进行数据处理时，GAMIT/GLOBK 就将从该模型中获取气压和温度参数。但内插出的精度毕竟不如实测数据，因此如果你有气象观测数据的话，最好还是将其引入。所以我推荐如下的设置：

Met obs source = RNX UFL GPT 50 　　 ; hierarchical list with humidity value at the end; e.g. RNX UFL GPT 50 ; default GTP 50

如此设置后，需要测站所在地区的气压和温度数据时，GAMIT/GLOBK 将首先检查有没有对应的 RINEX 格式气象数据，如果没有的话再检查有没有对应的 U-文件，最后才会尝试使用全球气压与温度模型提供的数据。

**map.grid**

全球的大气映射函数模型文件，该文件由澳大利亚国家大学的 Paul Tregoning 根据维也纳科技大学的 Johannes Boehm 提供的数据创建。GAMIT/GLOBK 目前支持 GMF、NMFW 和 VMF1 三种映射函数，但只有 VMF1 需要引入模型文件。该模型逐年更新，每年生成一个文件。你可以根据需要在 [MIT 的 FTP 服务器](ftp://everest.mit.edu/pub/GRIDS/)上下载，其命名方式为：vmf1grd.<year>。

要在数据处理时应用 VMF1 映射函数模型，需要首先下载适用于当年的模型文件放入 GAMIT/GLOBK 安装目录的 tables/ 文件夹内。然后确保在进行数据处理的工程文件夹的 tables/ 目录中，其下的 map.grid 正确链接到当年的模型文件。最后在 sestbl. 中设置：

DMap = VMF1 　　 ; GMF(default)/VMF1/NMFH; GMF now invokes GPT2 if gpt.grid is available (default)  
WMap = VMF1 　　 ; GMF(default)/VMF1/NMFW; GMF now invokes GPT2 if gpt.grid is available (default)  
Use map.list = N 　　 ; VMF1 list file with mapping functions, ZHD, ZWD, P, Pw, T, Ht  
Use map.grid = Y 　　 ; VMF1 grid file with mapping functions and ZHD

**met.grid**

虽然各版本的 GAMIT/GLOBK 一直保留了这个模型文件链接，但该模型并没有什么用。有 sestbl. 中对该项配置的说明为证：

Use met.list = N 　　 ; not yet supported  
Use met.grid = N 　　 ; not yet supported

**otl.grid**

全球的海潮模型文件，GAMIT/GLOBK 现在支持多种海潮模型。[MIT 的 FTP 服务器](ftp://everest.mit.edu/pub/GRIDS/)上提供的海潮模型文件有：

* otl\_CSR4.grid
* otl\_FES2004.grid
* otl\_FES99.grid
* otl\_GOT00.grid
* otl\_NAO99b.grid

要在 GAMIT 数据处理中应用全球海潮模型文件，首先下载对应的海潮模型文件放入 GAMIT/GLOBK 安装目录的 tables/ 文件夹内。然后确保在进行数据处理的工程文件夹的 tables/ 目录中，其下的 otl.grid 正确链接到对应的模型文件。最后在 sestbl. 中设置：

Tides applied = 31 　 ; Binary coded: 1 earth 2 freq-dep 4 pole 8 ocean 16 remove mean for pole tide  
　　　　　　　　　　; 32 atmosphere ; default = 31  
Use otl.list = N 　　　 ; Ocean tidal loading list file from OSO  
Use otl.grid = Y 　　 ; Ocean tidal loading grid file, GAMIT-format converted from OSO

这里的 “Tides applied”，以二进制的形式表示解算中应用的选项。如果你知道 UNIX/Linux 系统中的权限位表示方法，这一项应该很容易理解。否则，可以简易地以数字的对应关系计算：

* 1 ：earth;
* 2 ：freq-dep;
* 4 ：pole;
* 8 ：ocean;
* 16：remove mean for pole tide;
* 32：atmosphere

对于你要在解算中应用的项，只需将其对应的数字的和作为配置即可。例如这里的 31：

31 = 1 + 2 + 4 + 8 + 16

因此数据处理中应用到的选项为：

* earth;
* freq-dep;
* pole;
* ocean;
* remove mean for pole tide;

**补充**

文行至此，已经将 GAMIT/GLOBK 程序中所有的 .grid 文件都已做过简要的介绍。这里以一个小技巧作为最后的补充：或许你还发现，在 tables 文件夹内还存在一些与 .grid 文件同名的 .list 文件（除了 gpt.grid）。这是为了方便处理 IGS 跟踪站的观测数据，.list 文件是从模型文件中提取出的所有 IGS 站点对应的参数值。如果你要处理的全都是 IGS 站的观测数据，完全可以使用这些 .list 文件，只需在对应的配置项中将使用 .list 文件的选项设置为 “Y”，并将使用 .grid 文件的选项设置为 “N”。

**获取 IGS 数据的几种方法**

[2016-09-27](http://www.gnss.help/2016/09/27/download-igs-data/)

[RINEX](http://www.gnss.help/categories/RINEX/)

[IGS数据下载](http://www.gnss.help/tags/IGS%E6%95%B0%E6%8D%AE%E4%B8%8B%E8%BD%BD/)

解算 GNSS 观测数据时，经常需要下载一些由 IGS 等机构提供的数据。比如各卫星系统的广播星历、精密星历、IGS 跟踪站观测文件等等。

如果需要下载的数据不多，使用浏览器或资源管理器等访问 IGS 的 FTP 服务器，找到数据所在目录然后一个一个下载不是问题。但是如果重复性劳动过多，你肯定希望有更简单的方法。本文将以 Windows 和 Linux 操作系统为例，介绍从 IGS 服务下载所需数据的几种方式，希望能使部分读者摆脱一些数据准备的苦力工作。

**Windows 操作系统**

或许你早就已经注意到了，本站的 “[下载](http://www.gnss.help/download/)” 页面提供了一个名为 “苦力怕” 的 IGS 数据下载程序。这是我之前编写的一个数据下载程序，支持微软 Windows 7 及以上的操作系统。如果你希望在 Windows Vista 及更早的操作系统中使用，需要首先安装 [.NET 4.0](https://www.microsoft.com/zh-CN/download/details.aspx?id=17851) 运行环境。

目前该程序特性有：

* 支持设置 IGS 数据中心与登录邮箱；
* 下载过程中有进度条提示；
* 下载完成后自动解压。

程序支持的 IGS 数据中心：

* 武汉大学 GNSS 中心；
* 喷气推进实验室（JPL）；
* 斯克里普斯海洋研究所（SIO）。

目前支持的数据类型：

* IGS 跟踪站观测数据（Compact RINEX）；
* GPS 广播星历数据（BRDC）；
* 精密星历（IGSF）；
* 精密钟差（CLK）；
* 电离层文件（ION）；
* 地球自传文件（ERP）。

我会对该程序进行不定期地更新。如果本程序缺少你需要的数据类型，你可以给我发送邮件简单地描述你需要地数据类型及其下载地址，我会考虑在下次更新该程序时将其加入。或者，你还有其他对该程序的修改建议等，也可以通过邮件跟我联系来共同改进此程序。

**Linux 操作系统**

对于 Linux 操作系统的用户，获取 IGS 数据的方式更多样。不仅有很多写好的脚本可用，你还可以使用脚本语言来 DIY 自己的数据下载脚本。这里介绍两种较简单的方式：使用 GAMIT/GLOBK 提供的脚本命令和自己编写数据下载脚本。

这两种方式都需要使用 FTP 客户端程序，所以最好事先安装它。对于 Debian、Ubuntu 等 Linux 发行版，安装命令为：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install ftp |

CentOS、Fedora 等发行版对应的命令为：

|  |  |
| --- | --- |
| 1 | $ sudo yum install ftp |

安装之后，你可以运行 ftp 命令测试一下。

**使用 GAMIT 内置命令**

GAMIT/GLOBK 程序提供了许多的数据下载脚本，这些脚本使用 C Shell 语言编写，位于其安装目录的 com/ 文件夹内。在 GAMIT10.6 中，包含的数据下载命令及其功能为：

* sh\_get\_hfiles： 下载 IGS 网的 H-文件；
* sh\_get\_ion： 下载电离层文件；
* sh\_get\_met ： 下载 IGS 站气象数据；
* sh\_get\_nav： 下载广播星历文件；
* sh\_get\_orbits： 下载精密星历；
* sh\_get\_raw： 下载未处理过的原始数据；
* sh\_get\_rinex： 下载 IGS 站观测数据；
* sh\_get\_stinfo： 下载 IGS 站点的 station.info 文件。

这些命令的用法大同小异，而且在不输入参数运行命令时会显示帮助信息。限于篇幅，本文不逐个给出详细参数说明，只演示命令并说明运行效果。读者可自行查看帮助，非常简单。

获取 SOPAC 发布的 2016 年 9 月 12 日（年积日为 256）全球 IGS 网基线解算得到的 H-文件：

|  |  |
| --- | --- |
| 1 | $ sh\_get\_hfiles -yr 2016 -d 256 -net igsall |

运行命令后，等待程序运行完成，获取到 7 个全球 IGS 网数据处理文件：higs1a.16256.Z ~ higs7a.16256.Z。

获取 IGNI 发布的， 2016 年 5 月 7 日（年积日为 128）全球电离层数据：

|  |  |
| --- | --- |
| 1 | $ sh\_get\_ion -yr 2016 -doy 128 -archive igni |

等待程序运行完成，获得电离层文件：igsg1280.16i.Z。

获取 DAEJ 站自 2016 年 9 月 12 日起共两天的气象观测数据：

|  |  |
| --- | --- |
| 1 | $ sh\_get\_met -yr 2016 -doy 256 -ndays 2 -sites daej |

等待程序运行完成，获得气象数据：daej2560.16m，daej2570.16m。

获取由 CDDIS 发布的，2016 年 9 月 12 日 GPS 系统的广播星历数据：

|  |  |
| --- | --- |
| 1 | $ sh\_get\_nav -archive cddis -yr 2016 -doy 256 |

等待程序运行完成，获得 GPS 系统的广播星历数据：brdc2560.16n.Z。如果你希望下载多系统的混合广播星历数据（brdm），可以为其添加一个 navdir navmgex 参数。

获取 2016 年 5 月 7 日的 GPS 精密星历数据：

|  |  |
| --- | --- |
| 1 | $ sh\_get\_orbits -yr 2016 -doy 128 |

等待程序运行完成，获得精密星历数据：igs18956.sp3。

获取 GLONASS 系统的，2016 年 9 月 12 日的快速星历数据：

|  |  |
| --- | --- |
| 1 | $ sh\_get\_orbits -yr 2016 -doy 256 -pre r -gnss R |

等待程序运行完成，获得精密星历数据：igr19141.sp3。

获取 BJFS 和 WUHN 站点在 2016 年 9 月 12 日的观测数据：

|  |  |
| --- | --- |
| 1 | $ sh\_get\_rinex -yr 2016 -doy 256 -sites bjfs wuhn |

等待程序运行完成，获得观测数据：bjfs2560.16d.Z，wuhn2560.16d.Z。

获取包含所有 IGS 站点的最新 station.info 文件（供 GAMIT 程序使用）：

|  |  |
| --- | --- |
| 1 | $ sh\_get\_stinfo -format new |

等待程序运行完成，获得新版本的最新 station.info 文件。

对于 sh\_get\_raw 命令，运行之后一直提示下载失败，所以暂时没有这个脚本的例子。

以上的脚本仅适用于少量的下载任务，下载大量数据时易出现假死。另外，对于以上所有的命令，都可以通过增加一个参数 -ftp\_prog ncftp 将下载程序由 FTP 切换到 NCFTP，但此程序也需要额外安装。

**编写数据下载脚本**

如果上文提到的脚本命令尚不能满足你的需求，你可以尝试自己编写一个脚本，调用前文所安装的 FTP 程序进行下载。只要你知道目标 FTP 服务器的登录名、密码以及数据所在目录即可。这里以下载 Bernese 数据处理时常用的精密钟差文件为例。

我们知道：精密钟差文件可以在 CDDIS 提供的 FTP 服务器下载，该服务器支持匿名登录；精密钟差文件在 FTP 服务器上的路径为 “/pub/gps/products/” 目录内以 GPS 周命名的文件夹下。因此我们只需用户输入需要下载的文件所对应的 GPS 周以及周内天数。

新建一个文本文件，在其中加入以下内容：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | #!/bin/csh  ######## get IGS\*\*\*\*\*.CLK from cddis.gsfc.nasa.gov ########  echo -n "Input the GPS week(like: 1812): "  set gpsw = $<  echo -n "Input the day of week: "  set dow = $<  set clkname = igs${gpsw}${dow}.clk  echo "get $clkname from cddis.gsfc.nasa.gov ..."  echo "#################################################"  ######## start file download ########  ftp -inv <<EOF  open cddis.gsfc.nasa.gov  user anonymous youremail@163.com  cd pub/gps/products/$gpsw/  mget ${clkname}.Z  quit  EOF |

将该文件保存为 getclk.sh，为其添加可执行权限：

|  |  |
| --- | --- |
| 1 | $ chmod +x getclk.sh |

然后运行这个脚本：

|  |  |
| --- | --- |
| 1 | $ getclk.sh |

根据命令提示输入 GPS 周（如 1812）和周内天数（如 0），待脚本运行完成即可看到其工作成果。

# TEQC 常用的文件编辑命令

[2016-09-19](http://www.gnss.help/2016/09/19/teqc-edit/)

[TEQC](http://www.gnss.help/categories/TEQC/)

[RINEX编辑](http://www.gnss.help/tags/RINEX%E7%BC%96%E8%BE%91/), [TEQC](http://www.gnss.help/tags/TEQC/)

[前文](http://www.gnss.help/2016/09/17/teqc-config-translate/)介绍了 [TEQC](https://www.unavco.org/software/data-processing/teqc/teqc.html) 程序的配置和数据转换功能的使用。本文将专注于介绍该程序的文件编辑功能。

如你所知，作为最常用的 GNSS 数据交换格式，RINEX 对数据组织格式的规定非常严格。况且，迄今为止，已经发布了多个 RINEX 格式版本，版本之间也互有差异。因此手动编辑出一份合乎规范的 RINEX 数据很困难。TEQC 程序提供了编辑 RINEX 数据的功能，使用这些功能可以轻松地提取、合并和编辑 RINEX 数据。特别在处理大量数据时，得益于 TEQC 的非交互特性，还可以很容易地使用脚本程序来操作 TEQC，事半功倍。

## 文件编辑命令

这一部分，将介绍 TEQC 程序编辑 RINEX 文件的操作命令。下面的命令均假设你已经配置好了 TEQC 程序，并且你对该程序的操作模式有一定的了解。如果你对这些知识尚不熟悉，建议首先查看之前的博文：[TEQC 程序配置及其数据转换功能的使用](http://www.gnss.help/2016/09/17/teqc-config-translate/)。

指定 TEQC 程序的输入文件时可以使用通配符，但是考虑到作为示例的命令在 Windows 和 Linux 等平台的通用性，因此示例中只使用了两个系统都支持的通配符。

### 数据提取

有时候你可能希望从已有的 RINEX 文件中抽取某一时间段内的观测数据作为一个新的 RINEX 文件，这涉及到时间窗口的操作。TEQC 对于时间窗口的操作方式为：

|  |  |
| --- | --- |
| 1  2  3 | $ teqc [-st <StartTime>] [-e <EndTime>] <InputFile> # 指定时间窗的起止时刻  $ teqc [-st <StartTime>] +d[hms] <DeltaTime> <InputFile> # 指定开始时刻与持续时间  $ teqc -d[hms] <DeltaTime> [-e <EndTime>] <InputFile> # 指定持续时间与结束时刻 |

