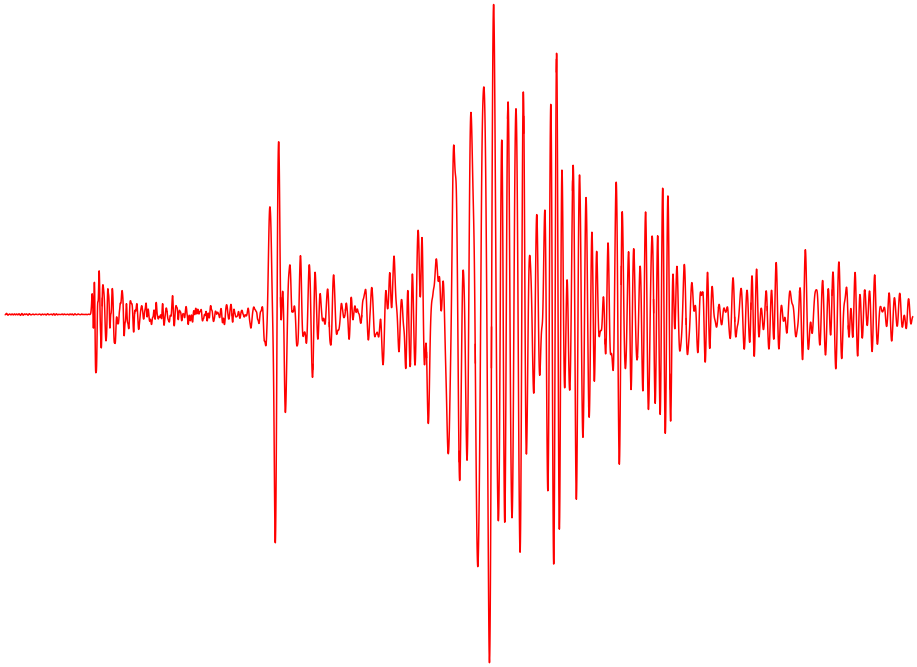

SAC 参考手册

基于 SAC v101.6a

作者: SeisMan

版本: 3.1-dev

日期: 2014-04-20

写在第三版前的一些废话

> “工欲善其事，必先利其器。”
>
> —《论语 · 卫灵公》

2010 年 10 月，大三，开始接触并学习 SAC；2011 年的暑假开始着手 SAC 文档的翻译工作；2012 年 01 月，文档的 v1.0 版本发布；2013 年 03 月，文档的 v2.0 版发布。3 年多的时间过去了，文档更新到了 v3.0 版。

v2.0 大体上算是官方英文文档的译本，整体结构上完全遵循了官方文档的风格。整个文档的条理不够清晰，教程部分稍显单薄，命令部分也不够完善。

2013 年 11 月，George Helffrich 著的 “*The Seismic Analysic Code : A Primer and User's Guide*” 一书出版了。该书基于 MacSAC，与本文档所关注的 SAC 有一些区别，但是精髓部分是一致的。v3.0 版借鉴了该书的整体结构和部分内容，重新设计了文档结构并重写了教程的大部分内容，希望能够有一个结构更清晰、内容更丰富的版本。

整个文档分为教程部分和命令部分。教程部分又分为如下几章：

SAC 简介 简单介绍 SAC 软件的相关信息；
SAC 基础 继续阅读所需的基础知识；
SAC 文件格式 详细介绍 SAC 文件格式；
SAC 数据处理 介绍如何利用 SAC 命令进行地震数据处理和分析；
SAC 图像 介绍如何控制 SAC 绘制的图像的细节；
SAC 编程 介绍如何用 SAC 进行数据批处理；
SAC 与脚本 如何在脚本语言（Bash 和 Perl）中调用 SAC；
SAC 函数库 在自己的 C 或 Fortran 程序中调用 SAC 提供的子函数；
SAC I/O 独立实现 SAC I/O 子函数；
SAC 相关工具 与 SAC 有关的一些工具；

对本文档的内容有疑问，或发现任何错误、笔误，欢迎邮件联系我或者在本文档的 GitHub 项目主页上提交 Issue，也欢迎感兴趣的读者 fork 该项目，共同完善文档。

个人博客: http://seisman.info
项目主页: https://github.com/seisman/SAC_Docs_zh
捐赠页面：https://me.alipay.com/seisman
文档发布页: http://seisman.info/sac-manual.html
联系方式: seisman.info@gmail.com

作者 : SeisMan
2014 年 04 月 14 日

版本说明

SAC 的开发一直在进行，每一个新版本都可能修订一些 Bug、加入一些新特性或新命令。下表列出了本文档的版本与 SAC 版本之间的对应关系。就本文档而言，仅保证其中的所有示例均可在 v101.6a Linux 64 位下正常运行，无法保证在其它版本或其它平台下是否正常。

文档版本	文档发布日期	SAC 版本	SAC 发布日期
1.0	2012-01-08	101.4	2010-06-07
1.1	2012-09-03	101.4	2010-06-07
1.2	2012-09-18	101.5	2011-11-15
2.0	2013-03-29	101.5c	2012-02-01
2.1	2013-04-06	101.5c	2012-02-01
2.2	2013-04-12	101.5c	2012-02-01
2.3	2014-02-22	101.5c	2012-02-01
3.0	2014-04-18	101.6a	2013-11-11
3.1-dev	2014-04-20	101.6a	2013-11-11

目　录

图目录

表目录

第 I 部分　SAC 教程

第 1 章　SAC 简介

1.1　SAC 是什么？

Seismic Analysis Code (SAC)，是天然地震学领域使用最广泛的数据分析软件包之一。

SAC 首先是一个软件，主要在命令行下工作，通过各种命令来处理时间序列数据（尤其是地震波形数据），同时也提供了一个简单的图形界面，使得用户可以方便地查看波形并拾取震相。

SAC 同时还是一种数据格式，定义了以何种方式存储单个时间序列数据。SAC 格式已经成为了地震学的标准数据格式之一，有很多工具可以实现 SAC 格式与其它地震数据格式间的相互转换。

SAC 实现了地震数据处理过程中的常用操作，包括重采样、插值、自/互相关、震相拾取、快速 Fourier 变换、谱估计、滤波、信号叠加等；同时为了满足数据批处理的需求，SAC 设计了一个基本的编程语言 [1]，包含了变量、参数、条件判断、循环控制等特性。

1.2　SAC 发展史

Lawrence Livermore 国家实验室 [2] 和 Los Alamos 国家实验室 [3] 是美国承担核武器设计工作的两个实验室。SAC 于 20 世纪 80 年代诞生于实验室的 Treaty Verification Program 小组里，该组由 W. C. Tapley 和 Joe Tull 共同领导。

起初，SAC 是用 Fortran 语言实现的，并将源代码分发给感兴趣的学者，允许用户进行非商业性的地震数据处理，用户和开发者之间的合作协议要求用户提交 bug 修正和改进以换取 SAC 的使用权。到了大概 1990 年，SAC 已经成为全球地震学家的数据处理标准软件。

从 1992 年开始，SAC 的开发逐渐由 Livermore 接管，并开始通过分发协议严格限制源代码的分发。与此同时，开发者认为 Fortran 是一种过于局限的编程语言，其阻碍了 SAC 特性的进一步开发，因而开发者使用 f2c[4] 转换工具将 SAC 的 Fortran 源码转换成了 C 源码[5]。接下来，Livermore 以转换得到的 C 源码为基础，计划开发一个商业版的地震数据处理产品，命名为 SAC2000。这个版本扩展了很多功能，其中一个功能是建立一个日志数据

[1] SAC 设计的编程语言，称之为 SAC 宏，在 “SAC 编程” 一章中会详细说明。
[2] http://en.wikipedia.org/wiki/Lawrence_Livermore_National_Laboratory
[3] http://en.wikipedia.org/wiki/Los_Alamos_National_Laboratory
[4] f2c (http://www.netlib.org/f2c/)，Fortran77 语言到 C 语言的自动转换工具。
[5] 个人猜测，目前 SAC 源码的混乱和不易读正是由于这次自动转换导致的。

库，记录一个波形从原始数据到最终产品之间的所有处理步骤。这样的设计允许用户暂时保存数据处理步骤，随时将处理的结果提交到内存或回滚到之前的状态。

约 1998 年，IRIS[1] 意识到，SAC 的核心用户群（主要是 IRIS 的成员）无法确保能够获取 SAC 的源码。IRIS 开始和 Livermore 协商，希望将 SAC 的开发分成两条线：一个包含数据库特性，供核监测机构使用；另一个不包含数据库特性，仅供学术机构使用。商业化的努力主要集中在含数据库功能的版本上。

终于，在 2005 年，IRIS 与 Livermore 签订了合同，Livermore 提供给 IRIS 一个 SAC 协议，允许其在 IRIS 社区内部分享 SAC/SAC2000 的源代码，并提供有限的支持以促进社区的发展。而学术圈对于商业版的 SAC 没有太大兴趣，因而 Livermore 逐渐撤出了对于 SAC2000 的支持。最终 IRIS 完全接手了 SAC 的开发和技术支持，并成为了一个新的版本，也就是我们现在正在使用的 SAC，有时为了区分，也称之为 SAC/IRIS。目前的最新版本为 101.6a。

1.3 SAC 变体

SAC 的发展史还是很曲折的，这也导致 SAC 存在多个不同的变体。

Fortran SAC 即 SAC 的 Fortran 语言实现。最后一个分发版本发布于 2003 年，版本号 10.6f。曾经以限制性的形式在 IASPEI 软件库中分发。

SAC2000 从 Fortran 源码转换为 C 源码，并以 C 源码为基础继续维护。该版本加入了数据库特性以及一些新的命令。目前该版本已不再分发。

SAC/IRIS 由 SAC2000 衍生的版本，不包含数据库特性[2]，也就是本文档所使用的版本。现在由 IRIS 下的 SAC 开发小组负责维护，并由 IRIS 分发。

MacSAC 也称为 SAC/BRIS，仅可在 Mac OS 下使用。该变种由 10.6d Fortran 源码衍生而来（后期与 10.6f 集成），其功能是 SAC/IRIS 功能的超集。相对于 SAC/IRIS 的最主要扩展在于宏语言功能的增强以及处理台阵数据的能力。其作者为 George Helffrich[3]，针对 MacSAC 写了一本教程 [4]。

1.4 安装 SAC

本节介绍如何在 Linux 下安装 SAC，要求读者了解 Linux 的一些基本概念和操作。

申请 SAC

在 “SAC 发展史” 中已经说到，SAC 协议仅允许在 IRIS 社区内部分享 SAC 的源码，所以 SAC 不像很多软件一样可以很方便地通过软件包管理器安装或者直接从网络下载源码。

软件包申请地址：`http://www.iris.edu/forms/sac_request.htm`

[1] `http://www.iris.edu`

[2] 目前的 SAC/IRIS 中还可以看到一些与该特性相关的命令和选项，这属于历史遗留问题。

[3] `http://www1.gly.bris.ac.uk/~george/gh.html`

[4] G.R. Helffrich, J. Wookey & I.D. Bastow, The Seismic Analysis Code : A Primer and User's Guide. *Cambridge University Press*, 2013。可以作为学习 SAC/IRIS 的辅助教程，但需要注意其中可能存在的一些微小差异。

认真填写个人信息，尤其注意 Email 那一栏，需要填写单位邮箱，如果使用 QQ、163 这样的邮箱很容易直接被拒绝，如果没有单位邮箱，需要提供其它信息以验证你的身份。

IRIS 提供了 SAC 源码包、Linux 64 位二进制包和 Mac 64 位二进制包。对于 Linux 用户，可以通过 “`uname -a`” 命令查看当前系统是 32 位还是 64 位。对于 64 位系统，可以直接使用 Linux 64 位二进制包；对于 Linux 32 位系统，则必须手动编译 SAC 源码。

考虑到在申请提交之后，需要人工审核，两三个工作日后才会通过邮箱获取 SAC 软件包，建议还是同时申请 Linux 64 位二进制包和 SAC 源码包。

安装依赖包

Linux 下安装软件最麻烦的一个问题就是软件之间的依赖关系。

对于 Ubuntu/Debian 系[1]：

```
$ sudo apt-get install build-essential
$ sudo apt-get install libncurses5-dev libsm-dev libice-dev \
        libxpm-dev libx11-dev zlib1g-dev
```

对于 CentOS/RHEL 系：

```
$ sudo yum groupinstall 'Development Tools'
$ sudo yum install glibc ncurses-devel libSM-devel libICE-devel \
        libXpm-devel libX11-devel zlib-devel
```

安装

安装的方法在这一步分成两个部分，分别是二进制包安装和源代码编译，根据实际情况二者选一。

二进制包安装

解压 sac 二进制包：

```
$ tar -zxvf sac-101.6a-linux_x86_64.tar.gz
```

复制 sac 文件夹到安装目录 (推荐安装目录为`/usr/local`)：

```
$ sudo cp -r sac /usr/local
```

源代码编译

```
$ tar -zxvf sac-101.6a_source.tar.gz
$ cd sac-101.6a
$ ./configure --prefix=/usr/local/sac
$ make
$ sudo make install
```

配置变量

向`~/.bashrc`中加入如下语句以配置环境变量和 SAC 全局变量：

```
1 export SACHOME=/usr/local/sac
2 export SACAUX=$SACHOME/aux
3 export PATH=$SACHOME/bin:$PATH
4
5 export SAC_DISPLAY_COPYRIGHT=1
```

[1]很久不用 Ubuntu，无法保证完全正确。

```
export SAC_PPK_LARGE_CROSSHAIRS=1
export SAC_USE_DATABASE=0
```

其中，

- SACHOME 为 SAC 的安装目录；
- SACAUX 中包含了 SAC 运行所需的辅助文件；
- PATH 为 Linux 系统环境变量；
- SAC_DISPLAY_COPYRIGHT 用于控制是否在启动 SAC 时显示版本和版权信息，一般设置为 1。在脚本中多次调用 SAC 时会重复显示版本和版权信息，影响脚本输出，因而一般将其值设置为 0，设置方法可以参考 “Bash 中调用 SAC” 和 “在 Perl 中调用 SAC” 中的相关内容；
- SAC_PPK_LARGE_CROSSHAIRS 用于控制震相拾取过程中光标的大小，在 “震相拾取” 一节会具体说明；
- SAC_USE_DATABASE 用于控制是否允许将 SAC 格式转换为 GSE 2.0 格式；一般用不到该特性，故而设置其值为 0；

执行以下命令使配置的环境变量生效：

```
$ source ~/.bashrc
```

启动 SAC

终端键入小写的 sac[1]，显示如下则表示 SAC 安装成功：

```
$ sac
 SEISMIC ANALYSIS CODE [11/11/2013 (Version 101.6a)]
 Copyright 1995 Regents of the University of California

SAC>
```

1.5 邮件组

邮件组是个好东西，有点我们熟悉的 QQ 群的味道。在加入了邮件组之后，如果你在使用 SAC 的过程中遇到问题，可以向这个邮件组的邮箱发送 HELP 邮件，该组内的所有成员都会收到你的邮件。如果某人知道答案，他或许就会给你回复。发现了 SAC 的 bug 也可以向这里报告，开发者会尽快给你回复的。当然问问题之前要思考[2]，提交 bug 的时候要详细指出 bug 是如何出现的，也可以给出代码或文件以使得开发者能够重现该 bug。

邮件组邮箱：sac-help@iris.washington.edu

订阅地址：http://www.iris.washington.edu/mailman/listinfo/sac-help

[1] Ubuntu 的源里有一个名叫 sac 的软件，是用来显示登录账户的一些信息；CentOS 的源里也有一个名叫 sac 的软件，是 CSS 语法分析器的 Java 接口。所以一定不要试图用发行版自带的软件包管理器安装 sac!!!

[2] 请阅读 < 提问的智慧 >http://www.wapm.cn/smart-questions/smart-questions-zh.html

第 2 章　SAC 基础

2.1　如何学习 SAC？

学习 SAC 最好的方式是找一个有经验且有耐心的人，让他/她给你演示 SAC 是如何工作的。如果没有这样一个人的话，那么你就需要打开终端从头开始自学。

我将 SAC 的学习过程分成三个阶段，下面列出了每个阶段的具体要求。普通用户需要达到“SAC 进阶”才能满足日常数据处理的要求。

SAC 初阶

1. 掌握 SAC 中最常用的命令，包括但不限于 `help`、`read`、`write`、`plot`、`quit`、`plotpk`、`listhdr`、`chnhdr`、`rmean`、`rtrend`、`bandpass`、`plot1`、`plot2`、`cut`、`fft`；
2. 理解地震数据处理流程，参见“SAC 数据处理”一章；
3. 了解 SAC 文件格式，掌握常见的 SAC 头段变量，理解 SAC 中的时间概念；
4. SAC 相关工具：saclst；

SAC 进阶

1. 掌握 SAC 的大部分命令，至少知道 SAC 可以实现哪些功能；
2. 了解 SAC 编程以及如何在脚本中调用 SAC，见第 6 、 7 章；
3. 掌握如何绘制精美的波形图，见第 5 章；
4. 学会在自己的程序中使用 SAC 提供的函数库，见第 8 章；

SAC 高阶

1. 了解 SAC 软件包的内部结构；
2. 自己写程序实现 SAC I/O 库；
3. 阅读 SAC 源码，了解命令的技术细节；
4. 向 SAC 贡献代码；

2.2　如何阅读本文档？

本文档的内容大体分为两个部分：教程部分和命令部分。命令部分详细的列出了 SAC 中的每一个命令的语法、参数以及一些技术细节，适合作为参考，在需要的时候查阅。

教程部分给出了很多日常数据处理的例子，初学者应该坐在计算机前，打开终端，键入[1]书中的例子，试着理解每一个步骤的原理以及结果。

在阅读教程的同时，应随时翻看相应命令的说明，在实践的过程中掌握基础命令的语法和用法。这样基本就完成了 SAC 初阶的要求。

[1] 严禁复制！不许偷懒！

在读完教程之后，应浏览 SAC 的几乎所有命令，并挑选其中感兴趣的一些进行尝试。此后，在平常的科研工作中经常使用 SAC，有了实践经验和对 SAC 的进一步认识之后，可以阅读文档中的进阶内容，达到 SAC 进阶的要求。

最后，如果对 SAC 的内部机理感兴趣，可以阅读 SAC 的源码，重新实现一些 SAC 底层的功能。

2.3 启动和退出

在终端键入`sac`以启动 SAC，显示如下版本号以及版权信息[1]：

```
$ sac
 SEISMIC ANALYSIS CODE [11/11/2013 (Version 101.6a)]
 Copyright 1995 Regents of the University of California

SAC>
```

其中，“`SAC>`” 是 SAC 程序特有的提示符。

退出 SAC：

```
SAC> quit
```

也可以使用`done`、`exit`命令退出 SAC，但不推荐。

一次完整的启动和退出称为一个 SAC 会话。

2.4 SAC 设计思想

SAC 的设计思想大概可以总结如下：

1. 每个信号[2] 被保存到单独的 SAC 格式数据文件中；
2. SAC 格式包含了描述数据特征的头段区和存储信号的数据区，参见 “ SAC 文件格式 ” 一章；
3. 将单个或多个[3] SAC 文件从磁盘读入内存；
4. 通过各种命令对内存中的数据进行操作；
5. 操作完毕，将内存中的数据写入到磁盘，可以覆盖原 SAC 文件或写入新文件中。

2.5 SAC 命令初探

2.5.1 SAC 命令长什么样？

一个完整的 SAC 命令一般由 “命令 + 选项 + 参数” 构成，其中命令必须有，选项和参数可以成对出现，也可以只出现其中一个。命令、选项以及参数之间用空格分开。如果要将多个命令写在一行，要用分号隔开每个命令。例如：

```
SAC> funcgen random delta 0.1 npts 1000
SAC> rmean; rtrend; taper                // 一行内多个命令用分号隔开
SAC> write rand.SAC
```

[1] Livermore 实验室由 University of California 于 1952 年创立，2007 年改由 University of California、Bechtel National、BWX Technologies、Washington Group International 共同组成的安全机构管理。

[2] 信号，或称之为 trace，即**单个**台站**单个**仪器**单个**分量的数字记录。

[3] 一次性最多处理 **1000** 个任意大小的文件，记住 1000 这个值！

其中，`funcgen`、`write`、`rmean`、`rtrend` 和 `taper` 是命令；`random` 是选项；`0.1` 是选项 `delta` 的参数、`1000` 是选项 `npts` 的参数；而 `rand.SAC` 则是一个无选项的参数[1]。

Tips

官方文档的原文是 “command”、“keyword” 和 “option”，本文档 v2.0 中译为 “命令”，“关键字” 和 “参数”。个人感觉，无论是官方的用词还是 v2.0 版的译词都很容易让使用 C 语言和 Linux 的人困惑，因而 v3.0 中一律将其改为命令（command）、选项（option）和参数（argument）。这里解释一下选项（option）和参数（argument）的区别。一个命令有哪些选项是由命令规定的，其控制了命令的一些特性，因而选项的作用是告诉命令 “我**要**改某个特性”。但是具体**怎么**改呢？这个就交给参数来控制了。命令或选项只规定了参数的类型（整型、浮点型、字符串、枚举型或者逻辑型），用户需要根据自己的需求给定参数值。

2.5.2　大小写

SAC 的命令和选项都是不区分大小写的，这意味着你可以根据自己的喜好使用 `funcgen` 或者 `FUNCGEN`，SAC 在解释命令前都会将其转换为大写字母。

需要注意的是，由于 Linux 本身是区分大小写的，所以对于出现在参数中的文件名、目录名或者由引号包围的字符串来说，大小写是完全不同的。比如 `rand.SAC` 和 `RAND.SAC` 是两个完全不同的参数。

2.5.3　命令简写

SAC 的大多数命令以及选项都有简写形式。比如上面的命令简写形式如下：

```
SAC> fg r d 0.1 n 1000
SAC> rmean; rtr; taper
SAC> w rand.SAC
```

命令和选项究竟可以简写成怎样的形式，是由 SAC 自身规定的。简写的好处在于，在不产生歧义的前提下尽量减少用户的击键数；坏处在于，若对命令不是足够熟悉，简写后的命令变得很难读和难理解。比如你一看就知道 `delta` 代表的是采样周期[2]，而 `d` 却不那么直观，可能是 `delta`，也可能是 `demon`。所以，简写很好用，但是应该仅用在那些常使用的命令上，不要滥用。

2.5.4　查看命令语法

SAC 自带了英文的帮助文档，详细解释了每个命令的语法，可以通过 `help` 命令查看相应文档:

```
SAC> help funcgen write   // 命令的简写是h fg w
```

也可以直接查看 `$SACHOME/aux/help` 下的文档，或者查看本文档的命令部分。

2.5.5　参数默认值

为了让 SAC 易学易用，几乎所有命令参数都有一个 “系统默认参数值”，这些 “系统默认参数值” 都是经过精心挑选的，同时用户又可以随时修改参数值。这样的设计使得 SAC 易用同时又不失灵活性。

下面以 C 语言为例做一些说明[3]，希望能够帮助理解 SAC 参数的一些特点。

在 C 语言中，函数有主函数和子函数之分，变量又有全局变量和局部变量之分。所有的变量都可以被初始化为适当的值。

[1] 其实可以有很多选项，这里都省略了。
[2] 也称为采样时间，即两次数据采样的时间间隔，本文档将统一使用 “采样周期”。
[3] 有些地方不是很准确。

任意一个子函数，都可以使用全局变量的值，即函数的执行可以被全局变量所控制；同时也可以修改全局变量的值，这使得代码的管理和调试变得困难。一种实际的做法是定义专门的子函数来修改全局变量。

任意一个子函数，又都有自己的局部变量。这些局部变量在每次子函数被调用时都会被定义、初始化、使用和赋值，一旦子函数调用结束，变量即被撤销。如果给这些变量加上 `static` 修饰符，则这些局部变量变身为静态局部变量。

静态局部变量，会在程序刚开始的时候就完成初始化，也是唯一的一次初始化。静态局部变量仅在定义它的子函数里可见，子函数可以任意修改静态局部变量的值，但是每次子函数调用结束时变量不会被撤销，因此再次调用一个子函数时，静态局部变量的值可能已经被上一次的子函数调用所修改。

SAC 中有与之相对应的一些概念。sac 就是一个主函数，每一个命令都是一个子函数。所以 SAC 命令可以分为 2 类：

操作执行类 对数据进行某些操作（受全局变量控制，同时又有自己的静态局部变量）；
参数设定类 改变 SAC 的全局参数值（即 C 语言中专门用于修改全局变量的子函数）；

在启动 SAC（主函数）的时候，所有的选项（C 语言中的全局变量和静态局部变量）都会被初始化为指定的 “系统默认参数值”（全局变量和静态局部变量的唯一一次初始化）。

使用参数设定类命令的时候，其修改了 SAC 的全局参数，会影响接下来与之相关的所有其它命令的执行效果。使用操作执行类命令的时候，在命令中设定参数，相当于修改静态局部变量的值，不仅会影响当前命令的执行，也会影响之后所有同名命令的执行。

即：

当你在某个命令中为某个选项指定了一个参数值的时候，该参数值会成为该命令的该选项的 “参数当前值”，该 “参数当前值” 即成为接下来所有该命令的该选项的 “当前默认值”。

鉴于 SAC 的这样一个特性，在一次会话中，多次执行同一个命令时，一定需要注意选项的当前值是多少，因为这可能会影响到后面的一系列结果，这个必须理解和牢记！

Tips

当你在一次会话中执行了很多个命令的时候，SAC 参数可能已经被弄得一片混乱，你可以使用 `inicm` 命令在不退出 SAC 的情况下重新初始化。

下面用例子解释一下：

```
SAC> funcgen
SAC> plot
SAC> funcgen step delta 0.1 npts 1000
SAC> plot
SAC> funcgen boxcar
SAC> plot
```

1. `funcgen` 的默认值为 `funcgen impulse npts 100 delta 1.0 begin 0.`。
2. 第一个 `funcgen` 命令没有使用任何选项和参数，其直接使用系统默认值，生成一个脉冲数据，并保存到内存中。该数据的起始时间为 `0` ，采样周期为 `1.0` ，数据点数为 `100` 。
3. `plot` 命令会打开一个绘图窗口，并将内存中的数据绘制在窗口中；
4. 第二个 `funcgen` 命令生成了一个 step 函数[1] ，并设置其采样周期为 `0.1` ，数据点数为 `1000`。
5. `0.1` 和 `1000` 分别成为 `delta` 和 `npts` 的 “参数当前值”。
6. 第三个 `funcgen` 命令生成了 boxcar 函数，从绘图结果可以看出 `delta` 的值为 `0.1` ，`npts` 的值为 `1000` 。即继承了上一次命令的参数值。

[1]注意：内存中的脉冲函数已经没了。

2.6　文档约定

约定这个事情，说起来容易做起来难，遇到不符合约定的地方只能靠读者自己领悟了。

语法约定

1. 命令和选项使用大写字母，参数使用小写字母；
2. 命令和选项均使用全称，简写形式可省略的部分用灰色表示；
3. “[ ]” 表示该项为可选项；
4. “A|B|C” 表示 A、B、C 中任选一项；

示例如下：

```
BANDPASS [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n]
    [PASSES n] [TRANBW v] [ATTEN v]
```

示例约定

1. 命令、选项、参数均使用小写字母；
2. 常见的命令和选项均使用简写表示；
3. 含有提示符 “SAC>” 的行是用户键入的命令，无提示符的行是 SAC 输出行；
4. 示例中加入注释以帮助用户理解，注释使用了 C 语言的行注释符号 “//”；
5. 除非上下文说明，否则每个例子都运行在单独的 SAC 会话中，即每个命令都省略了启动 sac 和退出 sac 的命令。
6. 除特别情况外，均省略 `plot` 命令，用户应该学会随时 `plot` 以查看当前内存中的波形结果。

示例如下：

```
$ sac                           // 该行省略
SAC> fg seis                    // 这是注释
SAC> p                          // 该行省略
SAC> lh o

  FILE: SEISMOGR - 1
 --------------

     o = -4.143000e+01
SAC> q                          // 该行省略
```

2.7　样本数据

想要学习 SAC，手头必须有 SAC 格式的数据，SAC 提供了两个命令可以用于生成 SAC 格式数据，分别是 funcgen 和 datagen。

2.7.1　funcgen

`funcgen`（简写为 fg）表示 “function generator”，即该命令可以生成一些特定的函数，比如脉冲、阶跃、正弦等等，还可以生成一个地震波形样本。

```
SAC> fg impulse         // 生成脉冲函数
```

上面的命令生成了一个脉冲函数并存储在 SAC 的内存中，可以用命令 `plot`（简写为 p）在图形界面上查看这个函数的样子：

```
SAC> p
```

在学习 SAC 的过程中，`funcgen` 可以生成地震波形样本：

```
SAC> fg seismogram      // 生成地震波形样本，简写为fg seis
```

这个命令在 SAC 内存中产生了一个地震波形样本[1]，同时删除了内存中刚才生成的脉冲信号，可以使用 `plot` 命令查看地震波形。这个地震波形样本在以后的教程中经常用到。

2.7.2 datagen

`datagen`（简写为 dg）表示 “data generator”。顾名思义，就是用来生成数据的。

下面的例子在内存中生成了 CDV 台站记录到的一个近震的三分量波形数据[2]，并用 `plot1`（简写 p1）将三个波形画在一张图上：

```
SAC> dg sub local cdv.n cdv.e cdv.z
SAC> p1
```

简单解释一下，datagen 命令中，sub 为选项，其参数可以取 `local`、`regional`、`teleseism`，对应近震、区域震和远震，分别又可以使用不同的参数以读取不同台站的 SAC 数据。具体参考 `datagen`。

p1 会将内存中的所有文件（本例中是 3 个）同时绘制在一张图上，当内存中的文件数目很多时，需要修改 p1 的其它参数，以控制每张图上显示的波形数目。

2.8 SAC 的读和写

SAC 的读命令是 `read`（简写为 r），写命令为 `write`（简写为 w）。读和写是紧密联系的，所以把这两者放在一起讲。

注意：本节的所有示例都运行在同一个 SAC 会话中。

要演示如何读 SAC 文件，首先得有一些 SAC 数据才行，利用上一节的 datagen 生成一些数据。

```
$ ls                // 空文件夹
$ sac               // 启动一个SAC会话
 SEISMIC ANALYSIS CODE [11/11/2013 (Version 101.6a)]
 Copyright 1995 Regents of the University of California

SAC> dg sub local cdv.n cdv.e cdv.z     // 生成三个SAC数据
SAC> w cdv.n cdv.e cdv.z                // 将SAC数据写入磁盘
SAC> ls                                 // 在SAC中也可以使用一些常见的系统命令
cdv.e  cdv.n  cdv.z
```

有了数据之后，就可以练习如何去读了，在读数据之前，先说一说通配符的概念。

SAC 中，在指定文件名的时候，可以使用绝对路径，也可以使用相对路径。可以使用其全名，也可以使用通配符。SAC 的通配符与 Unix 定义的通配符一致，只包含如下三种：

1. “*” 匹配任意长度的字符串（包括零长度）；
2. “?” 匹配任意单个非空字符；
3. “[]” 匹配列表中的任意单一字符；
    - “[ABC]” 匹配单个字符 A 或 B 或 C
    - “[A,B,C]” 匹配单个字符 A 或 B 或 C
    - “[0-9]” 匹配任意一位数字
    - “[a-g]” 匹配从 a 到 g 范围内的任意单个字符

[1] 此命令的本质就是读取 `$SACAUX` 目录中的 `seismogram` 文件到内存。
[2] 此命令本质上就是从 `$SACAUX/datagen` 目录中读取 SAC 文件到内存。

下面的例子展示了如何读取 SAC 文件：

```
SAC> r cdv.n cdv.e cdv.z    // 读入三个文件，分别指定其文件名
SAC> r cdv.?                // 问号可以匹配单个字符。
./cdv.e ...cdv.n ...cdv.z   // 注意！这里文件读入的顺序与上一个命令不同
SAC> r cdv.[nez]            // 还可以这样读
./cdv.e ...cdv.n ...cdv.z
SAC> r *                    // 也可以这样读
./cdv.e ...cdv.n ...cdv.z
```

需要注意的是，SAC 在每次执行读取命令时，都会读入新的波形数据，并删除内存中原有的波形数据，所以经过上面四次 read 之后，内存中依然只有三个波形。

当然，read 也有选项，使得读取时将波形追加到内存中的波形数据集之后，而不替换内存中的原有波形：

```
SAC> r ./cdv.n              // one file       0 -> 1
SAC> r more ./cdv.e         // one MORE file  1 -> 2
SAC> r more ./cdv.z         // one MORE file  2 -> 3
```

将数据读入到内存中之后，对内存中的数据做一些处理，然后就需要将内存中的数据写回到磁盘中：

```
SAC> w test.n test.e test.z         // 分别写入到三个新文件中
SAC> w over                         // 覆盖磁盘原文件
SAC> w append .new                  // 在原文件名的基础上加上后缀".new"
cdv.e.new cdv.n.new cdv.z.new
SAC> ls
cdv.e  cdv.e.new  cdv.n  cdv.n.new  cdv.z  cdv.z.new  tesn.n  test.e  test.z
```

2.9　绘图

SAC 中有四个常用的绘图命令，分别是 `plot`、`plot1`、`plot2`、`plotpk`。

plotpk 用于“人机交互”过程中拾取震相，在“震相拾取”一节中会详细讲解。

下面说说 plot、plot1 和 plot2 的区别。

当内存中只有一个波形的时候，这三个绘图命令是没有区别的，这种情况下一般直接使用 plot；而当内存中存在多个文件时，这三个命令的区别就非常明显了。

2.9.1　plot

plot 命令会在单个图形窗口中显示单个波形。

```
SAC> r cdv.[nez]
SAC> p
Waiting
Waiting
SAC>
```

鉴于在 SAC 绘图中有很多中文意思类似的名词，这里似乎有必要定义一下“窗口”。图 2.1 展示了一个 SAC 窗口。同很多其它软件界面类似，这个窗口在左上角显示图标，右上角显示“最小化”、“最大化”、“还原”和“关闭”按钮。

左上角的“Graphics Window: 1”指明了当前绘图窗口的编号为“1”。SAC 中一共可以同时使用 10 个类似的窗口。

窗口的中间部分为真正的绘图区，以后的图将只显示绘图区而不显示整个窗口。

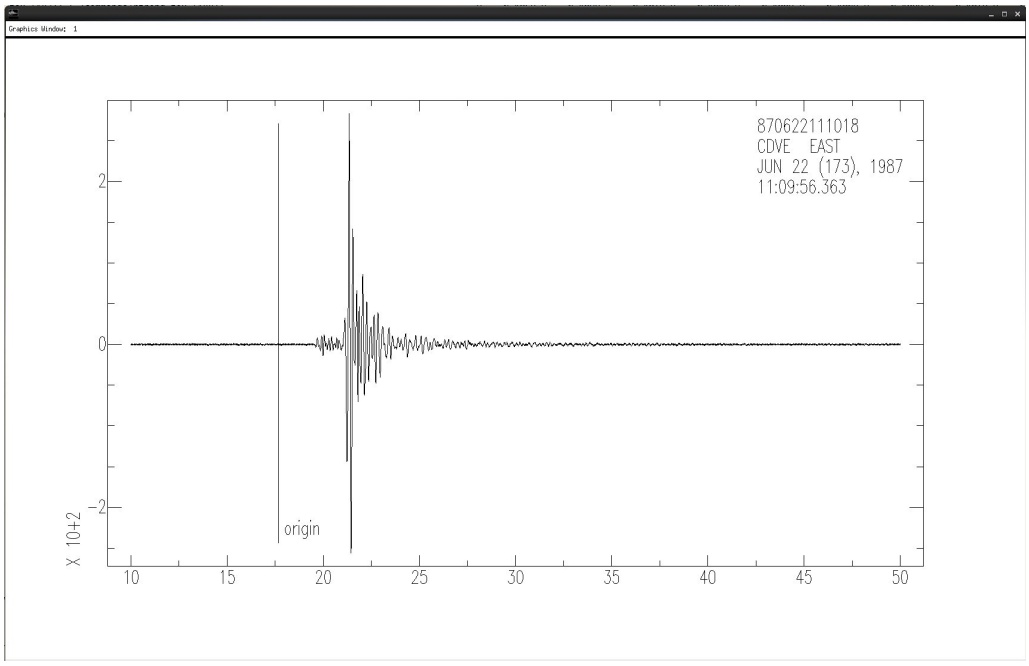

图 2.1: 绘图窗口

将三个波形数据读入内存，使用 plot 时，焦点位于绘图窗口，且绘图窗口上只显示第一个波形，终端中出现 “Waiting” 字样；将焦点切换[1]回终端，敲击回车键，绘图窗口中显示第二个波形，终端中出现第二个 “Waiting” 字样，焦点位于终端中；再次敲击回车键，窗口中显示第三个波形，焦点位于终端，由于已经没有更多的波形需要显示，此时终端中显示 SAC 提示符。

如果内存中还有波形在 “Waiting”，而你想要退出 plot，不想要再继续查看后面的波形，可以在终端中键入 “kill”（简写为 k），以直接退出 plot，如下例：

```
SAC> r cdv.[nez]
SAC> p
Waitingk
SAC>
```

也许你已经发现，即使 plot 结束或者中途退出 plot，绘图窗口依然没有被关闭，而且即便点击窗口的 “关闭” 按钮，窗口依然无法关闭。

```
SAC> r cdv.[nez]
SAC> begindevices xwindows      // 启动图像设备xwindows, 简写为bd x
SAC> p
Waiting
Waiting
SAC> enddevices xwindows        // 关闭图像设备xwindows, 简写为ed x
```

严格地说，SAC 绘图的流程应该是：启动图像设备（xwindows 或者 sgf）→ 绘图 → 关闭图像设备。这样稍显繁琐，SAC 将这一流程进行了简化，在每次绘图前偷偷启动了 SAC 默认的图像设备 xwindows，也就是上面所说的窗口，而关闭图像设备这一步需要用户自己完成，当然，在退出 SAC 时，SAC 也会自动关闭图像设备。

[1] Linux 下的快捷键是 Alt+Tab。

2.9.2　plot1

plot1 命令会在一个窗口中显示多个波形。这些波形共用一个 X 轴，但拥有单独的 Y 轴。

```
SAC> r cdv.?
cdv.e cdv.n cdv.z
SAC> p1
```

执行 plot1 命令后，焦点位于图形窗口，显示如图 2.2。

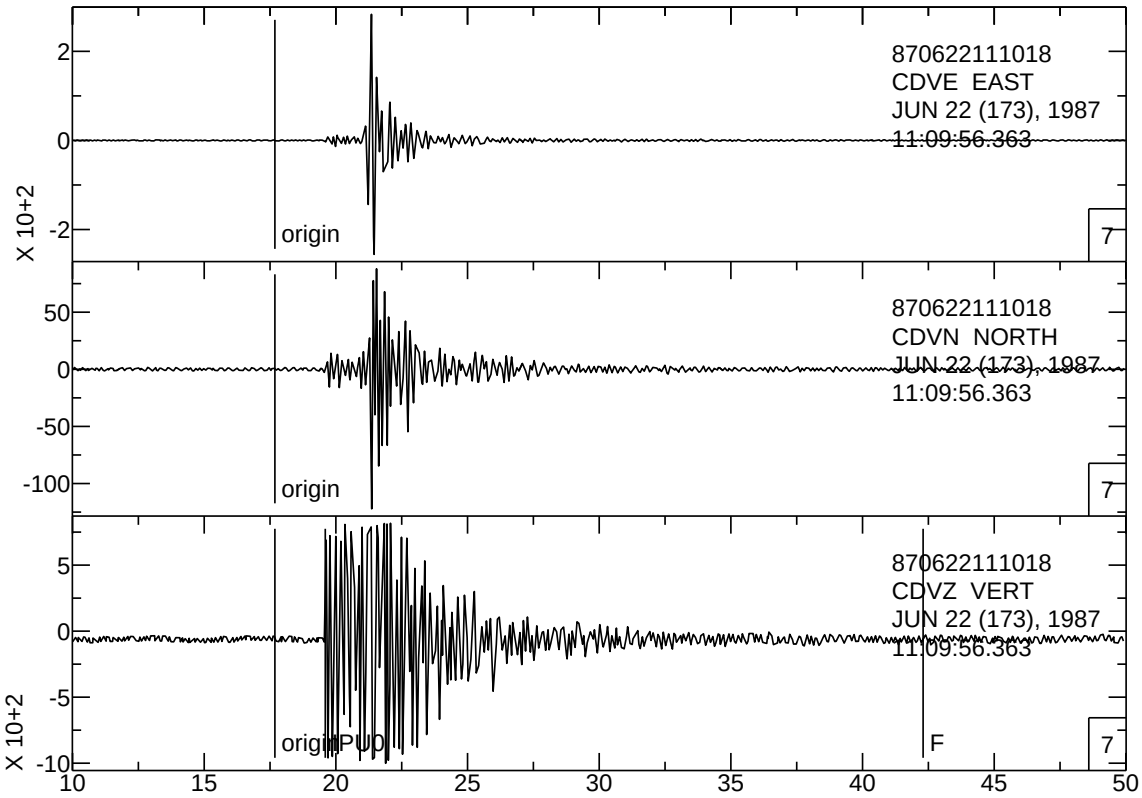

图 2.2: plot1 绘图效果

当一次性读入很多个地震数据时，如果直接使用 plot1 绘图，会导致一个窗口内有太多波形，反而什么都看不清，plot1 提供了额外的选项和参数以指定一个窗口内的最大波形数，多余的波形则处于等待状态。

```
SAC> dg sub local cdv.[enz] cvl.[enz] cvy.[enz]  // 生成9个地震波形
cdv.e cdv.n cdv.z cvl.e cvl.n cvl.z cvy.e cvy.n cvy.z
SAC> p1 p 3          // p是选项perplot的简写，3代表每次显示3个波形
Waiting
Waiting
SAC>
```

经常遇到的情况是，想要将每个台站的三分量波形记录放在一起看，这个时候设置选项 perplot 的参数值为 3 即可。

2.9.3　plot2

plot2 会在一个窗口内绘制多个波形，波形同时共用 X 轴和 Y 轴。

```
SAC> fg seis                        // 生成数据
SAC> rmean; rtrend; taper           // 预处理
SAC> w seis.0                       // 写入滤波前文件
```

```
SAC> bp c 0.05 10 n 4 p 2          // 滤波
SAC> w seis.1                      // 写入滤波后文件
SAC> r ./seis.[01]                 // 读入两个文件
./seis.0 ...seis.1
SAC> color red inc list red blue // 设置颜色
SAC> p2                            // 绘图
```

图 2.3 中红线为滤波前波形，蓝线为滤波后波形，二者共用 X 轴和 Y 轴。

plot2 适合做多个波形的对比，常用于数据处理前后波形对比或真实波形与合成波形间的对比。

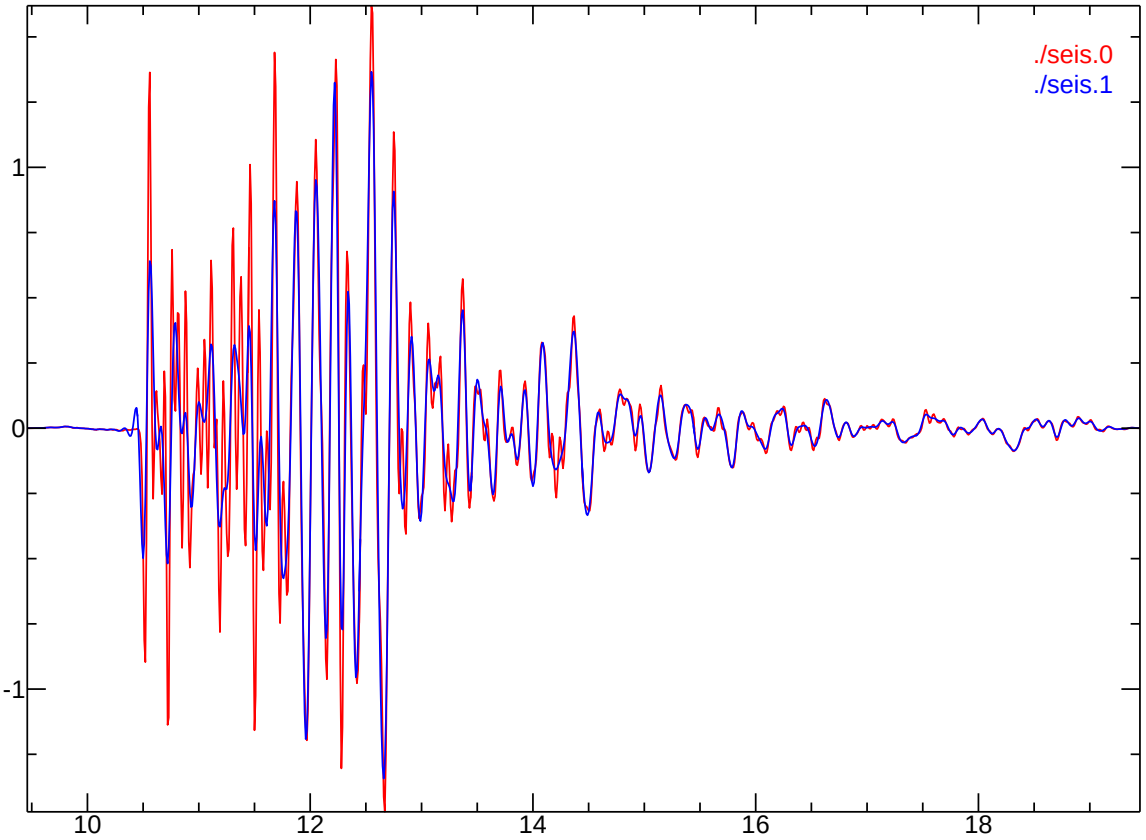

图 2.3: plot2 绘图效果。红色为滤波前波形，蓝色为滤波后波形。

第 3 章　SAC 文件格式

3.1　SAC 格式简介

一个地震波形数据包含了时间上连续的一系列数据点，数据点可以是等间隔或不等间隔采样。SAC 的数据格式要求一个文件中只包含一个地震波形数据，这样的定义更适合单个地震波形的处理。

每个 SAC 文件包含两个部分：一个头段区和一个/多个数据区。

头段区位于每个文件的起始处，其大小是固定的，用于描述数据的相关信息，比如数据点数、采样周期等等。

数据区紧跟在头段区之后，数据区的个数的规则大致如下：

- 如果数据是时间序列，且是等间隔采样的，则只有一个数据区，包含因变量（Y，也就是数据）的值，因变量（X）的信息可以直接从头段区中获得；
- 如果数据是时间序列，但是不等间隔采样的，则有两个数据区，分别包含自变量（X）和因变量（Y）的值；
- 如果数据是谱数据而非时间序列，则有两个数据区，分别包含振幅和相位或者实部和虚部；
- 如果数据是三维数据（XYZ），则包含三个数据，分别为 X 维度、Y 维度和 Z 值。

3.2　两种数据形式

SAC 文件格式有两种形式：二进制型和字符型[1]。

字符型与二进制型是完全等价的，只是字符型是给人看的，二进制型是给机器读写的。从 C 程序的角度来看，两者的区别在于，写文件时前者使用 `fprintf` 后者使用 `fwrite` 。

二进制型的 SAC 数据，占用更小的磁盘空间，读写速度更快，因而是最常用的 SAC 格式形式。

字符型适合于在文件出现问题时，临时查看文件内容的时候使用。

3.2.1　两种形式的互相转换

你是否想要一个字符型的 SAC 文件，用编辑器打开好好看看 SAC 数据究竟长什么样。SAC 自带的命令可以实现两种形式的转换 [2] 。

```
SAC> fg seis
SAC> w seis             // 先生成一个二进制型SAC数据，以做测试
```

将二进制型转换成字符型：

[1]原为 alphanumeric，译为文数字。

[2]也可以使用 SAC 的 `convert` 命令进行转换，不过此命令即将被淘汰。

```
SAC> r seis              // 读二进制型文件
SAC> w alpha seis.a      // 以字符型写入
```

将字符型转换成二进制型：

```
SAC> r alpha seis.a      // 读字符型文件
SAC> w sac seis.b        // 以二进制型写入，可以省略sac，写成w seis.b
```

试试用你最喜欢的文本编辑器打开字符型的 `seis.a` 吧，其内容如下：

```
     0.01000000      -1.569280       1.520640      -12345.00      -12345.00
       9.459999       19.45000      -41.43000       10.46400      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00       48.00000      -120.0000      -12345.00      -12345.00
       48.00000      -125.0000      -12345.00       15.00000      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
       373.0627       88.14721       271.8528       3.357465      -12345.00
      -12345.00    -0.09854718       0.000000       0.000000      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      1981        88        10        38        14
         0         6         0         0      1000
    -12345    -12345    -12345    -12345    -12345
         1        50         9    -12345    -12345
    -12345    -12345        42    -12345    -12345
    -12345    -12345    -12345    -12345    -12345
    -12345    -12345    -12345    -12345    -12345
         1         1         1         1         0
CDV      K8108838
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
    -0.09728001    -0.09728001    -0.09856002    -0.09856002    -0.09728001
    -0.09600000    -0.09472002    -0.09344001    -0.09344001    -0.09344001
    -0.09344001    -0.09344001    -0.09472002    -0.09472002    -0.09344001
    ......
```

第 1-30 行是头段区，31 及以后 N 行是数据区。目前你可能还看不懂头段区的这些数字或者字符代表什么。没关系，在下一节会详细介绍 SAC 头段区，记得一定要一边看下一节的内容，一边对照着这个例子，好好琢磨 SAC 的头段。

3.3 SAC 头段结构

SAC 头段区的大小为 632 个字节，由一系列头段变量构成。从这些头段变量中可以了解到波形记录的很多信息，比如台站经纬度、发震时刻、震相到时等等。

表 3.1 列出了 SAC 头段区的全部头段变量。

在源代码中，整个头段区被定义为一个结构体。结构体中的第一个头段变量是 `delta`，第二个头段变量是 `depmin`，第六个头段变量是 `b`，以此类推。表的第一列给出了当前行的第一个头段变量在文件中（或在结构体中）的起始字节。第二列给出了当前行的头段变量的变量类型码，包括 F、N、I、L、K 型，每个变量类型码的具体含义参见表 3.2。

头段区中，共有头段变量 133 个。若变量名为 `internal`，表示该变量为 SAC 内部使用的头段变量，用户不可对其进行操作。若变量名为 `unused` 表示该变量暂时尚未使用，为以后可能出现的新头段变量占位。

表 3.1: SAC 头段变量列表

Byte	Type	Names				
0	F	delta	depmin	depmax	scale	odelta
20	F	b	e	o	a	internal
40	F	t0	t1	t2	t3	t4
60	F	t5	t6	t7	t8	t9
80	F	f	resp0	resp1	resp2	resp3
100	F	resp4	resp5	resp6	resp7	resp8
120	F	resp9	stla	stlo	stel	stdp
140	F	evla	evlo	evel	evdp	mag
160	F	user0	user1	user2	user3	user4
180	F	user5	user6	user7	user8	user9
200	F	dist	az	baz	gcarc	internal
220	F	internal	depmen	cmpaz	cmpinc	xminimun
240	F	xmaximum	yminimum	ymaximum	unused	unused
260	F	unused	unused	unused	unused	unused
280	N	nzyear	nzjday	nzhour	nzmin	nzsec
300	N	nzmsec	nvhdr	norid	nevid	npts
320	N	internal	nwfid	nxsize	nysize	unused
340	I	iftype	idep	iztype	unused	iinst
360	I	istreg	ievreg	ievtyp	iqual	isynth
380	I	imagtyp	imagsrc	unused	unused	unused
400	I	unused	unused	unused	unused	unused
420	L	leven	lpspol	lovrok	lcalda	unused
440	K	kstnm	kevnm*			
464	K	khole	ko	ka		
488	K	kt0	kt1	kt2		
512	K	kt3	kt4	kt5		
536	K	kt6	kt7	kt8		
560	K	kt9	kf	kuser0		
584	K	kuser1	kuser2	kcmpnm		
608	K	knetwk	kdatrd	kinst		

表 3.2 列出了 SAC 头段中的全部头段变量类型及其相关信息。第一列为头段变量类型代码，第二类给出了其代表的头段变量类型，第三列指出 C 源码中该变量的是用什么类型定义的，第四列给出了每个变量所占据的字节数。第五列给出了写字符型 SAC 文件时的输出格式。最后一列则给

出该类型的未定义值。

表 3.2: 变量类型说明

Code	Type	C Type	sizeof	printf	未定义值
F	浮点型	float	4	%15.7f	-12345.0
N	整型	int	4	%10d	-12345
I	枚举型	int	4	%10d	-12345
L	逻辑型	int	4	%10d	FALSE
K	字符型	char*	8	%-8.8s	"-12345␣␣"
A	辅助型				

SAC 中定义了 6 种头段变量类型，分别为浮点型（F)、整型（N)、枚举型（I)、逻辑型（L)、字符型（K) 和辅助型（A)。其中辅助型不存在于 SAC 头段区中，其直接由其它头段变量推导得到。

除了 F 型变量外以外，其余所有头段变量的变量名均以变量类型码开头，比如 `nvhdr` 是 N 型变量，`leven` 是 L 型变量。

枚举型变量，本质上是 `int` 型，其只能在固定的几个值中取值；C 语言中本身是没有规定 Bool 类型的，SAC 自定义了 `TRUE` 和 `FALSE`，用于表示真和假。字符型变量长度为 8，只有 `kevnm` 很特殊，其长度为 16。

对于一个 SAC 文件，并非所有的头段变量都必须要包含有意义的值，若变量未定义称其包含未定义值，第 6 列给出了不同类型的未定义值的形式，比如如果一个整型头段变量的值为 -12345，则认为其处于未定义态。实际使用的时候，SAC 提供了参数 `undef`，其可以根据头段变量的类型自动转换成相应类型的未定义值。

3.4 SAC 头段变量

3.4.1 基本变量

nvhdr*

SAC 头段版本号。nvhdr[1]是 SAC 中很重要但是不太常用的头段变量。目前值为 6，旧版本的 SAC 文件（$nvhdr < 6$）在读入时会自动更新。

nzyear, nzjday, nzhour, nzmin, nzsec, nzmsec

分别表示 “年”、“一年的第几天”[2]、“时”、“分”、“秒”、“毫秒”[3]。这六个头段变量构成了 SAC 中唯一的绝对时刻，SAC 中的其它时刻都被转换为相对于该时刻的相对时间（单位为秒)。详细的信息参见第 3.5 节 “ SAC 中的时间概念 ” 中的介绍。

根据这六个头段还可以推导出其它一些辅助型变量：

- `kzdate`: 字符数字格式的参考日期，由 `nzyear` 和 `nzjday` 导出；
- `kztime`: 字符数字格式的参考时间，由 `nzhour`、`nzmin`、`nzsec`、`nzmsec` 导出；

如下例所示：

```
SAC> fg seis
SAC> lh nzyear nzjday nzhour nzmin nzsec nzmsec
```

[1]星号表示该头段变量在 SAC 中必须有定义值，下同。
[2]使用 jday 而不是 “month+day” 可以少用一个头段变量。
[3]$1s = 1000ms$

```
 FILE: SEISMOGR - 1
--------------

    nzyear = 1981
    nzjday = 88
    nzhour = 10
     nzmin = 38
     nzsec = 14
    nzmsec = 0
SAC> lh kzdate kztime

 FILE: SEISMOGR - 1
--------------

    kzdate = MAR 29 (088), 1981
    kztime = 10:38:14.000
```

iztype

等效参考时刻。SAC 的参考时刻是可以任意指定的，但一般选取某个特定的时刻（比如文件起始时刻、发震时刻等等）作为参考时刻。其可以取如下枚举值：

- IUNKN：未知
- IB：以文件开始时刻为参考时间
- IDAY：以参考日期当天的午夜作为参考时间
- IO：以事件发生时间为参考时间
- IA：以初动到时为参考时间
- ITn：以用户自定义的时间为参考时间（n 可取 0-9）

若 iztype=IO，则表示其值以发震时刻作为参考时刻，此时头段变量 o 的值应为 0。

iftype*

SAC 文件类型，其决定了头段区之后有几个数据区。该头段变量为枚举类型[1]，可以取如下几个枚举值[2]：

- ITIME：时间序列文件（即 Y 数据，一般的地震波形数据）
- IRLIM：频谱文件（实部 -虚部格式）
- IAMPH：频谱文件（振幅 -相位格式）
- IXY：一般的 X-Y 数据
- IXYZ：一般的 XYZ（3-D）文件

idep

因变量（Y）类型，该头段变量可以不定义，其可以取如下枚举值：

- IUNKN：未知类型
- IDISP：位移量，单位为 nm
- IVEL：速度量，单位为 nm/s
- IVOLTS：速度量，单位为 $volts$[3]
- IACC：加速度量：单位为 nm/s^2

[1]枚举型在 C 源码中实际为“#define ITIME 1”的形式。
[2]枚举值一律使用大写，且以 I 开头。
[3]不解

3.4.2 数据相关变量

npts*

数据点数，其值决定了在数据区有多少个数据点。

delta*

等间隔数据的数据点采样周期（标称值）。

odelta

采样周期的实际值，若实际值与标称值不同则有值，一般来说都是未定义的。

b*, e*

文件的起始时间和结束时间（相对于参考时刻的相对时间）。

leven*

若数据为等间隔则为 TRUE，否则为 FALSE。

depmin, depmax, depmen

因变量（Y）的最小值、最大值和均值。一般地，这几个头段会在读/写 SAC 文件时、数据处理过程中被自动计算。

scale

因变量比例因子，即真实物理场被乘以该比例因子而得到现有数据。比如真实物理场的 Y 值大概在 10^{-20} 量级，数据处理起来很不方便，这个时候可以设置 scale=10^{20}，将真实物理场的量级修改到 10^{1}。

xminimum, xmaximum, yminimum, ymaximum

仅用于 3D（XYZ）文件中，记录 X 和 Y 的最小/大值。

nxsize, nysize

仅用于 3D（XYZ）文件中，表示 X 和 Y 方向的数据点数。

iqual†

iqual[1] 标识数据质量，可取如下值：

- `IGOOD`：高质量数据；
- `IGLCH`：数据中有毛刺（`glitches`）；
- `IDROP`：数据有丢失（`Dropouts`）；
- `ILOWSN`：低信噪比数据；
- `IOTHER`：其它；

isynth†

合成数地震图标识。

- `IRLDTA`：真实数据；
- `??????`：其它合成地震图代码相应的标识；

3.4.3 事件相关变量

kevnm

事件名。在头段区，其占据 16 个字节。

[1]† 标识仅表示 SAC 程序内部未使用该头段变量，即变量有值或者无值、有何值，对于程序的运行不会产生任何影响，但用户可以在自己的程序中自由使用这些头段变量。下同。

evla, evlo, evel, evdp

分别代表事件的纬度（-90 到 90）、经度（-180 到 180）、高程（单位为 m，未使用）和深度（单位为 km，以前为 m）。

ievreg†

事件地理区域[1]。

ievtyp

事件类型，这里仅列出部分常见的枚举值：

- IUNKN：未知事件
- INUCL：核事件
- IEQ：地震
- IOTHER：其它

mag

事件震级。

imagsrc

震级信息来源，可以取如下枚举值：

- INEIC：NEIC
- IPDE：PDE
- IISC：ISC
- IREB：人工检查过的事件目录
- IUSGS：USGS
- IBRK：UC Berkeley
- ICALTECH：California Institute of Technology
- ILLNL：Lawrence Livermore National Laboratory
- IEVLOC：Event Location
- IJSOP：Joint Seismic Observation Program
- IUSER：The individual using SAC2000
- IUNKNOWN：未知

imagtyp

震级类型，取如下枚举值：

- IMB：体波震级
- IMS：面波震级
- IML：Local 震级
- IMW：矩震级
- IMD：持续时间震级
- IMX：用户自定义震级

gcarc, dist, az, baz

gcarc 全称 Great Circle Arc，即震中到台站的大圆弧的长度，单位为度；
dist 震中到台站的距离，单位为千米；
az 方位角，震中到台站的连线与地理北向的夹角；
baz 反方位角，台站到震中的连线与地理北向的夹角。

