新SUユーザーズ・マニュアル

John W. Stockwell, Jr. & Jack K Cohen
訳: 渡辺 俊樹

Version 4.0: Jan 2008
日本語版 4.0-J1.0: Mar 2015

Seismic Unix プロジェクトは下記のサポートを受けています。

Center for Wave Phenomena （波動現象センター）

Colorado School of Mines （コロラド鉱山大学）
Golden, CO 80401, USA

過去には以下からのサポートを得ていました。

The Society of Exploration Geophysicists （米国物理探査学会）

The Gas Reserch Institute （ガス研究協会）

CWP/SU マニュアルの日本語訳について

本文書は、米国コロラド鉱山大学 (Colorado School of Mines) 波動現象センター (Center for Wave Phenomena) で開発・保守されている CWP/SU パッケージのマニュアルの日本語訳です。

本訳出に使用した原典は、ウェブサイト http://www.cwp.mines.edu/cwpcodes/ および配布パッケージに含まれているマニュアル sumanual.[300, 600]dpi.[a4, letter].[ps, pdf] の Version 4.0: Jan 2008 です。(これ以前には、<Version 2.5: Oct 1998> に基づく日本語版 Version 2.5-J1.0 (1999) がありました。)

お断り: CWP/SU の使用に当たっては原語版のドキュメント、マニュアルを参照して下さい。日本語訳に従って操作した場合の結果について、訳者は責任を追いません。

訳文について: 訳文はなるべく原文に忠実な表現で翻訳しましたが、日本語としてわかりやすいように心がけ、意訳したところもあります (力不足のため、意味のわかりにくいところも一部残されています)。原語版に見られた誤りは訳文ではわかる限り訂正するか、あるいは、訳注としてコメントしました。用語や表記、フォントはできる限りマニュアル中で用法を統一しました。そのため、原語版と対応が異なっていることがあります。また、今回の改訂にあたり旧版の訳文を見直しています。

> 原語版の Version 4.0: Jan 2008 では、それ以前の版の “Chapter 5: Editing SU Data” が “Chapter 4: Viewing SU Data in X-Windows and PostScript” の **4.4** に移されています。内容から考えて旧版の方がより適切であること、編集のミスではないかと思われるふしがあることから、日本語訳では旧版通り第 5 章としました。そのため、原語版と日本語版とでそれ以降の章の番号が一つずれる結果になっています。ご了承下さい。

訳文中の誤り、特に技術的な誤り、不適切な用語、意味不明の文章、慣用句や専門用語の正しい訳語など、訂正すべき個所がありましたら、ぜひご指摘下さい。

日本語訳に関するご質問・ご指摘は下記へお送り下さい :

渡辺 俊樹

東京大学地震研究所 地震火山噴火予知研究推進センター

twatanabe@eri.u-tokyo.ac.jp (2016 年 3 月末まで)

名古屋大学大学院環境学研究科 附属地震火山研究センター

watanabe@seis.nagoya-u.ac.jp (2016 年 4 月以降)

改変履歴:

- 1998.03.12 Web 版マニュアルの日本語訳第 1 版完成、SU Janap Users' Group に公開をアナウンス。
- 1999.06.22 Version 2.5 に基づき LaTeX 文書化。
- 1999.07.06 日本語版 Version 2.5-J1.0 完成、公開。
- 2014.06.06 Version 4.0 に基づく改訂に着手。すぐ休眠。
- 2015.03.30 日本語版 Version 4.0-J1.0 完成、公開。
  - 2015.04.21 日本語版 Version 4.0-J1.1 改行の調整など。

目 次

ライセンスと法的事項

承認された引用形式:

出版物においては、SU を下記の例に従って参照して下さい。

Cohen, J. K. and Stockwell, Jr. J. W., (200_), CWP/SU: Seismic Un*x Release No. __: an open source softwarefree package for seismic research and processing, Center for Wave Phenomena, Colorado School of Mines.

査読誌に掲載された SU についての論文:

Saeki, T., (1999), A guide to Seismic Un*x (SU)(2)—examples of data processing (part 1), data input and preparation of headers, Butsuri-Tansa (Geophysical Exploration), vol. 52, no. 5, 465-477.

Stockwell, Jr. J. W. (1999), The CWP/SU: Seismic Un*x Package, Computers and Geosciences, May 1999.

Stockwell, Jr. J. W. (1997), Free Software in Education: A case study of CWP/SU: Seismic Un*x, The Leading Edge, July 1997.

Templeton, M. E., Gough, C. A., (1998), Web Seismic Un*x: Making seismic reflection processing more accessible, Computers and Geosciences.

謝辞:

SU は CWP/SU: Seismic Un*x の略称です。Colorado School of Mines で開発された処理ラインであり、一部 Stanford Exploration Project (SEP) のソフトウェアに基づいています。

謝 辞

Seismic Unix プロジェクトは、the Society of Exploration Geophysicists (SEG) および Center for Wave Phenomena (CWP), Department of Geophysical Engineering, Colorado School of Mines に部分的に助成を受けています。過去にはこれらのグループに加え Gas Research Institute (GRI) にもサポートされていました。

SEG および CWP の継続的なサポートに感謝します!

CWP コンソーシアム・プロジェクトのスポンサーは長きに渡り SU プロジェクトのパートナーであり、ここにその関係に明示的に謝意を表します。加えて、過去、グラフックス・ターミナル上で実装されていた SU がワークステーションに移植された 1980 年代後半の、IBM 社と Colorado School of Mines の Center for Geoscientific Computing の特別なサポートに感謝します。

私達のセンターまたその他の多くのスタッフと学生が SU に貢献したので、ここに全員の名前をあげることはできませんが、非常に重要な貢献をした人達を特にここで取り上げたいと思います。

Einar Kjartansson は、1970 年代の後半、まだ Jon Claerbout の Stanford Exploration Project (SEP) の大学院生であった頃に、現在 SU と呼ばれているもの (SY package) を書き始めました。彼は、80 年代の前半 University of Utah の教授であった間パッケージの拡張を続けました。1984 年、SEP への滞在中、Einar は SY を Shuki Ronen に、さらに Stanford の大学院生に紹介しました。Ronen は 1984 年から 1986 年にさらに SY の開発を進めました。SEP の他の学生はこれを使い始め、コードやアイディアに貢献しました。SY は SEP で開発された他の多くのソフトウェアから多くの着想を得て、Claerbout の基金と、多くの彼の学生; Rob Clayton, Stew Levin, Dave Hale, Jeff Thorson, Chuck Sword, そして 70 年代、80 年代初期に Unix 上の地震探査処理を開拓した他の人たちから恩恵を受けました。

1986 年に、Shuki Ronen は Colorado School of Mines での 1 年間のポスドク生活に際し、この仕事を私達のセンターに持ち込みました。この間、Ronen は Cohen を支援して、SU をサポート可能な移植可能なものへと作り変えました。

Chris Liner はセンターの学生であった間、SU のワークステーション以前 (すなわちグラフィックス・ターミナル) の時代の多くのグラフィックス・コードに寄与しました。地震学と地震データ処理の広い知識により、Liner は SU のコーディング哲学にポジティヴな影響を与え続けています。

Craig Artley は現在 Golden Geophysical に勤務していますが、CWP の学生の頃グラフィックス・コードに大きな貢献をし、さらに、パッケージ全体にも重要な貢献をし続けています。

Dave Hale は大部分の中心的な科学的ライブラリ、グラフィックス・ライブラリのみならず “ヘビィ・デューティ” な処理コードを書きました。彼の C 言語のコーディング実装に関する知識はパッケージを応用計算機科学者にとってのよい例とするのに役立ちました。

Ken Larner は “実社会” での地震処理の広い知識に基づいて、多くのユーザー・インターフェイスの設計に貢献しました。

John Scales は彼の電子テキストの「Theory of Seismic Imaging, Samizdat Press, 1994.」

において、授業でどのようにSUを効果的に使うかを示しました。このテキストはSamizdat Pressのサイト samizdat.mines.edu から利用できます。

John StockwellのSUへの関与は1989年に始まりました。彼は、パッケージを大部分のUnixプラットフォームに容易にインストールできるようにするためのMakefile構造の設計に責任を持っています。彼は、1992年9月の最初の公開リリース (Release 17) 以来、このプロジェクトの主要な窓口です。Jack Cohenが1996年になくなってから、Stockwellがこのプロジェクトの責任者 (PI) の役を担っていると目されてきており、現在に至っています。John StockwellがプロジェクトのPIになってからの11年間にコードの数は3倍以上に増えています。

プロジェクトは世界中のSUユーザー・コミュニティからの多くの技術的支援を受けて来ました。その中でここで述べておくべきなのは以下の方々です。Memorial University in NewfoundlのTony Kocurko、Baltic Sea Research Institute in WarnemuendeのToralf Foerster、Mobil OilのStewart A. Levin, John Anderson, Joe Oravetz、AmocoのJoe Dellinger、University of Southern California at ChicoのMatthew Rutty、Jens Hartmann, Alexander Koek, Michelle Miller、DelftのBerend Scheffers, Guy Drijkoningen、StrasbourgのDominique Rousset、Pau and Marc Schaming、Curtin University, AustraliaのMatt Lamont、University of UtahのWenying Cai、Phillips PetroleumのBrian Macy、TerrasysのRobert Krueger、University of Koeln, GermanyのTorsten Shoenfelder、Geological Survey of CanadaのIan Kay、University of KyotoのToshiki Watanabe、そして、独立コンサルタントのReg Beardsleyです。その他のSUへの貢献者の方々を省略したことに対してお詫び申します— きっと、あなたをこのマニュアルの将来の更新時に含めることをお約束します！

この文書だけでなく、我々が公共の会議で配付するSUのパンフレットやその他の書類について、Barbara McLenonの文章に関する細かい指摘と素晴らしいデザインに、特に感謝します。

追 悼

Jack K. Cohen博士は1996年10月に亡くなりました。Seismic Unixパッケージは彼の地震データ処理に関する知識、創造性、そして科学コミュニティへの永続的な寄与への希望とによって存在しています。彼の死は数学、地球物理学、そしてSUユーザーのコミュニティで彼と出会ったすべての人に惜しまれるでしょう。

まえがき

このマニュアルの最初のバージョンが配付されてから 7 年の間に、SU パッケージに多くの変更がありました。これらの変更は、ここ CWP での内部の活動の結果であるとともに、それと同様に重要な、CWP 外部の SU ユーザーからのコード、バグ修正、拡張および提案といった多くの寄与の結果です。これらの多くの変更や拡張を見返し、世界中の SU コミュニティのメンバーとの議論を経て、新しいマニュアルが必要であることが明らかになりました。

この新しいバージョンは、University of Calgary の CREWES プロジェクトにおける SU のショート・コースのための準備として始められました。元々のマニュアルにあった多くの事項、例えばパッケージの取得とインストールに関する情報は付録に移しました。プログラムの中心的なコレクションを説明する節が追加されました。マニュアル全体はコードのより詳細な説明を含むように拡張されました。

ユーザーが、ヘルプ機能を用いてパッケージをどう使うかよく知ることができること、デモを走らせることによってプログラムの走らせ方を学べること、そして、パッケージのソース・コードをサンプルとして参考にして、SU コードをどうやって書き始めたらよいかがわかること、これらがこのマニュアルの意図するところです。

第1章　SUについて

1987年に、Colorado School of Mines (CSM) のCenter for Wave Phenomena (CWP) のJack K. CohenとShuki Ronenは大胆な計画を思いつきました。その計画は、UNIXベースのシステム上にC言語で書かれた地震探査処理環境を作ろうというもので、UNIXオペレーティング・システムを地震探査の処理および研究の領域にまで拡張することになるものでした。さらに、彼らはそのパッケージを完全なソース・コードの形で、欲しい人が誰でも自由に利用できるように目指しました。

彼らは、Einar KjartanssonがJon ClaerboutのStanford Exploration Project (SEP) の大学院生であった1970年代後半に開発を始め、University of Utahの教授であった80年代に改良を加えてきた、SYと呼ばれるパッケージを用いて作業を開始しました。1984年に、EinarはClaerboutが夏期休暇の間、彼の学生（と家）の世話をするためにSEPに戻りました。Einarは、当時SEPの大学院生であったShuki RonenにSYを紹介しました（Claerboutが休暇から戻りEinarがUtahへ去る前日に）。1984年から1986年の間、RonenはSYをさらに開発しました。SEPの他の学生たちもSYを使い、コードやアイディアに貢献しました。SYはSEPで開発された他のソフトウェアから多くの着想を得ました。また、Claerboutの基金と、多くの彼の学生; Rob Clayton, Stew Levin, Dave Hale, Jeff Thorson, Chuck Sword、そして70年代、80年代始めにUnix上での地震探査処理を切り開いた他の人々から支援を得ました。

1986年にRonenがCSMでの1年間の客員教授を受けたとき、Claerboutに彼が育ててきたSEPのアイディアとソフトウェアを広めるように勧められました。インターネットとftp以前の時代、SYは9トラックのテープに乗って車のトランクに入れられてカリフォルニアからコロラドまでドライブしたのです。Ronenはサバティカルに出かけたJack Cohenの代わりにCSMに行きました。SUにとって幸運なことに、Jackの息子のDanの健康上の問題から、Jackのサバティカルは最初に計画された1年間ではなく2ヶ月しか続きませんでした。JackはCWPに戻ると、SYに興味を持ち始めました。このパッケージは、当時利用されていた商用の地震探査処理ソフトウエアからはっきりと先に進んだものものでした。この当時の業界標準は、VAX-VMSベースのシステム上で動くFortranプログラムでした。

1987年にCohenとRonenが最初のバージョンのSUを作り上げた頃までに、CWPのスポンサーはすでにそのパッケージに興味を示し始めていました。UNIXをベースとしたワークステーションが利用できるようになったことと、主要な研究機関からUNIXの知識のある地球物理学者が流入したこととが相いまって、産業界は研究や処理のためにUNIXをベースにしたシステムを優先的に使用するようになり、SUを含む、UNIXをベースとしたソフトウエアへの関心が高まりました。

1992年の9月まで、SUは主にCWPの内部でだけ使用されていました。初期のバージョンのSUはCWPのスポンサー会社に移植され、それらの会社で開発された社内用の地震探査処理パッケージの基礎になりました。一旦SUパッケージがインターネット上で一般に利用できるようになると、SUはもっと広いコミュニティで使われるようになりました。このパッケージは物理探査の研究者や技術者、地震学者、環境技術者、ソフトウエア開発者、そ

の他の人々に使われています。SU は、小さな地質工学会社、また、大きな石油・ガス会社の技術スタッフ、大学や政府機関の研究者にも、地震探査データ処理にもソフトウエア開発環境として使われています。

1.1　SU とは何か (What SU is)

SU パッケージは**フリー・ソフトウエア**です。すなわち、あなたがこのマニュアルの最初、そして現在のリリースのパッケージに含まれる LEGAL_STATEMENT ファイル（この後者のファイルが優先されます）にあるライセンスを尊重する限り、コードをデータ処理にもソフトウエア開発にも無制限に使用することができます。パッケージは 3〜6ヵ月の間隔で発表される新しいリリースによって定期的に保守され、拡張されています。更新の間隔は CWP の公式バージョンに蓄積される変更の程度に依存します。パッケージは完全なソース・コードの形で配布されます。したがって、ユーザーはその能力を変更したり、拡張することができます。このパッケージの背後にある哲学は、多くのユーザーの要望に合うように保守、拡張できるような、ある確固とした構造を持った自由 (無料) なデータ処理および開発環境を提供するということです。

しかし、このパッケージは必ずしも地震探査処理に限定されていません。SU によって波動に関連した幅広い処理を行うことができるので、“seismic” という語が示すよりもいくぶん一般的なパッケージになっています。SU は UNIX オペレーティング・システムの拡張として意図されており、したがって、UNIX の柔軟性と拡張性を含めた多くの UNIX の特徴を共有しています。基本的な UNIX の哲学は、すべてのオペレーティング・システムのコマンドはそのオペレーティング・システムで走るプログラムであるということです。ここでのアイディアは、個々の作業が同定され、それらの仕事をするためだけの小さいプログラムが書かれることです。コマンドには作業のバリエーションを認めるオプションがあっても構いません。しかし、基本的な考えは「一つのプログラムに一つの作業」です。UNIX はマルチタスクのオペレーティングシステムですから、複数のプロセスを “パイプ” (|) を通して一列につなげることができます。

このような分散化は、例えば Microsoft 社のアプリケーションやいくつかの商用の地震探査ユーティリティのように、すべてをこなそうとする単一の “怪物” アプリケーションを起動しないようにして、オーバーヘッドを小さくするという利点があります。

UNIX はさまざまなシェル言語をサポートするという特長を備えることにより、UNIX 自身もいわばメタ言語のようになりました。Seismic Unix も同様に、これらの特性から多くを得ています。Seismic Unix プログラムは、標準の UNIX プログラムと連携し、シェル・スクリプト中で用いることによって、パッケージの機能を拡張することができます。

もちろん、特定の作業を満たす UNIX あるいは Seismic Unix のプログラムがないこともあり得ます。これは、新しいプログラムが書かれないといけないということを意味しています。ソース・コードの標準セットが利用でき、人間に読みやすいように設計されているので、新しいソース・コードを追加してパッケージの機能を拡張するプロセスがはかどります。

1.2　SU とは何ではないか (What SU is not)

SU パッケージは、現在のところグラフィカルなユーザー・インターフェースで動くユーティリティではありません。Java や Tcl/Tk を含めて可能性はいくつかあり、将来には開発

されるかもしれませんが。多くの商用の地震探査処理パッケージはGUIベースなので、ユーザーがSUも同様であってほしいと期待するのは避けられないところです。しかし、それはいくつかの理由でフェアな比較ではありません。

先に述べたように、SUはUNIXオペレーティング・システムの延長にあります。UNIXのすべての能力をメニューから利用できるようなGUIで動くUNIXオペレーティング・システムがないのとまったく同様に、そのようなインターフェースを通してSeismic Unixのすべての機能にアクセスすることを期待するのはおかしな話です。せいぜい、どんなSUインターフェースもそのパッケージのもつ能力の一部を提供するに過ぎないでしょう。