这里的 <InputFile> 表示输入文件，[-st <StartTime>]、[-e <EndTime>] 分别表示时间窗口的起始时刻，而 +d[hms] 和 -d[hms] 表示时间窗的持续时间。其中被 “[ ]” 包裹的参数表示该参数是可选的。对于开始时刻，默认为数据的首次观测时刻；对于结束时刻，默认为数据最后观测时刻。命令中指定时刻的方式为：

|  |  |
| --- | --- |
| 1 | [[[[[[YY]YY]MM]dd]hh]mm]ss[.sss...] |

这里的 [YY]YY、MM、dd、hh、mm、ss、.sss 分别表示年、月、日、时、分、秒及秒的小数部分。也就是说，只有整数秒是必需的，其它都可以省略。并且，可以在其中插入 “/” 、“\_” 或 “:” 等作为分隔符以增强可读性。

对于其中的持续时间，可以采用时、分、秒等为单位计算:

|  |  |
| --- | --- |
| 1  2  3 | -dh <number> # 以小时为单位  -dm <number> # 以分钟为单位  -ds <number> # 以秒为单位 |

示例，对于观测时间为 2014 年 2 月 24 日全天的数据 demo0550.14o，提取从该天的 22:30:00 开始一直到观测结束的数据作为一个新的 RINEX 文件 demo0551.14o：

|  |  |
| --- | --- |
| 1 | $ teqc -st 2014/02/24\_22:30:00 demo0550.14o > demo0551.14o |

提取从该天 8:30:00 到 10:30:00 的观测数据作为一个新的 RINEX 文件 demo0552.14o：

|  |  |
| --- | --- |
| 1 | $ teqc -st 8:30:00 -e 10:30:00 demo0550.14o > demo0552.14o |

提取从 7:00:00 开始，持续 1.5 小时的观测数据（即从 7 点到 8 点半）作为 demo0553.14o：

|  |  |
| --- | --- |
| 1 | $ teqc -st 14/2/24\_7:00:00 +dh 1.5 demo0550.14o > demo0553.14o |

提取到 14:30:00 结束，持续时长为 30 分钟的观测数据（即从 14 点到 14 点半）作为 demo0554.14o：

|  |  |
| --- | --- |
| 1 | $ teqc -dm 30 -e 14:30:00 demo0550.14o > demo0554.14o |

### 文件分割

考虑这样一个需求：将文件 demo0550.14o 均匀切割成 12 份，每份包含 2 小时的观测数据。使用上一小节的数据提取命令可以完成这个任务，只是需要进行 12 次时间窗操作。但是 TEQC 还提供了更简单的方式：

|  |  |
| --- | --- |
| 1 | $ teqc -tbin <DeltaTime>[dhms] <Site> <InputFile> |

这里使用 -tbin <DeltaTime>[dhms] 指定观测时长，<Site> 指定观测站名以作为输出文件的前缀。对于这里的观测时长，可以以日、时、分、秒为单位。

示例，将 demo0550.14o 按每 2 小时一个文件分割：

|  |  |
| --- | --- |
| 1 | $ teqc -tbin 2h demo demo0550.14o |

该命令的输出为：

teqc: creating file ‘demo055a.14o’ …  
teqc: creating file ‘demo055c.14o’ …  
teqc: creating file ‘demo055e.14o’ …  
teqc: creating file ‘demo055g.14o’ …  
teqc: creating file ‘demo055i.14o’ …  
teqc: creating file ‘demo055k.14o’ …  
teqc: creating file ‘demo055m.14o’ …  
teqc: creating file ‘demo055o.14o’ …  
teqc: creating file ‘demo055q.14o’ …  
teqc: creating file ‘demo055s.14o’ …  
teqc: creating file ‘demo055u.14o’ …  
teqc: creating file ‘demo055w.14o’ …

查看运行命令时的工作目录，就会发现输出的以上 12 个文件。

### 文件拼接

说过 RINEX 文件的提取与分割，下面介绍它的逆操作：将多个 RINEX 数据文件拼接为一个文件。因为 RINEX 格式的特殊性，不能简单地将 RINEX 文件头尾相接连接到一起。TEQC 程序对于 RINEX 拼接的命令方式为：

|  |  |
| --- | --- |
| 1 | $ teqc [<options>] file1 file2 [file3 […]] > <OutputFile> |

这里的 file1 file2 [file3 […]] 为输入文件列表，<OutputFile> 为输出文件，[<options>] 为可选的控制选项。

示例，将上文分割输出的 12 个文件拼接到一起，形成新的文件 temp0550.14o：

|  |  |
| --- | --- |
| 1 | $ teqc demo055?.14o > temp0550.14o |

如此生成的新文件 temp0550.14o，在拼接接缝处会保留原始文件的注释。如果你不想保留这些注释，可以添加 -phc 参数：

|  |  |
| --- | --- |
| 1 | $ teqc -phc demo055?.14o > temp0550.14o |

需要注意的是，只有观测时间上连续的文件才能拼接到一起。拼接不连续的观测数据会产生错误。

### 采样率抽取

现在的高频接收机，因为观测时数据采样率很高，导致观测文件的体积非常大。例如，对于一个站点采样率为 1Hz 的时长 24 小时的观测数据，能轻松达到 100 MB 以上。但有时我们进行数据处理时并不需要如此高的采样率。

TEQC 程序提供了对高频采样率数据抽稀转换成较低采样率数据的功能。操作方式为：

|  |  |
| --- | --- |
| 1 | $ teqc [-O.int <OldInterval>] -O.dec <NewInterval> <InputFile> > <OutputFile> |

这里的 <OldInterval> 为源文件采样间隔，<NewInterval> 为新文件采样间隔。对于采样间隔在 1 秒及以上的数据文件，源文件的采样间隔总是可以省略。因此处理采样间隔在 1 秒以上的数据时，这样的操作没有问题：

|  |  |
| --- | --- |
| 1 | $ teqc -O.dec 30 demo0550.14o > temp0550.14o |

但是，如果源文件的采样间隔在 1 秒以下时，最好使用 -O.int 参数指定源数据的采样间隔。因为如果输入数据的头文件中没有采样间隔信息的话，处理很可能会出错。如当源文件采样率为 10Hz 时：

|  |  |
| --- | --- |
| 1 | $ teqc -O.int 0.1 -O.dec 30 demo0550.14o > temp0550.14o |

### 观测卫星筛选

TEQC 程序可以对数据中的观测卫星进行编辑，如删除某一卫星的观测数据，甚至还可以一次性的去掉某一卫星系统的所有观测量。

要去除某些卫星的观测数据，只需在 “-” 号后跟上要去掉的卫星列表，多个卫星之间以逗号连接。下面的命令用于删除 GPS 卫星 G1 与 G5 的观测数据：

|  |  |
| --- | --- |
| 1 | $ teqc -G1,5 demo0550.14o > temp0550.14o |

执行该命令后，检查生成的新文件 temp0550.14o，发现 G01 和 G05 的观测数据被删除了。

对于连续的卫星列表，还可以用 “-” 指定起止卫星号。下面的命令将删除 GPS 卫星 G1 到 G5 的观测数据：

|  |  |
| --- | --- |
| 1 | $ teqc -G1-5 demo0550.14o > temp0550.14o |

执行该命令后，检查生成的新文件 temp0550.14o，发现 G01、G02、G03、G04、G05 的观测数据都被删除了。

如果你希望删除或保留某一卫星系统的数据，也可以直接对卫星系统进行操作。下面的命令将去除所有 Galileo 卫星：

|  |  |
| --- | --- |
| 1 | $ teqc -E demo0550.14o > temp0550.14o |

需要注意的是，截至本文撰写时，TEQC 程序并不能正确处理北斗系统（BDS）的卫星。因此所有经过 TEQC 程序处理的观测数据都将丢失北斗卫星的观测值。

### 观测量编辑

TEQC 程序还可以单独对观测量进行编辑，多个观测量列表可以用逗号分隔，也可以不加间隔符号。但我建议使用逗号进行分隔，这样看起来更清晰。对观测量进行选择时，既可以使用 -O.-obs 指定要删除的观测量，也可以使用 -O.obs 指定要保留的观测量。

下面的命令将观测量 L1、L2、P1、P2、C1、C2 删除：

|  |  |
| --- | --- |
| 1 | $ teqc -O.-obs L1,L2,P1,P2,C1,C2 demo0550.14o > temp0550.14o |

与上面的命令相反，下面的操作将只保留观测数据中的 L1、L2、P1、P2、C1、C2 观测量：

|  |  |
| --- | --- |
| 1 | $ teqc -O.obs L1,L2,P1,P2,C1,C2 demo0550.14o > temp0550.14o |

### 信息编辑

这里的信息编辑指对 RINEX 数据文件头（Header）中的信息进行添加、编辑、替换等操作。使用 TEQC 程序可以对 RINEX 格式的观测文件（Obs）、导航文件（Nav）和气象文件（Met）中的信息进行编辑。

查看一个 RINEX 文件的头文件信息使用 +meta 参数，命令如下：

|  |  |
| --- | --- |
| 1 | $ teqc +meta <InputFile> |

该命令将输出输入数据的头文件信息。也可以使用 ++config 参数：

|  |  |
| --- | --- |
| 1 | $ teqc ++config <InputFile> |

该命令也能输出输入数据的头文件信息，但是其形式与上一个有些不同，你可以都试一试。

使用 TEQC 程序编辑头文件信息的命令格式如下：

|  |  |
| --- | --- |
| 1 | $ teqc <options> <InputFile> > <OutputFile> |

对于观测文件（Obs），常用的参数有（完整的信息编辑参数请查看文末整理出的列表）：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | -O.mo <Monument> # 标志名  -O.ag <Agency> # 观测机构  -O.rt <Receiver Type> # 接收机类型  -O.at <Antenna Type> # 天线类型  -O.px <X> <Y> <Z> # 观测点的概略坐标（以米为单位）  -O.pe <H> <E> <N> # 天线的偏心改正（hEN，以米为单位） |

示例，编辑 demo0550.14o，将观测机构修改为 “SDUST”，并形成新文件 temp0550.14o：

|  |  |
| --- | --- |
| 1 | $ teqc -O.ag "SDUST" demo0550.14o > temp0550.14o |

这里的 “SDUST” 也可以不加双引号。但如果字符串中有空格，就必须使用双引号包裹。例如，编辑文件 temp0550.14o，将接收机类型修改为 “TPS NETG3”，将天线类型修改为 “TPSCR.G3　　　TPSH”，并形成新文件 demo0550.14o：

|  |  |
| --- | --- |
| 1 | $ teqc -O.rt "TPS NETG3" -O.at "TPSCR.G3 TPSH" temp0550.14o > demo0550.14o |

如果你有很多项目和文件需要编辑，可以考虑将这些修改项写入一个文件。例如创建一个文件 meta.txt，在其中输入：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | -O.ag "SDUST"  -O.mo "DEMO"  -O.rt "TPS NETG3"  -O.at "TPSCR.G3 TPSH"  -O.pe 0.5300 0.0000 0.0000  -O.px -2592140.0000 4468620.0000 3728260.0000 |

然后使用 -config 参数来读取该文件：

|  |  |
| --- | --- |
| 1 | $ teqc -config meta.txt demo0550.14o > temp0550.14o |

这样就可以同时将 meta.txt 文件中的所有项目进行修改。最后，如果你希望使用 TEQC 程序对大量的数据进行格式标准化操作，可以查看一下本站 [GNSS 数据批量标准化脚本](http://www.gnss.help/2017/01/27/pinot-unificate/)。

## 完整的信息编辑参数列表

### Obs 文件信息编辑参数

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | -O.ag[ency] # 观测机构  -O.an # 天线号  -O.at # 天线类型  -O.c[omment] # 原头文件注释  +O.c[omment] # 添加新的注释  -O.d[ate] # 程序执行日期  -O.dec[imate] # 以指定的采样间隔进行数据抽取  -O.def\_wf # 缺省的L1,L2载波的波长因子  -O.e[nd] # 最后观测的日期和时间  -O.int[erval,sec] # 采样间隔，单位秒  -O.mn # 标志号  -O.mo[nument] # 标志名  -O.mod\_wf # 从不同的默认的波长因子中，为特定的卫星设置波长因子  -O.mov[ing] # 强制将天线位置设置为最初的运动状态  -O.o[perator] # 观测者  -O.-obs[\_types] # 在输出文件中删除 \_types 指定的观测值列表  -O.obs[\_types] # 在输出文件中只保留 \_types 指定的观测值列表  -O.pe[hEN,m] # 天线中心相对于地面点的偏心改正, 单位米  -O.pg[eo,ddm] # 标志的近似坐标（WGS-84），经度和纬度用小数表示，高层用米表示  -O.pr[ogram] # 生成 RINEX 文件的程序  -O.px[WGS84xyz,m] # 标志的近似坐标（WGS-84）, 单位米  -O.r[un\_by] # 运行程序的用户名  -O.rename\_obs # 改变 # / TYPES OF OBSERV 中观测值类型的性质  -O.rn # 接收机号  -O.rt # 接收机类型  -O.rv # 接收机软件的版本  -O.s[ystem] # 卫星系统  -O.sl[ant] # 以地面点为中心的垂直天线改正、斜高、天线直径和垂直相位中心偏差  -O.st[art] # 开始观测的日期和时间 |

### Nav 文件信息编辑参数

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | -N.a[alpha] # 电离层 α 参数  -N.b[eta] # 电离层 β 参数  -N.c[omment] # 原头文件注释  +N.c[omment] # 增加新的注释  -N.d[ate] # 程序执行日期  -N.leap # UTC 时间模型的跳秒  -N.pr[ogram] # 生成 RINEX 文件的程序  -N.r[un\_by] # 运行程序的用户名  -N.s[ystem] # 卫星系统  -N.UTC # UTC 时间模型 |

### Met 文件信息编辑参数

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | -M.c[omment] # 原头文件注释  +M.c[omment] # 增加新的注释  -M.d[ate] # 程序执行日期  -M.mo[nument] # 观测点名  -M.mn # 观测点号  -M.obs[\_types] # 在文件的数据部分，列出气象观测值类型列表  -M.pr[ogram] # 生成 RINEX 文件的程序  -M.r[un\_by] # 运行程序的名字 |

**TEQC 配置及其数据转换功能的使用**

[2016-09-17](http://www.gnss.help/2016/09/17/teqc-config-translate/)

[TEQC](http://www.gnss.help/categories/TEQC/)

[RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/), [TEQC](http://www.gnss.help/tags/TEQC/)

[TEQC（Translation, Editing, Quality Check）](https://www.unavco.org/software/data-processing/teqc/teqc.html)是最常用的 GNSS 数据预处理程序之一。该程序由 [UNAVCO](https://www.unavco.org/) 开发，支持 UNIX/Linux、Windows 等常见的操作系统，功能强大，使用方便。

TEQC 程序提供了对 RINEX 格式的观测文件、星历文件和气象文件等的格式转换、内容编辑和质量检核等功能。但是，和很多的科学软件一样，TEQC 没有图形操作界面，对初学者可能不太友好。本文将以演示的方式，介绍 TEQC 程序的配置与数据转换命令，以图在不查看程序说明手册的情况下也能照葫芦画瓢，快速上手。当然，如果你希望详细系统地了解该程序，还是建议花时间阅读其[官方文档](https://www.unavco.org/software/data-processing/teqc/doc/UNAVCO_Teqc_Tutorial.pdf)。