[1] Flinn-Engdahl Regions:http://en.wikipedia.org/wiki/Flinn-Engdahl_regions

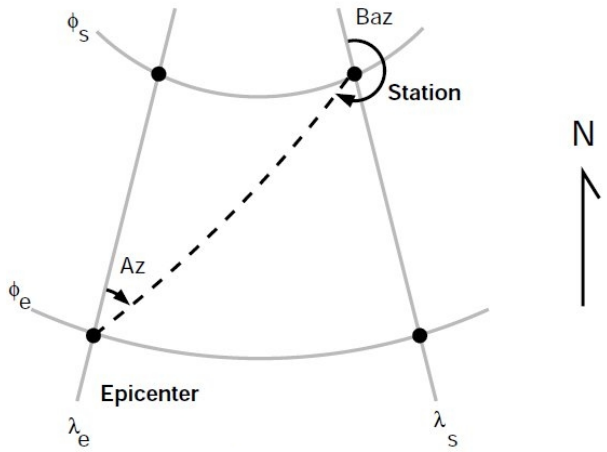

图 3.1: 震中距、方位角、反方位角示意图。图片来自于 Prof. Robert B. Herrmann 的课程讲义。

震中距、方位角和反方位角的计算涉及到球面三角的知识，具体细节可以参考该讲义: http://www.eas.slu.edu/People/RBHerrmann/Courses/EASA462/ASSIGNMENTS/Ass07/Ass07.pdf。

在不严谨的情况下，可以认为 dist = gcarc*111.195，更严谨的计算方法可以参考相关文献。

知道震中和台站的位置，计算震中距、方位角和反方位角是一个基本问题，有很多代码均实现了此功能，列举如下:

- http://www.seis.sc.edu/software/distaz/
- SAC 源码 `src/ucf/distaz.c`
- CPS330[1] 源码 `VOLI/src/udelaz.c`

`o, ko`

o 为事件的发生时刻相对于参考时刻的秒数。ko 是绘图时变量 o 的标识符。

`khole`

若为核爆事件，则其为孔眼标识；若为其它事件，则为位置标识。

`nevid, norid, nwfid`

三者分别标识事件 ID、起始时间 ID 和波形 ID，仅用于 CSS 3.0 文件中。CSS 3.0 是 SAC 可以处理的一种数据格式，应该是当初 SAC 商业化的产物，目前仍保留在 SAC 头段中。

3.4.4 台站相关变量

`knetwk`

地震台网名。

`kstnm`

台站名。

`istreg`†

台站地理区域。

[1] http://www.eas.slu.edu/eqc/eqccps.html

stla, stlo, stel, stdp

台站纬度（-90 到 90 度）、经度（-180 到 180 度）、高程（单位 m，目前未使用）、相对地表的深度（单位 m，目前未使用）。

cmpaz, cmpinc, kcmpnm, kstcmp

一个台站至少需要三个正交的通道（或分量）才能完整地记录地面运动物理量。cmpaz 和 cmpinc 指定了单个通道记录的方向矢量。

图 3.2 给出了 SAC 所使用的 NEU 坐标系，需要注意的是这是一个左手坐标系。图中蓝色箭头为通道所记录的方向矢量，若地面运动与该方向一致，则为正，否则为负。其中，头段变量 cmpaz 表征通道的方位角，其定义为从 N 向开始顺时针旋转的角度，即图中的角度 ϕ；cmpinc 表征通道的入射角，定义为 U 向向下旋转的度数，即图中的角度 θ。

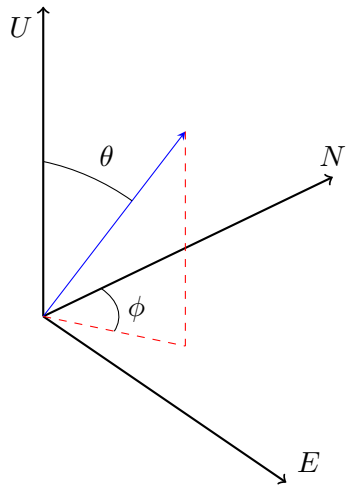

图 3.2: cmpaz 和 cpminc 示意图

根据定义，地震仪标准通道的 cmpinc 和 cmpaz 值如下表：

表 3.3: 标准地震通道的 cmpaz 和 cpminc

方向	CPMAZ	CPMINC
N	0	90
E	90	90
U	0	0

对于非标准方向的地震通道来说，很容易根据 cmpinc 和 cmpaz 的值，将其旋转到 NEU 坐标系或者 RTZ 坐标系，这些将在 “ 分量旋转 ” 一节中说到。

kcmpnm 为分量名称，SEED 格式规定使用三个字符中的最后一个字符代表通道的方位（如 BHE 代表东西向分量）。对于水平分量，目前更趋向于使用 1 和 2 代替 N 和 E。

kstcmp 为辅助型变量，表示台站分量，由 kstnm、cmpaz、cmpinc 推导得到。

lpspol

如图 3.2 所示，SAC 使用左手坐标系 NEU，若台站三通道的正极性分别与 NEU 三方向相同，则为真，否则为假。

3.4.5 震相相关变量

a, ka, f, kf

a 和 f 分别为事件的初动时刻和结束时刻相对于参考时刻的秒数。ka 和 kf 则是相应的时间标识。

tn, ktn

用户自定义的时刻，n 可以取 0-9，常用于震相拾取，ktn 为相应的时间标识。比如定义 t1 为 PcP 震相的到时，kt1 则一般定义为 “PcP”。

3.4.6 仪器相关变量

kinst, iinst†, respn†

kinst 为记录仪器的通用名称，iinst 为记录仪器的类型，respn 为仪器相应参数。

3.4.7 其它变量

usern, kusern

usern（n=0-9）和 kusern（n=0-2）均用来存储用户自定义值。不像 a 和 ka 那样成对存在，usern 和 kusern 之间没有对应关系。比如可以将波形的信噪比保存在 user0 中，将任意字符串保存到 kuser0 中。

lovrok

若为真，则数据可覆盖；若为假，则数据不可被覆盖。主要用于保护原始数据，一般来说很少用到，若是出于保护原始数据的目的，应优先考虑对原始数据做备份。

lcalda

全称为 Calculate Dist and Azimuth，若为真，则在事件和台站的坐标被写入或被修改时，dist、gcarc、az、baz 将自动计算。

kdatrd

数据被读入计算机的日期。

3.5 SAC 中的时间概念

3.5.1 基本思路

SAC 的头段区有很多与时间相关的头段变量，包括 nzyear、nzjday、nzhour、nzmin、nzsec、nzmsec、b、e、o、a、f、tn（n=0-9），正确使用它们的前提是理解 SAC 中的时间概念。这一节将试着说清楚这个问题。

首先，SAC 处理的是地震波形数据，SAC 格式里保存的是时间序列数据。先不管其它的一些台站经纬度、事件经纬度信息，就数据而言，至少需要一系列数据值以及每个数据值所对应的时刻。

在本节接下来的内容中，将严格区分两个高中物理学过的概念：时刻和时间。简单地说，在时间轴上，时刻是一个点，时间是一个线段。

一个简单的例子如下：

```
2014-02-26T20:45:00.000     0.10
2014-02-26T20:45:01.000     0.25
2014-02-26T20:45:02.000     0.33
2014-02-26T20:45:03.000     0.21
2014-02-26T20:45:04.000     0.35
2014-02-26T20:45:05.000     0.55
2014-02-26T20:45:06.000     0.78
2014-02-26T20:45:07.000     0.66
2014-02-26T20:45:08.000     0.42
2014-02-26T20:45:09.000     0.34
2014-02-26T20:45:10.000     0.25
```

其中第二列是数据点，每个数据点所对应的时刻放在第一列，格式为 “yyyy-mm-ddThh:mm:ss.xxx”。数据点是以 1s 的等间隔进行采样的。

若把这堆时刻以及数据点直接写入文件中，将占据大量的磁盘空间，读写也很不方便。考虑将某一个时刻定义为参考时刻，并把其它所有的时刻都用相对于该参考时刻的时间来表示。这样可以简化不少。

比如取 “`2014-02-26T20:45:00.000`” 为参考时刻，即

```
nzyear = 2014
nzjday = 57
nzhour = 20
nzmin  = 45
nzsec  = 00
nzmsec = 000
```

则上面的数据可以简化为

```
00.000     0.10
01.000     0.25
02.000     0.33
03.000     0.21
04.000     0.35
05.000     0.55
06.000     0.78
07.000     0.66
08.000     0.42
09.000     0.34
10.000     0.25
```

其中第二列是数据点，第一列是每个数据点对应的时刻相对于参考时刻的相对时间，下面简称其为相对时间。

显然参考时刻的选取是任意的，若取 “`2014-02-26T20:45:05.000`” 为参考时刻，则上面的数据简化为

```
-05.000     0.10
-04.000     0.25
-03.000     0.33
-02.000     0.21
-01.000     0.35
 00.000     0.55
 01.000     0.78
 02.000     0.66
 03.000     0.42
```

```
 04.000     0.34
 05.000     0.25
```

一般来说，会选取一个比较特殊的时刻作为参考时刻，比如第一个数据点对应的时刻，或者地震波形数据中的发震时刻。

下面还是回到以 “ `2014-02-26T20:45:00.000` ” 为参考时刻简化得到的结果。因为数据是等间距的，相对时间这一列完全可以进一步简化，比如用 “起始相对时间 + 采样间隔 + 数据点数” 或者 “起始相对时间 + 采样间隔 + 结束相对时间” 就完全可以表征第一列的相对时间。如果只能从二者之中选一个的话，我会选择第一种，毕竟 “数据点数” 太有用了。

SAC 选择了另外一种简化模式，“起始相对时间 + 采样间隔 + 数据点数 + 结束相对时间”，即头段变量中的 “`b+delta+npts+e`” ，这其实是存在信息冗余的，这就造就了头段变量 `e` 的一些特殊性，后面会提到。

按照 SAC 的模式在对相对时间进行简化之后，整个数据可以表示为

```
nzyear = 2014
nzjday = 57
nzhour = 20
nzmin  = 45
nzsec  = 00
nzmsec = 000
b      = 0.0
e      = 10.0
delta  = 1.0
npts   = 11

0.10
0.25
0.33
0.21
0.35
0.55
0.78
0.66
0.42
0.34
0.25
```

似乎到这里就结束了。

地震学里的一个重要问题是拾取震相到时（时刻），所以还需要几个额外的头段变量来保存这些震相到时（时刻），不过显然我们不会真的把时刻保存到这些头段变量中，不然上面的一大堆就真是废话了。SAC 将震相到时（时刻）相对于参考时刻的时间差（即相对时间）保存到头段变量 o、a、f、tn 中。

综上，SAC 中跟时间有关的概念有三个：

参考时刻 由头段变量 nzyear、nzjday、nzhour、nzmin、nzsec、nzmsec 决定；
相对时间 即某个时刻相对于参考时刻的时间差（单位为秒），保存到头段变量 b、e o、a、f、tn；
绝对时刻 = 参考时刻 + 相对时间；

3.5.2　在测试中学会领悟

下面以一个具体的数据为例，通过修改各种与时间相关的头段来试着去进一步理解 SAC 的时间概念。

生成样例数据

```
SAC> fg seis
SAC> lh iztype

  FILE: SEISMOGRAM - 1
   ------------

    iztype = BEGIN TIME
SAC> ch iztype IUNKN
SAC> w seis
```

lh 是命令 `listhdr` 的简写，用于列出头段变量的值。ch 是 `chnhdr` 的简写，用于修改头段变量的值。

这里额外多做了一个操作修改 iztype 的操作，这是由于这个数据稍稍有一点 bug。

iztype 指定了参考时刻的类型，其显示为 BEGIN TIME，实际上其枚举值是 IB，也就是说这个数据选取文件第一个数据点的时刻作为参考时刻，那么 b 的值应该为 0。而实际上这个数据的 b 值并不为 0，这其实是这个数据的一点小 bug。这也从另一个侧面说明 SAC 只有在修改与时间相关的头段变量时才可能会检查到这个错误/警告，所以这里先将其修正为 IUNKN。

修改文件起始时间 b

```
SAC> r seis
SAC> lh kzdate kztime b delta npts e o a f

  FILE: seis - 1
 --------------

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
          a = 1.046400e+01
SAC> ch b 10
SAC> lh

  FILE: seis - 1
   ----------

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 1.000000e+01
      delta = 1.000000e-02
       npts = 1000
          e = 1.999000e+01
```

```
          o = -4.143000e+01
          a = 1.046400e+01
```

修改 b 前后的变化仅在于 b 和 e 值的变化，而参考时刻以及其它相对时间并没有发生变化。

这意味着整段 SAC 数据中的任意一个数据点所对应的时刻[1]都向后延迟了 0.54 秒！这样做很危险，因为 b 和 e 的绝对时刻被修改了，而其它头段如 o、a、f、tn 的绝对时刻却没有变。

使用的时候必须非常小心：

- 如果 o、a、f、tn 都没有定义，那么修改 b 值可以用于校正仪器的时间零飘[2]以及时区差异[3]。
- 如果 o、a、f、tn 已经被定义，则修改 b 值会导致与震相相关的头段变量出现错误！[4]

修改文件结束时间 e

```
SAC> r ./seis
SAC> lh kzdate kztime b delta npts e o a f

  FILE: ./seis - 1
 ------------

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
          a = 1.046400e+01
SAC> ch e 0
SAC> lh

  FILE: ./seis - 1
 ------------

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
          a = 1.046400e+01
```

可以看到，修改前后所有变量均没有发生变化，即 e 的值是不可以随意改变的，根据上面的结果可知，e 的值是通过 b、delta、npts 的值动态计算的。这也与上一节说到的头段

[1] 好长的修饰语

[2] 零飘，即仪器中的时刻与标准时刻不同。

[3] 时区差异可以理解成另一种零飘。

[4] 如果只定义了 o 值，或者 a、f、tn 为理论震相到时而非计算机拾取或人工拾取的到时，修改 b 也是没有问题的。有些乱，不多说了。

变量冗余问题相符合。不要试图修改 delta、npts，这不科学！

修改 o、a、f、tn

这几个头段变量完全是由用户自定义的，因而任何的定义、修改、取消定义都不会对数据的正确性产生影响，因而这里不再测试。

修改参考时间

```
SAC> r ./seis
SAC> lh kzdate kztime b delta npts e o a f

  FILE: ./seis - 1
 ------------

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:14.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
          a = 1.046400e+01
SAC> ch nzsec 15
SAC> lh

  FILE: ./seis - 1
 ------------

     kzdate = MAR 29 (088), 1981
     kztime = 10:38:15.000
          b = 9.459999e+00
      delta = 1.000000e-02
       npts = 1000
          e = 1.945000e+01
          o = -4.143000e+01
          a = 1.046400e+01
```

试图修改参考时刻，整个 SAC 头段，除了参考时刻外其它时间变量都没有发生变化。根据 “绝对时刻 = 参考时刻 + 相对时间” 可知，这导致所有 SAC 数据点的绝对时刻发生了平移，这一点理论上可以用于校正零飘或者时区，但是由于 SAC 不支持智能判断时间（比如不知道 1 时 80 分实际上是 2 时 20 分），所以修改时区时需要获取参考时刻 6 个头段变量，加上时区的校正值，再写入到参考时刻 6 个变量中，相对较为繁琐。

修改发震时刻

数据处理中一个常见的需求是修改发震时刻，这可以通过修改头段变量 o 来实现，但是经常需要将参考时刻设置为发震时刻。上面的测试表明，直接修改参考时刻是很危险的，所以 SAC 的 ch 命令提供了 `allt` 选项来实现这一功能，在 “事件信息” 一节中会具体解释。

3.5.3　总结

将 SAC 中的时间变量分为三类：

1. 参考时刻：即 nzyear、nzjday、nzhour、nzmin、nzsec、nzmsec；

2. 相对时间：即 o、a、f、tn；
3. 特殊的相对时间：即 b[1]；

第二类时间变量可以随意修改，即震相拾取。

第一、三类时间变量的修改会导致数据绝对时刻发生改变。若小心使用，可用于校正时间零飘或时区不一致问题。

而为了修改发震时刻（此时应保证数据的绝对时刻不发生改变），则需要使用 ch 提供的 allt 选项来实现。

[1]由于 e 不可独立修改，所以不再考虑

第 4 章　SAC 数据处理

地震数据处理大概是这样一个流程：“数据组织”→“数据预处理”→ “人机交互”→“数据分析”→“绘制图件”。

本章将介绍数据组织的一些原则以及如何利用 SAC 完成数据预处理和人机交互。

4.1　数据获取与转换

地震波形数据的获取途径很多，列举如下：

1. IRIS：www.iris.edu
2. Hi-net：www.hinet.bosao.go.jp
3. GEOFON：geofon.gfz-potsdam.de

数据的传输介质主要是网络传输为主，也有以移动硬盘、光盘或 U 盘为传输介质的。

最常见的数据交换格式是 SEED 格式，用于储存多台站多分量的连续波形数据以及台站相关元信息。SEED 格式本质上是一个压缩文件，因而可以大大减少网络传输数据量以及硬盘空间。除了 SEED 格式之外，miniSEED 格式仅包含连续波形数据，而 dataless SEED 格式仅包含台站元信息。

IRIS 提供了 rdseed 软件，用于读取 SEED 格式，并将其中的波形数据解压成为多种地震数据格式。Hi-net 提供了 win32_tools，用于将 win32 格式的数据文件解压成为 SAC 格式。

4.2　数据重命名

从压缩数据格式中解压得到的 SAC 数据，其命名方式不够友好。比如用 rdseed 解压 SEED 数据得到的 SAC 数据，文件名的格式如下：

```
yyyy.ddd.hh.mm.ss.ffff.NN.SSSSS.LL.CCC.Q.SAC
```

其中，

- yyy.ddd.hh.mm.ss.fff 是 SAC 文件中第一个数据点对应的时刻；
- NN 为台网名，用两个字符表示；
- SSSSS 为台站名，一般为 3-4 字符，最多 5 字符；
- LL 为 location id，一般为空或两字符，常见的是 00、01、10，用于表征一个台站处的不同仪器；

- CCC 为通道名，3 字符，如 BHE、BHN、BHZ；
- Q 为质量控制标识，可以取 D、M、R、Q。

示例如下：

```
2012.055.12.34.56.7777.YW.MAIO.01.BHE.Q.SAC
2012.055.12.34.50.6666.YW.MAIO.01.BHN.Q.SAC
2012.055.12.34.54.5555.YW.MAIO.01.BHZ.Q.SAC
```

三个文件代表了 YW 台网 MAIO 台站的宽频地震仪记录的三分量波形数据。这样的长文件名在数据处理时显得很麻烦，一般都会根据实际需求进行适当的简化。

在某些情况下，我们会将同一事件在所有台站的波形数据放在同一个文件夹下，并将文件名以事件的发生日期/时间来命名。那么，SAC 文件名中的时间信息就可以被简化：

```
YW.MAIO.BHE
YW.MAIO.BHN
YW.MAIO.BHZ
```

有时候，我们会将不同事件在同一个台站的波形数据放在同一个文件夹下，并将文件名以台站名来命名，此时数据文件名中可能需要保留事件的日期信息：

```
MAIO.20120224.BHE
MAIO.20120224.BHN
MAIO.20120224.BHZ
```

鉴于 SAC 命令的语法，在数据命名时最好将分量名放在最后，而将台站名放在最前面。这样，在使用 SAC 的通配符读取特定事件的所有台站的垂直分量波形数据时，可以：

```
SAC> r *.20120224.BHZ
```

或者读取所有事件在同一台站的波形记录：

```
SAC> r MAIO.*.BHZ
```

4.3 合并数据

相关命令：`merge`

很多时候，从 SEED 数据中解压出来的同一个台站的 SAC 波形数据会被切割成多个等长或不等长的数据段。可能是因为仪器在某些时刻存在问题导致连续数据出现间断，也可能是出于其它考虑将数据进行切割。用户需要将这些数据段合并成单个包含连续波形数据的文件。

假定有三段连续/非连续的数据需要合并在一起，需要首先读入第一段数据，再 merge 其它段数据，最后使用 w over 将合并后的数据写入到第一个文件中，此时其余段数据 segment2.SAC 和 segment3.SAC 就可以删除了。

```
SAC> r segment1.SAC
SAC> merge segment2.SAC
SAC> merge segment3.SAC
SAC> w over
```

需要注意的是，在数据合并的过程中，可能会出现很多意外情况。

如果第一段数据的结束时刻早于第二段数据的开始时刻，即两段数据在时间上存在间断（gap），merge 命令提供了选项可以控制数据合并时如何对数据间断进行操作，是直接补零（zero）还是做线性插值（interp）。

如果第一段数据的结束时间晚于第二段数据的开始时间，则两段数据在时间上存在重叠（overlap）。对于这种情况，可以比较（compare）两段数据在重叠的时间段内的波形是否相同，若相同则正常合并，否则直接退出；也可对重叠的时间段内的波形直接做平均（average）。

4.4　事件信息

相关头段：evla, evlo, evdp, mag, o, nzyear, nzjday, nzhour, nzmin, nzsec, nzmsec

一般来说，从 SEED 连续波形中解压得到的 SAC 数据中是没有事件信息的。这需要用户搜索地震目录，获取事件的发震时刻、经度、纬度、深度和震级信息，并将这些信息写入到 SAC 文件的头段中。

可以使用 `chnhdr` 修改头段变量，再用 `writehdr` 将修改后的头段变量写回 SAC 文件中。比如想要修改事件的位置和深度，可以操作如下：

```
SAC> r cdv.?
SAC> ch evla 37.52 evlo -121.68 evdp 5.95   // 修改三个头段变量
SAC> ch mag 5.0                             // 修改一个头段变量
SAC> wh                                     // 将修改后的头段写入文件
```

这里需要注意 “`wh`” 与 “`w over`” 的区别。二者都是将内存中的数据写回到原文件中。前者只覆盖文件的头段区，后者则覆盖文件的头段区和数据区。若 SAC 数据从磁盘读入内存之后，未对波形数据做任何修改，仅修改了头段区，此时应使用 “`wh`” 命令将修改后的头段区写回文件，相对于 “`w over`” 来说速度要快很多。

向 SAC 头段中加入发震时刻的信息，要稍稍麻烦一点。前面已经说过，一般将 SAC 文件的参考时刻设置为发震时刻，这样的处理可以带来一堆好处。ch 提供了专门的选项用于处理发震信息。

假设发震时刻为 1987 年 06 月 22 日 11 时 10 分 10.363 秒：

```
SAC> r ./cdv.?
SAC> ch o gmt 1987 173 11 10 10 363
SAC> lh kzdate kztime o

  FILE: ./cdv.e - 1
 -------------

     kzdate = JUN 22 (173), 1987
     kztime = 11:09:56.363
          o = 1.400000e+01

SAC> ch allt -14
SAC> ch iztype IO
SAC> lh kzdate kztime o
```

```
 FILE: ./cdv.e - 1
------------

    kzdate = JUN 22 (173), 1987
    kztime = 11:10:10.363
         o = 0.000000e+00
SAC> wh
```

在上面的例子中，首先从地震目录中获取了地震的发震时刻，然后计算发震日期对应一年中的第几天，本例中为第 173 天，再利用“`ch o gmt yyyy ddd hh mm sss xxx`”的语法将发震时刻赋值给头段变量 o，SAC 会自动将发震时刻转换为相对于参考时刻的相对时间。由于此时 SAC 文件的参考时刻为“1987-06-22T11:09:56.363”，而 o 值对应的时刻为发震时刻“1987-06-22T11:10:10.363”，所以头段变量 o 的值为发震时刻相对于参考时刻的时间差，即 14s。

将发震时刻写入头段之后，还需要将参考时刻修改为发震时刻，与此同时还要修改所有的相对时间。“`ch allt xx.xx`”的功能是将所有已定义的相对时间加上 xx.xx 秒，同时从参考时刻中减去 xx.xx 秒，此时参考时刻即为发震时刻，而 o 值为 0。

在需要批量处理数据的时候，不可能通过“lh o”的方式查看 o 的值，然后使用到 allt 中。在 SAC 编程 一章中提供了适于批量处理的方式：

```
SAC> ch o gmt 1987 173 11 10 10 363
SAC> ch allt (0 - &1,o&) iztype IO
```

Tips

上例中的“`(0 - &1,o&)`”必须原样输入！

由于 101.6 版本重写了 SAC 宏的语法分析器，所以很多用法都发生了改变。你可能会发现，在 101.6 及其以后的版本中，`(0-&1,o&)`、`(-&1,o&)`、`(-&1,o)`都可以正常使用，但是上例中的版本是唯一一个在所有 SAC 版本中都正确的，为了命令的通用性，只要记住这个就可以了。

4.5 台站和分量信息

相关头段：stla、stlo、cmpaz、cmpinc

一般来说，从 SEED 数据中解压的 SAC 数据中都包含了准确的台站信息和分量信息。若数据中没有包含，则需要从其它途径获取这些信息，并写入到 SAC 数据头段中。

这很简单，只要使用前面提到的 `chnhdr` 命令即可。

4.6 震相理论到时

相关命令：`traveltime`

相关头段：tn（n=0-9）

traveltime 命令，可以计算 iasp91 或者 ak135 模型下的震相理论走时，并将其保存到 SAC 头段变量中。

```
SAC> dg sub teleseis nykl.z
SAC> traveltime model iasp91 picks 3 phase P S
traveltime: depth: 0.000000 km
SAC> lh t3 kt3 t4 kt4

  FILE: /opt/sac/aux/datagen/teleseis/nykl.z - 1
   ----------------------------------------

        t3 = 4.430530e+02
       kt3 = P
        t4 = 7.999642e+02
       kt4 = S
```

该命令会将震相 P、S 的理论到时依次写入头段变量 t3、t4 中，并写入相应的震相标识信息。

需要注意的是，该命令的正确运行要求内存中的所有波形数据的头段区中，必须包含地震位置、台站位置、发震时刻信息。

4.7　数据重采样

相关命令：decimate、interpolate

不同仪器的采样周期可能不同，在数据处理之前一般需要将所有的数据重采样到相同的采样周期；又或者数据的采样周期过小，导致数据量过大，此时也需要对数据进行重采样以减少数据量。

SAC 中有两个命令可以对数据进行重采样：一个是用于减采样的 decimate，另一个是用于插值的 interpolate。

```
SAC> fg seis
SAC> lh delta npts

  FILE: SEISMOGR - 1
 --------------

     delta = 1.000000e-02
      npts = 1000
SAC> decimate 5; decimate 2     // 减采样10倍
SAC> lh delta npts

  FILE: SEISMOGR - 1
 --------------

     delta = 9.999999e-02       // delta=0.001, 忽略此处的数值误差
      npts = 100
SAC> interpolate delta 0.015    // 重新插值到delta=0.015
SAC> lh delta npts

  FILE: SEISMOGR - 1
 --------------
```

```
    delta = 1.500000e-02
     npts = 666
```

需要注意的是，interpolate 会对数据进行插值，可能会引起插值误差；decimate 对数据进行减采样，为了避免混叠而加入了 FIR 滤波器。

4.8　去毛刺

相关命令：`rglitches`

地震仪器偶尔会出现问题，导致连续地震数据流中出现尖锋或者数据丢失。这些所谓的毛刺，肉眼很容易识别，但是在进行数据分析中却很容易被误认为是地震信号，因而需要在数据分析之前将毛刺去除。数据的毛刺在模拟地震记录中很常见，现在的数字地震记录中则很少见到。rglitches 命令可以在某种程序上去除地震信号中的毛刺。

rglitches 的效果可以从图 4.1 中直观地看到。

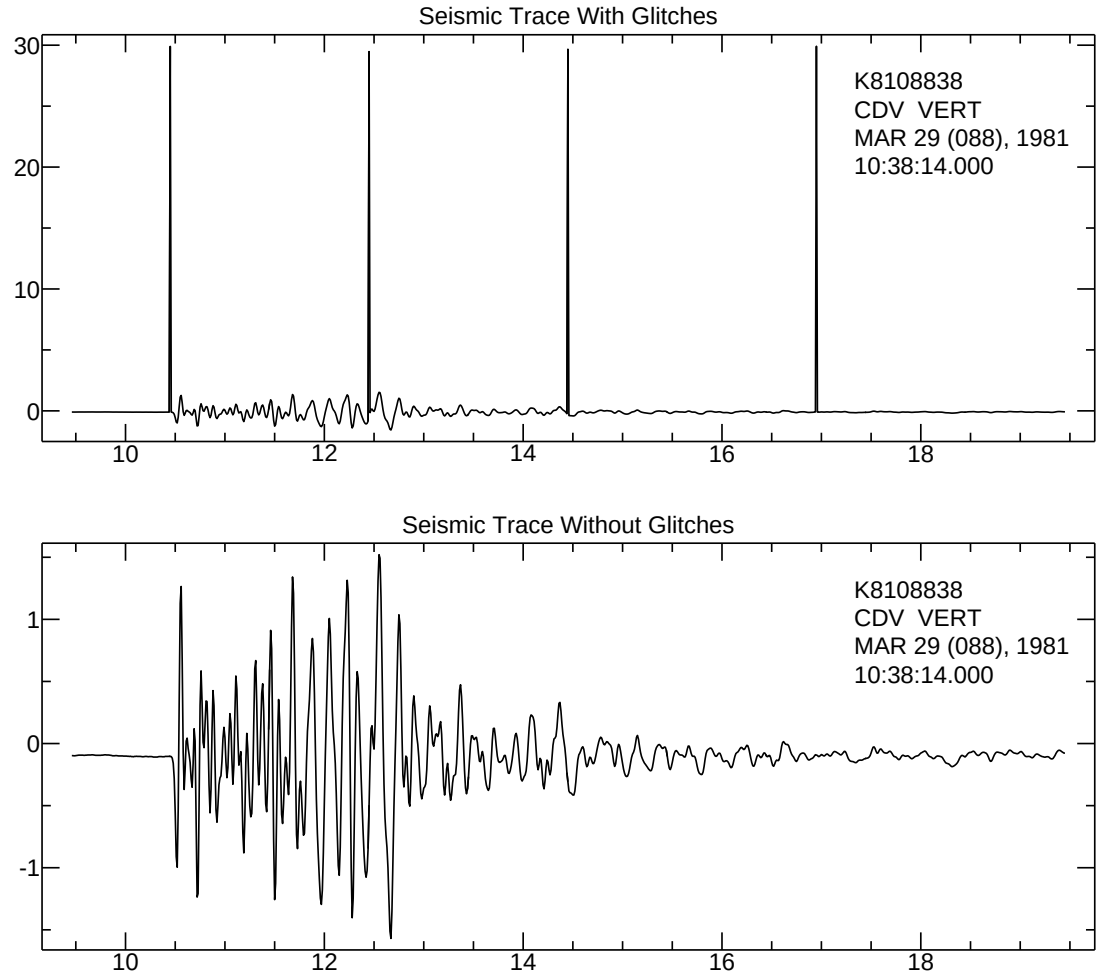

图 4.1: 地震波形去毛刺。上图为包含 glitches 的地震信号，下图为去除 rglitches 后的地震信号。

4.9　分量旋转

相关命令：`rotate`

相关头段：cmpinc、cmpaz

三个正交的地震传感器即可完全记录地面运动矢量。因此可以将三个正交的分量任意旋转到其它三个正交的方向上。

出于仪器安装的考虑，地震仪三分量一般都是 N、E、U 向的。而地震学里，由于 SH 波与 P-SV 波的解偶，更常处理的是 R、T、Z 向的三分量数据。因而地震信号的旋转是有必要的。

SAC 提供了 rotate 命令，用于旋转任意两个分量。在旋转之前，SAC 会检查两个分量的 cmpinc 和 cmpaz，确定二者是正交的。

```
SAC> dg sub teleseis ntkl.[enz]
/opt/sac/aux/datagen/teleseis/ntkl.e ...ntkl.n ...ntkl.z
SAC> w ntkl.e ntkl.n ntkl.z
SAC> r ./ntkl.n ./ntkl.e
SAC> lh cmpinc cmpaz

  FILE: ./ntkl.n - 1
 --------------

     cmpinc = 9.000000e+01
      cmpaz = 0.000000e+00

  FILE: ./ntkl.e - 2
 --------------

     cmpinc = 9.000000e+01
      cmpaz = 9.000000e+01
SAC> rotate to gcp              // 旋转到大圆路径
SAC> lh cmpinc cmpaz

  FILE: ./ntkl.n - 1
 --------------

     cmpinc = 9.000000e+01
      cmpaz = 2.440466e+01

  FILE: ./ntkl.e - 2
 --------------

     cmpinc = 9.000000e+01
      cmpaz = 1.144047e+02
SAC> w ntkl.r ntkl.t            // 保存为R分量和T分量
```

图 4.2 中，左图中从上至下为 N、E、Z 分量，右图中从上至下为 R、T、Z 分量。旋转到 R、T 分量后，可以很容易地识别出 Rayleigh 和 Love 波。

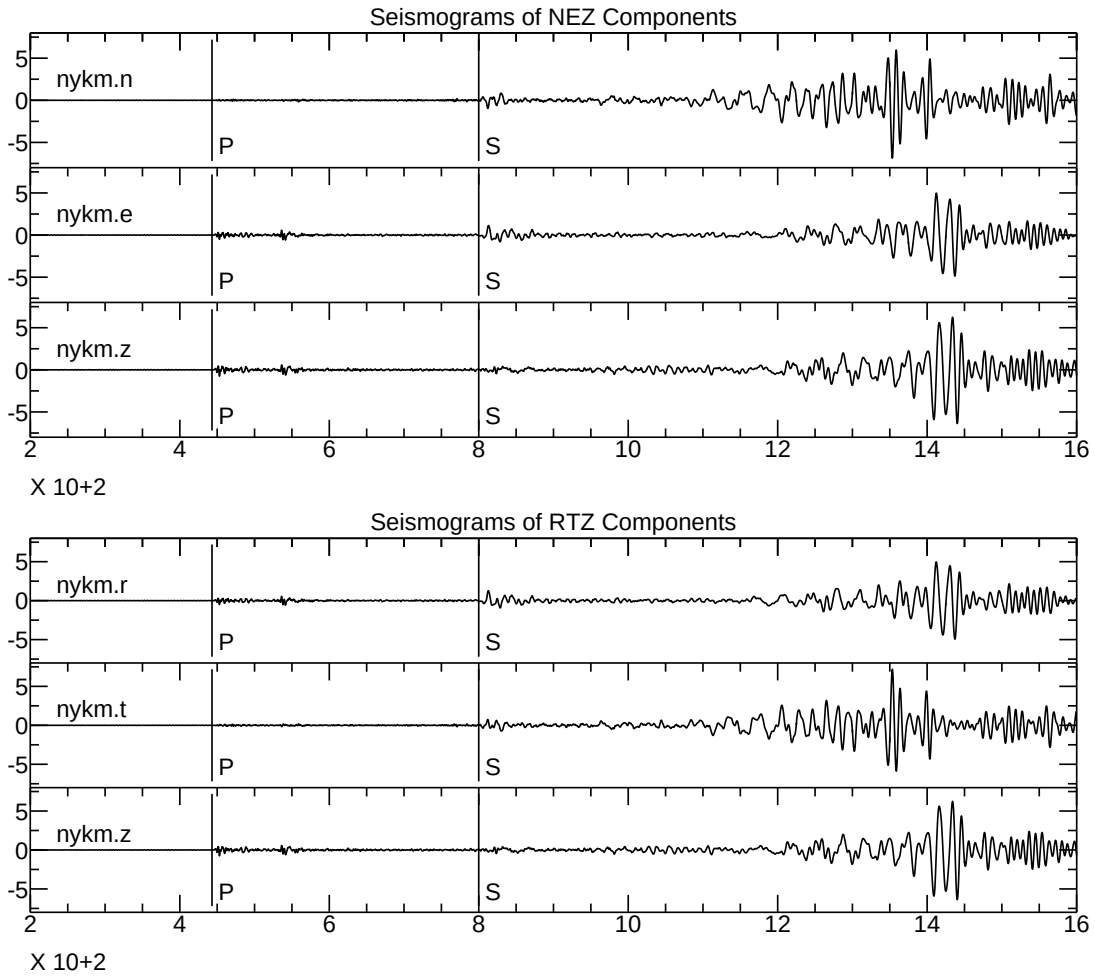

图 4.2: 将 N、E 分量旋转到 R、T 分量。

4.10 去均值、去线性趋势和波形尖灭

相关命令：`rmean`、`rtrend`、`taper`

通常，波形数据总会存在一个非零的均值或者存在一个长周期的线性趋势，这会影响到数据的分析，必须在数据分析前去除。另一方面，在对数据进行谱域操作（如 FFT、滤波等）时，若数据的两端不为零，则会出现谱域假象，因而实际数据经常需要做尖灭处理，使得数据两端在短时间窗内逐渐变成零值。

```
SAC> fg seis
SAC> rmean; rtr; taper
```

图 4.3 中，波形从上到下依次为原始波形、去均值、去线性趋势、和尖灭之后的波形。

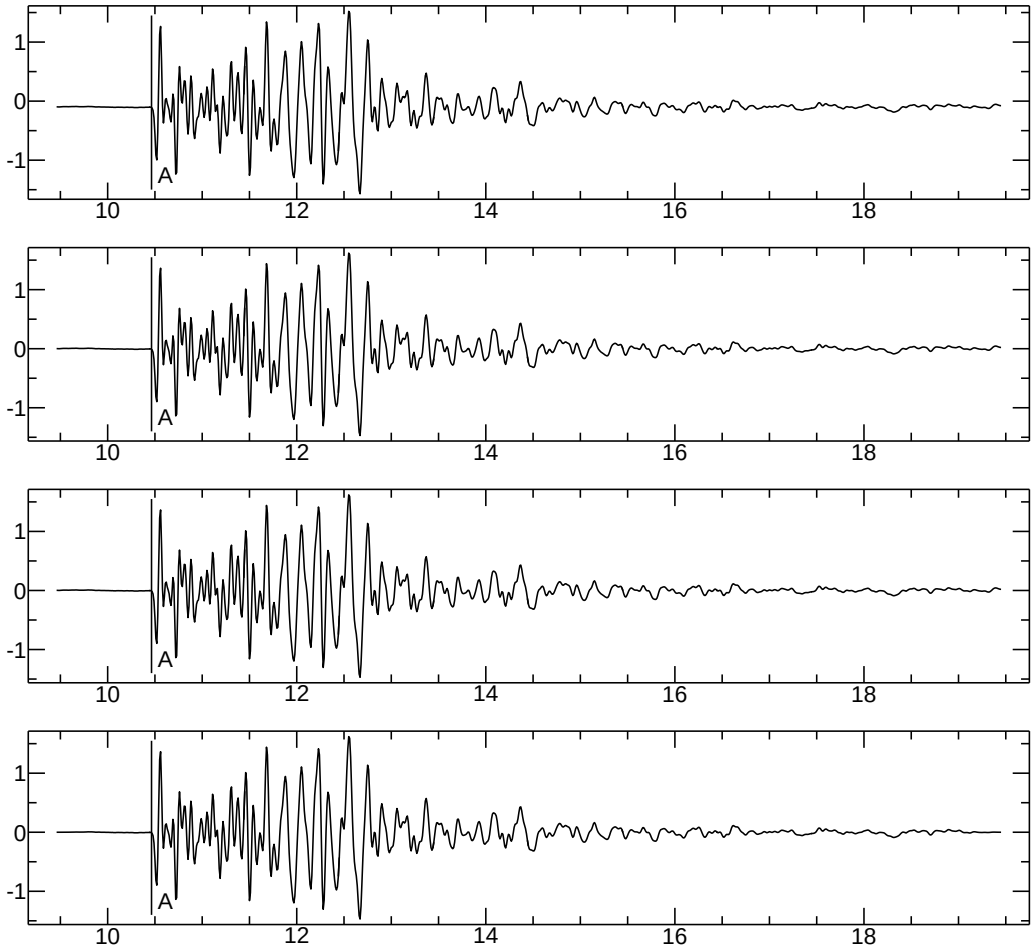

图 4.3: 去均值、去线性趋势和波形尖灭

4.11　仪器响应

相关命令：`transfer`

地震仪器观测到的地面运动记录可以表示为

$$u(t) = s(t) * g(t) * i(t)$$

其中 $s(t)$ 代表震源项，$g(t)$ 代表路径效应，$i(t)$ 代表仪器响应，星号代表卷积。

四个量中，$u(t)$ 是仪器记录的结果，已知；$i(t)$ 在仪器设计的时候会给出各种参数，已知；$s(t)$ 和 $g(t)$ 是未知的，也是地震学研究的两个主要内容：震源和结构。因而理解仪器响应 $i(t)$ 的基本原理并准确地去除仪器响应是研究的关键一步。

地震仪器记录的地面运动物理量有很多，常见的有速度和加速度，不太常见的还有位移、应变、旋转。这些物理量在被地震仪接收到之后，首先要将其转换为电信号，然后对电信号振幅进行放大以及滤波，再将连续时间序列离散化，最终以我们常见的波形的形式表现出来，所以我们最初见到的波形表示的是电信号的强弱，单位是 counts。

数据分析经常会遇到这样的情况，需要将电信号转换成真正的地面运动物理量，此时需要去除仪器响应；有时不同数据使用了不同类型的仪器，为了使得数据之间可以相互比较，则需要从原始数据中去除仪器响应，并加上同一类型仪器的仪器响应。

4.11.1 transfer

transfer 命令的基本语法是

```
TRANSFER [FROM type [SUBTYPE subtype]] [TO type [SUBTYPE subtype]]
[FREQLIMITS f1 f2 f3 f4] [PREWHITENING ON|OFF|n]
```

先不管 FREQ 和 PREW 这两个选项，该命令就是将波形数据从（from）一种仪器类型转换到（to）另一种仪器类型，即反卷积 `from` 选项给出的仪器响应，并卷积上 `to` 选项给出的仪器响应。

SAC 内置了很多标准地震仪器的仪器响应，即 type 的取值如表 4.1 所示。部分仪器类型还拥有子类型，如表 4.2 所示。

例如，数据 ABC.Z 的仪器类型是 LLL，通过如下命令将波形中的 LLL 仪器响应去除，并卷积上 SRO 仪器响应：

```
SAC> r ABC.z
SAC> rmean; rtr; taper
SAC> trans from LLL to SRO      // 未使用freq和prew
```

波形数据 XXX.Z 的仪器类型是 RSTN，子类型为 NYKM.Z，通过下面的命令将波形中的仪器响应去除，并卷积上 WWSP 仪器响应：

```
SAC> r XXX.z
SAC> trans from RSTN subtype NYKM.Z to DSS  // 未使用freq和prew
```

除了表 4.1 中列出的众多仪器类型之外，还有几个特别的仪器类型：

- none：即位移，也是 SAC 的默认值
- vel：速度
- acc：加速度

可以去除波形数据中的仪器响应，得到真实的位移场：

```
SAC> r XXX.z
SAC> trans from WWSP to NONE            // 未使用freq和prew
```

当然，也可以去除仪器响应，得到真实的速度场：

```
SAC> r XXX.z
SAC> trans from WWSP to VEL
```

随着地震仪器的不断发展，地震仪器的类型也越来越多，SAC 不可能包含已知的全部仪器类型，因而需要更灵活的方式来指定仪器响应。SAC 提供了三种方式：EVALRESP、POLEZERO 和 FAP。

4.11.2 EVALRESP 选项

RESP 文件

该选项需要 RESP 文件。RESP 文件包含了仪器的完整信息，其可以由 rdseed 程序直接从 SEED 数据中解压得到。想要了解 RESP 文件的具体细节，可以参考 SEED 格式的说明文档：http://www.fdsn.org/seed_manual/SEEDManual_V2.4.pdf。

表 4.1: SAC 内置仪器类型列表

type	说明
BBDISP	Blacknest specification of Broadband Displacement
BBVEL	Blacknest specification of Broadband Velocity
BENBOG	Blacknest specification of Benioff by Bogert
DSS	LLNL Digital Seismic System
DWWSSN	Digital World Wide Standard Seismograph Station
EKALP6	Blacknest specification of EKA LP6
EKASP2	Blacknest specification of EKA SP2
ELMAG	Electromagnetic
GBALP	Blacknest specification of GBA LP
GBASP	Blacknest specification of GBA SP
GENERAL	General seismometer
GSREF	USGS Refraction
HFSLPWB	Blacknest specification of HFS LPWB
IW	EYEOMG-spectral differentiation
LLL	LLL broadband analog seismometer
LLSN	LLSN L-4 seismometer
LNN	Livermore NTS Network instrument
LRSMLP	Blacknest specification of LRSM LP
LRSMSP	Blacknest specification of LRSM SP
NORESS	NORESS (NRSA)
NORESSHF	NORESS high frequency element
OLDBB	Old Blacknest specification of BB
OLDKIR	Old Blacknest specification of Kirnos
PORTABLE	Portable seismometer with PDR2
PTBLLP	Blacknest specification of PTBL LP
REDKIR	Blacknest specification of RED Kirnos
REFTEK	Reftek 97-01 portable instrument
RSTN	Regional Seismic Test Network
S750	S750 Seismometer
SANDIA	Sandia system 23 instrument
SANDIA3	Sandia new system with SL-210
SRO	Seismic Research Observatory
WA	Wood-Anderson
WABN	Blacknest specification of Wood-Anderson
WIECH	Wiechert seismometer
WWLPBN	Blacknest specification of WWSSN long period
WWSP	WWSSN short period
WWSPBN	Blacknest specification of WWSSN short period
YKALP	Blacknest specification of YKA long period
YKASP	Blacknest specification of YKA short period

表 4.2: 部分仪器子类型

主类型	子类型
LLL	LV, LR, LT, MV, MR, MT, EV, ER, ET, KV, KR, KT
LNN	BB|HF
NORESS	LP|IP|SP
RSTN	[CP|ON|NTR|NY|SD][KL|KM|KS|7S][Z|N|E]
SANDIA	[N|O][T|L|B|D|N|E][V|R|T]
SRO	BB|SP|LPDE
FREEPERIOD v	ELMAG, GENERAL, IW, LLL SUBTYPE BB, REFTEK
MAGNIFICATION n	ELMAG, GENERAL
NZEROS n	GENERAL, IW
DAMPING v	GENERAL, LLL SUBTYPE BB, REFTEK
CORNER v	LLL SUBTYPE BB, REFTEK
GAIN v	
HIGHPASS v	REFTEK

RESP 文件中包含了仪器响应的零极点信息及其所对应的台网名、台站名、通道名、日期和时间等信息。transfer 会从 SAC 头段区中提取相关信息，并与 RESP 文件中的信息进行匹配，若二者完全匹配方可继续执行 transfer 命令。通过给 evalresp 选项指定额外的选项和参数可以覆盖 SAC 文件中对应的头段变量，这些可能的选项包括：STATION、CHANNEL、NETWORK、DATE、TIME、LOCID、FNAME。

每个选项都必须有一个合适的值，如果 DATE 在 SAC 头段中为设定且在选项中未指定，则使用当前系统日期，TIME 同理。若 NETWORK 未指定，则默认使用任意台站名；若 LOCID 未指定，则默认使用任意 LOCID。也可以使用 FNAME 选项并加上 RESP 文件名，此时会不再检测其它信息是否匹配，强制 transfer 使用指定的仪器响应文件。

如果 FNAME 选项未指定，EVALRESP 将尝试按照一般格式在当前工作目录下寻找合适的 RESP 文件，其一般格式为 “RESP.<NET>.<STA>.<LOCID>.<CHAN>”，比如：“RESP.IU.ANMO..BHZ” 。

evalresp 本质上是独立的程序，SAC 将其内嵌到 SAC 的代码中。由于 evalresp 默认使用 m 作为基本单位，而 SAC 以 nm 作为基本单位，在使用 evalresp 选项对波形数据去仪器响应得到位移场、速度场或加速度场时，SAC 会自动完成 m 到 nm 的转换，对数据乘以 10^9 的因子以符合 SAC 的标准。

EVALRESP 例子

若仪器响应文件位于当前目录，且其具有标准的文件名，可以用如下命令去除仪器响应，transfer 命令会根据 SAC 文件的头段区自动寻找合适的 RESP 文件：

```
SAC> r 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.SAC
SAC> rtr
SAC> taper
SAC> trans from evalresp to none freq 0.004 0.007 0.2 0.4
```

若 RESP 文件位于 “/tmp/Responses/RESP.TA.N11A..LHZ” ：

```
SAC> setbb resp /tmp/Responses/RESP.TA.N11A..LHZ
SAC> r 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.SAC
SAC> rtr
SAC> taper
SAC> trans from evalresp fname %resp% to none freq 0.004 0.007 0.2 0.4
```

为了从文件 `16.42.05.5120.TS.PAS.BHZ.SAC` 中去除仪器响应，并添加台站 COL 相同周期的响应：

```
SAC> r 16.42.05.5120.TS.PAS.BHZ.SAC
SAC> rtr
SAC> taper
SAC> trans from evalresp to evalresp station COL
```

为了显示 IU 台网 COL 台站 BHZ 通道，1992 年 01 月 02 日 16:42:05 的仪器响应：

```
SAC> fg impulse npts 16384 delta .05 begin 0.
SAC> trans to evalresp sta COL cha BHZ net IU date 1992/2 time 16:42:05
SAC> fft
SAC> psp am
```

4.11.3　POLEZERO 选项

SAC PZ 文件

POLEZERO 选项需要使用另一种仪器响应文件，称之为 SAC PZ 文件。

RESP 文件中包含了仪器响应的所有细节，因而相对来说较为繁琐。SAC 从 RESP 文件中提取出仪器响应中的相关信息，重新定义了新的 PZ 响应文件。相对于 RESP 文件来说，PZ 文件中仅包含仪器响应的零极点和增益信息，在去仪器响应时速度更快。

地震仪器的响应函数（即线性系统脉冲响应）一般用 Laplace 域的零极点来表示。响应函数 $H(s)$ 由 nz 个零点和 np 个极点构成，如下式所示：

$$H(s) = \frac{(s-z_1)(s-z_2)\dots(s-z_{nz})}{(s-p_1)(s-p_2)\dots(p-z_{np})}$$

其中，Laplace 变量 $s = 2\pi fi$，f 为频率，单位 Hz。

PZ 文件中宏包含零点、极点、常数值以及注释行，其中常数是比例因子。常数行的缺省值为 1.0。

文件中包含关键字 “POLES” 的行给出了极点个数（np），接下来的 np 行列出仪器响应的极点，每行两个浮点数，分别代表实部和虚部。

以关键字 “ZEROS” 起始的行给出零点个数（nz）。一般地，零极点文件中可能包含一个或多个 `(0.0,0.0)` 这样的零点，在 SAC 中这样的零点不出现在 PZ 文件中，所以接下来列出的零点数可能会少于 nz 个。

PZ 文件中还包含格式化的注释行信息，以帮助用户组织和理解 PZ 文件中的内容。目前 PZ 零极点文件一般用 rdseed 从 SEED 中解压得到或者从 SAC PZ 网页服务[1] 中获取。若内存中的波形数据头段区 KSTNM、KCMNM、KZDATE、KZTIME 与零极点文件中的注

[1] http://www.iris.edu/ws/sacpz

释信息相匹配，则 transfer 命令会正常运行。一个 SAC PZ 文件中，可能包含多个仪器的零极点文件的信息，其跨越了多个时间段或包含多个台站多个通道，transfer 命令会对内存中与之匹配的波形数据进行处理。

下面给出 rdseed v5.2 产生的一个零极点文件的例子，早期版本的 rdseed 的输出可能与此略微不同：

```
* **********************************
* NETWORK   (KNETWK): II
* STATION    (KSTNM): PFO
* LOCATION   (KHOLE): 00
* CHANNEL   (KCMPNM): BHZ
* CREATED           : 2011-08-11T00:24:07
* START             : 2010-07-30T18:50:00
* END               : 2599-12-31T23:59:59
* DESCRIPTION       : Pinon Flat, California, USA
* LATITUDE          : 33.610700
* LONGITUDE         : -116.455500
* ELEVATION         : 1280.0
* DEPTH             : 5.3
* DIP               : 0.0
* AZIMUTH           : 0.0
* SAMPLE RATE       : 20.0
* INPUT UNIT        : M
* OUTPUT UNIT       : COUNTS
* INSTTYPE          : Streckeisen STS-1 Seismometer with Metrozet E300
* INSTGAIN          : 3.314400e+03 (M/S)
* COMMENT           : S/N #119005
* SENSITIVITY       : 5.247780e+09 (M/S)
* A0                : 7.273290e+01
* **********************************
ZEROS       6
-7.853982e+01       +0.000000e+00
-1.525042e-01       +0.000000e+00
-1.525042e-01       +0.000000e+00
POLES       6
-1.207063e-02       +1.224561e-02
-1.207063e-02       -1.224561e-02
-1.522510e-01       +9.643684e-03
-1.522510e-01       -9.643684e-03
-4.832398e+01       +5.817080e+01
-4.832398e+01       -5.817080e+01
CONSTANT    3.816863e+11
```

该仪器响应表明，响应函数含六个极点和六个零点（只列出三个非 0 零点）。

在某些平台上，rdseed v5.1 会产生错误的 end 时间，通常比 start 时间早一点，而导致 transfer 无法正常工作，手动将 end 时间修改为迟于 created 时间可以解决这个问题。SAC 为了解决这个问题，可以指定仪器类型为 POLEZERO，子类型为 PZ 文件名。此时 SAC 会强制使用该 PZ 文件作为响应文件，而不去检测头段是否匹配。在指定子类型为 PZ 文件名的前提下，PZ 文件中的注释行便不再被 transfer 命令所需要，因而有或者没有对于去仪器响应而言没有影响。

在处理大量数据时做了一下测试，一种情况是直接使用 rdseed 解压出的包含注释的

PZ 文件去仪器响应，另一种是用脚本重做一个无注释的 PZ 文件，再用无注释的 PZ 文件去仪器响应。测试结果表明，后者在速度上相对于前者快了非常多。

需要注意的是，如果零极点文件由 rdseed 从 SEED 文件中提取得到，则零极点文件的输入为位移，其单位与 SEED 格式相同，一般为 m。不像 evalresp，SAC 不知道零极点文件来自何处，也无法知道其输入单位是 m，因此在使用 PZ 文件去除仪器响应到位移量、速度量或加速度量时，应手动乘以比例因子 10^9 以保证数据的正确性。

POLEZEROS 例子

PZ 文件 `SAC_PZs_XC_OR075_LHZ` 是需要从波形 `OR075_LHZ.SAC` 中去除的仪器响应零极点文件：

```
SAC> setbb pzfile "SAC_PZs_XC_OR075_LHZ"
SAC> r OR075_LHZ.SAC
SAC> rtr
SAC> taper
SAC> trans from polezero subtype %pzfile% to none freq 0.008 0.016 0.2 0.4
SAC> mul 1.0e9          // 由于是to none, 这里必须乘以1.0e9
SAC> w OR075.z          // 此时数据为位移量, 单位为nm
```

加入在上面的例子中，没有使用 `SAC_PZs_XC_OR075_LHZ`，而是使用了错误的 PZ 文件 `SAC_PZs_wrong`，下面的过程给出如何调用 transfer 命令去除不正确的响应并加入正确的响应：

```
SAC> r OR075.z                  // 使用了错误的仪器响应文件
SAC> write OR075.zbad
SAC> setbb pzo "SAC_PZs_wrong"
SAC> setbb pzn "SAC_PZs_XC_OR075_LHZ"
SAC> trans from polezero s %pzn% to polezero s %pzo% freq 0.008 0.015 0.2 0.4
SAC> write OR075.z              // 纠正后的文件
```

最后一个例子，假设我们用 rdseed v5.2 在当前目录产生了同一个时间的多个台站的多个波形，并将所有的 SAC PZ 文件合并得到一个新的 PZ 文件 event.pz。下面的例子将读入全部 BH* 波形文件，经仪器响应校正并覆盖原文件：

```
SAC> r *BH*SAC          // 读入全部数据
SAC> rtr
SAC> taper
SAC> trans from polezero s event.pz freq 0.05 0.1 10.0 15.0
```

4.11.4　FAP 选项

FAP 文件

SAC v101.4 中重新引入了 FAPfile 选项，其需要 FAP 文件。

FAP 文件是响应函数的另一种表现形式，其包含了很多记录行，每行三个字段，分别是频率 (HZ)、振幅及相位。频率不需要等间隔分段。在执行 transfer 时，低于第一行频率的频段将使用第一行的振幅和相位；同理大于最后一行频率的频段将使用最后一行的振幅和相位。

EVALRESP v3.3.2 可以输出为 FAPfile，使用 EVALRESP 生成的 FAPFILE 而非 POLEZERO 的好处在于其可以包含更丰富的仪器响应，且可以显式控制需要校正的频率段。

FAPfile 例子

假设有 fapfile 文件 `fap.n11a.lhz_0.006-0.2`，其名字表示频率段位 0.006 Hz 到 0.2 Hz，要从波形 `2006.253.14.30.24.0000.TA.N11A..LHZ.Q.SAC`中移除该仪器响应：

```
SAC> r 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.SAC
SAC> rtr
SAC> taper
SAC> trans from fap s fap.n11a.lhz_0.006-0.2 freq 0.004 0.006 0.1 0.2
SAC> mul 1.0e9
```

4.11.5 其它

关于仪器响应的更多细节可以查看 SEED 格式手册以及参考如下几篇博文：

- 地震学的仪器响应
- 仪器响应的物理细节
- 仪器响应文件 RESP
- 仪器响应文件 SAC PZ
- 用 RESP 和 PZ 去除仪器响应的差别
- 仪器响应实例分析