SUは商用の地震探査パッケージに取って代わるものではありません。商用の地震探査ソフトウェア・パッケージは、生産レベルの地震探査データ処理を目的として特別に設計された産業的な強固さを備えた (industrial-strength) ユーティリティです。もし、あなたが商用レベルの処理に携わっていて、そのような商用ソフトウェアのライセンスを購入するための資金の導入に携わっていたり、そう計画しているのでしたら、商用ユーティリティの代わりにSUを使うというのはありえない話です。

しかし、SUはあなたが使っている商用パッケージを補助する上で重要な役割を果たし得ます。商用パッケージが生産の仕事に使用されているようなところでは、SUにはパッケージのプロトタイプとしての居場所があります。また、新しいコードを書かなくてはならないときに、SUは新しいソフトウェア・アプリケーションの出発点となることができます。実際、地震探査データ処理の能力を利用できるということで、処理に携わっていない人に処理を試みようという気を起こさせたり、ソフトウェア開発者でない人にソフトウェア開発をする気を起こさせたり、研究者でない人が研究に従事したりすることができるかもしれません。

SUは地震探査のアプリケーションに限られていません。地球物理学と信号処理の両方のアプリケーションとしての使い道があります。SUは“波動関係”の信号処理と、特にフーリエ変換についての学生の教育に確かに役立ちます。2、3あげるとすると、これにはレーダー、非地震の音響学、イメージ処理などが含まれます。

もう一つ、少なくとも現在のバージョンではSUは3Dのパッケージではありません。とは言え、特に2Dのパッケージである必要もありません。というのも、多くのフィルタ操作やトレース操作は2Dと3Dとで同じであるからです。しかし、SUの将来のリリースには3Dアプリケーションも含まれることもありそうです。Release 32で初めて3Dマイグレーションのコードが含まれました。望ましくはこれが新しい開発のステージとなることでしょう。

1.3 SUの取得とインストール

SUのコーディング標準は移植性を重要視しているので、パッケージはUNIXオペレーティング・システムが走っており、新しいバージョンの“make”とANSI準拠のCコンパイラを備えているどんなシステム上にもインストールできるようになっています。ほとんどのシステム[1]でGNU makeとGCCをこれらのプログラムの代わりに使用できます。

SUの新しいリリースは、CWPの公式バージョンの変更の蓄積の程度によりますが、3〜6カ月ごとに発表されています。古いリリースはサポートされません。しかし、バージョンアップの際に不都合な点があれば、私 (John Stockwell) に電子メールで連絡して下さい。そうすれば問題を解決できるでしょう。

[1]特筆すべきことは、最近GNUプロジェクトからリリースされた**CYGWIN32**というソフトウェア・パッケージは充分なUNIX風の機能を持っており、SUのソースコードやMakefileを変更することなしにSUをWindows NTに移植できます。

SUの取得とインストールに関する解説はこのマニュアルの付録Aにあります。

第2章　ヘルプの機能

UNIX オペレーティング・システムと同様に、Seismic Unix も言語として考えることができます。どのような言語も、便利に扱えるようになるまでにはある程度のボキャブラリーを身に付けなければなりません。SU は多くのプログラムから構成されているので、ボキャブラリーについて必ず生じる質問に答えられる “辞書” がなくてはなりません。このマニュアルをそのような辞書の第一歩とするつもりです。

SU には UNIX のような “man ページ” はありません。しかし、それと同等な内部文書機能があります。まず、何が利用できるかについて概説する一般的なレベルのヘルプ・ユーティリティがあります。あるコードの特定の点について情報を得るために、大部分のプログラムは、自己文書化された文書 **selfdoc** (self-documentation) を持っています。コマンドラインでオプションを与えずにプログラム名を入力することで、これを表示することができます。

(これ以降のすべてのサンプルでは、パーセント記号 “%” は UNIX のコマンドライン・プロンプトを表します。コマンドの一部として入力してはいけません。)

以下のツールは、パッケージのプログラム、シェル・スクリプト、ライブラリ関数の詳細な内部文書をいろんなレベルで提供します。

- SUHELP - CWP/SU プログラムとシェルのリストを表示する。
- SUNAME - self-doc から名前のリストとソースコードの場所を得る。
- selfdoc - 大部分の実行形式プログラムとシェルスクリプトにある内部文書化ユーティリティ。selfdoc は、コマンドラインでオプションを与えずに、また、パイプ | や UNIX のリダイレクト >, < によって入出力のリダイレクションを使用せずに、プログラムやシェル・スクリプトの名前を入力することで表示させることができます。非実行形式 (ライブラリ関数) や selfdoc 機能のないプログラムのためには、それらの情報のデータベースを提供するダミーの selfdoc があります。
- SUDOC - コードの DOC リストを得る
- SUFIND - selfdoc から情報を得る
- GENDOCS - selfdocs の完全なリストを LaTeX 形式で得る
- suhelp.html - SU プログラムの全体の概要の HTML 文書
- SUKEYWORD - segy.h 中の SU のキーワードのガイド

この章ではこれらのユーティリティについて説明します。SU プログラムについての最も一般的な情報から最も詳しい情報までを得る方法を読者に示すことを目的としています。

2.1　SUHELP - 実行形式プログラムとシェル・スクリプトのリスト

個々の実行形式 (すなわち、メイン・プログラムとシェル・スクリプト) を含む、パッケージ内の SU の内容の一般的なリストを得るには、以下のように入力して下さい:

```
% suhelp
CWP PROGRAMS: (no self-documentation)
ctrlstrip       fcat            maxints         t
downfort        isatty          pause           upfort

PAR PROGRAMS: (programs with self-documentation)
a2b             kaperture       resamp          transp          vtlvz
b2a             makevel         smooth2         unif2           wkbj
farith          mkparfile       smoothint2      unisam
ftnstrip        prplot          subset          unisam2
h2b             recast          swapbytes       velconv


press return key to continue
....
```

全部を見るためには、実際に **suhelp** と入力するか、付録 B を参照して下さい。もう一つ、役に立つコマンド操作は次の通りです:

```
% suhelp | lpr
```

これは **suhelp** の出力をプリンタへ出力するものです。

2.2　SUNAME - SU の個々の項目の名前と短い説明のリスト

CWP/SU パッケージの内容のもっと完全なリストは、以下のように入力することによって得られます。

```
% suname

 -----  CWP Free Programs -----
CWPROOT=/usr/local/cwp

Mains:

In CWPROOT/src/cwp/main:
* CTRLSTRIP - Strip non-graphic characters
* DOWNFORT - change Fortran programs to lower case, preserving strings
* FCAT - fast cat with 1 read per file
* ISATTY - pass on return from isatty(2)
* MAXINTS - Compute maximum and minimum sizes for integer types
* PAUSE - prompt and wait for user signal to continue
* T - time and date for non-military types
* UPFORT - change Fortran programs to upper case, preserving strings

In CWPROOT/src/par/main:
A2B - convert ascii floats to binary
B2A - convert binary floats to ascii
DZDV - determine depth derivative with respect to the velocity,
FARITH - File ARITHmetic -- perform simple arithmetic with binary files
FTNSTRIP - convert a file of floats plus record delimiters created
H2B - convert 8 bit hexidecimal floats to binary
KAPERTURE - generate the k domain of a line scatterer for a seismic array
MAKEVEL - MAKE a VELocity function v(x,y,z)
MKPARFILE - convert ascii to par file format
```

```
PRPLOT - PRinter PLOT of 1-D arrays f(x1) from a 2-D function f(x1,x2)
RAYT2D - traveltime Tables calculated by 2D paraxial RAY tracing
RECAST - RECAST data type (convert from one data type to another)
REGRID3 - REwrite a [ni3][ni2][ni1] GRID to a [no3][no2][no1] 3-D grid
RESAMP - RESAMPle the 1st dimension of a 2-dimensional function f(x1,x2)
...
```

全部を見るには、実際に **suhelp** と入力するか、付録 B を参照して下さい。

2.3 Selfdoc - 自己文書化 (Self-Documentation) プログラム

SU には UNIX のマニュアル・ページ (man pages) はありません。マニュアル・ページは UNIX の標準機能で、Seismic Unix も当然見習うべきものなので、これには驚く人も (がっかりする人も) いるかもしれません。しかし、パッケージは、**selfdoc** または、自己文書化 (self-documentation) 機能と呼ばれる同等の機能を有しています。

selfdoc は各プログラム内に書き込まれたパラグラフです。UNIX の端末ウィンドウのコマンドラインから、プログラムがオプションもリダイレクトも入力ファイルもなしに打ち込まれた場合、そのパラグラフが標準エラー (画面) にプリントされます。

例えば:

```
% sustack

 SUSTACK - stack adjacent traces having the same key header word

     sustack <input >output [Optional parameters]

 Required parameters:
        none

 Optional parameters:
        key=cdp         header key word to stack on
        normpow=1.0     each sample is divided by the
                        normpow'th number of non-zero values
                        stacked (normpow=0 selects no division)
        verbose=0       verbose = 1 echos information

 Note:  The offset field is set to zero on the output traces.
        Sushw can be used afterwards if this is not acceptable.
```

最初の行は、プログラム名 **sustack** とプログラムが何をするかの短い説明を示します。これは、**suname** の一覧の **sustack** の行に表示されるものと同じです。第 2 行:

```
    sustack <stdin >stdin [Optional parameters]
```

はプログラムをコマンドラインからどのように入力するかを示します。“stdin” と “stdout” はそれぞれ、標準入力と標準出力を表します。UNIX の言葉で言えば、ユーザーはディスクファイル、または、UNIX の “入力リダイレクト”< および “出力リダイレクト”> または、パイプ | を通じてデータの入出力ができるということです。

“Required parameters: (必須パラメータ)” と “Optional parameters: (オプション・パラメータ)” と記されているパラグラフは、プログラムの操作のために必要とされるコマンドライン・パラメータと、オプションのパラメータを示します。オプション・パラメータのデフォルト値は等式で与えられます。プログラムはコマンドライン引数なしでデータが与え

られた場合、それらのパラメータをデフォルト値であると仮定します。例えば: “key=cdp” は、**sustack** により共通反射点ギャザー番号フィールド “cdp” 全体にわたって重合することを示します。(シェル・スクリプト **sukeyword** は利用できるキーワードの選択肢を教えてくれます。)

2.4　SUDOC - SU の項目の完全なオンライン文書のリスト

このマニュアルの前の章で述べたように、メイン・プログラム、シェル・スクリプト、ライブラリ関数のそれぞれについて利用できる自己文書化項目のデータベースがあります。このデータベースは$CWPROOT/src/doc ディレクトリ中にあり、SU のすべての項目のすべての自己文書化パラグラフから成っています。

selfdoc のある項目のすべてが実行形式というわけではないので、そのような項目の selfdoc を見るためにはそれに加えて別の機構が必要です。例えば、Abel 変換ルーチンは$CWPROOT/src/cwp/lib/abel.c (CWP のシステムでは CWPROOT=/usr/local/cwp) にありますが、その情報は次のようにして得られます。

```
% sudoc abel

In /usr/local/cwp/src/cwp/lib:
ABEL - Functions to compute the discrete ABEL transform:

abelalloc   allocate and return a pointer to an Abel transformer
abelfree    free an Abel transformer
abel        compute the Abel transform

Function prototypes:
void *abelalloc (int n);
void abelfree (void *at);
void abel (void *at, float f[], float g[]);

Input:
ns      number of samples in the data to be transformed
f[]     array of floats, the function being transformed

Output:
at      pointer to Abel transformer returned by abelalloc(int n)
g[]     array of floats, the transformed data returned by
        abel(*at,f[],g[])

Notes:
The Abel transform is defined by:

              Infinity
    g(y) = 2 Integral dx f(x)/sqrt(1-(y/x)^2)
               |y|

Linear interpolation is used to define the continuous function f(x)
corresponding to the samples in f[].  The first sample f[0] corresponds
to f(x=0) and the sampling interval is assumed to be 1.  Therefore, the
input samples correspond to 0 <= x <= n-1.  Samples of f(x) for x > n-1
are assumed to be zero.  These conventions imply that

    g[0] = f[0] + 2*f[1] + 2*f[2] + ... + 2*f[n-1]
```

```
References:
Hansen, E. W., 1985, Fast Hankel transform algorithm:  IEEE Trans. on
Acoustics, Speech and Signal Processing, v. ASSP-33, n. 3, p. 666-671.
(Beware of several errors in the equations in this paper!)

Authors:  Dave Hale and Lydia Deng, Colorado School of Mines, 06/01/90
```

sudoc が、そのルーチンについて、名前、使用方法に関する情報 (関数プロトタイプ宣言によって)、その項目の利用に関する理論、公表された参考文献、そして最後に著者の名前を含んだ情報を示していることがわかります。

別のサンプルとして、以下をタイプしてみます。

```
% sugabor

 SUGABOR -  Outputs a time-frequency representation of seismic data via
                the Gabor transform-like multifilter analysis technique
                presented by Dziewonski, Bloch and  Landisman, 1969.

    sugabor <stdin >stdout [optional parameters]

 Required parameters:
        if dt is not set in header, then dt is mandatory

 Optional parameters:
        dt=(from header)        time sampling interval (sec)
        fmin=0                  minimum frequency of filter array (hz)
        fmax=NYQUIST            maximum frequency of filter array (hz)
        beta=3.0                ln[filter peak amp/filter endpoint amp]
        band=.05*NYQUIST        filter bandwidth (hz)
        alpha=beta/band^2       filter width parameter
        verbose=0               =1 supply additional info

 Notes: This program produces a muiltifilter (as opposed to moving window)
 representation of the instantaneous amplitude of seismic data in the
 time-frequency domain. (With Gaussian filters, moving window and multi-
 filter analysis can be shown to be equivalent.)

 An input trace is passed through a collection of Gaussian filters
 to produce a collection of traces, each representing a discrete frequency
 range in the input data. For each of these narrow bandwidth traces, a
 quadrature trace is computed via the Hilbert transform. Treating the narrow
 bandwidth trace and its quadrature trace as the real and imaginary parts
 of a "complex" trace permits the "instantaneous" amplitude of each
 narrow bandwidth trace to be compute. The output is thus a representation
 of instantaneous amplitude as a function of time and frequency.

 Some experimentation with the "band" parameter may necessary to produce
 the desired time-frequency resolution. A good rule of thumb is to run
 sugabor with the default value for band and view the image. If band is
 too big, then the t-f plot will consist of stripes parallel to the frequency
 axis. Conversely, if band is too narrow, then the stripes will be parallel
 to the time axis.

 Examples:
    suvibro | sugabor | suximage
```

```
   suvibro | sugabor | suxmovie n1= n2= n3=
    (because suxmovie scales it's amplitudes off of the first panel,
    may have to experiment with the wclip and bclip parameters
   suvibro | sugabor | supsimage | ... ( your local PostScript utility)
```

この出力を、以下のようにタイプした場合の出力と比較すると:

```
% sudoc sugabor
```

ソースコードの場所を示す行に続いて、上の場合と同じ内容が出力され、追加のパラグラフが続きます。

```
 Credits:

    CWP: John Stockwell, Oct 1994

 Algorithm:

 This programs takes an input seismic trace and passes it
 through a collection of truncated Gaussian filters in the frequency
 domain.

 The bandwidth of each filter is given by the parameter "band". The
 decay of these filters is given by "alpha", and the number of filters
 is given by nfilt = (fmax - fmin)/band. The result, upon inverse
 Fourier transforming, is that nfilt traces are created, with each
 trace representing a different frequency band in the original data.

 For each of the resulting bandlimited traces, a quadrature (i.e. pi/2
 phase shifted) trace is computed via the Hilbert transform. The
 bandlimited trace constitutes a "complex trace", with the bandlimited
 trace being the "real part" and the quadrature trace being the
 "imaginary part".  The instantaneous amplitude of each bandlimited
 trace is then computed by computing the modulus of each complex trace.
 (See Taner, Koehler, and Sheriff, 1979, for a discussion of complex
 trace analysis.

 The final output for a given input trace is a map of instantaneous
 amplitude as a function of time and frequency.

 This is not a wavelet transform, but rather a redundant frame
 representation.

 References:    Dziewonski, Bloch, and Landisman, 1969, A technique
        for the analysis of transient seismic signals,
        Bull. Seism. Soc. Am., 1969, vol. 59, no.1, pp.427-444.

        Taner, M., T., Koehler, F., and Sheriff, R., E., 1979,
        Complex seismic trace analysis, Geophysics, vol. 44,
        pp.1041-1063.

        Chui, C., K.,1992, Introduction to Wavelets, Academic
        Press, New York.

 Trace header fields accessed: ns, dt, trid, ntr
 Trace header fields modified: tracl, tracr, d1, f2, d2, trid, ntr
```

sudoc リストの方が、selfdoc のリストよりも多くの情報が得られます。selfdoc はちょっとした参照用として作られていますが、sudoc のリストは、例えば、単にあるパラメータの意味を知りたいときなどには必ずしも見たいとは限らないような、追加の情報も提供しています。

2.5 SUFIND - 与えられた文字列での SU の項目の検索

特定の文字列、または、トピックスに関して “doc” データベース ($CWPROOT/src/doc にある) を検索することができます。シェル・スクリプト **sufind** はこのためにあります。オプションなしで **sufind** とタイプすると、以下が表示されます。

```
% sufind

sufind - get info from self-docs about SU programs
Usage: sufind [-v -n] string
"sufind string" gives brief synopses
"sufind -v string" verbose hunt for relevant items
"sufind -n name\_fragment" searches for command name
```

SU パッケージの中のある特定のタイプの項目の検索を支援する高度な機能があることがわかります。

この例として、高速フーリエ変換アルゴリズムを用いた SU のプログラムを探しているとしましょう。次のようにタイプして下さい。

```
% sufind fft

 FFTLAB - Motif-X based graphical 1D Fourier Transform

 Usage:  fftlab


HANKEL - Functions to compute discrete Hankel transforms

hankelalloc     allocate and return a pointer to a Hankel transformer
hankelfree      free a Hankel transformer

PFAFFT - Functions to perform Prime Factor (PFA) FFT's, in place

npfa            return valid n for complex-to-complex PFA
npfar           return valid n for real-to-complex/complex-to-real PFA

 SUAMP - output amp, phase, real or imag trace from
        (frequency, x) domain data

 suamp <stdin >stdout mode=amp

 SUFFT - fft real time traces to complex frequency traces

 suftt <stdin >sdout sign=1


 SUFRAC -- take general (fractional) time derivative or integral of
           data, plus a phase shift.  Input is TIME DOMAIN data.
```

```
 sufrac power= [optional parameters] <indata >outdata

 SUIFFT - fft complex frequency traces to real time traces

 suiftt <stdin >sdout sign=-1

 SUMIGPS - MIGration by Phase Shift with turning rays

 sumigps <stdin >stdout [optional parms]

 SUMIGTK - MIGration via T-K domain method for common-midpoint stacked data

 sumigtk <stdin >stdout dxcdp= [optional parms]

 SURADON - forward generalized Radon transform from (x,t) -> (p,tau) space.

 suradon <stdin >stdout [Optional Parameters]

For more information type: "program\_name <CR>"
```