**环境配置**

TEQC 程序目前支持 Windows、UNIX/Linux 等众多操作系统。要获取 TEQC 程序，直接到其[官方下载页面](https://www.unavco.org/software/data-processing/teqc/teqc.html)下载对应系统的软件。

**Windows 操作系统**

对于 Windows 操作系统，从程序压缩包解压出可执行文件就能直接使用。只要打开 “命令提示符” 窗口，使用 cd 命令进入程序所在目录，然后运行 teqc 命令。

如果你不想每次使用 TEQC 前都切换工作目录，可以将程序可执行文件添加到系统的 Path 变量中，或者直接将其移动至 “C:\Windows\System32” 文件夹内。

**UNIX/Linux 操作系统**

对于 UNIX/Linux 操作系统，程序有采用静态链接库（statically-linked）和动态链接库（dynamically-linked）两种，一般下载动态链接库的程序包。下载程序压缩包解压出可执行文件之后，还需为程序添加可执行权限。使用 cd 命令进入程序所在目录，执行命令：

|  |  |
| --- | --- |
| 1 | $ chmod +x teqc |

之后便可以在终端中运行此程序，但是每次使用时也需要首先切换工作目录。如果你希望在任何目录下都能使用该程序，可以将其移动到 “/usr/bin” 目录下。

**操作模式**

为了便于理解和仿制后面的命令，首先介绍一下 TEQC 程序的基本操作模式。使用 TEQC 程序的一般的语法是：

|  |  |
| --- | --- |
| 1 | $ teqc <options> [file1 [file2 […]]] |

这里的 [file1 [file2 […]]] 表示输入文件列表，而 <options> 为控制参数。控制参数一般以 “+” 或 “-” 号开头：对于 “-” 号，一般指示输入命令（包括标准输入或文件），或表示关掉这个选项；对于 “+” 号，一般指示输出命令（包括标准输出或文件），或表示打开这个选项。参数后面则是希望被处理的文件，多个文件之间一般以空格分隔。

TEQC 是百分之百非交互的，也就是说，它不会在运行时停下来询问你。对于处理结果，一般会输出到标准输出或文件；对于处理时出现的错误或警告信息，则输出到标准错误输出。这个特性对于编写基于 TEQC 的脚本程序非常有利。

**格式操作命令**

**格式检验**

TEQC 程序可以检查一个程序是否为正确的 RINEX 格式文件，并输出文件的 RINEX 版本编号。该功能使用 +v 参数开启，其命令为：

|  |  |
| --- | --- |
| 1 | $ teqc +v file1 [file2 […]] |

示例，检查 urum2560.16m，urum2560.16o， brdc2560.16n 和 DEMO0224a.tps 这 4 个文件：

|  |  |
| --- | --- |
| 1 | $ teqc +v urum2560.16m urum2560.16o brdc2560.16n DEMO0224a.tps |

输出为：

teqc: ‘urum2560.16m’ readable as RINEX V.2.11 format  
teqc: ‘urum2560.16o’ readable as RINEX V.2.11 format  
teqc: ‘brdc2560.16n’ readable as RINEX V.2.00 format  
teqc: ‘DEMO0224a.tps’ readable as Topcon TPS format

**格式转换**

这里的格式转换一般指从接收机的原始观测数据到 RINEX 格式数据的转换。目前 TEQC 在这方面的支持比较有限，仅支持 Trimble、Topcon、Leica、Javad、Septentrio 等国际大厂的仪器。对于不受支持的接收机品牌或型号，只能使用随机附带的软件。

TEQC 程序的格式转化的基本命令格式为：

|  |  |
| --- | --- |
| 1 | $ teqc <options> raw\_file1 [raw\_file2 […]] > <RINEX\_file> |

这里的 raw\_file1 [raw\_file2 […]] 表示输入的原始数据列表。对于存储在本地的原始数据文件，大多数情况下 TEQC 程序通过文件名就能识别出其接收机品牌和格式类型。但对于数据流，或者某些无法自动识别的数据文件格式，则需要使用 <options> 参数来指明原始数据的接收机品牌或格式类型。完整的接收机参数请查看本文最后给出的汇总表。

示例，转换 Topcon 接收机在 2014 年 2 月 24 日观测产生的 .tps 文件（这里的 -top tps 可以省略）：

|  |  |
| --- | --- |
| 1 | $ teqc -top tps DEMO0224a.tps > demo0551.14o # 对应年积日为 55，GPS 周为 1781 |

运行这个命令后，将产生一个转换后的文件 demo0551.14o。同时，TEQC 还会输出一条警告：

! Notice ! GPS week initially set= 1781

这是因为，我们在进行数据转换时没有指出数据观测的日期，TEQC 将试图从原始数据中读取观测日期。这在少数的情况下可能会出现错误。如果你想明确指定数据的观测日期，可以使用 -week 参数：

|  |  |
| --- | --- |
| 1  2  3 | -week <week> # 输入观测 GPS 周  -week <year>/<doy> # 输入观测年与年积日  -week <year>/<month>/<day> # 输入观测年月日 |

示例，输入两个观测日期进行格式转换，同时将输出的文件合并：

|  |  |
| --- | --- |
| 1 | $ teqc -week 1781 DEMO0224b.tps DEMO0224c.tps > demo0552.14o |

对于更多的输入文件，还可以使用通配符：

|  |  |
| --- | --- |
| 1 | $ teqc -week 2014/55 DEMO0224?.tps > demo0550.14o |

到目前为止的转换命令只输出了 RINEX 格式的 Obs 文件，因为 TEQC 程序默认只输出这个文件。如果你还希望输出 Nav 文件和 Met 文件（如果有的话），可以为其添加 +nav 或 +met 参数。

对于 “+met” 参数，在其后指定输出的 Met 文件名即可。对于 +nav 参数，因为有多个 GNSS 系统，所以卫星星历的输出有以下顺序：

1. GPS
2. GLONASS
3. SBAS
4. Galileo
5. Beidou/Compass
6. QZSS

对于支持多系统的接收机，要输出多个卫星系统的星历时，按照该顺序依次指定输出文件名，其间以 “,” 号分隔。对于要跳过的系统，以 “-” 标识。

示例，转换时输出 GPS 星历数据和气象数据：

|  |  |
| --- | --- |
| 1 | $ teqc -week 2014/2/24 +nav demo0550.14n +met demo0550.14m DEMO0224?.tps > demo0550.14o |

转换时分别输出 GPS 和 GLONASS 星历数据：

|  |  |
| --- | --- |
| 1 | $ teqc -week 2014/2/24 +nav demo0550.14n,demo0550.14g DEMO0224?.tps > demo0550.14o |

转换时跳过 GPS 星历数据，只输出 GLONASS 星历数据：

|  |  |
| --- | --- |
| 1 | $ teqc -week 2014/2/24 +nav -,demo0550.14g demo0224?.tps > demo0550.14o |

最后，如果你不喜欢使用重定向符 “>”，类似于上文的 +nav 和 +met，还有一个输出 Obs 文件的 +obs 参数：

|  |  |
| --- | --- |
| 1 | $ teqc -week 2014/2/24 +nav -,demo0550.14g +obs demo0550.14o demo0224?.tps |

**完整的接收机参数表**

**各接收机生产商制定的数据格式**

本表中，对于相同功能的参数如 -­trimble 与 -tr、-aoa 与 -jpl 等做了合并处理，删去其中较长或字母表顺序靠后的项。在格式转换中必须指明的项，在行末以 “\*” 号标记。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | -aoa # for TurboRogue/TurboStar or Benchmark receiver  cb # Conan 二进制文件  tb # Turbo 二进制文件    -ash # for Ashtech  d # B[/E/S/D] 文件格式  s # RS-232 数据流 \*  r # R-文件  u # U-文件 \*    -cmc # for CanadianMarconi Corporation  allstar # allstar 文件格式    -­jav # for Javad  jps # JPS 文件格式    -lei # for leica  lb2 # LB2 omdb  mdb # MDB  d # DSdownload fileset (OBSfile required)    -nct # for Navcom  b # Navcom 二进制文件    -roc # for Rockwell  z # Zodiac 文件    -sep # for Septentrio  sbf # Septentrio 二进制文件    -ti # for Texas Instruments  g # TI-4100GESAR andBEPP/CORE 文件  rom # TI-4100 ROM 文件格式    -top # for Topcon  tps # tps 文件格式    -tr # for Trimble  d # DAT[/ION/EPH/MES] 文件  s # RS-232 RT17 数据流 \*  tsip # TSIP 格式    -ubl # for U-Blox  ubx # UBX 格式 |

**非接收机厂商制定的特殊数据格式**

这些格式只有参数没有选项。在格式转换中必须指明的项，在行末以 “\*” 号标记。

|  |  |
| --- | --- |
| 1  2  3  4 | -argo # ARGO  -binex # BINEX 文件  -rtigs # IGS 实时数据流 \*  -soc # JPL SOC \* |

# 为 GAMIT 配置自动发邮件功能

[2016-09-13](http://www.gnss.help/2016/09/13/config-mailx/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/), [LINUX](http://www.gnss.help/tags/Linux/)

使用 GAMIT 进行批处理基线解算时，可以设置在数据处理完成之后，将数据处理的主要指标通过电子邮件的形式发送到指定的邮箱地址。这在处理需大量耗时的数据时非常有用，但是很多用户都表示并没有收到过邮件。这是因为：GAMIT 需要调用 UNIX/Linux 系统的 mail 或 mailx 程序来实现发邮件的功能，你很可能没有正确配置这个程序。

本文中，我将以 CentOS 和 Ubuntu 两种常用的 Linux 发行版为例，介绍为 GAMIT 程序配置自动发送邮件功能的过程。读者可以作为自己配置时的参考。

## 配置 mailx 命令

### 安装 mailx

首先应该测试你的计算机有没有安装 mailx 程序，可以在终端中尝试运行命令：

|  |  |
| --- | --- |
| 1 | $ mailx |

如果提示 “command not found”，那么说明系统尚未安装该程序。对于 Fedora、CentOS 等相似发行版，可以采用如下命令安装：

|  |  |
| --- | --- |
| 1 | $ sudo yum install mailx |

对于 Debian、Ubuntu 等，则采用以下命令：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install heirloom-mailx |

### 配置 mailx

用 mailx 程序发送邮件的操作过程是这样的，首先输入命令：

|  |  |
| --- | --- |
| 1 | $ mailx -s <Subject> <Address@mail.com> |

这里的 <Subject> 和 <Address@mail.com> 分别代表邮件的主题和递送地址。键入回车键之后开始输入邮件内容，最后按组合键 CTRL + D 结束输入并发送。

理论上，成功安装 mailx 程序后就能用上述命令来发送邮件了，你可以给自己发送一封邮件试一下。但实际上很可能会发生递送错误。这是因为：目前大部分的垃圾邮件过滤机制都会拒收这样的邮件。

为了使 mailx 程序可以正确发送邮件，我们一般采用外部的 SMTP 发送邮件。简言之，就是为 mailx 设置一个公共邮箱服务，使其通过这个公共邮箱来发送邮件。我建议采用网易的邮件服务，配置起来最方便。

到[网易邮箱注册页面](http://reg.email.163.com/unireg/call.do?cmd=register.entrance&from=email163&regPage=163)注册一个邮箱地址，然后将该地址设置到 mailx 程序。这里以 “example@163.com” 邮箱为例，其密码假设为 “password”。

然后就是编辑 mailx 程序的配置文件了。对于 Fedora、CentOS 等发行版，打开 **/etc/mail.rc**；对于 Debian、Ubuntu 等，则打开 **/etc/nail.rc**。在文件最后添加如下内容（注意替换成你注册的邮箱和密码）：

|  |  |
| --- | --- |
| 1  2  3  4 | # set for 163 smtp  set from = example@163.com smtp = smtp.163.com  set smtp-auth-user = MyLinuxPC smtp-auth-password = password  set smtp-auth = login |

如此配置之后，别忘了重新运行 mailx 程序给自己发送邮件试一下。邮件很可能出现在垃圾信箱中，为了使该邮箱地址以后不会被当作垃圾邮件源，建议将其添加到你的邮箱联系人列表中。

## 配置 GAMIT

配置好 mailx 之后，接下来就是将你的收件地址加入到 GAMIT 的配置文件了。为 GAMIT 批处理命令设置自动发送邮件的选项在 process.defaults 文件中，打开该文件可以看到其中类似的设置项目：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # UNIX mail command  # Most machines  set umail = 'mail -s'  # HP  # set umail = 'mailx -s'  # Mail address for sending the processing report (if '' will default to `whoami` in sh\_gamit)  set mailto = '' |

该文件中，以字符 “#” 开头的行表示注释。从上段内容中可以看到，目前程序发送邮件时采用的命令是 mail -s，我们可以修改为采用 mailx -s 命令，并且设置 mailto 为你的邮箱地址（注意这里是你接收邮件的地址，而不是上文设置的发送邮件的地址）。此处以 “example@mail.com” 为例，修改配置后如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # UNIX mail command  # Most machines  # set umail = 'mail -s'  # HP  set umail = 'mailx -s'  # Mail address for sending the processing report (if '' will default to `whoami` in sh\_gamit)  set mailto = 'example@mail.com' |

如此设置后，每当数据处理完成时，sh\_gamit 命令将调用 mailx -s 命令向你设置的收件地址发送电子邮件。快去试一下吧！

**GLOBK 批处理网平差**

[2016-09-10](http://www.gnss.help/2016/09/10/sh-glred/)

[GLOBK](http://www.gnss.help/categories/GLOBK/)

[GLOBK使用](http://www.gnss.help/tags/GLOBK%E4%BD%BF%E7%94%A8/)

学习过 GLOBK 程序的分步操作之后，自然会想到使用批处理命令来节约操作和时间。GLOBK 程序的批处理操作主要使用 sh\_glred 命令实现。该命令封装了 htoglb、glred、globk、glorg 等程序，通过一次操作，可以分别输出自由网平差和约束网平差等的解算结果，非常方便。与 sh\_gamit 类似，sh\_glred 命令也由 C Shell 脚本语言编写，该脚本保存在 GAMIT/GLOBK 程序安装目录的 com 文件夹中。你可以通过阅读其代码来加深对 GLOBK 程序数据处理流程的理解。

本文将介绍使用 GLOBK 程序的批处理操作进行网平差的流程。

**文件组织**

使用 GLOBK 程序进行批处理数据操作时，我们通常这样组织文件：

|  |  |
| --- | --- |
| 1  2  3  4  5 | · work/  |—— <doy>/  |—— glbf/  |—— gsoln/  |—— tables/ |

在上面的文件结构中，首先依然是一个任意命名的工程目录，这里还是以 “work” 为例。工作目录下平行地创建 4 个文件夹。其中，年积日文件夹内须有输入 GLOBK 程序的 H-文件，glbf/ 内保存转化出二进制的 H-文件。程序在 gsoln/ 文件夹内进行数据处理，而 tables/ 文件夹内依旧是解算时需要引入的共用表文件。

首先说明：上面只列出必需的文件夹。因为我们经常在执行 GAMIT 批处理时所用的文件目录下继续运行 GLOBK 程序的批处理操作来得到测站坐标和速度，所以，沿用 sh\_gamit 命令为我们创建的目录结构完全没有问题。sh\_glred 在工程目录下运行，这一点与 sh\_gamit 命令类似，但其数据处理成果却保存在 gsoln/ 文件夹内。