4.12 滤波

相关命令：`bandpass`、`lowpass`、`highpass`、`bandrej`

几乎所有的数据分析都需要将数据限制在一定的频率范围内，对数据做低通、高通或带通滤波很有必要。

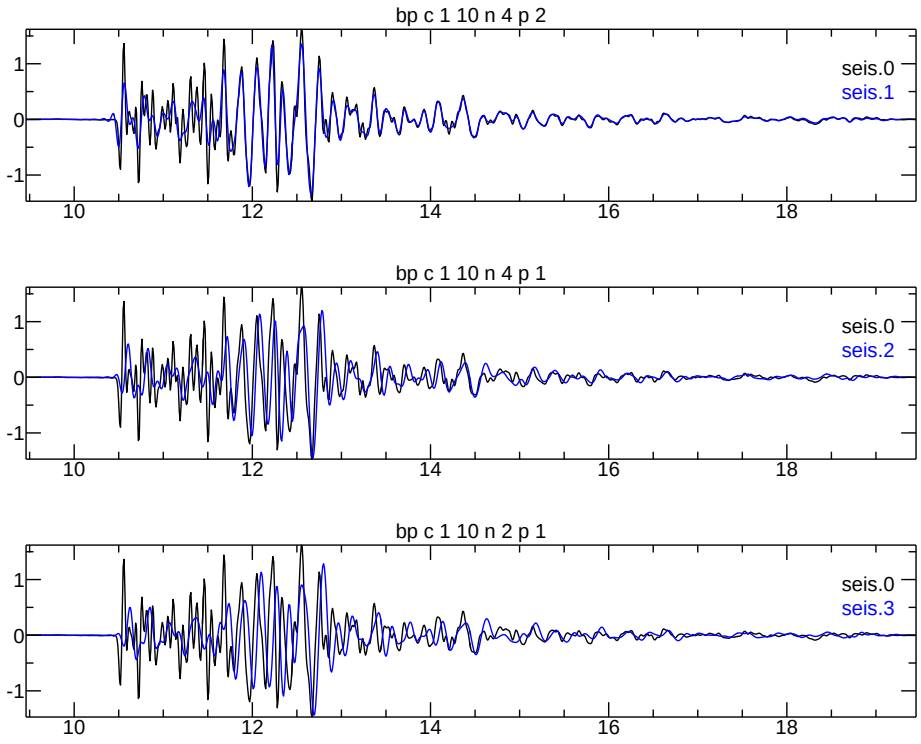

图 4.4: 带通滤波效果

```
SAC> fg seis
SAC> rmean; rtr; taper
SAC> w seis
SAC> bp c 1 10 n 4 p 2  // 4阶2通道butterworth带通滤波
SAC> r seis
SAC> bp c 1 10 n 4 p 1  // 4阶1通道butterworth带通滤波
SAC> r seis
SAC> bp c 1 10 n 2 p 1  // 2阶1通道butterworth带通滤波
```

图 4.4 中分别给出了使用三种滤波器之后的波形与原波形的对比图。

4.13　质量控制

质量控制就是标记/删除信噪比低或不合适的波形。

用户可以自己在程序中判断数据的好坏以进行质量控制，但这样做很困难，因为实际情况中，会遇到各种奇形怪状的 “坏” 波形，很难用统一的程序将这些 “坏” 波形挑选出来，所以更多时候需要人工的参与。

我个人的做法如下：

```
SAC> r *.SAC         // 读入全部的SAC数据
SAC> ppk p 5         // plotpk, 每次绘制5个波形
// 若波形质量很差, 则用t9标记
SAC> wh
SAC> q
```

解释一下以上做法，首先读入所有的 SAC 数据，然后利用 plotpk，每次绘制 n（n 取 5-10 比较合适）个波形。若波形质量很好，则不理会；若波形质量很差，则在波形的任意时刻标记 t9 的值（具体如何标记可以参考下一节的内容），然后使用 wh 将标记的 t9 保存到头段中，再退出 SAC。

完成上面的步骤之后，所有 “坏” 波形的 t9 都被标记，一般来说都是一个正值，而所有 “好” 波形的 t9 则都处于未定义状态，其值为 `t9=-12345.0`。

鉴于此，可以通过如下命令删除 “坏” 波形：

```
$ saclst t9 f seis | awk '$2>0{print "rm",$1}' | sh
```

注意：一定要在理解该命令的含义的前提下才可使用！

4.14　震相拾取

相关命令：`plotpk`

震相拾取是 SAC 中最常用的一个功能。plotpk 的用法很简单，但却是教程中最难解释的一部分。当在 SAC 中键入 plotpk（简写为 ppk）后，即进入 “ppk 模式”。

```
SAC> fg seis
SAC> ppk         // 进入ppk模式
// 将光标置于绘图窗口上, 键入"q"以退出ppk模式
SAC> wh
```

```
SAC> q
```

此时会出现一个绘图窗口，且焦点位于绘图窗口上，SAC 程序进入了 ppk 模式。

移动光标到绘图窗口内部，光标会以交叉瞄准线的形式出现。的形态出现。在 “安装 SAC” 一节中提到的 SAC 全局变量 SAC_PPK_USE_CROSSHAIRS 可以控制该光标的形态。当此全局变量为 0 时，光标形态为 “+”；当此全局变量为 1 时，会在 0 的基础上再加上水平和垂直线。个人推荐设置其值为 1。

此时所有的键入都会解释成 “ppk 模式” 下的命令，表 4.3 列出了 “ppk 模式” 支持的所有命令。

表 4.3: ppk 模式命令一览表

命令	含义	说明
a	定义事件初至 a	1,7
b	如果有，则显示上一张绘图	
c	计算事件的初至和结束	1,4,7
d	设置震相方向为 DOWN	
e	设置震相起始为 EMERGENT(急始)	
f	定义事件结束 f	1,2,3,7
g	以 HYPO 格式将拾取显示到终端	4
h	将拾取写成 HYPO 格式	3,4
i	设置震相起始为 IMPULSIVE	
j	设置噪声水平	2,6,8
l	显示光标当前位置	2,4
m	计算最大振幅波形	2,3,5
n	显示下一绘图	
o	显示前一个绘图窗，最多可以保存 5 个绘图窗	
p	定义 P 波到时	1,2,3,7
q	结束 ppk 模式	
s	定义 S 波到时	1,2,3,7
t	用户自定义到时 tn，输入 t 之后需要输入 0 到 9 中的任一数	1,2,7
u	设置震相方向为 UP	
v	定义一个 Wood-Anderson 波形	2,5
w	定义一个通用波形	2,5
x	使用一个新的 x 轴时间窗，简单说就是放大。	
z	设置参考水平	2,6,8
\	删除当前全部拾取的定义。当一个文件中包含多个事件时有用。	
+	设置震相方向为 PLUS	
-	设置震相方向为 MINUS	
␣	设置震相方向为 NEUTRAL	
n	设置震相质量为 n，n 取 0-4	

不同的命令效果可能不同，有些会在绘图窗口显示信息，有些会将信息写入头段变量，下面对表 4.3 中的说明进行一个说明：

1 会将信息写入头段变量
2 写入字符型震相拾取文件（若已打开）
3 写入 HYPO 格式震相拾取文件（若已打开）
4 在绘图窗口中显示信息
5 窗口显示包含波形的矩形
6 在指定的水平处放置水平光标
7 绘图窗口显示含有到时标识的垂直线
8 绘图窗口显示含有标识的水平线

“ppk 模式” 的命令几乎都是由单个字符组成的（唯一的例外是命令 “t”，由字符 “t” 和 0-9 的整数构成），每个命令都有相应的功能，同时也会产生一定的效果。经常使用的包括 “b”、“l”、“n”、“o”、“q”、“t” 和 “x”。

关于 “x”，首先需要将光标移动到窗口的左侧某位置，键入 “x”，再向右移动至某位置，再次键入 “x”，两次键入确定了一个时间窗，绘图窗口中将只显示该时间窗内的波形，即 X 轴的放大，常用于查看波形的细节。可以不断重复此步骤，进行多次放大。

在某些版本下，可以不使用 “x”，直接在左侧位置按下鼠标左键，拖动至右侧位置，再松开鼠标左键，则该时间窗内的波形会被放大。

放大之后的另一个问题就是缩小，这可以通过 “o” 来实现，“o” 最多可以回退 5 次绘图历史。

当只有一个波形时：

```
SAC> fg seis
SAC> ppk
// 键入"t"和"0"标记到时
SAC> wh          // 保存头段
```

当有多个波形时，一般每次只显示若干个：

```
SAC> dg sub teleseis ntkl.z nykl.z onkl.z sdkl.z
SAC> ppk p 1    // 每次显示1个波形
// 键入"n"显示下一个波形，键入"b"显示上一个波形
// 键入"t"和"0"标记到时
SAC> wh
SAC> q
```

当有多个波形时，每次显示三分量，且希望一次性标记三个分量的到时：

```
SAC> dg sub teleseis ntkl.[nez] nykl.[nez] onkl.[nez] sdkl.[nez]
SAC> ppk p 3 r m
// 键入"t0"标记到时，此时键入一次"t0"可以标记当前显示的全部波形
SAC> wh
SAC> q
```

4.15　数据截窗

相关命令：cut、cutim

数据申请时一般会选择尽可能长的时间窗，而实际进行数据处理和分析时可能只需要其中的一小段时间窗，这就需要对数据时间窗进行截取。

SAC 中有两个命令可以用于数据截窗，分别是 cut 和 cutim，二者的语法基本是相同的，需要注意的是 cut 是 “参数设定类” 命令，cutim 是 “操作执行类” 命令。

4.16 数据分析

数据分析是整个科学研究的核心。SAC 软件只是负责完成前期的预处理工作，真正核心的数据分析工作通常需要用户自己写程序完成。这就牵涉到如何在 C 程序中读写 SAC 格式的问题，这个问题放在 “使用 SAC 函数库” 和 “SAC I/O 自定义” 这两章中具体说明。

第 5 章　SAC 图像

5.1　图像输出设备

SAC 支持三类图像输出设备：

X Window System 也常称为 X11 或 X，是一种以位图方式显示的软件窗口系统。图 2.1 展示了 SAC 中的 X 窗口，这是 SAC 默认的绘图设备。

SGF 全称 SAC Graphic File，即 SAC 图像文件。SGF 文件中包含了绘制一个图像所包含的全部信息，因而可以通过转换程序将其转换到其它图像设备或图像格式。SAC 提供了三个 sgf 转换工具：sgftops、sgftoeps 和 sgftox。

PDF 和 PS 从 101.5 开始，SAC 加入了 `saveimg` 命令，可以将图像直接输出为 PDF 或 PS 格式 [1]。

SAC 提供了 `setdevice` 命令设置默认的图像设备；`begindevices` 和 `enddevices` 用于启动/关闭指定的图像设备。

对于 xwindows，SAC 提供了 10 个可以使用的窗口，编号为 1-10。`window` 命令可以设置每个绘图窗口的长宽比以及相对于屏幕的位置；`beginwindow` 命令可以启动某个编号的绘图窗口。

对于 SGF，每个图像都将保存到 sgf 文件中，文件名格式为 “fnnn.sgf”，其中 “nnn” 为图像编号，起始编号为 001。`sgf` 命令可以控制 SGF 图像设备的一些选项，比如 sgf 文件的前缀、起始编号、目录、文件尺寸等。

对于 PDF 和 PS，由于 saveimg 命令是在 101.5 之后才加入的，所以使用上与 xwindows 和 sgf 有很多不同之处。

这三者的区别会在 “图像设备与图像保存” 中详细分析。

5.2　绘图命令

SAC 提供了很多用于绘图的命令（操作执行类命令）以及控制图像外观的命令（参数控制类命令）。这一节列出 SAC 中的全部绘图命令。

前面已经介绍了 SAC 中常用的 plot、plot1、plot2 和 plotpk 命令的用法，其它的一些绘图命令的用法会在每个命令的语法说明中详细解释。

[1] 该命令也支持输出为 png 和 xpm 格式，但 png 和 xpm 为位图图像格式，精度不够，且依赖于其它函数库，因而不推荐使用。

表 5.1: 绘图命令一览

命令	说明
plot	一次绘制一个波形
plot1	一次绘制多个波形，共用 X 轴
plot2	一次绘制多个波形，共用 Y 轴
plotpk	用于交互式震相拾取
plotpm	根据一堆波形数据，绘制质点运动图
plotalpha	从磁盘读取列数据并直接绘图
plotc	利用光标在 SAC 绘图上做注释
plotdy	绘制带有误差棒的图
plotxy	以一个 SAC 文件为自变量、其它文件为因变量绘图
plotsp	绘制谱数据

5.3 图像外观

图 5.1 展示了一个标准的 SAC 图像所包含的元素。

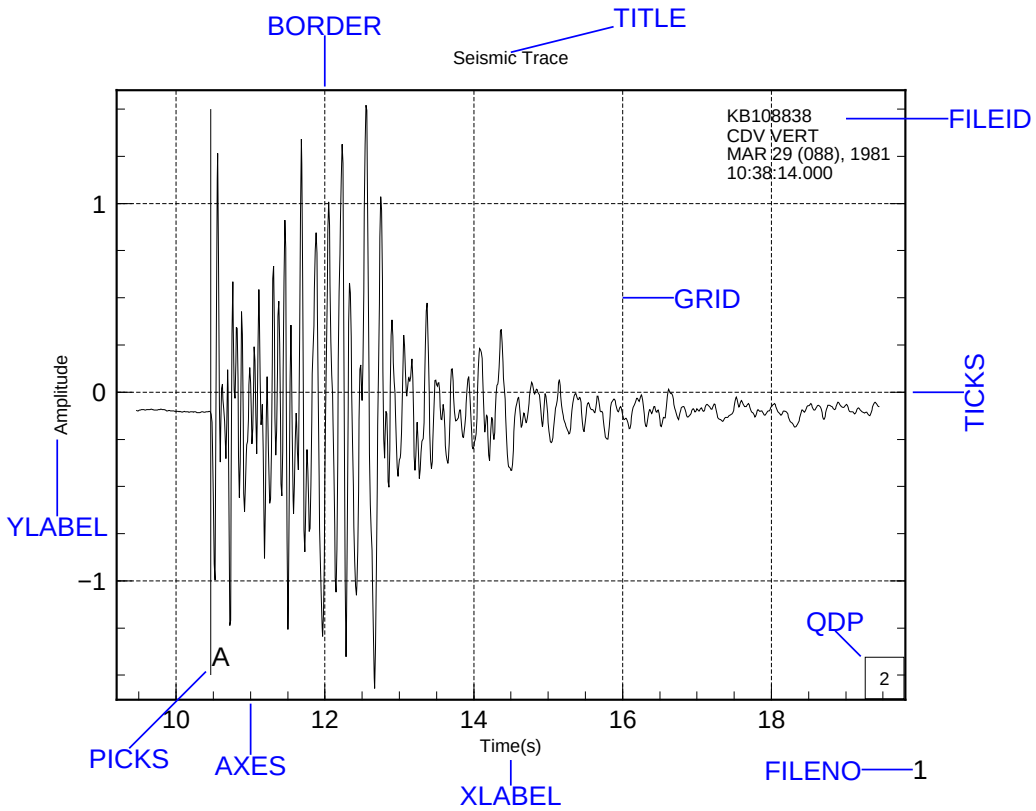

图 5.1: 绘图外观及其相关命令。图中蓝色部分为对绘图外观的说明。

图 5.1 可以用如下命令绘制得到：

```
SAC> fg seis                 // 生成数据
SAC> grid on                 // 显示网格
SAC> title 'Seismic Trace'   // 设置标题
SAC> xlabel "Time(s)"        // 设置x轴标签
```

```
SAC> ylabel "Amplitude"      // 设置y轴标签
SAC> filenumber on           // 显示文件号
SAC> axes only left bottom   // left和bottom显示axes
SAC> ticks only right        // right显示ticks
SAC> border on               // top显示border
SAC> p                       // 绘图
```

与之相关的命令列举如下：

fileid 控制文件 id 的显示效果（右上角 FILEID）
filenumber 控制文件号的显示（图右下角 FILENO）
picks 控制时间标记的显示效果（图左侧 PICKS）
title 指定标题及其属性（图上部 TITLE）
plabel 通用标签，相比 xlabel 和 ylabel 更加灵活
grid 控制网格的显示以及网格的线型
gtext 控制绘图文本的字体及字号
tsize SAC 自定义了四种文本大小，分别为 tiny、small、medium 和 large，此命令可以修改其代表的具体文本大小值

图中右下角的 “2” 称之为 “qdp 因子”，qdp 的全称是 “quick and dirty plot”。在 SAC 开发的早期，计算机的性能一般，若数据点数过多，绘图会耗费大量时间，因而 SAC 采用了 “qdp” 的方式，每隔若干个数据点绘制一个数据点。目前计算机的性能已经足以同时绘制所有数据点，因而一般关闭 qdp，可以使用 `qdp off` 命令。

每个绘图都有一个边框，每个边框有上下左右四个边。按照标准的说法，图5.1 中，上边为 “border”，右边为 “border+ticks”，左边和下边为 “border+ticks+annotations”。

SAC 对于边的定义稍有不同，其将左边和下边称之为 “axes”，右边称之为 “ticks”，上边称之为 “border”。用 axes 命令可以控制在哪些边使用 “axes”，只有不使用 “axes” 的边才可以用 ticks 命令控制是否使用 “ticks”，而只有不使用 “axes” 和 “ticks” 的边才可以使用 border 命令控制是否使用 “border”。

有一些命令可以分别针对 X 和 Y 轴进行设置，下面仅列出与 X 轴相关的命令：

xlabel 控制 X 轴标签
xlim 控制 X 轴范围
xdiv 控制 X 轴注释的间隔
xgrid 控制 X 轴网格的显示
xfudge 设置 fudge 因子以控制 X 轴范围

在绘制时间序列时一般都使用线性坐标，SAC 也提供了一堆命令以控制 X、Y 轴是线性坐标还是对数坐标。这些命令包括：`linlin`、`linlog`、`loglin`、`loglog`、`xlin`、`xlog`、`ylin`、`ylog`。

对于对数坐标，还有一些命令可以控制其外观，比如`xfull`、`loglab`、`floor`。

对于等值线图 (即 3D 数据) 而言，还有命令`zcolors`、`zlabels`、`zlevels`、`zlines`、`zticks`。

5.4 线条属性

线条的属性包括线型（line）、线宽（width）、颜色（color）和符号（symbol）。

```
SAC> fg seis
SAC> line 3          // 线型为3
SAC> width 2         // 线宽为2
SAC> color red       // 红色
SAC> p
```

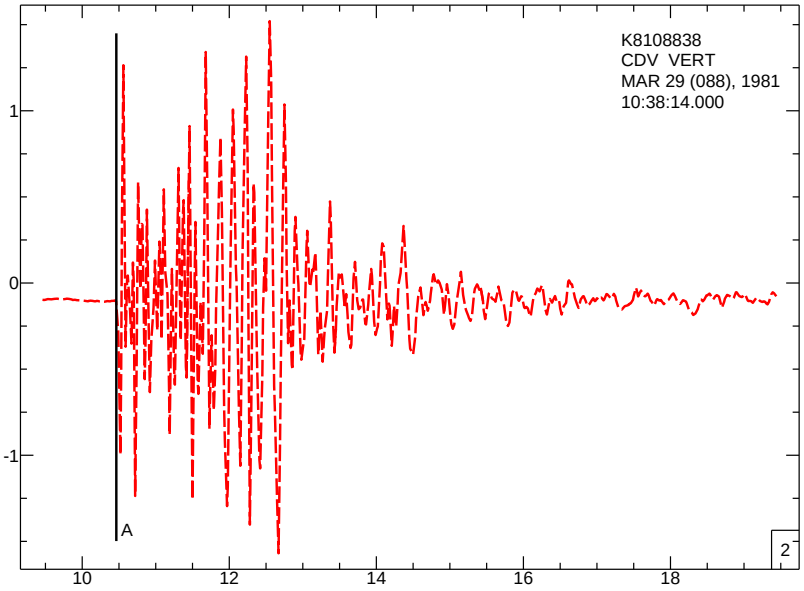

图 5.2: 线条属性示意图 1

```
SAC> dg sub teleseis ntkl.z nykl.z onkl.z sdkl.z
SAC> line incre
SAC> color black incre
SAC> p
```

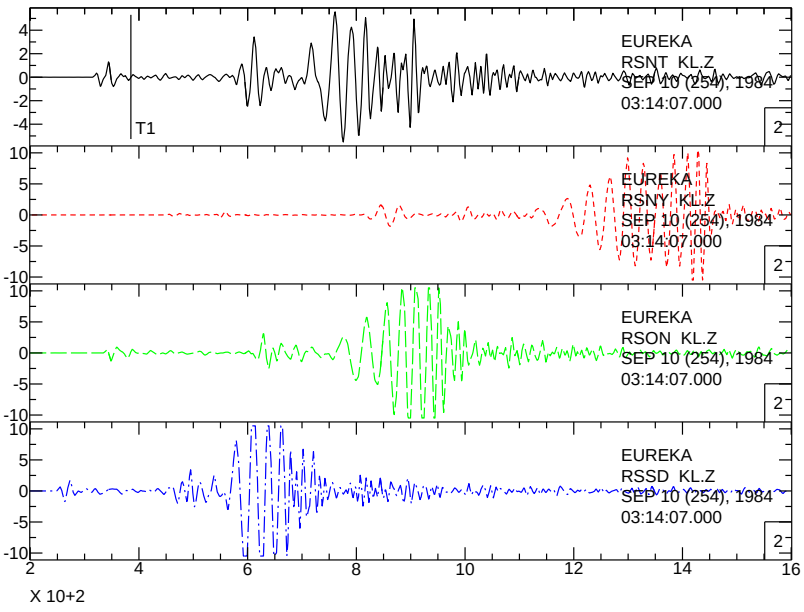

图 5.3: 线条属性示意图 2

5.5　组合图

SAC 中与窗口有关的概念有三个，如图 5.4：

- windows：对于 xwindows 设备，window 即图 2.1 中的大小，其默认长宽比为 11.0/8.5=1.294；对于 sgf 或者 pdf 和 ps，可以认为 window 的大小即为 A4 纸张的大小。
- viewspace：window 内可以用于绘图的部分；
- viewport：对于组合图，每个子图占据的空间；

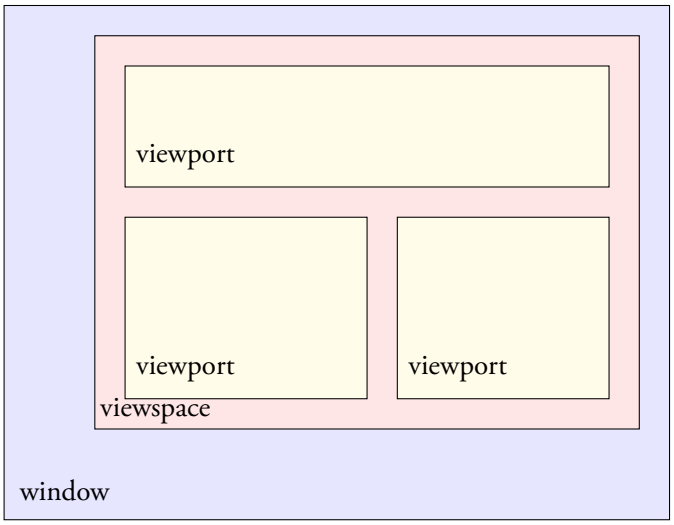

图 5.4: window、viewspace 和 viewport

默认情况下，每次在使用绘图命令时，整个绘图窗口会刷新，所以每次都会显示不同的图像。 `beginframe` 命令会关闭窗口的自动更新，因而可以用于绘制组合图。当然还有 `endframe` 来重新启动窗口的自动更新。

```
SAC> fg seis                          // 生成数据
SAC> beginframe                       // 关闭自动更新
SAC> xvport 0.1 0.9                   // 设定X viewport
SAC> yvport 0.7 0.9                   // 设定Y viewport
SAC> title 'Seismic Trace'            // 设定标题
SAC> fileid off                       // 不显示文件id
SAC> qdp off                          // 关闭快速绘图
SAC> p
SAC> fft wmean                        // FFT
SAC> xvport .1 .45
SAC> yvport .15 .55
SAC> title 'Amplitude Response (linlog)'
SAC> ylim 1e-5 1                      // Y轴范围
SAC> psp am linlog                    // 绘制振幅谱 (linlog)
SAC> xvport .55 .9
SAC> title 'Amplitude Response (loglog)'
SAC> xlim 1 60
SAC> psp am loglog                    // 绘制振幅谱 (loglog)
SAC> endframe                         // 启动自动更新
```

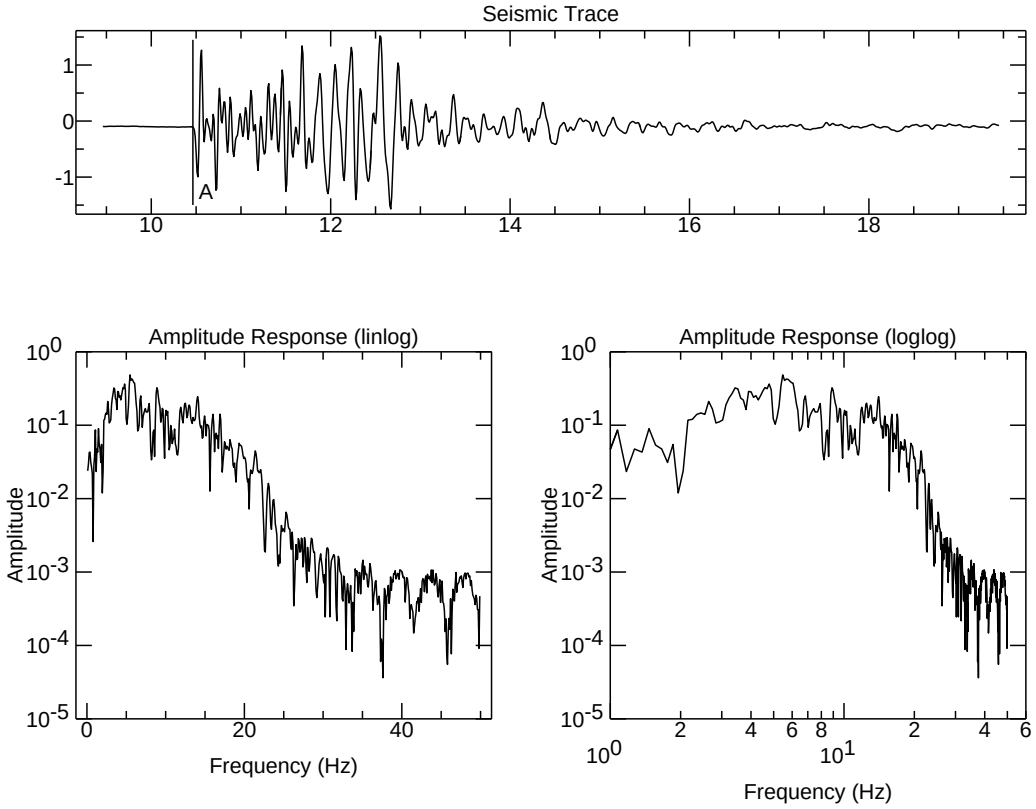

图 5.5: 绘制组合图

5.6 图像设备与图像保存

5.6.1 xwindows

xwindows 是 SAC 中最常用也是默认的绘图设备。对于震相拾取等交互式操作更是比不可缺。

```
SAC> fg seis
SAC> bd x       // begindevice xwinows, 可省略
SAC> p          // 绘图
SAC> ed x       // enddevice xwindows, 可省略
SAC> q
```

对于 xwindows，最简单的保存图像的方式是截图，常用的工具包括 gnome 下的 screenshot 或者 ImageMagick 的 import 命令。

5.6.2 sgf

SGF 图像设备是 SAC 自己设计的与平台无关的图像格式，其包含了 SAC 图像的全部信息。其流程为 “启用 sgf 设备”→“绘图到 sgf”→“关闭 sgf 设备，退出 SAC” →“将 sgf 文件转换为其它格式”。

```
SAC> fg seis
SAC> bd sgf         // 启动sgf设备, 不可省略
SAC> p
SAC> ed sgf         // 关闭sgf设备, 可省略
SAC> q
```

```
$ ls
f001.sgf            // 生成sgf文件
```

生成的 sgf 文件可以通过 sgftops 等命令转换为其它图像格式，在 “sgftops” 中会介绍，也可以使用 sgftox 直接将 sgf 文件显示在绘图窗口中。

5.6.3　PS 和 PDF

自 101.5 之后，SAC 加入了 saveimg 命令，以获得更高质量的绘图效果。saveimg 会将当前 xwindow 窗口显示的图像以 ps 或 pdf 格式保存到图像文件中。

```
SAC> fg seis
SAC> p                      // 首先在xwindows上绘图
SAC> saveimg foo.ps         // 将xwindows上的图像保存到foo.ps中
save file foo.ps [PS]
SAC> q
```

5.6.4　pssac

pssac 是 Prof. Lupei Zhu 写的 C 程序。其利用 GMT 的 PS 绘图库，直接读取 SAC 文件并绘制到 PS 文件中，得益于 GMT 的 PS 库的灵活性，利用 pssac 可以绘制出超高质量的图像。关于 pssac 的用法，参见 “pssac” 一节。

5.6.5　图像保存小结

- xwindows 绘图简单省事，但质量较差；
- sgf 转换稍显麻烦，但适合在脚本中批量做图；
- saveimg 生成的图件质量相对较高，可以满足大多数需求；
- pssac 功能强大，在需要设计复杂图像时非常有用；

第 6 章　SAC 编程

一个基本的编程语言需要包含哪些特性呢？变量、参数、函数、条件判断、循环控制等等。

这一章介绍 SAC 所设计的一个基本的编程语言，在官方文档中直接称其为 SAC 宏。SAC 与 SAC 宏的关系在某种程度上更像是 matlab 与 matlab 脚本之间的关系。最简单的，将一系列要一起执行的 SAC 命令放在一个文件中即构成了 SAC 宏文件。

本文档中使用了稍有不同的说法，并将这一章命令为 “SAC 编程”。其包含了三个主要的部分：

- 变量：由于 SAC 的特殊性，又分为头段变量和一般变量；
- 内置函数：基本的数学和字符串函数；
- SAC 宏：参数、条件判断、循环控制等；

在官方文档中，变量和内置函数都是 SAC 宏的一部分，本文档将其从 SAC 宏中提取出来是出于如下几个方面的考虑：

1. 变量和内置函数既可以在 SAC 命令中使用，也可以在 SAC 宏中使用；而其它特性如参数、条件判断、循环控制等几乎只能在 SAC 宏中使用；
2. SAC 设计的编程语言功能简单，不够友好；非常建议使用 Bash、Perl 或 Python 这些更成熟的脚本语言来替代 SAC 的编程功能。宏参数、条件判断、循环控制等特性都可以被脚本的相应功能完全取代，而由于 SAC 设计的特殊性，诸如变量和内置函数等特性在某些情况下不能完全取代。

所以，建议的做法是读完本章的内容，掌握如何引用头段变量、如何使用黑板变量以及内联函数，简单了解 SAC 宏的特性，选择 Perl 或者 Python 脚本语言 [1] 进行数据批量处理，尽量避免使用黑板变量和内联函数。就目前的个人经验而言，在脚本语言中，SAC 的引用头段变量功能必不可少，黑板变量和内联函数这两个功能或多或少都可以被取代。

另外，尤其需要注意的是，SAC 自 101.6 起彻底重写了 SAC 宏的语法分析器，因而导致 101.6 以后的 SAC 宏与之前的 SAC 宏有很大不同，本文档只讨论 101.6 重写的 SAC 宏语法。

[1] Bash 在很多地方还是不如 Perl 或 Python 方便，不推荐。

6.1 引用头段变量值

前面已经介绍了 SAC 中的很多头段变量，也知道如何使用 `listhdr` 查看头段变量的值，lh 命令的输出对于人来说很直观，但是对于机器来说却很不友好。有些时候需要直接使用头段变量的值，这就需要一些特殊的技巧。

最常见的情况是第 35 页给出的例子。在使用 “ch o gmt” 指定发震时刻后，需要获取头段变量 o 的值，对该值取负值，并用于 “ch allt” 中。

在这个例子中，我们需要知道头段变量 o 的值，并将其值用于其它命令中，准确的说这叫变量值的引用。在 SAC 命令中引用 SAC 头段变量的值有两种方式，分别是 “`&fname,header&`” 和 “`&fno,header&`” [1] 。

fname 和 fno 都唯一指向了内存中的某个波形数据，其中 fname 表示文件名，fno 表示文件号（即内存中的第几个文件，索引值从 1 开始），header 则为头段变量名。

下例展示了如何通过两种方式引用头段变量的值：

```
SAC> fg seis
SAC> w seis.SAC
SAC> r ./seis.SAC               // 注意"./"
SAC> lh kevnm o stla            // 查看三个头段变量的值

  FILE: ./seis.SAC - 1          // 这里给出了文件名和文件号
 ----------------

     kevnm = K8108838
         o = -4.143000e+01
      stla = 4.800000e+01
SAC> echo on processed          // 打开回显，显示处理信息
SAC> ch kuser0 &1,kevnm&        // 通过文件号引用头段变量kevnm
 ==>  ch kuser0 K8108838        // 实际执行的效果
SAC> ch user0 &./seis.SAC,o&    // 利用文件名，引用头段变量o
 ==>  ch user0 -41.43
SAC> ch user1 &seis.SAC,stla&   // 文件名少了"./"
 ERROR 1363: Illegal data file list name: seis.SAC
SAC> lh kuser0 user0 user1

  FILE: ./seis.SAC - 1
   ----------------

     kuser0 = K8108838
     user0 = -4.143000e+01
```

在通过文件名指定波形数据时要注意：SAC 记录的是文件的全路径。一般情况下，使用文件号会更方便些。

6.2 黑板变量

既然是 SAC 编程，就必然少不了变量，SAC 中的变量称之为黑板变量。

[1] 实际上，SAC 官方文档给出的引用方式中没有末尾的 & 符号，仅当一些特殊的情况下才使用，这样容易使得整个语法混乱不堪，所以这里采用了另外一种引用方式。所有示例均已通过测试。

黑板变量是 SAC 中用于临时储存和取回信息而设计的。黑板变量不需要声明即可直接使用，可以用 `setbb` 和 `evaluate` 命令给黑板变量赋值，用 `getbb` 获取黑板变量的值。也可以用 `writebbf` 将黑板变量保存在磁盘文件中，然后使用 `readbbf` 命令重新将这些变量读入 SAC 中。

引用黑板变量的值的方式为：“`%bbvname%`”，其中 bbvname 为黑板变量的变量名。

```
SAC> echo on processed
SAC> fg seis
SAC> p
SAC> setbb low 2.45          // 黑板变量low=2.45
SAC> setbb high 4.94         // 黑板变量high=4.94
SAC> bp c %low% %high%       // 引用黑板变量low和high的值作为滤波的频带
 ==>  bp c 2.45 4.94         // echo on processed 显示代入值后的命令
SAC> p
SAC> getbb low high          // 查看黑板变量的值
 low = 2.45
 high = 4.94
```

下例展示了如何将黑板变量写入磁盘文件，等需要时再从磁盘文件中获取：

```
$ sac
SAC> setbb var1 10           // 整型
SAC> setbb var2 "text"       // 字符串
SAC> setbb var3 0.2          // 浮点型
SAC> wbbf bbf.file           // 写入到文件
SAC> q
$ ls
bbf.file
$ sac
SAC> readbbf ./bbf.file
SAC> getbb
 NUMERROR = 0
 SACERROR = 'FALSE'
 SACNFILES = 0
 VAR1 = 10
 VAR2 = 'text'
 VAR3 = 0.2
SAC> getbb var2 var3
 var2 = 'text'
 var3 = 0.2
SAC> q
```

6.3　内联函数

内联函数是 SAC 实现的一些函数，其可以在 SAC 命令中使用。在执行命令时，内联函数会首先被调用，内联函数的结果将替代命令中的内联函数的位置。

SAC 提供了如下几类内联函数：

- 算术运算符；
- 常规算术运算函数；
- 字符串操作函数；
- 其他函数；

所有的内联函数的共同形式是："(func)"，其中 func 为内联函数名。某种程度上，内联函数与前面说到头段变量 ("& &") 和黑板变量 ("% %") 类似，可以认为是通过 "( )" 引用了内联函数的结果或值。

内联函数支持嵌套，目前最多可以嵌套 10 层。

6.3.1 算术运算符

算术运算符即常规的加减乘除运算符，但又有不同，其一般形式如下：

```
( number operator number )
```

所有的操作数都被认为是实型的，所有的算术运算都按照双精度浮点型进行运算；

SAC 支持的操作符是包括："+ - * / ** "

看几个简单的例子：

```
SAC> echo on
SAC> setbb var1 4+7              // 忘记加括号了！"4+7"被当成了字符串
 setbb var1 4+7
SAC> setbb var2 (4+7)
 setbb var2 (4+7)
 ==>  setbb var2 11              // 4+7=11
SAC> setbb var3 (4+7/3)          // 优先级正确
 setbb var3 (4+7/3)
 ==>  setbb var3 6.33333
SAC> setbb var4 ((4+7)/3)        // 括号改变优先级
 setbb var4 ((4+7)/3)            // 可以看作是内联函数的嵌套
 ==>  setbb var4 3.66667
SAC> setbb var1 ( ( 4 + 7 ) / 3 )   // 支持空格
 setbb var1 ( ( 4 + 7 ) / 3 )
 ==>  setbb var1 3.66667
```

6.3.2 常规算术运算函数

SAC 提供了 20 个常规算术运算函数，其基本形式为 "(func arg1 arg2 ...)"。具体函数如表 6.1所示。

演示如下：

```
SAC> echo on processed
SAC> setbb var1 (add 1 3 4)          // 1+3+4
 ==>  setbb var1 8
SAC> setbb var2 (subtract 1 3 4)     // 1-3-4
 ==>  setbb var2 -6
SAC> setbb var3 (multiply 1 3 4)     // 1*3*4
 ==>  setbb var3 12
SAC> setbb var4 (divide 1 3 4)       // 1/3/4
 ==>  setbb var4 0.0833333
SAC> setbb var5 ( absolute -5.1 )    // abs(-5.1)
 ==>  setbb var5 5.1
SAC> setbb var6 ( power 5 )          // 10^5
 ==>  setbb var6 100000
SAC> setbb var7 ( alog10 10000 )     // log10(10000)
 ==>  setbb var7 4
SAC> setbb var8 ( alog 10000 )       // ln(10000)
 ==>  setbb var8 9.21034
```

表 6.1: 常规算数运算函数

命令	语法	功能
add	(add v1 v2 ... vn)	v1+v2+...+vn
subtract	(subtract v1 v2 ... vn)	v1-v2-...-vn
multiply	(multiply v1 v2 ... vn)	v1*v2*...*vn
divide	(divide v1 v2 ... vn)	v1/v2/.../vn
absolute	(absolute v)	取绝对值
power	(power v)	取 10 的 v 次方
alog10	(alog10 v)	以 10 为底取 v 的对数
alog	(alog v)	取 v 的自然对数
exp	(exp v)	取 e 的 v 次方
sqrt	(sqrt v)	求 v 的平方根
pi	(pi)	返回 pi 值
sine	(sine v)	正弦（v 为弧度，下同）
cosine	(cosine v)	余弦
tangent	(tangent v)	正切
arcsine	(arcsine v)	反正弦
arccosine	(arccosine v)	反余弦
arctangent	(arctangent v)	反正切
integer	(integer v)	取整
maximum	(maximum v1 v2 ... vn)	求最大值
minimum	(minimum v1 v2 ... vn)	求最小值

```
SAC> setbb var9 ( exp 5 )           // e^5
 ==>  setbb var9 148.413
SAC> setbb var10 ( sqrt 9 )         // sqrt(9)
 ==>  setbb var10 3
SAC> setbb var11 ( pi )             // PI
 ==>  setbb var11 3.14159
 SAC> setbb var12 ( sine (pi/6) )   // sin(30)
 ==>  setbb var12 0.5
SAC> setbb var13 ((arcsine 0.5)*180/(pi))
 ==>  setbb var13 30
SAC> setbb var14 (integer 3.11)
 ==>  setbb var14 3
SAC> setbb var15 (max 3.11 -1.5 5)  // maximum 简写为 max
 ==>  setbb var15 5
SAC> setbb var16 (min 3.11 -1.5 5)  // minimum 简写为 min
 ==>  setbb var16 -1.5
```

为了对一组数据做归一化，首先要找到所有数据中的绝对最大值，如下：

```
SAC> r file1 file2 file3 file4
SAC> echo on processed
SAC> setbb vmax (max &1,depmax& &2,depmax& &3,depmax& &4,depmax&)
 ==> setbb vmax 1.87324
SAC> setbb vmin (min &1,depmin& &2,depmin& &3,depmin& &4,depmin&)
 ==> setbb vmin -2.123371
SAC> div ( max (abs %vmax%) (abs %vmin%) )      // 嵌套
```

```
 ==>  div 2.123371
```

此例可以通过多重嵌套的方式在单个命令中完成，但上面的写法可读性更强。

6.3.3 字符串操作函数

SAC 提供了若干个函数用于字符串的处理，如表 6.2 所示：

表 6.2: 字符串操作函数

命令	语法（简写形式）	功能
change	(cha s1 s2 s3)	在 s3 中用 s1 代替 s2
substring	(substring n1 n2 s)	取 s 中第 n1 到第 n2 个字符
delete	(del s1 s2)	从 s2 中删去 s1
concatenate	(conc s1 s2 ... sn)	将多个字符串拼接起来
before	(bef s1 s2)	得到 s2 中位于 s1 前的部分字符串
after	(aft s1 s2)	得到 s2 中位于 s1 后的部分字符串
reply	(rep s1)	发送信息 s1 到终端并得到回应

下面的例子展示了部分函数的用法：

```
SAC> echo on processed
SAC> setbb var1 (cha short long "this is short")
 ==>  setbb var1 this is long
SAC> set var2 (del def abcdefghi)
 ==>  set var2 abcghi
SAC> set var4 (before de abcdefg)
 ==>  set var4 abc
SAC> set var4 (after de abcdefg)
 ==>  set var4 fg
SAC> fg seis
SAC> setbb month (substring 1 3 &1,kzdate&)
 ==>  setbb month MAR
SAC> setbb val "1234567890"
SAC> message (substring 1 5 %val%)
 ==>  message 12345
 12345
```

下面的例子展示 concatenate 函数的用法以及如何灵活定义标题：

```
SAC> fg seis
SAC> echo on processed
SAC> setbb var (conc Seismogram of &1,kevnm& &1,kstnm&)
 ==>  setbb var SeismogramofK8108838CDV                // 没有空格
SAC> setbb var (conc "Seismogram of " &1,kevnm& " " &1,kstnm&)
 ==>  setbb var Seismogram of K8108838 CDV             // 含空格
SAC> getbb var
 var = 'Seismogram of K8108838 CDV'
SAC> title (conc "Seismogram of " &1,kevnm& " " &1,kstnm&)
 ==>  title Seismogram of K8108838 CDV                 // 错误标题！
SAC> title '(conc "Seismogram of " &1,kevnm& " " &1,kstnm&)'
 ==>  title "(conc "Seismogram of " K8108838 " " CDV)"  // 错误标题！
SAC> title "Seismogram of &1,kevnm& &1,kstnm&"
 ==>  title "Seismogram of K8108838 CDV"               // 正确标题！
```

下面的例子使用 reply 函数实现了交互：

```
SAC> fg seis
SAC> echo on processed
SAC> rmean; rtr; taper
SAC> setbb low (reply "Enter low freqency limit for bandpass: ")
Enter low freqency limit for bandpass: 2.1          // 用户输入2.1
 ==>  setbb low 2.1
SAC> setbb high (reply "Enter low freqency limit for bandpass: ")
Enter low freqency limit for bandpass: 6.5          // 用户输入6.5
 ==>  setbb high 6.5
SAC> bp c %low% %high%
 ==>  bp c 2.1 6.5
```

下面的例子中 reply 函数包含了一个默认值值：

```
SAC> setbb bbday (reply "Enter the day of the week: [Monday]")
Enter the day of the week: [Monday]Tuesday      // 用户输入Tuesday
SAC> getbb bbday
 bbday = 'Tuesday'
SAC> setbb bbday (reply "Enter the day of the week: [Monday]")
Enter the day of the week: [Monday]             // 用户无输入
SAC> getbb bbday
 bbday = 'Monday'
```

当 reply 函数执行时，引号中的字符串将出现在屏幕上，提示用户输入。如果用户输入，SAC 会将输入的字符串作为返回值，如果用户只是敲击回车键，SAC 则会使用该默认值“MONDAY”。

6.3.4　其他函数

这类函数目前只有一个：gettime，其语法为“`(gettime max|min [value])`。

gettime 函数用于返回数据中首先出现大于或小于 value 的时间相对于文件参考时刻的相对时间；若没有指定 value，max 会返回文件中第一个最大值的相对时间，min 会返回文件中第一个最小值的相对时间。

对于所有的文件有一个最大振幅，要找到这些文件中第一个文件中第一次大于该值所对应的时间偏移量：

```
SAC> fg seis
SAC> echo on processed
SAC> setbb maxtime (gettime max)
 ==>  setbb maxtime 12.55
SAC> setbb mintime (gettime min)
 ==>  setbb mintime 12.67
```

为了找到第一个大于或等于 1.0 的数据点的时间偏移，可以使用如下命令：

```
SAC> fg seis
SAC> echo on processed
SAC> setbb valuetime ( gettime max 1.0 )
 ==> setbb valuatime 10.55
```

6.4 SAC 宏

6.4.1 简单的例子

假如你有一些重复的工作需要完成，那么 SAC 宏显然可以帮你节省不少时间。例如，要经常读取三个文件 ABC、DEF 和 XYZ，每个文件分别乘以不同的值，做 Fourier 变换，然后将频谱的振幅部分绘制到 SGF 文件中，这样的一系列命令可以写入到 SAC 宏文件中：

```
** This certainly is a simple little macro.
r ABC DEF XYZ
mul 4 8 9
fft
bg sgf
psp am
```

假设上面的代码保存到文件 mystuff 中，且该文件位于当前目录中，可以通过下面的命令执行该宏文件：

```
SAC> macro mystuff
```

终端中并不会显示正在执行的宏文件中的命令，可以使用 echo 命令来设置在终端显示哪些东西。另外，任意一行的第一列若有星号则意味着该行为注释行，SAC 不会不解释该行。

6.4.2 宏搜索路径

当你执行一个宏文件而又没有给出宏文件的绝对路径时,SAC 会按照下面的路径顺序搜索宏文件：

1. 在当前目录搜索；
2. 在 setmacro 命令设置的搜索目录中搜索；
3. 在 SAC 的全局宏目录（`sac/aux/macros`）中搜索；

所有人都可以使用全局宏目录中的宏文件，可以使用 installmacro 命令将自己的宏文件安装到这个目录中。你也可以通过绝对/相对路径指定搜索路径。

6.4.3 宏参数

如果想要每次读取不同的文件或者乘以不同的值那么必须每次都修改该文件，让宏文件在执行之前允许用户输入参数可以大大增加宏文件的灵活性。

SAC 宏参数的格式为：“`$n$`”，其中 n 从 1 开始。

下面将对先前的宏文件进行修改以使其可以接收文件名作为参数：

```
r $1$ $2$ $3$
mul 4 8 9
fft
bg sgf
psp am
```

“`$1$`”、“`$2$`”和“`$3$`”分别表示宏文件接收到的第一、二、三个参数，用下面的命令执行这个宏文件：

```
SAC> macro mystuff ABC DEF XYZ
```

可以用下面的命令再次执行这个宏文件，但读取不同的文件：

```
SAC> macro mystuff AAA BBB CCC
```

6.4.4　关键字驱动参数

关键字驱动参数允许用户按照任意顺序输入参数，这也使得宏文件的内容变得简单易懂。

当参数的数目以及宏文件的大小不断增大的时候这就变得更加重要了。下面将再一次修改这个例子以使其可以接受文件列表以及乘数的列表：

```
$keys$ files values
r $files$
mul $values$
fft
bg sgf
psp am
```

`$keys$` 表明 “files” 和 “values” 是关键字。可以按照下面的输入来执行这个宏文件：

```
SAC> macro mystuff files ABC DEF XYZ values 4 8 9
```

因为参数的顺序不再重要，所以你可以像下面这样输入：

```
SAC> macro mystuff values 4 8 9 files ABC DEF XYZ
```

这个宏文件并不限于读取三个文件，它对于文件的数目没有限制，只要文件数与值数目相匹配就好。

6.4.5　宏参数缺省值

有些时候会遇到这样的情况，宏文件的有些参数在多次执行的过程中经常但并不总是拥有相同的值。为这些参数提供缺省值可以减少输入那些相同值的次数同时又保有宏参数本身的灵活性。如下例所示：

```
$keys$ files values
$default$ values 4 8 9
r $files$
mul $values$
fft
bg sgf
psp am
```

`$default$` 指定了宏参数 `values` 的缺省值，若在执行宏文件时不输入 values 的参数值那么这些参数将使用缺省值：

```
SAC> macro mystuff files ABC DEF XYZ
```

如果想要使用不同的值，可以像下面这样输入：

```
SAC> macro mystuff values 10 12 3 files ABC DEF XYZ
```

6.4.6 参数请求

若执行宏文件时没有输入参数而这些参数又没有缺省值，SAC 会在终端中提示你输入相应的参数值。在上面的例子中，如果你忘记输入参数则会出现下面的情况：

```
SAC> macro mystuff
files? ABC DEF XYZ          // 用户输入ABC DEF XYZ
```

注意到 SAC 并不会提示输入参数 values 的值，因为它们已经有了缺省值。SAC 并非在一开始就提示输入参数，其等到需要计算参数值却发现没有缺省值或者输入值时才会提示需要输入该参数。

6.4.7 联接

头段变量、黑板变量、宏参数以及字符串可以直接联接在一起。

```
$keys$ station
fg seis
echo on
setbb sta $station$.z
setbb tmp ABC
setbb tmp1 XYZ%tmp%
setbb tmp2 (&1,o&)
setbb fname $station$%tmp%%tmp1%%tmp2%.SAC
```

执行效果如下：

```
SAC> m stuff station STA
 setbb sta $station$.z
 ==>  setbb sta STA.z
 setbb tmp ABC
 setbb tmp1 XYZ%tmp%
 ==>  setbb tmp1 XYZABC
 setbb tmp2 @(&1,o&@)
 ==>  setbb tmp2 (-41.43)
 setbb fname $station$%tmp%%tmp1%%tmp2%.SAC
 ==>  setbb fname STAABCXYZABC(-41.43).SAC
```

6.4.8 条件判断

条件判断在任何一个编程语言中都是必不可少的，SAC 宏的条件判断语句与 Fortran77 类似，但不完全相同，要注意区分。

SAC 宏的条件判断格式如下：

```
  IF expr
        commands
  ELSEIF expr
        commands
  ELSE
        commands
  ENDIF
```

逻辑表达式 expr 具有如下形式：

```
    token 关系运算符 token
```

其中 token 可以是一个常数、宏参数、黑板变量或头段变量，关系运算符则是 GT、GE、LE、LT、EQ、NE 中的一个。上面的逻辑表达式在计算之前 token 会被转换为浮点型数。

条件判断语句目前最多支持 10 次嵌套，且 elseif、else 是可选的，elseif 的次数没有限制。下面给出一个例子：

```
r $1$
markptp
if &1,user0& ge 2.45
    fft
    psp am
else
    message "Peak to peak for $1 below threshold."
endif
```

在这个例子中，一个文件被读入内存，markptp 测出其最大峰峰值，并保存到头段变量 user0 中，若该值大于某一确定值，则对其做 Fourier 变换并绘制振幅图，否则输出信息到终端。

6.4.9　循环控制

循环特性允许在一个宏文件中重复执行一系列命令。通过固定循环次数、遍历元素列表或者设定条件来执行一系列命令，也可以随时中断一次循环。循环的最大嵌套次数为 10 次。其语法可以有多种形式：

```
DO variable = start, stop [,increment]
    commands
ENDDO
```

```
DO variable FROM start TO stop [BY increment]
    commands
ENDDO
```

```
DO variable LIST entrylist
    commands
ENDDO
```

```
DO variable WILD [DIR name] entrylist
    commands
ENDDO
```

```
WHILE expr
    commands
ENDDO
```

其中大写字符串均为关键字，不可更改：

- variable 是循环变量名，在变量名前后加上 “$” 即可在 do 循环中引用该变量；
- start、stop、increment 循环变量的初值、终值、增值，start、stop 必须为整型数，increment 缺省值为 1
- entrylist 是 do 循环执行时变量可以取的所有值的集合，值之间以空格分开，其可以为整型、浮点型或字符型。DO WILD 中 entrylist 由字符串和通配符构成，循环执行前，这个列表将根据通配符扩展为一系列文件名。

下面给出一些 DO 循环的例子：

该宏文件对数据使用了 DIF 以进行预白化处理，进行 Fourier 变换，然后使用 DIVOMEGA 命令去除预白化的影响，有时需要在做变换之前多次预白化，那么就可以这样写：

```
$keys$ file nprew
$default$ nprew 1
r $file
do j = 1 , $nprew$
    dif
enddo
fft amph
do j = 1 , $nprew$
    divomega
enddo
```

下面这个例子，用相同的数据绘制 5 个不同的两秒时间窗的质点运动矢量图：

```
r abc.r abc.t
setbb time1 0
do time2 from 2 to 10 by 2
    xlim %time1% $time2$
    title 'Particle motion from %time1% to $time2$'
    plotpm
    setbb time1 $time2$
enddo
```

在下面的例子中，一个宏文件调用另一个名为 preview 的宏文件，通过 do 循环以达到多次调用 preview 的目的：

```
do station list abc def xyz
    do component list z n e
        macro preview $station$.$component$
    enddo
enddo
```

在下面的例子中我们修改上一个宏文件使得其可以处理目录 mydir 中所有以 ".Z" 结束的文件：

```
do file wild dir mydir *.Z
    macro preview $file$
enddo
```

最后一个例子有三个参数，第一个是文件名，第二个是一个常数，第三个是一个阀值。宏文件读取了一个数据文件，然后每个数据点乘以一个常数直到其超过某一阀值：

```
r $1$
while &1,depmax& gt $3$
    mul $2$
enddo
```

另一个与 break 有关的宏文件：

```
r $1$
while 1 gt 0
```

```
    div $2
    if &1,depmax& gt $3$
        break
    endif
enddo
```

这个 while 循环是一个无限循环，它只能通过 break 来中断。

6.4.10　嵌套与递归

SAC 宏提供嵌套功能，不支持递归，但是 SAC 并不会去检查宏的调用是否保证不是递归，因而需要用户去保证宏文件不要直接或间接调用自己。

6.4.11　中断宏

有些时候需要临时中断宏文件的执行，用户自己从终端输入一些命令，然后继续执行宏文件。这个可以利用 SAC 的 pause 和 resume 特性做到。当 SAC 在宏文件中遇到 “$TERMINAL$” 时它会临时停止执行宏文件，更改提示符为宏名，然后提示从终端输入命令，然后当 SAC 在终端中看到 “$RESUME$” 时则会停止从终端读取命令继续从宏文件读取。如果你不想再继续执行宏文件中的命令，可以在终端输入 “$KILL$” ，SAC 将关闭宏文件，回到上一层。在一个宏文件中可以有多个 “$TERMINAL$” 中断。

6.4.12　宏文件中执行其它程序

你可以在 SAC 宏内部执行其他程序，可以向程序传递参数。如果程序是交互式的你也可以将输入行发送给它，语法如下：

```
$RUN$ program message
inputlines
ENDRUN
```

宏参数、黑板变量、头段变量、内联函数均可使用，在程序执行之前它们会被计算，当程序执行结束，SAC 宏会在 ENDRUN 之后继续执行。

6.4.13　转义字符

字符 “$” 和 “%” 在 SAC 中具有特殊的含义，有时在字符串中需要使用这些特殊字符，但 SAC 会将其解释成一个变量，此时就需要使用转义字符，SAC 中的转义符为 “@” ，可以被转义的特殊符号包括：

- $ 宏参数标识符
- % 黑板变量标识符
- & 头段变量标识符
- @ 转义字符本身
- ( ) 内联函数起始符

第 7 章　在脚本中调用 SAC

7.1　Bash 中调用 SAC

7.1.1　简介

SAC 宏的功能相对比较单一，难以满足日常数据处理的需求，可以在 Bash 脚本中直接调用 SAC，这样可以利用 Bash 脚本的更多特性。

下面的例子展示了如何在 Bash 脚本中调用 SAC:

```
#!/bin/bash
export SAC_DISPLAY_COPYRIGHT=0

sac << EOF
fg seis
lh evla evlo
q               # 必须！
EOF
```

SAC 在启动是默认会显示版本信息，当用脚本多次调用 SAC 时，版本信息也会显示多次，可以通过设置环境变量 “`export SAC_DISPLAY_COPYRIGHT=0`” 的方式隐藏版本信息。

脚本中从 “`sac << EOF`” 开始到 “`EOF`” 的全部内容，都会被 Bash 传递给 SAC，SAC 会逐一解释并执行每行命令。

7.1.2　头段变量和黑板变量

想要在 Bash 脚本中引用头段变量，需要借助于 SAC 宏的语法。

```
#!/bin/bash
export SAC_DISPLAY_COPYRIGHT=0

sac << EOF
fg seis
ch kuser0 &1,kevnm&
setbb tmp ABC
ch kuser1 %tmp%
lh kuser0 kuser1
quit
EOF
```

7.1.3 内联函数

bash 可以完成基本的数学运算，但是所有的运算只支持整型数据，浮点型运算或者其它更高级的数学运算需要借助 bc 或者 awk 来完成。Bash 中的变量以 “$” 作为标识符，Bash 会首先做变量替换再将替换后的命令传递给 SAC。

```
#!/bin/bash
export SAC_DISPLAY_COPYRIGHT=0

declare -i var1 var2 var3 var4
var1=(1+2)*3
var2=10/4
var3=10/4
echo $var1 $var2 $var3

sac << EOF
echo on
fg seis
bp c $var2 $var1
q
EOF
```

本例中的变量 “$var1” 和 “$var2” 会首先被 SAC 解释成为 1 和 2，因而 SAC 实际接收到的命令是 “bp c 1 2”。

借助于 awk、sed 等工具，也可以实现部分字符串处理函数：

```
#!/bin/bash
str1=`echo "this is long" | sed 's/long/short/'`    # 替换
str2=`echo "abcdefghi" | sed 's/def//'`             # 删除

sac << EOF
fg seis
title "$str1"
p
saveimg string.ps
q
EOF
```

7.1.4 条件判断和循环控制

Bash 具有更灵活的条件判断和循环控制功能，但由于 Bash 自身的限制，这些特性仅能在 SAC 外部使用，因而下例中需要多次调用 SAC，在某些情况下会相当耗时。

```
#!/bin/sh
export SAC_DISPLAY_COPYRIGHT=0

for file in *.SAC; do
   sac <<EOF
     read $file
     rmean
     rtrend
     lp co 1.0 p 2 n 4
     write ${file}.filtered
     quit
EOF
done
```

7.2　在 Perl 中调用 SAC

7.2.1　简介

下面的脚本中给出了一个简单的例子，展示了如何在 Perl 中调用 SAC。相对于 Bash 来说，Perl 脚本似乎需要更多的键入，但是相对于 Perl 的优势来说，这些都不算什么。

```
#!/usr/bin/env perl
use strict;
use warnings;
$ENV{SAC_DISPLAY_COPYRIGHT}=0;

open(SAC, "| sac ") or die "Error opening sac\n";
print SAC "fg seismo \n";
print SAC "lh evla kstnm \n";
print SAC "quit \n";
close(SAC);
```