この出力の最後の行はユーザーはキャリッジ・リターン [1] をタイプすることを示す記号で終わっています。ご覧の通り、これは、CWP/SU の prime factor FFT ルーチン (pfafft) のデモ、そのルーチンを含むライブラリ、あるいは、FFT ルーチンを用いているプログラムなどが混ざったコレクションになっています。

2.5.1　SU プログラムの情報の取得

sufind を使って DMO プログラムを探すという、もっと具体的な例です。

```
% sufind dmo

 SUDMOFK - DMO via F-K domain (log-stretch) method for common-offset gathers

 sudmofk <stdin >stdout cdpmin= cdpmax= dxcdp= noffmix= [...]

 SUDMOTX - DMO via T-X domain (Kirchhoff) method for common-offset gathers

 sudmotx <stdin >stdout cdpmin= cdpmax= dxcdp= noffmix= [optional parms]

 SUDMOVZ - DMO for V(Z) media for common-offset gathers

 sudmovz <stdin >stdout cdpmin= cdpmax= dxcdp= noffmix= [...]

 SUFDMOD2 - Finite-Difference MODeling (2nd order) for acoustic wave equation

 sufdmod2 <vfile >wfile nx= nz= tmax= xs= zs= [optional parameters]
```

[1] “キャリッジ・リターン” という文句は古い技術、タイプライターに由来するものです。詳細については両親 (あるいは祖父母) に尋ねて下さい。

```
 SUSTOLT - Stolt migration for stacked data or common-offset gathers

 sustolt <stdin >stdout cdpmin= cdpmax= dxcdp= noffmix= [...]
```

最後の 2 つの “ヒット” は偽物ですが、3 つの DMO プログラムが検索されたことがわかります。

そこで、**sudmofk** についてより詳しい情報を得るために、self-doc 機能を使います:

```
% sudmofk

 SUDMOFK - DMO via F-K domain (log-stretch) method for common-offset gathers

 sudmofk <stdin >stdout cdpmin= cdpmax= dxcdp= noffmix= [...]

 Required Parameters:
 cdpmin                  minimum cdp (integer number) for which to apply DMO
 cdpmax                  maximum cdp (integer number) for which to apply DMO
 dxcdp                   distance between adjacent cdp bins (m)
 noffmix                 number of offsets to mix (see notes)

 Optional Parameters:
 tdmo=0.0                times corresponding to rms velocities in vdmo (s)
 vdmo=1500.0             rms velocities corresponding to times in tdmo (m/s)
 sdmo=1.0                DMO stretch factor; try 0.6 for typical v(z)
 fmax=0.5/dt             maximum frequency in input traces (Hz)
 verbose=0               =1 for diagnostic print

 Notes:
 Input traces should be sorted into common-offset gathers.  One common-
 offset gather ends and another begins when the offset field of the trace
 headers changes.

 The cdp field of the input trace headers must be the cdp bin NUMBER, NOT
 the cdp location expressed in units of meters or feet.

 The number of offsets to mix (noffmix) should typically equal the ratio of
 the shotpoint spacing to the cdp spacing.  This choice ensures that every
 cdp will be represented in each offset mix.  Traces in each mix will
 contribute through DMO to other traces in adjacent cdps within that mix.

 The tdmo and vdmo arrays specify a velocity function of time that is
 used to implement a first-order correction for depth-variable velocity.
 The times in tdmo must be monotonically increasing.

 For each offset, the minimum time at which a non-zero sample exists is
 used to determine a mute time.  Output samples for times earlier than this
 mute time will be zeroed.  Computation time may be significantly reduced
 if the input traces are zeroed (muted) for early times at large offsets.

 Trace header fields accessed:  ns, dt, delrt, offset, cdp.
```

sufind を selfdoc 機能 (または **sudoc**) と連動して使い、検索に賢い文字列を選べば、SU の機能についての詳細な情報を見つけることが可能です。

2.6 GENDOCS - すべての Selfdoc を含んだ簡単な LaTeX 文書

sudoc データベースを活用する究極のシェル・スクリプトは **gendocs** です。

```
% gendocs -o
```

というコマンドは 528 ページを越える “selfdocs.tex” という LaTeX フォーマットの文書を生成します。これを LaTeX で処理することもできます。そのサイズのことを考えたとしても、きっとあなたはこの文書を印刷したいと**本当に**思うでしょう。これにはすべての CWP/SU のプログラム、ライブラリ・ルーチン、シェル・スクリプトの自己文書 (self-documentation) が含まれており、「研究室に一冊」の役立つ参考書となるでしょう。

2.7 Suhelp.html

長期に渡る SU への貢献者である University of Tulsa の Christopher L. Liner 博士が以下のドキュメントを作ってくれました。そこへは CWP/SU のウェブ・サイトから、あるいは以下の彼のサイトからアクセスできます:

```
http://douze.utulsa.edu/~cll/suhelp/suhelp.html
```

```
                                SeismicUn*x

                            Version 33 (5 April 1999)

                              An HTML Help Browser

   * This is a help browser for the SeismicUn*x free software package
     developed and maintained by the Center for Wave Phenomena at the
     Colorado School of Mines. TheSUproject is directed at CWP by John
     Stockwell.
   * The author of this help facility is Dr. Christopher Liner (an alumnus
     of CWP) who is a faculty member in the Department of Geosciences at The
     University of Tulsa.
   * Last updated January 16, 1998



        o The arrangement below is by funtionality
        o Clicking on a program name pulls up the selfdoc for that item
        o Your web browser's Find capability is useful if you have a
          fragment in mind (e.g. sort or nmo)
        o While programs may logically apply to more than one catagory
          below, each program appears only once
     ------------------------------------------------------------------------

  1. Functional Listing
       1. Data Compression
       2. Editing, Sorting and Manipulation
       3. Filtering, Transforms and Attributes
       4. Gain, NMO, Stack and Standard Processes
       5. Graphics
       6. Import/Export
       7. Migration and Dip Moveout
       8. Simulation and Model Building
```

```
      9. Utilities

   * Alphabetical name list            * 258 items
   * List is for scanning only, it     * For further info on an
     does not pull up the selfdoc of     interesting item use your
     a selected item.                    browser's find command, then
                                         follow the link

   ------------------------------------------------------------------------
 Data Compression

   Discrete Cosine Transform
                                * dctcomp          * dctuncomp
   Packing                      * supack1          * suunpack1
                                * supack2          * suunpack2
                                * wpc1comp2        * wpc1uncomp2
   Wavelet Transform            * wpccompress      * wpcuncompress
                                * wptcomp          * wptuncomp
                                * wtcomp           * wtuncomp

                               Go to top
...
```

全部のリストを見るには、付録 B を見るか、ブラウザで上記の http アドレスを見て下さい。【訳注：上記サイトは本文書作成時（2015 年 3 月）には無効になっていました。CWP/SU のサイトからは下記にリンクが貼られています。

http://sepwww.stanford.edu/oldsep/cliner/files/suhelp/suhelp.html

こちらは 2015 年 3 月現在で有効です。】

2.8 デモ

SU パッケージにはデモ一式が含まれています。デモは$CWPROOT/src/demos ディレクトリ内にあるシェル・スクリプトです。

デモの実行についての指示は$CWPROOT/src/demos/README にあります。

- Making_Data デモは susynlv を使って合成地震データのショットギャザーおよび共通オフセット断面を作る基本を示しています。上手な表示のラベル付けを例示するように特に注意が払われています。
- Filtering/Sufilter デモは地動（グラウンド・ロール）と初動を除去する実際のデータ処理について示しています。Filtering の下位ディレクトリ中のデモには CWP の ftp サイトからデータにアクセスするための指示があります。
- Deconvolution デモは、簡単な合成スパイク・トレースを用いて、supef やその他のツールを用いた残響の除去とスパイキング・デコンを図示します。デモには、ループを用いてフィルタ・パラメータの影響を系統的に調べるコマンドが含まれています。
- Sorting_Traces チュートリアルは会話型のスクリプトで、この文書で論じられている基本的な UNIX と SU の知識を補充するものです。会話性はペースを設定するだけに限られています。このようなチュートリアルにはすぐにイライラしてしまいますが、標準のデモの形式に適しないいくつかの項目をカバーするために、こういったデモが

一つは必要であると考えました。このトピックについては、もう一つ、標準的ですが未完成のデモがあります。

- 次のステップは Selecting_Traces デモを実行することです。その次に NMO デモに進んで下さい。

何よりも、興味のあるデモのディレクトリをのぞいてみて下さい。**demos** ディレクトリ・ツリーはまだ活発に開発中です—デモが役に立つか、どのように改良したらよいか知らせて下さい。

2.9　その他のヘルプの機能

- SUKEYWORD - SU ヘッダ・フィールドのキーワードをリストする。ヘッダ・フィールドの情報を用いる多くの SU プログラムでは、selfdoc 中に “keywords” への参照とともに “key=” パラメータが列挙されています。SU のキーワードは SEG-Y トレース・ヘッダ・フィールドに基づいています。（これについては後のテープからの読み込みとデータ・フォーマット変換の節で説明します。）これらのヘッダ・フィールドが何かを知り、SU データで何を表しているかを知るには、

  ```
  % sukeyword -o
  ```

  と入力して下さい。

  すべての出力テキストは付録 B を見て下さい。また、このコマンドの使用方法の詳細は第 3.5.3 節を見て下さい。多くのプログラムが SU ヘッダ・フィールドの値を利用して、データをソートし、ウインドウし、表示しますから、**sukeyword** はもっとも使われるユーティリティの一つになっています。
- 本質的に SU の使用方法は、コーディネートされたデータ処理を実行するためのシェル・プログラムを作ることです。**su/examples** ディレクトリにはそのような多くのプログラムが含まれています。ところで、“シェル・スクリプト”、“シェル・プログラム”、“シェル・ファイル”、“シェル” という用語は UNIX の文脈では互換性があります。
- **faq** ディレクトリには SU に関するよく聞かれる質問への回答集があります。そこには、テープの読み込みや、データ・フォーマットの質問、地震探査データ処理のちょっとした役立つ情報があります。
- John A. Scales の教科書「**弾性波イメージングの理論** *(Theory of Seismic Imaging)*」(Samizdat Press, 1994) が 300 および 400 ドット/インチの PostScript フォーマットで、以下の私達の anonymous ftp サイトから利用できます: `pub/samizdat/texts/imaging/imaging_300dpi.ps.Z` または `imaging_400dpi.ps.Z`。教科書の演習には SU が広範囲にわたって用いられています。

ソース・コードそのものを読んでみることをためらってはいけません。第 11.2 節では重要な SU コーディング語法について説明されています。ソースとその説明文との間に間違いを見つけたら私達に連絡して下さい。また、コードのコメントやスタイルを改良する提案も歓迎します。SU パッケージの強みは、それがソース・コードで提供されることです。当たり前のことですが、商用パッケージではソース・コードは入手できません。

SU に関するコメント、質問、意見があれば、あるいは、パッケージに貢献するあなたの SU 形式のプログラムがあれば、john@dix.mines.edu まで直接メールを下さい。

第3章　核となるSeismic Unixプログラム

核となるSeismic Unixプログラム群は幅広い作業を実行します。それらの作業は研究と処理の多くの分野に共通していると見ることもできるかもしれません。しかし、このプログラムの多くは純粋に地震探査用のものであり、そういうものとして本文中で示しています。

そのような作業には以下のものが含まれます:

- 入力/出力
- データのフォーマット変換
- トレース・ヘッダー・フィールドの設定、表示、編集
- SUデータの表示
- データのウィンドウニング、並び変え、編集
- 一般的操作
- 変換およびフィルタ操作
- SUデータの地震探査処理操作

以下の章ではこれらの事項の多くを取り扱います。ここで、どのプログラムも、より詳しい情報はコマンドライン上で引数なしでプログラム名をタイプして:

```
% programname
```

あるいは以下のようにタイプすることによって:

```
% sudoc programname
```

selfdoc情報を見ることによって得られることを忘れないで下さい。

この章では、SUプログラムをすべて網羅していませんが、UNIXの経験のある人がパッケージを使用開始するのに十分なだけの数を網羅しています。一旦どのようにSUを使うかがわかれば、パッケージ中の他のプログラムを探すのにヘルプ機能を頼ることができます。

3.1　テープからのデータの読み込みおよびテープへの書き込み

テープの読み込みは科学というより技術 (more of an art than a science) です。【訳注：理論よりも技？】これは一般的に正しいですが、特にSUの場合にはまったくその通りです。ハードウェア・フォーマットもデータ・フォーマット型も同じように変わりやすいので、“一般的なテープ読み込みユーティリティ”を作るということはやりがいのある（難しい）提案です。不可能でなければの話ですが。

以下のプログラムは、内部SUデータ・フォーマットと同様、地球物理学的アプリケーションに関する特別なデータの入出力作業に役立ちます。

- BHEDTOPAR - バイナリ・テープ・ヘッダ・ファイルのPARファイル・フォーマットへの変換

- DT1TOSU - Sensors & Software社のX.dt1 GPRフォーマットの地中レーダー・データのSUフォーマットへの変換
- SEGDREAD - SEG-Dテープの読み込み
- SEGYCLEAN - ヘッダの未割り当て部のデータのゼロ化
- SEGYREAD - SEG-Yテープの読み込み
- SEGYHDRS - **segywrite** のためのSEG-Yアスキーおよびバイナリ・ヘッダの作成
- SEGYWRITE - SEG-Yへの書き込み
- SETBHED - SEG-Yバイナリ・テープ・ヘッダ・ファイル中のフィールドの設定
- SUADDHEAD - トレースへのヘッダの付加と'tracl'および'ns'フィールドの設定
- SUSTRIP - トレースからのSEG-Yヘッダの除去
- SUPASTE - 既存のSEG-Yヘッダの既存のデータへの張りつけ

以下のプログラムは、一般的なデータ入力、出力、データ型変換に役立ちます。これらもテープからの読み込みに使えることがあります。

- A2B - アスキー浮動小数点データのバイナリへの変換
- B2A - バイナリ浮動小数点データのアスキーへの変換
- FTNSTRIP - Fortranの浮動小数点データのCのスタイルの浮動小数点への変換
- H2B - 8ビット16進数浮動小数点データのバイナリへの変換
- RECAST - データ型のリキャスト (あるデータ型から別の型への変換)
- TRANSP - n1×n2要素マトリックスの転置

3.1.1 SEG-Yフォーマットと SUデータ・フォーマット

名前が'su'という文字で始まっているCWP/SUパッケージのすべてのプログラム (プログラム **subset** は例外) が期待するデータフォーマットは"プログラムが動作するマシン本来の (native) バイナリ・フォーマットで書かれたSEG-Yトレース" からなっています。このフレーズの意味を理解するには、SEG-Y標準とは何かを理解しなければなりません。

1980年代の始め、最も広く使われていたデータ・フォーマットはSEG-Yでした。これは、the Society of Exploration Geophysicists Yフォーマットのことで、SEGの出版物「**デジタル・テープ標準** *(Digital Tape Standards)*」に記述されています。このフォーマットは"規則に従って (by the book)" 使用されているという保証はないものの、今日に至ってもまだ広く使われています。

SEG-Yデータ・フォーマットは3つのパートからなっています。最初のパートは3200バイトのEBCDICカード・イメージ・ヘッダで、テープを記述するテキスト・データの40枚のカード (すなわち1行80文字、40行のテキスト) です。第2のパートは400バイトのバイナリ・ヘッダで、テープ・リールの内容についての情報が含まれています。SEG-Yフォーマットの第3のパートは実際の地震トレースからなっています。それぞれのトレースには240バイトの**トレース・ヘッダ**があります。その後に、IBM Form GA 22-6821で定義されたIBM浮動小数点表示の32のフォーマットのうちの4つの可能なフォーマットのうちの一つ (one of 4 possible 32 formats) で書かれたデータが続きます。(この"IBMフォーマット" は現在のIBM PCに見られる共通のIEEEフォーマットではないことに注意して下さい。)

SU データ・フォーマットは SEG-Y フォーマットのトレース部に基づいています。SEG-Y トレースと SU トレースの主な違いは、SU フォーマットのデータ部は SU を走らせているマシン本来のバイナリ浮動小数点フォーマットで書かれた浮動小数点実数データであることです。SU データは SEG-Y トレース**のみ！**からなっています。SU フォーマットでは EBCDIC とバイナリのリール・ヘッダは保存されません。ですから、単に SEG-Y ファイルをリダイレクトするだけでは SU のプログラムは動作しません。

SEG-Y データを SU のプログラムで使用できる形式に変換するには、**segyread** を使う必要があります。

3.1.2 SEGYREAD - SEG-Y データの SU への取り込み

データを SEG-Y フォーマットから SU フォーマットへ変換するためには **segyread** プログラムを用います。

```
% segyread
```

とタイプすると、このプログラムの selfdoc を見ることができます。

SEG-Y テープ、あるいはデータファイルを読み込む際には、走らせているマシンのバイト順 (エンディアン) について知っておく必要があります。いわゆる “ビッグ・エンディアン (big-endian)” あるいは上位バイト (high-byte) IEEE フォーマットは SGI、SUN、IBM RS6000 とすべての Motorola チップに基づくシステムに見られます。“リトル・エンディアン (little-endian)” あるいは下位バイト (low-byte) システムは Intel と Dec のチップに基づくシステムです。また、テープ・ドライブが何という UNIX のデバイスかも知っておかなければならないでしょう。