本文以[之前基线解算的例子](http://www.gnss.help/2016/09/01/sh-gamit/)处理过的 2016 年 8 月 17 日的 6 个 IGS 测站（BJFS，CHAN，DAEJ，SHAO，SUWN，WUHN 等）的基线结果为例，在此基础上继续进行 GLOBK 网平差操作。因为我之前使用 GAMIT 程序的批处理命令 sh\_gamit进行了基线解算处理，该命令已经为我们创建了所需的文件组织结构：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | · work/  |—— 230/  |—— htesta.16230  ...  |—— glbf/  |—— gsoln/  |—— tables/  ... |

工程目录下有很多文件夹，但我们本次要提到的只有以上明确列出的 4 个，其他的以 “…” 表示。另外，基线解算处理完成后，年积日文件夹和 tables/ 文件夹下也有很多的文件，但这里不再列出。

**参数配置**

在 GAMIT/GLOBK 10.7 之前， GLOBK 程序所使用的控制文件是 globk\_comb.cmd 和 glorg\_comb.cmd，之后的版本中是 globk.cmd 和 glorg.cmd。你可以通过它们来控制网平差的整个过程，当然也包括所使用的固定点和已知点坐标。在较旧的版本中，固定点也可以直接在 glorg\_comb.cmd 或者 sites.default 文件中配置，但是这种办法已经不再推荐。更好的方法是在两个 .cmd 文件中设置解算过程控制参数，在 .apr 文件中输入已知点坐标，在 .stab\_site 文件中设置固定点。这是一种数据与配置分离的思想，可以最大程度地提高控制文件和数据的重用性。

你可以使用本博客[已知点坐标文件的创建与使用](http://www.gnss.help/2017/03/24/create-apr/)一文介绍的办法创建自己的 .apr 文件，并将其应用到网平差处理过程中。但本次任务我处理的是 IGS 站的观测数据，它们的坐标早已存在于默认的 .apr 文件中，因此沿用默认配置即可。我需要做的是通过编辑自己的 .stab\_site 文件来设置固定点。该文件的格式如下：

|  |  |
| --- | --- |
| 1  2 | # This is a .stab\_site file for demo  stab\_site <name> |

这里的 <site> 表示固定点的点名。一个配置好的文件示例如下：

|  |  |
| --- | --- |
| 1  2  3  4 | # This is a .stab\_site file for demo  stab\_site BJFS  stab\_site CHAN  stab\_site SHAO |

接下来的工作就是配置控制解算过程的 .cmd 文件。如果 gsoln/ 文件夹中还没有这两个文件，你可以在工程目录中运行以下命令将获取它们：

|  |  |
| --- | --- |
| 1 | $ sh\_glred -cmd |

以上命令将拷贝两个合适的配置文件放置到你的 gsoln/ 文件夹内，你可以编辑它们来控制网平差过程。在此类文件中，以字符 “\*”、“#” 或 “x” 开头的行表示注释，而控制项的第一个字符为空格。在 globk\_comb.cmd 中，类似以下的内容用于设置已知点坐标文件：

|  |  |
| --- | --- |
| 1  2  3  4  5 | \* ITRF2008 augmented by now-defunct sites and recent IGS solutions;  # matched to itrf08\_comb.eq  apr\_file ~/gg/tables/itrf08\_comb.apr  # Optionally add additional apr files for other sites  x apr\_file ../../tables/regional.apr |

从这里可以看出，默认的已知点坐标文件为 GAMIT/GLOBK 程序安装目录 tables/ 文件夹内的 itrf08\_comb.apr。这里面包含了 IGS 核心站的坐标和站速度。

在 glorg\_comb.cmd 中，可以看到类似以下的内容：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | \* A priori coordinates  # ITRF2008 may be replaced by an apr file from a priori velocity solution  apr\_file ~/gg/tables/itrf08\_comb.apr  x apr\_file ../../tables/regional.apr  \* List of stabilization sites  # This should match the well-determined sites in the apr\_file  stab\_site clear  source ~/gg/tables/igb08\_hierarchy.stab\_site  x source ../../tables/regional\_stab\_site |

这里的上半部分也是已知点坐标的来源文件，下半部分用于指定固定点配置文件。这里的默认配置同样为 GAMIT/GLOBK 安装目录 tables/ 文件夹下的文件。但这次我需要应用自己的配置，因此将这部分修改为：

|  |  |
| --- | --- |
| 1  2  3  4  5 | \* List of stabilization sites  # This should match the well-determined sites in the apr\_file  stab\_site clear  x source ~/gg/tables/igb08\_hierarchy.stab\_site  source ../tables/my\_sites.stab\_site |

这里指定网平差中使用解算工程目录 tables/ 文件夹内的 my\_sites.stab\_site 文件中的固定点。

**开始解算**

现在已经可以运行 sh\_glred 命令进行网平差。对于生成单天解，这个命令的常用参数如下：

|  |  |
| --- | --- |
| 1 | $ sh\_glred -expt <expt> -d <year> <doy> -opt <opts> |

这里的 <opts> 表示网平差时的控制选项，它共有以下几个可选（多个控制项以空格分隔）：

* R ： 在操作之前把旧的二进制 H-文件从 glbf 目录中删除；
* F ： 根据 <networks> 参数，通过 FTP 下载全球 IGS 站网的 H-文件进行联合解算；
* H ： 对现有的或者链接到 glbf 的文本 H-文件运行 htoglb；
* LA ： 链接所有本地的文本 H-文件到组合中；
* LB ： 链接所有本地的文本 H-文件到组合中，从 htnd 开始向下搜索，其中 htnd 是系统默认的；
* LC ： 链接本地的二进制 H-文件；
* U ： 对二进制 H-文件运行 hfupd；
* G ： 运行 glred 以进行组合或重新组合；
* E ： 运行 ensum 和 sh\_baseline 进行绘图；
* C ： 解算后压缩文本 H-文件，删除所有的链接并整理 H-文件；
* A ： 选中以上所有项。

对于观测日期上连续的数据，类似于 sh\_gamit，还有一种指定数据起始日期的方式：

|  |  |
| --- | --- |
| 1 | $ sh\_glred -expt <expt> -s <yr1> <d1> <yr2> <d2> -opt <opts> |

如果你希望得到多天解，还可以为该命令增加一个参数指定合并的天数：

|  |  |
| --- | --- |
| 1 | $ sh\_glred -expt <expt> -s <yr1> <d1> <yr2> <d2> -ncomb <days> -opt <opts> # <days> 指点合并的天数 |

具体到本次任务，只需要运行一个命令：

|  |  |
| --- | --- |
| 1 | $ sh\_glred -expt test -d 2016 230 -opt H G |

**成果查看**

GLOBK 批处理网平差完成后，解算结果保存在 gsoln/ 文件夹下的 .org 结果文件中。如果你设置了在解算后绘图，可能还会看到 .ps 格式的站点重复性图，不过这需要有 [GMT](http://gmt.soest.hawaii.edu/) 程序的支持。

如果解算过程中出现错误，与分步操作相同，错误信息保存在 gsoln 目录内的 GLOBK.fatal 文件内。

**GLOBK 分步进行网平差**

[2016-09-07](http://www.gnss.help/2016/09/07/GLOBK-StepbyStep/)

[GLOBK](http://www.gnss.help/categories/GLOBK/)

[GLOBK使用](http://www.gnss.help/tags/GLOBK%E4%BD%BF%E7%94%A8/)

GLOBK 是 GAMIT/GLOBK 程序的另一个主要组件。该程序使用卡尔曼滤波的方式对 GAMIT、GIPSY 和 Bernese 等产出的结果及经典大地测量和 SLR 观测数据进行处理，最后得到观测点的坐标、站速度、EOP 参数和卫星的轨道参数等。

与 GAMIT 操作类似，GLOBK 程序的操作也可以分为分步处理和批处理两种方式。本文将试图介绍使用 GLOBK 程序对 GAMIT 基线解算结果以分步处理的方式进行网平差的过程。

**程序简介**

GLOBK 程序的主要功能包括：

* 将单天解合并为多天解；
* 将多天解合并为多年解，并估计测站速度；
* 估计测站坐标重复性，进而评价多天观测解的精度。

与 GAMIT 程序类似，GLOBK 也由多个模块组成。其主要功能模块包括 htoglb、glred、globk、glorg，各自的功能如下：

* htoglb：将 GAMIT 产生的文本 H-文件转换为输入 GLOBK 的二进制 H-文件。
* glred：合并输入文件并计算单天解的重复性。
* globk：估计测站的坐标和站速度。
* glorg：计算添加约束的联合解。

**文件组织**

GLOBK 程序的输入文件是由 GAMIT 进行基线解算得到的 H-文件。在使用 GLOBK 进行网平差解算时，我们习惯于这样组织文件：

|  |  |
| --- | --- |
| 1  2  3  4 | · work/  |—— glbf/  |—— gsoln/  |—— tables/ |

首先创建一个文件夹作为工作目录，该目录的命名可以是任意的。在工作目录下平行地创建上面的 3 个文件夹。其中在 gsoln/ 存放 GAMIT 基线解算结果文件并在文件夹中进行网平差操作，glbf/ 文件夹存放二进制的 H-文件，tables/ 文件夹存放解算时所需的一些配置文件和解算需引入的共用表文件。

当然，上面只是列出了必需的两个文件夹。如果你在之前进行 GAMIT 基线解算的工作目录下继续进行网平差操作，这时很可能 sh\_gamit 命令已经为你创建了 gsoln 等文件夹，这时直接应用已有的目录结构即可。

将 GAMIT 解算结果的 H-文件拷贝到 gsoln/ 文件夹中。以前文 [GAMIT 分步基线解算](http://www.gnss.help/2016/08/27/GAMIT-StepbyStep/)的解算结果作为示例，具体的文件结构应为：

|  |  |
| --- | --- |
| 1  2  3  4  5 | · work/  |—— glbf/  |—— gsoln/  |—— htesta.16050  |—— tables/ |

**参数配置**

**链接表文件**

如果你在之前使用 GAMIT 程序进行基线解算的工作目录下继续进行网平差操作，那么不需要再链接共用表文件，因为基线解算时已经链接过了。但如果你在新建的工作目录下操作，需要在工作目录下执行命令：

|  |  |
| --- | --- |
| 1 | $ sh\_setup -yr 2016 -doy 050 |

这个命令在之前介绍[基线解算的博文](http://www.gnss.help/2016/08/27/GAMIT-StepbyStep/)中已经出现过多次，本文中不做解释。

当然，如果你处理的 H-文件是在其他的计算机上解算得到的，在链接共用表文件之前也要先检查某些文件是否需要更新。具体的细节之前介绍 [GAMIT/GLOBK 更新方法](http://www.gnss.help/2016/08/29/UpdateGAMIT/)的博文中也已经讲过，这里不再赘述。

**参数设置**

GLOBK 程序分步网平差的控制文件通常使用这两个：globk\_comb.cmd 和 glorg\_comb.cmd。这两个文件保存在 GAMIT/GLOBK 安装目录的 tables 文件夹内，你可以将其拷贝到工作目录的 gsoln 文件夹中，也可以在工作目录下使用命令：

|  |  |
| --- | --- |
| 1 | $ sh\_glred -cmd |

这个命令将帮你拷贝两个控制文件至 gsoln 文件夹内。

在此类控制文件中，以字符 “\*”、“#” 或 “x” 开头的行表示注释，而控制项的第一个字符为空格。浏览 globk\_comb.cmd 文件，你将在其中看到已知点坐标来源、地震记录文件来源等配置项，我的控制文件中这两项如下：

eq\_file ~/gg/tables/itrf08\_comb.eq  
apr\_file ~/gg/tables/itrf08\_comb.apr

这表明解算时将使用 GAMIT/GLOBK 安装目录的 tables 文件夹下 ITRF2008 框架下的 IGS 站坐标作为已知点坐标，而地震信息从 ITRF2008 框架的地震记录文件中读取。如果你需要使用其它文件作为这两项的输入，直接修改其后的文件路径即可。为简单起见，我决定对这两项使用默认配置。

继续向下浏览，你将看到对网平差时各站点的约束配置，其内容大概是这样：

\* Coordinate parameters to be estimated and a priori constraints  
apr\_neu all 10 10 10 0 0 0

在本次网平差任务中，我计划以 BJFS、CHAN 和 SHAO 这 3 个站点作为固定站，其三方向（NEU）约束设置为 0.05 m，0.05 m，0.1 m。因此需要编辑此处配置为如下这样（注意保持非注释行第一列为空格）：

\* Coordinate parameters to be estimated and a priori constraints  
apr\_neu all 10 10 10 0 0 0  
apr\_neu bjfs 0.05 0.05 0.1 0 0 0  
apr\_neu chan 0.05 0.05 0.1 0 0 0  
apr\_neu shao 0.05 0.05 0.1 0 0 0

再浏览 glorg\_comb.cmd 文件，会发现这里也有一些类似的配置项。找到其中对固定站的配置，其内容像这样：

# This should match the well-determined sites in the apr\_file  
stab\_site clear  
source ~/gg/tables/igb08\_hierarchy.stab\_site

这里的第一行为注释，第二行将之前配置的固定站列表清空，第三行从 GAMIT/GLOBK 安装目录的 tables 文件夹中加载 “igb08\_hierarchy.stab\_site” 文件作为固定站列表。我们直接将第三行注释掉，然后在其之前添加一个固定站列表即可。最终这一项配置如下（注意保持非注释行第一列为空格）：

# This should match the well-determined sites in the apr\_file  
stab\_site clear  
stab\_site +bjfs  
stab\_site +chan  
stab\_site +shao  
x source ~/gg/tables/igb08\_hierarchy.stab\_site

**初始文件生成**

配置好解算参数后，现在开始生成 GLOBK 程序需要的二进制格式的 H-文件。这需要使用 htoglb 命令：

|  |  |
| --- | --- |
| 1 | $ htoglb <outPath> <emphmerisPath> <hFiles> |

这里的 <outPath> 表示二进制 H-文件输出路径，<emphmerisPath> 表示卫星状态文件路径，<hFiles> 为要转换的 H-文件列表。

具体到本次任务，在 gsoln 文件夹内执行的是：

|  |  |
| --- | --- |
| 1 | $ htoglb ../glbf/ ~/gg/tables/svnav.dat htesta.16050 |

现在，查看工作目录下的 glbf 文件夹，将看到转换出的两个文件：松散约束下的模糊度实数解 .glr 文件以及松散约束下的模糊度整数解 .glx 文件。这两种文件选取一种作为 GLOBK 的输入即可。

一般而言，处理短基线时，建议采用 .glx 文件；而对于数千公里的长基线，才采用 .glr 解。本次任务中，最长的基线是 CHAN（长春）和 SHAO（上海）之间的连线。利用高德地图的“测距”功能，发现这两个城市的距离大约为 1400 km，并不算太长。因此，本次任务中我将选用 .glx 文件作为输入。

在得到所需的二进制 H-文件之后，我们需要制作一个 .gdl 文件作为使用的二进制 H-文件列表。生成这个文件通常使用 Linux 系统的 ls 命令，若使用 .glx 文件，可以使用命令：

|  |  |
| --- | --- |
| 1 | $ ls ../glbf/\*.glx > <expt>\_glx.gdl # <expt> 表示项目名 |

该命令将 glbf 目录下所有的 .glx 文件写入生成的 .gdl 文件中。类似的，使用 .glr 文件时创建 .gdl 文件的命令为：

|  |  |
| --- | --- |
| 1 | $ ls ../glbf/\*.glr > <expt>\_glx.gdl |

具体到本次任务：

|  |  |
| --- | --- |
| 1 | $ ls ../glbf/\*.glx > test\_glx.gdl |

**开始解算**

网平差解算主要使用两个命令：glred 和 globk。它们的输入参数类似：

|  |  |
| --- | --- |
| 1  2 | $ glred <stdOut> <printFile> <logFile> <hList> <cmdFile>  $ globk <stdOut> <printFile> <logFile> <hList> <cmdFile> |