7.2.2　头段变量

Perl 无法直接引用 SAC 文件的头段变量值，依然需要利用 SAC 宏的的语法。Perl 的优势在于可以一边向 SAC 传递信息，一边运行自身的命令。

```
#!/usr/bin/env perl
use strict;
use warnings;
$ENV{SAC_DISPLAY_COPYRIGHT}=0;

open(SAC, "| sac ") or die "Error opening sac\n";
print SAC "fg seismo \n";
print SAC "ch kuser0 &1,kevnm& \n";
my $tmp = "ABC";                # Perl中的变量
print SAC "ch kuser1 $tmp \n";
print SAC "lh kuser0 kuser1 \n";
print SAC "quit \n";
close(SAC);
```

本例中第 9 行，在 SAC 运行的同时 Perl 临时定义了一个变量，并成功将其传递给了 SAC，这在 Bash 中是不容易做到的。

7.2.3　内联函数

Perl 可以完成各种复杂的数学运算：

```
#!/usr/bin/env perl
use strict;
use warnings;
use Math::Trig;
$ENV{SAC_DISPLAY_COPYRIGHT}=0;

my $var1 = (1+2)*3/4;
print "$var1 $var2 $var3\n";

open(SAC, "| sac") or die "Error in opening sac\n";
print SAC "fg seis \n";
print SAC "rmean; rtr; taper \n";
my $var2 = sqrt(9);
```

```
my $var3 = sin(30*pi/180);
print SAC "bp c $var3 $var1 \n";
print SAC "q \n";
close(SAC);
```

Perl 对字符串的处理更是 Perl 的杀手锏：

```
#!/usr/bin/env perl
use strict;
use warnings;

my $str = "abcdefghi";

my $pos = index($str, "def");
my $len = length($str);
my $sub = substr($str, 2, 2);

print "$pos, $len, $sub \n";
```

7.2.4 条件判断和循环控制

Perl 也可以很容易地实现条件判断和循环控制：

```
#!/usr/bin/env perl
use strict;
use warnings;

$ENV{SAC_DISPLAY_COPYRIGHT}=0;
open(SAC, "| sac ") or die "Error opening sac";
print SAC "echo on \n";
foreach $file ( glob("*.SAC") ) {
    print SAC "r $file \n";
    print SAC "rmean \n";
    print SAC "rtrend \n";
    print SAC "lp co 1.0 p 2 n 4 \n";
    print SAC "write ${file}.filtered \n";
}
print SAC "quit \n";
close(SAC);
```

这个例子中只启动了一次 SAC，然后开始 Perl 的循环控制，读取当前目录下的每一个 SAC 文件，做一些数据处理，然后写到新文件中。

该 Perl 脚本与上一个 Bash 脚本相比，实现了几乎相同的功能，但 Perl 脚本中仅启动和退出 SAC 一次，与 Bash 脚本的多次启动相比，其效率要高很多。

第 8 章　使用 SAC 函数库

8.1　SAC 库简介

SAC 提供了两个函数库：libsacio.a 和 libsac.a，用户可以在自己的 C 或 Fortran 程序中直接使用函数库中的子函数。这些库文件位于 `sac/lib` 中。

8.1.1　libsacio 库

这个库文件的子函数可用于读写 SAC 数据文件、头段变量、黑板变量。这些子函数可以在用户的 C 或 Fortran 程序中直接使用。

libsacio.a 中可用的子函数包括：

表 8.1: libsacio 子函数

子函数	说明
`rsac1`	读取等间隔文件
`rsac2`	读取不等间隔文件和谱文件
`wsac1`	写入等间隔文件
`wsac2`	写入不等间隔文件
`wsac0`	可以写等间隔文件或不等间隔文件
`getfhv`	获取浮点型头段变量值
`setfhv`	设置浮点型头段变量值
`getihv`	获取枚举型头段变量值
`setihv`	设置枚举型头段变量值
`getkhv`	获取字符串头段变量值
`setkhv`	设置字符串头段变量值
`getlhv`	获取逻辑型头段变量值
`setlhv`	设置逻辑型头段变量值
`getnhv`	获取整型头段变量值
`setnhv`	设置整型头段变量值
`readbbf`	读取一个黑板变量文件
`writebbf`	写一个黑板变量文件
`getbbv`	获取一个黑板变量的值
`setbbv`	给一个黑板变量赋值

对于 C 源码，用如下命令编译

```
$ gcc -c source.c -I/usr/local/sac/include
$ gcc -o prog source.o -lm -L/usr/local/sac/lib -lsacio
```

也可以利用 SAC 提供的 sac-config 命令简化此编译命令：

```
$ gcc -c source.c `sac-config -c`
$ gcc -o prog source.o -lm `sac-config -l sacio`
```

对于 Fortran77 源码，用如下命令编译

```
$ gfortran -c source.f
$ gfortran -o prog source.o -L/opt/sac/lib/ -lsacio
```

也可以利用 SAC 提供的 sac-config 命令简化此编译命令：

```
$ gfortran -c souce.f
$ gfortran -o prog source.o `sac-config -l sacio`
```

8.1.2 libsac.a 库

这个库是从 101.2 版本才引入的，其是 libsacio.a 的超集，包含了几个数据处理常用的子函数。

libsac.a 包含如下子函数：

- xapiir 无限脉冲响应滤波器；
- firtrn 有限脉冲滤波器，Hilbert 变换；
- crscor 互相关；
- next2 返回比输入值大的最小的 2 的幂次；
- envelope 计算包络函数；

对于 C 源码，用如下命令编译

```
$ gcc -c source.c -I/usr/local/sac/include
$ gcc -o prog source.o -lm -L/usr/local/sac/lib -lsac
```

也可以利用 SAC 提供的 sac-config 命令简化此编译命令：

```
$ gcc -c source.c `sac-config -c`
$ gcc -o prog source.o -lm `sac-config -l sac`
```

对于 Fortran77 源码，用如下命令编译

```
$ gfortran -c source.f
$ gfortran -o prog source.o -L/opt/sac/lib/ -lsac
```

也可以利用 SAC 提供的 sac-config 命令简化此编译命令：

```
$ gfortran -c souce.f
$ gfortran -o prog source.o `sac-config -l sac`
```

8.2 调用 libsacio 库

8.2.1 rsac1

子函数 rsac1 用于读取等采样间隔的 SAC 数据。

函数定义如下：

```
void
rsac1(  char    *kname,     // 要读入的文件名
        float   *yarray,    // 数据被保存到yarray数组中
        int     *nlen,      // 数据长度
        float   *beg,       // 数据开始时间，即头段变量b
        float   *del,       // 数据采样周期，即头段变量delta
        int     *max_,      // 数组yarray的最大长度，若nlen>max_则截断
        int     *nerr,      // 错误标记，0代表成功，非零代表失败
        int      kname_s    // 数组kname的长度
)
```

相关示例代码为 `rsac1c.c` [1] 和 `rsac1f.f`。

8.2.2　rsac2

子函数 rsac2 用于读取非等间隔采样的。

```
void
rsac2(  char    *kname,     // 要读入的文件名
        float   *yarray,    // 因变量数组
        int     *nlen,      // 数据长度
        float   *xarray,    // 自变量数组
        int     *max_,      // 数组最大长度
        int     *nerr,      // 错误标记
        int      kname_s    // 数组kname的长度
)
```

相关示例代码为 `rsac2c.c` 和 `rsac2f.f`。

8.2.3　wsac1

子函数 wsac1 用于写等间隔 SAC 文件。

```
void
wsac1(  char  *kname,       // 要写入的文件名
        float *yarray,      // 要写入文件的数组
        int   *nlen,        // 数组长度
        float *beg,         // 数据起始时刻
        float *del,         // 数据采样周期
        int   *nerr,        // 错误标记
        int    kname_s      // 文件名长度
)
```

相关示例代码为 `wsac1c.c` 和 `wsac1f.f`。

8.2.4　wsac2

写非等间隔 SAC 文件。

```
void
wsac2(  char  *kname,       // 文件名
        float *yarray,      // 因变量数组
        int   *nlen,        // 数组长度
        float *xarray,      // 自变量数组
        int   *nerr,        // 错误码
        int    kname_s      // 文件名长度
)
```

相关示例代码为 `wsac2c.c` 和 `wsac2f.f`。

[1] 代码位于 `sac/doc/examples`，下同。

8.2.5 wsac0

子函数 wsac0 相对来说更加通用也更复杂，利用该函数可以创建包含更多头段的 SAC 文件。

```
void
wsac0(  char  *kname,       // 文件名
        float *xarray,      // 自变量数组
        float *yarray,      // 因变量数组
        int   *nerr,        // 错误码
        int    kname_s      // 文件名长度
)
```

要使用子函数 wsac0，首先要调用子函数 `newhdr()` 创建一个完全未定义的头段区，并利用其它头段变量相关子函数设置头段变量的值，并由 wsac0 写入到文件中。必须要赋值的头段变量为 delta、b、e、npts、iftype。

相关示例代码为 `wsacnc.c` 、 `wsacnf.f`。n=3,4,5。

8.2.6 getfhv

获取浮点型头段变量的值。

```
void
getfhv( char  *kname,       // 头段变量名
        float *fvalue,      // 浮点型头段变量的值
        int   *nerr,        // 错误码
        int    kname_s      // 变量名长度
)
```

相关示例代码为 `gethvc.c` 和 `gethvf.f`。

至于如何获取和设置其它类型的头段变量，方法类似，不再多说。

8.2.7 readbbf

读取一个黑板变量文件。

```
void
readbbf( char   *kname,     // 要读取的文件
         int    *nerr,      // 错误码
         int    kname_s     // 文件名长度
)
```

writebbf 与之类似，不再列出。

8.2.8 getbbv

获取一个黑板变量的值。

```
void
getbbv( char    *kname,     // 黑板变量名
        char    *kvalue,    // 黑板变量的值
        int     *nerr,      // 错误码
        int     kname_s,    // 变量名长度
        int     kvalue_s    // 变量值的长度
)
```

setbbv 的函数定义与其类似，不再列出。

8.3　调用 libsac 库

8.3.1　next2

SAC 在做 FFT 时要保证数据点数为 2^n 个，对于不足 2^n 个点的数据需要补零至 2^n 次方个点。

next2 函数定义为：

```
int next2(int num)  // 输入为num, 返回值为大于num的最小2次幂
```

8.3.2　xapiir

xapiir 用于设计 IIR 滤波器，并对数据进行滤波。这个子函数底层调用了 design 和 apply 两个子函数。

```
void
xapiir(float    *data,      // 待滤波的数据, 滤波后的数据保存在该数组中
       int       nsamps,    // 数据点数
       char     *aproto,    // 滤波器类型
                            //  - 'BU'  :   butterworth
                            //  - 'BE'  :   bessel
                            //  - 'C1'  :   chebyshev type I
                            //  - 'C2'  :   chebyshev type II
       double    trbndw,    // chebyshev滤波器的过渡带宽度
       double    a,         // chebyshev滤波器的衰减因子
       int       iord,      // 滤波器阶数
       char     *type,      // 滤波类型, 'LP','HP','BP','BR'
       double    flo,       // 低频截断频率
       double    fhi,       // 高频截断频率
       double    ts,        // 采样周期
       int       passes     // 通道数, 1或2
)
```

相关示例代码为 `filterc.c` 和 `filterf.f` 。

8.3.3　firtrn

```
void
firtrn( char    *ftype,     // 类型, 取'HILBERT'或'DERIVATIVE'
        float   *x,         // 输入数据
        int      n,         // 数据点数
        float   *buffer,    // 临时数组, 长度至少4297
        float   *y          // 输出数组
)
```

8.3.4　envelope

该子函数用于计算数据的包络函数，其底层调用了 firtrn 函数。

```
void
envelope( int    n,     // 数据点数
          float *in,    // 输入数据
          float *out    // 输出数据
)
```

相关示例代码为 `envelopec.c` 和 `envelopef.f`。

8.3.5 crscor

该子函数用于计算两个数据的互相关，此互相关在频率域中完成，相对时间域互相关而言效率更高。

```
void
crscor( float   *data1,     // 数据1
        float   *data2,     // 数据2
        int      nsamps,    // 数据点数
        int      nwin,      // 相关窗数目
        int      wlen,      // 窗内数据点数，最大值为2048
        char    *type,      // 窗类型，可以取'HAM','HAN','C','R','T'
        float   *c,         // 输出数据，长度为2*wlen-1
        int     *nfft,      // 相关序列的数据点数
        char    *err,       // 错误消息
        int      err_s      // 错误消息长度
)
```

示例代码为 `convolvec.c`、`convolvef.f`、`correlatec.c` 和 `correlatef.f`。

第 9 章　SAC I/O 自定义

这一章的内容一直没有想好怎么写，姑且先放在这里，等等再说。

SAC 自身已经提供了一系列用于读写 SAC 文件的子函数，但是函数功能过于单一。比如读函数只能一次性读取整个文件，无法只读取文件的一部分（即没有截窗的功能）；再比如，想要获取多个头段变量的值，必须多次调用相应的子函数。

在理解了 SAC 的内部结构之后，可以完全自定义 SAC I/O 函数库。

Prof. Lupei Zhu 实现了一个比较方便的 SAC I/O 函数库，可以直接使用或者作为参考。相关地址为：http://www.eas.slu.edu/People/LZhu/downloads/pssac.tar。

另一方面，我自己正在计划重写 SAC I/O 函数库以及相应的一些辅助工具。重写 SAC I/O 函数库的原因在于 Prof. Lupei Zhu 的 I/O 库中存在一些潜在的 Bug 以及考虑不周之处，并且现有的诸如 saclst 等辅助工具功能尚有欠缺。

项目主页位于：https://github.com/seisman/sac_tools。

第 10 章　SAC 相关工具

10.1　字节序转换

关于什么是字节序，可以参考维基百科相关条目。

SAC 提供了三个工具用于处理可能存在的字节序转换问题，sacswap 用于转换 SAC 文件格式，sgfswap 用于转换 SGF 文件，bbfswap 用于转换黑板变量文件。

10.2　sgftops

SGF 格式是 SAC 自定义的图像文件格式，转换到常见的其他图像格式，需要使用转换工具 sgftops。

sgftops 可以将 SGF 格式的文件转换为 PS 格式。其用法如下：

```
$ sgftops
Usage: sgftops sgf_file ps_file [line_width scale_id]
    sgf_file   :  SGF文件名
    ps_file    :  PS文件名
    line_width :  图像线宽，可以取1,1.5,2等等
    scale_id   :    - i : landscape模式加上文件id
                    - s : 对图像进行平移、旋转、缩放
                    - si: landscape模式+文件id+平移+旋转+缩放
```

示例如下：

```
$ sgftops foo.sgf foo.ps 2 si
[seisman@saturn sac]$ sgftops f001.sgf f001.ps 2 si
First translates (x and y), then rotates, then scales:
   [Default] landscape: 8 0 90 1  to prompts
   Sample portrait:  0.5 0.5 0 0.75

x translation : 0.5
y translation : 0.5
rotation angle: 0
scale........ : 0.75
```

sgftoeps 和 sgftox 通过调用 sgftops，将 sgf 文件转换为 eps 文件或直接显示在图形窗口中，这二者均依赖于 ghostscript，不再多说。

10.3 sac-config

sac-config 是 SAC 提供的一个简单的配置脚本，用于返回编译、链接 SAC 函数库时所需要的一些信息。

其用法如下：

```
Usage: sac-config [-clvp] [--prefix[=DIR]] [--version] [--libs] [--cflags]
    libs
       Display information regarding SAC, specifically used
         during the compilation of programs using the libraries
         of SAC: sacio and sac

          --cflags       Output Compilation Flags
          -c
          --libs         Output Sac Libraries
          -l
          --prefix=path  Set alternative Prefix to path for SAC
          --prefix       Display Current SAC Prefix Path
          -p
          --version      Display SAC version information
          -v
```

10.4 saclst

saclst 是很常用的一个 SAC 工具，用于列出头段变量的值，其语法很简单：

```
$ saclst header_lists f file_lists
```

其中 `header_lists` 为要查看的头段变量名列表；`f` 为关键字，表明接下来的所有参数都是 SAC 文件；`file_lists` 为 SAC 文件列表。需要注意的是，头段变量名是不区分大小写的，除了头段变量 F 以外。大写的 F 被当作头段变量名，小写的 f 被作为关键字。[1]

查看单个文件的单个头段：

```
$ saclst npts f seis.SAC
seis.SAC            1000
```

查看多个文件的多个头段：

```
$ saclst stla stlo evla evlo gcar f N.*.U
N.AAKH.U      36.3726      137.92      -5.514     151.161     43.4752
N.ABNH.U      34.6326     137.231      -5.514     151.161     42.0392
N.AC2H.U      35.4786     137.735      -5.514     151.161     42.6857
N.AGMH.U       35.787     137.717      -5.514     151.161     42.9798
N.AGWH.U      43.0842      140.82      -5.514     151.161     49.2714
N.AHIH.U      38.2799     139.549      -5.514     151.161     44.8874
```

在 Bash 脚本中将头段变量的值赋值给变量：

```
1 #!/bin/bash
2 stla=`saclst stla f seis | awk '{print $2}'`
3 stlo=`saclst stlo f seis | awk '{print $2}'`
4 echo $stla $stlo
```

[1] 这是设计不合理的地方。

在 Perl 脚本中将头段变量的值赋值给变量：

```perl
#!/usr/bin/env perl
use strict;
use warnings;

my ($fname, $stla, $stlo) = split /\s+/, `saclst stla stlo f seis`;
print "$stla $stlo \n";
```

10.5　pssac

pssac 是 Prof. Lupei Zhu 写的一个 C 程序，其利用了 GMT 强大的绘图库将 SAC 波形数据绘制到 PS 文件中。得益于 GMT 强大的绘图功能，pssac 可以制作出精致的图像，以满足出版的要求。

pssac 的用法相对较为复杂，暂且不在本文档中细说，仅列出相关的几篇博文：

1. pssac 之安装
2. pssac 之用法
3. pssac 绘图之单个地震图
4. pssac 绘图之地震剖面图
5. pssac 绘图之 reduce velocity 对齐
6. pssac 绘图之读 stdin 读入文件

第 II 部分　SAC 命令手册

第 11 章　SAC 命令

11.1　about

概要

显示版本和版权信息

语法

```
ABOUT
```

11.2　abs

概要

对每一个数据点取其绝对值

语法

```
ABS
```

说明

此命令会对 SAC 内存块中的所有波形数据的每个数据点取绝对值。取绝对值之后，会重新计算数据的最大值、最小值和均值，并更新 SAC 头段区的相应头段变量。

错误消息

- 1301: 未读入数据文件
- 1307: 对谱文件的非法操作

头段变量

depmin、depmax、depmen

11.3　add

概要

为每个数据点加上同一个常数

语法

```
add [v1 [v2 ... vn]]
```

输入

v1 加到第一个文件的常数
v2 加到第二个文件的常数
vn 加到第 n 个文件的常数

缺省值

```
add 0.0
```

说明

此命令给内存块中数据文件的每个数据点都加上同一个常数。

对于每个数据文件，这个常数可以是相同的也可以是不同的。若内存块中数据文件的个数比命令中常数的个数要多，则余下的所有数据文件将都加上命令的最后一个常数值；若内存块中数据文件的个数比给出的常数个数少，则多余的常数被忽略。

例子

为了给文件 f1 的每个数据点加上常数 5.1，f2 和 f3 的每个数据点加上常数 6.2:

```
SAC> r f1 f2 f3          // 三个文件
SAC> add 5.1 6.2         // 两个常数
```

错误消息

- 1301: 未读入数据文件
- 1307: 对谱文件的非法操作

头段变量

depmin、depmax、depmen

11.4 addf

概要

读入一组数据文件，并与内存中的另一种数据文件相加

语法

```
ADDF [NEWHDR ON|OFF] filelist
```

输入

- NEWHDR ON|OFF：默认情况下，文件相加生成的文件会使用内存中的原文件的头段。若 NEWHDR ON，则生成的文件将使用 filelist 中的新文件的头段。
- filelist：SAC 二进制文件列表。

说明

这个命令用于将一组文件同时加上同一个文件，或者将一组文件与另一组文件相加。下面针对每种情况各给出一个例子。为了使得文件与文件之间能够相加，必须保证这些文件是等间隔的且有相同的采样周期和数据点数。可以通过使用 binoperr 命令解除对于必须拥有相同采样周期和数据点数的限制。

若内存中的文件数多于 filelsit 中文件数，则 filelist 的最后一个文件将加到余下的全部文件中。这个命令可以用于数据的迭加，实际处理数据时这样的情况会经常遇到，对于文件的限制也很容易满足。

例子

将一个文件加到其他三个文件中:

```
SAC> r FILE1 FILE2 FILE3
SAC> addf FILE4
```

将两个文件分别加到另外两个文件中:

```
SAC> r FILE1 FILE2
SAC> addf FILE3 FILE4
```

头段变量改变

depmin, depmax, depmen

错误消息

- 1301: 未读入文件
- 1803: 未读入二进制数据文件
- 1306: 对不等间隔文件的非法操作
- 1307: 对谱文件的非法操作
- 1801: 头段值不匹配（采样周期或数据点数不相等）

警告消息

- 1802: 时间重叠

相关命令

read、binoperr

11.5　apk

概要

对波形使用自动事件拾取算法（由连续信号判断是否其中是否包含地震事件）

语法

```
APK [param v [param v] ... ] [VALIDATION ON|OFF]
```

输入

- param v：给参数赋值，param 的可能取值在说明中会说到。
- VALIDATION ON：打开震相检验。
- VALIDATION OFF：关闭震相检验。

缺省值

```
apk c1 0.985 c2 3.0 c3 0.6 c4 0.03 c5 5.0 c6 0.0039 c7 100. c8 -0.1
    d5 2. d8 3. d9 1. i3 3 i4 40 i6 3 validation on
```

说明

用于自动震相拾取的算法最初来自于 USGS 的 Rex Allen 的工作。

事件的检测依据是基于信号短期滑动平均值与长期滑动平均值的比值的突变。一旦事件被检测到，这次拾取将被赋予一个可选的验证状态以试图从噪声中区分出真正的事件。一旦检测到的事件的有效性被确认，这次读取到的事件将被计算以决定事件的其他特征。目前只能给出事件持续的时间，如果需要其他比如最大振幅、周期以及衰减率之类的信息也可以加入。注意这里检测的是一次事件而非一个震相!

这个命令的大多数参数永远不需要改变，如果用户想要改进算法这些参数也可以做修改。这些参数中的大多数和参考文献中有相同的含义：

- C1 是用于滤去直流偏差的递归高通滤波器中的常数
- C2 是用于改变特征函数的振幅以及一阶微分的权重的常数
- C3 是用于计算特征函数的短期平均值的时间常数
- C4 是用于计算特征函数的长期平均值的时间常数
- C5 是用于计算参考水平阀值的常数。当信号的短期平均值大于 C5 乘以长期平均值，那么这样一个信号就是一个有效事件
- C6 是用于计算滤波后数据的滑动平均绝对值的时间常数
- C7 当特征函数的绝对值大于 C7 则认为此台站无效
- C8 是用于确定信号终止的参数。当信号的绝对值低于 C8 的时间超过 D8 秒则认为信号已经终止。目前有两种不同的算法所以 C8 有两种不同的解释。如果 C8 为正，那么终止水平为 C8 乘以事件到达之前的信号的滑动平均绝对值。这个方法对于对于背景噪声很大的台站是很有用的。如果 C8 为负，则终止水平为 C8 的绝对值。如果噪声水平比终止水平低的多，则这种方法将给出不同台站的较为一致的终止判据。
- D5 是一个事件被判定为有效所要达到的最小持续时间的秒数。
- D9 是用于初始化特征函数长期平均值的持续时间值的，单位为秒
- I3、I4 和 I6 是用于震相验证的整型常数，需要保证不被改变

头段变量改变

事件读取到的时间储存在 A(即初动到时) 中，运动的质量和方向储存在 KA 中，事件结束时间储存在 F 中

例子

```
SAC> fg seis                  //利用这个数据做个例子
SAC> lh a                     //这个数据本身是标有A的, 即初动到时
  FILE: SEISMOGR - 1
 --------------
     a = 1.046400e+01
```

```
SAC> apk v on                 //apk, 且打开震相验证
 CDV IPD0 81 329103824.49     //台站名，KA的值， 后面两个不知道是什么
SAC> lh a ka f
  FILE: SEISMOGR - 1
 --------------
      a = 1.049000e+01        //新标记的初动到时，可以p看一下效果
     ka = IPD0                //初动信息，I表示急始，P表示初动P波，
                              //D表示初动向下，0不清楚
SAC> apk v off                //apk, 关闭震相验证
 CDV -123 81 329103824.49                                          53897
SAC> lh a ka f
  FILE: SEISMOGR - 1
 --------------
     a = 1.049000e+01         //初始震相
     f = 5.389773e+06         //事件结束，这里的f好像有问题？
```

错误消息

- 1301: 未读入文件
- 1306: 对非等间隔文件的非法操作
- 1307: 对谱文件非法操作

警告消息

- 1910: 下面的文件没有找到有效的事件:

相关命令

ohpf、oapf

11.6 arraymap

概要

利用 SAC 内存中的所有文件产生一个台阵或联合台阵的分布图

语法

```
ARRAYMAP [ARRAY|COARRAY]
```

输入

- ARRAY: 根据头段变量中的偏移 X、Y 值绘制台站分布
- COARRAY: 根据各台站之间的相对坐标绘制台站分布图

缺省值

```
arraymap array
```

头段数据

下面的两个头段变量必须使用 SAC 宏文件 wrxyz 或者与之功能相似的其他函数提前设定，所有的偏移是相对于某个参考点的千米数。

- USER7: 向东的偏移 (x).
- USER8: 向北的偏移 (y).

说明

不是很清楚这个命令的作用是什么，对于每个数据来说，需要用宏文件 wrxyz 定义头段变量 user7 和 user8，然后才能利用该命令绘制出 arraymap，从命令的名字来理解，应该是绘制某个台站的台站分布图，理论上只需要台站的真实位置即可。不知这个究竟在什么场合要使用。

限制

在 bbfk 中允许的最多台站数

相关命令

wrxyz 是一个 SAC 宏文件，位于 `$SACAUX/macros` 中

11.7 axes

概要

控制注释轴的位置（注释轴表示该轴具有边框、刻度和注释）

语法

```
AXES ON|OFF|ONLY ALL|TOP|BOTTOM|RIGHT|LEFT
```

输入

- ON: 显示列表列出的注释轴，其他不变
- OFF: 不显示列表列出的注释轴，其他不变
- ONLY: 只显示列表列出的注释轴，其他的不显示
- ALL: 所有的四个注释轴
- TOP: 绘图上部的 X 注释轴
- BOTTOM: 绘图下部的 X 注释轴
- RIGHT: 绘图右部的 Y 注释轴
- LEFT: 绘图左部的 Y 注释轴

缺省值

```
axes only bottom left
```

即只有下边和左边使用注释轴

说明

坐标轴可以绘制在一张图四边的任意一或多个边，有很多命令可以控制坐标轴长什么样。坐标轴的注释间隔用 xdiv 命令设定 (即隔多长显示一个数字)，刻度标记的间距可以用 ticks 命令单独控制。

only 表示仅在后面列表中指定的边上使用注释轴，而 on 和 off 则表示仅对列表中的边打开或关闭注释轴，对其他不在列表中的边不起作用。

要获得自己想要的效果，使用 on 或者 off 时你必须要知道当前已经显示的轴有哪些，哪些是你想要打开或关闭的。这是一个有点容易弄错的问题，不如只使用 only 加上想要显示的轴更加简单一点。

例子

```
SAC> fg seis
SAC> p           //看看SAC的默认设置，左边和底部有注释
SAC> axes on t   //打开顶部注释，左边和底部注释依然保留
SAC> p           //看到的结果是只有顶部注释，没有左边和底部注释，
                 //这里和说明中强调的不一样，应该是程序的bug，
                 //将on认为是only的简写了
SAC> axes on a   //打开所有注释轴
SAC> axes off b  //仅关闭底部注释轴（off选项和说明是一致的）
SAC> axes only b //仅显示底部注释轴
```

相关命令

xdiv、ticks

11.8 bandpass

概要

对数据文件使用无限脉冲带通滤波器

语法

```
BANDPASS [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n] [PASSES n]
    [TRANBW v] [ATTEN v]
```

输入

- BUTTER: 应用一个 Butterworth 滤波器
- BESSEL: 应用一个 Bessel 滤波器
- C1: 应用一个 Chebyshev I 型滤波器
- C2: 应用一个 Chebyshev II 滤波器
- CORNERS v1 v2: 设定拐角频率分别为 v1 和 v2，即频率通带为 v1-v2
- NPOLES n: 设置极数为 N，范围: 1-10
- PASSES n: 设通道数为 N，范围: 1-2
- TRANBW v: 设置 Chebyshev 转换带宽为 v
- ATTEN v: 设置 Chebyshev 衰减因子为 v

缺省值

```
bandpass butter corner 0.1 0.4 npoles 2 passes 1 tranbw 0.3 atten 30
```

说明

在 SAC 中有一系列无限脉冲滤波器 (IIR) 可以使用。这些递归的数字滤波器是基于传统的模拟滤波器设计的：Butterworth, Bessel, Chebyshev I 型以及 Chebyshev II 型。这些模拟滤波器经过双线性变换 (一种可以保持模拟滤波器稳定性的变换方式) 转换成数字滤波器。

一般来说，多数情况下 Butterworth 滤波器是个不错的选择。因为它有一个相当尖锐的转换带以及平缓的群延迟响应。Butterworth 滤波器是默认的滤波器类型，它的 3 db 点指定在截止频率处。Bessel 滤波器对于那些需要线性相位而没有双通滤波的应用来说是最好的。它的振幅响应并不够好。SAC 的 Bessel 滤波器经过归一化因此它的 3 db 点也在指定的截止

频率处。两个 Chebyshev 滤波器可以用于适应通带与阻带之间具有较为尖锐的转变的要求。尽管他们有较好的振幅响应，但是它们的群延迟响应是 SAC 所有滤波器中最差的。

在使用这些滤波器时需要小心。首先，所有的递归滤波器都有非线性相位响应，这将导致滤波后波形的频散。对于滤波后波形的相位很重要的应用来说，SAC 提供了一个递归滤波的零相位工具。零相位滤波器可以通过正向和反向 (而不仅仅只是正向滤波) 两次滤波来实现。这个双向操作的滤波器产生一个等效振幅响应，它等于原来振幅响应的平方。它同时也产生一个非因果的滤波脉冲响应，使得信号在尖锐起始时间之前附加一个虚假的前驱信号。因此双通滤波后数据无法正确的给出起跳到时。对于信号前驱不可忽略的情况，例如读取震相起跳到时，使用双通滤波器不是一个很好的选择。其次，当滤波器的通带宽度相比折叠频率很小时，滤波器可能会出现数值不稳定。这个问题在增加极数时会更加严重。在要求使用一个非常窄的通带时，一个有效的办法是首先对数据进行采样，对采样之后的数据用一个通带宽一些的滤波器进行滤波，最后对数据进行插值回到原始采样率。当所需通带宽降到折叠频率的百分之几时很有必要使用这种策略。

一般来说，随着极数的增加滤波器将有一个从通带到阻带的尖锐的转变。然而，极数太大是要付出代价的。滤波器群延迟一般随着极数的增加而变得更宽，结果导致波形混淆。那需要 3 或 4 个极的应用应该重新考虑。

Butterworth 和 Bessel 滤波器的设计特别简单。你只需要指定截止频率和极数即可。

Chebyshev 滤波器设计起来更复杂一点，你还需要提供转换带宽以及阻带衰减因子。转换带宽是滤波器通带和阻带之间的区域的宽度，它被指定为模拟滤波器通带宽度的一部分由于双线性变换频率轴的非线性弯曲，递归数字滤波器的转换带宽可能会比设计时指定的要小。在 SAC 里，模拟滤波器的截止频率在双线性变换之后要做补偿以保证其满足设计要求。阻带边界同样也是不真实的，因此，如果明确的设置阻带边界是重要的，当你的截止频率选定后你必须对其进行补偿。

阻带衰减是通带增益与阻带增益的比值。

例子

应用一个四极 Butterworth 滤波器，拐角频率为 2 、 5 Hz.:

```
SAC> bp n 4 c 2 5
```

在此之后如果要应用一个二极双通具有相同频率的 Bessel:

```
SAC> bp n 2 be p 2
```

错误消息

- 1301: 未读入文件
- 1306: 对不等间隔文件的非法操作
- 1307: 对谱文件的非法操作
- 1002: 由于拐角频率大于 Nyquist 频率而出现的坏值

头段变量改变

depmin, depmax, depmen

11.9 bandrej

概要

应用一个无限脉冲带阻滤波器

语法

```
BANDREJ [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n] [PASSES n]
    [TRANBW v] [ATTEN v]
```

输入

- BUTTER: 应用一个 Butterworth 滤波器
- BESSEL: 应用一个 Bessel 滤波器
- C1: 应用一个 Chebyshev I 型滤波器
- C2: 应用一个 Chebyshev II 滤波器
- CORNERS v1 v2: 设定拐角频率分别为 V1 和 V2
- NPOLES n: 设置极数为 N, 范围: 1-10
- PASSES n: 设通道数为 N, 范围: 1-2
- TRANBW v: 设置 Chebyshev 转换带宽为 v
- ATTEN v: 设置 Chebyshev 衰减因子为 v

缺省值

```
bandrej butter corner 0.1 0.4 npoles 2 passes 1 tranbw 0.3 atten 30
```

说明

参见命令 bandpass 的说明

例子

应用一个四极 Butterworth 滤波器，拐角频率为 2 、 5 Hz.:

```
SAC> br n 4 c 2 5
```

在此之后如果要应用一个二极双通具有相同频率的 Bessel:

```
SAC> br n 2 be p 2
```

错误消息

- 1301: 未读入文件
- 1306: 对不等间隔文件的非法操作
- 1307: 对谱文件的非法操作
- 1002: 由于拐角频率大于 Nyquist 频率而出现的坏值

头段变量改变

depmin, depmax, depmen

有关命令

bandpass

11.10 bbfk

概要

利用 SAC 内存中的所有文件计算宽频频率 -波数谱估计

语法

```
BBFK [FILTER] [NORMALIZE] [EPS v] [MLM|PDS] [EXP n] [WAWVENUMBER v] [SIZE m n]
    [LEVELS n] [DB] [TITLE text] [WRITE [ON|OFF fname] [SSQ n] [PRINT pname]]
```

输入

- FILTER: 使用最近一次 filterdesign 命令设计的带通滤波器
- NORMALIZE: 用 Capo 方法归一化协方差矩阵，如果各信号道的振幅差别比较大，这是一个好方法
- EPS v: 调整协方差矩阵的分量值，矩阵对角线的项是 (1.0+EPS) 的整数倍。
- MLM: 在高分辨率估计中使用最大似然法
- PDS: 不采用最大似然法的功率谱密度
- EXP n: 波数谱增加的幂次
- WAVENUMBER v: 从中采样谱估计的波数目
- SIZE m n: 极坐标中等值线的尺寸：m 是方位角方向上的采样点数；n 是在波数方向上的采样点数。m、n 必须为偶数，而且其乘积最大限为 40000
- LEVELS n: 等值线间隔数
- DB: 以分贝为单位的对数坐标图形
- TITLE text: 图形标题
- WRITE ON|OFF fname: 是否计算二维等值线数据并写入磁盘 (xyz 类型的 SAC 文件)。fname 是要写入的文件名或路径名。如果没有指定文件名，则默认为 BBFK
- SSQ n: 二维图的尺寸（取沿着正方形每个边的采样数据点），最大允许值为 200。
- PRINT pname: 将结果由打印机打印

缺省值

```
bbfk eps .01 pds exp 1 wvenumber 1.0 size 90 32 levels 11 write off ssq 100
```

说明

BBFK 命令允许用户计算宽频频率 -波数谱。

头段数据

分情况决定头段的信息:

- 若参考台站设置在 KUSER1 中并且其对于所有文件是相同的，则所有文件的 USER7 和 USER8 都需要设置为偏移量
- 若所有文件台站纬度 (STLA) 以及台站经度 (STLO) 都设置了，则偏移量通过这些经纬度计算，以第一个文件作为参考台站
- 若所有文件的 USER7 和 USER8 都设置了，则它们直接作为偏移量
- 若所有文件的事件纬度 (EVLA) 以及事件经度 (EVLO) 都设置了，则他们用于计算偏移量，使用第一个台站作为参考台站

输出

polar 输出立即被绘制出（不保留），square 输出会写入到硬盘。FK 的峰值、反方位角以及波数将分别写入黑板变量 BBFK_AMP, BBFK_BAZIM 以及 BBFK_WVNBR。

错误消息

- 尺寸 m 或者 n 不是一个偶数。
- 偏移量 X、Y、Z 未设置在头段变量 USER7,8,9 中。
- 未找到 filterdesign 得到的系数数据，或者滤波器类型不是 “BP”

限制

- 台站最多允许有 100 个。
- 极性等值线的最大尺寸是 m x n = 40000.
- 二维等值线输出的最大尺寸是 i = 200.

相关命令

`arraymap`

11.11　beam

概要

利用内存中的全部数据文件计算射线束

语法

```
BEAM [BEARING v] [VELOCITY v] [REFERENCE ON|OFF| lat lon [el]]
    [OFFSET REF|USER|STATION|EVENT|CASCADE] [EC anginc survel]
    [CENTER x y z] [WRITE fname]
```

输入

- BEARING v: 方位，由北算起的度数
- VELOCITY v: 速度，单位为公里每秒
- REFERENCE lat lon [el]: 参考点，打开 REFERENCE 选项并定义参考点，这样其他文件的偏移量以此而定。lat、lon、el 分别代表纬度、经度、深度（下为正）
- REFERENCE ON|OFF: 开或关 REFERENCE 选项
- OFFSET REF: 偏移量是相对于 REFERENCE 选项设置的参考点的。这要求开启 REFERENCE 选项
- OFFSET USER: 偏移量直接从 USER7, USER8 以及 USER9 中获取，(分别代表纬度、经度以及海拔)。这就要求所有文件的 USER7 和 USER8 必须定义。如果设置了 EC 选项，则 OFFSET USER 要求 USER9 必须被设置。
- OFFSET STATION: 偏移量相对于第一个台站的位置，这要求所有文件的 STLA、STLO 必须定义
- OFFSET EVENT: 偏移量相对于第一个事件的位置，这要求所有文件的 EVLA、EVLO 必须定义
- OFFSET CASCADE: SAC 将会按照前面给出的顺序考虑决定偏移量的方法，并检查必要的数据是否具备。它将使用第一个满足要求的方法

- EC: 高程校正。anginc: 入射角，从 z 轴算起，单位为度（震源距离越远，入射角越小）；survel: 表面介质速度 (km/s)。
- CENTER : 用于计算射线束的中心台站。X 为距参考台站的东向偏移；Y 为距参考台站的北向偏移；Z 为距参考台站的向上偏移，其单位为米；
- WRITE fname: 将射线束写入磁盘

缺省值

```
beam  b 90  v 9.0 ec 33  6.0 c  0. 0. 0. w BEAM
```

说明

BEAM 不覆盖 SAC 内存中的数据，因而当变换方位和速度时这一操作可以重复执行。射线结果写入到磁盘文件中，并且每次可以写到不同的文件。这个设计考虑到了用户的需求，即比较多次使用这一命令的不同结果，以寻找最佳射线束的方位和速度。

头段数据

参见 BBFK 命令

错误消息

CENTER 参数缺失偏移量，EC 参数需要正值

相关命令

map

11.12 begindevices

概要

启动某个图像设备

语法

```
BEGINDEVICES SGF|XWINDOWS
```

输入

- SGF: SAC 图像文件
- XWINDOWS: X Window System 窗口显示系统

说明

该命令用于启动一种图像设备，该命令之后的所有绘图都会被传送到该设备中，直到再次执行 begindevices 启动其它图像设备或 enddevices 结束该图像设备为止。

SAC 默认使用 xwindows 图像设备。具体用法参考 “图像设备与图像保存” 一节的示例。

相关命令

enddevices、sgf

11.13　beginframe

概要

关闭图像设备的自动刷新功能

语法

```
BEGINFRAME [PRINT [pname]]
```

输入

- PRINT pname: 当在 BEGINFRAME 中使用 PRINT 时，SAC 将把图形打印到名为 pname 的打印机，如果 pname 没有指定则打印到默认的打印机

说明

一般情况下，在每次使用绘图命令绘制图像之时，SAC 首先会对绘图设备执行刷新操作，以清除上一次绘图命令绘制的图像，然后再显示本次命令绘制的图像，这样可以保证每次绘图命令绘制的图像不会重叠在一起。

beginframe 命令会关闭绘图设备的自动刷新功能，直到 endframe 命令恢复自动刷新功能为止。在这两个命令中间执行的所有绘图命令所产生的图像将会叠加在一起，形成组合图。

通过这两个命令，并结合 xvport 和 yvport 定义每次绘图的 viewport，可以很容易地绘制出复杂的组合图。

关于如何绘制组合图以及这几个命令的使用，可以参考 “组合图” 一节。

相关命令

endframe、xvport、yvport

11.14　beginwindow

概要

启动一个新的 X 绘图窗口

语法

```
BEGINWINDOW n
```

输入

- n: 要启用的绘图窗口的编号，目前 n 的取值为 1 到 10

缺省值

```
beginwindow 1
```

说明

现在的计算机系统大多支持多窗口，即同时在多个窗口中显示相同或不同的图像。

windows 命令可以控制每个 X 绘图窗口的位置和形状，而 beginwindow 则用于启用该绘图窗口，接下来所有的绘图命令都将显示在该绘图窗口中。若你你所选择的绘图窗口没有打开，则 beginwindow 会首先创建这个窗口。

需要注意的是，window 命令只在绘图窗口被创建之前起作用，即 window 命令是一个参数设定类命令。在多数系统上，均允许通过鼠标拖曳的方式动态改变这些窗口的大小。一般情况下，在动态改变窗口大小或比例之后，当前窗口的绘图会自动重画以适应新窗口。

需要注意的是，这个命令没有与之对应的 endwindow。

相关命令

window

11.15 benioff

概要

对数据使用一个 Benioff 滤波器

语法

```
benioff
```

说明

这个命令用于近似的模拟一类短周期地震仪的仪器响应，其在 1960 年左右被美国空军在 VELA 计划中使用。

头段变量改变

depmin, depmax, depmen

11.16 binoperr

概要

控制发生在二进制文件操作中的错误

语法

```
BINOPERR [NPTS FATAL|WARNING|IGNORE] [DELTA FATAL|WARNING|IGNORE]
```

输入

- NPTS: 改变由于数据点不相等的错误条件
- DELAT: 改变由于采样间隔不相等的错误条件
- FATAL: 是错误条件为致命性条件。一旦发生错误，控制立即返回终端，忽略剩余的全部命令。

- WARNING: 给用户发送警告消息，校正错误条件并继续执行
- IGNORE: 纠正错误条件继续执行

缺省值

```
binoperr npts fatal delta fatal
```

说明

在对文件执行二元运算（addf、divf 等）时，SAC 会检测两个文件的数据点数和采样周期是否匹配。

使用这个命令，你可以控制 SAC 在遇到不匹配时如何处理。如果你设置错误条件为致命，那么 SAC 在遇到这种错误时将停止执行当前命令，忽略接下来的所有命令，打印错误信息到终端并将控制权交还。如果设置错误条件为警告，则发送一个警告消息，尽可能纠正错误并继续执行。如果设置错误条件为忽略，则 SAC 会纠正错误并继续而不告诉你发生了什么。

若要操作的两个数据文件数据点数不匹配，SAC 会使用数据点最少的那个文件的数据点数作为参考，以保证正常操作。

若要操作的两个文件采样周期不匹配，则 SAC 会使用第一个数据文件的采样周期作为结果文件的采样周期。

例子

假定 FILe1 有 1000 个数据点，file2 有 950 个数据点::

```
SAC> binoperr npts fatal
SAC> read file1
SAC> addf file2
ERROR:  Header field mismatch: NPTS file1 file2
```

上例文件加法未执行，假设你输入:

```
SAC> binoperr npts warning
SAC> addf file2
WARNING:  Header field mismatch: NPTS file1 file2
```

文件加法对每个文件的前 50 个数据执行了加法操作。

相关命令

addf、subf、mulf、divf

11.17　border

概要

控制图形周围边界的绘制

语法

```
BORDER [ON|OFF]
```

输入

- ON: 打开边界绘图
- OFF: 关闭边界绘图

缺省值

```
border off
```

说明

参考 “图像外观” 一节。

相关命令

xvport、yvport、axes、ticks

11.18 capf

概要

关闭目前打开的字符数字型震相拾取文件

语法

```
CAPF
```

相关命令

oapf

11.19 chnhdr

概要

改变指定的头段变量值

语法

```
CHNHDR [file n1 n2 ...] field v [field v ...] [ALLT v]
```

输入

- file : 可选关键字，后跟数字列表，由于 chnhdr 只能对内存中的文件头段进行操作，因而只需要给出数字即可指定要对内存中的哪些文件进行操作。
- n1、n2: 文件号，指定对哪些文件进行操作
- field : SAC 头段变量名。注意为了保证数据内部一致性，下面的头段变量值不可用 CHNHDR 更改： NVHDR, NPTS, NWFID, NORID, 和 NEVID
- ALLT v: 在所有已定义的时间相关头段变量的值上加 v 秒，同时将参考时刻减去 v 秒。常用于调整数据的参考时间
- v : 设置 field 代表的头段变量的值为 v。变量和值的类型必须匹配。

说明

关于头段变量值 v 的说明：

- 对于有内部空格的字符串要用单引号括起来。
- 逻辑字段用 TRUE 或 FALSE，YES 或 NO 也可以接受。
- 对于相对时间字段 (B, E, O, A, F,Tn)，v 可以是相对参考时间的时间偏移量，也可以是下面的形式 `GMT v1 v2 v3 v4 v5 v6`，其中 v1,v2,v3,v4,v5,v6 是 GMT 年、儒略日、小时、分钟、秒、毫秒，如果 v1 是两位数，SAC 假设其为当前世纪，除非那个时间是未来时间，那种情况下 SAC 假定是上个世纪，最好还是用 4 位整数表示年。
- UNDEF: 关键字，使头段为未定义状态

这个命令允许你修改指定的一个或多个文件的头段变量值，在未指定文件号的情况下，则对所有内存中的文件进行操作。要改变磁盘中文件的头段你还需要使用 write 或 writehdrR 命令，SAC 会对新值做有效性检查，不过你可以使用 LISTHDR 自己检查。

头段中用 6 个变量定义了参考时刻，这是 SAC 中唯一的绝对时刻，其它时刻都被转换成相对于参考时刻的相对时间。可以使用 “ALLT v” 修改参考时刻以及相对时间。参考时间被减去了 v 秒，相对时间被加上了 v 秒，这保证了数据的绝对时刻不发生改变。为了方便，你可以通过输入绝对时刻而非相对时间来改变时间偏移变量的值。绝对时刻首先被转换为相对时间，然后再存入头段中。

例子

为了定义内存中所有文件的事件经纬度、事件名::

```
SAC> ch evla 34.3 evlo -118.5
SAC> ch kevnm 'LA goes under'
```

为了定义第二、四个文件的事件经纬度、事件名:

```
SAC> ch file 2 4 EVLA 34.3 EVLO -118.5
SAC> ch file 2 4 KEVNM 'LA goes under'
```

设定初动到时为无定义状态:

```
SAC> ch A UNDEF
```

假设你知道事件的 GMT 起始时间，你想要快速改变头段中所有的时间变量，使得发震时刻是 0 即参考时间为发震时刻，并且所有的相对时间根据这个时间去纠正相对值。

首先用 GMT 选项设置事件起始时间:

```
SAC> ch o GMT 1982 123 13 37 10 103
```

现在使用 LISTHDR 检查发震时刻 o 相对于当前参考时间的描述:

```
SAC> lh o
 o = 123.103
```

现在使用 ALLT 选项从所有的偏移时间中减去这个值，并加到参考时间上, 你同时需要改变描述参考时间类型的字段:

```
SAC> ch allt -123.103 iztype iO
```

注意这里的负号意味着从偏移时间中减去这个值。

头段变量改变

几乎所以头段变量都可以被改变。

错误消息

- 1006: 字符串变量长度太长，注意每个头段都是有字节限制的。
- 1301: 未读入数据文件。

相关命令

listhdr、write、writehdr

11.20 chpf

概述

关闭当前打开的 HYPO 震相拾取文件

语法

```
CHPF
```

说明

自动附加指令 card "10" 到被关闭的文件的结尾。

相关命令

ohpf、whpf

11.21 color

概要

控制彩色图形设备的颜色选项

语法

```
COLOR [ON|OFF|color] [INCREMENT [ON|OFF]] [SKELETON color]
    [BACKGROUND color] [LIST STANDARD|colorlist]
```

color 是下面中的一个:

```
WHITE|RED|GREEN|YELLOW|BLUE|MAGENTA|CYAN|BLACK
```

这里有些参数在缩写的情况下可能会有歧义，请谨慎使用，而且 LIST 选项必须放在命令的最后

输入

- color: 颜色表中标准颜色名或颜色的数字代码
- COLOR ON : 打开颜色选项单数不改变其他选项
- COLOR OFF: 关闭颜色选项

- COLOR color : 打开颜色选项并将数据设置为颜色 color
- INCREMENT ON: 每个数据文件绘出后，根据 colorlist 的顺序改变颜色
- INCREMENT OFF: 不改变数据颜色
- SKELETON color: 按照标准颜色名或颜色号修改边框颜色
- BACKGROUND color: 修改背景色为 color，暂时不可用
- LIST colorlist: 改变颜色列表，将数据颜色设置为列表中第一个颜色，并打开颜色开关
- LIST STANDARD: 将颜色列表设为标准列表，将数据颜色设置为列表中第一个颜色，并打开颜色开关

缺省值

```
color black increment off skeleton black background white
    list standard
```

说明

该命令控制设备的颜色属性，数据颜色是用于绘制这个数据文件的颜色。当一个数据文件绘制完毕后，数据颜色可以根据颜色列表自动改变。skeleton 颜色是用于绘制注释轴、标题、网格、框架的颜色。背景色是空框架在未绘制任何图形之前的颜色。

多数情况下你会选择标准颜色名，比如 red，这是与图形设备无关的。然而有时候你可能想选择一个非标准颜色，比如 aquamarine，这个可以将颜色表装入图形设备来实现。

这个表将特定的颜色、亮度、对比度等与一个数字联系起来，然后你就可以通过设定对应的整数值选择 aquamarine 作为你的绘图的一个部分的颜色，这个需要点工作量，可是如果你喜欢，这就值得。

如果你正在同一张图上绘制多个数据文件，通过 INCRMENT 选项可以使得不同数据有不同的颜色。标准颜色表顺序如下:

```
RED, GREEN, BLUE, YELLOW, CYAN, MAGENTA, BLACK
```

例子

为了使数据颜色从红色开始不断变换:

```
SAC> color red increment
```

为了设置数据颜色为红色，背景白色，蓝色边框:

```
SAC> color red background white skeleton blue
```

为了设置一个数据颜色不断变换，颜色列表为 red,white,blue，背景色为 aquamarine(!!!):

```
SAC> color red increment backgroud 47 list red white blue
```

上面的例子假设 aquamarine 是颜色表的 47 号。

11.22　comcor

概要

控制 SAC 的命令纠正选项

语法

```
COMCOR [ON|OFF]
```

缺省值

```
comcor off
```

说明

SAC 会检查你输入的每一个命令的格式和内容。当 SAC 发现一个错误时，其会发送一个错误信息到终端告诉错误是什么、在哪里出现了错误。如果命令纠正选项开启，SAC 将允许你纠正这个错误然后 SAC 自动重新执行它。如果纠正选项关了，SAC 只是打印错误消息，将控制权返回给你。

11.23 contour

概要

利用内存中的数据绘制等值线图

语法

```
CONTOUR [ASPECT ON|OFF]
```

输入

- ASPECT ON: 打开视图比开关。当这个开关打开时，等值线图的视口将会调整保持数据中 y 与 x 的比值
- ASPECT OFF: 关闭视图比开关，这时将使用整个视口。

缺省值

```
contour  aspect  off
```

说明

这个命令用于绘制二维数组数据的等值线图，包括 SPECTROGRAM 命令的输出。这个文件操作的 SAC 文件必须 “XYZ” 类型的（SAC 头段中 IFTYPE 为 “IXYZ”）。有些命令可以控制数据显示的方式：ZLEVELS 控制等值线的数目以及间隔，ZLINES 控制线型，ZLABELS 控制等值线标签，ZTICKS 控制方向标记，ZCOLORS 控制线条颜色。根据 contour 选项的不同，有两种不同的绘制等值线算法。一种快速扫描方法用于既不选择实线型也没有时标和标识的情况。另一种慢一点的方法，在绘图之前要组合全部的线段。你可以使用快速扫描方法粗看你的数据，然后选择其他选项绘制最终图形。

例子

下面的例子中，读入了 XYZ 文件 contourdata，从头段中找出 Z 数据的范围。选择等值线范围为 700km 到 1150km，增量为 25km。选择包括四种线型的线型表，其中第一个为实线。这个列表将每四条等值线重复一次。然后给等值线图起了个名字，最后绘制出来:

```
SAC> r ./contourdata
SAC> lh iftype depmin depmax

  FILE: ./contourdata - 1
 -------------------
       IFTYPE = GENERAL XYZ (3-D) FILE
       DEPMIN = 6.977119e+02
       DEPMAX = 1.154419e+03
SAC> zlevels range 700 1150 increment 25
SAC> zlines list 1 2 3 4
SAC> title 'Katmai topography from survey data [inc = 25 km]'
SAC> contour
```

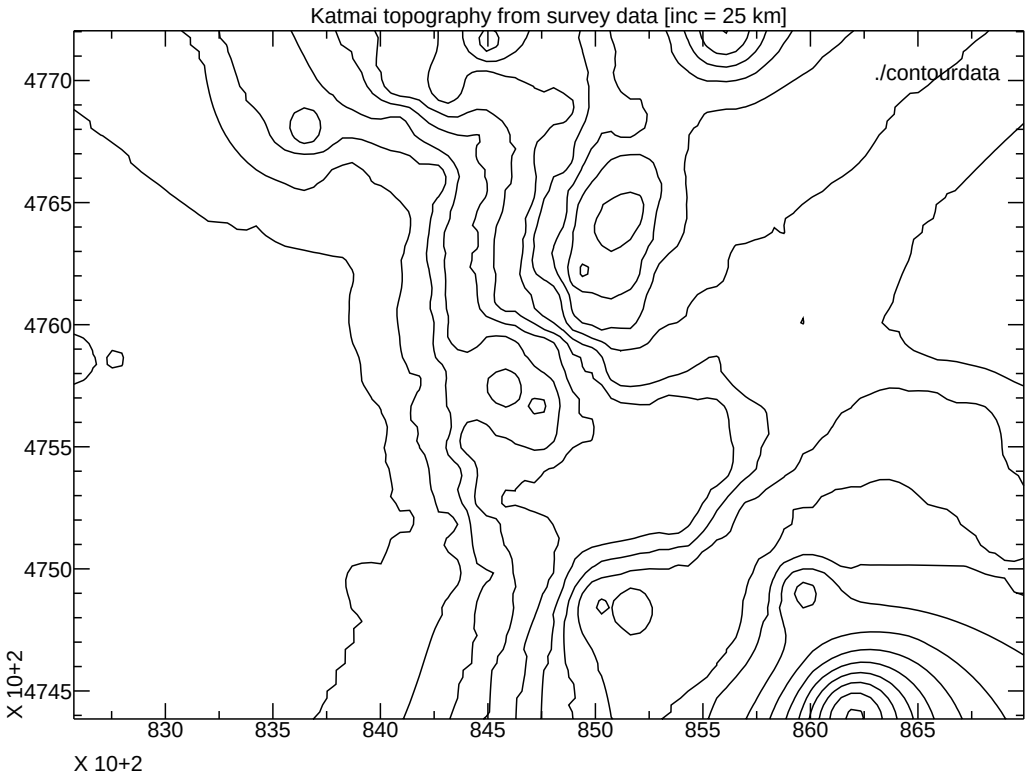

图 11.1: contour 绘制等值线 I

下面的例子中，使用同样的文件，但是显示选项不同。每四条等值线有一个整数标签。每条等值线之间都有一个指向向下的箭头。所有等值线为实线型：

```
SAC> r ./contourdata
SAC> zlevels range 700 1150 increment 25
SAC> zlabels on list int off off off
SAC> zticks on direction down
SAC> zlines list 1
SAC> title 'Katmai topography from survey data [labels and ticks]'
SAC> contour
```

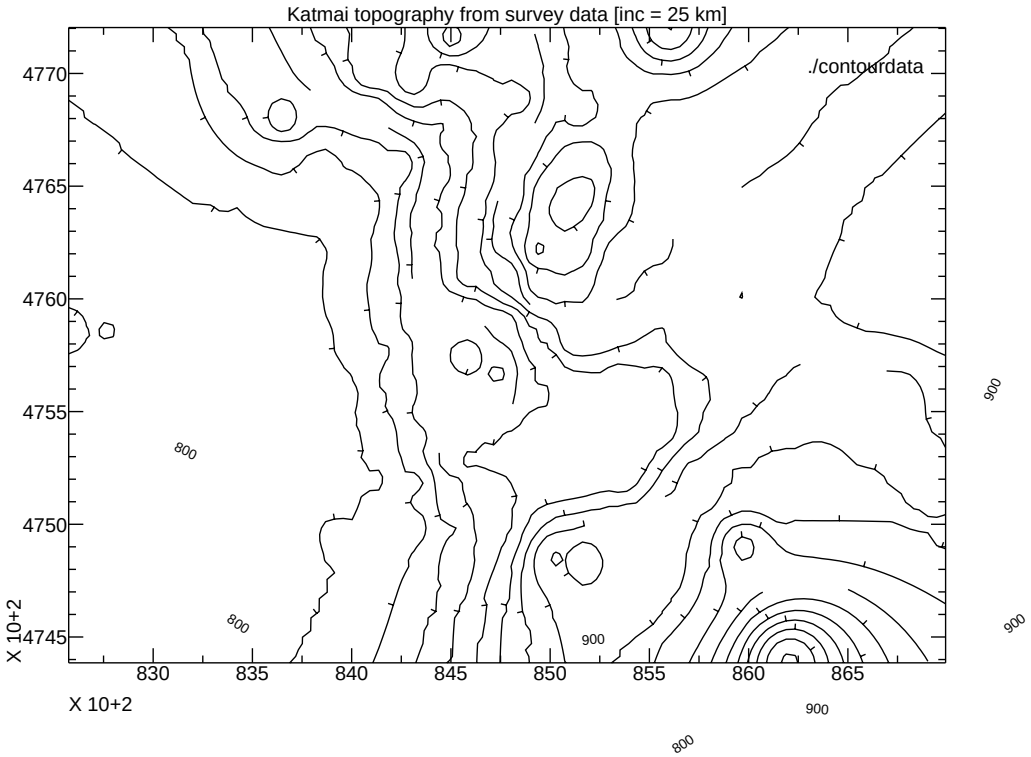

图 11.2: contour 绘制等值线图 II

头段变量改变

要求: iftype (为 “IXYZ”), nxsize, nysize

使用: xminimum, xmaximum, yminimum, ymaximum

相关命令

zcolors、zlabels、zlevels、zlines、zticks、spectrogram

11.24 convert

概要

实现数据文件格式的转换

语法

```
CONVERT [FROM] [format] infile [TO [format] outfile]|[OVER [format]]
```

其中 format 可以为 SAC|ALPHA

输入

- infile: 输入文件名
- outfile: 输出文件名
- OVER: 输出覆盖输入文件
- SAC: SAC 格式二进制文件
- ALPHA: 与 SAC 二进制文件等效的字母数字型文件