ビッグ・エンディアンのマシンでの典型的な **segyread** の実行はこのようになります:

```
% segyread tape=/dev/rmt0 verbose=1 endian=1 > data.su
```

以下の方法を用いなければならないことが多いでしょう。

```
% segyread tape=/dev/rmt0 verbose=1 endian=1 | segyclean > data.su
```

SEG-Y トレース・ヘッダにはオプションのヘッダ・フィールド (181-240 バイト) があります。これらのフィールドに何を入れるかの標準はありません。そこで、多くの人が自分達自身の目的のためにこれらのフィールドに置く項目を設けています。SU もその例外ではありません。SU のグラフィック・プログラムが使う 2、3 のパラメータがこれらのフィールドに保存されることがあります。**segyclean** プログラムはオプションのヘッダ・フィールドの値を消去するので、SU のグラフィック・プログラムはこれらの情報に惑わされることがありません。

その他にも、実際にテープを読み込もうとするときに試さなければならないかもしれない項目があります。例えば、デバイスがバッファされるテープかバッファされないテープか (すなわち、9 トラック 1/2 リール・テープか 8mm Exabyte か) などです。テープが作られたシステムと違うシステムでテープを読みだそうとしている場合に、単にテープを読み込めないということが起こり得ます。

テープの読み込みの際に最もよく起こる問題は、テープが書き込まれた記録密度とテープを読み込もうとしているテープドライブのマッチングです。例えば、Silicon Graphics (SGI) システムのようないくつかのシステムには多くのテープ・デバイスがあり、異なるハードウェア設定とテープ密度をサポートしています。その他のシステムには、最もはっきりして

いるのは最新のバージョンの Linux ですが、“setdensities” オプションのある UNIX の “mt” コマンドの改良バージョンがあります。どちらの場合にも、テープを作るにはテープ・ドライブのデフォルトの設定、また、デフォルトの記録密度を用いるのが普通です。

テープ読み込みに関するいかなる状況においても最後の頼みの綱として、UNIX のデバイスからデバイスへのコピー・プログラム **dd** を使ってテープ全体のイメージをディスク上に作ることが可能です。

```
% dd if=/dev/rmtx of=filename bs=32767 conv=noerror
```

ここで、“/dev/rmtx” はあなたのテープドライブに、“filename” はあなたの選んだファイル名に置き換えて下さい。もし、これがうまくいけば、次のステップは “tape=filename” として前述したように **segyread** を使ってみることです。もし、**dd** がうまくいかなければ、あなたのテープドライブのハードウェア・フォーマットはテープと互換性がないと考えてよいようです。

もちろん、テープの読み込みの際の問題を防ぐ最良の方法は、テープに書き込む前に書き込みを行う人と話をすることです。特に、SGI のシステムではテープ・フォーマットの可能な選択肢が多いので、読み込み可能が保証されるテープを作る以前に、テープを作る人はテープを読み込もうとするプラットフォームの情報を持っていないといけません。

3.1.3　SEG-Y の乱用

不幸なことに、“SEG-Y” と呼ばれてはいるものの、SEG-Y に関する SEG の標準に照らして正しくないフォーマットが存在します。よくあるバリエーションの一つは、大部分の SEG-Y の取り決めを尊重しているものの、トレースが IEEE フォーマットであるというものです。

そのようなデータは以下のようにして読むことができます:

```
% segyread tape=/dev/rmt0 verbose=1 endian=1 conv=0 | segyclean > data.su
```

ここで、“conv=0” はプログラムに IBM から浮動小数点実数への変換を行わないことを告げるためのものです。

DOS SEG-Y

“DOS SEG-Y” フォーマットというものもあり、これは先のフォーマットと似ていますが、トレースとヘッダがすべてリトル・エンディアン・フォーマットで書かれている点が違います。ビッグ・エンディアンのマシンでは、そのようなデータを読むコマンドは次のようになります。

```
% segyread tape=/dev/rmt0 verbose=1 endian=0 conv=0 | segyclean > data.su
```

ここで、endian=0 はバイトをスワップするために設定します (すべてのバイト、ヘッダおよびデータはスワップしたフォーマットになります。)。リトル・エンディアンのマシンでは、処理は次のようになります。

```
% segyread tape=/dev/rmt0 verbose=1 endian=1 conv=0 | segyclean > data.su
```

endian=1 として、この場合はバイト・スワップを防ぎます。

それぞれの場合、ファイル名が ‘filename’ のディスクファイルから読み込む場合は “tape=filename” を使って下さい。

SEISWORKS の Landmark BCM2D フォーマット

商用の地震探査ソフトウェアも非標準の SEG-Y フォーマットの出所となります。公式のヘッダ・フィールドの非標準使用に加えて、商用の SEG-Y のバリエーションはオプションの SEG-Y ヘッダ・フィールドの一部を定義していることがあります。

Virginia Tech の Matthias Imhoff によると、この問題の治療法は、**segyread** のリマッピング機能で、非標準のフィールドを SU ヘッダ中の互換性のある場所にリマップすることができるます。“remap=” オプションにより目的地の SU ヘッダ・フィールドを指定し、“byte=” で SEG-Y トレース・ヘッダの位置とそのデータ型を指定します。

Landmark BCM2D の例だと、ヘッダ・フィールドの 73 と 77 は浮動小数点型ですが、標準の SEG-Y フォーマットでは整数型で、そのため SU フォーマットでも整数型です。BCM2D には long 型に設定されたヘッダ・フィールドも 181 と 185 バイト目にあります。以下の **segyread** の使用で、

```
% segyread tape=... remap=d1,d2,gelev,selev byte=73f,77f,181l,185l > ...
```

73 と 77 の浮動小数点は d1 と d2 にマップされ、一方、181 と 185 の long 型の整数は gelev と selev、SU フォーマットでは整数型です、にマップされます。互換の目的フィールドを選ぶことで、精度は失われません。

3.1.4 SEGYWRITE - SEG-Y テープ、ディスクファイルへの書き込み

segyread と対になるプログラムは **segywrite** です。このプログラムはテープあるいはディスクファイルに SEG-Y フォーマットで書き込む際に若干のバリエーションを許容します。このプログラムは、データを商用の地震探査パッケージで読み込むことができる形式に変換する際に役立ちます。**segywrite** がどのように使われるかサンプルを示す前に、実際のテープの書き込みの準備を行う際に説明しなければならない、いくつかの前処理のステップがあります。

3.1.5 SEGYHDRS - segywrite のための SEG-Y アスキーおよびバイナリ・ヘッダの作成

SEG の “Digital Tape Standards” の本で指定されている通りの SEG-Y フォーマットでテープを書き込むには、アスキーおよびバイナリのテープ・リールヘッダ・ファイルを与える必要があります。それらのファイルは SEG-Y テープあるいはファイル中では EBCDIC およびバイナリのテープ・リールヘッダ・ファイルになります。これらは **segywrite** 【訳注 : **segyread** の誤り？】 によって作成される “header” および “binary” というファイルです。

“binary” および “header” ファイルがなければ、**segyhdrs** プログラム (SEG Y headers と読みます) を用いてそれらを作成する必要があります。

```
% segyhdrs < data.su
```

というコマンドによって現在のワーキング・ディレクトリに “header” と “binary” というファイルが作成されます。例として、**suplane** を使ってテスト・データを作り、**segyhdrs** を走らせてみます。

```
% suplane > data.su
% segyhdrs < data.su
```

binary と *header* というファイルが現在のワーキング・ディレクトリに作成されたことがわかるでしょう。

このプログラムにはバイナリ・ヘッダ・フィールドの値を設定することができるオプションがあります。これらのフィールドは以下のようにタイプすることによって見ることができます:

```
% sukeyword jobid

        int jobid;      /* job identification number */

        int lino;       /* line number (only one line per reel) */

        int reno;       /* reel number */

        short ntrpr;    /* number of data traces per record */

        short nart;     /* number of auxiliary traces per record */

        unsigned short hdt; /* sample interval in micro secs for this reel */

        unsigned short dto; /* same for original field recording */
...
```

ここで、"jobid" はバイナリ・ヘッダの最初のフィールドです。

"header" ファイルはASCIIファイルで、通常のテキスト・エディタで編集することもできます。1行80文字で40行以内であれば、何を書いても構いません。**segywrite** は自動的にこれを変換します。**segyhdrs** が生成するデフォルトのヘッダ・ファイルはこのようになります。

```
C      This tape was made at the
C
C      Center for Wave Phenomena
C      Colorado School of Mines
C      Golden, CO, 80401
C
...
C
C
```

3.1.6 BHEDTOPAR, SETBHED - バイナリ・ヘッダ・ファイルの編集

バイナリ・ヘッダ・ファイルを編集するためにはASCII形式に変換しなければなりません。**bhedtopar** プログラムを用いると、"binary" ファイルを "parfile" の形式で出力することができます。

```
% bhedtopar < binary outpar=binary.par
```

テストSUデータ用に作られたヘッダ・ファイルの場合には、以下のようになります。

```
jobid=1
lino=1
reno=1
ntrpr=0
nart=0
```

```
hdt=4000
...
```

さまざまなヘッダ・ファイルに与えられた値を編集することができ、それらの値を **setbhed** を用いてヘッダ・ファイルに再読み込みすることができます。

```
% setbhed bfile=binary par=binary.par
```

個々のヘッダ・フィールド値も設定することができます。例えば、

```
% setbhed bfile=binary par=binary.par lino=3
```

このようにするとヘッダ値に binary.par の内容を用いますが、lino フィールドは 3 に設定されます。

最後に、以下のようなコマンド列でテープに書くことができます。

```
% segywrite tape=/dev/rmtx verbose=1 < data.su
```

“header” と “binary” というファイルがカレント・ディレクトリ内にあることに気をつけて下さい。これらのファイルに違う名前を用いることもできます。**segywrite** には “bfile=” と “hfile=” というオプションがあり、選択した違う名前のファイルを入力することができます。

3.1.7 SEGDREAD - その他の SEG フォーマット

SEG-Y の他にも SEG フォーマットがあります (SEG-A, SEG-B, SEG-X, SEG-C, SEG-D, SEG-1, SEG-2)。中でも SEG-D, SEG-B, SEG-2 はよく見かけるタイプです。**segdread** というプログラムがありますが、SEG-D の多くのバリエーションの内の 1 つしかサポートしていません。

$CWPROOT/Third_Party/ディレクトリ内に SEG-2 フォーマットを SEG-Y に変換する **seg2segy** 変換プログラムがあります。**seg2read** プログラムを作る将来計画はあります。これは **segyread** や **segdread** とよく似たものになるでしょう。

3.1.8 DT1TOSU - SEG 以外のテープ・フォーマット

現在のところ、SU パッケージが完全にサポートしている SEG 以外のテープ・フォーマットは一つしかありません。また、パッケージに統合されていないサード・パーティ製のコードを通してサポートしているものが他に 2 つあります。最初のものは、**dt1tosu** を用いた Sensors & Software の DT1 フォーマットがそれです。これは GPR (地中レーダー) のフォーマットです。また、$CWPROOT/src/Third_Party ディレクトリには SEG 以外のデータの変換プログラムが 2 つあります。それらは **segytoseres** と **bison2su** です。将来的にはこれらのコードをメインの CWP/SU パッケージに含めることを計画しています。

3.2 データ・フォーマットの変換

データを他のシステムに変換したり、さまざまなフォーマットのデータを入力する必要があることがしばしばあります。SU ではこれらの事項を扱う多くのツールや仕掛けが利用できます。

以下のプログラムはそのような変換上の問題に役立つでしょう。

- A2B - アスキー浮動小数点実数をバイナリへ変換
- B2A - バイナリ浮動小数点実数をアスキーへ変換
- FTNSTRIP - Fortranの浮動小数点実数データをCの浮動小数点実数へ変換
- H2B - 8ビット16進数浮動小数点実数をバイナリへ変換
- RECAST - データ型のリキャスト (あるデータ型から他への変換)
- TRANSP - n1×n2要素マトリックスの転置
- FARITH - ファイル演算—バイナリ・ファイルの簡単な演算の実行
- SUADDHEAD - トレースへのヘッダの付加と'tracl'および'ns'フィールドの設定
- SUSTRIP - トレースからSEG-Yヘッダの除去
- SUPASTE - 既存のSEG-Yヘッダの既存のデータへの張りつけ
- SWAPBYTES - さまざまなデータ型のバイトのスワップ
- SUSWAPBYTES - ビッグ・エンディアンからリトル・エンディアンへ、あるいはその逆の変換を行うためのSUデータのバイトのスワップ

この節では、これらのプログラムを用いることができる状況について説明します。

3.2.1 A2B, B2A - アスキーからバイナリへ、バイナリからアスキーへ

あらゆるデータのフォーマットの中で、最も可搬性がある (そして容量を消費する) のはアスキー (ASCII) です。どのようなシステム上で作業していても、アスキー・データから、あるいはアスキー・データへ変換することは可能です。また、テキスト・エディタはアスキーをサポートしているので、通常どんなシンプルなテキスト・エディタでもデータ・エントリーやデータそのものを編集できます。

そういったデータはおそらく複数の列のフォーマットになっていて、空白またはタブで区切られています。このような、例えば5列のデータセットをバイナリ浮動小数点実数に変換するには、以下のようにタイプして下さい:

```
% a2b < data.ascii n1=5 > data.binary
```

逆の操作はこのようになります。

```
% b2a < data.binary n1=5 > data.ascii
```

3.2.2 FTNSTRIP - FortranデータのCへのインポート

Fortranは地震探査データ処理ではよく使われる言語なので、Fortranを用いて作成された、あるいは、処理されたデータがしばしばあります。Fortranのバイナリ・データはレコード先頭 (beginning-of-record) デリミタおよびレコード終端 (end-of-record) デリミタで区切られています。Cプログラムで作成されたバイナリ・データにはこのようなデリミタはありません。CプログラムでFortranデータを使うためには、以下のようにしてFortranのラベルを取り去る必要があります。

```
% ftnstrip < fortdata > cdata
```

これによりCのスタイルの浮動小数点実数が作成されます—たいていの場合。このプログラムは、Fortranデータの各レコードの先頭と末尾にはレコードのサイズをバイト単位で表す整数が付加されていると仮定しています。以前は、これらの整数マーカーが片一方にしかないFortranデータのタイプもありました。しかし、今では、レコード先頭 beginning-of-record (BOR) およびレコード終端 end-of-record (EOR) マーカーの両方があるのが標準のようです。

3.2.3 C から Fortran へ

Fortranコードで作られたデータをCへ移すのはかなり簡単なのですが、逆の方向はそれほど簡単ではありません。SGI Power Challengeでは、“infile” というCの浮動小数点実数ファイルをopen文とread文を用いて以下のように読むことができます:

```
OPEN(99,file='infile',form='system')

DO i=1,number
    READ(99) tempnumber
    Array(i)=tempnumber
END DO

CLOSE(99)
```

”form=’system’” という文はすべてのマシンでうまくいくとは限りません。というのも、これは標準のFortranではなさそうだからです。バイナリを読む際の一般的なフォーマット・コマンドは”form=’unformatted’”です。これは他のシステムではうまくいかない可能性があります (例えば、SUN)。実際、(SUのような) Cプログラムで作成されたバイナリ・ファイルをFortranで読めるということは一般に保証されていません。

バイナリの問題が生じた場合、入力ファイルがあまり大きくなければ、(’b2a’ を用いて) アスキーに変換し、書式つき入出力 (formatted I/O) を用いることができます。

```
OPEN(99,file='infile')

DO i=1,number
    READ(99,*) tempnumber
    Array(i)=tempnumber
END DO

CLOSE(99)
```

FTNUNSTRIP - C バイナリ浮動小数点の Fortran 形式浮動小数点への変換

しかし、別の可能性として、Cプログラムで “にせの”Fortran浮動小数点データを作る方法があります。このようなプログラムが **ftnunstrip** です。

このプログラムは、レコード長が入力および出力ファイルを通じて一定であると仮定しています。これらの浮動小数点データを読んでいるFortranコード中で、下に示されているDOループ文法が使われているとします。

```
DO i=1,n2
    READ (10) (someARRAY(j), j=1,n1)
END DO
```

ここで n1 はレコードあたりのサンプル数、n2 はレコード数、10 は form='unformatted' の形式でオープンされるデフォルトのファイル (例えば、fort.10) です。ここで "someARRAY(j)" はサイズを n1 として配列宣言された配列です。

BOR および EOR マーカーを付けるという Fortran 形式は、適切に用いられれば賢明ですが、不適切に用いられればバカげていることに注意して下さい。Fortran の READ 文は BOR マーカーから後ろに何バイト続いているかを知り、バイト数を数えながら指定された数の値を読み込みます。読み終わると、読み込まれたバイト数と EOR 値に記録されたバイト数とを比較し、読み込んだバイト中のエラーや早すぎる EOR を効果的にトラップすることができます。

余分の "sizeof(int)" (通常 4 バイト) が各レコードの最初と終わりにあるので、レコード数を必要な限り少なくするのが賢明です。最悪のケースはすべてのデータ値がレコードであるという場合です。この場合はファイルのサイズの 2/3 が BOR および EOR マーカーで、データは 1/3！しかありません。

3.2.4　H2B - 8 ビット 16 進数のインポート

8 ビット 16 進数を変換することはあまりないように思えます。しかし、8 ビット 16 進数はビットマップ・イメージ (グレイスケール PostScript) に共通のフォーマットで、もしスキャンしたイメージを取り込んで、さらに処理のために浮動小数点実数に変換したい場合に、こういったことが生じます[1]。

スキャンしたイメージがあり、256 諧調グレー・スケールのビットマップ化 PostScript イメージであるならば、ビットマップ部は 8 ビット 16 進数です。すべての PostScript コマンドを取り除いてビットマップのみを残し、以下のコマンド：

```
% h2b < hexdata > floatdata
```

を用いれば、ビットマップは CWP/SU パッケージのプログラムで表示したり、処理することができる形式に変換されます。

3.2.5　RECAST - C バイナリ・データ型の変換

もちろん、C はさまざまなデータ型をサポートしています。それぞれの型からそれぞれ違う型へ変換するたくさんのプログラムを持つのではなく、これらのたくさんの型変換に関する作業を行う "recast" というプログラムがあります。

recast の入出力としてサポートされているのは:

- float - 単精度浮動小数点実数
- double - 倍精度実数
- int - (符号つき) 整数
- char - 文字
- uchar - 符号なし char
- short - 短い整数

[1] このプログラムは元々以下の経緯でできました。ある学生が誤ってオリジナルのデータ・セットを壊してしまい、ビットマップ化された PostScript イメージのみが残されました。**h2b** を使って PosScript ファイルからデータを回復することができました。

- long - 長い整数
- ulong - 符号なし長い整数

例えば、整数を浮動小数点実数に変換するには、

```
% recast < data.ints  in=int out=float > data.floats
```

このプログラムの名前は、C 言語の明示的型変換が “cast” と呼ばれていることに由来しています。

3.2.6 TRANSP - バイナリ・データの転置

何らかの操作中に、データセットの転置が必要になることがよくあります。特に、複数列の ASCII フォーマットで表されているデータは、**a2b** を用いて ASCII からバイナリへ変換した後、転置する必要があります。その理由は、地震探査データ処理のためにはデータの最初のディメンジョンが時間 (あるいは深度) である方が都合がよいからです。

先に用いたサンプルは 5 列のデータセットでしたが、これを、ファイル中で隣合った、各トレースで同じサンプル数をもつ 5 本の地震トレースと考えることもできます。一連の処理 :

```
% a2b < data.ascii n1=5 > data.binary
```

によって、ファイル中の最初のディメンジョンがサンプル数方向ではなくトレース番号である結果が得られます。続く一連の処理 :

```
% transp < data.binary n1=5 > data.transp
```

では最初のディメンジョンがサンプル数方向である望み通りの形式にデータが出力されます。それによって各トレースに連続的にアクセスできます。

3.2.7 FARITH - バイナリ・データの簡単な演算の実行

バイナリ・データのファイルに対して、あるいは 2 つのファイル間で演算操作を行う必要がしばしばあります。これらの作業の多くをこなすように **farith** プログラムが提供されています。

farith がサポートしている単一ファイルに対する演算には次のようなものがあります。

- 値のスケーリング
- 極性反転
- 符号関数
- 絶対値
- 指数関数
- 対数
- 平方根
- 二乗
- 逆 (打ち切り)
- 逆二乗 (打ち切り)

- 逆平方根 (打ち切り)

バイナリ操作 (2 つのファイル間の操作) には次のようなものがあります。値と値ごとの

- 和
- 差
- 積
- 商
- カルテシアン積 (内積)

地震探査データ用の操作 (ファイルが波動の速度からなっていると仮定する) には次のようなものがあります。

- スローネス (slowness) の摂動 (ファイルの逆数の差)
- sloth 【訳注 : slowness の二乗】 の摂動 (ファイルの逆数の差)

farith を用いたサンプルを以下に示します。

```
% farith in=data.binary op=pinv out=data.out.bin
% farith in=data1.binary in2=data2.binary op=add > data.out2.bin
```

3.3　トレース・ヘッダ操作

SU のデータ・フォーマットは SEG-Y データの地震トレース・ヘッダを継承しています。データが SEG-Y ではなく他のタイプからの変換によって作成されている場合、最低限設定する必要のあるヘッダがあります。

この節で取り扱う事項は以下の通りです。

- トレース・ヘッダの付加
- トレース・ヘッダの削除
- トレース・ヘッダの貼りつけ

3.3.1　SUADDHEAD - SU ヘッダのバイナリ・データへの付加

データが正しい形式、すなわち、C 形式の浮動小数点実数で最初のディメンジョンがトレースごとにサンプル番号が増加していく方向であるように構成された配列で作成されたら、他の SU プログラムでアクセスできるように、データにヘッダを付ける必要があります。

データが SEG-Y, SEG-D, DT1, Bison/Geometrics のデータである場合、テープあるいはディスクファイルから読み込めば、それぞれ **segyread**, **segdread**, **dt1tosu**, **bison2su** の各プログラムがトレース・ヘッダ値を設定するので、出力は “SU データ” となります。

他の方法で読み込まれたデータ、例えば、**a2b** で読み込んだ ASCII データセットなどの場合にはヘッダを付ける必要がありますが、これは **suaddhead** を用いて行うことができます。データセットがバイナリの C 形式浮動小数点実数のファイルで、各トレースが例えば 1024 サンプルの場合、以下のコマンド列

```
% suaddhead < data.bin ns=1024 > data.su
```

によって SU データファイル “data.su” が作成されます。

3.3.2 SUSTRIP - SU ヘッダの SU データからの削除

suaddhead の逆の操作が **sustrip** です。SU ヘッダが付いているデータがあり、そのデータを SU ヘッダを理解しない別のプログラムにエクスポートしたいことがあります。以下のコマンド列：

```
% sustrip < data.su head=data.headers > data.bin
```

は SU ヘッダを削除し、それらを “data.headers” ファイルに保存します。

3.3.3 SUPASTE - SU ヘッダのバイナリ・データへの貼りつけ

バイナリ・データに何らかの操作を行った後、再度ヘッダを貼りつけたいこともあります。これは **supaste** によって行うことができます。以下のコマンド列：

```
% supaste < data.bin head=data.headers > data.su
```

は “data.headers” に含まれるヘッダをデータに貼りつけます。

3.4 バイト・スワップ

最上の人間の意図でさえ出し抜かれることがあります (Even the best of human intentions can be circumvented.)。IEEE 浮動小数点データ・フォーマットはまさにそれです。チップ製造会社による IEEE 標準の実装は、広く見られる 2 種類のデータ型を生むという結果になりました。それらは “ビッグ・エンディアン (big-endian)” (上位バイト: high-byte) もしくは “リトル・エンディアン (little-endian)” (下位バイト: low-byte) と呼ばれています。リトル・エンディアンのマシンは Intel あるいは Dec ベースのもので、一方、他のチップ製造会社のものはビッグ・エンディアンです。

データを読み込むためには (通常の浮動小数点データ、または SU データのどちらかのデータ) を移動する際にバイトをスワップ (交換) する必要があります。2 つの事項について触れる必要があります。それらは、

- 通常の浮動小数点データ
- SU データ

のバイトのスワップに関する事項です。

これらの作業のために 2 つのプログラムが用意されています。それらは、

- SWAPBYTES - さまざまなデータ型のバイトのスワップ
- SUSWAPBYTES - SU データのビッグ・エンディアンからの変換を行うためのバイトのスワップ

です。

3.4.1 SWAPBYTES - バイナリ（非SU）・データのバイト・スワップ

バイナリ・データ (SUヘッダのないデータ) を自分が使っているプラットフォームと “エンディアン”(すなわち、IEEEバイト順) の異なるプラットフォームへ移動する場合、そのデータを使うにはバイトをスワップしなければなりません。

このような問題が生じるのは、IEEEデータ標準の元でバイナリ・データの仮数部と指数部の順序に2つの異なる順序が可能であるからです。いわゆる “ビッグ・エンディアン” または上位バイトIEEEフォーマットはSGI, SUN, IBM RS6000およびMotorolaチップをベースとしたシステムに見られます。“リトル・エンディアン” または下位バイト・システムはIntelおよびDecチップをベースにしたシステムです。

swapbytes プログラムはSUデータ以外のさまざまなデータ・フォーマット・タイプ用にこれを行うように用意されています。例えば、

```
% swapbytes < data.bin in=float > data.swap
```

とすると、Cの形式の浮動小数点実数を含むファイルのバイトをスワップします。

3.4.2 SUSWAPBYTES - SUデータのバイト・スワップ

SUデータ (SUヘッダのあるデータ) をシステム間で移動する場合、データおよびSUヘッダの両方のバイトをスワップしなければなりません。**suswapbytes** プログラムはこのために提供されています。コマンド列：

```
% suswapbytes < data.su format=0 > data.su.swapped
```

はSUフォーマットのデータを逆エンディアンのマシンから自分のシステムのバイト順へとスワップします。

コマンド列:

```
% suswapbytes < data.su format=1 > data.su.swapped
```

はSUフォーマットのデータを自分のシステムのバイト順から逆バイト順へとスワップします。これをSUデータを自分のシステムからバイト順が逆の別のシステムに移動する際に利用できます。

3.5 トレース・ヘッダ・フィールドの設定、編集、表示

地震探査データには、それに関する非常に多くのパラメータがあります。SUデータでは (SEG-Yフォーマットの例に倣って) それらのパラメータの値はトレースのヘッダ・フィールドに保存されます。さまざまな目的のために、ヘッダ・フィールドにアクセスしこれらのフィールドを設定、表示、修正することができるプログラムがたくさんあります。

ここで扱う作業は、以下の通りです。

- SUヘッダの付加
- SUヘッダの削除と貼りつけ
- SUキーワードの判別
- SUヘッダ値の範囲の表示
- 指定したSUヘッダ・フィールドの設定

- 与えた 2 つのフィールドからの第 3 のヘッダ・フィールドの計算
- ヘッダ・フィールドの値の取得
- 指定したヘッダ・フィールドの編集

これらはすべて必要なものであり、“ジオメトリの設定” という表題の部類に入るかもしれません。

本章では以下のリストのプログラムについて説明します。

- SUADDHEAD - ヘッダのトレースへの付加と tracl および ns フィールドの設定
- SUSTRIP - トレースから SEG-Y ヘッダの削除
- SUPASTE - 既存のヘッダを既存のトレースに貼りつけ
- SUKEYWORD - segy.h 中の SU キーワードのガイド
- SURANGE - 非ゼロのヘッダ・エントリの最大値および最小値の取得
- SUSHW - トレース番号、剰余と整数除算を用い、ヘッダ・ワード値を計算、または、ファイルからのヘッダ・ワード値の入力することによるヘッダ・ワードの設定。
- SUCHW - 1 あるいは 2 つのヘッダ・ワード・フィールドを用いたヘッダ・ワードの変更
- SUGETHW - SU データのヘッダ・ワードの取得
- SUEDIT - SEG-Y ディスクファイルの検査とヘッダの編集
- SUXEDIT - SEG-Y ディスクファイルの検査とヘッダの編集

いくつかの事項については既に前節で説明しましたが、完全を期すために、それらのプログラムの働きをもう少し詳しい情報を加えて、もう一度見ることにしましょう。

3.5.1 SUADDHEAD - SU (SEG-Y 形式) トレース・ヘッダの付加

C 形式のバイナリ浮動小数点実数からなるデータにヘッダを付加するには、以下のようにして下さい:

```
% suaddhead < data.bin ns=1024 > data.su
```

他のデータ型、例えば整数、の場合は **recast** を用いて下さい。

```
% recast < data.ints in=int out=float | suaddhead ns=1024 > data.su
```

ここで、処理の流れを接続するためにパイプ | を用いました。

データが元々Fortran で作られた整数である場合には処理の流れはこのようになるでしょう:

```
% ftnstrip < data.fortran | recast in=int out=float | suaddhead ns=1024 > data.su
```

もちろん他のバリエーションも可能です。

3.5.2 SUSTRIP と SUPASTE - SU ヘッダの削除と貼りつけ

```
% sustrip < data.su head=data.head > data.strip
... トレース数を変更しない他の処理 ...
% supaste < datanew.strip head=data.head > datanew.su
```

先の **sustrip** と **supaste** の説明を見て下さい。

3.5.3 SUKEYWORD - SUキーワードの表示

この節で説明する次の5つのプログラムには、selfdoc中に “key=” オプションがあり、トレース・ヘッダ・フィールドに “keywords” への参照があります。第2章で説明したように、これらのヘッダ・キーワードが何であるかを決めるには **sukeyword** が用いられます。

```
% sukeyword -o
```

とタイプすると、このリストが表示されます。(付録Bのリストでも見ることができます。) SUヘッダには80を越えるフィールドが定義されていますが、ほとんどの場合は比較的小さな下位セットしか用いられていません。これらのフィールドは **sukeyword** による一覧によって以下のように表示されます。

```
int tracl; /* trace sequence number within line */
int tracr; /* trace sequence number within reel */
int cdp; /* CDP ensemble number */
int cdpt; /* trace number within CDP ensemble */
 ...
short trid; /* trace identification code:
 ... 1 = seismic data

int offset; /* distance from source point to receiver
 ...
int  sx; /* X source coordinate */
int  sy; /* Y source coordinate */
int  gx; /* X group coordinate */
int  gy; /* Y group coordinate */
short counit; /* coordinate units code:
 ...
short delrt; /* delay recording time, time in ms between
   initiation time of energy source and time
   when recording of data samples begins
   (for deep water work if recording does not
   start at zero time) */
unsigned short ns; /* number of samples in this trace */
unsigned short dt; /* sample interval; in micro-seconds */

 ...
/* local assignments */
float d1; /* sample spacing for non-seismic data */

float f1; /* first sample location for non-seismic data */

float d2; /* sample spacing between traces */

float f2; /* first trace location */
```

モデリング・プログラムで作成したデータ、あるいはフィールド・データを用いる場合に、これらのキーワードの集合について知っておくのはよいことです。

3.5.4 SURANGE - ヘッダ値の範囲の取得

与えられたデータセットのヘッダの値の範囲を知ることによって、トレース・ヘッダに関する役立つ情報が得られます。

```
% surange < data.su
```

とタイプすることにより、ゼロでないすべての SU ヘッダ・フィールドの範囲を返します。例えば:

```
% suplane | surange
32 traces:
 tracl=(1,32)  tracr=(1,32)  offset=400 ns=64 dt=4000
```

suplane プログラムは共通オフセット記録のデータセットを模擬したテスト・パターンを生成します。**suplane** のデフォルト・パラメータは 32 トレースで 4 ms (SEG の表記では 4000 μs) のサンプリング間隔でトレース当たり 64 サンプル、オフセット=400 です。出力は 3 本の交差するスパイクの並びからなっています。

壊れたデータは非常に多くのヘッダ・フィールドに対しておかしな値を示すことがあることに注意して下さい。このような問題を検出することも **surange** の使用法の一つです。

3.5.5 SUGETHW - SU データ中のヘッダ・ワードの取得

データにヘッダ・フィールドがあるということは、地震トレースの値だけでなく、多くの付加的な情報を保存しておく必要があるということです。SEG-Y フォーマットのテープからデータを読む際に、**segyread** は SEG-Y ヘッダの主要部分内のトレース・ヘッダ情報を保存します。先に、**surange** によってデータセット全体のヘッダ値の最大、最小を知ることができることがわかりました。トレースごとに、また、選択した順序で、トレース・ヘッダ・フィールドの値を知りたいことがしばしばあります。

sugethw プログラム (SU get header word と読みます) はまさにそのようなユーティリティです。例えば、コマンド列:

```
% sugethw < data.su key=keyword1,keyword2,... | more
```

は列挙されたキーワードによって指定したそれぞれのヘッダフィールドの値を出力します。もっと実感的な例として、suplane データをパイプ | を通して入力すると、以下のようになります。

```
% suplane | sugethw key=tracl,tracr,offset,dt,ns | more
 tracl=1         tracr=1         offset=400          dt=4000         ns=64

 tracl=2         tracr=2         offset=400          dt=4000         ns=64

 tracl=3         tracr=3         offset=400          dt=4000         ns=64

 tracl=4         tracr=4         offset=400          dt=4000         ns=64

 tracl=5         tracr=5         offset=400          dt=4000         ns=64

 tracl=6         tracr=6         offset=400          dt=4000         ns=64

 tracl=7         tracr=7         offset=400          dt=4000         ns=64
...
```

最低 1 つのキーワードが指定されていれば、指定したキーワードの順序やキーワードの数に関する制約はありません。

何らかの理由で、バイナリ・フォーマットで値を出力する必要があるときには、例えば、再度 suplane データを用いて

```
% suplane | sugethw key=tracl,tracr,offset,dt,ns output=binary > file.bin
```

とすると、トレースごとに、与えられたキーワードの順でシーケンシャルに値を出力します。

“ジオメトリの設定” には以下のコマンド列を用いることのがよいかもしれません (ここでも、suplaneデータをパイプを通して **sugethw** に通して例示します)。

```
% suplane | sugethw key=tracl,tracr,offset,dt,ns output=geom > hdrfile
```

“hdrfile” の内容は以下のように表示することもできます。

```
% more hdrfile

1 1 400 4000 64
2 2 400 4000 64
3 3 400 4000 64
4 4 400 4000 64
5 5 400 4000 64
...
```

ここでは、データの最初の5行しか示していません。

3.5.6 SUSHW - SUデータ中のヘッダ・ワードの設定

sushw プログラム (SU set header word と読みます) は地震トレースのヘッダの値を設定する全目的型ユーティリティです。ユーザーはこのプログラムを用いて一つあるいはそれ以上のトレース・ヘッダを設定することができます。**sushw** の使い方は普通は単にあるフィールドに値を与えることです。例えば “dt” フィールドが設定されていないデータが時々あります。データが 2 ms でサンプルされていたとしましょう。**sukeyword** を用いると、dt は μs 単位であることがわかります。

```
% sukeyword dt


...skipping
        unsigned short ns;      /* number of samples in this trace */

        unsigned short dt;      /* sample interval; in micro-seconds */
...
```

以下のコマンド列により、すべての dt 値を 2000 μs に設定します。

```
% sushw < data.su key=dt a=2000 > data.out.su
```

もっと実際的な例として、suplaneデータを **sushw** にパイプすると以下のようになります。

```
% suplane | sushw key=dt a=2000 | sugethw key=dt | more

    dt=2000

    dt=2000

    dt=2000

    dt=2000
```

```
    dt=2000
...
```

sushw の selfdoc を見ると、以下のオプションのパラメータが定義されていることがわかります。

```
 Optional parameters ():
 key=cdp,... header key word(s) to set
 a=0,... value(s) on first trace
 b=0,... increment(s) within group
 c=0,... group increment(s)
 d=0,... trace number shift(s)
 j=ULONG_MAX,ULONG_MAX,... number of elements in group
```

これらのオプションによりもっと複雑な操作を実行することが可能になります。ヘッダ・フィールドとデータセット中のトレースの位置にはしばしば関係があるため、これらが必要となります。ヘッダ・フィールドの値は以下の公式で計算されます。

```
  i = itr + d
  val(key) = a + b * (i % j) + c * (i / j)
 where itr is the trace number (first trace has itr=0, NOT 1)
```