参数释义：

* <stdOut> 表示输出路径，常设为 6 使其输出到当前窗口；
* <printFile> 表示结果文件的输出路径；
* <logFile> 表示解算记录文件输出路径；
* <hList> 即上文制作的 .gdl 文件；
* <cmdFile> 表示输入的 .cmd 文件。

具体到本次任务，使用的命令为：

|  |  |
| --- | --- |
| 1  2 | $ glred 6 glred.org glred.log test\_glx.gdl globk\_comb.cmd  $ globk 6 globk.org glred.log test\_glx.gdl globk\_comb.cmd |

**成果查看**

如果 GLOBK 程序运行成功，其网平差结果将保存在 .org 文件中。例如对于本次解算任务，GLRED 程序的运行成果在 glred.org 文件内，GLOBK 程序的运行成果保存在 globk.org 文件内。如果你足够仔细，会发现本次任务输出的两个结果文件的内容是一样的。

若程序运行中出现错误，错误信息将保存在 gsoln 文件夹的 GLOBK.fatal 文件内，可以根据其提示对数据处理过程进行适当调整。

有时可能出现网平差解算出的结果中，已知点不能成功固定的现象。如果你的操作和设置没有问题，那么可能是网形结构等原因导致的，可以尝试通过调整联测和平差时的固定点重新解算来解决。

# Linux 系统安装 Mono

[2016-09-03](http://www.gnss.help/2016/09/03/Install-Mono/)

[LINUX](http://www.gnss.help/categories/Linux/)

[LINUX](http://www.gnss.help/tags/Linux/)

C# 是我非常喜欢的编程语言，但其开发跨平台应用的能力一直被人诟病，在 UNIX/Linux 操作系统上使用 [Mono](http://www.mono-project.com/) 是一个解决方案。Mono 是一个由 Novell 公司主持开发的项目。该项目的目标是创建一系列跨平台的 .Net 工具，包括 C# 语言编译器和公共语言运行时。Mono 项目不仅可以运行于 Windows 系统上，还可以运行于 Linux、Unix、macOS 和 Solaris 等操作系统。在微软的开源 .Net 项目 .Net Core 成熟之前，Mono 是跨平台运行 C# 程序的不二选择。

本文介绍在 Ubuntu 和 CentOS 操作系统安装 Mono 的过程，读者可作为参考。

## Ubuntu 操作系统

Ubuntu 操作系统使用 Mono，只需从“软件中心”查找“Mono Runtime”进行安装，或者运行命令：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install mono-devel |

即可将 Mono 运行环境安装至你的计算机。

## CentOS 操作系统

### 下载安装包

根据你的需求，在 [Mono 项目官网](http://download.mono-project.com/sources/mono)下载你需要的程序安装包。我下载的是 mono-5.12.0.301.tar.bz2，下面就以这个文件包为例。

### 编译安装

将 Mono 软件安装包拷贝至要部署的计算机上，使用下面的命令解压软件安装包：

|  |  |
| --- | --- |
| 1 | $ tar -jxvf mono\*tar.bz2 |

解压完成后会在软件包所在的目录中出现一个与软件包文件名类似的文件夹。使用 cd 命令进入解压得到的文件夹，依次运行以下命令开始 Mono 的编译安装：

|  |  |
| --- | --- |
| 1  2  3 | $ ./configure --prefix=/usr/local/mono  $ make  $ make install |

取决于你的计算机配置，整个过程大约需要 20~50 分钟。

### 路径配置

程序编译完成之后，执行命令：

|  |  |
| --- | --- |
| 1  2 | $ cd /usr/bin/  $ ln -s /usr/local/mono/bin/mono-sgen mono |

将 Mono 启动程序添加到系统 Shell 的查找目录，然后重新加载 Shell 配置文件或注销并重新登录以应用配置。

上述操作完成之后可以在终端中运行 mono 命令检查 Mono 是否安装成功。若安装成功将显示 Mono 软件的相关提示信息。

## 使用 Mono

成功安装 Mono 之后，使用命令：

|  |  |
| --- | --- |
| 1 | $ mono <exePath> |

即可在 Mono 中运行 C# 编写的程序，这里的 <exePath> 为 .exe 文件的路径。

# Linux 系统搭建离线软件源

[2016-09-02](http://www.gnss.help/2016/09/02/build-offline-software-source/)

[LINUX](http://www.gnss.help/categories/Linux/)

[LINUX](http://www.gnss.help/tags/Linux/)

为运行 Linux 操作系统的计算机安装程序时，处理软件包的之间依赖关系是一个苦恼的问题。幸好主流的 Linux 发行版都有自己的解决方案：比如红帽系（CentOS、RHEL 和 Fedora）有 yum，Debian 系（Debian、Ubuntu 和 Deepin 等）有 dpkg，可以通过网络上的软件源来解决依赖问题。但是，如果你遭遇到龟速的网络甚至离线情况下，如何解决较大型软件的安装问题呢？搭建一个离线的软件源是一个好办法！

本文以 RHEL 和 Ubuntu 两种发行版为例，介绍主流 Linux 系统离线软件源的搭建方法。因为软件源搭建属于权限较高的操作，所以文中的操作最好在 root 账户下执行。

## RHEL 系统搭建方法

首先说明，尽管这里以 RHEL（Red Hat Enterprise Linux）为例，但是这里的方法也适用于 CentOS 和 Fedora 等同样使用 yum 包管理器的 Linux 发行版。

### 获取所需程序包

软件源实际是就是一个软件包的仓库，通过为仓库中的软件包建立一个索引，使系统在遇到软件包依赖问题时可以按照索引自动安装所需的程序包，这样来达到解决依赖问题的目的。所以，在搭建软件源之前，你要有创建软件源所需的软件包。

一般情况下，所搭建的离线软件源应该至少包含你想要安装的程序以及最常用的一些程序包。需要安装的程序可以去项目的官网下载，而 RHEL 系统的安装光盘里包含了最常用的软件包，通常有这两样就够了。

### 创建本地软件源

把系统安装盘放到光驱里，使用下面的命令挂载光盘：

|  |  |
| --- | --- |
| 1 | $ mount /dev/cdrom |

挂载成功后，把你准备的 .rpm 格式软件包以及系统安装盘的 /Packages 目录下的软件包拷贝到本机的一个文件夹下（例如 /mnt/cdrom，本文即以这个文件夹为例）。

接下来使用 createrepo 命令为 /mnt/cdrom/ 目录创建索引信息：

|  |  |
| --- | --- |
| 1 | $ createrepo -v /mnt/cdrom/ |

然后新建一个 repo 文件（以 local.repo 为例）并编辑：

|  |  |
| --- | --- |
| 1 | $ vi /etc/yum.repos.d/local.repo |

在其中添加以下内容：

|  |  |
| --- | --- |
| 1  2  3  4  5 | [local\_server] # 库名称  name=This is a local repo # 名称描述  baseurl=file:///mnt/cdrom # 程序包所在目录  enabled=1 # 是否启用该软件源，0为禁用  gpgcheck=0 # 是否检查 GPG-KEY，0为不检查，1为检查 |

之后保存，退出。

### 应用配置

最后就是应用我们的配置，这需要清理现有的 yum 缓存并建立新缓存：

|  |  |
| --- | --- |
| 1  2 | $ yum clean all  $ yum makecache |

现在，可以使用离线软件源来安装一个程序试试了，比如安装 FTP 客户端：

|  |  |
| --- | --- |
| 1 | $ yum install ftp |

若能够安装成功，说明我们搭建得没有问题。但是在离线环境下，执行 yum makecache 时，可能会出现类似如下的错误提示：

One of the configured repositories failed (Unknown)

这是因为系统访问其他的软件源失败导致的，禁用其他软件源即可解决该问题：

|  |  |
| --- | --- |
| 1  2 | $ mkdir /etc/yum.repos.d/repo\_bak  $ mv /etc/yum.repos.d/CentOS-\*repo /etc/yum.repos.d/repo\_bak |

## Ubuntu 系统搭建方法

现在介绍 Ubuntu 系统搭建离线软件源的方法，但该方法也适用于 Debian、Deepin 等发行版。我实际操作使用的是基于 Ubuntu 的发行版 elementary OS Loki 0.4.1 或 [elementary OS Juno 5.0](https://elementary.io/)。目前 elementary OS 和 Ubuntu 非常相近，配置过程可以相互参考。

### 获取所需程序包

Ubuntu 系统的安装光盘仅包含很少量的程序包，这些程序包位于光盘的 /pool 文件夹内。但 Ubuntu 有一个特性：安装或更新过的程序，其安装包会缓存在 /var/cache/apt/archives 目录下。如果你还有一台能够联网的 Ubuntu 系统计算机（可使用虚拟机代替），并且你要安装的程序已经包含在网络上的软件源内，你可以利用这个特性来方便地获取所需的程序安装包。在 Ubuntu 操作系统上，你可以使用如下命令通过软件源下载程序：

|  |  |
| --- | --- |
| 1 | $ apt-get -d install <software> |

这个命令中的 <software> 代表你要安装的程序名。该命令将下载你需要的程序及其所需要的依赖程序的安装包，将其放在 /var/cache/apt/archives 目录下。拷贝该目录下的所有文件即可获取到所需的安装包了。

为 Ubuntu 制作离线软件源需要使用一个程序 “dpkg-dev”，Ubuntu 系统默认通常不会安装它（elementary OS Loki 0.4.1 中已经安装了）。但是其安装方法非常简单，只需要在联网的计算机上执行：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install dpkg-dev |

### 创建本地软件源

在联网的计算机上，将你准备好的程序包拷贝到某个文件夹下，例如 /mnt/localpacks，我们将为该文件夹内的所有安装包创建索引以作为本地的软件源。在终端中执行以下命令来创建索引：

|  |  |
| --- | --- |
| 1  2 | cd /mnt/localpacks  dpkg-scanpackages . /dev/null | gzip -9c > Packages.gz |

程序扫描所有的软件包后，将会在 /mnt/localpacks 目录中生成一个名为 Packages.gz 的文件，存放这个文件夹中的软件包的信息及其依赖关系。

现在这个文件夹就可以作为一个离线的软件仓库来使用了。你可以利用这个文件夹直接为本机创建离线的软件源，当然也可以将其拷贝到其它的计算机上使用。

## 应用配置

要应用上文生成的文件夹作为离线软件源，需要编辑文件 /etc/apt/sources.list，该文件保存了 Ubuntu 操作系统可用的软件源列表。编辑前切记备份以前的文件：

|  |  |
| --- | --- |
| 1 | $ cp /etc/apt/sources.list /etc/apt/sources.list.bak |

然后编辑 /etc/apt/sources.list，添加离线软件仓库的目录，此处依然假设离线软件源位于 /mnt/localpacks 文件夹。如果你使用的是 Ubuntu 16.04 LTS，在 sources.list 文件中添加以下内容：

|  |  |
| --- | --- |
| 1 | deb file:/mnt/localpacks ./ |

在较新的操作系统版本中（如 Ubuntu 18.04 LTS），系统默认将拒绝使用未认证过的软件源。因此在配置 sources.list 文件时，需要同时设置对我们搭建的软件源不做安全校验：

|  |  |
| --- | --- |
| 1 | deb [ allow-insecure=yes ] file:/mnt/localpacks ./ |

保存，退出。最后，更新系统软件源以应用配置：

|  |  |
| --- | --- |
| 1 | $ apt-get update |

至此，离线软件源配置完成，你可以安装程序试试了。比如安装 dos2unix 程序：

|  |  |
| --- | --- |
| 1 | $ apt-get install dos2unix |

# GAMIT 批处理基线解算

[2016-09-01](http://www.gnss.help/2016/09/01/sh-gamit/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/)

[前文](http://www.gnss.help/2016/08/27/GAMIT-StepbyStep/)已经介绍过使用 GAMIT 程序分步基线解算的过程。本文将介绍 GAMIT 批处理基线解算的流程。

首先说明，分步和批处理两种方式只是操作上的不同。如果使用的观测数据、解算配置等其他条件一致，两种方式的解算结果也是相同的。但相比分步操作，批处理方式确实可是大大节省操作时间，而且因为不必重复输入相同参数，出错概率也可以相应降低。

GAMIT 批处理基线解算的功能主要依赖于 sh\_gamit 脚本命令，该脚本使用 C Shell 语言编写。如果你有兴趣深入研究其代码实现，它位于 GAMIT/GLOBK 安装目录的 com 文件夹内。

## 文件组织

[前文](http://www.gnss.help/2016/08/27/GAMIT-StepbyStep/)中我曾提到 GAMIT 处理数据时的文件组织方式。在使用 GAMIT 进行批处理处理时，文件组织的方式是一致的：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | · work/  |—— <doy>/  |—— brdc/  |—— igs/  |—— rinex/  |—— tables/ |

我们首先创建一个文件夹作为工作目录，工作目录命名可以是任意的（这里以 work 为例）。在工作目录下平行地创建上面的 5 个文件夹。其中年积日文件夹中存放数据处理结果和中间文件，brdc 文件夹存放广播星历文件，igs 文件夹存放精密星历文件，rinex 文件夹存放观测数据，tables 文件夹存放解算时所需的一些配置文件和解算需引入的共用表文件。

有些情况下，你可能还需要引入观测站点的气象观测数据，即 RINEX 格式的气象数据，这时你可以在工作目录下创建一个名为 met 的文件夹，然后将所有气象数据拷贝至其中。

在下文中，我将以处理 2016 年 8 月 17 日的几个 IGS 站点（BJFS，CHAN，DAEJ，SHAO，SUWN，WUHN）的观测数据作为示例。这一天是星期三，对应的年积日是 230，对应的 GPS 周是1910。我写这篇博文的日期是 2016 年 9 月 1 日，截至现在还没有对应日期的最终星历，所以只好使用快速星历。因此具体的文件结构是：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | · work/  |—— 230/  |—— brdc/  |—— brdc2300.16n  |—— igs/  |—— igr19103.sp3  |—— rinex/  |—— bjfs2300.16o  |—— chan2300.16o  |—— daej2300.16o  |—— shao2300.16o  |—— suwn2300.16o  |—— wuhn2300.16o  |—— tables/ |

## 参数配置

### 表文件更新

在进行数据处理之前最好先检查一下，GAMIT 安装目录的 tables 目录里是否有对应观测数据日期的共用表文件。详细的共用表文件更新方法可以查看本站和 [GAMIT/GLOBK 程序更新](http://www.gnss.help/2016/08/29/UpdateGAMIT/)相关的博文。这里只列出需要检查的共用表文件：

|  |  |
| --- | --- |
| 1  2  3 | soltab.2016.J2000 # 2016 年的太阳历表文件  luntab.2016.J2000 # 2016 年的月历表文件  nutabl.2016 # 2016 年的章动表文件 |