缺省值

```
convert from sac INFILE over sac
```

说明

这个命令将单个文件从一种格式转换为另一种格式。这个命令将会逐渐被 READ 和 WRITE 取代

11.25　convolve

概要

计算主信号与其自己以及一个或多个其他的卷积

语法

```
CONVOlVE [MASTER name|n] [NUMBER n] [LENGTH ON|OFF|v] [TYPE type]
```

其中 type 可以取

```
RECTANGLE|HAMMING|HANNING|COSINE|TRIANGLE
```

输入

- MASTER name|n: 通过文件名或文件号指定某文件为主文件，其他文件将对这个文件卷积
- NUMBER n: 设置要使用的卷积窗的数目
- LENGTH ON: 打开固定窗长选项开关
- LENGTH OFF: 关闭固定窗长开关
- LENGTH v: 打开固定窗长选项开关，并将窗长度设置为 v 秒
- TYPE RECTANGLE: 对每个窗应用一个矩形函数，这等价于不对窗加上函数
- TYPE HAMMING|HANNING|COSINE|TRIANGle: 对每个窗应用 xx 函数

缺省值

```
convolve master 1 number 1 length off type rectangle
```

说明

这个卷积命令允许用于将一个主信号对自己卷积以及和其他信号做卷积。这个命令按照如下公式计算卷积:

$$CV(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

这里没有 1/N 的归一化。其非常类似于如下定义的互相关:

$$CC(t) = \int_{-\infty}^{\infty} f(\tau)g(t+\tau)d\tau$$

其语法和输出都与 CORRELATE 命令很相似，除了互相关波形被卷积波形取代之外。

头段变量改变

depmin, depmax, depmen

11.26 copyhdr

概要

从内存中一个文件 copy 头段变量给其他所有文件

语法

```
COPYHDR [FROM name|n] hdrlist
```

输入

- FROM name: 从内存中文件名为 name 的文件中 copy 头段列表
- FROM n : 从内存中第 n 个文件中 copy 头段列表
- hdrlist: 用空格分割的要 copy 的头段变量

缺省值

```
copyhdr from 1
```

说明

这个命令允许你从内存中的一个文件 copy 任意头段变量值到内存中其他所有文件。你可以择从哪个文件 copy 头段。其目的是为了使数据有相同的头段使得数据易于处理。

例子

假设你使用 PPK 在文件 FILE1 中获得了多个时间标记，并将其储存到头段变量 T3 和 T4 中，为了将这些标记 copy 到 FILE2 和 FILE3 中:

```
SAC> r FILE1
SAC> ppk
SAC> ... use cursor to mark times T3 and T4.
SAC> r more FILE2 FILE3
SAC> copyhdr from 1 T3 T4
```

假设你读取了很多文件，你想要 copy 文件 ABC 中的头段变量 evla 和 evlo 到其他所有文件中去，这时使用文件名而非数字会更简单:

```
SAC> copyhdr from abc stla stlo
```

头段变量改变

几乎所有头段变量都可以改变

11.27 correlate

概要

计算自相关和互相关函数

语法

```
CORRELATE [MASTER NAME|N] [NUMBER N] [LENGTH ON|OFF|V] [TYPE TYPE]
```

其中 type 可以取

```
RECTANGLE|HAMMING|HANNING|COSINE|TRIANGLE
```

输入

- MASTER name|n : 通过文件名或文件号指定某文件为主文件，其他文件将对这个文件相关
- NUMBER n : 设置要使用的卷积窗的数目
- LENGTH ON : 打开固定窗长选项开关
- LENGTH OFF : 关闭固定窗长开关
- LENGTH v : 打开固定窗长选项开关，并将窗长度设置为 v 秒
- TYPE RECTANGLE : 对每个窗应用一个矩形函数，这等价于不对窗加上函数
- TYPE HAMMING|HANNING|COSINE|TRIANGLE : 对每个窗应用 xx 函数

缺省值

```
conrrelate master 1 number 1 length off type rectangle
```

说明

对于你申明为主信号的信号将做自相关，主信号与内存中的其他信号将计算互相关函数。这个命令的窗特性允许你计算一系列数据窗的平均相关函数，窗的数目可以选择，而且你有 5 个标准的窗函数可以选择。当这个窗特性被打开，对每一个窗将计算一个互相关函数，然后这些互相关函数将被平均，剪裁到与原始数据文件相同的长度，并且取代内存中的文件。你也可以选择窗的长度。当窗长度 (LENGTH 选项) 以及窗数目 (NUMBER 选项) 超过数据文件中的点数 (NPTS) 时，SAC 可以自动计算出窗口的混叠。缺省状态下这个窗特性是关闭的。

例子

利用内存中第三个文件作为主文件计算互相关函数:

```
SAC> correlate master 3
```

你也可以通过文件名指定主文件。假设你有数据文件，每个包含 1000 个噪声点。为了使用 10 个包含 100 个数据点的窗（即没有窗的混叠）计算平均相关函数，每个窗应用一个 HANNING 窗:

```
SAC> correlate type hanning number 10
```

为了使每个窗有 20% 的混叠，可以设置窗长度为 120 个数据点，假设采样间隔为 0.025(即每秒 40 个采样点)，这就意味着窗长为 3 秒:

```
SAC> correlate type hanning number 10 length 3.0
```

头段变量改变

depmin, depmax, depmen

11.28 cut

概要

定义要读入文件中的哪部分

语法

```
CUT [ON|OFF|pdw|SIGNAL]
```

输入

- ON : 打开 cut 开关但不改变 pdw
- OFF : 关闭 cut 开关
- pdw : 打开 cut 选项并设置 pdw。
- SIGNAL : 即 pdw 为 A -1 F +1，代表读取文件中从到时前一秒到信号结束后一秒的数据

缺省值

```
cut off
```

说明

pdw 即 partial data window，其定义了自变量的一个开始值和一个结束值，这两个值定义了文件中要被读取的部分，其一般格式为 `ref offset ref offset`。其中 ref 为参考时刻标识，可以取 `Z|B|E|O|A|F|N|Tn`，而 offset 为相对于 ref 的时间偏移量。

如果起始或结束 offset 省略则认为其为 0，如果起始参考值省略则认为其为 Z，如果结束参考值省略则认为其值与起始参考值相同。若 cut 选项为关，则读取整个文件，cut 为开，则值读取由 pdw 定义的部分。

下面的一些助记符号用于代表自变量的确定值:

- B: 磁盘文件起始值
- E: 磁盘文件结束值
- Z: 参考时间
- N: 将 offset 解释为数据点数而非自变量。其只用于结束值。

下面的助记符号代表储存在 SAC 头段中的参考时间，常用于 cut 时间序列文件:

- O: 事件开始时间
- A: 初动到时
- F: 信号结束时间
- Tn: 用户自定义时间标记，n = 0,1...9

上面的助记符再加上可选的 offset 即定义了一个一个起始或结束值。O、A、F 以及 Tn 可以使用 CHNHDR 命令来定义。A 和 F 也可以用自动事件拾取算法 APK 拾取得到或者人工绘图拾取 (PLOTPK) 到。如果你想对于一组有不同参考时间的文件使用同样的时间窗，你必须在执行 cut 和 read 命令之前先使用 SYNCHRONIZE 命令使得所有文件具有的参考时间。SYNCHRONIZE 修改了文件的头使得所有文件具有相同的参考时间，并且调整所有相对到时，使得事件的 GMT 时间保持不变。由于 cut 命令作用在磁盘文件的头段，你

必须在 SYNCHRONIZE 命令之后、READ 命令之前，使用 WRITEHDR 将修改后的头段写入磁盘文件中，这样才能得到正确的结果。

例子

在下面的例子展示了 pdw 的可能情况，我们假定时间是自变量，单位为秒:

```
B E   磁盘文件开始到文件结束 - 与cut off相同
B 0 30   磁盘文件起始的30秒
A -10 30   初动前10秒到初动后30秒
B N 2048    磁盘文件最初的2048个点
30.2 48   相对磁盘文件0点的30.2到48秒
```

你也可以在 CUTERR 命令中使用 FILLZ 选项，在文件的开始或结尾处补 0，这种做法适用于定义一个 cut 长度大于文件长度的 cut 操作，然后使用 READ 命令读入内存。

```
SAC> r N11A.lhz
SAC> lh npts
FILE: N11A.lhz - 1
npts = 3101
SAC> cuterr fillz; cut b n 4096
SAC> r
SAC> lh npts
FILE: N11A.lhz - 1
npts = 4096
```

错误消息

- 1322: 未定义文件剪裁的起始（可能原因是头段中的参考值未定义，这个错误可以用 CUTERR 命令控制，当这个错误被关闭时，使用磁盘文件起始值作为替代）
- 1323: 未定义文件裁剪的结束值
- 1324: 起始裁剪值小于文件开始值（这个错误可以使用 CUTERR 来控制，当这个错误被关闭时，使用文件的起始值代替错误值或者在文件起始补 0）
- 1325: 结束裁剪值大于文件结束值
- 1326: 起始裁剪值大于文件结束值（错误的 cut 参数）

注意: 由于这是一个参数设定类型的命令，因而上面的错误在执行 read 命令之前是不会出现的。当然上面的一些错误也可以通过 cuterr 命令转换为警告。

限制

目前不支持非等间隔文件或谱文件的截断。该命令对 ASCII 格式的 SAC 文件无效。

相关命令

read、apk、plotpk、synchronize 、cuterr

11.29　cuterr

概要

控制由于坏的截窗参数引起的错误

语法

```
CUTERR FATAL|USEBE|FILLZ
```

输入

- FATAL : 将裁剪错误设置为致命
- USEBE : 将坏的起始和结束裁剪参数设置为文件开始和文件结束
- FILLZ : 在文件开始时间之前或文件结束时间之后填补适当数目的 0 以弥补坏的裁剪参数.

缺省值

对于信号迭加子程序默认值为 FILLZ，其他的默认值为 USEBE

说明

这个命令控制由于坏的裁剪参数引起的错误条件。其可以定义为致命错误 (FATAL)。如果裁剪参数的起始值或结束值在文件头段中未定义，则可以选择为 USEBE。如果裁剪参数有定义但是其裁剪起始值小于文件起始值或者裁剪参数结束值大于文件结束值，则可以分别用文件起始结束值代替裁剪参数，或者也可以使用 FILLZ 在文件前后补适当的 0

例子

假设文件 FILE1 起始时间为 B=25s，初动到时 A=40s, 采样率为 0.01s。

```
SAC> cut a -20 e
SAC> read file1
```

裁剪起始值为 20s，产生了一个错误条件。在 USEBE 模式下，裁剪起始值将替换为 25s(即 B)。在 FILLZ 模式下，在数据之前将将插入 500 个 0(5 秒钟，每秒 100 个点)，裁剪起始值保持为 20s。

相关命令

cut、read

11.30 cutim

概要

裁剪内存中的文件，其可以从每个文件裁剪多段

语法

```
CUTIM pdw [pdw ... ]
```

输入

- pdw : Partial Data Window，它包含了一个自变量的起始值和结束值，其定义了文件中要读取的部分。pdw 的一般形式为`ref offset ref offset`
- ref : 参考值，可以取 Z|B|E|O|A|F|Tn（n=0,1...9）中的一个，具体参见 cut
- offset : 加到参考值上的一个正数或负数

缺省值

如果起始或结束 offset 省略则认为其为 0，如果起始参考值省略则认为其为 Z，如果结束参考值省略则认为其值与起始参考值相同。

说明

CUT 命令只能要截取的数据点然后等待下一个 READ 命令，CUTIM 则在这个命令给出的时候执行截取操作，用户可以 READ 一个文件，然后输入 CUTIM 以及需要的截取区间，SAC 将不通过磁盘直接裁剪文件。CUTIM 也允许多对数据截取区间，用户可以 READ 三个文件到 SAC，然后使用有 4 个截取区间的 CUTIM 命令，最终内存中将得到 12 个文件。

** 注意 ** N 选项对于 CUTIM 不可用

例子

如果一个结束时间是用震相拾取信息标记的，并且下一个起始时间是一个相对 0 时刻的偏移量，那就要避免给出一个以 0 时刻为参考的结束时间 “T1 T2 0 1000 1500”，否则，1000 会被当作相对 T2 的偏移，SAC 将认为第二个起始时间 (1500) 缺少了一个结束时间

错误消息

- 1322: 未定义文件剪裁的起始
- 1323: 未定义文件裁剪的结束值
- 1324: 起始裁剪值小于文件开始值
- 1325: 结束裁剪值大于文件结束值
- 1326: 起始裁剪值大于文件结束值

** 注意 ** 上面的一些错误也可以通过 CUTERR 命令变成警告

限制

目前不支持截取非等间隔数据或谱文件

相关命令

read、apk、plotpk、synchronize 、cuterr

11.31 datagen

概要

产生样例波形文件并储存在内存中。

语法

```
DATAGEN [MORE] [SUB name] [filelist]
```

其中 name 可以是

```
LOCAL|REGIONAL|TELESEISMIC
```

输入

- MORE : 将新产生的样本数据放在内存中旧文件后，如果此项省略，新数据将替代旧数据。
- SUB name : 选择要读数据的子目录名, 其中子目录名可以使 local、regional、teleseismic，后接文件列表。
- name : LOCAL|REGIONAL|TELESEIS
- filelist : 样本数据文件列表，文件名在下面给出。可以使用简单文件名或通配符，参见 READ 和 WILD。

缺省值

```
datagen commit sub local cdv.z
```

说明

该命令从一系列样本地震图中读取一个或多个到内存。事实上，该命令与 READ 相同，只是此处从特殊的数据目录读取特殊文件。你可以使用通配符来读取多个文件。

LOCAL 事件

近震发生在 Livermore Valley of California。震级 ML=1.6，其被 Livermore Local Seismic Network (LLSN) 所记录，LLSN 拥有一系列垂直和三分量台站。这个数据集中包含了 9 个三分量台站的数据。数据时长 40s，每秒 100 个采样点。台站信息、事件信息、p 波及尾波到时都包含在头段中，这些文件包括：

```
    cal.z, cal.n, cal.e
    cao.z, cao.n, cao.e
    cda.z, cda.n, cda.e
    cdv.z, cdv.n, cdv.e
    cmn.z, cmn.n, cmn.e
    cps.z, cps.n, cps.e
    cva.z, cva.n, cva.e
    cvl.z, cvl.n, cvl.e
    cvy.z, cvy.n, cvy.e
```

REGIONAL 事件

区域地震发生在 Nevada，被 Digital Seismic Network (DSS) 所记录。DSS 包含了美国西部的四个宽频带三分量台站。这些台站包括：

```
    elk: Elko, NV
    lac: Landers, CA
    knb: Kanab, UT
    mnv: Mina, NV
```

采样率为每秒 40 个采样点，数据时长为 300s，从事件发震前 5s 开始，文件名为：

```
    elk.z, elk.n, elk.e
    lac.z, lac.n, lac.e
    knb.z, knb.n, knb.e
    mnv.z, mnv.n, mnv.e
```

TELESEISMIC 事件

远震事件于 September 10, 1984 发生在加州北海岸靠近 Eureka 的地方。其为中等偏大的地震 (ML 6.6, MB 6.1, MS 6.7) ，多地有感。Regional Seismic Test Network(RSTN) 包含了美国和加拿大的 5 个台站，这些台站分别为:

```
cpk: Tennessee
ntk: Northwest Territories, Canada
nyk: New York
onk: Ontario, Canada
sdk: South Dakota
```

该数据集中包含了中等周期和长周期数据，cpk 数据不可得，sdk 的长周期数据被截断。这个数据集的数据时长 1600s，长周期每秒 1 个采样点，中等周期每秒 4 个采样点。文件包括:

```
ntkl.z, ntkl.n, ntkl.e, ntkm.z, ntkm.n, ntkm.e
nykl.z, nykl.n, nykl.e, nykm.z, nykm.n, nykm.e
onkl.z, onkl.n, onkl.e, onkm.z, onkm.n, onkm.e
sdkl.z, sdkl.n, sdkl.e, sdkm.z, sdkm.n, sdkm.e
```

错误信息

- 1301: 未读入数据（未指定文件列表或列表中的文件不可读）
- 1314: 数据文件列表不得以数字开头
- 1315: 超过最大文件数

警告信息

- 0101: 打开文件
- 0108: 文件不存在

11.32 `decimate`

概要

抽取数据 (减采样)，包括一个可选的抗混淆 FIR 滤波器

语法

```
DECIMATE [n] [FILTER ON|OFF]
```

输入

- n : 设置减采样因子为 n，其范围为 2 到 7。为了得到各大的减采样因子这个命令可以执行多次.
- FILTER ON : 打开抗混淆 FIR 滤波器
- FILTER OFF : 关闭抗混淆 FIR 滤波器

缺省值

```
decimate 2 filter on
```

说明

这个命令用于对于读入内存中的数据进行减采样。一个可选的 FIR 滤波器可以用于在减采样过程中避免对数字模拟信号减采样产生的混叠效应，这些滤波器保留了相位信息。FIR 滤波器的使用有时会导致一些不期望简单的位于数据末尾的短时现象，因而结果需要通过绘图来检查。只有当高频响应的精度不重要的时候，比如绘图时，才可以关闭 FIR 滤波器

例子

为了使减采样因子为 42:

```
SAC> r FILE1
SAC> decimate 7
SAC> decimate 6
```

头段变量改变

npts, delta, e, depmin, depmax, depmen

错误消息

- 1003: 值超出允许范围（减采样因子的范围是 2 到 7）
- 1301: 未读入文件
- 1306: 对非等间距文件的非法操作
- 1307: 对谱文件的非法操作

注意：减采样因子为 7 时滤波器偶尔不稳定

11.33 deletechannel

概要

从内存中文件列表中删去一个或多个文件

语法

```
DELETECHANNEL ALL
```

或

```
DELETECHANNEL FILENAME|FILENUMBER|RANGE [FILENAME|FILENUMBER|RANGE ... ]
```

输入

- ALL : 删除内存中全部文件，用户不需要指定文件名或文件号
- filename : 要删除的内存文件列表中的文件名
- filenumber : 文件列表中指定文件的文件号，第一个文件是 1，第二个文件是 2 等等
- range : 用破折号隔开的两个文件号，例如 11-20.

例子

```
dc 3 5                          // 删除第三、五个文件
dc SO01.sz SO02.sz              // 删除这些名字的文件
dc 11-20                        // 删除11-20的全部文件
dc 3 5 11-20 SO01.sz SO02.sz    // 删除上面的全部
```

错误消息

- 5106: 文件名不在文件列表中
- 5107: 文件号不在文件列表中

相关命令

filenumber

11.34　dif

概要

对内存中的数据进行微分

语法

```
DIF [TWO|THREE|FIVE]
```

输入

- TWO : 两点差分操作
- THREE : 三点差分操作
- FIVE : 五点差分操作

缺省值

```
dif two
```

说明

两点差分算法:

$$OUT(j) = \frac{DATA(j+1) - DATA(j)}{DELTA}$$

输出的最后一个点在这个算法中没有定义，它也不是一个中心算法。SAC 通过将文件点数 (NPTS) 减去 1 并且将开始时间 (B) 增加采样间隔 (DELTA) 的一半来解决输出点数的问题。

三点差分 (中心两点) 算法:

$$OUT(j) = \frac{1}{2}\frac{DATA(j+1) - DATA(j-1)}{DELTA}$$

这个算法未定义输出点的第一个和最后一个。SAC 将 NPTS 减去 2，将 B 加上 DELTA

五点差分算法 (中心四点) 算法:

$$OUT(j) = \frac{2}{3}\frac{DATA(j+1) - DATA(j-1)}{DELTA} - \frac{1}{12}\frac{DATA(j+2) - DATA(j-2)}{DELTA}$$

算法未定义输出的前两个和后两个点。SAC 对第二个和倒数第二个两个点运用三点操作，将 NPTS 减去 2，将 B 增加 DELTA

错误消息

- 1301: 未读入文件
- 1306: 对不等间隔文件非法操作

头段变量改变

npts, b, e, depmin, depmax, depmen

11.35 div

概要

给每一个数据点除以一个常数

语法

```
DIV [v1 [v2 ... vn] ]
```

输入

- v1 : 乘到第一个文件的常数
- v2 : 乘到第二个文件的常数
- vn : 乘到第 n 个文件的常数

缺省值

```
DIV 1.0
```

说明

这个命令将给内存中的每一个数据文件的每个元素除一个常数。这个常数对于每一个数据文件来说可以是相同的也可以是不同的。如果内存中数据文件的数目比命令给出的常数要多，那么最后命令中的最后一个常数将用于剩下的所有文件。如果内存中数据文件数目比给出的常数少，则多余的常数将被忽略。

错误消息

- 1301: 未读入数据文件
- 1307: 对谱文件的非法操作

头段变量改变

depmin, depmax, depmen

11.36 divf

概要

将一组文件中的数据分别除以内存中不同文件的数据

语法

```
DIVF [NEWHDR ON|OFF] filelist
```

输入

- NEWHDR ON|OFF : 操作得到的结果文件默认从内存中的原始文件获取头段值，如果 NEWHDR ON，则结果文件的头段将从 filelist 中的新文件获取

- filelist : SAC 二进制文件列表。这个列表可以包含简单的文件名，绝对或相对路径以及通配符。详细信息参见 READ 命令。

说明

这个命令用于将一个文件除以一群文件中或者将一群文件除以另一群文件。下面针对每种情况各给出一个例子。为了使得文件与文件之间能够相除，必须保证这些文件是等间隔的而且需要有相同的采样间隔和数据点数。后面两个限制可以通过使用 BINOPERR 命令得到削弱。如果内存中的文件数多于 filelsit 中文件数，则 filelist 的最后一个文件将加到余下的全部文件中。

例子

将一个文件除以其他三个文件:

```
SAC> READ FILE1 FILE2 FILE3
SAC> DIVF FILE4
```

将两个文件除以另外两个文件:

```
SAC> READ FILE1 FILE2
SAC> DIVF FILE3 FILE4
```

头段变量改变

如果 NEWHDR OFF(默认情况) 内存中的头段大部分不会被改变。

如果 NEWHDR ON，内存中的头段将被 filelist 文件的头段取代。

下面列出可能会改变的几个头段变量：depmin, depmax, depmen

错误消息

- 1301: 未读入文件
- 1803: 未读入二进制文件
- 1306: 对不等间隔文件的非法操作
- 1307: 对谱文件的非法操作
- 1317: 文件不是 SAC 数据文件
- 1801: 头段值不匹配（采样周期或数据点数不匹配）

警告消息

- 1802: 时间重叠

相关命令

read、binoperr

11.37　divomega

概要

在频率域进行积分

说明

这个命令将谱文件中的每一个点除以它的角频率。这相当于在时间域内对数据积分谱文件可以是振幅 -相位格式也可以是实部 -虚部格式。

这个运算对于普通数据来说还是很方便的，但不适于谱范围跨越几个量级的数据。比如，假设你已经使用 DIF 命令对数据进行预白化，然后对数据进行 Fourier 变换。用这个命令在频率域积分可以去除在时间域微分的效果。

例子

下面给出上面的例子所执行的命令:

```
SAC> READ FILE1
SAC> DIF
SAC> FFT AMPH
SAC> DIVOMEGA
```

头段变量改变

depmin, depmax, depmen

相关命令

dif、fft、mulomega、writesp、readsp

11.38 echo

概要

控制输入输出回显到终端

语法

```
ECHO ON | OFF ERRORS | WARNINGS | OUTPUT | COMMANDS | MACROS | PROCESS
```

输入

- ON: 打开下面列出的 echo 选项
- OFF: 关闭下面列出的 echo 选项
- ERRORS: 命令执行过程中的错误信息
- WARNINGS: 命令执行过程中的警告信息
- OUTPUT: 命令执行过程中的输出信息
- COMMANDS: 终端输入的原始命令
- MACROS: 宏文件中的原始宏命令
- PROCESSED: 经过处理后的终端或宏文件命令，命令中的变量等都被替换成具体数值

缺省值

```
echo on errors warnings output off commands macros processed
```

说明

这个命令控制 SAC 输入输出流中哪一类要被回显到终端或屏幕。输出有三大类：错误消息、警告消息、输出消息；输入也有三大类：终端中输入的命令、宏文件中执行的命令、以及处理后的命令。处理后的命令指所有的宏参数、暂存块变量、头段变量、内嵌函数首先被计算，并代入后来的命令而形成的命令。你可以分别控制这些类的回显。当你在终端输入命令时，操作系统一般会回显每个字符，因此这个命令在交互式程序中没有太大作用，而宏命令和处理后的命令选项在调试宏文件时是很有用的。

11.39 enddevices

概要

结束某个图像设备

语法

```
BEGINDEVICES SGF|XWINDOWS
```

输入

- SGF : SAC 图形文件设备驱动
- XWINDOWS : X Window System 图像窗口系统

说明

参见命令 begindevices 的说明。

相关命令

begindevices

11.40 endframe

概述

恢复图像设备的自动刷新操作

语法

```
ENDFRAME
```

说明

参见 beginframe 命令的相关说明。

相关命令

beginframe

11.41 envelope

概要

利用 Hilbert 变换计算包络函数

语法

```
ENVELOPE
```

说明

这个命令用于对内存中的数据计算包络函数。包络函数定义为

$$Envelope(n) = \sqrt{x(n)^2 + y(n)^2}$$

其中 x(n) 是原始信号，y(n) 是它的 Hilbert 变换（参见 HILBERT）。和 HILBERT 一样，超长数据需要在处理之前进行减采样。

头段变量改变

depmin, depmax, depmen

相关命令

hilbert、decimate

11.42 erase

概要

清除图形显示区域

语法

```
ERASE
```

说明

只有 SAC 知道你在使用的图形设备的情况下这个命令才可以工作。而且这只有在你已经进行了一些绘图操作之后才可以使用。这个命令对于没有清楚屏幕键的 ADM 终端很有必要。特别是你想在发送大量文本之前清除屏幕时，这个命令在命令文件中非常有用。

11.43 evaluate

概要

计算简单算术表达式

语法

```
EVALUATE [TO TERM|name] [v] op v [op v ...]
```

其中 op 为下面中的一个：

```
ADD|SUBTRACT|MULTIPLY|DIVIDE|
POWER|SQRT|EXP|ALOG|ALOG10|
SIN|ASIN|COS|ACOS|TAN|ATAN|
EQ|NE|LE|GE|LT|GT
```

输入

- TO TERM : 结果写入用户终端
- TO name : 结果写入暂存块变量 name.
- v : 浮点数或整数，SAC 中所有的运算都是浮点运算，整数会首先转换为浮点型
- op : 算术或逻辑操作符

其他形式

- + 代替 ADD
- - 代替 SUBTRACT
- * 代替 MULTIPLY
- / 代替 DIVIDE
- ** 代替 POWER

注意要在每个操作符两边加上空格

缺省值

```
evaluate to term 1. * 1.
```

说明

这个命令允许你计算算术或逻辑表达式。算术表达式可以是包含多个操作符的复合表达式，在这种情况下表达式由左向右计算，不支持嵌套功能。逻辑表达式可以只包含一个操作符。计算结果可以写入用户终端或者指定的暂存块变量。暂存块变量可以在稍后的命令中直接使用。这在宏文件中非常有用。

例子

一个简单的例子:

```
SAC> eval 2*3
 6
SAC> eval tan 45
1.61978
```

下面将一个以度为单位的角度转换为弧度并计算其正切值:

```
SAC> eval 45*pi/180
 0.785398
SAC> eval tan 0.785398
 1
```

下面将计算的结果保存到黑板变量:

```
SAC> evaluate to temp1 45*pi/ 180
SAC> evaluate tan %temp1
 1
```

限制

单个表达式中操作数的最大数目为 10.

相关命令

getbb

11.44 exp

概要

对每个数据点做自然指数

语法

```
EXP
```

错误消息

- 1301: 未读入文件
- 1307: 对谱文件非法操作

头段变量改变

depmin, depmax, depmen

11.45 exp10

概要

对每个数据点做以 10 为底的指数

语法

```
EXP10
```

错误消息

- 1301: 未读入文件
- 1307: 对谱文件非法操作

头段变量改变

depmin, depmax, depmen

11.46 fft

概要

进行离散 Fourier 变换

语法

```
FFT [WOMEAN|WMEAN] [RLIM|AMPH]
```

输入

- WOMEAN : 变换之前去除均值
- WMEAN : 变换中保留均值
- RLIM : 输出为实部 -虚部格式
- AMPH : 输出为振幅 -相位格式

缺省值

```
fft wmean amph
```

说明

在变换进行之前，每个数据文件都要补零，使得数据点数为 2 的整数次幂。在磁盘和内存中的数据文件，包括时间序列数据和谱数据。谱数据可以是振幅 -相位格式或实部 -虚部格式。头段变量 IFTYPE 告诉你文件是什么格式的。多数命令只对一种文件类型起作用。像 FFT,IFFT,UNWRAP 这样的命令将内存中的文件从一种文件格式转换为另一种格式。这个命令产生的谱文件可以用 plotsp 绘图、或者使用 write 或 writesp 命令保存在磁盘上。如果内存中有多于余个文件，PLOT2 可以用于绘制谱文件振幅或者实部。

头段变量改变

在变换时 B, E 和 DELTA 被改成起始频率、结束频率以及采样频率。B, E, NPTS 和 DELTA 的初始值被保存在 SB, SE, NSNPTS 和 SDELTA，如果做反 Fourier 变换这些值会被送回

错误消息

- 1301: 未读入文件
- 1306: 对非等间隔数据的非法操作
- 1307: 对谱文件非法操作
- 1606: 超过 DFT 的最大允许数据点数

限制

变换最大数据点数为 $2^{24} = 16777216$

相关命令

plotsp、ifft、writesp

11.47 fileid

概要

控制文件 id 在 SAC 绘图上的显示

语法

```
FILEID [ON|OFF] [Type Default|Name|List hdrlist] Location [UR|UL|LR|LL]
    [Format Equals|Colons|Nonames]
```

输入

- FILEID ON : 打开文件 id 选项，不改变文件 id 类型或位置
- FILEID OFF : 关闭文件 id 选项
- TYPE DEFAULT : 设置文件 id 为默认类型
- TYPE NAME : 使用文件名作为文件 id
- TYPE LIST hdrlist : 定义在文件 id 中显示的头段列表
- LOCATION UR|UL|LR|LL : 文件 id 放置位置，分别表示右上角、左上角、右下角、左下角
- FORMAT EQUALS : 格式为头段名，等于号以及头段值
- FORMAT COLON : 格式为头段名，冒号，头段值
- FORMAT NONAMES : 格式只包含头段值

缺省值

```
fileid on type default location ur format nonames
```

说明

文件 id 标识了绘图的内容，默认的文件 id 包括事件名、台站名、分量、参考日期及时间。如果需要也可以使用文件名代替默认的文件 id。还可以定义一个特殊的文件 id，这个 id 最多由 10 个 SAC 头段变量构成。文件 id 的位置以及格式也可以修改。

例子

将文件名放在左上角:

```
SAC> fileid location ul type name
```

定义一个特殊的文件 id，包含台站分量、经纬度:

```
SAC> fileid type list kstcmp stla stlo
```

文件 id 为头段名后加一个冒号:

```
SAC> fileid format colon
```

11.48 filenumber

概要

控制绘图时文件号的显示

语法

```
FILENUMBER [ON|OFF]
```

缺省值

```
filenumber off
```

说明

当该命令选项为开时，文件号将出现在绘图中，这可以用来通过号码识别一个指定的波形。

11.49 filterdesign

概要

产生一个滤波器的数字和模拟特性的图形显示，包括：振幅，相位，脉冲响应和群延迟。

语法

```
FilterDesign [PRINT [pname]] [FILE [prefix]] [filteroptions] [delta]
```

其中 filteroptions 与 SAC 中其他的滤波命令相同，包括滤波器类型。delta 是数据的采样间隔。

注意：选项的顺序很重要，如果使用了 PRINT 选项，则其必须是第一个选项。如果使用了 FILE 选项，其必须放在滤波器选项之前。

输入

- PRINT pname : 将输出结果打印到打印机 pname，若未给定名字，则使用默认打印机。
- FILE prefix : 生成的三个 SAC 文件的前缀

这三个 SAC 文件分别为 prefix.spec、prefix.grd、prefix.imp。其中 prefix.spec 为该命令产生的振幅相位信息，为振幅 -相位格式谱文件。prefix.gd 为该命令产生的群延迟信息，是时间序列文件。需要注意的是，尽管这个文件是时间序列文件，但是实际上群延迟是频率的函数。用户要记住，虽然绘图时横轴单位是秒，实际的单位却是 Hz。prefix.imp 是时间序列文件，包含脉冲响应信息。

在这三个 SAC 文件中，用户自定义头段变量 USERn、KUSERn 设置如下:

- user0: 表示 pass code。1 代表 LP；2 代表 HP；3 代表 BP；4 代表 BR；
- user1: type code。1 代表 BU，2 代表 BE，3 代表 C1，4 代表 C2；
- user2: number of poles
- user3: number of passes
- user4: tranbw
- user5: attenuation
- user6: delta
- user7: first corner
- user8: second corner if present, or -12345 if not
- kuser0: pass (lowpass, highpass, bandpass, or bandrej)
- kuser1: type (Butter, Bessel, C1, or C2)

缺省值

只有 delta 参数有一个缺省值 (0.025s)。其他参数必须给出

说明

FILTERDESIGN 命令使用了一个工具程序 xapiir，它是一个递归数字滤波器包。xapiir 通过模拟滤波器原型的双线性变换实现标准递归数字滤波器的设计。这些原型滤波器由零点和极点给定，然后使用模拟谱变换，变换到高通、带通和带阻滤波器。

FILTERDESIGN 用实线显示数字滤波器响应，用虚线显示模拟滤波器响应。在彩色显示器上，数字曲线是蓝色的而模拟曲线是琥珀色的。

例子

下面的例子展示了如何使用 FILTERDESIGN 命令产生一个高通，拐角频率为 2Hz，六极、双通滤波器的数字和模拟响应曲线，数据采样间隔为 0.025s:

```
SAC> fd hp c 2 n 6 p 2 delta .025
```

相关命令

highpass、lowpass、bandpass、bandrej

11.50 fir

概要

应用一个有限脉冲响应 (FIR) 滤波器

语法

```
FIR [ REC | FFT ] file
```

输入

- FFT : 通过 FFT 变换方法应用 FIR 滤波器
- REC : 递归应用 FIR 滤波器
- file : 包含 FIR 滤波器的文件名

缺省值

```
fir fft fir
```

说明

这个命令中使用的滤波器必须首先用 DFIR 交互式滤波器设计。这个滤波器通过变换方法应用，除非你要求使用递归方法或者数据点数对于变换方法来说太大。这些滤波器都没有相位失真但在脉冲信号前会产生前驱波。

头段变量

depmin, depmax, depmen

错误消息

- 1301: 未读入文件
- 1306: 对不等间隔文件非法操作
- 1307: 对谱文件非法操作
- 1601: 文件和滤波器采样频率不匹配
- 1603: 内存不足以进行 FIR 滤波

警告消息

- 1602: 内存不足以使用 FFT 进行 FIR 滤波（自动使用递归方法）

限制

变换方法的最大数据点数是 4096。

11.51　floor

概要

对于对数数据给出一个最小值，小于这个值的任何数据将被赋予这个最小值

语法

```
FLOOR [ON|OFF|v]
```

输入

- ON : 打开 floor 选项开关但不改变其值
- OFF : 关闭 floor 选项开关
- v : 打开 floor 选项开关并改变阀值

缺省值

```
FLOOR 1.0E-10
```

说明

当坐标轴取对数坐标时，对于 x 和 y 轴，当其值小于 floor 设置的值时，则在绘图前将这些值改为 floor 设置的值。floor 使用一个小的正值，对于非正数的对数运算是不允许的。

11.52　funcgen

概要

生成一个函数并将其存在内存中

语法

```
FUNCGEN [type] [DELTA v] [NPTS n] [BEGIN v]
```

其中 type 是下面中的一个：

```
IMPULSE|STEP|BOXCAR|TRIANGLE|SINE [v1 v2]|LINE [v1 v2] |
QUADRATIC [v1 v2 v3] | CUBIC [v1 v2 v3 v4] | SEISMOGRAM | RANDOM [v1 v2] |
IMPSTRIN  [n1 n2 ... nN]
```

输入

- IMPULSE : 在中心数据点处的脉冲函数
- IMPSTRIN : 一系列在指定数据点处的脉冲函数
- STEP : 阶越函数，在数据的前半段为 0，后半段为 1
- BOXCAR : 矩形函数，前后三分之一的数据为 0，中间三分之一为 1
- TRIANGLE : 三角函数，前后四分之一为 0，在中间两个四分之一线性增加到 1，再线性减小到 0

- SINE v1 v2 : 正弦函数，v1 表示频率，单位为 Hz，v2 为以度为单位的相位角，振幅为 1，注意在相位参数中有一个 2π 因子: $Fsin = 1.0\sin(2\pi(v_1t + v_2))$
- LINE v1 v2 : 线性函数，斜率为 v1，截距为 v2
- QUADRATIC v1 v2 v3 : 二次函数 $v_1t^2 + v_2t + v_3$
- CUBIC v1 v2 v3 v4 : 三次函数 $v_1t^3 + v_2t^2 + v_3t + v_4$
- SEISMOGRAM : 地震样本。对于这个函数 DELTA, NPTS 和 BEGIN 选项忽略，这个样本有 1000 个数据点
- RANDOM v1 v2 : 随机序列 (高斯白噪声) 生成器。v1 是要生成的随机序列文件数，v2 是用于产生第一个随机数的 “种子”，这个种子值保存在 USER0 中，因而如果需要你可以稍后生成一个完全相同的随机序列。
- DELTA v : 设置采样间隔为 v，储存在头段 DELTA 中
- NPTS n : 设置函数的数据点数为 n，储存在头段 NPTS 中
- BEGIN v : 设置起始时间为 v，储存在头段 B 中

缺省值

```
funcgen impulse npts 100 delta 1.0 begin 0.
```

对于正弦函数频率和相位缺省值分别为 0.05 和 0.

一次、二次、三次函数的系数都是 1。

随机序列数为 1，种子是 12357。

说明

执行这个命令相当于读取了单个文件 (除了 RANDOM 选项会产生多个文件之外) 到内存中，产生的文件名即为函数名，内存中先前的数据都会丢失。

头段变量改变

在函数生成时内存中的头段变量也会相应生成。

相关命令

datagen

11.53 getbb

概要

获取或打印黑板变量的值

语法

```
GETBB [TO TERMinal|filename] [NAMES ON|OFF] [NEWLINE ON|OFF]
    ALL|variable [variable ...]
```

输入

- TO TERMINAL : 打印值到终端
- TO filename : 将值附加到文件后
- NAMES [ON] : 输出格式为头段变量名后加一个等于号以及其值
- NAMES OFF : 只打印头段变量的值

- NEWLINE [ON] : 在每一个头段变量后换行
- NEWLINE OFF : 咋每个值后不换行
- ALL : 打印当前定义的全部头段变量
- variable : 打印列表指定的头段变量值

缺省值

```
getbb to terminal names on newline on all
```

说明

这个命令允许你打印指定的黑板变量。变量可以由 SETBB 定义，也可以由 EVALUATE 命令对黑板变量进行基本算术操作并将结果储存在新黑板变量中。黑板变量也可以在 SAC 命令中直接引用。

命令控制打印哪些值以及打印的格式，你可以将其打印到终端或者文件中。你可以使用这些选项对一系列数据文件进行测量，将结果保存到文本文件中，然后用 READALPHA 命令将这个文件读回 SAC，绘图或者进行更多的分析。

例子

假设你已经设置了一些头段变量:

```
SAC> setbb c1 2.45 c2 4.94
```

稍后可以这样打印他们的值:

```
SAC> getbb c1 c2
 c1 = 2.45
 c2 = 4.94
```

想要在一行内只打印其值:

```
SAC> getbb names off newline off c1 c2
 2.45 4.94
```

假设你有一个宏文件叫 GETXY，其可以对单个文件进行某些分析操作，并将结果储存在两个头段变量中 X 和 Y 中。你想要对当前目录中所有垂直分量进行操作，保存每对 X 和 Y 的值，然后绘图。下面的宏文件的第一个参数是用于储存这些结果的文本文件:

```
DO FILE WILD *Z
  READ FILE
  MACRO GETXY
  GETBB TO 1 NAMES OFF NEWLINE OFF X Y
ENDDO
GETBB TO TERMINAL
READALPHA CONTENT P 1
PLOT
```

最终这个文本文件将包含成对的 x-y 数据点，每行一个，对应一个垂直分量的数据文件。为了关闭文本文件并清空缓存区，最后将输出重定向到终端的 GETBB 命令是必要的。

相关命令

setbb、evaluate

11.54 grayscale

概要

产生内存中数据的灰度图像

语法

```
GRAYSCALE [VIDEOTYPE NORMAL|REVERSED] [SCALE v] [ZOOM n] [XCROP n1 n2|ON|OFF]
    [YCROP n1 n2|ON|OFF]
```

注意：这个命令使用了未在 SAC 中发布的命令，要使用这个命令你必须安装 Utah Raster Toolkit.。

输入

- VIDEO NORMAL : 设置 video 类型为 normal。在 Normal 模式中，最小值附近的数据位黑色，最大值附近的数据为白色
- VIDEO REVERSED : 设置 video 类型为 reversed。在 Reversed 模式中，最小值附近的数据位白色，最大值附近的数据为黑色
- SCALE v : 改变数据的比例因子为 v，The data is scaled by raising it to the vth power. 小于 1 的值将平滑图像、降低峰和谷，大于 1 的值将伸展整个数据
- ZOOM n : Image is increased to n times its normal size by pixel replication.
- XCROP n1 n2 : Turn x cropping option on and change cropping limits to n1 and n2. The limits are in terms of the image size.
- XCROP ON : Turn x cropping option on and use previously specified cropping limits.
- XCROP OFF : Turn x cropping option off. All of the data in the x direction is displayed.
- YCROP n1 n2 : Turn y cropping option on and change cropping limits to n1 and n2. The limits are in terms of the image size.
- YCROP ON : Turn y cropping option on and use previous specified cropping limits.
- YCROP OFF : Turn y cropping option off. All of the data in the y direction is displayed.

缺省值

```
GRAYSCALE VIDEOTYPE NORMAL SCALE 1.0 ZOOM 1 XCROP OFF YCROP OFF
```

说明

这个命令可以用于绘制 SPECTROGRAM 命令输出的灰度图，用这个命令显示的 SAC 数据须是”xyz” 文件。

注意：SAC 启动了一个脚本来运行图像操作和显示程序，然后再显示 SAC 的提示符。对于大型图像或较慢的机器，这中间会有个明显的延迟。

限制

最大只能显示 512*1000

头段变量改变

需要：IFTYPE, NXSIZE, NYSIZE

错误消息

SAC > getsun: Command not found. （需要 Utah Raster Toolkit 提供一些工具程序）

相关命令

spectrogram

11.55　grid

概要

控制绘图时的网格线

语法

```
GRID [ON|OFF|SOLID|DOTTED]
```

输入

- ON : 打开网格绘图但不改变网格类型
- OFF : 关闭网格绘图
- SOLID : 打开网格绘图并使用实网格线
- DOTTED : 打开网格绘图并使用虚线绘制网格线

缺省值

```
GRID OFF
```

说明

这个命令控制 X 和 Y 两个坐标轴，XGRID 和 YGRID 分别控制单个坐标轴的网格类型.

相关命令

xgrid、ygrid

11.56　gtext

概要

控制绘图中文本质量以及字体

语法

```
GTEXT [Software|Hardware] [Font n] [Size Tiny|Small|Medium|Large]
```

输入

- SOFTWARE: 绘图中使用软件文本
- HARDWARE: 绘图中使用硬件文本
- FONT n: 设置软件文本字体为 n,n 取值为 1 到 8
- SIZE size: 改变缺省文本大小，具体请参考 TSIZE

缺省值

```
gtext software font 1 size small
```

说明

软件文本使用了图形库的文本显示功能，字符作为小线段存储，因而具有任意的大小以及旋转至任意的角度，使用软件文本将在不同图形设备上产生相同的结果，但是其速度会慢于硬件文本，尤其对于终端来说。目前有 8 种可用的软件字体：simplex block (font 1), simplex italics (2), duplex block(3), duplex italics (4), complex block (5), complex italics(6), triplex block (7), 以及 triplex italics (8).

硬件文本使用图形设备自身的文本显示功能，因而文本大小因不同的设备而不同，所以使用硬件文本会导致在不同的图形设备上看到不同的图。如果一个设备有超过一个硬件文本尺寸，那么最究竟预期值的那个将被使用。其最主要优点在于速度较快，因而当速度比质量重要时可以使用。

例子

选择 triplex 软件字体:

```
SCA> gtext software font 6
```

相关命令

tsize

11.57 hanning

概要

对每个数据文件应用一个 “hanning” 窗

语法

```
HANNING
```

说明

“hanning” 窗是一个针对每个内部数据点 j 的递推平滑算法:

$$Y(j) = 0.25Y(j-1) + 0.50Y(j) + 0.25Y(j+1)$$

每个结束点设为等于其最近的内部点。

错误消息

- 1301: 未读入数据文件
- 1306: 对非等间隔文件非法操作

头段变量改变

depmin, depmax, depmen

11.58　help

概要

显示 SAC 命令的信息以及特性

语法

```
HELP [item ... ]
```

输入

- item：命令（全名或简写）、模块、子程序、特性等等。如果 item 为空，则输入帮助文件包的介绍

说明

帮助文档包中的每一个帮助文档都会按照 item 中的顺序列出，在输入整屏之后用户可以通过上下箭头查看该 item 的其他内容。输入 “q” 可以退出当前 item 的帮助文档，进入下一 item 帮助文档。

例子

```
SAC> h                  //获得帮助文档包的介绍
SAC> h r cut bd p       //一次获取多个命令信息
屏幕出现一堆输出，输入q进入下一item的帮助文档，直至最后一项
```

错误信息

- 1103: 没有可用的帮助文档包

11.59　highpass

概要

对数据文件应用一个无限脉冲高通滤波器

语法

```
HIGHPASS [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n] [PASSES n]
    [TRANBW v] [ATTEN v]
```

输入

- BUTTER : 应用一个 Butterworth 滤波器
- BESSEL : 应用一个 Bessel 滤波器
- C1 : 应用一个 Chebyshev I 型滤波器
- C2 : 应用一个 Chebyshev II 滤波器
- CORNERS v : 设定拐角频率分别为 v
- NPOLES n : 设置极数为 N，范围: 1-10
- PASSES n : 设通道数为 N，范围: 1-2
- TRANBW v : 设置 Chebyshev 转换带宽为 v
- ATTEN v : 设置 Chebyshev 衰减因子为 v

缺省值

```
highpass butter corner 0.2 npoles 2 passes 1 tranbw 0.3 atten 30
```

说明

参见 bandpass 的相关说明。

例子

应用一个四极 Butterworth，拐角频率为 2Hz.:

```
SAC> hp n 4 c 2
```

在此之后如果要应用一个二极双通具有相同频率的 Bessel:

```
SAC> hp n 2 be p 2
```

错误消息

- 1301: 未读入文件
- 1306: 对不等间隔文件的非法操作
- 1307: 对谱文件的非法操作
- 1002: 由于拐角频率大于 Nyquist 频率而出现的坏值

头段变量

depmin, depmax, depmen

相关命令

bandpass

11.60 hilbert

概要

应用 Hilbert 变换

语法

```
HILBERT
```

说明

每个文件列表中的数据文件 y(n) 将会被它的 Hilbert 变换 x(n) 所替代。这个变换通过在时间域与一个 201 点的 FIR 滤波器卷积实现。滤波器脉冲响应是具有 HANNING 窗的理想 Hilbert 变换的脉冲响应。在频率域，这个滤波器近似为传递函数，这里的变换满足相位判据（每个频率上相移为 90 度)。

注意这个操作在直流和重叠频率附近的小区域这个操作是不精确的，如果是取频率很低的数据，比如长周期面波，信号首先要进行抽取。由于是在时间域进行变换，所以计算使用迭加 -储存算法，其对于文件长度没有限制

头段变量改变

depmin, depmax, depmen

11.61　history

概要

打印最近使用的 SAC 命令列表

语法

```
HISTORY
```

说明

命令历史模块提供了在 unix C-shell 中的历史功能的一个子集。“history” 会打印最近使用的命令的列表（最多 100 个），C-shell 中的一些特殊功能在这里也可以使用：

- !! : 重复上一命令
- !n : 重叠第 n 行的命令
- !-n : 重复最近的命令行减去 n 得到的命令
- !str : 重复最近的以字符串 str 开始的命令

例子

打印命令历史列表:

```
SAC> history
```

重复命令 1:

```
SAC> !1
```

重复最后一条命令:

```
SAC> !!
```

重复倒数第二个命令:

```
SAC> !-2
```

重复以 ps 开头的命令::

```
SAC> !ps
```

11.62　ifft

概要

作离散反 Fourier 变换

语法

```
IFFT
```

说明

数据文件必须在之前使用过 FFT 命令，其操作对象为两种格式的谱文件

头段变量改变

B, DELTA 和 NPTS 被改成起始频率，采样频率和变换的数据点数。B, DELTA 和 NPTS 的原始值保存在 SB, SDELTA,NSNPTS 中，当进行 IFFT 时这些值回到原来的位置

错误消息

- 1301: 未读入文件
- 1305: 对时间序列的非法操作
- 1606: 超过允许作 IFFT 的最大数据点数

限制

允许作 IFFT 的最大数据点数为 65536

相关命令

fft

11.63 image

概要

利用内存中的数据文件绘制彩色图

语法

```
IMAGE [COLOR|GREY] [BINARY|FULL] [PRINT [pname]]
```

输入

- COLOR|GREY : 绘制彩图或者灰度图
- BINARY|FULL : 绘图时所有正值是一个颜色，所有负值是另一种颜色，或者根据数据值不同变换颜色
- PRINT pname : 将输出图形打印到打印机

缺省值

```
image color full
```

说明

这个命令允许用户根据 SAC 三维数据绘制彩图或灰度图，三维数据一般由 spectrogram,scallop 或 bbfk 命令产生，你也可以导入自己生成的 SAC 格式三维数据。可以使用 xlim 和 ylim 查看图形的不同区域，也可以通过一位操作符修改幅度

例子

以 SAC 101.5c 自带的 contourdata 为例：

```
SAC> r contourdata
SAC> image
```

头段变量

需要: iftype (设为 “IXYZ”), nxsize, nysize

使用:xminimum, xmaximum, yminimum, ymaximum

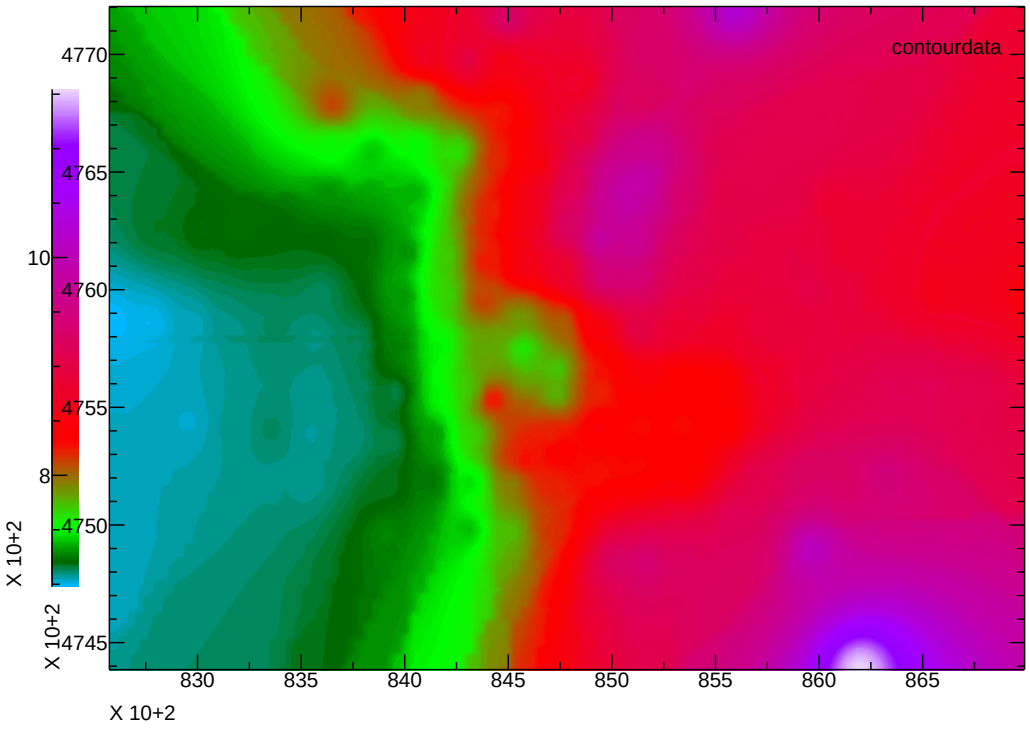

图 11.3: image 示意图

11.64 inicm

概要

重新初始化 SAC

语法

```
INICM
```

说明

这个命令用于初始化 SAC 到它该启动的状态。所有激活的图形设备中终止，全局变量初始化到初始值，内存中数据丢失。

11.65 installmacro

概要

将一个宏文件安装到 SAC 全局宏目录中

语法

```
INSTALLMACRO name [name ...]
```

输入

- name SAC 宏文件名

说明

这个命令让你将你自己的宏文件安装到全局宏目录中，这样所有人都可以使用，全局宏目录位于 `sac/aux/macros`。

相关命令

macro

11.66 int

概要

利用梯形或矩形方法对数据进行积分

语法

```
INT TRAPEZEIDAL|RECTANGULAR
```

缺省值

```
INT TRAPEZOIDAL
```

说明

这个命令使用梯形或矩形方式对数据积分，输出的第一个数据点值为 0，如果使用梯形积分，数据点数将少一个，数据不必等间隔

错误消息

- 1301: 未读入文件
- 1307: 对谱文件非法操作

头段变量改变

depmin, depmax, depmin

11.67 interpolate

概要

插值等间距或不等间距文件，获得新采样率

语法

```
INTERPOLCATE [DELTA v] [EPSILON v] [BEGIN v|OFF] [NPTS n|OFF]
```

输入

- DELTA v: 设置新采样率为 v.
- EPSILON v: 设置内插收敛因子为 v
- BEGIN v : 在 v 处开始内插，这个值作为内插生成文件的起始值
- BEGIN OFF : 在未插值数据的起始时间开始插值
- NPTS n : 强制插值后文件数据点为 n
- NPTS OFF : 由 SAC 根据起始、结束时间以及新采样率自动计算插值后的数据点数.

缺省值

```
interpolate delta 0.025 epsilon 0.0001 begin off npts off
```

说明

这个命令使用 Wiggins 内插方法将不等间距数据转换为等间距数据并重采样等间距数据得到新的采样率

例子

假设文件 filea 是不等间距文件，为了将其转换为采样率为 0.01s 的等间距文件:

```
SAC> read filea
SAC> interpolate delta 0.01
```

假设文件 fileb 是采样率为 0.025s 的等间隔文件，将其转换为采样率为 0.02s:

```
SAC> read fileb
SAC> interpolate delta 0.02
```

假设你想强制插值后文件采样率为 0.005，起始时间为 18.237，有 400 个点:

```
SAC> read filec
SAC> interpolate delta 0.005 begin 18.237 npts 400
```

警告消息

- 2008: 要求的开始时间小于文件起始时间。
- 2009: 要求的结束时间大于文件结束时间

参考文献

Wiggins, 1976, BSSA, 66, p.2077.

头段变量改变

delta, npts, e

11.68　keepam

概要

保留 SAC 内存中谱文件的振幅部分（谱文件可以是 AMPH 或 RLIM 格式）

语法

```
KEEPAM
```

说明

这个命令使得用户可以很容易的丢弃相位部分，以便对幅度部分进行一维数组的代数运算操作。如果文件是 RLIM 格式，则数据在丢弃相位之前首先转换为 AMPH 格式，最后得到的结果文件中的振幅分量数据为一般 xy 文件，因此其可以与时间域文件区分开。

11.69 khronhite

概要

对数据应用 Khronhite 滤波器

语法

```
KHRONHITE [v]
```

输入

- v : 截断频率，单位 Hz

缺省值

```
KHRONHITE 2.0
```

说明

这个低通滤波器是带有串取二级四阶 Butterworth 低通滤波器的模拟滤波器的数值逼近。其拐角频率为 0.1Hz 以加强区域范围内基谐瑞利波 (Rg) 的振幅信号。

头段变量改变

depmin, depmax, depmen

11.70 line

概要

控制绘图中线型的选择

语法

```
LINE [ON|OFF|SOLID|DOTTED|n] [INCREMENT [ON|OFF]] [LIST STANDARD|NLIST]
```

输入

- ON : 打开线型选项，不改变线型
- OFF : 关闭线型开关
- SOLID : 改变线型为实线型，并打开线型开关
- DOTTED : 改变线型为虚线型，并打开线型开关
- n : 将线型设置为 n 并打开线型开关。线型为 0 代表关闭线型开关。线型的编号随设备变化
- INCREMENT ON : 每个数据被绘出之后，按照线型表中的次序改变为另一个线型
- INCREMENT OFF : 不改变线型
- LIST nlist : 改变线型列表的目录，输入线型编号的列表
- LIST STANDARD : 设置为标准线型列表

缺省值

```
line solid increment off list standard
```

说明

这个命令控制绘图时的线型，图形的框架（轴、标题等等）通常使用实线。网格的线型用 GRID 命令控制。并非所有的图形设备都有除实线型之外的其他线型的，在那些设备上显然这个命令没有什么效果。而对于不同的设备线型号 n 也可能是不同的。

还有其他命令可以控制数据显示的其他方面。SYMBOL 命令用于将每个数据点的值用一个符号显示在图上。

COLOR 命令控制彩色图形设备的颜色选择。所有的这些属性都是独立的。如果你想的话你可以选择在每个数据点上选择带符号的蓝色虚线。线型为 0 代表关闭画线选项。在 LIST 选项和 SYMBOL 命令中可以利用线型为 0 的特性，在同一张图上将某些数据用线显示某些数据用符号显示

例子

选择依次变化的线型，从线型 1 开始:

```
SAC> line 1 increment
```

改变线型表使之包含线型 3, 5 和 1:

```
SAC> line list 3 5 1
```

使用 PLOT2 在同一个图形上绘制三个文件，第一个使用实线无符号，第二个没有线条，用三角符号，第三个无线条，用十字符号:

```
SAC> read file1 file2 file3
SAC> line list 1 0 0 increment
SAC> symbol list 0 3 7 increment
SAC> plot2
```

相关命令

symbol、color

11.71 linefit

概要

计算内存中数据的线性拟合，结果写到黑板变量中

语法

```
LINEFIT
```

说明

对数据计算最小平方拟合得到一条直线，斜率以及 y 轴截距，斜率的标准差，截距的标准差，数据的标准差以及数据之间的互相关系数将分别被写入黑板变量 SLOPE，YINT，SDSLOPE，SDYINT，SDDATA 以及 CORRCOEF。数据不必是等间隔文件。

错误消息

- 1301: 未读入数据文件
- 1307: 对谱文件的非法操作

11.72 linlin

概要

设置 X、Y 轴均为线性

语法

```
LINLIN
```

相关命令

linlog、loglog、loglin

11.73 linlog

概要

设置 X 轴为线性，Y 轴均为对数坐标

语法

```
LINLOG
```

相关命令

linlin、loglog、loglin

11.74 listhdr

概要

列出被选择的头段变量的值

语法

```
ListHdr [Default|Picks|SPecial] [FILES ALL|NONE|list] [COLUMNS 1|2]
    [INCLUSIVE ON|OFF] [hdrlist]
```

输入

- DEFAULT : 使用默认的头段列表，它列出了所有定义了的头段变量。
- PICKS : 使用 picks 列表，包含了用于定义时间拾取的头段
- SPECIAL : 使用用户定义的特别列表
- FILES ALL : 列出内存中所有文件的头段
- FILES NONE : 不列出头段，为将来的命令设置默认值
- FILES list : 列出部分文件的头段，list 是要处理的文件号

- COLUMNS 1 : 每行一列的输出格式
- COLUMNS 2 : 每行两列的输出格式
- INCLUSIVE : ON 表示包含没有定义的头段变量，OFF 则不包含
- hdrlist : 要列出的头段列表

缺省值

```
LISTHDR DEFAULT FILES ALL COLUMNS 1 INCLUSIVE OFF
```

说明

用户可以自己定义要列出的项或使用标准列表中的任一个，DEFAULT 列表包含全部头段变量。PICKS 列表包含直接或间接用于定义时间拾取的头段变量，这个列表包含 B, E,O, A, Tn, KZTIME, KZDATE。用户自己可以随时定义一个特殊列表而且可以通过 SPECIAL 选项再次使用。命令输出列表包含头段变量名，等于号以及当前值。有些头段可能未定义。SAC 在这些未定义的头段中储存一个特别的标志以标记它们，对于整数和浮点数未定义值为 -12345，字符串以及那些用于表示字符串含义的整数未定义值为 “UNDEFINED”.