ここで、パーセント%は “モジュロ (剰余)” 関数を、/は除算を表します。

例えば、最初の 5 トレースの sx フィールドを 6400 に、次の 5 トレースを 6300 に、各 5 トレース・グループごとに-100 だけ減らして設定したい場合は、以下のようにします。

```
% sushw < data.su key=sx a=6400 c=-100 j=5 > data.new.su
```

suplane データを **sushw** にパイプすると、以下のようになります。

```
% suplane | sushw  key=sx a=6400 c=-100 j=5 | sugethw key=sx | more

    sx=6400

    sx=6400

    sx=6400

    sx=6400

    sx=6400

    sx=6400

    sx=6300

    sx=6300

    sx=6300

    sx=6300

    sx=6300

    sx=6200

    sx=6200
...
```

別の例として、“offset” フィールドを5トレースからなるグループで200, 400,..., 6400に設定したい場合は、以下のようにします。

```
%  sushw  < data.su key=offset a=200 b=200 j=5 > data.out.su
```

前と同様に、suplaneデータを **sushw** にパイプすると、以下のようになります。

```
% suplane | sushw  key=offset a=200 b=200 j=5 | sugethw key=offset | more

offset=200

offset=400

offset=600

offset=800

offset=1000

offset=200

offset=400

offset=600

offset=800

offset=1000

offset=200

...
```

以下のように、これら3つの操作すべてを **sushw** を一度コールするだけで実行できます:

```
% sushw < data.su key=dt,sx,offset a=2000,6400,200 b=0,0,200 c=0,-100,0 j=0,5,5
 > newdata.su
```

あるいは、suplaneデータのパイプを用いて、

```
% suplane | sushw key=dt,sx,offset a=2000,6400,200 b=0,0,200 c=0,-100,0 j=0,5,5 |
  sugethw key=dt,sx,offset | more

    dt=2000         sx=6400     offset=200

    dt=2000         sx=6400     offset=400

    dt=2000         sx=6400     offset=600

    dt=2000         sx=6400     offset=800

    dt=2000         sx=6400     offset=1000

    dt=2000         sx=6300     offset=200

    dt=2000         sx=6300     offset=400

    dt=2000         sx=6300     offset=600
```

```
    dt=2000          sx=6300      offset=800

    dt=2000          sx=6300      offset=1000

    dt=2000          sx=6200      offset=200

    dt=2000          sx=6200      offset=400
...
```

ジョブの流れを制御するにはパイプとリダイレクトを用いるのが自然です。しかし、見た通り、1 行のコマンド・ライン上では見苦しくなります。この文書の後ろの方では、複雑な処理の流れをシェル・スクリプトの制御された環境を用いて構築する方法について見ていくことにします。

3.5.7 ジオメトリの設定 - 観測者の記録のトレース・ヘッダへの変換

フィールド・データに対して行わなければならない “ジオメトリの設定” と呼ばれる作業はあまり楽しくない作業です。SEG-Y テープには、ヘッダ・フィールドのセットのうち基本的なセットしかデータに入っていないことがよくあります。残りの情報 (震源位置、受振点位置、その他) は観測者の記録という形で提供されます。

ジオメトリを設定するには、指定した一連のヘッダ・フィールドをファイルにダンプして、変更できるようにそのファイルをテキスト・エディタや表計算プログラムに読み込みたいでしょう。

例えば、いくつかのヘッダ・フィールドが不正確に、あるいは不完全に設定された “sudata” ファイルがあるとします。このようなデータを扱うことのできる方法は以下のコマンド列で示されます。まず、選択したヘッダ・フィールドを “hdrfile” ファイルに読み込むことから始めます。

```
% sugethw < sudata output=geom key=key1,key2,... > hdrfile
```

そして、ASCII ファイル “hdrfile” をエディタで編集し、フィールドを適切に設定します。以下の方法で hdrfile をバイナリ・フォーマットに変換します:

```
% a2b < hdrfile n1=nfields > binary_file
```

ここで “nfields” は先の “key=..” リスト中のヘッダ・フィールドの数です。そして、以下のように、新しいヘッダ・フィールドのファイルをロードします:

```
% sushw < sudata infile=binary_file key=key1,key2,... > sudata.edited
```

ここで、“key=key1,key2,...” は先の **sugethw** 文のリストと同じです。最終的にファイル “sudata.edited” が作成されます。

ヘッダ・フィールドの設定を始めたばかりであれば、望みの方法で ASCII ヘッダ・ファイル “hdrfile” を作ることができます。好きなテキスト・エディタ、または表計算プログラムを使うことができます。先に示した並びの複数列の ASCII フォーマットでありさえすれば、ASCII ファイルを作る方法は何でも構いません。

もちろん、トレースごとに列挙された C 形式の浮動小数点実数からなるヘッダ値のファイルがあれば (C プログラムで作成したものか、**ftnstrip** を通した Fortran データのいずれか) “binary_file” がすでにあることになるので、最後のコマンドを実行するだけです。

3.5.8 SUCHW - SUデータ中のヘッダ・ワードの変更 (または計算)

"cdp" のようなヘッダ・フィールドは既存のヘッダ・フィールドを用いて計算することもできます。**suchw** プログラムはこの機能を提供します。

suchw のselfdocによると、

```
...
 key1=cdp,... output key(s)
 key2=cdp,... input key(s)
 key3=cdp,... input key(s)

...
 a=0,... overall shift(s)
 b=1,... scale(s) on first input key(s)
 c=0,... scale on second input key(s)
 d=1,... overall scale(s)
```

このプログラムでは、2つのヘッダ・フィールド "key2" と "key3" を用いて次式に基づいて "key1" を計算します。

```
...
val(key1) = (a + b * val(key2) + c * val(key3)) / d
..
```

例えば、cdpヘッダ・フィールドの値を一定値 (-1としましょう) だけシフトするには、

```
% suchw <data >outdata a=-1
```

あるいは、一定量 (1000としましょう) をヘッダ・フィールド (仮に "tracr" としましょう) に加えるには、

```
% suchw key1=tracr key2=tracr a=1000 <infile >outfile
```

別の例として、**suchw** を用いて "offset" と "sx" (ショット番号) 値を加え、"sx" と "gx" とを平均して "cdp" フィールドを計算して、"gx" フィールドを設定します。ここで、通常の 1, 2, 3,... という数字ではなく、実際のCDPの位置をCDP番号として使っています。

```
% suchw <indata key1=gx key2=offset key3=sx b=1 c=1 |
% suchw key1=cdp key2=gx key3=sx b=1 c=1 d=2 >outdata
```

以下のようにして、両方の操作を一回で実行することができます:

```
% suchw<indata key1=gx,cdp key2=offset,gx key3=sx,sx b=1,1 c=1,1 d=1,2 >outdata
```

3.5.9 SUEDITとSUXEDIT - SUデータ中のヘッダ・ワードの編集

最後に、2、3のヘッダだけを検査して変更したいことがあります。この目的のために、**suedit** と **suxedit** があります。SU編集用プログラムは以下のように実行します:

```
% suedit diskfile    (書き込み可であればヘッダ修正のためにオープンする)
% suedit < diskfile  (リード・オンリーでオープンする)
```

ヘッダ・フィールドの表示と編集が会話的にできます。

例えば、**suplane** を用いてテスト・データを作成すると、

```
% suplane > data.su
% suedit  data.su
```

以下のようになります。

```
32 traces in input file
 tracl=32 tracr=32 offset=400 ns=64 dt=4000
>                          <------- 会話的に使用するためのプロンプト
```

suedit と **suxedit** で会話的に用いることのできるコマンドは、プロンプトでクエスチョン・マーク (?) をタイプすると表示されます。例えば、

```
32 traces in input file
 tracl=32 tracr=32 offset=400 ns=64 dt=4000
> ?

 n              read in trace #n
 <CR>           step
 +              next trace;   step -> +1
 -              prev trace;   step -> -1
 dN             adv N traces; step -> N
 %              percentiles
 r              ranks
 p [n1 [n2]]    tabplot
 ! key=val      modify field
 ?              print this file
 q              quit
>
```

このプログラムにより、ユーザーがデータ・サンプル値のタブ・プロットとしてトレースを見たり、個々のヘッダ値を見たり変更したりすることができます。

suxedit プログラムは **suedit** とよく似ていますが、トレースを表示する X-windows 上のグラフィックスが追加されています。

```
% suxedit diskfile （書き込み可であればヘッダ修正のためにオープンする)
% suxedit <diskfile （リード・オンリーでオープンする)
```

```
% suxedit data.su
32 traces in input file
 tracl=32 tracr=32 offset=400 delrt=5 ns=64 dt=4000

> ?

 n              read in trace #n
 <CR>           step
 +              next trace;   step -> +1
 -              prev trace;   step -> -1
 dN             adv N traces; step -> N
 %              percentiles
 r              ranks
 p [n1 [n2]]    tabplot
 g [tr1 tr2] [&quot;opts&quot;]   wiggle plot
 f              wig plot Fourier Transf
 ! key=val      modify field
 ?              print this file
 q              quit
```

このプログラムのオプションは大部分は自明です。クエスチョン・マーク “?” をタイプして得られるヘルプ・メニューよりも、selfdocの方が情報が多いことを思い出して下さい。

第4章　X-WindowsおよびPostScriptによるSUデータの表示

Seismic Unix パッケージには、一般的な C の浮動小数点フォーマットおよび SU フォーマットのデータを表示するための簡単なグラフィックス・ユーティリティがあります。画面上に表示するための X-Windows 環境およびハード・コピーのための PostScript 形式に対応しています。

SU で利用できる描画形式には、

- コンター表示
- 濃淡およびカラー・イメージ表示
- ウィグル・トレース表示
- 線グラフ
- 動画
- 3-D 立体表示 (PostScript のみ)

があります。

これらのプログラムには、表示方法の選択や図のラベル付けのための多くのオプションがあるので、長い selfdoc がついています。しかし、データにウィンドウを施すような操作は、前処理のステップとして **subset** や **suwind** プログラムによって、データが実際に描画プログラムに渡される前になされなければなりません。生の地震記録データセットはしばしば巨大なので、これらの表示プログラムはデータにウィンドウを施すように設計されていません。

UNIX の “小さいことは美しいことだ (small is beautiful)” という哲学に沿って、ハードコピーを描画するために PostScript 出力を生成するほとんど同じコードが別にあります。

4.1　X-Windows 上の表示プログラム

X-windows は、画面上のグラフィックス・ルーチンの作成のための統合された環境を提供します。一般的な X のディストリビューションに合うことが保証された部品ばかりを用いて書かれたコードであれば、プログラムは非常に移植性がよいからです。

したがって、SU の X-windows 用のコードは、直接 X コール (X-call) か X のツールキット (X-Toolkit) を用いて書かれています。Motif のようなウィジェット・セット (widget set) を使って書くことは多くの点でより容易ですが、コードの移植性が失われます。さまざまなプラットフォームにおける商用のウィジェット・セットの実装にはしばしば大きな違いがあります。

4.1.1 一般の浮動小数点データの表示

X-window 環境における一般の浮動小数点データ (SU ヘッダのないデータ) を表示するために用いられるプログラムには以下のものがあります。

- XCONTOUR - ベクトル・プロット・コールによる f(x1,x2) の X 上のコンター表示
- XIMAGE - 均一にサンプルされた関数 f(x1,x2) の X 上のイメージ表示
- XWIGB - ビットマップによる f(x1,x2) の X 上のウィグル・トレース表示
- XGRAPH - n[i] 組の (x,y) 座標のグラフの描画
- XMOVIE - 均一にサンプルされた関数 f(x1,x2) の一つ以上のフレームのイメージ表示

以下を試してみて下さい。SU データのヘッダを削除してバイナリ・データを作ります。例えば、

```
% suplane | sustrip > data.bin
n1=64 n2=32 d1=0.004000
nt=64 ntr=32 dt=0.004000
ns=64
```

後に続く項目は、データ・セットの次元が “n1=64”×“n2=32” であることを示します。では、上に掲げたそれぞれの表示プログラム (**xgraph** は除く) でこれらのデータを見てみましょう:

```
% xcontour < data.bin n1=64 n2=32 title="contour"  &
% ximage < data.bin n1=64 n2=32  title="image"  &
% xwigb < data.bin n1=64 n2=32  title="wiggle trace"  &
% xmovie < data.bin n1=64 n2=32  title="movie"  &
```

アンパサンド “&” はシェルにプログラムをバックグラウンドで走らせるように指示する UNIX コマンドです。

xgraph を試すために、表示するデータの組の 2 列のリストからなる ASCII ファイルを作ります。例えば、以下のようなものです。

```
1 1
2 1.5
3 3
4 8
10 7
```

このファイルを “data.ascii” と呼びます。次に、**a2b** を用いてファイルをバイナリに変換し、**xgraph** でプロットします。

```
% a2b < data.ascii n1=2 > data.bin
n=5
% xgraph < data.bin n=5
```

a2b が返した “n=5” は **xgraph** の入力と同じでなければならないことに注意して下さい。

終了してウィンドウを消去したい時は、ウィンドウをクリックして、文字 “q” をタイプして終了 (quit) して下さい。システムによっては、ウィンドウ・フレームの上部の左端の正方形をクリックしドラッグして、“destroy” オプションを選択するものもあります。

これらのプログラムにはかなり多くの機能があることを反映して、プログラムには非常に多くのオプションがあることに注意して下さい。

```
% programname
```

または、

```
% sudoc programname
```

とタイプして、それぞれのプログラムの selfdoc を見て下さい。

4.1.2 SU データの X-Windows 上での表示

SU フォーマットで書かれたデータを表示するために、多くのプログラムが作られました (その多くはこれまでに説明してきたプログラムと同等の機能を持っています)。それらは以下の通りです:

- SUXCONTOUR - ベクトル・プロット・コールによる Seismic UNIX トレース・ファイルの X-Windows 上のコンター表示
- SUXIMAGE - SU データ・セットの X-Windows 上のイメージ表示
- SUXWIGB - SU データ・セットの X-Windows 上のビットマップ・ウィグル・トレース表示
- SUXGRAPH - SU データの X-Windows 上のグラフ表示
- SUXMOVIE - SU データ・セットの X MOVIE 表示
- SUXMAX - SU データ・セットのそれぞれのトレースの最大値、最小値、絶対値の最大値の X-windows 上のグラフ

複数のコードを保持しているのではなく、これらのプログラムは実際はそれぞれ前の小節でリストにあげた X-windows グラフィック・プログラムを一つあるいはそれ以上呼び出しています。非 SU バージョンのグラフィックス・プログラムの selfdoc は SU バージョンにも適用できることに気をつけて下さい。つまり、これらのプログラムも多くの機能を持っています。

suplane データを用いて以下のようにこれらのプログラムを試すことができます:

```
% suplane | suxcontour title="contour"  &
% suplane | suximage  title="image"  &
% suplane | suxgraph  title="graph" &
% suplane | suxmovie  title="movie" &
% suplane | suxmax    title="max"  &
```

もう一度繰り返しますが、アンパサンド “&” はシェルにプログラムをバックグラウンドで走らせるように指示する UNIX コマンドです。終了してウィンドウを消去したい時は、ウィンドウをクリックして、文字 “q” をタイプして終了 (quit) して下さい。システムによっては、ウィンドウ・フレームの上部の左端の正方形をクリックしドラッグして、“destroy” オプションを選択するものもあります。

4.1.3 X-Windows プログラムの特別の機能

これらのプログラムの多くのオプションのすべてを探すためには、selfdoc を見て下さい。コマンドラインでそれぞれのプログラムの名前をタイプして下さい。

```
% suxcontour
% suxwigb
% suximage
% suxmovie
% suxmax
```

さらに、これらのプログラムの内のいくつかは、非 SU バージョンの名前をタイプすることでさらに情報を得ることができます。

```
% xcontour
% xwigb
% ximage
% xmovie
```

しかし、これらのプログラムのもつ特性のうちには例を用いてしか説明できないものもあります。

SUXWIGB を用いた真のオフセットでのウィグル・トレース表示

ウィグル・トレースを真のオフセットでプロットすることができます。つまり、ヘッダ内の値からウィグル・プロットの水平方向の次元に用いる値を取得することが可能です。これは “key=” パラメータを用いて行えます。

例えば、**suplane** を用いてテスト・データを作成し、以下のようにして “key=offset” を使ってプロットしてみましょう。

```
% suplane | suchw key1=offset key2=tracl a=0 b=100 | suxwigb key=offset &
```

結果は、x2 軸がオフセット・ヘッダ・フィールドの値 (100 づつカウントされている) でラベル付けられたものとなります。

SUXMOVIE を用いた動画の作成

suxmovie を用いて地震データの動画を作ることができます。この例では **suplane** を用いていくつかの合成データ・パネルを作り、各々連続したパネルを二重リダイレクト記号 “>>” を用いて追加していきます。

```
% suplane > junk1.su
% suplane | suaddnoise sn=20 >> junk1.su
% suplane | suaddnoise sn=15 >> junk1.su
% suplane | suaddnoise sn=10 >> junk1.su
% suplane | suaddnoise sn=5 >> junk1.su
% suplane | suaddnoise sn=3 >> junk1.su
% suplane | suaddnoise sn=2 >> junk1.su
% suplane | suaddnoise sn=1 >> junk1.su

% suxmovie < junk1.su n2=32 title="frame=%g" loop=1 &
```

最後のコマンドではデータの 1 フレームごとに 32 トレースあるということを示す “n2=32” を設定しています。“%g” を使うことによってフレーム番号をタイトルの一部として表示することができ、“loop=1” により動画は連続ループします。

動画の速度を速くしたり遅くしたりするには、動画の右下角をクリックしドラッグして単にウィンドウを拡大あるいは縮小して下さい。一番右のマウス・ボタンを一度クリックするとフレームが停止し、もう一度クリックすると動画を再開します。

4.1.4 PostScript 出力プログラム

X-Windows 表示ユーティリティ集を補完するものとして、同様な PostScript コードがあります。先に示した X-Windows 用のコードのそれぞれに対応する PostScript 出力コードを作るという考えです。