确认以上文件存在。然后分别检查文件 ut1.usno 和 pole.usno，这两个文件的第一行标注了文件的更新日期。要确保其更新日期在我们的数据观测日期之后。

### 链接表文件

确定我们所需的所有文件都齐备之后，在工作目录中使用 sh\_setup 命令将所需的共用表文件和解算配置文件链接至工作目录的 tables 文件夹中。前文已经提到过 sh\_setup 的使用方法：

|  |  |
| --- | --- |
| 1 | $ sh\_setup -yr <year> -doy <doy> |

这里我们具体使用的命令为：

|  |  |
| --- | --- |
| 1 | $ sh\_setup -yr 2016 -doy 230 |

### 参数配置

前文介绍 [GAMIT 分步基线解算](http://www.gnss.help/2016/08/27/GAMIT-StepbyStep/)的博文中，提到过两个解算过程的配置文件：sestbl.和 sittbl.，而 GAMIT 批处理基线解算的操作中，又多了两个配置文件：process.defaults 以及 sites.defaults。这些配置文件都已经被之前的命令拷贝到 tables 文件夹中了。我们依然只修改必要的配置，对于其他设置项，保存默认状态。

修改 sestbl. 文件，找到项目：

Use otl.list = Y         ; Ocean tidal loading list file from OSO  
Use otl.grid = N         ; Ocean tidal loading grid file, GAMIT-format converted from OSO

这两项是 GAMIT 基线解算中所需的海潮改正模型文件的配置。如果你的文件中这两项的配置不是这样的，你可以直接修改文件成上面这样。

## 初始文件生成

在前面介绍 [GAMIT 分步基线解算](http://www.gnss.help/2016/08/27/GAMIT-StepbyStep/)的博文中，我们制作了两个初始文件：station.info 和 lfile.。但在批处理操作时，程序可以自动生成 lfile.，所以只需要制作 station.info 文件就可以了。

因为这次我只处理 IGS 测站的观测数据，所以使用下载的 station.info 文件即可。

## 开始解算

现在开始进行数据处理。与分步操作方式不同的一点是，sh\_gamit 命令并不是在年积日文件夹下运行的，而是在工作文件夹下运行。它的常用参数为：

|  |  |
| --- | --- |
| 1 | $ sh\_gamit -expt <expt> -d <year> <doys> -orbit <IGST> |

其中，<expt> 代表解算项目名；<IGST> 代表解算时使用的精密星历类型，其值可以为：IGSF、IGFR、IGSU 等；<doys> 代表以列表形式列出的所有待处理数据的年积日。

如果你有大量观测日期上连续的数据需要处理，sh\_gamit 还有指定数据观测的起止年积日方式：

|  |  |
| --- | --- |
| 1 | $ sh\_gamit -expt <expt> -s <year> <d1> <d2> -orbit <IGST> # <d1> <d2> 分别为起止年积日 |

例如要处理 2016 年前 3 天的观测数据，你可以这样输入：

|  |  |
| --- | --- |
| 1 | $ sh\_gamit -expt demo -d 2016 001 002 003 -orbit IGSF # IGSF 表示使用最终星历 |

也可以这样：

|  |  |
| --- | --- |
| 1 | $ sh\_gamit -expt demo -s 2016 001 003 -orbit IGSF |

对于本文作为示例的处理任务，输入是这样的：

|  |  |
| --- | --- |
| 1 | $ sh\_gamit -expt work -d 2016 230 -orbit IGSR # IGSR 表示使用快速星历 |

## 成果查看

与分步操作相同，解算完成之后，数据处理成果保存在 3 个文件中，即 O-文件，Q-文件和 H-文件，这些文件在年积日文件夹下。具体到本文的示例任务，它们是：oworka.230、qworka.230 以及 hworka.16230。

若解算出现问题，错误原因保存在 GAMIT.fatal 文件内，可以根据其中信息对解算操作进行调整。

# 文件名大小写批量转换

[2016-08-31](http://www.gnss.help/2016/08/31/Letter-Case-Switch/)

[RINEX](http://www.gnss.help/categories/RINEX/)

[RINEX重命名](http://www.gnss.help/tags/RINEX%E9%87%8D%E5%91%BD%E5%90%8D/), [大小写转换](http://www.gnss.help/tags/%E5%A4%A7%E5%B0%8F%E5%86%99%E8%BD%AC%E6%8D%A2/)

由于 GAMIT/GLOBK 程序在进行基线解算时，只能识别小写字母命名的观测文件，大写字母命名的文件在解算时容易出错。因此，在数据处理之前要先做好文件名的标准化。对于大量数据，手动进行文件重命名既费时又费力，而用一个程序完成这项工作最好不过了。

本文介绍我写的两个脚本：sh\_low2up 和 sh\_up2low，实现对指定文件夹内的所有文件进行文件名的大小写转换。脚本使用 C Shell 语言编写，适用于 UNIX/Linux 操作系统。

## 环境配置

要使用这两个脚本，你需要保证系统安装了 C Shell 或 TC Shell 解释器。对于 CentOS、Fedora 或 RHEL 等 Linux 发行版，系统已经默认安装了该解释器，如果你用的是 Ubuntu 等发行版，首先使用命令：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install csh |

安装 C Shell 语言解释器。提示程序安装成功之后，下载脚本文件并把它们放到你的 Shell 搜索路径下（比如 GAMIT/GLOBK 安装目录下的 com 文件夹，或者 /usr/bin 文件夹）。然后在脚本文件所在目录下执行两个命令：

|  |  |
| --- | --- |
| 1  2 | $ chmod +x sh\_up2low  $ chmod +x sh\_low2up |

为这两个脚本添加可执行权限。再重新加载一下 C Shell / TC Shell 配置文件（.cshrc 文件）或直接注销重新登录系统。之后你就可以快乐地使用它们了。

## 使用说明

脚本说明：

|  |  |
| --- | --- |
| 1  2 | $ sh\_up2low -dir <path> # 大写转小写，<path> 为目标文件夹  $ sh\_low2up -dir <path> # 小写转大写 |

例子：

将 ~/document/RINEX 目录内的所有文件的文件名更名为小写：

|  |  |
| --- | --- |
| 1 | $ sh\_up2low –dir ~/document/RINEX |

将当前目录的所有文件的文件名更名为大写：

|  |  |
| --- | --- |
| 1 | $ sh\_low2up –dir ./ |

将当前目录下的 RINEX 文件夹和 IGS 文件夹下的所有文件更名为小写：

|  |  |
| --- | --- |
| 1 | $ sh\_up2low –dir RINEX/ IGS/ |

命令使用方法记不住怎么办？直接输入脚本名回车就可以查看帮助！

最后，下载链接依旧在本站的[下载](http://www.gnss.help/download/)页面。

# RINEX 与 Compact RINEX 格式的转换

[2016-08-30](http://www.gnss.help/2016/08/30/Rnx-Crx-Switch/)

[RINEX](http://www.gnss.help/categories/RINEX/)

[RINEX格式转换](http://www.gnss.help/tags/RINEX%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2/), [RNXCMP](http://www.gnss.help/tags/RNXCMP/)

RINEX 格式和 Compact RINEX（CRINEX，即 RINEX D-文件）格式都是常用的 GNSS 观测数据保存格式，说到这两个数据格式之间的转换，你首先想到的可能是 rnx2crx 和 crx2rnx 两个程序。不错，本文将介绍这两个程序的使用方法，但不仅仅是这两个程序。实际上，你还能从本文了解更多其他的扩展方法。

## RNXCMP 程序包

### RNXCMP 简介

提起 RNXCMP 程序包可能会觉得有些陌生，但它就包括本文开头提到的 crx2rnx 和 rnx2crx 等程序。除了这两个常见的程序外，该程序包还包含两个脚本 crz2rnx 与 rnx2crz。其中，rnx2crz 将 RINEX 格式转换为 Compact RINEX 后，进一步将其压缩为 .Z 文件格式，而 crz2rnx 是其逆向操作。

### 环境配置

RNXCMP 程序提供适用于 Windows、UNIX、Linux 等常见操作系统的版本。访问 RNXCMP 程序包的[官方下载地址](http://terras.gsi.go.jp/ja/crx2rnx.html)需要科学上网，但源代码已在 [Github](https://github.com/CecilHarvey/rnxcmp) 网站开源。本站的[下载](http://www.gnss.help/download/)页面亦提供适用于 Windows、Linux 以及苹果 macOS 操作系统的最新程序包。

#### Windows 操作系统

对于 Windows 操作系统，下载程序包后，使用 cd 命令进入程序所在文件夹，运行 crx2rnx 或程序包中任何一个程序文件名即可看到提示。如果你不想每次使用前都进行文件夹跳转操作，可以为其配置环境变量，或直接将其中所含的可执行文件转移至 “C:\Windows\System32” 文件夹内。

#### UNIX/Linux 操作系统

对于 UNIX 或 Linux 系统，下载适用的程序包后，需首先为其添加可执行权限：

|  |  |
| --- | --- |
| 1  2  3  4 | $ chmod +x crx2rnx  $ chmod +x crz2rnx  $ chmod +x rnx2crx  $ chmod +x rnx2crz |

然后于程序包所在目录中键入其中任何一个程序名，即可使用其对应的功能。

你很可能希望在任何目录中都能直接使用该程序包，此时可以将其中包含的 4 个程序移动至 “/usr/bin” 文件夹。这样就可以在任何目录下的随时调用了。

### 参数说明与示例

#### crx2rnx 与 rnx2crx

crx2rnx 和 rnx2crx 程序的参数是一致的（被方括号包裹的内容表示该参数是可选的）：

|  |  |
| --- | --- |
| 1  2 | $ crx2rnx [-] [-f] [-h] <file>  $ rnx2crx [-] [-f] [-h] <file> |

参数释义：

* -：该参数指定将内容输出到标准输出；
* -f：该参数表示强制覆盖已有的同名文件；
* -h：用于显示帮助信息；
* <file>：表示要处理的文件名。

示例，由 bjfs0010.16o 得到 bjfs0010.16d。若文件夹内已有名为 bjfs0010.16d 的文件，将询问是否覆盖：

|  |  |
| --- | --- |
| 1 | $ rnx2crx bjfs0010.16o |

由 bjfs0010.16d 得到 bjfs0010.16o。若文件夹内已有名为 bjfs0010.16o 的文件，直接覆盖（该选项在编写批量处理脚本时特别有用）：

|  |  |
| --- | --- |
| 1 | $ crx2rnx -f bjfs0010.16d |

处理 RINEX 3 格式的文件，得到 MAO000USA\_S\_20170010000\_01D\_30S\_MO.rnx：

|  |  |
| --- | --- |
| 1 | $ crx2rnx MAO000USA\_S\_20160010000\_01D\_30S\_MO.crx |

#### crz2rnx 与 rnx2crz

文首已提到，rnx2crz 程序可以将 RINEX 观测文件转换为 Compact RINEX 后，进一步压缩为 .Z 格式，而 crz2rnx 可实现该过程的逆向操作。它们接收的参数也是一致的：

|  |  |
| --- | --- |
| 1  2 | $ crz2rnx [-c] [-d] [-f] [-h] <file>  $ rnx2crz [-c] [-d] [-f] [-h] <file> |

参数释义：

* -c 将转换后的文件输出到当前文件夹；
* -d 转换成功后删除源文件；
* -f 强制覆盖已有的同名文件；
* -h 用于显示帮助信息；
* <file> 参数表示要处理的文件名。

示例，由 chan0010.16d.Z 得到 chan0010.16o，保留源文件，若文件夹内已有名为 chan0010.16o 的文件，直接覆盖：

|  |  |
| --- | --- |
| 1 | $ crz2rnx -f chan0010.16d.Z |

由 chan0010.16o 得到 chan0010.16d.Z，并且转换成功后删除源文件：

|  |  |
| --- | --- |
| 1 | $ rnx2crz -d chan0010.16o |

处理 RINEX 3 格式的文件，得到 MAO000USA\_S\_20160010000\_01D\_30S\_MO.crx.Z：

|  |  |
| --- | --- |
| 1 | $ rnx2crz MAO000USA\_S\_20160010000\_01D\_30S\_MO.rnx |

## 使用 GAMIT/GLOBK 内置脚本

你或许不知道，GAMIT/GLOBK 软件中自带了将 RINEX 格式、Compact RINEX 格式以及 .Z 压缩文件格式之间互相转换的命令。这两个命令分别是 sh\_rnx2crx 和 sh\_crx2rnx。如果你正在使用 GAMIT/GLOBK，了解它们的用法将给你带来许多便利。它们封装了 RNXCMP，以实现对观测文件的批量操作。

两个命令接受的参数是一致的：

|  |  |
| --- | --- |
| 1  2 | $ sh\_rnx2crx -d <y/n> -c <y/n> -f <files>  $ sh\_crx2rnx -d <y/n> -c <y/n> -f <files> |

参数释义：

* -d <y/n> 参数指定转换后是否删除原始数据（默认为 yes）；
* -c <y/n> 参数指定是否在转换前/后对其进行解压/压缩（压缩格式为 \*.Z，默认为 yes）；
* <files> 代表要处理的文件名，可使用通配符指定。

示例，由 bjfs0010.15o 得到 bjfs0010.15d.Z，原有的 bjfs0010.15o 被删除：

|  |  |
| --- | --- |
| 1 | $ sh\_rnx2crx -c y -d y -f bjfs0010.15o |

由 bjfs0010.15o 得到 bjfs0010.15d，原有的 bjfs0010.15o 被删除：

|  |  |
| --- | --- |
| 1 | $ sh\_rnx2crx -c n -d y -f bjfs0010.15o |

由 bjfs0010.15d.Z 得到 bjfs0010.15o，原有的 bjfs0010.15d.Z 被保留：

|  |  |
| --- | --- |
| 1 | sh\_crx2rnx -c y -d n -f bjfs0010.15d.Z |

由 bjfs0010.15d 得到 bjfs0010.15o，原有的 bjfs0010.15d 被删除：

|  |  |
| --- | --- |
| 1 | sh\_crx2rnx -f bjfs0010.15d |

值得一提的是，尽管这两个脚本已经能够处理 RINEX 3 格式的观测数据，但尚不支持该格式的标准文件名。因此，使用这两个脚本处理 RINEX 3 格式的观测数据时，你需要首先使用 sh\_rename\_rinex3 命令将待处理的文件重命名为 RINEX 2 格式的文件名。sh\_rename\_rinex3 的详细介绍请参考 [GAMIT 的 RINEX 3 文件重命名脚本](http://www.gnss.help/2017/09/13/sh-rename-rinex3/)。

## 使用 Pinot 中的脚本

如果你浏览过本网站的许多页面，你会发现 [Pinot](http://www.gnss.help/tags/Pinot/) 亦存在两个脚本：crnx2rnx.py 和 rnx2crnx.py。实际上，它们也是对 RNXCMP 程序包的封装：借助于 Python 语言的跨平台特性，实现对 RINEX 观测数据的批量处理。相比 GAMIT/GLOBK 内置的脚本，它们具有以下特点：

* 适用于 Windows、UNIX/Linux 等操作系统；
* 选择要处理的文件时更灵活（支持多个通配符）；
* 支持 RINEX 2 和 RINEX 3 格式的文件和文件名；
* 多线程并行处理，某些情况下（配置足够高）可能更快；
* 处理过程中有进度条显示。

如果你希望详细了解这两个脚本，可以移步本站文章：[RINEX 与 Compact RINEX 批量转化脚本](http://www.gnss.help/2017/03/27/pinot-crx-rnx-switch/)。

赏

# GAMIT/GLOBK 更新方法

[2016-08-29](http://www.gnss.help/2016/08/29/UpdateGAMIT/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT更新](http://www.gnss.help/tags/GAMIT%E6%9B%B4%E6%96%B0/)