错误消息

- 1301: 未读入文件

例子

获取 picks 列表，输出为两列显示:

```
SAC> lh picks column 2
```

获得第三、四个文件的默认头段列表:

```
SAC> lh files 3 4
```

列出文件开始和结束时间:

```
SAC> lh b e
```

定义一个包含台站参数的特殊列表:

```
SAC> lh KSTNM STLA STLO STEL STDP
```

稍后再次使用上面的特殊列表:

```
SAC> lh SPECIAL
```

只是设置默认两列输出:

```
SAC> lh COLUMNS 2 FILES NONE
```

11.75 load

概要

导入外部命令（在 SAC 的 linux 版本中外部命令的导入需要额外的工作）

语法

```
LOAD comname [ABBREV abbrevname]
```

输入

- comname : 从一个共享目标文件中载入的一个外部函数名
- ABBREV abbrevname : 命令的简写

说明

这个命令允许用户载入由 SAC 外部命令接口写的命令。(参见 EXTERNAL_INTERFACE 文档)。这个命令必须是一个储存在共享目标库 (.so 文件) 中。SAC 将检查所有环境变量 SACSOLIST 中的共享目标库，这个环境变量需要包含一个或多个共享目标库，每个库以空格分割。到这些共享目标库的路径必须在环境变量 LD_LIBRARY_PATH 中指定。如果 SACSOLIST 为设置，则 SAC 将使用由 LD_LIBRARY_PATH 中指定的路径在共享文件库 libsac.so 中寻找。一个叫做 libcom.so 的库随着 SAC 发布。

例子

设置你的环境变量使得 SAC 在当前目录从库文件 libbar.so 中查找一个称为 foo 的命令，并为 foo 设置别名为 myfft:

```
% export SACSOLIST = "libcom.so libbar.so"
#  Add the current directory to the search path.
% export LD_LIBRARY_PATH {$LD_LIBRARY_PATH}:.
% sac
SAC> load foo abbrev myfft* load the command
SAC> read file1.z file2.z file3.z  * input files to pass to the command
SAC> myfft real-imag* invoke command with its arguments,
* commands must parse their own args.
SAC> psp
```

如何创建一个包含你的命令的共享目标库:

```
Solaris::
cc -o libxxx.so -G extern.c foo.c bar.c
SGI::
cc -g -o libxxx.so -shared foo.c bar.c
LINUX: (gcc)::
gcc -o libxxx.so -shared extern.c foo.c bar.c sac.a
```

其中 sac.a 可以从你得到 sac 的地方获得

SAC 发布的外部命令

在 SAC 的发布版中有一个外部命令，叫做 FLIPXY。FLIPXY 将一个或多个 X-Y 数据文件作为输入，并调换 X 和 Y。这个命令在 SAC/aux/external 中的 libcom.so 中，同时还有 FLIPXY 的源代码作为参考。为了导入 FLIPXY，libcom.so 必须包含在 SACSOLIST 中

错误消息

- 1028: 外部命令不存在：这意味着 SAC 无法找到你的外部命令

这个错误产生的原因很多，一个可能是环境变量 LD_LIBRARY_PATH 不包含到达你的共享库的目录。另一个可能的原因是 SACSOLIST 不包含你的共享库的名字

11.76 loadctable

概要

允许用户在彩色绘图中选择一个新的颜色表

语法

```
LOADCTABLE n|[DIR CURRENT|name] [filelist]
```

输入

- n : 标准 SAC 颜色表对于的号，n 可以取 1-17
- DIR CURRENT : 从当前目录载入颜色表，当前目录指的是你启动 SAC 的目录
- DIR name : 从目录 name 中载入颜色表，这是一个相对或绝对目录名
- filelist : 颜色表文件列表
- file : 一个合法的颜色表文件名，其可以是简单文件名或或者一个路径名，这个路径可以是相对路径或绝对路径

说明

这个命令允许用户选择一个新的颜色表或者通过指定颜色表的文件以及路径提供他们自己的颜色表。如果未使用 DIR 选项，SAC 将首先在 SACAUX 中寻找颜色表，然后在用户的工作目录中寻找.

* 在 SACAUX 下可以的 ctables 下有 21 个文件，其中一个为 gmt.cpt，用于在 SAC 中绘制与 GMT 有关的彩色图形，另外 20 个文件为三列数据，应该就是颜色表了，前面说的 n 取 1 到 17 可能有些过时了。具体谁对应谁还不清楚，just try them!

11.77 log

概要

对每个数据点做自然对数

语法

```
LOG
```

错误消息

- 1301: 未读入文件
- 1307: 对谱文件非法操作
- 1340: 文件中数据点值超出允许的范围（对数要求值为正）

头段变量改变

depmin, depmax, depmen

11.78 log10

概要

对每个数据点做以 10 为底对数

语法

```
LOG10
```

错误消息

- 1301: 未读入文件
- 1307: 对谱文件非法操作
- 1340: 文件中数据点值超出允许的范围（对数要求值为正）

头段变量改变

depmin, depmax, depmen

11.79 loglab

概要

控制对数轴的标签

语法

```
LOGLAB [ON|OFF]
```

输入

- ON : 打开对数标签开关
- OFF : 关闭对数标签开关

缺省值

```
LOGLAB ON
```

说明

标签一般位于轴上每个 10 的对数值处，如果这个选项被打开并且标签之间有足够的空间的话在整数对数值中间将加入第二标签。

11.80 loglin

概要

设置 X 轴为对数坐标、Y 轴为线性

语法

```
LOGLIN
```

相关命令

linlog、loglog、linlin

11.81　loglog

概要

设置 X、Y 轴均为对数坐标

语法

```
LOGLOG
```

相关命令

linlog、linlin、loglin

11.82　lowpass

概要

对数据文件应用一个无限脉冲高通滤波器

语法

```
LOWPASS [BUTTER|BESSEL|C1|C2] [CORNERS v1 v2] [NPOLES n] [PASSES n]
    [TRANBW v] [ATTEN v]
```

输入

- BUTTER: 应用一个 Butterworth 滤波器
- BESSEL: 应用一个 Bessel 滤波器
- C1: 应用一个 Chebyshev I 型滤波器
- C2: 应用一个 Chebyshev II 滤波器
- CORNERS v1 v2: 设定拐角频率分别为 v1 和 v2，即频率通带为 v1-v2
- NPOLES n: 设置极数为 N，范围: 1-10
- PASSES n: 设通道数为 N，范围: 1-2
- TRANBW v: 设置 Chebyshev 转换带宽为 v
- ATTEN v: 设置 Chebyshev 衰减因子为 v

缺省值

```
lowpass butter corner 0.2 npoles 2 passes 1 tranbw 0.3 atten 30
```

说明

参见 bandpass 命令的相关说明。

例子

应用一个四极 Butterworth，拐角频率为 2Hz.:

```
SAC> lp n 4 c 2
```

在此之后如果要应用一个二极双通具有相同频率的 Bessel:

```
SAC> lp n 2 be p 2
```

错误消息

- 1301: 未读入文件
- 1306: 对不等间隔文件的非法操作
- 1307: 对谱文件的非法操作
- 1002: 由于拐角频率大于 Nyquist 频率而出现的坏值

头段变量改变

depmin, depmax, depmen

11.83 macro

概要

执行 SAC 宏文件

语法

```
MACRO name [arguments]
```

输入

- name : 要执行的 SAC 宏文件名
- arguments : 宏文件参数

说明

关于 SAC 宏文件的详细信息请参考相关文档。宏文件可以在当前目录，也可以在用 setmacro 预先指定的目录，或者在 SAC 的全局宏目录 `sac/aux/macros` 下。

相关命令

`setmacro`、`installmacro`

11.84 map

概要

利用 SAC 内存中的所有数据文件生成一个包含台站/事件符号、地形以及台站名的 GMT 地图，也可以在命令行上指定一个事件文件。每个地震事件符号可以根据震级、残差等确定其大小。这个命令会产生一个 ps 文件，并将该文件在屏幕上显示，同时产生一个绘制该图的 shell 脚本。

语法

```
MAP [MERCATOR|EQUIDISTANT|AZIMUTHAL_EQUIDISTANT|ROBINSON]
    [WEST minlon] [EAST maxlon] [NORTH maxlat] [SOUTH minlat]
    [MAGNITUDE|RESIDUAL|RMEAN_RESIDUAL] [EVEVNTFILE filename]
    [TOPOGRAPHY] [STANAMES] [MAPSCALE on|off] [PLOTSTATIONS on|off]
    [PLOTEVENTS on|off] [PLOTLEGEND on|off] [LEGENDXY x y] [FILE output-file]
```

输入

SAC 中可以使用的投影方式包括:

- MERCATOR: 投影方式为 Mercator 投影。
- EQUIDISTANT: 投影方式为等间距圆柱投影，经纬度为线性。
- ROBINSON: 投影方式为 Robinson 投影，适用于世界地图。
- LAMBERT: 适用于东西范围较大的区域。
- UTM：通用横向 Mercator（尚未实现)。

下面的选项允许用户指定地图的区域，其默认使用台站以及事件经纬度的最小最大值(如果真是如此，这样的缺省值并不合适，因为那样意味着某些台站或事件将位于地图的边界处，但是实际上地图范围给的还是不错的):

- WEST : 地图的最小经度
- EAST : 地图的最大经度
- NORTH : 地图的最大纬度
- SOUTH : 地图的最小纬度
- AUTOLIMITS: 自动决定地图的区域 [缺省值]

下面的选项允许用户向地图中添加位置和注释:

- STANames on | off : 在地图上绘制台站名 [默认为 off]
- MAPSCALE on | off : 在地图上绘制地图比例尺 [默认为 off]
- PLOTSTATIONS on | off : 绘制地震图给出的全部台站 [默认为 on]
- PLOTEVENTS on | off : 绘制 eventfile 和/或地震图给出的全部事件 [默认为 on]

下面的选项允许用户根据不同的值给出不同地震事件符号的大小。默认值是所有符号大小一样:

- MAGnitude : user0 定义地震震级，user0 越大，则事件符号越大。
- REsidual : user0 定义残差。根据 user0 的绝对值定义事件符号的大小。正值为 (+) 负值为 (-).
- RMean_residual : 与 residual 相同，除了将所有残差去除均值之外
- PLTLEGEND on | off : 绘制地震震级以及残差的图例 [默认为 on]
- LEGENDXY x y : 绘制图例的绝对位置，默认为 [1,1]。位置是相对于页面的左下角，其单位为 inch。这是一个与地震震级和残差有关的图例。
- EVENTFILE : 指定一个自由格式的 ASCII 文本文件，其包含了额外的事件数据, 文件的每一行包含单个事件的数据。每行的头两列必须包含纬度和经度（单位为度)。第三列可以包含符号大小信息（比如震级、深度、走时残差等)。
- TOPOgraphy on | off : 设置 TOPO 为开允许用户向地图中添加地形和海洋深度。这个命令读取 GMT 中 grdraster.info 的第一个地形文件，当然地形文件中必须要有该区域的数据。地形彩色图使用 $SACAUX/ctables/gmt.cpt。网格文件被写入当前目录
- FILE : 默认的输出文件名为 gmt.ps，你可以通过 FILE 选项指定文件名

可以用 SAC 的 TITLE 命令指定其标题

缺省值

```
map mercator topo off stan off file gmt.ps plotstations on plotevents on
```

例子

利用 SAC 提供的一些数据作为例子：

```
SAC> dg sub regional *.z
SAC> map stan on
Using Default Postscript Viewer
        gs -sDEVICE=x11 -q -dNOPROMPT -dTTYPAUSE
        Set an alternative through the SACPSVIEWER environment variable
        Press any key to continue
```

绘制出的地图如图11.4所示，整个地图的边界控制的还算不错，还算比较美观，三角形代表台站位置，圆形代表地震位置，大小也控制的不错。生成这个图的同时，还有一个可以用于生成该地图的 shell 脚本。

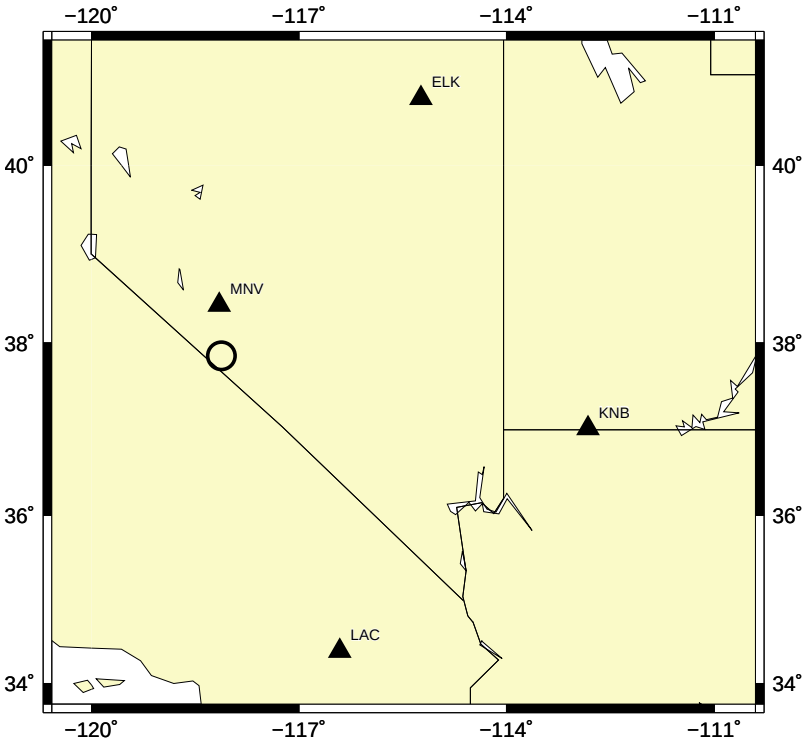

图 11.4: map 绘制地震、台站分布图

头段数据

台站纬度 (stla) 以及经度 (stlo) 必须在头段中被定义。如果事件纬度 (evla) 以及经度 (evlo) 被定义则其会被包含在地图中。如果这个命令在执行 BBFK 之后执行，MAP 将沿着反方位角方向绘制大圆弧路径。这个版本的 MAP 是基于 4.0 版本的 Generic Mapping Tools，要执行这个命令，你需要将 GMT4.0 安装在你的机器上并保证可执行文件位于路径中。

每个 MAP 命令的结果将写入当前目录下一个称为 gmt.csh 的脚本中。用户可以修改这个文件以利用更多 SAC 未利用的选项。默认单位是 inch，当然可以在脚本中修改。

在使用 pscoast 绘制海岸线时，SAC 采用了 -Dl 选项，其中 l 代表低精度的海岸线数据。用户可以在脚本中修改使用更高精度的海岸线数据。

该脚本采用的 shell 是 csh，其很容易改成 bash 脚本。

MAP 命令的结果将自动被显示。用于显示的默认程序是 gs(ghostscript)。用户可以通过

设置环境变量 SACPSVIEWER 选择其他显示工具。
在 csh 下可以这样设置：

```
setenv SACPSVIEWER "gs -sDEVICE=x11 -q -dNOPROMPT -dTTYPAUSE"
```

在 bash 下可以这样设置：

```
export SACPSVIEWER="gs -sDEVICE=x11 -q -dNOPROMPT -dTTYPAUSE"
```

11.85　markptp

概要

在测量时间窗内测量并标记最大峰峰值

语法

```
MARKPTP [LENGTH v] [TO marker]
```

输入

- LENGTH v：设置滑动窗的长度为 v 秒
- TO marker：定义头段中用于储存结果的第一个时间标记。最小值的时间记录在这个时间标记中，最大值的时间标记储存在下一个时间标记中。
- marker：Tn(n=0-9)

缺省值

```
markptp length 5.0 to t0
```

说明

这个命令测量数据在当前测量时间窗内最大峰峰值对应的时间和幅度。结果写入头段变量中。波谷对应的时间写入命令给出的时间标记中，波峰对应的时间写入下一个命令标记。峰峰值写入 USER0 中，这个结果也可以通过 OAFP 命令写入字符数字型震相拾取文件。

例子

设置测量时间窗为头段 T4 和 T5 之间，并使用默认的滑动时间窗长和标记：

```
SAC> mtw t4 t5
SAC> markptp
```

设置测量时间窗为初动之后的 30s，滑动时间窗为 3s，起始时间标记为 T7：

```
SAC> mtw a 0 30
SAC> markp l 3. to t7
```

头段变量

Tn, USER0, KTn, KUSER0

相关命令

mtw、oapf

11.86 marktimes

概要

根据一个速度集得到走时并对数据文件进行标记

语法

```
MARKTIMES [TO marker] [DISTANCE HEADER|v] [ORIGIN HEADER|v|GMT time]
    [VELOCITIES v ... ]
```

输入

- TO marker : 定义头段中用于储存结果的第一个时间标记。对下一个的速度使用下一个时间
- marker : T0|T1|T2|T3|T4|T5|T6|T7|T8|T9
- DISTANCE HEADER : 使用头段中的 DIST 代表距离用于走时计算
- DISTANCE v : 使用 v 作为走时计算中的距离
- ORIGIN HEADER : 使用头段中的参考时间 (O) 用于走时计算
- ORIGIN v : 使用相对参考时间偏移 v 秒用于走时计算
- ORIGIN GMT time : 使用 Greenwich 平均时间作为参考时间
- time : GMT 时间包含六个整数：年、儒略日、时、分、秒、毫秒
- VELOCITIES v ... : 设置用于走时计算的速度集，最多可以输入 10 个速度

缺省值

```
marktimes velocities 2. 3. 4. 5. 6.  distance header origin header to t0
```

说明

这个命令在头段中标记震相的到时，给定事件的发生时间，震中距以及速度即可计算到时。下面的方程用于简单估计走时:

$$time(j) = origin + \frac{distance}{velocity(j)}$$

结果被写入指定的时间标记中

例子

使用默认的速度集，强制距离为 340km，第一个时间标记为 T4:

```
SAC> marktimes distance 340. to t4
```

选择一个不同的速度集:

```
SAC> markt v 3.5 4.0 4.5 5.0 5.5
```

设置新的参考时间并将结果保存在 T2 中:

```
SAC> markt origin gmt 1984 231 12 43 17 237 to t2
```

头段变量改变

Tn, KTn

11.87　markvalue

概要

在数据文件中搜索并标记某个值

语法

```
MARKVALUE [GE|LE v] [TO marker]
```

输入

- GE v : 搜索并标记第一个大于或等于 v 的数据点
- LE v : 搜索并标记第一个小于或等于 v 的数据点
- TO marker : 定义头段中储存结果的时间标记
- marker : T0|T1|T2|T3|T4|T5|T6|T7|T8|T9

缺省值

```
markvalue ge 1 to t0
```

说明

这个命令在每一个数据文件中搜索，满足要求的值并将第一次出现该值的时间标记下来，如果测量时间窗已经被定义 (参见 MTW)，则只有那部分的文件才会被搜索。否则将搜索整个文件，结构写入头段中的时间标记中

例子

搜索文件中第一个值大于 3.4 的点并将结果保存在头段 T7 中:

```
SAC> markvalue ge 3.4 to t7
```

稍后在以 T4 起始的 10s 测量时间窗中使用相同的搜索:

```
SAC> mtw t4 0 10
SAC> markvalue
```

头段变量改变

Tn, KTn

相关命令

mtw

11.88　merge

概要

将一系列文件合并 (首尾相连) 内存中的数据中

语法

```
MERGE [ filelist ]
```

输入

- filelist: SAC 二进制数据文件列表

说明

在文件列表中的数据将会附加在内存中数据之后。每一对要合并的文件将被检查以保证他们拥有相同的采样间隔以及台站名。第一个文件的结束时间同样也要与第二个文件的开始时间做比较。如果在这些时间之间有间断，则产生警告并在两个文件之间放入正确数目的零再合并。如果时间之间有重叠，则给出错误信息并不再合并

例子

分别将文件 file3 和 file4 与 file1 和 file2 合并:

```
SAC> read file1 file2
SAC> merge file3 file4
```

合并一个台站在不同目录下的四个不同的事件:

```
SAC> r /data/event1/elko.z
SAC> merge /data/event2/elko.z
SAC> merge /data/event3/elko.z
SAC> merge /data/event4/elko.z
```

头段变量改变

npts, depmin, depmax, depmen, e

错误消息

- 1301: 未读入数据文件
- 1803: 未读入二进制文件
- 1307: 对谱文件非法操作
- 1306: 对不等间隔文件的非法操作
- 1801: 头段不匹配（采样间隔或台站名不同）
- 1802: 时间混叠

警告消息

- 1805: 时间间断

相关命令

read

11.89 message

概要

发送信息到用户的终端

语法

```
MESSAGE text
```

输入

- text 要发送的信息文本，若文本中有空格则必须用单引号括起来

说明

这个命令用于在 SAC 宏文件执行时发送状态及其他信息给用户。在交互式模式下没有用。

例子

发送无空格的信息:

```
SAC> message finished
 finished
```

发送带有空格的信息，你需要加上单引号或双引号:

```
SAC> message 'Job has finished.'
 Job has finished.
```

11.90　mtw

概要

决定接下来命令中所使用的测量时间窗

语法

```
MTW [ON|OFF|pdw]
```

输入

- ON : 打开测量时间窗选项但不改变窗的值
- OFF : 关闭测量时间窗选项。测量将对整个文件进行操作
- pdw : 打开测量时间窗并设置其新值。pdw 包含你想要进行测量的自变量的范围，详细信息参考 CUT 命令

缺省值

```
mtw off
```

说明

当这个选项为开时，测量只对这个窗内有效，当这个选项为关时，测量针对整个文件。这个选项目前只对 markptp 和 markvalue 命令有效

例子

pdw 的一些例子如下:

- B 0 30: 文件起始的 30s
- A -10 30: 初动到时前 10s 到初动后 30s
- T3 -1 T7: 时间标记 T3 前 1s 到 T7
- B N 2048: 文件最初的 2048 个数据点
- 30.2 48: 相对文件零时的 30.2s 到 48s

相关命令

cut、markptp、markvalue

11.91 mul

概要

给每一个数据点乘以一个常数

语法

```
MUL [v1 [v2 ... vn]]
```

输入

- v1 : 乘到第一个文件的常数
- v2 : 乘到第二个文件的常数
- vn : 乘到第 n 个文件的常数

缺省值

```
mul 1.0
```

说明

这个命令将给内存中的每一个数据文件的每个元素乘一个常数。这个常数对于每一个数据文件来说可以是相同的也可以是不同的。如果内存中数据文件的数目比命令给出的常数要多，那么最后命令中的最后一个常数将用于剩下的所有文件。如果内存中数据文件数目比给出的常数少，则多余的常数将被忽略。

错误消息

- 1301: 未读入数据文件
- 1307: 对谱文件的非法操作

头段变量改变

depmin, depmax, depmen

11.92 mulf

概要

将一组文件中的数据分别乘以内存中不同文件的数据

语法

```
MULF [NEWHDR ON|OFF] filelist
```

输入

- NEWHDR ON|OFF : 操作得到的结果文件默认从内存中的原始文件获取头段值，如果 NEWHDR ON，则结果文件的头段将从 filelist 中的新文件获取

- filelist : SAC 二进制文件列表。这个列表可以包含简单的文件名，绝对或相对路径以及通配符。详细信息参见 READ 命令。

说明

这个命令用于将一个文件或者将一群文件乘以另一群文件。下面针对每种情况各给出一个例子。为了使得文件与文件之间能够相除乘，必须保证这些文件是等间隔的而且需要有相同的采样间隔和数据点数。后面两个限制可以通过使用 BINOPERR 命令得到削弱。如果内存中的文件数多于 filelsit 中文件数，则 filelist 的最后一个文件将加到余下的全部文件中。

例子

将一个文件乘以其他三个文件:

```
SAC> read file1 file2 file3
SAC> mulf file4
```

将两个文件乘以另外两个文件:

```
SAC> read file1 file2
SAC> mulf file3 file4
```

头段变量

如果 NEWHDR OFF(默认情况) 内存中的头段大部分不会被改变。如果 NEWHDR ON，内存中的头段将被 filelist 文件的头段取代。下面列出可能会改变的几个头段变量: depmin, depmax, depmen

错误消息

- 1301: 未读入文件
- 1803: 未读入二进制文件
- 1306: 对不等间隔文件的非法操作
- 1307: 对谱文件的非法操作
- 1317: 文件不是 SAC 数据文件
- 1801: 头段值不匹配（采样周期或数据点数不相等）

警告消息

- 1802: 时间重叠 * 什么意思？如何犯错？ *

相关命令

read、binoperr

11.93　mulomega

概要

在频率域进行微分

说明

这个命令对谱文件的每个数据点乘以它的频率:

$$\Omega = 2\pi f$$

这是对时间域微分的等效模拟。谱文件可以是振幅 -相位型或者实部 -虚部型

头段变量改变

depmin, depmax, depmen

11.94 news

概要

打印关于 SAC 的新消息

语法

```
NEWS
```

说明

如果有新消息要报告，在 SAC 启动的时候会有一个头条写到终端，你可以使用 news 看到更多细节

11.95 null

概要

控制绘图过程中空值的绘制

语法

```
NULL [ON|OFF|value]
```

输入

- ON : 打开绘图时的 NULL 选项
- OFF : 关闭绘图时的 NULL 选项
- value : 设置数据中的空值为该值

缺省值

```
NULL OFF
```

说明

在有些情况下，数据中有间断，此时没有有用的值，称为空值。在多数情况下将空值设置为一个预订的值，一般来说是 0.0, -1.0, -99。通常用户不希望这些值在绘图时显示。NULL 命令允许用户定义 NULL 值并在绘图时不连接这些点，为了设置 NULL 值为 -1.0 并打开 NULL 选项:

```
SAC> null on -1.0
```

11.96　oapf

概要

打开一个字母数字型震相拾取文件

语法

```
OAPF [STANDARD|NAME] [file]
```

输入

- STANDARD : 在写震相拾取文件时使用标准文件 id。标准文件 id 包含了头段中事件名、台站名、分量方位角以及入射角。
- NAME : 使用 SAC 文件名代替标准文件 id
- file : 要打开的字符数字型震相拾取文件，如果该文件已经存在，则其将会被打开，新的震相拾取会加到文件的底部

缺省值

```
oapf standard apf
```

说明

震相拾取文件可以作为自动拾取 (APK) 以及人工拾取 (PPK) 命令产生的简单数据库。每个震相拾取被写在文件的一行上。这个文件的每个常规行包含一个文件 id、一个震相拾取 id、震相拾取的时间、拾取的幅度以及一些格式信息。这些行为 80 个字符长。文件 id 是一些标准的头段信息或者文件名。拾取到的时间是 GMT 时间或者时间偏移。这依赖于产生这些拾取的命令如 APK 或 PPK 中指定的选项。这将导致 4 个不同的格式，在第 79 列上以不同的字符标识这些行，如那些波形和峰峰值的读取数据，在 80 列以后不附加字段。一行的最大长度为 200。下面会展示不同格式的行。

错误消息

- 1903: 不能关闭前面的震相文件
- 1902: 不能打开震相文件（可能是文件名中的非法字符引起的）

相关命令

plotpk、apk、capf

11.97　ohpf

概要

打开一个 HYPO 格式的震相文件

语法

```
OHPF [file]
```

输入

- file 要打开的文件名。如果文件已经存在，则打开并将新的震相加到文件底部

缺省值

```
OHPF HPF
```

说明

SAC 产生的 HYPO 震相拾取文件可以用于程序 HYPO71 以及其他类似事件定位程序的输入。由 APK 和 PPK 得到的震相拾取信息被写入这个打开的文件中，这个文件可以使用 CHPF 关闭。打开一个新的 HYPO 文件会自动关闭前一个已经打开的文件。打开一个已经存在的 HYPO 文件的同时也会自动删除文件的最后一行，这一行原本有一个指令标记 10 作为 HYPO 文件的结束标志符，删除最后一行意味着可以在其后添加新的拾取。终止 SAC 也会自动关闭任何已经打开的拾取文件，事件分割符能够用 WHPF 命令写入震相拾取文件。

错误消息

- 1901: 无法打开 HYPO 震相拾取文件（可能是文件名只能中的非法字符）

相关命令

apk、plotpk、whpf、chpf

11.98 pause

概要

发送信息到终端并暂停

语法

```
PAUSE [MESSAGE text] [PERIOD ON|OFF|v]
```

输入

- MESSAGE text : 在暂停之前发送到终端的文本，若文本包含空格则应用引号括起来
- PERIOD ON : 打开 period 选项但不改变暂停的长度，如果这个选项为开，SAC 会暂停一段时间然后自动恢复执行
- PERIOD OFF : 关闭 period 选项开关。当这个选项关闭时，SAC 将暂停知道输入回车键
- PERIOD v : 打开 period 选项开关并改变暂停的时间长度为 v 秒

缺省值

```
PAUSE MESSAGE 'Pausing' PERIOD OFF
```

说明

这个命令使你能够暂时挂起 SAC 宏文件的执行。当这个命令被执行时，SAC 发送信息到终端，暂停，然后等到你输入回车或者暂停时间到。当你想要研究一个特定命令的输出时可以让宏文件暂停执行再继续。它在用于说明或者教学时特别有用。ECHO 命令也很有用的

相关命令

echo

11.99 pickauthor

概要

告诉 SAC 从一个用户定义的参考文件中读入作者列表（也可能是震相拾取信息），或在该命令行上输入作者列表

语法

```
PICKAUTHOR author1 [ author2 author3 ... ]
PICKAUTHOR FILE [ filename ]
PICKAUTHOR PHASE [ filename ]
```

输入

- authorlist : SAC 使用输入创建作者列表
- FILE : 如果使用了 FILE 选项，SAC 将从参考文件中读取作者列表。如果参考文件的文件名在命令行上给出，则 SAC 将读取这个指定的文件，否则 SAC 将根据上一次执行 PICKAUTHOR 读取最近输入的文件名. 如果未给出文件名，则 SAC 使用 `sac/aux/csspickprefs`
- PHASE : 这个选项和 FILE 选项相似，其另一个功能是允许 SAC 读取指定的头段变量信息

缺省值

```
pickauthor file
```

说明

PICKAUTHOR 提供了一种在命令行上覆盖参考文件的方法。其可以用于在命令行提供作者的优先列表信息。或者将 SAC 从一个参考文件重定向到另一个。更多关于参考文件的信息，参见 PICKPREFS 以及 READCSS

注意：如果当数据在数据缓冲区内而用户修改了参考文件，那么在 SAC 缓冲区中的震相拾取将可能被修改。(缓冲区的信息可以通过 LISTHDR 和 CHNHDR 查看)。

例：如果作者列表是 “john rachel michael”，并且一些文件被用 READCSS 命令读入，一些到时可能以 author = michael 读入。(用户可能不会注意到对于某个给出的震相拾取其作者是谁，因为在 CSS 中的作者字段在 SAC 格式中并不会出现)。如果用户稍后使用了 PICKAUTHOR 命令并修改作者列表为 “peter doug rachel”，然后 READCSS MORE 读入更

多文件，没有 author = michael 的文件读入，已经在内存中的文件将丢失以 michael 作为作者的拾取。不了解这个用户可能会莫名地发现似乎震相拾取会随机消失。更多的信息参见 PICKPREFS

相关命令

pickprefs、pickphase

11.100 pickphase

概要

告诉 SAC 从一个用户定义的参考文件中读入震相列表（也可能是作者信息），或在该命令行上输入震相列表

语法

```
PICKPHASE header phase author [ header phase author ... ]
PICKPHASE FILE [filename]
PICKPHASE AUTHOR [filename]
```

输入

- header : 头段变量名： t0 - t9.
- phase : 对于给出的头段变量对应的拾取的震相名
- author : 告诉 SAC 作者列表，为某个头段所对应的作者，或者是”-”
- FILE : 如果使用了 FILE 选项，SAC 将从参考文件中读取震相列表。如果参考文件的文件名在命令行上给出，则 SAC 将读取这个指定的文件，否则 SAC 将根据上一次执行 PICKPHASE 读取最近输入的文件名. 如果未给出文件名，则 SAC 使用 `sac/aux/csspickprefs`
- AUTHOR : 这个选项和 FILE 选项相似，其另一个功能是允许 SAC 读取指定的头段变量信息

缺省值

PICKPHASE FILE

说明

PICKPHASE 用于在命令行上覆盖参考文件。其可以用于在命令行提供指定的头段/震相/作者信息，或者将 SAC 从一个参考文件重定向到另一个。更多关于参考文件的信息，参见 PICKPREFS 以及 READCSS

注意：如果当数据在数据缓冲区内而用户修改了参考文件，那么在 SAC 缓冲区中的震相拾取将可能被修改。(缓冲区的信息可以通过 LISTHDR 和 CHNHDR 查看)。

例：如果当一些含有某些 pP 或 PKiKP 震相的 SAC 文件通过 READ 命令读入时，被允许的震相包括 pP 以及 PKiKP，那么这些拾取将出现在 Tn 时间标记中。如果 PICKPHASE 在稍后再次使用将 pP 以及 PKiKP 从允许的震相中去除，那么 pP 以及 PKiKP 到时将不会从 CSS 文件中读取，已经在内存中的数据的 pP 和 PKiKP 拾取将从 Tn 时间标记中去除。

相关命令

pickprefs、pickauthor

11.101　pickprefs

概要

这个命令用于控制 SAC 管理震相和/或从不同的输入数据格式 (例如 CSS,GSE,SUDS...) 读入到时间标记变量 T0 到 T9，到这个选项为 OFF(缺省状态), 则载入到时间标记的震相为 SAC 在输入文件中发现的第一个拾取，如果这个选项为 ON，SAC 将使用 READCSS 命令中描述的参考文件。

语法

```
PICKPREFS ON|OFF
```

输入

- ON : 指示 SAC 通过参考文件将到时信息由 CSS 缓冲区传送到 SAC 缓冲区。这对于需要特定到时位于特定头段变量的宏文件来说很有用
- OFF : 指示 SAC 绕过参考文件，载入给定文件的前 10 个震相拾取。这个是默认值，它允许用户注意一些在 PICKPREFS ON 时无法注意的拾取。

缺省值

```
pickprefs off
```

说明

从版本 0.58 开始, sac2000 就有两个不同的头段缓冲区：一个根据 SAC 文件格式构建，一个根据有关的 CSS3.0 文件格式。添加了 CSS 数据缓冲区使得读取如 CSS、GSE、SUDS 等这里相关格式变得更加容易。两个缓冲区同时也使得下面的几个命令得以使用：COMMIT、ROLLBACK、RECALLTRACE（参看具体文档）有两个缓冲区的一个缺点是将到时从动态的 CSS 到时表转移到 SAC 格式中相对静态的 Tn 拾取变得更加复杂了，这个问题在 0.58 版本中通过设置了一个称为 csspickprefs 的参考文件得以解决。这个文件在 aux 目录下，你可以覆盖它写入你想要的信息。更多关于 csspickprefs 的信息，参见 READCSS 命令。关于如何覆盖默认参考文件，参看 PICKAUTHOR 或 PICKPHASE

使用参考文件的缺点是它将只接收列在参考文件中或在命令 PICKPHASE、PICKAUTHOR 中输入的震相或作者列表。换句话说，如果一个 CSS 数据文件有一个 pP 的到时，不管其来自于平面文件还是 Oracle 数据库，而 pP 未在参考文件中指定，那么用户就绝不会知道 pP 在那里。pP 震相将读入 SAC 中的 CSS 数据缓冲区，但是它不会转变到 SAC 数据缓冲区中，也不会参与任何 SAC 命令。它可以通过 WRITECSS 命令写出，或者可能通过 COMMIT 命令刷新然后全部丢失。

我们给出的解决办法是允许用户绕过参考文件。在 0.59 版本中，默认是从 CSS 缓冲区中直接读入前 10 个可用的拾取到 SAC 缓冲区中，通过这个新命令 PICKREFS 的使用，用户可以告诉 SAC 使用指定的参考文件。

相关命令

pickauthor、pickphase

11.102 picks

概要

控制在多数绘图上时间标记的显示

语法

```
PICKS [ON|OFF] [tmarker VERTICAL|HORIZONTAL|CROS] [WIDTH v] [HEIGHT v]
```

输入

- PICKS ON : 打开震相拾取显示
- PICKS OFF : 关闭震相拾取显示
- tmarker : SAC 的时间标记头段变量名，可以取 O | A | F | Tn（n=0-9）
- VERTICAL : 在时间标记处绘制垂直线，震相拾取 id 位于线的右下方。
- HORIZONTAL : 位于最接近时间标记的数据点处绘制水平线，震相拾取 id 位于线上或线下
- CROSS : 在时间标记处绘制垂直线，在最近的数据点处绘制水平线
- WIDTH v : 拾取标记的显示宽度为 v
- HEIGHT v : 改变拾取标记的显示高度为 v。高度和宽度仅对 HORIZONTAL 和 CROSS 使用，其数值为占图形的比例

缺省值

```
pick on width 0.1 height 0.1
```

所有的拾取标记类型都是 VERTICAL

说明

这个命令控制多数 SAC 绘图上时间标记的显示信息。这些时间标记标识了如震相到时、事件发生时刻等。当打开显示选项时，每一个定义了的时间标记都会在绘图上相应时刻处绘制一条线，并且在其旁有一个时间标记 id。时间标记 id 是一个 8 字符长的头段变量。头段变量中 KA、KF、KO 以及 KTn 分别是 A、F、O 和 Tn 的时间标记 id。如果时间标记 id 未定义，那么标记的名字就是就本身。每一个时间标记可以被显示为一条垂直线、一条水平线或一个交叉线。

例子

以交叉符号显示时间标记 T4、T5 和 T6，并改变交叉符号的高度和宽度:

```
SAC> picks t4 c t5 c t6 c w 0.3 h 0.1
```

11.103　plabel

概要

定义一般绘图的标签以及它们的属性

语法

```
PLABEL [n] [ON|OFF|text] [SIZE TINY|SMALL|MEDIUM|LARGE]
    [BELOW|POSITION x y [a]]
```

输入

- n : 绘图标签号目前有 5 个，如果省略，则为上一个标签号加 1
- ON : 打开绘图标签选项
- OFF : 关闭绘图标签选项
- text : 改变绘图标签的文本内容，同时打开了绘图标签选项
- SIZE size : 改变绘图标签的尺寸
- TINY : 微小尺寸，每行 132 个字符
- SMALL : 小尺寸，每行 100 个字符
- MEDIUM : 中等尺寸，每行 80 字符
- LARGE : 大尺寸，每行 50 字符
- BELOW : 将这个标签放在以前的标签的下面
- POSITION x y a : 定义该标签的位置，其中 x 的取值为 0 ～ 1，y 的取值为 0 到最大视口（一般为 0.75），a 是标签相对于水平线的顺时针旋转的角度

缺省值

默认字体大小为 small，标签 1 的位置为 0.15 0.2 0.。默认其他标签的位置为上一个标签之下。

说明

这个命令允许你定义一般用途的绘图标签用于后面的绘图命令，你可以定义每个标签的位置。文本质量以及字体可以用 GTEXT 命令设定，也可以使用 TITLE、XLABEL、YLABEL 生成图形的标题以及轴标签。

例子

下面的命令将在接下来的绘图中在左上角产生一个四行的标签:

```
SAC> PLABEL 'Sample seismogram' POSITION .12 .5
SAC> PLABEL 'from earthquake'
SAC> PLABEL 'on January 24, 1980'
SAC> PLABEL 'in Livermore Valley, CA'
```

一个额外的小标签可以放在左下角:

```
SAC> PLABEL 5 'LLNL station: CDV' S T P .12 .12
```

相关命令

gtext、title、xlabel、ylabel

11.104 plot

概要

绘制单道 (single-trace) 单窗口图形

语法

```
PLOT [PRINT [pname]]
```

输入

- PRINT pname : 打印绘图结果到打印机

说明

每个数据文件绘制在单独的图上，图形的整天大小由当前是口决定 (参见 XVPORT 和 VPORT)。每个图形的 Y 轴范围由数据的极值决定，也可以自己限制 y 轴的范围（参见 YLIM 命令)。X 轴的范围由 XLIM 命令控制。用户可以控制每张图的文件 id(参见 FILEID)。时间标记也可以显示出来 (see PICKS)。SAC 会在每张图之间暂停给你机会检查每张图，其将会在终端输出 “Waiting” 并等待你的响应。你可以输入回车以查看下一张图，关键字 “go” 或 “g” 可以不暂停地绘制余下的所有文件，或者关键字 “kill” 或 “k” 可以中止绘制余下的文件。

错误消息

- 1301: 未读入数据文件

相关命令

xvport、yvport、xlim、ylim、fileid、picks、filenumber

11.105 plot1

概要

绘制多道 (multi-trace) 多窗口图形

语法

```
PLOT1 [ABSOLUTE|RELATIVE] [PERPLOT ON|OFF|n] [PRINT pname]
```

输入

- ABSOLUTE : 图形文件的时间为绝对时间。具有不同开始时间的文件将做相对移动。
- RELATIVE : 所有的文件绘制时相对于自己的开始时间
- PERPLOT ON : 每张图绘制几个文件，使用 n 的旧值
- PERPLOT OFF : 所有文件绘制在一张图上
- PERPLOT n : 每张图上绘制 n 个文件
- PRINT pname : 打印结果到打印机

缺省值

```
plot1 absolute perplot off
```

说明

每个数据文件共享一个共同的 x 轴，但是各自拥有一个单独的 y 轴。绘图的总尺寸由由当前视口决定 (参见 XVPORT 和 YVPORT)。每一个子图的大小这个视口的大小以及当绘图的文件数目决定。每个子图的 Y 轴范围由文件极值决定或者可以通过 YLIM 命令自己设置。X 轴的范围可以是固定的 (参见 XLIM) 也可以是与数据成比例的。对于这种类型的绘图 X 轴标度有两种类型：relative 和 absolute。在绝对标度中 X 轴的范围为绘图文件的自变量的最小值和最大值。在不同子图上两点之间的时间差也按照起始时间的不同而得到校正。在相对标度中，X 轴将从 0 到文件中的最大时间差（即取所有文件中开始时间和结束时间差最大的那个时间差以保证每个文件都可以被完整地绘制，尽管这样会导致某些图有很多的无数据区域），每个文件都将从图形的左边界即 X 轴的零点开始绘制，对每个文件这个零点所对应的真实值是不同的，其将在文件名下方给出。这种标度类型在你相对某个时间标记（比如初动到时）裁剪文件时很有用。这样使得很容易看到每个文件波形之间的相似或差别。

例子

下面的例子是由 LLNL DSS 的 4 个台站 Elko、Kanab、Landers 和 Mina 记录到的美国西部的一个地震。参考时间被设置为事件发生时刻 (KZDATE and KZTIME):

```
SAC> cut -5 200
SAC> read *v
 elk.v knb.v lac.v mnv.v
SAC> fileid location ul type list kstcmp
SAC> title 'regional earthquake:  :1,kztime:  :1,kzdate:'
SAC> qdp 2000
SAC> plot1
```

注意 read 命令中的 UNIX 通配符的使用。

错误消息

- 1301: 未读入数据文件

相关命令

xlim、ylim、fileid、picks、filenumber

11.106　plot2

概要

产生一个多道（multi-trace）单窗口 (即重叠) 绘图

语法

```
PLOT2 [ABSOLUTE|RELATIVE] [PRINT pname]
```

输入

- ABSOLUTE : 所有文件时间为绝对时间，具有不同的文件开始时间的文件将会相对于第一个文件移动

- RELATIVE : 每个文件相对于自己的开始时间绘图
- PRINT pname : 打印绘图结果到打印机

缺省值

```
p2 absolute
```

说明

所有文件列表中的文件都将绘制在同一个绘图窗口中。可以选择绘制一个包含绘图符号以及文件名的图例以区别不同的文件，在使用这个命令之前 X 和 Y 的范围可以被定义。参见 XLIM 和 YLIM 命令。如果没有给出轴的范围那么将根据文件列表中的极值确定轴范围。图例的位置可以通过 FILEID 命令控制。不像 PLOT 和 PLOT1，PLOT2 可以绘制谱文件。实部 -虚部数据被绘制为实部 -频率。振幅/相位数据被绘制为振幅 -频率。虚部和相位信息忽略。谱文件总是用相对模式绘图。注意到在频率域 b、e 和 delta 分别被设置为 0、Nqquist 频率以及频率间隔 df。头段值 depmin 和 dapmax 未改变。如同 PLOTSP，如果 XLIM 关闭，绘图将从 df=DELTA 处开始，而非 0。如果 XLIM 或 YLIM 在数据变换到频率域之前被改变了，最好在使用 PLOT2 绘图之前输入 XLIM off 和 YLIM off。

注意: 由于某些原因，可能在内存中同时存在时间序列文件和谱文件并且没有选择相对绘图选项，则时间序列将以绝对模式绘图，谱文件将以相对模式绘图。相对模式意味着相对于第一个文件。因而内存中文件的顺序将影响绘图之间的关系。

例子

```
SAC> read mnv.z.am knb.z.am elk.z.am
SAC> xlim 0.04 0.16
SAC> ylim 0.0001 0.006
SAC> linlog
SAC> symbol 2 increment
SAC> title 'rayleigh wave amplitude spectra for nessel'
SAC> xlabel 'frequency (hz)'
SAC> plot2
SAC> fft
SAC> xlim off ylim off
SAC> line increment list 1 3
SAC> plot2 print
```

错误消息

- 1301: 未读入数据文件

相关命令

xlim、ylim、fileid、filenumber

11.107 plotalpha

概要

从磁盘读入字符数据型文件到内存并将数据绘制出来，这个与 readtable 之后跟个 plot 不同，因为它允许你在每个数据点上绘制标签

语法

```
PLOTALPHA [MORE] [DIR CURRENT|name] [FREE|FORMAT text] [CONTENT text]
    [PRINT printer] [filelist]
```

输入

- MORE : 将新读入的文件加到内存中老文件之后。如果没有这个选项，新文件将代替内存中的老文件。参见 READ 命令。
- DIR CURRENT : 从当前目录读取并绘制所有文件
- DIR name : 从文件夹 name 中读取并绘制所有文件，其可以是相对或绝对路径
- FREE : 以自由格式（以空格分隔数据各字段）读取并绘制 filelist 中的数据文件
- FORMAT text : 以固定格式读取并绘制 filelist 中的数据文件，格式声明位于 text 中
- CONTENT text : 定义 filelist 中数据每个字段的含义。text 的含义参见 READTABLE 命令中的
- PRINT pname : 打印输出结果到打印机
- filelist : 字符数字型文件列表，其可以包含简单文件名，绝对/相对路径，通配符。

缺省值

```
plotalpha free content y. dir current
```

说明

参考 readtabel 命令的相关说明。

例子

读取并绘制一个自由格式的 X-Y 数据，且其第一个字段是标签:

```
SAC> plotalpha content lp filea
```

错误消息

- 1301: 未读入数据文件（未给出文件列表或列表中的文件不可读）
- 1020: 无效的内置函数名
- 1320: 可用内存太小不足以读取文件
- 1314: 数据文件列表不能以数字开头
- 1315: 数据文件列表中文件的最大数目为:

警告消息

- 0101: 打开文件
- 0108: 文件不存在

头段变量改变

b, e, delta, leven, depmin, depmax, depmen.

相关命令

read、write、readtable

11.108 plotc

概要

使用光标标注 SAC 图形和创建图件

语法

```
PLOTC [REPLAY|CREATE] [FILE|MACRO filename] [BORDER ON|OFF]
```

输入

- REPLAY : 重新显示或绘制一个已经存在的文件或宏，关于文件和宏的区别见下
- CREATE : 创建一个新的文件或宏
- FILE filename : 重新显示或创建一个文件。如果省略文件名则使用上一个文件
- MACRO filename : 重新显示或创建一个宏
- BORDER ON|OFF : 打开/关闭图形的边界

缺省值

```
plotc create file out border on
```

说明

这个命令让你可以注释一个 SAC 绘图或者创建一个用于会议或报告的图件。简单的说就是一个简单的 “画图” 软件。你需要一个具有光标的图形设备。你可以通过在终端屏幕上放置一个目标（比如圆、方）或者文本创建一个图件。光标的位置决定了目标绘制的位置，敲入的字符决定了要绘制哪个目标。目标包括圆、矩形、多边形、线、箭头、弧，还有多种放置文本的方法。

这个命令绘制出的图可以直接截图利用，绘制过程中用到的所有命令将作为输出文件输出。这个命令有两种输出文件格式: 简单文件或宏文件。两者都是字符数字型文件，可以直接用文本编辑器修改。它们包含了 PLOTC 命令中光标响应的历史以及位置。宏文件一旦被创建，可以用于更多的绘图。其比例和旋转角度均可以修改。简单的 PLOTC 文件名以 “.PCF” 为后缀，宏文件名则以 “.PCM” 为后缀。这使得你可以区分你的目录下的这些文件。

当你创建一个新的文件或宏时，SAC 在屏幕上绘制一个矩形，代表你可以绘图的区域，你可以将光标移动到该区域内任何你想放置的位置，并输入代表你想绘制的目标或你想要光标执行的动作所对应的字符。

有两种光标选项类型：动作类和参数设置类。动作类选项将做一些事情 (绘制一个矩形，放一些文本)，他们如何执行这个操作部分基于当前参数设置选项的值 (例如多边形的边数，文本大小等)。这个区别与 SAC 自己的操作命令和参数设定命令相同。下面会列出动作和参数设定选项。

当你重新显示一个文件或者宏时，图形在终端屏幕上将会重新绘制，光标也将打开。你可以像你当初创建这个文件一样向其中加入目标。当你完成创建图件之后你可以将其发送到不同的图形设备，使用 BEGINDEVICES 命令临时关闭终端屏幕打开其他图形设备 (比如 sgf)，然后重新显示这个文件。

为了注释一个 SAC 绘图，要执行 VSPACE 命令设置正确的横纵比，然后执行 BEGINFRAME 命令关闭自动刷新，执行需要的 SAC 绘图命令，执行 PLOTC 命令（创建或者重新显示），然后执行 ENDFRAME 命令恢复自动刷新。

例子

下面的例子展示了如何使用 PLOTC 命令给一个 SAC 标准绘图添加注释。命令如下，括号中的内容为注释:

```
SAC> fg impulse npts 1024                    //生成文件
SAC> lp c2 n 7 c 0.2 t 0.25 a 10             //低通滤波
SAC> fft
 DC level after DFT is 1
SAC> axes only l b                           //左和下坐标轴设置
SAC> ticks only l b
SAC> border off
SAC> fileid off
SAC> qdp off
SAC> vspace 0.75                            //修改图形尺寸
SAC> beginframe                             //开始绘图
SAC> psp am linlin                          //绘图
SAC> plotc create file bandpass             //开始在图上做注释
...用光标和键盘进行各种操作...
SAC> endframe
```

PLOTSP 用于绘制滤波响应曲线以及两个轴，PLOTC 用于用于交互式地添加注释。VSPACE 命令限制了图形中纵横比为 3:4 的区域为绘图区域。这个对于之后将输出发送到具有纵横比 3:4 的 SGF 设备来说很有必要。在这之后你将有一个叫做 “BANDPASS.PCF” 的文件，其中包很了这个图形的注释信息。

为了将注释写入 SGF 文件:

```
SAC> begindevices sgf                       //打开sgf设备
SAC> beginframe
SAC> plotsp
SAC> plotc replay                           //重新绘制上一注释图
SAC> endframe
```

这样一个包含注释绘图的 SGF 文件就建立了。

注意

1. 只有当设置正方形视窗 (VSPACE 1.0) 时绘制的圆形和扇形才是正确的，否则只能产生一个椭圆, 其纵横比等于视窗的纵横比。
2. 除文本之外的所有操作码都按比例适应图形窗口。

文本尺寸并不是当前标度的。当你生成一个图像并想要将文本放在一个矩形或圆中时会产生一个问题。在这种情况下，图形窗口必须与输出页具有相同的尺寸，以避免图形的偏差。这可以通过使用 WINDOW 命令设置窗的水平 (x) 尺寸为 0.75，垂直 (y) 尺寸为 0.69。例如：WINDOW 1 X 0.05 0.80 Y 0.05 0.74。这个命令必须在窗口被创建之前执行。(即在 BEGINWINDOW 或 BEGINDEVICES 之前)

相关命令

vspace、begindevices、beginframe、endframe

表 11.1: plotc 命令表

字符	含义
A	绘制一条到 ORIGIN 到 CURSOR 的箭头
B	在绘图区周围绘制边界的 tick 标记
C	绘制一个圆心在 ORIGIN，且经过 CURSOR 的圆
D	从 replay 文件中删除最后一个动作选项
G	设置 ORIGIN，并将其全局化
L	绘制一条从 ORIGIN 到 CURSOR 的线
M	在 CURSOR 处插入一个宏文件（输入宏文件名，比例因子和旋转角。若没有指定，则使用上一次的值，默认是 OUT，1.0，0）
O	设置 ORIGIN 为 CURSOR
N	绘制一个中心在 ORIGIN，一个顶点位于 CURSOR 的 n 边形
Q	退出 PLOTC
R	绘制对脚位于 ORIGIN 和 CURSOR 的长方形
S	绘制一个圆心位于 ORIGIN 的扇形（用光标的移动来指定扇形的角度，键入 S 来绘制一个小于 180 度的扇形，或者键入 C 绘制它的补集）
T	在 CURSOR 处放置一行文本，文本以回车键结束
U	在 CURSOR 处放置多行文本，文本以空白行结束

关于 PLOTC 命令表的说明

- CURSOR 表示当前光标位置
- ORIGIN 一般为上次光标的位置
- G 选项强制 ORIGIN 固定
- O 选项再次允许 ORIGIN 移动
- Q 选项不自动拷贝至文件，但是可以通过文本编辑器直接加入

如果 SAC 在 replay 模式没有在文件中看到 Q 选项，则其在显示文件内容之后回到光标模式，这使得你可以在文件结束之后继续增加更多的选项。如果 SAC 在文件中看到 Q 选项，则显示其内容并退出。文件中以星号开头的行为注释行。PLOTC 还有一些更复杂的选项，但是运行起来好像有点问题，有兴趣的可以试试 “help plotctable”。

11.109 plotdy

概要

绘制一个带有误差棒的图

语法

```
PLOTDY [ASPECT ON|OFF] [PRINT pname] name|number [name|number]
```

输入

- ASPECT : ON 保持 3/4 的纵横比。OFF 允许纵横比随着窗口维度的变换而变换
- PRINT pname : 打印输出到打印机

- name : 数据文件列表中数据名
- number : 数据文件列表中的数据号

说明

这个命令允许你绘制一个带有误差棒的数据集。你选择的第一个数据文件（通过名字或文件号指定）将沿着 y 轴绘制第二个数据文件是 dy 值，如果选择了第三个数据文件则其为正的 dy 值

例子

假定你有一个等间距的 ASCII 文件，其包含了两列数据。第一列是 y 值，第二列是 dy 值，你可以像下面那样读入 SAC 并用数据绘制误差棒:

```
SAC> readtable content yy myfile
SAC> plotdy 1 2
```

错误消息

- 1301: 未读入数据文件

11.110　plotpk

概要

产生一个用于拾取到时的图

语法

```
PLOTPK [PERPLOT ON|OFF|n] [BELL ON|OFF] [ABSOLUTE|RELATIVE]
    [REFERENCE ON|OFF|v] [MARKALL ON|OFF] [SAVELOC ON|OFF]
```

输入

- PERPLOT ON : 一张图绘制 n 个文件，使用 n 的旧值
- PERPLOT OFF : 在一张图上绘制所有文件
- PERPLOT n : 一张图上绘制 n 个文件，比如拾取震相时常使用 n=3（三分量)
- BELL [ON]|OFF : 击键时是否响铃
- ABSOLUTE : 显示绝对 GMT 格式时间
- RELATIVE : 显示相对于每个文件的参考时间
- REFERENCE [ON] : 打开参考线显示
- REFERENCE OFF : 关闭参考线选项
- REFERENCE v : 打开参考线选项并设置参考值为 v，设置为 0 会很方便
- MARKALL [ON] : 同时储存一张图上的所有文件的时间标记，用于拾取震相
- MARKALL OFF : 只保存光标位置所在的那个文件的震相拾取标记
- SAVELOCS [ON] : 保存拾取的位置（由命令 L 得到，稍后介绍）到暂存块变量中
- SAVELOCS OFF : 不保存拾取位置到暂存块变量

缺省值

```
plotpk perplot off absolute reference off markall off savelocs off
```

说明

这个命令的格式类似于 PLOT1，这个绘图需要一个带有光标的终端。在该命令后将在屏幕上出现光标，输入单个字符即可执行不同的操作。有些但不是全部字符将在屏幕上产生图形输出。错误以及输出信息将在屏幕上部打印，当前在头段中的震相拾取将自动显示在屏幕上。不同的光标响应的输出可以定向到 SAC 头段中，也可以定向到一个字符数字型震相拾取文件（OAPF），或者到一个 HYPO 震相拾取文件，或者到终端。如果你使用 SAVELOCS 选项保存由 L 光标选项得到的光标位置到暂存块变量，那么下面的暂存块变量将被定义:

- NLOCS: 在命令执行期间拾取的光标位置数，每次执行 PPK 时初始化为 0，每次拾取光标位置则加 1
- XLOCn: 拾取光标位置的 x 值，如果头段中的参考时间被定义其是 GMT 时间，否则其为偏移时间
- YLOCn: 第 n 个拾取光标位置的 y 值

头段变量改变

A, KA, F, KF, Tn, KTn.