4.1.5 一般の浮動小数点データの PostScript 出力

一般の浮動小数点データ (SU ヘッダのないデータ) の PostScript 出力を行うために用いられるプログラムには以下のようなものがあります。

- PSCONTOUR - 2 次元関数 f(x1,x2) の PostScript コンタリング
- PSIMAGE - 均一にサンプルされた関数 f(x1,x2) の PostScript イメージ
- PSCUBE - データ・キューブの PostScript イメージ表示
- PSGRAPH - (x,y) 座標の n[i] 組の PostScript グラフ表示
- PSMOVIE - 均一にサンプルされた関数 f(x1,x2,x3) の PostScript ムービー (動画) 表示
- PSWIGB - ビットマップによる f(x1,x2) の PostScript ウィグル・トレース表示
- PSWIGP - ポリゴンによる f(x1,x2) の PostScript ウィグル・トレース表示

繰り返しになりますが、**suplane** のデータのヘッダを削除することによってバイナリ・データを作成し、これらのプログラムを試すことができます。

```
% suplane | sustrip > data.bin
n1=64 n2=32 d1=0.004000
nt=64 ntr=32 dt=0.004000
ns=64
```

データのディメンジョンは 1 トレースに n1=64 サンプルで n2=32 トレースです。

```
% pscontour < data.bin n1=64 n2=32 title="contour" > data1.eps
% psimage < data.bin n1=64 n2=32 title="image" > data2.eps
% pscube < data.bin n1=64 n2=32 title="cubeplot" > data4.eps
% pswigb < data.bin n1=64 n2=32 title="bitmapwiggletrace" > data3.eps
% pswigp < data.bin n1=64 n2=32 title="wiggletrace" > data4.eps
% psmovie < data.bin n1=64 n2=32 title="movie" > data5.eps
```

出力ファイルは Adobe Level 2 Encapsulated PostScript を含みます。これらのファイルは、標準的な X-windows 上の PostScript プレビュア (例えば “ghostview”) を用いて見ることができます。

気をつけていただきたいのですが、“psmovie” の出力はあなたのシステムでは動作しないかもしれません。出力は NeXTStep では動作します。しかし、マルチ・ページ Encapsulated PostScript は一般には PostScript デバイスではサポートされていません。

psgraph を試すには、表示する 2 列のリストのデータの組からなる ASCII ファイルを作ります。例えば、以下のようなものです。

```
1 1
2 1.5
3 3
4 8
10 7
```

このファイルを “data.ascii” と呼びます。次に、**a2b** を用いてファイルをバイナリに変換し、**psgraph** でプロットします。

```
% a2b < data.ascii n1=2 > data.bin
n=5
% psgraph < data.bin n=5 > data6.eps
```

a2b が返した “n=5” は **xgraph** の入力と同じであることに注意して下さい。これによって、同じ出力を得るのに以下のようなトリックを使うことができます。

```
% a2b < data.ascii outpar=junk.par n1=2 > data.bin
% psgraph < data.bin par=junk.par > data6.eps
```

4.1.6 SU データの PostScript 表示

SU データを表示するための X-Windows コードがあるのと同様に、PostScript 表示を行うコードもあります。PostScript グラフィックス用のプログラムは以下の通りです。

- SUPSCONTOUR - SU データ・セットの PostScript コンター表示
- SUPSIMAGE - SU データ・セットの PostScript イメージ表示
- SUPSCUBE - SU データ・セットの PostScript キューブ (立体) 表示
- SUPSGRAPH - SU データ・セットの PostScript グラフ表示
- SUPSWIGB - SU データ・セットのビットマップによる PostScript ウィグル表示
- SUPSWIGP - SU データ・セットのポリゴンによる PostScript ウィグル表示
- SUPSMAX - SU データ・セットの各トレースの最大値、最小値、絶対値の最大値の PostScript 表示

X-Windows コードの時と同様に、suplane データを用いてこれらのプログラムをそれぞれ試してみることができます。

```
%suplane > junk.su
```

```
% supscontour < junk.su title="contour" > data1.eps
% supsimage < junk.su title="image" label1="sec" label2="tracenumber" > data2.eps
% supscube < junk.su title="cubeplot" > data4.eps
% supswigb < junk.su title="bitmapwiggletrace" > data3.eps
% supswigp < junk.su title="wiggletrace" > data4.eps
% supsmovie < junk.su title="movie" > data5.eps
% supsmax < junk.su title="max" > data5.eps
```

出力ファイルは Adobe Level 2 Encapsulated PostScript を含んでおり、TEX、LATEX、その他のドロー・ツールと互換性があります。これらのファイルを画面上で見るためには、GhostScript や Ghostview のような PostScript プレビュアが必要です。

繰り返しになりますが、非 SU バージョンのあるプログラムは、それらのコードの selfdoc 情報も同様に当てはまることに注意して下さい。

4.2 追加の PostScript サポート

SU には PostScript 操作をサポートする追加のツールがいくつかあります。

- PSBBOX - PostScript ファイルの BoundingBOX の変更
- PSMERGE - PostScript ファイルのマージ
- MERGE2 - 2 つの PostScript 形式の図の 1 ページへのマージ
- MERGE4 - 4 つの PostScript 形式の図の 1 ページへのマージ
- PSLABEL - 背景を指定した 1 行のテキスト文字列からなる PostScript ファイルの出力。(図にラベルを付けるには psmerge を使用して下さい)
- PSMANAGER - HP 4MV および HP 5Si Mx Laserjet 用の PostScript 印刷のマネージャー
- PSEPSI - EPSI フォーマットのプレビュー・ビットマップの EPS ファイルへの付加

psbbox、**pslabel**、**psmerge**、**merge2**、**merge4** の各プログラムは SU 形式のグラフィックス・プログラムで作成された図の構成を支援するために設計されており、他の方法で生成された EPS ファイルでの動作は保証されません。

4.2.1 PSBBOX - BoundingBox の変更

例として、**suplane** と **supswigb** とを用いて以下のようにテスト PostScript データを作成します。

```
% suplane | supswigb > junk1.eps
```

ここで、拡張子 ".eps" はこのファイルがカプセル化された (encapsulated) PostScript であることを示すために選んでいます。この図の周りに空白が多すぎるとします。これを直すために、ファイルの先頭にある BoundingBox のサイズを変更しようと思います。例えば、

```
% morejunk1.eps
```

とすると BoundingBox の大きさが表示されます。

```
%!PS-Adobe-2.0EPSF-1.2
%%DocumentFonts:
%%BoundingBox: 13 31 603 746
...
```

これを手で編集することもできますが、**psbbox** を用いて、以下のようにタイプすると、

```
% psbbox < junk1.eps llx=40 lly=80 urx=590 ury=730 > junk2.eps
Original: %%BoundingBox: 13 31 603 746
Updated:  %%BoundingBox: 40 80 590 730
```

より小さな BoundingBox となり、したがって空白が少なくなります。

4.2.2 PSMERGE, MERGE2, MERGE4 - PostScript 形式の図のマージ

いくつかの図を複合した図を作成するために、図をマージするができれば便利です。**psmerge** プログラムはこのための SU の一般的ツールです。**merge2** と **merge4** という **psmerge** をコールする 2 つのシェル・スクリプトがあります。いくつかのテスト・データセットを作り、

```
% suplane > junk.su
% suplane | sufilter > junk1.su
```

これらをいろいろな方法で表示します。

```
% supswigb < junk.su title="Wiggletrace" label1="sec" label2="tracenumber" > junk1.eps
% supsimage < junk.su title="ImagePlot" label1="sec" label2="tracenumber" > junk2.eps
% supscontour < junk.su title="ContourPlot" label1="sec" label2="tracenumber" > junk3.eps
% supswigb < junk1.su title="Filtered" label1="sec" label2="tracenumber" > junk4.eps
```

これらの 4 つの PostScript ファイルを新しい図にマージします。

2 つの図のマージは以下のようにしてできます:

```
% merge2 junk1.eps junk2.eps > junk.m2.eps
```

すべての 4 つの図は以下のようにしてマージできます。

```
%merge2 junk1.eps junk2.eps junk3.eps junk4.eps > junk.m4.eps
```

もちろん、**merge2** と **merge4** のどちらも、あらゆる図のサイズを取り扱えるほど丈夫ではありません。ですから、**psmerge** を用いて手でマージする必要があることもあります。また、ウィグル・トレースの上にグラフを重ねるなど図を重ねたい場合にも **psmerge** を使う必要があるでしょう。

図を作り、**psmerge** でマージするサンプルを示します。コマンド行を列挙するのは見苦しいので、シェル・スクリプトとして表現します。

```
#! /bin/sh
# shell script for demonstrating PSMERGE

# make data
suplane > junk.su
suplane | sufilter > junk1.su

# make PostScript Plots of data
supswigb < junk.su wbox=6 hbox=2.5 \
 title="Wiggle trace" label1="sec" label2="traces" > junk1.eps
supscontour < junk.su wbox=2.5 hbox=2.5  \
 title="Contour Plot" label1="sec" label2="traces" > junk3.eps
supswigp < junk1.su wbox=2.5 hbox=2.5 \
 title="Filtered" label2="traces" > junk4.eps

# merge PostScript plots
psmerge in=junk1.eps translate=0.,0. \
 in=junk3.eps translate=0.0,3.7 \
 in=junk4.eps translate=3.3,3.7 > junk5.eps

echo "You may view the files: junk1.eps, junk3.eps, junk4.eps, junk5.eps"
echo "with your PostScript Previewer"

exit 0
```

この場合、元のファイルは 8-1/2”×11” のウィンドウ内にフィットするように小さくしました。しかし、**psmerge** には図をスケールする機能があります (だから **merge2** と **merge4** シェルがうまく動くのです)。もっと詳しい情報は、以下のようにタイプしてテキストをよく読めばわかります。

```
% more $CWPROOT/bin/merge2
```

または、

```
% more $CWPROOT/bin/merge4
```

サイズの異なる 3 つの図をマージするサンプルは以下のシェル・スクリプトで与えられます。

```
#! /bin/sh
# shell script for demonstrating PSMERGE

# make data
suplane > junk.su
suplane | sufilter > junk1.su

# make PostScript Plots of data
supswigb < junk.su wbox=7 hbox=4 \
 title="Wiggle trace" label1="sec" label2="traces" > junk1.eps
supscontour < junk.su \
 title="Contour Plot" label1="sec" label2="traces" > junk3.eps
supswigp < junk1.su label1="sec" \
 title="Filtered" label2="traces" > junk4.eps

# merge PostScript plots
psmerge in=junk1.eps translate=0.,0. scale=.6,.6 \
 in=junk3.eps scale=.4,.4 translate=0.0,3.7 \
 in=junk4.eps scale=.4,.4 translate=3.3,3.7 > junk5.eps

echo "You may view the files: junk1.eps, junk3.eps, junk4.eps, junk5.eps"
echo "with your PostScript Previewer"

exit 0
```

この場合は、図は標準のサイズで、8-1/2”×11” のウィンドウにフィットするようにスケールされます。

4.3 トレース・ピッキング・ユーティリティ

この項目について説明するもっとよい場所がないので、ここで “トレース・ピッキング” ユーティリティの一覧を示します。X-Windows のウイグル・トレース、イメージ、コンター表示プログラムには、すべてこの属性があります。カーソルをピックする点に移動し、文字 ‘s’ をタイプするとその点の座標はメモリーに保存されます。文字 ‘q’ がタイプされるとその値はユーザー指定のファイル “mpicks” に保存されます。

ピッキング専用のプログラムがもう 2 つあります。

- XPICKER - Bitmap を用いたピッキング機能付き X ウイグル・トレース表示
- SUXPICKER - SU データ・セットの X-windows 用ウイグル・トレース表示
- SUPICKAMP - ユーザー定義あるいはリサンプルされたウインドウ内の振幅のピック

“xpicker” は “suxpicker” の非 SU バージョンです。xpicker/suxpicker プログラムはピッキングを行う会話型ツールです。“supickamp” プログラムは簡単な自動ピッキング・プログラムです。ユーザー指定のウインドウ内での最大振幅値を探します。

“supickamp” についての詳しい情報は、$CWPROOT/src/demos/Picking 内のデモを見て下さい。

第5章　SUデータの編集

一旦データが読み込まれヘッダが正しく設定されたら、次に行われるのはデータの操作と編集に関する事項です。

データを

- ウインドウ
- ソート
- 打ち切り
- テーパ
- ゼロ化
- サンプル数の統一
- 結合

するために、さまざまな作業を実行する必要がしばしばあります。これらはデータセット編集に関する事項です。

この節では、以下のトレース編集ユーティリティのプログラムを取り扱います。

- SUWIND - キーワードによるトレースのウィンドウ操作
- SUSORT - SEG-Y ヘッダ・キーワードのソート
- SURAMP - トレースの先頭あるいは終端のゼロへの線形テーパ
- SUTAPER - データ・パネルの端トレースのゼロへのテーパ
- SUNULL - ヌル・トレース (すべてゼロ) の作成
- SUZERO - 時間ウィンドウ内のデータのゼロ化
- SUKILL - トレースのゼロ化
- SUMUTE - ユーザー定義の多角曲線より上 (あるいは下) のミュート。曲線に沿う距離はキー・ヘッダ・ワードで指定される。
- SUVLENGTH - 可変長トレースの共通長さへの調整
- SUVCAT - あるデータ・セットの他のデータ・セットへの追加 (トレースごと)

5.1　SUWIND - キーワードによるトレースのウィンドウ操作

地震データ・セットの一部のみを表示したり、処理したりすることはよくあります。地震データにはウィンドウを施したいパラメータが多いので **suwind** が書かれました。

5.1.1 トレース・ヘッダ・フィールドによるウィンドウ操作

最も簡単な使用法として、**suwind** ではユーザーが特定のヘッダ・フィールドの最小値、最大値を設定することができます。

```
key=tracl       Key header word to window on (see segy.h)
min=LONG_MIN    min value of key header word to pass
max=LONG_MAX    max value of key header word to pass
```

例えば、トレース番号で suplane データにウィンドウを施すと以下の結果が得られます。

```
% suplane  | suwind key=tracl  min=5 max=10 | sugethw key=tracl | more

 tracl=5

 tracl=6

 tracl=7

 tracl=8

 tracl=9

 tracl=10
```

大きなデータセットでは、最大値を設定するのではなく “count” パラメータを使う方がよいでしょう。もし明示的に “max” 値を設定すると、**suwind** は最小値と最大値の間の値をもつすべてのトレースを取り出すために全データセットを調べます。なぜなら、プログラムは重複したトレースのラベルつけが可能であると仮定しているためです。例えば、

```
% suplane ntr=100000 | suwind key=tracl min=5 max=10 | sugethw tracl | more
```

と (数分したら “コントロール-c” をタイプしても OK です)

```
% suplane ntr=100000 | suwind key=tracl min=5 count=5 | sugethw tracl | more
```

とを比較して下さい。ここで suplane はそれぞれの場合について 100000 トレースを作成するように設定されています。

より複雑なウインドウ操作 (例えばデータの間引き) を行うばあいには、

```
j=1             Pass every j-th trace ...
s=0             ... based at s  (if ((key - s)%j) == 0)
```

とすることにより、suplane データを用いてトレースを 1 本おきに、

```
% suplane  | suwind key=tracl j=2 | sugethw key=tracl | more

 tracl=2

 tracl=4

 tracl=6

 tracl=8

 tracl=10
...
```

または、1 から始めて 1 本おきに、

```
% suplane  | suwind key=tracl j=2 s=1 | sugethw key=tracl | more

 tracl=1

 tracl=3

 tracl=5

 tracl=7

 tracl=9

...
```

表示することができます。トレースの取捨選択も **suwind** を用いて行うことができます。

```
...
       reject=none     Skip traces with specified key values
...    accept=none     Pass traces with specified key values(see notes)
```

“reject” パラメータは番号をつけたトレースをそのままリジェクトします。例えば、このようにすると、

```
% suplane | suwind key=tracl reject=3,8,9 | sugethw key=tracl | more
 tracl=1

 tracl=2

 tracl=4

 tracl=5

 tracl=6

 tracl=7

 tracl=10

 tracl=11

 tracl=12
```

トレース 3, 8, 9 はリジェクトされます

“accept” オプションは少し違います。アクセプト・リストにあるトレース**のみ**をアクセプトすることを意味しているのでは**ありません**！別の方法でリジェクトされた場合でも、それらのトレースはアクセプトされることを意味しているのです。例えば:

```
% suplane | suwind key=tracl reject=3,8,9 accept=8 | sugethw key=tracl
| more
 tracl=1

 tracl=2

 tracl=4
```

```
 tracl=5

 tracl=6

 tracl=7

 tracl=8

 tracl=10

 tracl=11
....
```

リストにあるトレース**のみをアクセプト**したいのであれば、“max=0” に設定する必要があります。

```
% suplane | suwind key=tracl accept=8 max=0 | sugethw key=tracl | more
 tracl=8
```

この例ではトレース 8 のみが通ります。

“count” パラメータは “accept” パラメータを無視します。ですから、本当に無条件でアクセプトしたいのであれば、“count” パラメータを指定してはいけません。

個別のサンプルについては$CWPROOT/src/demos/Selecting_Traces 内のデモを見て下さい。

5.1.2 時間ゲート操作

ウィンドウ操作の 2 番目の事項は時間ゲートです。実際、データにウィンドウを施したいとき、トレースと時間ゲートの両方を操作したいことが多いのです。

```
Options for vertical windowing (time gating):
      tmin = 0.0              min time to pass
      tmax = (from header)    max time to pass
      itmin = 0               min time sample to pass
      itmax = (from header)   max time sample to pass
      nt = itmax-itmin+1      number of time samples to pass
```

“itmin と itmax” または “tmin と tmax” のどちらかを設定すると、最も近いサンプルに時間ゲートを設定したことになります。任意の (サンプル間) ウィンドウに時間ゲートを施したいと思うのであれば、これはデータのリサンプルの問題になるので、**suresamp** が選択されるべきプログラムとなります。

5.2 SUSORT - SEG-Y ヘッダ・キーワードによるソート

UNIX オペレーティング・システム上で作業している場合の利点の一つとして、ある操作を行う優れた UNIX のシステム・コールがある場合には、必要な作業を実行するのにそのコールを使用できるという点があります。ソーティング (並び替え) はそのような作業であり、UNIX の “sort” コマンドはまさにそのようなユーティリティです。

susort は UNIX システムのソート・コマンドの利点を使用して、ヘッダ・フィールド・キーワードによるトレースのソートができるようにしています。

例えば、2 つのフィールド (CDP とオフセット) を用いたデータのソート (昇順の) は以下のようにして行うことができます:

```
% susort <indata.su >outdata.su cdp offset
```