GAMIT/GLOBK 软件经常会进行更新，其中的更新分为两种：共用表文件更新和程序更新。

共用表文件的更新可以为 GAMIT/GLOBK 提供最新的观测数据的解算支持，如果不及时更新这些文件，可能会造成观测日期较晚的数据不能解算；而程序更新为 GAMIT/GLOBK 程序提供 Bug 修复和功能的增加，如果程序更新不及时，可能无法使用最新的功能。

本文将试图讲述 GAMIT/GLOBK 软件的更新策略和方法。

## 共用表文件更新

在使用 GAMIT/GLOBK 进行数据处理时，共用表文件是必不可少的，它们保存在程序安装目录的 tables 文件夹下。

如果待处理数据的观测日期在 GAMIT/GLOBK 程序的发布日期之前，那么无需进行任何更新。反之，就可能需要更新某些共用表文件。这些共用表文件分为两种：按年发布的和累积更新的。

对于按年更新的文件，每年发布一个更新，并且从文件名可以看出其适用的年份。这样的文件有 3 个，Solar（太阳历）、Lunar（月历）、Nutation（章动）。沿用本博客一贯以尖括号表示变量的方法，其命名方式是这样的：

|  |  |
| --- | --- |
| 1  2  3 | soltab.<year>.J2000 # SOLAR 表文件  luntab.<year>.J2000 # LUNAR 表文件  nutabl.<year> # Nutation 表文件 |

对于累积更新的文件，其更新一般没有固定周期，每次更新时会把更新内容直接加到原来的文件里。所以如果你不确定自己电脑里的文件是不是适用的，那么直接下载最新的文件覆盖以前的也没有问题。这样的文件一般有以下几个：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ut1.usno # 地球自转参数  pole.usno # 极移表  leap.sec # 跳秒表  rcvant.dat # 接收机信息表  antmod.dat # 天线信息表  igs\_08.atx # 天线相位中心改正表 |

所有的这些文件都可以在[SOPAC 的 GAMIT 页面](http://sopac.ucsd.edu/GAMIT.shtml)或其[FTP 服务器](http://garner.ucsd.edu/pub/gamit/tables/)下载，下载之后将其直接放在 GAMIT/GLOBK 安装目录的 tables/ 文件夹内。

在进行数据处理时，可以使用 sh\_setup 或者 sh\_links.tables 脚本命令将所需要的所有共用表文件自动链接到工作目录的 tables/ 文件夹下。前面讲 [GAMIT 分步基线解算](http://gnss.help/2016/08/27/GAMIT-StepbyStep/)的博文中曾经讲过 sh\_setup 命令的使用方法，这里顺便讲一下 sh\_links.tables。它和 sh\_setup 不同之处在于该命令不会链接解算中用的配置文件和测站先验坐标文件（lfile.）。其接受主要参数为：

|  |  |
| --- | --- |
| 1 | $ sh\_links.tables -frame <frame> -year <year> |

其中 <frame> 表示共用表文件采用的时间历元，可选择 B1950 或 J2000，而 <year> 表示共用表文件的使用年份。当然，链接成功的前提是程序安装目录的 tables/ 文件夹下有对应的表文件，否则会看到链接无效的提示。

## 程序更新

我们都知道，GAMIT/GLOBK 程序的安装包可以在[麻省理工学院（MIT）的 FTP 服务器](ftp://chandler.mit.edu/)上免费获取。麻省理工学院还专门为 GAMIT/GLOBK 程序创建了一个[主页面](http://www-gpsg.mit.edu/~simon/gtgk/)，提供该程序的最新消息和文档。

更新 GAMIT/GLOBK 程序有两种方式：重新安装整个程序或使用更新包。但不论使用哪一种都会同时更新表文件到截至程序发布日期的最新版本。重新安装程序可以参考[前面的博文](http://www.gnss.help/tags/GAMIT%E5%AE%89%E8%A3%85/)，所以此处介绍一下使用更新包进行更新。

麻省理工学院（MIT）在其 FTP 服务器上提供了一个可用于更新 GAMIT/GLOBK 程序的压缩文件，该文件打包了该软件自发布以来的所有共用表文件和程序的更新：incremental\_updates.<yymmdd>.tar.gz，其中的 <yymmdd> 代表更新日期。下载这个文件，并将其放到你的 GAMIT/GLOBK 软件的安装目录，使用 cd 命令进入该目录后，执行 install\_updates 脚本：

|  |  |
| --- | --- |
| 1  2 | $ chmod +x install\_updates  $ ./install\_updates |

该脚本将首先将更新的共用表文件解压到 tables/ 目录，然后询问你是否要更新程序，输入 y 即可重新编译更新的部分代码，将程序更新到最新版本。

赏

# GAMIT 分步进行基线解算

[2016-08-27](http://www.gnss.help/2016/08/27/GAMIT-StepbyStep/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT使用](http://www.gnss.help/tags/GAMIT%E4%BD%BF%E7%94%A8/)

使用 GAMIT/GLOBK 软件进行 GNSS 数据处理时，我们习惯将处理方式分为两种——分步操作和批处理。你或许已经发现，GAMIT/GLOBK 软件是一个没有图形界面的程序，只能在终端中使用它。其中，GAMIT 主要进行基线解算，而 GLOBK 用来对基线解算成果进行网平差。但实际上这套程序并不是只由这两个软件组成的，GAMIT/GLOBK 将整个数据处理划分为很多小的过程，每个过程对应一个小程序，每个小程序实现一个小功能。我们使用这套程序进行数据处理就是按照数据处理的流程依次操作这些小程序，最终得到需要的成果。

所谓分步处理，就是按照数据处理流程利用 GAMIT/GLOBK 所提供的小程序逐步完成整个数据处理流程。但是分布处理使用得多了你会发现：大多数时候所做的处理流程都是相同的，后来便发展出了两个脚本 sh\_gamit 和 sh\_glred，将基线解算和网平差中重复的操作封装起来。我们将使用这两个脚本进行数据处理的过程称为 “批处理”。

现在的 GAMIT/GLOBK 程序包含许多的脚本，可以提供丰富的功能。这些脚本程序保存在程序安装目录的 com/ 文件夹中。但本文专注于介绍分步基线解算的流程。

## 文件组织

在使用 GAMIT 进行数据处理时，我们习惯于这样组织文件：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | · work/  |—— <doy>/  |—— brdc/  |—— igs/  |—— rinex/  |—— tables/ |

我们首先创建一个文件夹作为工作目录，工作目录命名可以是任意的，只要容易记忆即可（这里以 work 为例）。在工作目录下平行地创建上面的 5 个文件夹。其中在年积日文件夹中进行数据处理操作，而 brdc 文件夹存放广播星历文件，igs 文件夹存放精密星历文件，rinex 文件夹存放观测数据，tables 文件夹存放解算时所需的一些配置文件和解算需引入的共用表文件。

例如，我要处理 2016 年 2 月 19 日几个 IGS 测站（BJFS，CHAN，DAEJ，SHAO，SUWN）的观测数据。这一天对应的年积日是 050，对应的 GPS 周是 1884。这样具体的文件结构变成：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | · work/  |—— 050/  |—— brdc/  |—— brdc0500.16n  |—— igs/  |—— igs18845.sp3  |—— rinex/  |—— bjfs0500.16o  |—— chan0500.16o  |—— daej0500.16o  |—— shao0500.16o  |—— suwn0500.16o  |—— tables/ |

## 参数配置

### 表文件更新

准备好参与解算的所需文件之后，现在把 GAMIT 解算所要用到的共用表文件和解算配置文件链接到工作目录的 tables 文件夹中。此时需要先检查一下，GAMIT 安装目录的 tables 目录里是否有对应观测数据日期的共用表文件。

对于例子中的 2016 年 2 月 19 日的观测数据，我需要的这 3 个文件应该是 soltab.2016.J2000、luntab.2016.J2000 和 nutabl.2016。详细的共用表文件更新方法可以查看本站和 [GAMIT/GLOBK 程序更新](http://www.gnss.help/2016/08/29/UpdateGAMIT/)相关的博文。

所有的这些文件都可以在[SOPAC 的 GAMIT 页面](http://sopac.ucsd.edu/GAMIT.shtml)下载，下载之后直接放在 GAMIT/GLOBK 安装目录的 tables 文件夹内。

### 链接表文件

确保所需表文件都已齐备之后，可以进行链接表文件的操作。这里需要使用一个批处理命令：sh\_setup，它的详细用法是这样的：

|  |  |
| --- | --- |
| 1 | $ sh\_setup -yr <year> -doy <doy> |

这里的 <year> 和 <doy> 即数据观测的年与年积日。具体到本次任务，需要执行的应该是（在工作目录下执行）：

|  |  |
| --- | --- |
| 1 | $ sh\_setup -yr 2016 -doy 050 |

### 参数配置

执行过上面的表文件链接之后，查看 tables 文件夹，会发现里面多了一些文件，其中有些是实体文件，而有些是快捷方式链接。其中就有包含解算配置的两个文件：sestbl.和 sittbl.。

如果你对其中的各项配置参数不熟悉，那么使用默认配置也可以。只是需要检查一个地方，在 sestbl. 文件中：

Use otl.list = Y         ; Ocean tidal loading list file from OSO  
Use otl.grid = N         ; Ocean tidal loading grid file, GAMIT-format converted from OSO

这两项是 GAMIT 基线解算中所需的海潮改正文件的配置。如果你的文件中这两项的配置不是这样的，你可以直接修改文件成上面这样。因为我要处理的是 IGS 站的观测数据，直接引入 otl.list 文件就够了。

## 初始文件生成

### 生成 station.info

station.info（测站信息）文件是要处理的观测站点的信息，如测站的接收机、天线等。

SOPAC 网站上提供的这个文件里，已经包含了所有 IGS 测站的信息。因为我处理的都是 IGS 测站的观测数据，所以直接使用下载的 station.info 文件就可以了。如果你要处理的观测数据中包含 IGS 测站之外的数据，那么可以在工作目录的 tables 目录中使用有希望命令对现有的 station.info 文件进行更新：

|  |  |
| --- | --- |
| 1 | $ sh\_upd\_stnfo -files <RinexOfilePath> # <RinexOfilePath> 为观测文件路径，如 ../rinex/\*o |

关于 sh\_upd\_stnfo 脚本的详细使用说明，可以参考博文[站点信息文件的创建与更新](http://www.gnss.help/2017/03/05/create-station-info/)。

### 生成 lfile.

lfile.（测站概略坐标）文件包含待处理的站点的概略坐标。GAMIT 安装目录的 tables 文件夹里自带的 lfile. 已经包含了所有 IGS 站点的概略坐标，并且你会看到该文件已经被 sh\_setup 命令复制到了工作目录的 tables 文件夹里了。对于只处理 IGS 站的观测数据而言，这样就足够了。但是如果你还有其他的观测数据，获取它们的概略坐标将是一个值得探讨的过程。这里只简单地给出一个方案，因为该方案使用 RINEX 文件头中的概略坐标制作 lfile.，所以不一定适用于所有情况。在工作目录的 tables 目录执行：

|  |  |
| --- | --- |
| 1  2  3 | $ grep "APPROX POSITION XYZ"$ ../rinex/\*o > lfile.rnx # 提取观测文件中的概略坐标  $ rx2apr lfile.rnx <year> <doy> # 生成 lfile.rnx.apr 文件  $ gapr\_to\_l lfile.rnx.apr lfile. <year> <doy> # 生成 lfile. 文件 |

详细的概略坐标获取方法，请移步[获取站点概略坐标的几种方法](http://www.gnss.help/2017/02/26/get-approx-position/)。

## 开始解算

### 链接文件

经过前文的操作，我们已经完成了所有的文件准备，终于可以开始进行数据处理。

我们之前已经创建了一个以年积日命名的文件夹，之后的所有的数据处理操作都将在这个文件夹中进行。

首先，把前文准备的所有文件都链接到年积日文件夹内。这里使用了 Linux 系统的命令 ln -s 来创建软链接。

|  |  |
| --- | --- |
| 1  2  3  4 | $ ln -s ../rinex/\*o ./ # 将观测数据链接到年积日文件夹  $ ln -s ../brdc/\*n ./ # 将广播星历链接到年积日文件夹  $ ln -s ../igs/\*sp3 ./ # 将精密星历链接到年积日文件夹  $ ln -s ../tables/\* ./ # 将配置文件和表文件链接到年积日文件夹 |

### 开始解算

依次执行下面的命令完成基线解算过程。

首先执行 makexp 命令。该命令进行解算前的预处理工作，用以生成后续命令的输入控制文件：

|  |  |
| --- | --- |
| 1 | $ makexp |

执行该命令后，系统将引导你完成数据的预处理并生成 session.info 文件，只需按照提示键入项目名（4 位字符）、精密星历类型（4 位字符：igsf 表示最终星历、igsr 表示快速星历、 igsu 表示超快星历）、GNSS 系统代号（1 位字符：G 表示 GPS、R 表示 GLONASS、C 表示 BDS）、数据观测日期年（4 位数字）、年积日（3 位数字）、任务编号（2 位数字，99 表示所有任务）、测站概略坐标文件名、广播星历文件名、采样信息（采样间隔，开始时间、历元数）。

具体到本次使用的 2016 年 2 月 19 日的观测数据，输入依次为：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | $ test # 项目名  $ igsf # 星历类型  $ g # GNSS 系统代号  $ 2016 # 数据观测日期的年  $ 050 # 数据观测日期的年积日  $ 99 # 任务编号  $ lfile. # 测站概略坐标文件名  $ brdc0500.16n # 广播星历文件名  $ 30 00 00 2880 # 采样间隔，开始时间（时、分），历元数 |

需要注意的是，在 GAMIT/GLOBK 10.7 中，makexp 程序生成的 session.info 文件有错误：其中缺失了所有待处理的卫星 PRN 号。该版本的用户可以使用如下命令生成 session.info 文件：

|  |  |
| --- | --- |
| 1 | $ sh\_makexp -expt test -orbt igsf -yr 2018 -doy 152 -sess 99 -nav brdc0500.16n -sp3file igs18845.sp3 -jclock sp3 -apr lfile. -sinfo 30 00 00 2880 |

现在链接精密星历，生成轨道初值（G-文件）：

|  |  |
| --- | --- |
| 1 | $ sh\_sp3fit -f <sp3file> -o <name> -d <year> <doy> |

这里的 <sp3file> 为 sp3 文件名，<name> 为 4 个字符的轨道名，该值将被包括在生成的 G-文件的文件名中，因此最好能够体现出轨道信息的来源或类型。具体到本次任务：

|  |  |
| --- | --- |
| 1 | $ sh\_sp3fit -f igs18845.sp3 -o igsf -d 2016 050 |

生成卫星钟文件：

|  |  |
| --- | --- |
| 1 | $ makej <navfile> jbrdc<y>.<doy> |

这里的 <navfile> 为输入的广播星历文件，<y> 代表一位数字表示的观测年（即最后一位），<doy> 为年积日。具体到本次任务：

|  |  |
| --- | --- |
| 1 | $ makej brdc0500.16n jbrdc6.050 |

现在生成双差观测文件（X-文件）：

|  |  |
| --- | --- |
| 1 | $ makex <expt>.makex.batch |

这里的 <expt> 表示项目名，具体到本次任务：

|  |  |
| --- | --- |
| 1 | $ makex test.makex.batch |

现在运行 fixdrv 命令，该程序根据解算控制文件生成一系列的批处理脚本：

|  |  |
| --- | --- |
| 1 | $ fixdrv d<expt><y>.<doy> |

具体到本次任务：

|  |  |
| --- | --- |
| 1 | $ fixdrv dtest6.050 |

最后启动数据解算脚本

|  |  |
| --- | --- |
| 1 | $ csh b<expt><y>.bat |

具体到本次任务：

|  |  |
| --- | --- |
| 1 | $ csh btest6.bat |

之后静待数据解算完成即可。

## 成果查看

若解算成功，最终的数据处理成果主要在 3 个文件中，即 O-文件，Q-文件和 H-文件。如果你在 sestbl. 文件中设置了

Output met = Y

那么你还可以得到包含对流层参数解算结果的 Z-文件。可以由这个文件进一步处理得到可降水汽 PWV 的值。

若解算过程中出现错误，则会产生 GAMIT.fatal 文件，其中保存了错误原因，可以根据其中提示对解算过程进行调整。

# Ubuntu 操作系统安装 GAMIT

[2016-08-26](http://www.gnss.help/2016/08/26/ubuntu-install-gamit106/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT安装](http://www.gnss.help/tags/GAMIT%E5%AE%89%E8%A3%85/)