错误消息

- 1301: 未读入数据文件
- 1202: 变量块个数超过了最大数目

警告消息

- 1502: 错误的光标位置，请重新尝试（光标位于绘图窗口之外）
- 1503: 无效字符，请重新尝试（输入的字符 SAC 无法识别为合法响应）
- 1905: 需要一个整数, 请重新尝试（在响应 T 之后未输入整数）
- 1906: 需要一个 0 到 4 的整数，请重新尝试（在响应 Q 之后未输入 0, 1, 2 或 3）
- 1907: HYPO 行已经写入
- 1908: HYPO 震相拾取文件未打开
- 1909: 无法计算波形（调整光标位置并重试，绘图总是 ABSOLUTE 模式）

相关命令

plot1、ohpf、oapf、apk

11.111 plotpm

概要

针对一对数据文件产生一个 “质点运动” 图

语法

```
PLOTPM [PRINT pname]
```

输入

- PRINT pname : 打印结果到打印机

说明

在一个质点运动图中，一个等间距文件相对于另一个等间距文件绘图。对于没有自变量的值，第一个文件的因变量的值将沿着 y 轴绘制，第二个文件的因变量值沿着 x 轴绘制。对于一对地震图来说，这种图展示了一个质点在两个地震图所确定的平面内随时间的运动。它将产生一个方形图，每个轴的范围为因变量的极大极小值。注释轴沿着底部和左边绘制。轴标签以及标题可以通过 XAXIS, YAXIS 和 TITLE 命令设置。如果未设置 x、y 标签，则台站名和方位角将用作轴标签。XLIM 可以用于控制文件的哪些部分需要被绘制。

例子

创建一个两个地震图的质点运动图:

```
SAC> read xyz.t xyz.r
SAC> xlabel 'radial component'
SAC> ylabel 'transverse component'
SAC> title 'particle-motion plot for station xyz'
SAC> plotpm
```

如果你想要值绘制每个文件在初动附近的一部分，你可以使用 XLIM 命令:

```
SAC> xlim a -0.2 2.0
SAC> PLOTPM
```

你也可以使用 PLOTPK 在之前的绘图窗口中设置新的区域，然后绘制命令如下:

```
SAC> beginwindow 2
SAC> plotpk
 ... mark the portion you want using X and S
 ... terminate PLOTPK with a Q
SAC> beginwindow 1
SAC> plotpm
```

11.112　plotsp

概要

用多种格式绘制谱数据

语法

```
PLOTSP [ASIS|RLIM|AMPH|RL|IM|AM|PH] [LINLIN|LINLOG|LOGLIN|LOGLOG]
```

输入

- ASIS : 按照谱文件当前格式绘制分量
- RLIM : 绘制实部和虚部分量
- AMPH : 绘制振幅和相位分量
- RL : 只绘制实部分量
- IM : 只绘制虚部分量
- AM : 只绘制振幅分量
- PH : 只绘制相位分量

- LINLIN|LINLOG|LOGLIN|LOGLOG : 设置 x-y 轴为线型还是对数型，与单独的 LINLIN 等命令区分开

缺省值

```
plotsp asis loglog
```

说明

SAC 数据文件可能包含时间序列文件或谱文件，IFTYPE 决定文件所有哪种类型。多数绘图命令只能对时间序列文件起作用，这个命令则可以绘制谱文件。

你可以使用这个命令绘制一或两个分量。每一个分量绘制在一张图上。你也可以设置横纵坐标为线型或对数型，其仅对该命令有效.

例子

获得一个谱文件振幅的对数 -线性的绘图:

```
SAC> read file1
SAC> fft
SAC> plotsp am loglin
```

错误消息

- 1301: 未读入数据文件
- 1305: 对时间序列的非法操作

11.113 plotxy

概要

以一个文件为自变量，一个或多个文件为因变量绘图

语法

```
PLOTXY name|number name|number [name|number ...]
```

输入

- name : 数据文件表中的一个文件名
- number : 数据文件表中的一个文件号

说明

你选择的第一个文件（通过文件名或文件号）为自变量，沿着 x 轴绘制。余下的数据文为因变量，沿着 y 轴绘制。所有的图形环境命令比如 TITLE, LINE 和 SYMBOL 都可以使用以控制绘图的属性。这个命令用于绘制由 READTABLE 命令读入的多列数据。在多情况下，其可以看作像用绘图命令一样作出的图表。

例子

假设你有一个 ASCII 文件，其包含 4 列数。你想要将其读入 SAC 并绘图。下面的命令读入这个文件，将其储存为 SAC 内部 4 个分开的文件，打开线型增量开关，然后以第二列为自变量，其他列为因变量绘图:

```
SAC> readalpha content ynnn myfile
SAC> line increment on
SAC> plotxy 2 1 3 4
```

11.114　print

概要

打印最近的 SGF 文件，这个命令需要在此之前至少产生一个 SFG 文件

语法

```
PRINT [printer]
```

输入

printer 将输出发送到打印机 printer，如果没有指定则使用系统默认打印机

如果自从登录之后 SGF 设备一直保持关闭，那么 print 命令就不会工作。使用 bd 打开 SGF 设备，使用 SGF 命令设置 SGF 设备的属性

缺省值

```
print
```

错误信息

- 2405: 无法打印：没有产生 SGF 文件

相关命令

sgf、begindevices

11.115　printhelp

概要

打印 SAC 命令的帮助文档到打印机

语法

```
PRINTHELP [item ...]
```

输入

- item : 命令、模块、子程序、特性的全称或简称

默认值

若没有指定 item，则打印 help 包的简介

说明

该命令与 help 命令的用法相同，只是其会调用系统的默认打印机将文档打印出来

错误消息

- 1130：没有可用的帮助文档包
- SAC 无法找到文档包，检查你的环境变量 SACAUX
- lpr：错误 - 没有默认目的位置。系统未设置默认打印机

11.116 production

概要

控制作业方式的选择

语法

```
PRODUCTION ON|OFF
```

输入

- ON|OFF 打开/关闭作业模式

缺省值

```
prod off
```

说明

当这个选项打开时，致命错误将立刻终止 SAC。当这个选项为关时，在致命后将控制回到终端，可以继续执行 SAC

11.117 qdp

概要

控制低分辨率快速绘图选项 “quick and dirty plot”

语法

```
QDP [ON|OFF|n] [TERM ON|OFF|n] [SGF ON|OFF|n]
```

输入

- ON|OFF : 打开/关闭终端和 SGF 文件的 QDP 选项
- n : 打开终端和 SGF 文件的 QDP 选项，并改变绘制的数据点数为 n
- TERM ON|OFF : 打开/关闭终端 qdp 绘图选项
- TERM n : 打开终端的 QDP 选项，并改变绘制的数据点数为 n
- SGF ON|OFF : 打开/关闭终端 qdp 绘图选项
- SGF n : 打开终端的 QDP 选项，并改变绘制的数据点数为 n

缺省值

```
qdp term 5000 sgf 5000
```

说明

绘制大型文件会消耗很长的时间，“quick and dirty plot” 选项通过不绘制每一个点来加速绘图。当这个选项为开时，SAC 将通过文件中数据点数除以你要求显示的数据点计算得到区间大小。文件越大，每个区间中数据点就越多。

SAC 然后后计算出每个区间内最小和最大数据点数。如果这个选项为开则 SAC 将在绘图的角落的小框中显示采样因子（区间尺寸的一半）。实际显示的数据点可能接近于它的值也可能相差很多，因为这个命令仅仅绘制区域中的极值。

对于目前的机器来说，一般大文件的绘图已经不会耗费太多时间了，所以一般都直接 qdp off

例子

假设文件 FILE1 有 20000 个数据点，文件 FILE2 有 40000 个数据点，如果你输入:

```
sac> r file1 file2
sac> bd x
sac> p
```

那么两张图都将准确包含 5000 个点，第一个文件将每 4 个点取一个点用于绘图，第二个每 8 个点取一个点绘图。数据区间的尺寸尽量取下限以保证你至少可以看到你所要求的点数。如果你要将图绘制到 SGF:

```
SAC> begindevices sgf
SAC> plot
```

11.118 quantize

概要

将连续数据数字化

语法

```
QUANTIZE [GAINS n ...] [LEVEL v] [MANTISSA n]
```

输入

- GAINS n ...: 设置允许的增益表，他们必须递减的，增益最大值为 8
- LEVEL v : 设置最低增益的量子化水平，这是最小有效位数的伏特值
- MANTISSA n : 设置尾数的位数

缺省值

```
QUANTIZE GAINS 128 32 8 1 LEVEL 0.00001 MANTISSA 14
```

说明

这个命令类似于 Oppenheim 和 Schafer(1975, Fig. 9.1) 描述的 “rounding” quantization 算法。在这个算法中位数划分为多个用于描述特征的位、符号位和尾数位。用户可以指定尾数的位数。量化水平 (最小有效数字位或 LSB) 也可由用户指定，缺省的量化水平为 10 微

伏，这个量化函数描述的信号误差是这个量化水平的一半。在频率域中，这个误差或量化噪声为:

其中 DELTA 是采样频率。量化噪声单位是 counts*counts/Hz，相当于功率谱密度。量化噪声的均方根为: 然而，仅当信号的均方根水平远大于量化噪声的均方根值时，这才是量化引起的噪声的一个精确的近似。换句话说，如果分辨率不是几百个 counts 那么在量化噪声与量化的信号之间便是有关联的。关联系数近似于 LEVEL 与量化信号的均方根之比。(Fig. 11.13, Oppenheim and Schaffer, 1975).

增益可以由用户指定，以在自动增益系统中模拟增益步骤，缺省的增益是区域地震实验台网的增益。

头段变量改变

depmin, depmax, depmen

11.119 quit

概要

终止 SAC

语法

```
QUIT
```

说明

这个命令可以正常终止 SAC，在终止之前，SAC 结束所有活跃的图形的设备，关闭所有输出文件，销毁临时文件

11.120 quitsub

概要

终止当前活动的子程序

语法

```
QUITSUB
```

说明

这个命令用于终止当前活动的子程序，返回到 SAC 主程序，内存中的文件保留。

目前 SAC 提供了两个子程序:

- SES Spectral Estimation Subproess 谱估计子程序（目前其称为 SPE，而非 SES）
- SSS Signal Stacking Subprocess 信号迭加子程序

11.121　read

概要

从磁盘读取 SAC 文件到内存

语法

```
READ [MORE] [TRUST ON|OFF] [DIR CURRENT|name] [XDR|ALPHA|SEGY]
    [SCALE ON|OFF] [filelist]
```

所有的选项必须位于 filelist 之前。

输入

- MORE : 将新文件添加到内存中老文件之后，如果这个选项忽略，则新数据将替代老数据。
- TRUST ON|OFF : 这个选项用于解决将文件从 SAC 转换到 CSS 格式过程中出现的问题。当转换数据时，匹配的事件 id 意味着文件含有相同的事件信息，或者可能是两个不同格式合并之后出现的干扰。当 TRUST 打开时，SAC 将更有可能接受匹配事件 id 作为相同的事件信息，这依赖于 READ 命令与当前内存中数据的历史。
- DIR CURRENT : 从当前目录读取所有简单文件名，这个目录是你启动 SAC 的目录
- DIR name : 从目录 name 中读取全部简单文件，其可以为绝对/相对路径
- XDR : 输入的文件是 XDR 格式，这个格式用于实现不同构架的二进制数据的转换
- ALPHA : 输入文件是 SAC 格式的字符数字型文件，ALPHA 与 XDR 选项不兼容
- SEGY : 读取 IRIS/PASSCAL 定义的 SEGY 格式文件，这种格式允许一个文件包含一个波形
- SCALE : 只和 SEGY 选项搭配使用，SCALE 选项默认是关的。当 SCALE 选项为 OFF 时，SAC 从 SEGY 中读取 counts，当 SCALE 为 ON 时，SAC 将 counts 乘以文件给出的一个 SCALE 因子。如果 SCALE 为 OFF，则这个文件中的 SCALE 值将储存在 SAC 头段 SCALE 中，如果 SCALE 为 ON，SAC 的 SCALE 字段将被设置为 1.0，SCALE 由于仪器响应的原因是一个粗糙的方法。更好的方法是使用 TRANSFER 命令。最好在 READ 中不要使用 SCALE 选项，而是直接使用 TRANSFER 命令。SCALE 只应在 TRANSFER 命令需要的响应信息无法获得时才使用
- filelist : file | wild
- file : 一个文件名。其可以是简单文件名或路径名。路径名可以是相对/绝对路径。
- wild : 通配符以扩展文件名列表

缺省值

```
read commit dir current
```

说明

SAC 几乎所有命令只对当前内存中的文件操作。内存中的数据类似于文本编辑器使用的临时工作文件。READ 命令从一个或多个磁盘文件传递数据到内存。默认是读取文件中全部数据到内存。

CUT 命令可以用于指定只有磁盘文件的一部分要被读入。SAC 文件在 2000 年之后产生的文件被假定年的值为四位数字。年份为两个数字的文件被假定为 20 世纪，会被加上

1900。正常情况下内存中原先的数据会由于另一个 READ 命令的执行而丢失。新数据将取代老数据。如果关键字 MORE 是命令的第二个符号，则新数据将放在内存中老数据的后面。数据文件列表称为老文件列表和新文件列表的连接。在三种情况下 MORE 项可能有用:

- 文件列表太长在一行下无法打印出
- 在长文件列表中一个文件名拼错
- 一个文件被读入，作了些处理，然后与原始数据比较

文件名可以包含通配符，你可以使用他们去匹配单个字符、多个字符等等。

例子

read 命令的简单示例位于 “SAC 的读和写” 一节。

现在假设你启动 SAC 的位于目录 “/me/data” 下，你想要处理子目录 “event1” 和 “event2” 下的文件:

```
SAC> read dir event1 f01 f02
```

读取了目录/me/data/event1 下的文件:

```
SAC> read f03 g03
```

相同目录下的文件被读入:

```
SAC> read dir event2 *
```

/me/data/event2 下的全部文件被读入:

```
SAC> read dir current f03 g03
```

目录/me/data 下的文件被读入。

错误消息

- 1301: 未读入数据文件（未给出要读取的文件列表或文件列表中文件均不可读）
- 1320: 可用内存不足以读取文件
- 1314: 数据文件列表不得以数字开头
- 1315: 文件列表的最大数目为 1000
- 6002: 没有可用数据集

警告消息

- 0101: 打开文件
- 0108: 文件不存在
- 0114: 读取文件

头段变量改变

e, depmin, depmax, depmen, b

相关命令

cut、readerr、wild

11.122　readbbf

概要

将黑板变量文件读入内存

语法

```
READBBF [file]
```

输入

- file : 暂存块变量文件的名字，其可以是简单文件名或相对/绝对路径。

缺省值

```
READBBF BBF
```

说明

这个命令使你能够读取一个暂存块变量文件。这个文件必须先前使用 WRITEBBF 命令写入磁盘。这个特性让你能够将一次 SAC 执行的信息保存起来运用在另一次执行中。你也可以自己写代码获取暂存块文件中的信息。这使你可以在 SAC 和自己的程序之间传送信息。

相关命令

writebbf、setbb、getbb

11.123　readerr

概要

控制发生在 READ 命令过程中的错误

语法

```
READERR [BADFILE FATAL|WARNING|IGNORE] [NOFILES FATAL|WARNING|IGNORE]
    [MEMORY SAVE|DELETE]
```

输入

- BADFILE : 当文件不可读或不存在时出现的错误
- NOFILES : 文件列表中没有文件可读时出现的错误
- FATAL : 设置错误条件为 fatal，发送错误消息并停止执行这个命令
- WARNING : 发送警告消息，但继续执行命令
- IGNORE : 忽略错误条件，继续执行命令
- MEMORY : 如果无文件可读则对内存中原有的数据进行处理
- DELETE : 先前在内存中的数据将被删除
- SAVE : 内存中的数据将保持在内存中

缺省值

```
readerr badfile warning nofiles fatal memory delete
```

说明

当你试着使用 READ 命令将数据文件读入内存时可能会发生错误。文件可能不存在或虽然存在但不可读。当 SAC 遇到这些 badfiles 时，一般会发送警告消息，然后试着读取文件列表中余下的文件。如果你想要 SAC 在遇到坏文件时停止读取文件可以设置 BADFILE 为 FATAL。如果你不想看到警告信息，可以设置 BADFILE 为 INGORE。如果文件列表中文件均不可读，SAC 将发送错误信息并停止处理，如果你想要 SAC 发送警告信息或完全忽略这个问题，设置 NOFILES 为 INGORE。当然，SAC 内存中先前的文件也可以从内存中删除或者保留在内存中。

相关命令

read、cuterr

11.124 readhdr

概要

从 SAC 数据文件中读取头段到内存

语法

```
READHDR [MORE] [TRUST ON|OFF] [DIR CURRENT|name] [filelist]
```

输入

- MORE: 将新数据头段放在内存中老文件头段之后，如果忽略，则新数据文件头段将代替老的。
- TRUST ON|OFF : 这个选项用于解决将文件从 SAC 转换到 CSS 格式过程中出现的问题。当转换数据时，匹配的事件 id 意味着文件含有相同的事件信息，或者可能是两个不同格式合并之后出现的干扰。当 TRUST 打开时，SAC 将更有可能接受匹配事件 id 作为相同的事件信息，这依赖于 READ 命令与当前内存中数据的历史。
- DIR CURRENT : 从当前目录读取所有简单文件名，这个目录是你启动 SAC 的目录
- DIR name : 从目录 name 中读取全部简单文件，其可以为绝对/相对路径
- filelist : file | wild
- file : 一个文件名。其可以是简单文件名或路径名。路径名可以是相对/绝对路径。
- wild: 通配符以扩展文件名列表

说明

这个命令将一系列 SAC 文件的头段读入内存，你可以列出头段内容 (LISTHDR)，改变头段值 (CHNHDR)，将头段写回磁盘 (WRITEHDR)，这比读取整个文件到内存快的多，当然这在你只需要头段的时候才可行。

错误消息

- 1301: 未读入数据文件（未给出要读取的文件列表或列表中文件不可读）
- 1314: 数据文件列表不得以数字开头
- 1315: 文件列表中的最大数目为 1000
- 1335: 非法操作 -只有头段在内存中

警告消息

- 0101:: 打开文件错误
- 0108:: 文件不存在
- 0114:: 读取文件错误

相关命令

read、listhdr、chnhdr、writehdr、readerr

11.125　readsp

概要

读取 WRITESP 和 WRITESPE 写的谱文件

语法

```
READSP [AMPH|RLIM|SPE] [filelist]
```

输入

- RLIM : 读入实部和虚部分量
- AMPH : 读入振幅和相位分量
- SPE : 读取谱估计子程序文件，这个数据被从功率转换为振幅，相位分量设置为 0
- filelist : SAC 二进制数据文件列表

缺省值

```
READSP AMPH
```

说明

WRITESP 命令将每个谱数据分量作为一个单独的文件写入磁盘，你可以分别处理每个分量。这个命令让你能从两个分量重建谱数据，参见 WRITESP。SPE 选项允许你读取并转换由 WRITESPE 写出的谱文件格式。这也使你可以使用 MULOMEGA 和 DIVOMEGA 命令

相关命令

writesp

11.126　readtable

概要

从磁盘读取按列排列的字符数字型数据文件到内存

语法

```
READTABLE [MORE] [TRUE ON|OFF] [DIR CURRENT|name] [FREE|FORMAT tex]
    [CONTENT text] [HEADER number] [filelist]
```

所有的选项必须位于 filelist 之前。最后两个选项可以放在文件自己的第一行

输入

- MORE : 将新文件添加到内存中老文件之后，如果这个选项忽略，则新数据将替代老数据。
- TRUST ON|OFF : 这个选项用于解决将文件从 SAC 转换到 CSS 格式过程中出现的问题。当转换数据时，匹配的事件 id 意味着文件含有相同的事件信息，或者可能是两个不同格式合并之后出现的干扰。当 TRUST 打开时，SAC 将更有可能接受匹配事件 id 作为相同的事件信息，这依赖于 READ 命令与当前内存中数据的历史。
- DIR CURRENT : 从当前目录读取所有简单文件名，这个目录是你启动 SAC 的目录
- DIR name : 从目录 name 中读取全部简单文件，其可以为绝对/相对路径
- FREE : 用自由格式读取文件列表中的数据（以空格分割）
- FORMAT text : 以固定格式读取文件列表中的数据，格式声明放在 text 中 0
- CONTENT text : 定义数据每列的内容，关于 text 见下
- HEADER : 文件中有几个头段行需要跳过
- filelist : 字符数字型文件列表

缺省值

```
readtable commit free content y. dir current
```

说明

READ 命令将二进制文件读入内存，这个命令用于读取字符数字型文件，这些数据文件可以是固定格式或自由格式。它们可能包含多个数据集，一旦读入了内存就可以用 WRITE 命令创建 SAC 二进制数据文件。

这个命令最简单的使用就是一个输入一个 Y 数据集，这也是默认的。X-Y 对也可以通过修改 content 的内容读入。这个命令可以被用于直接读取其他程序输出的复杂格式数据。也可以用这个方法读入多个 Y 数据集，但只允许一个 X 数据集。

处理过程中最基本的头段变量将被计算，包括 npts、b、e、delta、leven、depmin、depmax 和 depmin。如果只有一个 y 数据集，内存中的数据文件名将和那个字符数据型磁盘文件名相同。如果有多个 y 数据集，则在文件名之加上一个两位数字。

字符数字型数据文件的每一行都将以自由格式或声明的格式读入。每行最多 160 个字符。content 选项用于决定对于数据每行的每个输入该如果处理。在 content text 中的每个字符分别代表了不同的数据元素，这些字符的顺序与数据中每行的输入所代表的含义相对应。content 字段允许的字符如下:

- Y: 下一个输入属于 Y(因变量) 数据集
- X: 下一个输入属于 X(自变量) 数据集
- N: 下一个输入属于数据集
- P: 下一对输入使用 X-Y 数据集
- R: 下一对输入使用 Y-X 数据集
- I: 忽略这个输入

还有一个重复计数器可以跟在上面的任何字符之后。这个重复计数器是一个 1 位或 2 位整数，其代表重复前面那个字符多少次，“.” 是一个无穷次重复的计数器，其只能出现在 content 的 text 的最后，意味着最后一个字符可以表示接下来的所有输入列。

例子

为了读取一个或多个自由格式的 X-Y 数据对：

```
SAC> readtable content p. filea
```

你不能在文件行之间打断一个 X-Y 数据对。假设你有一个包含了格式化数据的文件，在每行的中间有一个 X-Y 数据对。每行的其它数据都没有用。假设每行 Y 数据在 X 数据之前，一旦正确的格式声明给出了，就可以用下面的命令：

```
SAC> readtable content r format \(24x,f12.3,14x,f10.2\) fileb
```

注意：在左括号和右括号两边的"\"SAC 的转义字符，这很重要，因为 SAC 使用括号作为内联函数。由于没有重复计数器，因而只有一个 Y-X 数据对被从文件的每行读入。

假设你有一个文件 FILEC，其每行包括一个 X 值和 7 个不同数据集的 Y 值，其为 (8F10.2) 格式。为了在内存中创建 7 个不同的数据集，可以使用下面的命令：

```
SAC> readtable content xn . format \(8f10.2\) filec
```

这将在内存中产生 7 个不同的数据文件，其名称分别为 FILEC01, FILEC02 等等。

现在假设你不想读入第 5 个 Y 数据集，可以执行下面的命令：

```
SAC> readtable content xn6 format \(5f10.20x,2f10.2\) filec
```

另一个可以少敲键盘但是稍微低效一点的命令如下：

```
SAC> READTABLE CONTENT XN4IN2 FORMAT \(8F10.2\) FILEC
```

错误消息

- 1301: 未读入数据文件（未给出要读入的文件列表或文件列表中的文件不可读）
- 1020: 无效的内联函数名:
- 1320: 可用内存不足以读取文件
- 1314: 数据文件列表不得以数字开始
- 1315: 数据文件列表的文件最大个数为 1000

警告消息

- 0101: 打开文件
- 0108: 文件不存在

头段变量改变

b, e, delta, leven, depmin, depmax, depmen.

相关命令

read、write

11.127　report

概要

告知用户当前 SAC 的状态

语法

```
REPORT APF|COLOR|CUT|DEVICES|FILEID|GTEXT|HPF|LINE|MEMORY|MTW|PICKS|
    SYMBOL|TITLE|XLABEL|XLIM|YLABEL|YLIM
```

输入

- APF : 字符数字型震相拾取文件名
- COLOR : 当前颜色属性，需要一个图形设备被激活
- CUT : 当前 CUT 状态
- DEVICES : 在你的系统上可用的图形设备
- FILEID : 当前文件 id 显示属性
- GTEXT : 当前图形文本属性
- HPF : HYPO 震相拾取文件名
- LINE : 当前线型属性
- MEMORY : 内存管理器的有效存储块的转储。
- MTW : 当前的测量时间窗状态
- PICKS : 当前时间拾取显示属性
- SYMBOL : 当前符号绘制属性
- TITLE : 当前绘制标题属性
- XLABEL : 当前 x 轴标签属性
- XLIM : 当前 x 轴范围
- YLABEL : 当前 y 轴标签属性
- YLIM : 当前 y 轴范围

说明

这个命令用于找出一些 SAC 选项的当前值，其值将打印到终端

例子

为了获取当前颜色属性的列表:

```
SAC> report color
 COLOR option is ON
 DATA color is YELLOW
 INCREMENT data color is OFF
 SKELETON color is BLUE
 BACKGROUND color is NORMAL
```

为了获取 HYPO 文件名:

```
SAC> report apf hpf
 Alphanumeric pick file is MYPICKFILE
 HYPO pick file is HYPOPICKFILE
```

11.128 reverse

概要

反转数据点的顺序

语法

```
REVERSE
```

说明

这个命令反转内存中数据点的顺序

11.129　rglitches

概要

去掉信号中的坏点和时间标记

语法

```
RGLITCHES [THRESHOLD v] [TYPE LINEAR|ZERO] [WINDOW ON|OFF|pdw]
    [METHOD ABSOLUTE|POWER|RUNAVG]
```

输入

- THRESHOLD v : 设置起始的阀值水平为 v，校正那些绝对值大于或等于 v 的数
- TYPE LINEAR : 在坏数据点的两边的数据点之间线性内插以校正坏点
- TYPE ZERO : 将坏点设置为 0
- METHOD ABSOLUTE : 校正绝对值不小于阀值 v 的数据点
- METHOD POWER : 使用反向差分方法计算信号功率，并校正那些功率超过阀值 v 的数据点
- METHOD RUNAVG : 通过将一个长为 SWINLEN 的窗从整个数据的末尾以一点的增量移动到数据的开始，并计算滑动平均和标准差。每个新点都将和平均值相比，如果它们的差超过了 THRESH2 乘以当前标准差，又或者差大于 MINAMP 计数器，它则被当前平均值代替。

RUNAVG 方法总是用于整个地震图，与这个 RUNAVG 方法有关的三个选项为:

- SWINLEN v: 设置滑动平均窗的长度为 v
- THRESH2 v: 设置坏点的阀值
- MINAMP v: 设置坏点的最小幅度
- WINDOW ON: 仅校正前面定义的 pdw
- WINDOW OFF: 校正整个数据文件中的数据点
- WINDOW pdw: 仅校正当前定义的 pdw 中的数据点。

缺省值

```
rglitches threshold 1.0e+10 type linear window off method absolute
    swinlen 0.5 thresh2 5.0 minamp 50
```

说明

这个命令可以用于平滑由于坏点引起的不规则信号或某些数据采集系统产生的时标。这个命令检查每一个数据点，看它的起始改变量是否大于或等于给定的“起始阀值水平”。然后将这些坏数据点置 0 或者利用数据点前后的两点进行内插得到一个数据点。可以在整

个文件或者某个时间窗内去掉假信号。使用这个选项可以使你去掉那些比整个数据文件中的假信号。

头段变量改变

depmin, depmax, depmen

错误消息

- 1301: 未读入数据文件
- 1306: 对不等间隔文件非法操作
- 1307: 对谱文件非法操作

11.130 rmean

概要

去除平均值

语法

```
RMEAN
```

错误信息

- 1301: 未读入文件
- 1307: 对谱文件非法操作

头段变量

depmen

11.131 rms

概要

利用测量时间窗计算数据的均方根

语法

```
RMS [NOISE ON|OFF|pdw ] [TO USERn]
```

输入

- NOISE ON : 打开噪声归一化选项
- NOISE OFF : 关闭噪声归一化选项
- NOISE pdw : 打开噪声归一化选项并改变噪声的部分数据窗 “partial data window”。关于 pdw 的详细信息参考 CUT 命令
- TO USERn : 定义用于储存结果的头段变量 USERn，其中 n 取 0 到 9

缺省值

```
rms noise off to user0
```

说明

这个命令计算当期测量时间窗（参见 MTW）中的数据的均方根。结果写入一个浮点型头段变量 USERn 中，，如果定义了一个噪声时间窗结果可以用于纠正噪声。计算的一般形式是：对信号窗做一次求和并对可选的噪声窗做另一次求和

例子

为了计算两个头段 T1 和 T2 间的数据的未修正的均方根，并将结果保存在头段 USER4 中：

```
SAC> mtw t1 t2
SAC> rms to user4
```

使用一个 5 秒长的噪声窗（结束于头段值 T3），并就算修正后的均方根：

```
SAC> mtw t1 t2
SAC> rms noise t3 -5.0 0.0
```

头段变量改变

USERn

相关命令

mtw、cut

11.132　rotate

概要

以一个给定的角度选择数据的两个分量

语法

```
ROTATE [TO GCP|TO v|THROUGH v] [NORMAL|REVERSED]
```

输入

- TO GCP : 旋转到大圆弧路径（“great circle path”）。两个分量必须都是水平分量。必须定义台站和事件坐标的头段变量
- TO v : 旋转角度为 v 度，两个分量必须都是水平分量
- THROUGH v : 旋转角度为 v 度，一个分量可以是垂直的
- NORMAL : 输出分量为正极性
- REVERSED : 输出分量为负极性

缺省值

```
ROTATE TO GCP NORMAL
```

说明

这个命令可以将两个分量以一定角度旋转。每对数据分量必须有相同的台站名、事件名、采样率。在 THROUGH 选项中两个分量可简单地以给定的旋转角度旋转数据的两个

分量，其中一个可以是垂直分量，在水平面上的旋转由北起算顺时针旋转。垂直分量的旋转为由上起顺时针旋转。

如果使用 TO 选项，两个分量必须都是水平分量，这意味着 CMPAZ 必须定义 CMPINC 必须是 90 度。旋转完成后，每一对的第一个分量将指向 TO 选项给出的角度，如使用 TO GCP 选项，则这个分量将指向台站 -事件向后方位角加减 180 度的方向。因此，这个分量由事件指向台站，台站和事件坐标头段变量 (STLA,STLO,EVLA，EVLO) 必须定义，以便由这些值出发计算向后方位角 NORMAL 和 REVERSED 选项也可以仅用于水平旋转。如果使用 NORMAL 选项，则第二个分量比第一个分量超前 90 度，如果使用 REVERSED 选项则第二个分量比第一个分量落后 90 度。

例子

以 123.43 度旋转一对水平分量:

```
SAC> read xyz.n xyz.e
SAC> rotate to 123.43
```

以大圆弧路径旋转两个水平分量数据集:

```
SAC> read abc.n abc.e def.n def.e
SAC> rotate to gcp
SAC> w abc.r abc.t def.r def.t
```

上面的例子中如果 BAZ 头段变量为 33 度，则径向分量指向 213 度，切向分量指向 303 度，如果设置反极性，切向分量指向 123 度。

头段变量改变

cmpaz, cmpinc

错误消息

- 1301: 未读入数据文件
- 2001: 命令需要一个偶数个数据文件
- 2004: 旋转没有足够的头段信息（对于 GCP 选项 STLA, STLO, EVLA, EVLO 必须定义）
- 2002: 下面的文件不是一个正交对:
- 2003: 下面的文件不都是水平分量，TO 选项只对水平分量作用

11.133 rq

概要

去除谱文件中的地震 Q 因子

语法

```
RQ [Q v] [R v] [C v]
```

输入

- Q v : 设置质量因子为 v
- R v : 设置距离为 v，单位为 km
- C v : 设置群速度为 v，单位 km/s

缺省值

```
RQ Q 1. R 0. C 1.
```

说明

用于校正振幅的方程如下:

$$AMP_{corrected}(f) = AMP_{uncorrected}(f) * e^{\frac{\pi RF}{QC}}$$

其中 f 为频率，单位为 Hz，R 为距离，单位 km，C 是群速度，单位为 km/s。Q 是一个无量纲质量因子

头段变量改变

depmin, depmax, depmen

错误消息

- 1301: 未读入数据文件
- 1305: 对时间序列非法操作

警告消息

- 1604: 下面的文件是振幅 -相位格式（文件应该是实部 -虚部格式）

限制

各参数目前只能接受常数

11.134　rtrend

概要

去除线性趋势

语法

```
RTREND
```

说明

将数据做最小二乘法拟合成一条直线，然后从数据中减去这条直线所表征的线性趋势。数据不必等间隔输出数据文件列表的最后一个文件的最佳拟合的直线参数写入以 RTR 开头的黑板变量中:

- RTR_SLP 直线斜率
- RTR_SDSLP 斜率的标准差
- RTR_YINT 直线的 y 截距
- RTR_SDYINT y 截距的标准差

- RTR_SDDTA 数据的标准差
- RTR_CORRCF 数据的互相关系数

错误消息

- 1301: 未读入数据文件
- 1307: 对谱文件的非法操作

头段变量改变

depmin, depmax, depmin

11.135 saveimg

概要

将图像显示窗口的中图像保存到多种格式的图形文件中

语法

```
SAVEIMG filename.format
```

输入

- filename : 要保存的图像文件名。
- format : 图像文件格式，支持 PS、PDF、PNG 和 XPM。

说明

这个命令将当前绘图保存到图像文件中, 可用的格式包 Postscript(ps) , Portable Document Format(pdf),Image file(png) 以及 Pixmap file(xpm)。

SAVEIMG 相对于 SGF 文件的好处在于, SGF 文件中的字母和数字是由线段组成的, 而 SAVEIMG 产生的 ps 或 pdf 图像采用 Postscript 特性直接产生字体。对大多数情况，低精度的.png 或.xpn 文件也能满足要求。.png 和.xpm 将拥有当前窗口的横纵比（参见 WINDOW 命令），.pdf 或.ps 文件拥有固定的横纵比 X/Y=11/8.5=1.2941，对这些绘图，如果显示窗口设置为 1.2941 会看起来比较好。

正如.sgf 文件，绘图没有边框。对于.sgf 文件，脚本 sac/bin/sgftoeps.csh 可以产生一个有边框的.eps 文件 (要求程序 Ghostscript 在路径中)。相似的脚本未来会写出来应用于 SAVEIMG。

为了使用 SAVEIMG 保存一个绘图，图像必须是可见的，即通过 PLOT、PLOT1 等命令绘制出来。SAVEIMG 在子程序 SSS 中无法工作，但如果输入 qs 退出子程序，此时图像窗口未关闭，SAVEIMG 此时可用于保存该图像。

例子

将图像保存为 PDF 文件:

```
SAC> read PAS.CI.BHZ.sac
SAC> p1
SAC> saveimg pas.ci.pdf
```

将谱图用多种格式保存:

```
SAC> fg seismo
SAC> spectrogram
SAC> save spectrogram.ps
SAC> save spectrogram.png
SAC> save spectrogram.pdf
```

11.136　setbb

概要

设置暂存块变量值

语法

```
SETBB variable  [APPEND] value [variable [APPEND] value ...]
```

输入

- variable : 暂存块变量名，可以是一个新变量或一个已经有值的变量，变量名最多 32 字符长
- value : 暂存块变量的新值，如果也空格必须用引号括起来
- APPEND : 将值加到变量的旧值之后，如果该选项忽略，则新值将代替旧值

说明

setbb 命令可以给黑板变量赋值，这些值可以通过 GETBB 命令获取，或在命令中直接引用。你可以使用 EVALUATE 对黑板变量做基本算术操作，并将结果保存在新的黑板变量中，你也可以通过 UNSETBB 命令删除一个黑板变量。

例子

同时设置多个黑板变量:

```
SAC> setbb c1 2.45 c2 4.94
```

稍后在命令中使用这些变量:

```
SAC> bandpass corners %c1% %c2%
```

设置包含空格的暂存块变量:

```
SAC> setbb mytitle 'sample filter response'
```

检查以确保值正确:

```
SAC> getbb mytitle
 MYTITLE = Sample filter response
```

相关命令

getbb、evaluate、unsetbb

11.137 setdevice

概要

定义后续绘图时使用的缺省图形设备

语法

```
SETDEVICE name
```

输入

- name 图形设备名

说明

这个命令使你能够定义接下来绘图时使用的默认图形设备。这个命令用在所有绘图之前，并应放在缺省宏文件之中。你可以使用 BEGINDEVICES 指定新的图形设备取代这个命令指定的图形设备。

相关命令

begindevices

11.138 setmacro

概要

定义执行 SAC 宏文件时搜索的一系列目录

语法

```
SETMACRO  [MORE] directory [directory ...]
```

输入

- directory : 放置 SAC 宏文件的地方，可以是相对或绝对路径。

说明

这个命令让你能够定义一系列执行宏文件时搜索的目录，最多可以定义 100 个。当 setmacro 使用 more 选项时，指定的文件会加到已经存在的列表的后面，若没有使用 more 选项，则已经存在的列表被新列表取代。当 macro 命令执行时，SAC 首先搜索当前目录，若没有找到则搜索 setmacro 指定的目录，若依然没有找到则在全局宏目录中寻找。

相关命令

macro

11.139 sgf

概要

控制 SGF 设备选项

语法

```
SGF [PREFIX text] [NUMBER n] [DIRECTORY CURRENT|pathname]
    [SIZE NORMAL|FIXED v|SCALED v] [OVERWRITE ON|OFF]
```

输入

- PREFIX text : 设置图形框架的前缀为 text(最多 24 字符长)
- NUMBER n : 设置下一个框架的序号为 n，如果 n 为 0，则 SAC 搜索 SGF 文件目录并设置框架序数为序列中的下一个值。
- DIRECTORY CURRENT : 将 SGF 文件放在当前目录
- DIRECTORY pathname : 将 SGF 放在指定目录下
- SIZE NORMAL : 产生一个常规大小绘图，常规图形有一个 10*7.5 英寸的视窗（最大绘图区域)。视口的缺省值（视窗中除轴和标签之外的绘图区）为 8*5 英寸。
- SIZE FIXED v : 产生一个在 X 方向视口为 v 英寸长的图形。
- SIZE SCALED v : 产生一个视口在 X 方向上为 v 乘以 X 坐标极限值的图形
- OVERWRITE ON|OFF : 当打开时，文件号不递增，每个新文件将擦除先前的文件，这个在多数绘图命令的 PLOT 选项中特别有用。

缺省值

```
sgf prefix f number 1 directory current size normal
```

说明

这个命令控制图形框架的命名规定和后续 SGF 文件的最后图形尺寸。每一个框架被储存在磁盘上单独的文件中，每一个框架名由 4 部分组成，分别为:

- pathname 目录路径名
- prefix 框架前缀
- number 三位数的图形号
- .sgf 用于表示 SAC 图形文件的后缀

默认图形文件的后缀是简单的字母 “f”，框架号为 1，文件放置在当前目录 (即第一个文件名为 “f001.sgf”)。你可以改变前缀以标识你想要保存的文件集。你也可以指定一个目录去储存这些文件。这在你运行 SAC 时改变目录但却想将所有框架文件放在一个地方时非常有用。每个框架被创建时，框架号递增。你可以强制框架号从一个给定的值开始。果你为准备一个报告需要工作几天时间，而同时又希望所有图形都按一个统一的顺序排列，那么令框架号不从 1 开始便很有用了。

有多个选项可以控制绘图的尺寸，常规的绘图为 10*7.5 英寸的视口范围，使用默认的视口的结果是产生一个近似为 8*5 英寸的图形区。你可强制视口的 x 方向为固定长度或将视口的 x 方向与整个坐标范围成比例。这个尺寸的信息写入 SGF 文件。把 SGF 文件转换到一个特定输出设备上产生正确尺寸的图形编码是程序中要做的事。

SGFTOPS 命令可以完成这个转换，尽管大于单页的图形需要适当的后期处理

例子

定义一个不是你当前目录的目录并重置图形框架号为序列中的下一个值:

```
SAC> sgf directory /mydir/sgfstore frame 0
```

设置视口 x 方向绘图尺寸为 3 英寸:

```
SAC> sgf size fixed 3.0
```

创建一个大小相当于贴在墙上的海报的图形

```
SAC> sgf size fixed 30.0
```

设置视口 x 方向的绘图尺寸为每 10 秒地震数据 1 英寸长:

```
SAC> sgf size scaled 0.1
```

在最后的例子中，持续 60 秒的数据图形将有 6 英寸长，而持续 600 秒的数据将有 60 英寸长，并且需要特殊的后期处理。

相关命令

begindevices

11.140 smooth

概要

对数据应用算术平滑算法

语法

```
SMOOTH [MEAN|MEDIAN] [HALFWIDTH n]
```

输入

- MEAN : 应用均值平均算法
- MEDIAN : 应用中值平滑算法
- HALFWIDTH n : 设置移动窗的半宽为 n。滑动窗将包含要平滑的数据点的前面和后面各 n 点

缺省值

```
smooth mean halfwidth 1
```

说明

这个命令对每个数据点应用算术平滑算法。算法类型和围绕每个数据点的滑动窗尺寸是可以变化的。每个窗的尺寸由指定其半宽度来定义，使其滑动窗以每个数据点为中心，并使窗的长度为奇数个点，这样使算法更加简单易行，且不会引起歧义。

头段变量改变

depmin, depmax,depmen

11.141 sonogram

概要

计算一个频谱图，其等价于同一个谱图的两个不同的平滑版本的差

语法

```
SONOGRAM [WINDOW v] [SLICE v] [ORDER n] [CBAR ON|OFF]
    [YMIN v] [YMAX v] [FMIN v] [FMAX v] [BINARY|FULL]
    [METHOD PDS|MEM|MLM] [COLOR|GRAY] [PRINT pname]
```

输入

- WINDOW v : 设置滑动数据窗的长度为 v 秒，这个窗长决定了 FFT 的尺寸
- SLICE v : 设置数据滑动间隔为 v 秒，对每个滑动间隔将产生一个频谱图线
- ORDER n : 指定用于计算谱估计的自相关中的点数
- CBAR ON|OFF : 打开/关闭参考颜色条
- BINARY|FULL : 产生一个双色或彩色图像
- YMIN v : 指定绘图的最小频率
- YMAX v : 指定绘图最大频率
- FMIN v : 指定每次滑动频谱图被平滑的最小带宽
- FMAX v : 指定每次滑动频谱图被平滑的最大带宽
- METHOD PDS|MEM|MLM : 指定使用的谱估计方法，PDS 代表功率密度谱估计，MLM 代表最大似然，MEM 代表最大熵谱估计。
- COLOR|GRAY : 指定是彩色图还是灰度图
- PRINT pname : 打印结果到打印机

缺省值

```
sonogram window 2 slice 1 method mem order 100 ymin 0 ymax
    fnyquist fmin 2.0 fmax 6.0 full color
```

说明

sonogram 命令计算了一个谱图，其等效于同一谱图两种不同平滑版本的差。依赖于平滑参数 fmin 和 fmax 的选择，结果谱图可以加强小幅度谱特性，而其在传统谱图中很难观测到。这个在矿井爆炸的地震信号中寻找高频谱时很有用。(c.f., Hedlin, 1990,Wuster,1993).

限制

图形在频率方向的尺寸为 512.

问题

目前在头段检查以确定他们有相同的分量且在时间上连续方面还有些错误。

头段变量

需要：delta

修改：npts, delta, b, e, iftype, depmin, depmax, depmen

创建：nxsize, xminimum, xmaximum, ,break nysize, yminimum, ymaximum

11.142　sort

概要

根据头段值对内存中的文件进行排序

语法

```
SORT header [ASCEND|DESCEND] [header [ASCEND|DESCEND]...]
```

输入

- header : 依据该头段排序文件
- ASCEND : 按头段值升序排列
- DESCEND : 按头段值降序排列

说明

根据给出的头段值对内存中的文件进行排序。头段变量在命令行中出现的越早，这个变量字段就具有越高的优先权，后面的变量字段用于解决无法根据第一个变量字段排序的情况。最多可以输入 5 个头段变量。

每个都可以跟着 ASCEND 或 DESCEND 来表明那个特定字段的排序方式。如果未指定，则默认为升序排列。如果使用 sort 命令但未指定任何头段值，它将根据上一次执行 sort 命令时的头段去排序，如果第一次调用 sort 但没有给出头段，则会产生错误 1379.

缺省值

默认所有的字段都是升序排列

错误消息

- 301: 内存溢出
- 1379: 未给出 SORT 参数
- 1380: SORT 参数过多
- 1381: 不是一个有效的 SORT 参数
- 1383: SORT 失败

11.143 spectrogram

概要

使用内存中的所有数据计算频谱图

语法

```
SPECTROGRAM [WINDOW v] [SLICE v] [ORDER n] [CBAR ON|OFF]
    [SQRT|NLOG|LOG10|NOSCALING]
    [YMIN v] [YMAX v] [METHOD PDS|MEM|MLM] [COLOR|GRAY]
    [PRINT pname]
```

输入

- WINDOW v : 设置滑动数据窗的长度为 v 秒，这个窗长决定了 FFT 的尺寸
- SLICE v : 设置数据滑动间隔为 v 秒，对每个滑动间隔将产生一个频谱图线
- ORDER n : 指定用于计算谱估计的自相关中的点数
- CBAR ON|OFF : 打开/关闭参考颜色条
- SQRT|NLOG|LOG10|NOSCALING: 指定振幅的自然对数、以 10 为底的对数或平方根
- YMIN v : 指定绘图的最小频率

- YMAX v : 指定绘图最大频率
- METHOD PDS|MEM|MLM : 指定使用的谱估计方法，PDS 代表功率密度谱估计，MLM 代表最大似然，MEM 代表最大熵谱估计。
- COLOR|GRAY : 指定是彩色图还是灰度图
- PRINT pname : 打印结果到打印机

缺省值

```
spectrogram window 2 slice 1 method mem order 100 noscaling
    ymin 0 ymax fnyquist color
```

说明

频谱图是通过计算连续并可能重叠的数据时间窗的功率谱并将谱沿着时间轴绘制产生的。谱是由一个使用 MLM 或 MEM 或 PDS 得到的被截断的自相关函数。一般会选择高精度的 MLM 和 MEM 方法，因为他们提高了精度且不会产生由于不同频率之间能量泄漏导致的人工干扰。这些技术的介绍可以参考 Kanasewich(1981) 以及 Lacoss(1971)。被截断的互相关函数的长度由 ORDER 参数决定。为了保持和 spe 子程序的一致性，设置了 PDS 的默认 order 为 200，MEM 和 MLM 的 order 默认为 100。在 SAC 中每个数据窗的长度由 window 参数决定。沿着频谱图时间轴的谱之间的间隔由 slice 参数决定。这两个参数的不同决定了临近时间窗的重叠量，如下图所示:

```
Time ->
0  1  2  3  4  5  6  7  8  9 10 11
|.....|.....|.....|.....|.....|.....|.....|.....|.....|.....|..|
|__^__| window 1, First time will be at the center of this window.
      |__^__| window 2
            |__^__| window 3

|.....| Slice: 临近时间窗的开始时间的差
```

频谱图时间轴的起始和结束点依赖于要被分析的时间序列的长度以及 window、slice 参数。频谱图的起始时间是时间序列起始时间迟半个窗对应的时间，因为它被定义为第一个窗的中间时刻。SAC 不会对数据的前面补 0.

限制

图形在频率方向的尺寸为 512.

问题

目前在头段检查以确定他们有相同的分量且在时间上连续方面还有些错误。

头段变量改变

需要：delta

改变：npts, delta, b, e, iftype, depmin, depmax, depmen

创建：nxsize, xminimum, xmaximum, ,break nysize, yminimum, ymaximum

11.144 sqr

概要

对每个数据点做平方

语法

```
SQR
```

错误消息

- 1301: 未读入文件
- 1307: 对谱文件非法操作

头段变量改变

depmin, depmax, depmen

11.145 sqrt

概要

对每个数据点做平方根

语法

```
SQRT
```

错误消息

- 1301: 未读入文件
- 1307: 对谱文件非法操作

头段变量改变

depmin, depmax, depmen

11.146 stretch

概要

拉伸 (内插) 数据，并加上一个可选的 FIR 滤波器

语法

```
STRETCH  n [FILTER ON|OFF]
```

输入

- n : 设置增采样因子，必须在 2 和 7 之间
- FILTER ON|OFF: 打开/关闭内插 FIR 滤波器选项

缺省值

```
stretch 2 filter on
```

说明

通过使用内插 FIR 滤波器，这个命令可以用于创建一个更小的采样间隔（更多数据点）但看起来与原始文件相似的文件。在使用这个命令时要注意，因为滤波器对频谱成分是有影响的，当关闭这个滤波器选项时，在原始数据点之间，将简单地插入适当数目的零值。

头段变量改变

npts, delta, e, depmin, depmax, depmen

11.147　sub

概要

给每一个数据点减去一个常数

语法

```
SUB  [v1 [v2 ... vn]]
```

输入

- v1 : 乘到第一个文件的常数
- v2 : 乘到第二个文件的常数
- vn : 乘到第 n 个文件的常数

缺省值

```
sub 0.0
```

说明

这个命令将给内存中的每一个数据文件的每个元素减一个常数。这个常数对于每一个数据文件来说可以是相同的也可以是不同的。如果内存中数据文件的数目比命令给出的常数要多，那么最后命令中的最后一个常数将用于剩下的所有文件。如果内存中数据文件数目比给出的常数少，则多余的常数将被忽略。

错误消息

- 1301: 未读入数据文件
- 1307: 对谱文件的非法操作

头段变量改变

depmin, depmax, depmen

11.148　subf

概要

将一组文件中的数据分别减去内存中不同文件的数据

语法

```
SUBF [NEWHDR ON|OFF] filelist
```

输入

- NEWHDR ON|OFF : 操作得到的结果文件默认从内存中的原始文件获取头段值，如果 NEWHDR ON，则结果文件的头段将从 filelist 中的新文件获取
- filelist : SAC 二进制文件列表。这个列表可以包含简单的文件名，绝对或相对路径以及通配符。详细信息参见 READ 命令。

说明

这个命令用于将一个文件减去一群文件中或者将一群文件减去另一群文件中。下面针对每种情况各给出一个例子。为了使得文件与文件之间能够相减，必须保证这些文件是等间隔的而且需要有相同的采样间隔和数据点数。后面两个限制可以通过使用 BINOPERR 命令得到削弱。如果内存中的文件数多于 filelsit 中文件数，则 filelist 的最后一个文件将加到余下的全部文件中。如果 NEWHDR OFF(默认情况) 内存中的头段大部分不会被改变。如果 NEWHDR ON，内存中的头段将被 filelist 文件的头段取代。

例子

将一个文件减去其他三个文件:

```
SAC> read file1 file2 file3
SAC> addf file4
```

将两个文件减去另外两个文件:

```
SAC> read file1 file2
SAC> addf file3 file4
```

头段变量改变

depmin, depmax, depmen

错误消息

- 1301: 未读入文件
- 1803: 未读入二进制文件
- 1306: 对不等间隔文件的非法操作
- 1307: 对谱文件的非法操作
- 1317: 文件不是 SAC 数据文件
- 1801: 头段值不匹配（采样间隔或数据点数不相等）

警告消息

- 1802: 时间重叠 * 什么意思？如何犯错？ *

相关命令

read、binoperr

11.149　symbol

概要

控制符号绘图属性

语法

```
SYMBOL [ON|OFF|n] [SIZE v] [SPACING v] [INCREMENT ON|OFF]
    [LIST STANDARD|nlist]
```

输入

- ON : 打开符号绘图开关，不改变符号号
- OFF : 关闭符号绘图开关
- n : 打开符号绘图开关。将符号号设置为 n。目前有 16 个不同的符号，符号 0 意味着关闭符号开关
- SIZE v : 设置符号尺寸为 v，值为 0.01 意味着占据整个绘图尺寸的 1%
- SPACING v : 设置符号间隔为 v，这是符号之间的最小间隔，如果你想每个数据点都有符号，则其值为 0，对注释行使用 0.2 到 0.4
- INCREMENT ON : 对每个数据文件操作完成后按符号列表中先后顺序改变为另一个符号。
- INCREMENT OFF : 关闭上述 INCREMENT 选项
- LIST nlist : 改变符号表的内容。输入符号号码表。设置表中的第一个符号码，并打开符号绘图开关
- LIST STANDARD : 改变到标准符号表。设置表中的第一个符号表，并打开符号绘制选项。

缺省值

```
symbol off size 0.01 spacing 0. increment off list standard
```

说明

这些符号属性独立于由 LINE 命令定义的画线属性。打开画线选项，它们也可以用于注释在相同图形上的不同的线。关闭画线选项，则可以绘制散点图。如果你要将几个数据文件画在同一张图上，也许需要使用不同的符号。这是可以使用 INCREMENT 选项。当这个选项打开时，每次绘制数据文件，都从符号表中将原来的符号码增加 1，缺省符号表包含符号表从 2 到 16 的符号，你也可以使用 LIST 选项改变这个表的次序和内容。如果你在绘制一系列重叠绘图，并需要经相同符号用在相同次序的每个图形上，这样做很有用。符号码为 0 相当于关闭符号绘制选项。这个选项用于 LIST 选项和 LINE 选项，以在一张图上用线表示一些数据，用符号表示另外一些数据。

例子

为了创建一个散点分布图，关闭画线选项，选择适当的符号，然后绘图:

```
SAC> line off
SAC> symbol 5
SAC> plot
```

为了用符号 7、4、6、8 注释四条实线，间隔用 0.3，用 PLOT2 绘图:

```
SAC> line solid
SAC> sym spacing .3 increment list 7 4 6 8
SAC> r file1 file2 file3 file4
SAC> plot2
```

使用 PLOT2 在相同图形上绘制三个文件，第一个文件图形使用实线无符号；第二个没有线，为三角符号；第三个没有线，带有交叉符号:

```
SAC> read file1 file2 file3
SAC> line list 1 0 0 increment
SAC> symbol list 0 3 7 increment
SAC> plot2
```

相关命令

line

11.150 synchronize

概要

同步内存中所有文件的参考时间

语法

```
SYNCHRONIZE [ROUND ON|OFF] [BEGIN ON|OFF]
```

输入

- ROUND ON : 打开开始时间取整的选项。当这个选项打开时，每个文件的开始时间都将以最近采样间隔的倍数取整
- ROUND OFF : 关闭开始时间取整的选项
- BEGIN ON : 设置每个文件的开始时间为 0
- BEGIN OFF : 保持参考时间的 GMT 值

缺省值

```
synchronize round off begin off
```

说明

这个命令同步内存中所有文件的参考时间。它通过检查所有文件的参考时间和文件起始的偏移时间决定最晚的开始时间，并取这和最晚的开始时间为内存中所有文件的参考时间。然后计算每个文件中的所有相对时间 (B, E, A, Tn 等) 相对于新参考时间的值。当一系列文件具有不同的参考时间，而你想要用 CUT 或 XLIM 命令分析或绘制这些文件中的一部分数据这个命令会很有用。一旦参考时间被同步了，则 CUT 操作将针对严格相同的 GMT 时间窗进行。

如果使用了 BEGIN 选项，参考时间的 GMT 值则不被保留，BEGIN 选项将所有文件的 kztime、kzdate 设置为相同，而且将所有文件的开始时间设置为 0。其他的点保持与文件开始时间的关系。

例子

假定你读取两个不同参考时间的文件到内存:

```
SAC> read file1 file2
SAC> listhdr b kztime kzdate

  FILE: FILE1
  -
  B = 0.
  KZTIME = 10:38:14.000
  KZDATE = MAR 29 (088), 1981

  FILE: FILE2
  -
  B = 10.00
  KZTIME = 10:40:10.000
  KZDATE = MAR 29 (088), 1981
```

这些文件有相同的参考日期，不同的参考时间以及不同的开始时间偏移量。你可以执行 SYNCHRONIZE 然后使用 LISTHDR，你会发现:

```
SAC> synchronize
SAC> listhdr

  FILE: FILE1
  -
  B = -126.00
  KZTIME = 10:40:20.000
  KZDATE = MAR 29 (088), 1981

  FILE: FILE2
  -
  B = 0.
  KZTIME = 10:40:20.000
  KZDATE = MAR 29 (088), 1981
```

现在内存中的所有文件有相同的参考时间，如果头段中有任何已定义的时间标记，它们的值也会调整以使得其 GMT 值是不变的。

头段变量

nzyear, nzjday, nzhour, nzmin, nzsec, nzmsec, b, e, a, o, tn.

11.151　systemcommand

概要

从 SAC 中执行系统命令

语法

```
SYSTEMCOMMAND command [ options ]
```

输入

- command : 系统命令名
- options : 命令需要的选项

说明

这个命令允许你在执行 SAC 的同时执行系统命令

11.152 taper

概要

在数据两端应用对称的 taper 函数，使得数据在两端平滑地衰减到零

语法

```
TAPER [TYPE HANNING|HAMMING|COSINE] [WIDTH v]
```

输入

- TYPE HANNING|HAMMIN|COSINE : 应用 Hanning、Hamming、余弦衰减窗
- WIDTH v : 设置位于数据首末两端的衰减窗的宽度为 v，这个值在 0.0 和 0.5 之间

缺省值

```
taper type hanning width 0.05
```

说明

taper 函数是在 0 和 1 之间取值的单调变化的函数。它对称地施加于数据的首尾两端。对于数据的首端，taper 函数的第一个点[1]值为 0，第 NPTS*WIDTH 个点的值为 1 （NPTS 为数据总点数，WIDTH 为命令中设置的参数[2]）。将 taper 函数与数据相乘即使得数据的首端从 0 开始不断光滑地增加到其原始值。数据末端完全类似，只是数据由其原始值不断光滑地减小到 0。正如 taper 的英译，“逐渐消失”，有时候也翻译为 “尖灭”。

taper 命令的本质如下式所示：

$$Data(j) = \begin{cases} Data(j) * Taper(j) & \text{j} \in \text{[1, npts*width]} \\ Data(j) * Taper(npts - j) & \text{j} \in \text{[npts-npts*width, npts]} \end{cases}$$

其中 taper 函数可以表示为:

$$Taper(j) = F_0 - F_1 \cos(\omega(j-1)) \quad j \in [1,\ npts * width]$$

下表定义了不同的衰减的各种参数。在这个表中，N 为首尾两端的衰减数据的长度，即 NPTS*WIDTH。

[1]究竟是第 0 个点还是第 1 个点，这个问题比较纠结，当初从 FORTRAN 改写成 SAC 的时候，遗留下来一些历史问题。

[2]这下知道为什么 WIDTH 只能取 0.0-0.5 了吧。

类型	ω	F_0	F_1
HANNING	$\frac{\pi}{N}$	0.50	0.50
HAMMING	$\frac{\pi}{N}$	0.54	0.46
COSINE	$\frac{\pi}{2N}$	1.00	1.00

错误消息

- 1301: 未读入数据文件
- 1306: 对不等间隔文件的非法操作

头段变量改变

depmin, depmax, depmen

11.153 ticks

概要

控制绘图上时间标记的位置

语法

```
TICKS [ON|OFF|ONLY] [ALL|TOP|BOTTOM|RIGHT|LEFT]
```

输入

- ON : 在所列出的边上打开时标，其他不变
- OFF : 在所列出的边上关闭时标，其他不变
- ONLY : 仅在列出的边上打开时标，其他关闭
- ALL : 所有四边都有时标
- TOP : 时标位于视口上部的 X 轴
- BOTTOM : 时标位于视口的下部上的 X 轴
- RIGHT : 时标位于视口右边的 Y 轴
- LEFT : 时标位于视口左边的 Y 轴

缺省值

```
TICKS ON ALL
```

说明

时标可以画在图形四边的一边或几边上。他们以刻度间隔的当前值绘出，其刻度间隔由 XDIV 命令控制。时标自动画在由 AXES 命令指定的轴上。

例子

打开顶端时标，其他不变:

```
SAC> ticks on top
```

关闭所有时标:

```
SAC> ticks off all
```

打开底部时标，其余关闭:

```
SAC> ticks only bottom
```

相关命令

xdiv、axes

11.154 title

概要

定义绘图的标题和属性

语法

```
TITLE [ON|OFF|text] [LOCATION TOP|BOTTOM|RIGHT|LEFT]
    [SIZE TINY|SMALL|MEDIUM|LARGE]
```

输入

- ON : 打开标题选项，不改变标题内容
- OFF : 关闭标题选项
- text : 打开标题选项，改变标题内容，如果文本中包含空格，则标题需要用单引号括起来
- LOCATION location : 改变标题的位置 TOP|BOTTOM|RIGHT|LEFT
- SIZE size : 改变标题文本尺寸 TINY|SMALL|MEDIUM|LARGE

缺省值

```
title off location top size small
```

说明

如果这个选项打开，则在每个图形上都给出标题，标题的尺寸和位置及内容均可改变，文本质量和字体可以通过 GTEXT 命令设置。

相关命令

gtext

11.155 trace

概要

跟踪黑板变量和头段变量

语法

```
TRACE [ON|OFF] name [name ...]
```

输入

- ON : 打开后随变量的跟踪
- OFF : 关闭后随变量的跟踪
- name : 要跟踪的暂存块变量或头段变量名。如果是一个头段，则其形式为 filename,hdrname。这里 filename 是 SAC 数据文件名或文件号，hdrname 是 SAC 头段变量名

缺省值

```
trace on
```

说明

这个命令用于在 SAC 执行过程中跟踪 SAC 黑板变量或头段变量的值。其对于调试长的或复杂的宏文件很有用。对一个变量进行跟踪时，将显示其当前值。跟踪选项打开时，每次命令执行后都对变量值进行检查。每次黑板变量或头段变量改变时都将显示一行的信息。跟踪选项关闭时，也最后一次显示它的当前值。

例子

跟踪黑板变量 TEMP1 和属于文件 MYFILE 的头段变量 T0:

```
SAC> trace on temp1 myfile,t0
  TRACE  (on) TEMP1 = 1.45623
  TRACE  (on) MYFILE,T0 = UNDEFINED
```

执行命令时，不论是在终端键入命令还是执行一个宏文件，SAC 都将检查新的变量值和内存中的变量值，并且不论哪个值改变，都将有关的信息显示出来。假设在完成一些计算之后改变了 TEMP1，并定义了 T0 的值，则 SAC 将显示信息:

```
  TRACE (mod) TEMP1 = 2.34293
  TRACE (mod) MYFILE,T0 = 10.3451
```

稍后的处理中 TEMP1 可能再次改变:

```
  TRACE (mod) TEMP1 = 1.93242
```

当跟踪选项关闭时，SAC 最后一次显示当前值:

```
SAC> trace off temp1 myfile,t0
  TRACE (off) TEMP1 = 1.93242
  TRACE (off) MYFILE,T0 = 10.3451
```

11.156 transcript

概要

控制输出副本文件

语法

```
TRANSCRIPT [OPEN|CREATE|CLOSE|CHANGE|WRITE|HISTORY] [FILE filename]
    [CONTENTS ALL|ERRORS|WARNINGS|OUTPUT|COMMANDS|MACROS|PROCESSED]
    [MESSAGE text]
```

输入

- OPEN : 打开副本文件，并在已存在的文件底部添加副本
- CREATE : 创建一个新的副本文件
- CLOSE : 关闭一个已经打开的副本文件
- CHANGE : 改变一个已经打开的副本文件的内容
- WRITE : 写信息到一个副本文件，不改变其状态或内容
- HISTORY FILE filename : 将命令行历史保存到文件
- FILE filename : 定义副本文件的名字
- MESSAGE text : 向副本文件中写入 text 给出的信息。这个信息可以用于指定正在进行的进程或指定正在处理的不同事件，在两次执行这个命令的过程之中这个信息不保存。
- CONTENTS ALL : 定义副本文件的内容为全部输入输出的信息。
- CONTENTS list : 定义副本文件的内容，即包含在文件中的输入输出的类型表

其中 list 可以取 :

- ERRORS : 执行命令期间产生的错误消息
- WARNINGS : 执行命令期间产生的警告消息
- OUTPUT : 执行命令期间的输出消息
- COMMANDS : 终端输出的原始命令
- MACROS : 宏文件中出现的原始命令
- PROCESSED : 经终端或宏处理之后的命令。处理之后的命令其所有宏参数、黑板变量、头段变量、内联函数都会被计算并带入原始命令中形成新的命令。

缺省值

```
transcript open file transcript contents all
```

说明

副本文件用于记录 SAC 执行的结果。其可以是一个全部结果或部分结果的副本，可以包含一个或多个操作的执行结果，你可以一次最多拥有 5 个活动的副本文件，每个文件用于跟踪不同的方面。一个用处是记录在终端输入的命令然后用于一个宏文件中，如下例所示。

例子

为了创建一个新的副本文件，文件名为 MYTRAN，包含了除已处理命令之外的其他全部类型:

```
SAC> transcript create file mytran cont err warn out com macros
```

如果之后不想把宏命令送入这个文件，你可以使用 CHANGE 选项:

```
SAC> transcript change file mytran contents errors warnings output commands
```

为了定义一个名为 MYRECORD 的副本文件，其记录了终端输入的命令:

```
SAC> transcript create file myrecord contents commands
```

以后，经过适当的编辑，这个文件可以用作宏命令文件，去自动执行相同的一组命令。在最后的例子中假设你需要彻夜处理许多事件。你可以设置每个事件一个副本文件（用不同的副本文件名）去记录处理的结果。另外你可以将处理所有事件得到的任何错误消息保存到一个副本文件中:

```
SAC> transcript open file errortran contents errors
SAC> transcript write message 'processing event 1'
```

这些命令将放在处理每个事件的宏文件中，它假设事件名作为第一个参数带入宏。使用打开选项，运行信息和出错信息将天骄到文件的后面，第二天检查一下这个出错信息副本文件，就可以快速查阅在处理期间是否出现了错误。

为了将保存一个命令行副本，以记录 SAC 当前和将来要运行的命令:

```
SCA> transcript history file .sachist
```

这就在当前目录创建并写入了一个副本文件 "./.sachist"。任何储存在那里的文件将被载入命令历史中。如果这个命令位于你的启动初始化宏文件中，则每次你运行 SAC 时将在当前目录产生一个单独的命令行历史。在一个新执行的 SAC 中，上下键将浏览完整的命令历史，你可以修改以前输入的命令并再次执行它，如果你没有在 SAC 内或初始化宏文件中输入这个命令，则命令行历史将自动保存到 `~/.sac_history`

11.157　transfer

概要

反卷积以去除仪器响应并卷积以加入其它仪器响应。

语法

```
TRANSFER [FROM type [options]] [TO type [options]]
    [FREQLIMITS f1 f2 f3 f4] [PREWHITENING ON|OFF|n]
```

输入

- FROM type : 通过谱域除法做反卷积要去除的仪器类型。
- TO type : 通过谱域乘法做卷积来加入的仪器类型。
- FREQLIMITS f1 f2 f3 f4：设定频率段，以处理超高频和超低频信号。
- PREWHITENING ON|OFF|n：预白化处理。

缺省值

```
trans from none to none
```

说明

关于仪器响应以及 transfer 命令，参考 “仪器响应” 一节。

在 TRANSFTER 中默认仪器的输入输出都是位移，在 SAC 中定义为 NONE。因而如果 FROM 类型或者 TO 类型未指定，SAC 会假定其为 NONE。

若输出仪器是 NONE，则 SAC 头段中的 IDEP 将设置为 DISPLACEMENT(单位为 nm). 如果 TRANSFER 选择为 TO VEL 或 TO ACC，SAC 中的头段变量 IDEP 将根据内存中的波形而改变。

如果 TO 指定为除 NONE、VEL、ACC 外的其他类型，内存中的波形将被卷积上相应的仪器类型。如果 FROM 仪器类型是 NONE，则不会去除仪器响应，原始的地震道数据认定为位移，这常用于向合成地震图中加入仪器响应。

在同一个 SAC 会话中，当第二次调用 TRANSFER 时一定要特别小心，因为第二次调用 TRANSFER 时会使用第一次调用时的参数，除非显式指定其参数。

所有的地震仪在 0 频率处都具有零响应。当做反卷积但不卷积其他响应的时候 (e.g. “TO NONE”)，就有必要修改超低频率处的响应以防止频率域除 0。在高频区域，信噪比通常很低，因而有必要对其响应进行抑制。FREQLIMITS 即可满足此要求。FREQLIMITS 包含了低通和高通的尖灭器（taper)。四个频率满足 f1<f2<f3<f4，尖灭器在 f2 和 f3 间为 1，在 f1 以下和 f4 以上为 0。频率 f1 和 f2 确定了高通滤波器的特性，频率 f3 和 f4 指定了低通滤波器的特性。f1 与 f2 之间以及 f3 与 f4 之间分别为余弦波的四分之一个周期。

如果你想要一个低通滤波器但在低频处不滤波，一种方法是设置 f1=-2 和 f2=-1；如果你想要一个高通滤波器但在高频处不滤波，对于 Nyquist 频率为 0.5，设置 f3=10 和 f4=20。

注意：由于滤波器还有零相位，因而其是非因果。如果数据点数 npts 不为 2 的指数幂次，会导致在频段 (f1,f4) 之外振幅不为 0。如果想要数据点数为 2 的幂次方个，可以参考 SAC 中的 CUT 命令。

缺省值中是没有 FREQLIMITS 选项的，在使用 FREQLIMITSEVALRESP 和 POLEZERO 选项时，应该加上 FREQLIMITS 选项以处理超高频和超低频。

PREWHITENING 控制数据的预白化：

- PREWHITENING: 预白化可以将输入时间序列在变换到频率域之前，进行谱的平化。这会减小谱值的动态范围，并提高数据在高频的计算精度。默认 prewhitening 是关闭的。参见 WHITEN 命令。
- PREWHITENING ON : 在谱操作之前在频率域进行谱白化，并在谱操作后在时间域做谱白化的补偿。
- PREWHITENING OFF : 关闭预白化。
- PREWHITENING n : 打开预白化选项并设置其阶数。如果打开预白化但不指定阶数，则缺省 n=6, 除非在 WHITEN 命令中已经修改了预白化阶数。

例子

为了将 NYKM.Z 去除台站 RSTN 的仪器响应，并应用 DSS 的仪器响应：

```
SAC> read nykm.z
SAC> trans from rstn subtype nykm.z to dss prew off
```

为了去除 LLL 宽频带仪器响应，应用 SRO 仪器响应，并对频带做尖灭及预白化：

```
SAC> read abc.z
SAC> trans from lll to sro freq .02 .05 1. 2. prew 2
```

得到的地震数据的通带，在 0.05Hz 到 1Hz 是平的，在 0.02Hz 以下及 2Hz 以上为 0。

在反卷积前在时间域做二阶预白化，在卷积之后该效应被移除。

为了将电磁仪器响应转换为位移：

```
SAC>  read xyz.z
SAC>  transfer from elmag freep 15. mag 750. to none
```

11.158 traveltime

概要

根据预定义的速度模型计算指定震相的走时。

语法

```
TRAVELTIME [MODEL string] [PICKS number] [PHASE phase list]
    [VERBOSE|QUIET] [M|KM]
```

输入

- MODEL : iasp91 或 ak135，缺省为 iasp91
- Picks : 如果 number 值位于 0 到 9，则第一个震相的到时将储存在头段 Tn 中。如果命令行选项中没有包含 PICKS，VERBOSE 就会打开并显示震相到时而不写入头段。
- PHASE : 要 pick 或显示的震相列表，如果有 PICKS n 选项，则则震相到时以及其标签将写入到头段 Tn 和 KTn 中。
- VERBOSE|QUIET : 若使用 VERBOSE，则震相走时将以相对发震时刻 (O) 和文件起始时间 (B) 两种方式显示；若使用 QUIET，则不在屏幕上显示到时，若二者都没有显示，则显示 TRAVELTIME 命令使用的深度
- M|KM : EVDP 的单位为 m 或者 km。

缺省值

```
MODEL iasp91 KM PHASE P S Pn Pg Sn Sg
```

说明

该命令使用iaspei-tau 程序计算走时，要求内存中的波形文件的事件和台站位置以及发震时刻必须定义。

内存中所有文件拾取的震相的到时存储在头段变量 Tn 中，其中 n 的范围为 0 到 9。计算的走时是相对于发震时刻 (O) 的，但存储在 Tn 中的时间是相对于文件起始时间 (B)。

走时表根据地震深度及震中距获得走时，震中距 (GCARC) 使用球三角几何，根据事件和台站经纬度计算。

由于历史原因，事件深度 EVDP 以前的单位为 m，RDSEED 产生的 SAC 波形数据单位为 m。

在 v101.5 中，默认的 EVDP 单位为 km，但由于很多波形数据 EVDP 仍然为 m，这里引入这个命令选项以指定深度单位。

在 SAC2000 中，命令 TRAVELTIME 位于子程序 SSS 中，当前版本 (V101.5) 没有 SAC2000 版本的全部功能，但可以不用进入 SSS 而直接运行。

例子

区域事件，使用默认震相：

```
SAC> fg seismo
SAC> traveltime
traveltime: depth: 15.000000
traveltime: error finding phase P
traveltime: error finding phase S
traveltime: setting phase Pn       at 10.464321 s [ t = 51.894321 s ]
traveltime: setting phase Pg       at 22.904724 s [ t = 64.334724 s ]
traveltime: setting phase Sn       at 50.047722 s [ t = 91.477722 s ]
traveltime: setting phase Sg      at 66.414337 s [ t = 107.844337 s ]
```

对于区域事件，初至波为 Pn 或 Pg，因而这里没有 P 波到达，对上面的例子，拾取不会写入头段中。

```
SAC> fg seismo
SAC>  traveltime picks 0 phase Pn Pg Sn Sg
traveltime: depth: 15.000000
SAC> lh AMARKER T0MARKER T1MARKER T2MARKER T3MARKER
AMARKER = 10.464
T0MARKER = 10.464           (Pn)
T1MARKER = 22.905           (Pg)
T2MARKER = 50.048           (Sn)
T3MARKER = 66.414           (Sg)
SAC> write seismo-picks.z
```

可以看到已经将 A 定义为 Pn 的到时，文件 seismo-picks.z 将有 T0 到 T3 四个头段，命令 PLOT1 可以看到有各震相在相应到时处有标签名。注意尽管 VERBOSE 没有开启，深度 (单位 km) 还是会被打印出来。这是为了保证用户对 EVDP 的单位有正确的了解，可以通过 QUIET 选项取消深度的显示。

下面例子给出的波形文件是由 RDSEED(V5.0) 产生的，EVDP 单位为 m：

```
SAC> r 2008.052.14.16.03.0000.XC.OR075.00.LHZ.M.SAC
SAC> lh evdp
evdp = 6.700000e+03
SAC> traveltime M picks 0
traveltime: depth: 6.700000 km
SAC> lh t0marker t1marker t2marker t3marker
t0marker = 61.48            (Pn)
t1marker = 76.413           (Pg)
t2marker = 109.66           (Sn)
t3marker = 132.11           (Sg)
SAC> ch evdp (0.001 * :1,evdp:)
SAC> setbb station :1,KSTNM:
SAC> write %station%.z
```

保存的文件 OR075.z 单位为 km，并且各个震相 Pn、Pg、Sn 及 Sg 都有注释。震相名是大小敏感的，具体参见 iaspei-tau 的文档。

11.159 tsize

概要

控制文本尺寸属性

语法

```
TSIZE [TINY|SMALL|MEDIUM|LARGE v ] [RATIO v] [OLD|NEW]
```

输入

- size v : 改变文本尺寸之一的值为 v
- RATIO v : 改变文本的宽高比为 v
- OLD : 将所有文本尺寸值设置为其旧值，这些旧值用在 SAC 9 之前的版本中
- NEW : 改变所有文本尺寸值为 SAC 初始化时的缺省值。

缺省值

```
tsize ratio 1.0 new
```

说明

大多数的文本注释命令 (TITLE, XLABEL, FILEID 等) 允许你改变要显示的文本的尺寸。你可以从四个已命名的尺寸中选择 (TINY, SMALL, MEDIUM, 和 LARGE.)。每一个命名尺寸有一个初始值，如下表所示。其尺寸是一个字符相对整个视窗的高度。有些时候你想要使用一些不同与默认尺寸的注释。TSIZE 允许你重新定义这四个已命令的尺寸。你也可以使用这个命令改变字符的宽 -高比。

表 11.2: SAC 标准文本尺寸

NAME	A	B	C	D	E
TINY	0.015	66	50	68	110
SMALL	0.020	50	37	66	82
MEDIUM	0.030	33	25	44	55
LARGE	0.040	25	18	33	41

上面做各列的定义如下:

- A 自如相对整个视窗的高度
- B 全视窗下文本的行数
- C 正常视窗下文本的行数。正常视窗是指 x 为 0. 到 1.，y 为 0. 到 0.75
- D 正常视窗中，每行的最小字符数
- E 正常视窗中每行字符的平均数

例子

为了改变 MEDIUM 的定义，并使用它创建一个特别尺寸的标题:

```
SAC> tsize medium 0.35
SAC> title 'rayleigh wave spectra' size medium
SAC> plot2
```

为了重置尺寸定义到其缺省值:

```
SAC> tsize new
```

相关命令

title、xlabel、fileid、plotc

11.160 unsetbb

概要

删除暂存块变量

语法

```
UNSETBB ALL|variable ...
```

输入

- ALL : 删除当前定义的全部暂存块变量
- variable : 删除暂存块变量

说明

黑板变量通过 getbb 命令赋值，通过 GETBB 命令获取黑板变量的值，也可直接在命令行中引用黑板变量。unsetbb 命令可以删除全部或部分黑板变量。

例子

一次删除多个黑板变量:

```
SAC> unsetbb c1 c2 x
```

删除所有黑板变量:

```
SAC> unsetbb all
```

相关命令

setbb、evaluate、getbb

11.161 unwrap

概要

计算振幅谱并展开相位谱

语法

```
UNWRAP [FILL ON|OFF|n] [INTTHR v] [PVTHR v]
```

输入

- FILL ON | OFF : 打开/关闭零点填充选项
- FILL n : 打开零点填充选项并将填充值设置为 n
- INTTHR v : 改变积分阀值常量为 v
- PVTHR v : 改变主阀值常量为 v

缺省值

```
UNWRAP FILL OFF INTTHR 1.5 PVTHR 0.5
```

说明

这个命令将内存中的时间序列数据转换为谱数据，其包含了振幅谱和展开相位谱的频谱数据。这个过程作用于具有光滑变换相位的数据。数据在转换之前补 0 以使数目为 2 的幂数倍，你可以使用 FILL 选项指定大量补充零点。

这是 Tribolet 算法的工具程序。其中用两种方法来估算在任何频率的展开相位。一个是通过快速傅氏变换做相位导数的数字积分。如果需要得到一个稳定一致的估算常数，则可将这个梯形积分的步长在每个频率上对分。你可以使用 INTTHR 选项控制这个验算的阀值，此值单位为弧度。减少 INTTHR 将改进相位计算结果，然而这个值太小时，会导致发散。

算法中使用的第二个方法是使用反正切函数先计算相位的主值。展开相位的计算方法是相位主值加上 2*PI 的整数倍，直到相位的突变小于给定的阀值为止。可以使用 PVTHR 选项控制这个验算的阀值。与上一个算法类似，减少这个阀值将改进相位估算的结果，但也增加了无解的可能性。

这两个阀值的初值通常经验地取为:

$$\pi/4 < PVTHR < INTTHR < 2\pi$$

错误消息

- 1301: 未读入数据文件
- 1306: 对不等间隔文件非法操作
- 1606: 超过 DFT 允许的最大数据点数

警告消息

- 1610: 对文件中数据点的相位展开失败（调整阀值然后重试）

头段变量改变

b, e 和 delta 分别改变为变换的起始频率、结束频率和采样频率。原始的 b,e 和 delta 被保存在为 sb、se、sdelta，当进行反变换时将值带回。

限制

目前可以转换的数据最大长度为 4096。

11.162 vspace

概要

改变图形的最大尺寸和形状

语法

```
VSPACE FULL|v
```

输入

- FULL : 使用整个视窗，这是可能的最大屏幕或窗口尺寸
- v : 使视窗比 y:x 为 v，具有这个纵横比的最大的区域称为视窗

缺省值

```
VSPACE FULL
```

说明

视窗代表了屏幕上可以用于绘图的部分。视窗形状和尺寸在不同图形设备之间有很大的变化。

1. 尽管在尺寸上有很大不同，许多图形终端都具有 0.75 的纵横比。
2. SGF 文件的纵横比为 0.75，其大约是标准的 8.5*11 英寸纸张的纵横比。
3. 由 XWINDOWS 或 SUNWINDOWS 图形设备建立的窗口可以有你想要的任意纵横比

这个命令可以控制纵横比，从而使你能够控制图形的形状缺省绘图是在整个视窗上，如果确定了一个纵横比，则视窗就是设备上具有这个纵横比的最大区域。

当你使用 PLOTC 命令在交互设备上建立一张图，并且最终要将它发送到 SGF 设备上，这个命令特别有用，在绘制任何图形之前，必须设置纵横比为 0.75。这将保证图形在 SGF 文件上与在交互设备上相同。如果你要建立一个独立于图形设备的正方形视窗，则可以简单地设置纵横比为 1.0

11.163 wait

概要

通知 SAC 在绘制不同图形的操作之间是否暂停

语法

```
WAIT [ON|OFF|EVERY]
```

输入

- ON : 按常规模式打开等待选项
- OFF : 关闭等待选项
- EVERY : 按每个图形模式打开等待选项

缺省值

```
wait on
```

说明

当你读取了多个数据文件并使用 PLOT 绘制，每个文件将产生一个框架，如果你绘制到终端，正常情况下 SAC 将在每张图后暂停并发送信息 “WAITING” 到终端。然后你可以键入回车看到下一张图，或输入 “GO” 使 SAC 不暂停地绘制剩下的图形，或键入 “KILL” 终止绘制这组文件。SAC 绘制最后一张图之后不再暂停，因为通常的输入提示符提供了相同的功能。当这个选项关闭时，SAC 在不同的绘图之间不暂停。在每个图形模式下，SAC 不仅在用 PLOT 命令产生的绘图之间，而且在绘制每个图形时都暂停。如果你是在命令文件或者作业控制程序的控制下使用 SAC，那么这一点将很有用。

11.164　whiten

概要

平滑输入的时间序列的频谱。

语法

```
WHITEN n [FILTERDESIGN]
```

输入

- n : 阶数 (极数)。这个数越大，结果数据就越平滑。高阶可以更好的清除一些数据，但是也可能会导致对数据处理过多而丢掉一些重要的数据。默认值为 6。
- FD : 进行一些类似于 filterdesign 的命令，使用白化系数，设计一个白化滤波器。详情可以参考 filterdesign 命令。

缺省值

```
whiten 6
```

说明

对数据中加入白噪声。平滑输入时间序列的频谱。当这个命令在谱分析命令 (比如子程序 SPE 中的命令、transfer 或 spectrogram) 之前执行，其减少了频谱值的动态范围，提供了对地震数据高频操作的精度。

WHITEN 可以在 SPE 子程序内部调用，或者从 SAC 的主 shell 中调用。SPE 中的 WHITEN 和主 shell 中的 WWHITEN 分别有不同的阶数。在主 shell 中，你可以调用 WHITEN 4，下一次在主 shell 中调用 WHITEN 时阶数为 4，但是在 SPE 中调用 WHITEN 时依然是缺省的 6 阶，除非你在 SPE 的命令行中进行了修改。进一步，SPE 中的阶数与 SPE COR 命令的 PREWHITEN 选项是一样的（设置了一个其他的也就设置了)。当然主 shell 中 WHITEN 命令与 TRANSFER 命令中的 PREWHITEN 选项也是一样的。

相关命令

transfer

11.165 whpf

概要

将辅助内容写入 HYPO 格式的震相拾取文件中

语法

```
WHPF IC n m
```

输入

- IC n m 在第 18 和 19 列插入带有两个整数的 n 和 m 的指令卡。n 的允许值为 0、1、5，m 的允许值为 0、1、9。

说明

“指令卡” 用于分开在 HYPO 文件中的不同事件，参见 HYPO71 手册。关闭一个已经打开的 HYPO 震相拾取文件 (CHPF) 或者退出 SAC 时，将自动添加 “10” 指令卡到震相读取文件中。

错误消息

- 1908: HYPO 震相拾取文件未打开

相关命令

chpf、ohpf

11.166 width

概要

控制所选图形设备的线宽

语法

```
WIDTH  [ON|OFF|linewidth] [SKELETON width] [INCREMENT ON|OFF]
    [LIST STANDARD|widthlist]
```

其中 linewidth, width, widthlist 是整数值，且 LIST 选项必须放在命令的最后

输入

- WIDTH ON : 打开 WIDTH 选项但是不改变当前线宽值
- WIDTH OFF : 关闭 width 选项
- WIDTH linewidth : 改变数据的线宽为 linewidth 并打开 WIDTH 选项。
- SKELETON width : 改变图形边框宽度为 width 并打开 WIDTH 选项
- INCREMENT ON : 按照 widthlist 表中的次序，依次改变一个宽度值
- INCREMENT OFF : 关闭线宽递变功能
- LIST widthlist : 改变宽度列表的内容。输入宽度列表。设置数据宽度为列表中的第一个宽度，并打开 width 选项。
- LIST STANDARD : 设置为标准线宽列表，设置数据宽度为列表中的第一个宽度，并打开 width 选项。

缺省值

```
WIDTH OFF SKELETON 1 INCREMENT OFF LIST STANDARD
```

说明

这个命令控制那些可显示很多不同线宽的设备的宽度属性。数据宽度是指绘制数据文件时使用的宽度。数据宽度可在每个数据文件绘图之后，按照宽度表中的次序自动地变成另一个宽度。边框宽度是用于绘制坐标轴的宽度。在 SKELETON 选项下，仅修改坐标轴的宽度。网格、文本、标签和框架号，总是用 1 号细线显示。如果在同一张绘图中同时绘制几个数据文件，你也许需要每个文件有不同的宽度。此时可使用 INCREMENT 选项。在这个选项打开时，每次绘制一个数据文件后，都按照宽度表中的次序自动地变成另一个宽度。宽度值和次序在标准宽度表中为:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

你可以使用 LIST 选项改变这个表的次序或内容。这个命令常用于重叠绘图 (参见 PLOT2)，此时你可能需要每张图上的数据宽度都按相同的顺序排列。

例子

选择自动变换的数据宽度起始值为 1:

```
SAC> width 1 increment
```

边框宽度起始值为 2，并按 1、3、5 的增量变化:

```
SAC> width skeleton 2 increment list 1 3 5
```

11.167　wiener

概要

设计并应用一个自适应 Wiener 滤波器

语法

```
WIENER [WINDOW pdw] [NCOEFF n] [MU OFF|ON|v] [EPS ILON OFF|ON|e]
```

输入

- WINDOW pdw : 设置滤波器设计窗口为 pdw。关于 pdw 参见 CUT 命令
- NCOEFF n : 设置滤波器系数为 n 个
- MU off | on | v : 设置自适应步长参数
- Off : 将 mu 设置为 0
- On : 将 mu 设置为 mu = 1.95 / Rho(0)。其中 Rho(0) 是 pdw 中延迟为 0 时的自相关系数。
- v : 设置 mu = v.
- EPSILON e : Set ridge regression parameter to epsilon.

缺省值

```
wiener window b 0 10 ncoeff 30 mu off epsilon off
```

说明

误差预测滤波器使用 Yule-Walker 方法，从指定的部分数据窗中由自相关函数给出。这个窗口可以是文件的任何部分，然后滤波器应用到这个个信号，即信号由残差序列置换。这个滤波器可以用作预白化或用作瞬时信号的检波预处理器。对 MU 指定非 0 值，滤波器可作成时域自适应的，大 MU 可能会导致不稳定。

例子

下面的命令将应用一个非自适应滤波器，将第一个十秒指定为窗:

```
SAC> wiener window b 0 10 mu 0.
```

下面命令将应用带 40 个系数的滤波器，指定设计窗为从文件开始到第一个到时前 1 秒:

```
SAC> wiener ncoeff 40 window b a -1
```

头段变量改变

depmin, depmax, depmen

错误消息

- 1301: 未读入数据文件
- 1306: 对不等间隔文件的非法操作
- 1307: 对谱文件的非法操作
- 1608: 不良的 Wiener 滤波器噪声窗口（滤波器设计窗口不在文件窗口内，或用于窗口的头段变量值未定义）

警告消息

- 1609: Wiener 滤波器数字不稳定

相关命令

cut

11.168 wild

概要

设置读命令中用于扩展文件表的通配符

语法

```
WILD [ECHO ON|OFF] [SINGLE char] [MULTIPLE char] [CONCATENATION chars]
```

输入

- ECHO ON : 打开扩展文件表回应开关，当这个选项打开并且在文件表中有通配符时才有回应。
- ECHO OFF : 关闭扩展文件表回应开关
- SINGLE char : 改变用于匹配单个字符的通配符
- MULTIPLE char : 改变用于匹配多个字符的通配符
- CONCATENATION chars : 改变用于字符串两侧的两个字符连接串

缺省值

选项	UNIX	VAX	PRIME
ECHO	ON	ON	ON
SINGLE	?	?	+
MULTIPLE	*	*	’
CONCATENATION	[,]	(,)	[,]

说明

很多现代操作系统都提供了通配符特性，也可以称为文件扩展。它是一个可以让你使用简短文件名以及简单的简写形式去指定一组文件的代号。SAC 在 READ、READTABLE 以及 READHDR 命令中使用通配符及一些扩展名，使用这些代号，你可以很容易地访问一些文件组:

- 所有以字母 “abc” 开头的文件
- 所有以 “z” 结尾的文件
- 所有文件名中严格包含三个字母的文件

通配符代号有三个元素。对于不同的系统三个元素会有不同的缺省符。你可以使用这个命令改变通配符。多重匹配字符 (“*”) 用于匹配字符串中任意字符串，包括空字符串。单个匹配符 (“?”) 用于匹配任意单个字符。连接符号 (“[” 和 “]”) 用于包围由逗号分隔的要匹配的字符串。在这个字符串中，可以包含单通配符或多通配符。

SAC 使用通配符完成文件名的扩展，通常有几个步骤:

1. 如果标识目录部分存在的话将将其去掉，否则使用当前目录
2. 做系统调用，以得到目录中所有文件的列表
3. 如果在标识中是一个连接表，就用其他字符形成连接表中每个字符的新的标识，然后匹配它们到文件表中。如果没有连接表标识，则可简单匹配标识到文件表
4. 去掉形成扩展文件表的所有重复的匹配
5. 如果需要，回显扩展文件表
6. 试着将扩展文件表读入内存

每个操作系统都使用一些不同的步骤在一个目录中存取文件。上面第一步的系统调用反映了这些不同。例如，在 UNIX 中以字母顺序显示文件名，但在 PRIME 或 VAX 上就不是这样。在 PRIME 目录中文件次序是随意的。这些不同反映在扩展文件表的文件次序上。你可以用各种不同的通配符和连接表进行实验，以确定扩展文件表中的文件次序是否重要。

下例将帮助你理解怎样使用这些通配符元素，一个有用的特征是 SAC 保存包含在连接表上的字符串，当你输入一个空表，则前面的表将被重复使用，这可以节省许多输入的操作。

例子

假定当前目录中包含如下次序的文件:

```
ABC DEF STA01E STA01N STA01Z STA02E STA02N STA02Z STA03Z
```

同样假定扩展文件设置回显，下面显示怎样使用各种通配符去将上面文件表的一部分读入内存:

```
SAC> READ S*
 STA01E STA01N STA01Z STA02E STA02N STA02Z STA03Z
SAC> READ *Z
 STA01Z STA02Z STA03Z
SAC> READ ???
 ABC DEF
SAC> READ STA01[Z,N,E]
 STA01Z STA01N STA01E
SAC> READ *[Z,N,E]
 STA01Z STA02Z STA03Z STA01N STA02N STA01E STA02E
SAC> READ *1[Z,N,E] *2[ ]
 STA01Z STA01N STA01E STA02Z STA02N STA02E
```

限制

在一个标识中只可以有一个连接串

相关命令

read、readtable、readhdr

11.169 window

概要

设置图形窗口被创建时的位置和形状

语法

```
WINdow n [XSIZE xwmin xwmax]  [YSIZE ywmin ywmax]
```

输入

- n : 要设置属性的图形窗口号，n 取值 1 到 10.
- X xwmin xwmax : 设置图形窗口相对屏幕的水平位置。xwmin 是窗口左边界位置，xwmax 是窗口右边界位置，相对屏幕的坐标取值为 0.0 到 1.0
- Y ywmin ywmax : 设置图形窗口相对屏幕的垂直位置，其他与 X 选项含义相同。

窗口位置缺省值（前 5 个）:

说明

参见 beginwindow 命令

表 11.3: SAC 标准窗口

n	xwmin	xwmax	ywmin	ywmax
1	0.05	0.85	0.05	0.60
2	0.07	0.87	0.07	0.62
3	0.09	0.89	0.09	0.64
4	0.11	0.91	0.11	0.66
5	0.13	0.93	0.13	0.68

例子

为了设置窗口 3 的垂直位置而不修改其水平位置:

```
SAC> window 3 y 0.1 0.9
```

11.170 write

概要

将内存中的数据写入磁盘

语法

```
WRITE [SAC|ALPHA|XDR] [DIR OFF|CURRENT|name] [KSTCMP]
    [OVER|APPEND text|PREPEND text|DELETE text|CHANGE text1 text2] filelist
```

输入

- no arguments : 使用以前的格式和写文件列表
- SAC : 将 SAC 二进制文件格式写入磁盘
- ALPHA : 写 SAC 字符数字型数据文件
- SEGY : 写 IRIS/PASSCAL 的 SEGY 格式文件。这个文件允许一个文件包含一个波形而非一个分量，SEGY 格式只能用于等间距时间序列文件。在 SAC 中 SCALE 字段被忽略，因为 SAC 将波形以一系列浮点数的形式储存起来，而 SEGY 则以一系列长整数储存。从 SAC 中得到的数据将被归一到允许的最大整数。SEGY 中的 scale 字段用于决定将波形复原到 SAC 文件时需要的因子。
- XDR : 用 SAC 二进制 xdr 格式写文件。这个格式用于实现不同构架的二进制数据的转换
- DIR OFF : 关闭目录选项，即写入当前目录
- DIR CURRENT : 打开目录选项并设置写目录为当前目录
- DIR name : 打开目录选项并设置写目录为 name。将所有的文件写入目录 name 中，其可以是相对或绝对路径
- KSTCMP : 使用 KSTNM 和 KCMPNM 头段变量为内存中每个数据文件定义一个文件名。生成的文件名将检查是否唯一，如果不唯一，则在文件名后加序号以避免冲突
- OVER : 使用当前读文件表作为写文件表，用内存中的文件覆盖磁盘上的文件
- APPEND text : 通过在当前读文件列表后附加文本创建写文件表

- PREPEND text : 通过在当前读文件列表前附加文本创建写文件表
- DELETE text : 通过在当前读文件列表中删除第一次出现的 text 创建写文件表
- CHANGE text1 text2 : 通过将当前读文件表中每个文件名第一次出现的 text1 修改为 text2 来创建写文件表
- filelist : 将写文件表设置为 filelist，这个列表可以包含文件名、相对/绝对路径，不可以包含通配符。

缺省值

```
write sac
```

说明

这个命令允许你在数据处理的任一步将结果写入磁盘。这个命令运用于几种磁盘文件格式。内存中的每个文件都将不经过裁剪或减采样地写入磁盘。

多数情况下，需要使用 SAC 数据文件格式。这是供快速读写的压缩二进制文件格式。它包含一个较大的头段和一段或两段数据记录。详情参见具体章节。

你可以直接指定写文件名，也可以通过修改内存中的当前文件名间接地指定它们。OVER 选项把写文件表设置到读文件表。它用于覆盖包含当前内存的数据的读入的最后一组磁盘文件。APPEND, PREPEND, DELETE,CHANGE 选项通过以所需要的方式修改每个读文件名的方式建立一个写文件表，这在宏命令中非常有用，在宏命令中你通常需要自动处理大量数据文件，并保持输出文件风格的一致。当选择这四个选项中的一个时，便可输出写文件，这使你可以看到实际使用的文件名。

例子

对一组数据文件进行滤波，然后将结果存入一组新数据文件:

```
SAC> read d1 d2 d3
SAC> lowpass butter npoles 4
SAC> write f1 f2 f3
```

也可以使用 CHANGE 选项完成这一操作:

```
SAC> read d1 d2 d3
SAC> lowpass butter npoles 4
SAC> write change d f
```

注意这种情况下 SAC 输出写文件表，若用滤波数据置换原始数据，则上例的第三行要变成:

```
SAC> write over
```

错误消息

- 1301: 未读入数据文件
- 1311: 没有要写的文件列表
- 1312: 写文件表中文件数目错误（写文件表中的文件数必须等于读入内存中的文件的数目）
- 1303: 文件未打开覆盖写标志（头段变量 LOVROK 是.FALSE. ）

相关命令

read

11.171　writebbf

概要

将一个暂存块变量写入磁盘

语法

```
WRITEBBF [file]
```

输入

- file : 暂存块变量文件的名字，其可以是简单文件名或相对/绝对路径。

缺省值

```
WRITEBBF BBF
```

说明

这个命令让你能够将暂存块变量写入磁盘，其可以稍后用 READBBF 命令读入 SAC，这个特性允许你保存 SAC 某次执行得到信息到另一个。你也可以在自己的程序中写代码去访问这些暂存块变量文件的信息。这使你可以在自己的程序与 SAC 之间交换信息。

相关命令

readbbf、setbb、getbb

11.172　writehdr

概要

用内存中的头段值覆盖磁盘文件中的相应头段值

语法

W!RITE!H!DR!

输入

说明

该命令并不会覆盖磁盘文件中的数据，使用 WRITE 将覆盖头段和数据。当 CUT 选项打开时，WRITEHDR 命令无法使用，内存中的头段被修改以反应 CUT 选项的效果，但是磁盘上的数据不会被修改。对被 CUT 的数据使用 WRITEHDR 命令将可能导致磁盘中的数据产生类似于平移或截断的影响。

错误消息

- 1301: 未读入数据文件

头段变量

更新磁盘的头段

限制

参见 cut 和 writehdr

相关命令

cut、write

11.173 writesp

概要

将谱文件作为一般文件写入磁盘

语法

```
WRITESP [ASIS|RLIM|AMPH|RL|IM|AM|PH] [OVER|filelist]
```

输入

- ASIS : 按照谱文件当前格式写入
- RLIM : 写入实部和虚部分量
- AMPH : 写入振幅和相位分量
- RL : 只写入实部分量
- IM : 只写入虚部分量
- AM : 只写入振幅分量
- PH : 只写入相位分量
- filelis : SAC 二进制数据文件列表，这个列表可以包含简单文件名和绝对/相对路径名

缺省值

```
writesp asis
```

说明

SAC 数据文件可以为时间序列文件或谱文件。头段中的 IFTYPE 用于区分这两种格式。当你读取一个时间序列到内存，对其做快速 Fourier 变换，然后将数据写回磁盘，此时的文件即为谱文件。

某些操作只能对时间序列文件进行，而某些操作只能对谱文件进行。比如，你无法对一个谱文件应用 taper 命令或者将两个谱文件乘起来。这是 SAC 的保护机制。

然而有时你需要对谱文件做这些操作，为了越过 SAC 的保护机制，你可以使用这个命令将谱文件像时间序列数据一样写入磁盘。每一个分量都将作为一个单独文件写入磁盘。然后你可以将这些文件读入 SAC 并进行任何你想要的操作。因为 SAC 认为其为时间序列文件。一旦这些计算完成了，你可以将修改之后的数据通过 WRITE 命令写回磁盘。如果你想要读回这个谱文件，可以使用 READSP 命令。

为了帮助你跟踪磁盘上的数据，SAC 将在你给出的文件名后加一个后缀以标识储存在文件的谱分量。后缀分别为 “.RL”, “.IM”, “.AM” 和 “.PH” 分别对应不同的分量。

例子

假设你想要对 FILE1 的谱文件振幅进行一些操作:

```
SAC> read file1
SAC> fft amph
SAC> writesp over
```

SAC 将输出两个文件 FILE1.AM 和 FILE1.PH，现在可以对振幅文件进行操作:

```
SAC> read file1.am
SAC> ...perform operations.
SAC> write over
```

现在磁盘中的文件为修改后的谱文件，如果你想要重建 SAC 谱数据并进行反变换:

```
SAC> readsp file1
SAC> ifft
SAC> write file2
```

错误消息

- 1301: 未读入数据文件
- 1305: 对时间序列文件非法操作

头段变量改变

磁盘文件中的 b, e, delta 将包含频率的起始值、结束值和增值，单位 hz

相关命令

`readsp`

11.174　xdiv

概要

控制 x 轴的刻度间隔

语法

```
XDIV [NICE|INCREMENT v|NUMBER n] [POWER ON|OFF]
```

输入

- NICE : 使用 “nice-numbered” 刻度间隔，由系统决定
- INCREMENT v : 设置刻度间隔增量为 v
- NUMBER n : 设置刻度间隔数为 n
- POWER ON : 打开幂选项，当这个选项打开时，SAC 以一个数字自乘的 10 次幂的形式给出刻度值
- POWER OFF : 关闭幂选项

缺省值

```
xdiv nice power on
```

说明

这个命令控制 x 轴刻度间隔的选择，多数时候默认的 “nice-numbered” 间隔可以满足需求。SAC 的刻度间隔是基于坐标轴的最小最大值、坐标轴的长度以及当前坐标轴字符的尺寸。你也可以使用 INCREMENT 选项强制刻度区间为一个定值，或者使用 NUMBER 选项设置刻度间隔的数目

11.175 xfudge

概要

改变 X 轴的 “插入因子”

语法

```
XFUDGE ON|OFF|v
```

输入

- ON : 打开插入选项，但不改变插入因子
- OFF : 关闭插入选项
- v : 打开插入因子，改变插入因子为 v

缺省值

```
xfudge 0.03
```

说明

当这个选项打开时，实际轴范围将由插入因子改变，确定线性坐标轴范围的方法是:

$$XDIFF = XFUDGE * (XMAX - XMIN)$$

$$XMIN = XMIN - XDIFF$$

$$XMAX = XMAX + XDIFF$$

其中 XMIN 和 XMAX 是数据的极值，XFUDGE 是插入因子，这个算法对于对数坐标也是相似的。插入选项值应用于坐标轴的范围设置为数据极值时 (参见 XLIM)

相关命令

xlim

11.176 xfull

概要

控制 X 轴的绘图为整对数方式

语法

```
XFULL ON|OFF
```

输入

- ON|OFF 打开/关闭整对数绘图选项

缺省值

```
xfull on
```

说明

整对数绘图选项仅应用于使用对数坐标且固定范围选项关闭的情况。这个选项打开时，实际坐标轴范围将被设置为数据范围前后的第一个整十数，当关闭这个选项时，将使用实际数据范围。

相关命令

xlim

11.177　xgrid

概要

控制绘图时的 x 方向的网格线

语法

```
XGRID ON|OFF|SOLID|DOTTED
```

输入

- ON : 打开 x 方向网格绘图但不改变网格类型
- OFF : 关闭 x 方向网格绘图
- SOLID : 打开 x 方向网格绘图并使用实网格线
- DOTTED : 打开 x 方向网格绘图并使用虚线绘制网格线

缺省值

```
xgrid off
```

说明

这个命令控制 X 坐标轴网格

相关命令

grid、ygrid

11.178　xlabel

概要

定义 X 轴标签及属性

语法

```
XLABEL [ON|OFF|text] [LOCATION TOP|BOTTOM|RIGHT|LEFT]
    [SIZE TINY|SMALL|MEDIUM|LARGE]
```

输入

- ON : 打开 X 轴标签选项，不改变文本
- OFF : 关闭 X 轴标签选项
- text : 打开 X 轴标签选项，改变文本内容，如果文本包含空格，需要用引号括起来
- LOCATION : 改变 X 轴标签位于窗口的位置 location 可以选 TOP | BOTTOM | RIGHT | LEFT ，不多解释
- SIZE : 改变绘图标签的尺寸
- TINY : 微小尺寸，每行 132 个字符
- SMALL : 小尺寸，每行 100 个字符
- MEDIUM : 中等尺寸，每行 80 字符
- LARGE : 大尺寸，每行 50 字符

缺省值

```
xlabel off location bottom size small
```

说明

如果这个选项为开，则 X 轴标签将放在每张图上，其尺寸和位置以及文本均可以改变。文本质量以及字体可以使用 GTEXT 命令设置

相关命令

gtext

11.179 xlim

概要

确定图形中 x 轴的范围

语法

```
XLIM [ON|OFF|pdw|SIGNAL]
```

输入

- ON : 打开 x 轴范围选项，但不改变范围值
- OFF : 关闭 x 轴范围选项
- pdw : 打开 x 轴范围选项并设置范围为新的”partial data window”。关于 pdw 请参见 CUT 命令
- SIGNAL : 等同于输入： A -1 F +1.

缺省值

```
xlim off
```

说明

当这个选项打开时，x 轴的范围固定，当这个选项为关时，这个范围与数据的大小成正比。界定 x 轴上的绘图范围可以可以用于 “放大” 当前内存中数据的图形。

相关命令

cut

11.180 xlin

概要

设置 X 轴为线性坐标

语法

```
XLIN
```

11.181 xlog

概要

设置 X 轴为对数坐标

语法

```
XLOG
```

11.182 xvport

概要

定义 X 轴的视口

语法

```
XVPORT xvmin xvmax
```

输入

- xvmin : X 轴视口的最小值，范围为 0.0 到 xvmax
- xvmax : X 轴视口的最大值，范围为 xmin 到 1.0

缺省值

```
xvport 0.1 0.9
```

说明

视口是实际绘出的图形窗口的一部分 (参见 VSPACE)。用于定义视口和视窗的坐标系称为虚拟坐标系。虚拟坐标系不依赖于特定物理设备显示表面的尺寸、形状或分辨率。AC 的坐标系在 x 和 y 方向都是 0 到 1 的范围。视窗左下角的坐标为 (0.0, 0.0)，右上角的坐标为 (1.0, 1.0)。这个坐标系的使用便于你指定一个图形的位置，而不必考虑特定的输出设备。

XVPORT 和 YVPORT 命令控制在视窗中哪个位置上绘制指定的图形。默认值使用了视窗的部分，在图形的每边留下一些空间绘制坐标轴、标签和标题。你可以使用这个命令把一个给定的图形安排在任何位置。当与 BEGINFRAME 和 ENDFRAME 命令一起使用，这些命令让你创建特殊的构图以在相同的框架中放置若干不同的图形。

例子

参见 beginframe

相关命令

vspace、xvport、beginframe

11.183 ydiv

概要

控制 Y 轴的刻度间隔

语法

```
YDIV [NICE|INCREMENT v|NUMBER n] [POWER ON|OFF]
```

输入

- NICE : 使用 “nice-numbered” 刻度间隔，由系统决定
- INCREMENT v : 设置刻度间隔增量为 v
- NUMBER n : 设置刻度间隔数为 n
- POWER ON : 打开幂选项，当这个选项打开时，SAC 以一个数字自乘的 10 次幂的形式给出刻度值
- POWER OFF : 关闭幂选项

缺省值

```
ydiv nice power on
```

说明

这个命令控制 y 轴刻度间隔的选择，多数时候默认的 “nice-numbered” 间隔可以满足需求。SAC 的刻度间隔是基于坐标轴的最小最大值、坐标轴的长度以及当前坐标轴字符的尺寸。你也可以使用 INCREMENT 选项强制刻度区间为一个定值，或者使用 NUMBER 选项设置刻度间隔的数目

11.184　yfudge

概要

改变 Y 轴的 “插入因子”

语法

```
YFUDGE ON|OFF|v
```

输入

- ON : 打开插入选项，但不改变插入因子
- OFF : 关闭插入选项
- v : 打开插入因子，改变插入因子为 v

缺省值

```
yfudge 0.03
```

说明

当这个选项打开时，实际轴范围将由插入因子改变，确定线性坐标轴范围的方法是:

$$YDIFF = YFUDGE * (YMAC - YMIN)$$

$$YMIN = YMIN - YDIFF$$

$$YMAC = YMAX + YDIFF$$

其中 YMIN 和 YMAX 是数据的极值，YFUDGE 是插入因子，这个算法对于对数坐标也是相似的。插入选项值应用于坐标轴的范围设置为数据极值时 (参见 YLIM)

相关命令

ylim

11.185　yfull

概要

控制 Y 轴的绘图为整对数方式

语法

```
YFULL ON|OFF
```

输入

- ON|OFF 打开/关闭整对数绘图选项

缺省值

```
yfull on
```

说明

整对数绘图选项仅应用于使用对数坐标且固定范围选项关闭的情况。这个选项打开时，实际坐标轴范围将被设置为数据范围前后的第一个整十数，当关闭这个选项时，将使用实际数据范围。

相关命令

ylim

11.186 ygrid

概要

控制绘图时的 Y 方向的网格线

语法

```
YGRID ON|OFF|SOLID|DOTTED
```

输入

- ON : 打开 Y 方向网格绘图但不改变网格类型
- OFF : 关闭 Y 方向网格绘图
- SOLID : 打开 Y 方向网格绘图并使用实网格线
- DOTTED : 打开 Y 方向网格绘图并使用虚线绘制网格线

缺省值

```
ygrid off
```

说明

这个命令控制 Y 坐标轴网格

相关命令

grid、ygrid

11.187 ylabel

概要

定义 Y 轴标签及属性

语法

```
YLABEL [ON|OFF|text] [LOCATION TOP|BOTTOM|RIGHT|LEFT]
    [SIZE TINY|SMALL|MEDIUM|LARGE]
```

输入

- ON : 打开 Y 轴标签选项，不改变文本
- OFF : 关闭 Y 轴标签选项
- text : 打开 Y 轴标签选项，改变文本内容，如果文本包含空格，需要用引号括起来

- LOCATION : 改变 Y 轴标签位于窗口的位置 location 可以选 TOP | BOTTOM | RIGHT | LEFT ，不多解释
- SIZE : 改变绘图标签的尺寸
- TINY : 微小尺寸，每行 132 个字符
- SMALL : 小尺寸，每行 100 个字符
- MEDIUM : 中等尺寸，每行 80 字符
- LARGE : 大尺寸，每行 50 字符

缺省值

```
ylabel off location bottom size small
```

说明

如果这个选项为开，则 Y 轴标签将放在每张图上，其尺寸和位置以及文本均可以改变。文本质量以及字体可以使用 GTEXT 命令设置

相关命令

gtext

11.188 ylim

概要

确定图形中 Y 轴的范围

语法

```
YLIM [ON|OFF|pdw|SIGNAL]
```

输入

- ON : 打开 Y 轴范围选项，但不改变范围值
- OFF : 关闭 Y 轴范围选项
- pdw : 打开 Y 轴范围选项并设置范围为新的”partial data window”。关于 pdw 请参见 CUT 命令
- SIGNAL : 等同于输入： A -1 F +1.

缺省值

```
ylim off
```

说明

当这个选项打开时，Y 轴的范围固定，当这个选项为关时，这个范围与数据的大小成正比。界定 Y 轴上的绘图范围可以可以用于 “放大” 当前内存中数据的图形。

相关命令

cut

11.189 ylin

概要

设置 Y 轴为线性坐标

语法

```
YLIN
```

11.190 ylog

概要

设置 Y 轴为对数坐标

语法

```
YLOG
```

11.191 yvport

概要

定义 Y 轴的视口

语法

```
YVPORT yvmin yvmax
```

输入

- yvmin : Y 轴视口的最小值，范围为 0.0 到 yvmax
- yvmax : Y 轴视口的最大值，范围为 ymin 到 1.0

缺省值

```
yvport 0.1 0.9
```

说明

视口是实际绘出的图形窗口的一部分 (参见 VSPACE)。用于定义视口和视窗的坐标系称为虚拟坐标系。虚拟坐标系不依赖于特定物理设备显示表面的尺寸、形状或分辨率。AC 的坐标系在 x 和 y 方向都是 0 到 1 的范围。视窗左下角的坐标为 (0.0，0.0)，右上角的坐标为 (1.0, 1.0)。这个坐标系的使用便于你指定一个图形的位置，而不必考虑特定的输出设备。

XVPORT 和 YVPORT 命令控制在视窗中哪个位置上绘制指定的图形。默认值使用了视窗的部分，在图形的每边留下一些空间绘制坐标轴、标签和标题。你可以使用这个命令把一个给定的图形安排在任何位置。当与 BEGINFRAME 和 ENDFRAME 命令一起使用，这些命令让你创建特殊的构图以在相同的框架中放置若干不同的图形。

例子

参见 beginframe

相关命令

vspace、xvport、beginframe

11.192　zcolors

概要

控制等值线的颜色显示

语法

```
ZCOLORS  [ON|OFF] LIST c1 c2 ... cn
```

输入

- ON : 打开等值线颜色显示开关
- OFF : 关闭等值线颜色显示开关
- LIST c1 c2 . cn : 设置等值线要使用的颜色列表，每一个颜色对应一条等值线，如果等值线数目多于这个列表长度，则整个列表不断重复
- cn : SAC 当前颜色表的颜色名

缺省值

```
zcolors off list red green blue
```

相关命令

contour、color

11.193　zlabels

概要

根据等值线的值控制等值线的标记

语法

```
ZLABELS [ON|OFF] [SPACING v1 [v2 [v3]]] [SIZE v] [ANGLE v] [LIST c1 c2 ... cn]
```

注意：LIST 选项只能放在这个命令的最后

输入

- ON|OFF : 打开/关闭等值线标签选项开关
- SPACING v1 v2 v3 : 设置相邻标签名的最小、适中和最大间隔（视口坐标系）分别为 v1、v2 和 v3。如果第二、三个值省略则使用前面一个值
- SIZE v : 设置标签的尺寸（高度）为 v
- ANGLE v : 设置标签文本最大角度为 v（自水平方向起算的角度，单位为度）

- LIST c1 c2 . cn : 设置使用的等值线标签的列表。在这个表上的每个输入用于相应的等值线，如果等值线数目大于这个表的长度，则重复使用整个等值线表
- cn : ON|OFF|INT|FLOATn|EXPn|text
- ON : 在相应的等值线上放置标签，使用 FORTRAN 自由格式，用等值线值形成标签名
- OFF : 在相应的等值线上不放置标签名
- INT : 在相应的等值线上放置整数标签名
- FLOATn : 在相应的等值线上放置小数点后面 n 位的浮点数作为标签名。如果 n 被忽略则使用先前值
- EXPn : 在相应的等值线上放置小数点后面 n 位数的指数幂形式标签名，如果 n 忽略则使用先前值
- text : 使用文本标注相应的等值线

缺省值

```
ZLABELS  OFF  SPACING 0.1 0.2 0.3  SIZE  0.0075 ANGLE 45.0  LIST ON
```

相关命令

contour

11.194 zlevels

概要

控制后续等值线图上的等值线间隔

语法

```
ZLEVELS [SCALE] [RANGE v1 v2] [INCREMENT v] [NUMBER n] [LIST v1 v2 ... vn]
```

输入

- SCALE : 根据数据自动确定等值线的标尺范围
- RANGE v1 v2 : 用户设置等值线的范围（最小和最大）为 v1 和 v2。可以使用 SCALE 选项，也可以使用 RANGE 选项，但不可同时使用二者
- INCREMENT v : 设置等值线之间的增量为 v
- NUMBER n : 设置等值线的条数为 n，你可以使用 INCREMENT 或 NUMBER 选项但不可二者同时使用
- LIST v1 v2 .. vn : 设置一系列等值线上的值为 v1、v2 等等，如果使用这个选项，则其他选项均被忽略

缺省值

```
zlevels scale number 20
```

例子

参考 contour 中 zlevels 的使用

限制

等值线的最多数目为 40

相关命令

contour

11.195　zlines

概要

控制后续等值线绘图上的等值线线型

语法

```
ZLINES  [ON|OFF] [LIST n1 n2 ... nn] [REGIONS v1 v2 ... vn]
```

输入

- ON|OFF : 打开等值线显示选项
- LIST n1 n2 .. nn : 设置要使用的线型表，这个表上的每个输入用于相应的等值线。如果等值线的数目大于这个表中给出的线型的数目，则使用整个线型表
- REGIONS v1 v2 .. vn : 设置等值线范围表。这个表的长度应小于线型表的长度，小于范围值的等值线使用线型表中相应的线型。超过最后一个范围值的等值线采用线型表中最后一个线型的值

缺省值

```
zlines on list 1
```

例子

循环四种不同线型，建立等值线:

```
SAC> zlines list 1 2 3 4
```

设置虚线表示低于 0.0 等值线，实线表示高于 0.0 的等值线:

```
SAC> zlines list 2 1 regions 0.0
```

相关命令

contour

11.196　zticks

概要

用方向标记标识等值线

语法

```
ZTICKS [ON|OFF] [Spacing v] [LENGTH v] [DIRECTION DOWN|UP] [LIST c1 c2 ... cn]
```

输入

- ON|OFF : 打开/关闭等值线方向标记
- SPACING v : 在每条线段上设置项链标识之间的间隔为 v（视口坐标系）
- LENGTH v : 设置每个标识的长度为 v（视口坐标系）
- DIRECTION DOWN|UP : 标识在 z 值减小/增加的方向上
- LIST c1 c2 . cn : 设置要使用的等值线标识表。在这个表上的每个输入都用于相应的等值线。如果等值线数多于这个列表的长度，则重复使用整个标识表。ON 意味着标识画在等值线上，OFF 意味着标识不画在等值限上。

缺省值

```
zticks off spacing 0.1 length 0.005 direction down list on
```

例子

参考 contour 例子中 zticks 的使用

相关命令

contour