[Ubuntu OS](https://www.ubuntu.com/) 无疑是目前最流行的 Linux 发行版，虽然网络上 Ubuntu 系统安装 GAMIT/GLOBK 软件的教程多如牛毛，但质量却良莠不齐。作者在此处进行整理，以供读者进行参考。我操作时所用的 Ubuntu 版本为 16.04 LTS，GAMIT/GLOBK 版本为 10.6。但是鉴于 GAMIT/GLOBK 程序安装方法一直无甚大变化，因此本安装流程也适用于 10.5、10.7 等版本，读者在参考时只需作少量变通即可。

提示：为了顺利执行本文提到的操作，你需要先了解 Linnux 系统的 sudo 和 cd 命令。需要在终端中输入的命令，在本文中以 “$” 字符开头，但操作时该字符不需要键入。

## 安装必要组件

在开始 GAMIT/GLOBK 软件的安装之前，需要首先安装 TC Shell 解释器、libX11-dev 图形开发库和 GNU-Fortran（gfortran）编译器。libX11-dev 是 GAMIT/GLOBK 所依赖的一个必要组件，而 GNU-Fortran 是一个开源的 Fortran 编译器，我们使用它来编译 GAMIT/GLOBK 源代码以生成可执行程序。

### 安装 TC Shell

Ubuntu 操作系统没有默认安装 C Shell 和 TC Shell 解释器，要运行 GAMIT/GLOBK 软件，这是必不可少的组件。因此我们首先安装 TC Shell 解释器。在终端中执行以下命令：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install tcsh |

### 安装 libx11-dev

在终端中执行下面的命令以安装 libx11-dev：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install libx11-dev |

### 安装 GNU-Fortran

在终端中执行下面的命令来安装 GNU-Fortran（gfortran）：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install gfortran |

需要注意的是，gfortran 7 的一个 Bug 会导致编译 GAMIT 10.7 时出现错误，造成安装失败。因此如果你使用的操作系统基于 Ununtu 18.04 LTS 并且要编译 GAMIT 10.7，则不能安装其默认的 gfortran 7。可以使用如下命令安装 gfortran 6：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install gfortran-6 |

然后使用如下命令创建指向 gfortran-6 的软链接：

|  |  |
| --- | --- |
| 1  2 | $ cd /usr/bin  $ sudo ln -s gfortran-6 gfortran |

### 安装 FTP 工具

FTP 客户端工具主要用于被 GAMIT 的数据下载脚本所调用，没有这个程序将无法使用数据下载脚本，执行下面的命令安装 FTP 客户端：

|  |  |
| --- | --- |
| 1 | $ sudo apt-get install ftp |

## 编译安装 GAMIT/GLOBK

### 为安装包分配权限

将解压之后的 GAMIT/GLOBK 程序安装包拷贝到你想安装至的位置，此处以 /opt 目录为例。

使用 cd 命令进入 /opt 目录，执行命令：

|  |  |
| --- | --- |
| 1 | $ sudo chmod 755 -R gamit |

该命令将为 gamit 文件夹下的文件分配可执行权限。

### 进行安装

进入 gamit 文件夹，执行安装脚本：

|  |  |
| --- | --- |
| 1 | $ ./install\_software |

此时，GAMIT/GLOBK 程序已经开始进行编译之前的文件准备，之后程序会提示

Could not verify path to X11 libraries (libX11.\*) from X11LIBPATH in libraries/Makefile.config  
Perform a search for paths to X11 libraries and header files? (y/n)

这时，输入 n。打开 libraties 文件夹下的 Makefile.config 文件，找到

# Generic (will work on any system if links in place)  
X11LIBPATH /usr/lib/X11  
X11INCPATH /usr/include/X11

将其修改为

# Generic (will work on any system if links in place)  
X11LIBPATH /usr/lib/  
X11INCPATH /usr/include/

再找到：

MAXSIT 80  
MAXSAT 32  
MAXATM 13  
MAXEPC 2880

此处是对 GAMIT 程序解算参数的配置，建议修改为

MAXSIT 80  
MAXSAT 32  
MAXATM 25  
MAXEPC 2880

如果你安装的 GAMIT/GLOBK 程序是较早的版本（10.5 及更早），那么你还需要再找到

# —– for Linux from 0.0.1 to 3.9.5.3 – #  
OS\_ID Linux 0001 3953

将 “0001” 后面的4位数字（此处的是 3953）修改为你的 Linux 系统内核的版本号的前 4 位。查看系统的 Linux 内核版本可以使用命令：

|  |  |
| --- | --- |
| 1 | $ uname -a |

如我的电脑上显示

Linux ubuntu 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86\_64 x86\_64 x86\_64 GNU/Linux

那么我应该修改 Makefile.config 文件的对应部分为：

# —– for Linux from 0.0.1 to 3.9.5.3 – #  
OS\_ID Linux 0001 3130

更改以上内容之后，保存。重新执行：

|  |  |
| --- | --- |
| 1 | $ ./install\_software |

之后的一切询问输入 y，安装成功之后会看到提示：

++++++++++++++++  
GLOBK installed  
++++++++++++++++

### 配置运行路径

为了在任何目录都能运行 GAMIT/GLOBK，需要将程序目录加入到系统的 PATH 变量中。环境变量配置方式依所使用的 Shell 有所不同。在终端内执行如下可以查看当前使用的 Shell：

|  |  |
| --- | --- |
| 1 | $ echo $SHELL |

对于 Bash 用户，打开主目录的 ~/.bashrc，在其中添加以下内容：

|  |  |
| --- | --- |
| 1  2  3 | gg="/opt/gamit"  export PATH="$gg/gamit/bin:$gg/kf/bin:$gg/com:$PATH"  export HELP\_DIR=$gg/help/ |

对于 C Shell 或 TC Shell 用户，打开主目录的 ~/.cshrc 文件，在底部添加以下内容：

|  |  |
| --- | --- |
| 1  2  3 | set gg = /opt/gamit  setenv PATH "$gg/gamit/bin:$gg/kf/bin:$gg/com:$PATH"  setenv HELP\_DIR $gg/help/ |

保存，退出。再执行命令：

|  |  |
| --- | --- |
| 1 | $ source ~/.bashrc |

或

|  |  |
| --- | --- |
| 1 | $ source ~/.cshrc |

至此，GAMIT/GLOBK 软件安装完成。你可以在终端下运行 doy 命令进行检验。另外，若要完整使用 GAMIT/GLOBK 软件的各项功能，如绘图等，你可能还需要安装 [Ghostscript](https://www.ghostscript.com/)，[ImageMagick](http://www.imagemagick.org/) 和 [GMT](http://gmt.soest.hawaii.edu/) 等软件。

# CentOS 操作系统安装 GAMIT

[2016-08-26](http://www.gnss.help/2016/08/26/centos-install-gamit105/)

[GAMIT](http://www.gnss.help/categories/GAMIT/)

[GAMIT安装](http://www.gnss.help/tags/GAMIT%E5%AE%89%E8%A3%85/)

有关 GAMIT/GLOBK 软件的 Ubuntu OS 下的安装博文已经很多了，但是在 [CentOS 操作系统](https://www.centos.org/)下的安装教程还不多。这里介绍一下我的安装过程。所用环境为 CentOS 7，GAMIT/GLOBK 版本为 10.61。但经过之后的测试，该安装流程其实也适用于 CentOS 和 GAMIT 的其他版本。

提示：在进行 GAMIT/GLOBK 软件的安装之前，你最好了解一下 Linux 的 sudi 和 cd 命令。该命令将用于在执行下文的一些操作时进行必要的文件目录跳转。在本文中，约定需要在终端中输入的命令以 “$” 字符开头。

## 安装必要组件

### 安装 libX11-devel

libX11-devel 是运行 GAMIT 所需的一个必要的图形库。在终端执行以下命令即可安装该图形库：

|  |  |
| --- | --- |
| 1 | $ sudo yum install libX11-devel |

### 安装 GNU-Fortran

GNU-Fortran 用于编译 GAMIT 中由 Fortran 语言编写的计算程序，使用如下命令安装它：

|  |  |
| --- | --- |
| 1 | $ sudo yum install gcc-gfortran |

### 安装 FTP 客户端

FTP 客户端用于供 GAMIT 自动下载命令调用，在终端中执行下面的命令来安装 FTP 客户端：

|  |  |
| --- | --- |
| 1 | $ sudo yum install ftp |

## 编译安装 GAMIT/GLOBK

### 为安装包分配权限

将解压后的安装包拷贝到 /opt 目录内（当然，此处可以是你希望安装 GAMIT/GLOBK 程序的任何目录，本文以 /opt 为例）。

在 /opt 目录中执行以下命令为 GAMIT 所在目录分配可执行权限：

|  |  |
| --- | --- |
| 1 | $ sudo chmod 755 -R gamit |

### 进行安装

进入 gamit 文件夹，执行安装脚本：

|  |  |
| --- | --- |
| 1 | $ ./install\_software |

在提示安装参数：

MAXSIT 60  
MAXSAT 32  
MAXATM 13  
MAXEPC 2880

处时，输入 n。打开 libraties 文件夹下的 Makefile.config 文件，需修改以下两处：

MAXSIT 60  
MAXSAT 32  
MAXATM 13  
MAXEPC 2880

这几个数字为你的任务所需参数。

释义：MAXSIT（最大测站数，视你的解算任务）；MAXSAT（最大卫星数，一般不用修改）；MAXATM（最大对流层延迟数，建议修过为25）MAXEPC（最大解算历元数，一般不用修改）。

修改

# —– for Linuxfrom 0.0.1 to 3.0.0 – #  
OS\_ID Linux 0001 3000

为

# —– for Linuxfrom 0.0.1 to 3.0.0 – #  
OS\_ID Linux 0001 2632

释义：该处的修改值是你 Linux 的内核版本号，修改值视情况而定，我的系统内核版本号是 2.6.32。可新开一个终端，执行 uname -a 命令查看你所用的系统版本。

更改以上内容之后，保存。重新执行：

|  |  |
| --- | --- |
| 1 | $ ./install\_software |

之后的一切询问输入 y，安装成功之后会看到提示：

++++++++++++++++  
GLOBK installed  
++++++++++++++++

### 配置运行路径

为了在任何目录都能运行 GAMIT/GLOBK，需要将程序目录加入到系统的 PATH 变量中。环境变量配置方式依赖你使用的 Shell 有所不同。在终端内执行如下命令可以查看当前使用的 Shell：

|  |  |
| --- | --- |
| 1 | $ echo $SHELL |

对于 Bash 用户，打开主目录的 ~/.bashrc，在其中添加以下内容：

|  |  |
| --- | --- |
| 1  2  3 | gg="/opt/gamit"  export PATH="$gg/gamit/bin:$gg/kf/bin:$gg/com:$PATH"  export HELP\_DIR=$gg/help/ |

对于 C Shell 或 TC Shell 用户，打开主目录的 ~/.cshrc 文件，在底部添加以下内容：

|  |  |
| --- | --- |
| 1  2  3 | set gg = "/opt/gamit"  setenv PATH "$gg/gamit/bin:$gg/kf/bin:$gg/com:$PATH"  setenv HELP\_DIR "$gg/help/" |

保存，退出。再执行命令：

|  |  |
| --- | --- |
| 1 | $ source ~/.bashrc |

或

|  |  |
| --- | --- |
| 1 | $ source ~/.cshrc |

至此，GAMIT/GLOBK 软件安装完成。你可以在终端下运行 doy 命令进行检验。

如果你的计算机有多名用户，你希望其他的用户也可以使用你安装的 GAMIT/GLOBK 程序，只需要为这些用户的主目录下配置同上的环境变量并添加一个指向 GAMIT/GLOBK 安装目录的 gg 链接即可。另外，若要完整使用GAMIT/GLOBK软件的各项功能，如绘图等，你可能还需要安装 [Ghostscript](https://www.ghostscript.com/)，[Imagemagic](http://www.imagemagick.org/) 和 [GMT](http://gmt.soest.hawaii.edu/) 等软件。

# Hello World

[2016-08-25](http://www.gnss.help/2016/08/25/hello-world/)

[网站](http://www.gnss.help/categories/%E7%BD%91%E7%AB%99/)

[网站](http://www.gnss.help/tags/%E7%BD%91%E7%AB%99/)

作为此网站的第一篇文章，我想在此谈一谈建立此站点的初衷。

在意大利作家卡尔维诺的小说《看不见的城市》中，马可·波罗向忽必烈讲述他一路上的奇遇，讲他遇到过的城市。美丽的城市，丑陋的城市，轻盈的城市，滨水的城市，建在沙漠上的城市，精密得如钟表般的城市，活在记忆里的城市……其中有一座灰石建造的城市菲朵拉。它的中心有一座博物馆，里面摆放着许多的玻璃圆球，每个圆球里都是一个理想中的菲朵拉的模型。

菲朵拉本可以成为模型里的样子，却由于种种原因变成了现在我们所见的模样。在每个时代里都有某些人，看着当时的菲朵拉，想象着如何把她改建成理想的城市，然而当他们制作理想城市的模型时，菲朵拉已经不再是从前的城市，而那个直至昨日还是可能的未来之城也就只能成为玻璃球里的一件玩具。

生活在石头的菲朵拉的市民，也只能看着玻璃球里的菲朵拉，想象运河池水中水母的倒影和骑在大象背上行走在城市里的滋味。——这听起来像是一个寓言。

回想我刚刚成为 GNSS 专业的研究生的时候，处处都含着好奇心，最擅长不着边际的幻想，总想着能为这个领域做些什么。但是种种新奇的脑洞却没有几个能够实施起来，想想这两年自己做过什么，总觉得实在是一事无成。

现在我终于知道，要是想完成一件事，那么我最好现在就去做。否则我就不是在努力做事，而是在过努力做事这一种生活。正如《白银时代》的总结：

自己在家里闷头就写，不要告诉任何人。这样就是真正在写小说。

——这是王小波教给我的道理。

所以我现在搭建了这样的一个网站，虽然它在我想象中的模样不是这样的：它应该有飞快的响应速度，提供各种实用的在线和离线工具，还应该有简洁优雅又有效的 UI 设计…… 但如果我现在不能让它上线，那么它可能永远也不会上线了。

它现在是这个一个简单的博客，日后会是什么样子的，我也不知道，但肯定不会一直保持现在的面貌。希望这可以督促我一直对它进行改进。

博文的内容并非是一成不变的，会不定时地增删修改，看心情。以前的一些博文发表在[科学网](http://blog.sciencenet.cn/u/PurpleSky)和[博客园](http://www.cnblogs.com/purpleskyfall)上，会逐步的搬运到这里来。对于一些重复的文章，希望读者不要介意。

**Hello world!**

赏