オフセットに関しては降順、CDP に関しては昇順にする場合は、

```
% susort <indata.su >outdata.su cdp -offset
```

注意: 次のタイプの入力/出力しかサポートされていません。ディスク入力から任意の出力へ、パイプ入力からディスク出力へ。

トレース・ソーティングの個々のサンプルについては$CWPROOT/src/demos/Sorting_Traces 内のデモを見て下さい。

5.3 SURAMP, SUTAPER - データ値のテーパ

多くの地震探査データ処理アルゴリズムでは、データセットの端がシャープであると偽像を生じます。データセットの端の振幅を徐々に小さくすること (テーパ) は、これらの偽像を抑制する最も簡単な方法の一つです。これを目的として、データセットの端をテーパするために **sutaper** があります。ここで例えば、データセットの終端の 5 トレースにわたる線形テーパは以下のようになります。

```
% sutaper <diskfile >stdout ntaper=5
```

suramp はトレースの先端と終端とを滑らかにします。ここで例えば、時間 0 から tmin=.05 秒までをランプ・アップし tmax=1.15 秒からトレースの終端までの部分をランプ・ダウンする場合は以下のようになります。

```
% suramp <diskfile tmin=.05 tmax=1.15 >stdout
```

5.4 SUKILL, SUZERO, SUNULL, SUMUTE - データのゼロ化

ノイズの卓越したトレースをゼロ化したり、データセットの端のトレースをゼロ化したり (テーパの場合と同様)、表示の際にデータの連続パネル間のセパレータとしてヌル・トレースを生成したりすることができれば、しばしば便利です。

5.4.1 SUKILL - トレースのゼロ化

あるブロックのトレースをゼロ化するには、以下のようにタイプして下さい。

```
% sukill <stdin >stdout min=MIN_TRACE count=COUNT
```

ここで、COUNT はゼロ化するトレース数、MIN_TRACE はトレース・ブロック中の最小のトレース番号です。

5.4.2　SUNULL - 空トレースのパネルの生成

ゼロ値のトレースのパネルを作成することが必要になることがあります。時間サンプル数 NT の NTR 本のトレースのパネルを作成するには、以下のようにして下さい。

```
% sunull nt=NT ntr=NTR <stdin >stdout min=MIN_TRACE count=COUNT
```

5.4.3　SUZERO - 時間ウィンドウ内のデータのゼロ化

ある時間ウィンドウ内のデータをゼロ化するには **suzero** が使えます。

```
% suzero itmin=MIN_TIME_SAMPLE itmax=MAX_TIME_SAMPLE <indata.su > outdata.su
```

5.4.4　SUMUTE - データの外科的ミュート

前にあげたプログラム集はデータの粗野なミュートの方法を実行するものです。より精密なミュート操作のために、SU データの外科的ミュートを実行する **sumute** があります。このプログラムは、ミュートするトレース・ヘッダ・フィールドを “key=” パラメータを用いて指定して実行することができます。

```
% sumute <indata.su >outdata.su key=KEYWORD xmute=x1,x2,x3,... tmute=t1,t2,t3,...
```

ここの suplane データのサンプルでは

```
% suplane | suxwigb &
```

によって作られたオリジナルの suplane データと、

```
% suplane | sumute key=tracl xmute=1,10,12 tmute=.06,.1,.11 | suxwigb &
```

によって外科的にミュートされたデータとを比較して下さい。“xmute=” と “tmute=” の値で定義された多角曲線（折れ線）より上のすべての波動をミュートしています。

このプログラムでは、“nmute, xfile, tfile” オプションを設定することによって、バイナリ・ファイルから読み込んだ x, t 値を用いることもできます。このプログラムの selfdoc の一部を示します。

```
...
 nmute= number of x,t values defining mute
 xfile= file containing position values as specified by
  the ‘key’ parameter
 tfile= file containing corresponding time values (sec)
...
  =tracl  use trace number instead
 ntaper=0 number of points to taper before hard
mute (sine squared taper)
 below=0 =1 to zero BELOW the polygonal curve
...
```

ここで、“上 (above)” と “下 (below)” は **suxwigb** プロットのような地震データ表示の見え方を指すもので、時間の値を指すものではないことに注意して下さい。

5.5 SUVCAT, CAT - データの結合

あるデータセットを他のデータセットに追加する (結合する) には 2 つの方法があります。第 1 の方法は 2 番目のファイルのトレースが単に最初のファイルのトレースに続くように、一つのデータセットを他のデータセットに追加する方法です。これは UNIX の “cat” コマンドを用いて行えます:

```
% cat data1.su data2.su > data3.su
```

加えて、トレースのリナンバーが必要になるかもしれません:

```
% cat data1.su data2.su | sushw key=tracl a=1 > data3.su
```

こうすればパラメータ “tracl” は単調に増加します。

あるデータセットを他のデータセットに追加する 2 番目の方法は、2 番目のデータセットのそれぞれのトレースを最初のデータセットの最後に “鉛直に” 追加する【訳注：トレース毎にトレースをつなげて追加する】方法です。これは **suvcat** により行えます。

```
% suvcat data1.su data2.su > data3.su
```

この場合は、ヘッダ・フィールドの修正は必要ありません。

5.6 SUVLENGTH - 可変長トレースの共通サンプル数への調整

トレースごとにサンプル数が異なるトレースからなるデータがしばしばあります。suplane データを用いてこのサンプルを作ります。

```
% suplane nt=64 > data1.su
% suplane nt=32 > data2.su
% cat data1.su data2.su > data3.su
```

結果のファイル “data3.su” を用いて SU プログラムを使おうとしても結果は失敗するでしょう。なぜなら、大部分の SU プログラムでは、データのパネル上のサンプル数が一定であることが必要だからです。**suvlength** を適用してすべてのトレースを同じ長さにすることにより、この問題を解決します。

```
% suvlength ns=64 < data3.su > data4.su
% suxwigb < data4.su title="Test of suvlength" &
```

第6章　SUデータの一般的操作

以下の処理を実行する操作レベルのコードはSUデータの編集作業の域を越えています。

- ゲイン
- 再サンプリング
- ユナリ操作 (単一データ・ファイル内の演算操作)
- バイナリ操作 (2つのデータ・ファイル間の演算操作)

この節の目的は以下の操作を実行するプログラムについて説明することです。

- SUADDNOISE - トレースへのノイズの付加
- SUGAIN - トレース表示のため、各種のゲインの適用
- SUOP - SEGYデータの演算操作
- SUOP2 - 2データ・セット間のバイナリ演算操作

6.0.1　SUADDNOISE - SUデータへのノイズの付加

地震探査データにノイズを**加える**プログラムがあるというのは、逆効果のように思えるかもしれません。しかし、ノイズをシミュレートする機能はテストの目的にしばしば役立ちます。

また、デモンストレーションにも役立ちます。以下のデモの多くでも、suplaneのテストパターンの“空白を埋める”ために**suaddnoise**を用いています。suplaneデータを用いた、このプログラムの出力の例を2、3示します。

```
% suplane | suxwigb title="no noise" &
% suplane | suaddnoise | suxwigb title="noise added" &
% suplane | suaddnoise sn=2 | suxwigb title="noise added" &
```

6.0.2　SUGAIN - SUデータのゲイン操作

ゲイン操作という表題の下には非常に多くの操作があります。**sugain**はこれらを行います。これらの操作には以下が含まれます。

- データのスケーリング
- 時間のパワーとデータとの乗算
- データのパワーの算出
- 自動ゲイン制御
- ノイズ・スパイク・トレースのトラッピング
- 指定した振幅あるいは変位値 (quantile) でのクリッピング

- 変位値クリップ、RMS 値、平均値によるトレースのバランシング
- データのバイアスの添加と除去

操作の階層は次の方程式で説明されます。

```
out(t) = scale * BAL{CLIP[AGC{[t^tpow * exp(epow * t) * ( in(t)-bias )]^gpow}]}
```

sugain がどのように作用するか、suplane データを用いた以下のサンプルを走らせることによって理解できるでしょう。AGC の効果をはっきりさせるために **suaddnoise** を使ってノイズを加えました。パーセント記号%の後の項目のみをタイプして下さい。(ノイズを加えた) SU データを作成します。

```
% suplane | suaddnoise > data.su
```

```
% suxwigb < data.su title="Ungained Data"  &
% sugain < data.su scale=5.0 | suxwigb title="Scaled data"  &
% sugain < data.su agc=1 wagc=.01 | suxwigb title="AGC=1 WAGC=.01 sec " &
% sugain < data.su agc=1 wagc=.2 | suxwigb title="AGC=1 WAGC=.1 sec " &
% sugain < data.su pbal=1 | suxwigb title="traces balanced by rms" &
% sugain < data.su qbal=1 | suxwigb title="traces balanced by quantile" &
% sugain < data.su mbal=1 | suxwigb title="traces balanced by mean" &
% sugain < data.su tpow=2 | suxwigb title="t squared factor applied" &
% sugain < data.su tpow=.5 | suxwigb title="square root t factor applied" &
```

端末ウィンドウに “clip=数” というメッセージが表示されることに気をつけて下さい。例えば:

```
xwigb: clip=1
```

これはこの値以上の振幅値がクリップしたことを示します。この値をトレースの最大値と考えることもできます。

6.0.3　SUOP - SU データ内部の演算操作

時折、データのゲイン操作の域を越えて、数学関数やその他の操作を施したい場合があります。このような操作には、

- 絶対値
- 符号つき平方根
- 二乗
- 符号つき二乗
- 符号関数 (signum function)
- 指数関数
- 自然対数
- 符号つき常用対数
- コサイン
- サイン
- タンジェント

- ハイパボリック・コサイン
- ハイパボリック・サイン
- ハイパボリック・タンジェント
- 最大値によるトレースの除算
- トレースのデシベル表示: 20 * slog10 (データ)
- 符号の反転
- 正値のみの透過
- 負値のみの透過

などが含まれます。値に 0 が含まれている場合、対数に関する操作は “中断” し、値 0 が返されます。

suplane データを用いた **suop** のサンプルは次のようにして簡単に走らせることができます。

```
% suplane | suaddnoise > data.su
% suop < data.su op=abs | suxwigb title="absolute value" &
% suop < data.su op=ssqrt | suxwigb title="signed square root" &
% suop < data.su op=sqr | suxwigb title="signed square" &
...
```

selfdoc やその他のオプションを見るためには:

```
% suop
```

とタイプして下さい。

6.0.4 SUOP2 - SU データ間の演算操作

2 つの SU データセット間の操作を実行するために、**suop2** プログラムが与えられています。サポートされている操作は、

- 差
- 和
- 積
- 商
- パネルと単一トレースとの差
- パネルと単一トレースとの和
- パネルと単一トレースとの積
- パネルと単一トレースとの商

の計算です。始めの 4 つの操作はそれぞれの SU データファイル中のトレースの数は等しいと仮定しています。後の 4 つの操作では、2 番目のファイルには 1 本のトレースしかないと仮定しています。

suop2 の selfdoc を見ればわかりますが、これらの操作を実行する 8 つの等価なシェル・スクリプト・コマンドがあることに注意して下さい。

```
...
  susum file1 file2 == suop2 file1 file2 op=sum
  sudiff file1 file2 == suop2 file1 file2 op=diff
  suprod file1 file2 == suop2 file1 file2 op=prod
  suquo  file1 file2 == suop2 file1 file2 op=quo

 For:  panel "op" trace  operations:
  suptsum  file1 file2 == suop2 file1 file2 op=ptsum
  suptdiff file1 file2 == suop2 file1 file2 op=ptdiff
  suptprod file1 file2 == suop2 file1 file2 op=ptprod
  suptquo  file1 file2 == suop2 file1 file2 op=ptquo
...
```

これらすべては計算を実行するために **suop2** を呼び出します。

以下を試してみて下さい。**suplane** を用いて 2 つの SU データのファイルを作ります。

```
% suplane > junk1.su
% suxwigb < junk1.su | suxwigb title="Data without noise" &
% suplane | suaddnoise > junk2.su
% suxwigb < junk2.su  | suxwigb title="Data with noise added" &
% suop2 junk2.su junk1.su op=diff | suxwigb title="difference" &
```

ファイル名は “op=” の前に置かなければならないことに注意して下さい。

6.1　変換とフィルタ操作

地震探査の研究と地震探査データ処理の大半の面は、数学的**変換**に基づく操作に関係しています。特に、多くの地震探査データ処理は数値フーリエ変換なしではあり得ないでしょう。フィルタ操作も関連した問題です。というのも、大部分のフィルタは周波数領域で適用されるか、少なくとも周波数領域の操作として数学的に表現できるからです。

標準のフーリエ変換に加えて、2、3 の他の変換もあります。例えば、ヒルベルト (Hilbert) 変換とガボール (Gabor) 変換も SU パッケージに含まれています。これらは処理ツールというよりも、教育的ツールとして使い道があるでしょう。

6.1.1　フーリエ変換操作

SU パッケージには 1D と 2D の両方のアプリケーション用のフーリエ変換操作があります。1D 変換によって地震探査データのパネル中のそれぞれのトレースのスペクトル情報 (振幅あるいは位相) が与えられます。

2D 変換には地震データの F-K 変化、ここでは入力データの最初の次元が時間、2 番目の次元が空間と仮定しています、および、非地震データの K1-K2 変化が含まれます。後者では入力は純粋に空間的 (x1,x2) データであると仮定しています。

6.1.2　1D フーリエ変換

- SUFFT - 実数時間トレースの複素数周波数トレースへの FFT
- SUIFFT - 複素数周波数トレースの実数時間トレースのへの FFT
- SUAMP - (周波数, x) データから振幅、位相、トレースの実部あるいは虚部の出力

- SUSPECFX - トレース領域データのフーリエ・スペクトル (時間領域から周波数領域)

sufft プログラムはフーリエ変換操作の出力を複素数データ型として作成します。**suifft** プログラムは、逆フーリエ変換を実行するため、**sufft** によって作成された複素数の入力ができるように設計されています。これらの操作を連続して行うと、何も操作しない場合とまったく同じにはなりません。というのは、変換のために自動的にゼロ・パッディング (ゼロ値の追加) が行われるからです。ヘッダ・フィールドも地震探査データのものとは異なります。例えば、以下を試して下さい:

```
% suplane | suxwigb title="Original Data"  &
% suplane | sufft | suifft | sushw key=d1,dt a=0,4000 | suxwigb &
```

結果は入力と同じですが、変換に必要なゼロパッディングによってサンプル数が増えています。

振幅および位相スペクトルや、**sufft** の出力の実部および虚部を見るためには、以下のようにして下さい。

```
% suplane | sufft | suamp mode=amp | suxwigb title="amplitude" &
% suplane | sufft | suamp mode=phase | suxwigb title="phases" &
% suplane | sufft | suamp mode=real | suxwigb title="real" &
% suplane | sufft | suamp mode=imag | suxwigb title="imaginary" &
```

SU データには複素数型データタイプのデータの実部および虚部を保存することのできるフォーマットがあります。そのフォーマットのヘッダ・フィールドの設定を見るには、以下のようにタイプして下さい。

```
% suplane | sufft | surange
sufft: d1=3.571428
32 traces:
 tracl=(1,32)  tracr=(1,32)  trid=11 offset=400 ns=72
 dt=4000 d1=3.571428
```

トレース ID (trid) の設定が 11 であることに気づくでしょう。

```
% sukeyword trid
...
                        11 = Fourier transformed - unpacked Nyquist
                             xr[0],xi[0],...,xr[N/2],xi[N/2]
...
```

とタイプすることによって、“trid=11” の場合 FFT の出力中でデータがどのように並んでいるかが示されます。

もちろん、ほとんどの場合は、地震トレースあるいは地震トレースのパネルの振幅スペクトルをちょっと見たいだけです。そのためには **suspecfx** を使って下さい。

```
% suplane | suspecfx | suximage title="F-X Amplitude Spectrum" &
```

こうすると、入力 SU データのそれぞれのトレースの振幅スペクトルが直接表示されます。

6.1.3 2D フーリエ変換

地震探査データは通常は少なくとも 2D のデータセットです。データが実際に (時間, 空間) 座標のデータである場合、出力は (周波数, 波数) すなわち F-K 領域にあることになり

ます。しかし、データが 2 つの空間次元 (x1,x2) にあり、出力は (k1,k2) にあるとみなすアプリケーションもあります。

これらの場合のそれぞれにプログラムがあります。

- SUSPECFK - データ・セットの F-K フーリエ・スペクトル
- SUSPECK1K2 - (x1,x2) データ・セットの 2D (K1,K2) フーリエ・スペクトル

suplane データを使ったサンプルは以下のように簡単に走らすことができます。

```
% suplane | suspecfk | suximage title="F-K Amplitude Spectrum"  &
% suplane | suspeck1k2 | suximage title="K1-K2 Amplitude Spectrum"  &
```

これらは**表示**プログラムであるので、ここでは出力のプロットを元々のデータのプロットと比較して、正しいことを示そうと意図していることに留意して下さい。(k1,k2) データを (x1,x2) データのプロットと同じようにプロットした場合、スパイクの並びに対する 2D フーリエ変換の効果は、元々の並びと直行するスパイクの並びを生成します。

6.2　ヒルベルト変換、トレース属性、時間-周波数領域

瞬間的なトレース属性を表す古典的な手法が何年もの間数多く作られてきました。これらの手法の多くは、“複素トレース” の虚数部 (実数部が実データ) として用いるクアドラチャ・トレース (quadrature trace)【訳注 : 位相が 90° ずれたトレース】の構成に関するものです。クアドラチャ・トレースは、いわゆる “共役関数 (allied function)” の作成に続いてヒルベルト変換を用いて作成することができます。この表現を用いると、それぞれ係数、位相、位相の時間微分をとることにより、データセットの “瞬間振幅、位相、周波数” 情報を作成することができます。もう一つのアプローチとして、データに多重フィルタ解析を行い時間と周波数の関数として表す方法があります。

これらの操作を行う SU のツールは、

- SUHILB - ヒルベルト変換
- SUATTRIBUTES - トレース属性、すなわち、瞬間振幅、位相、周波数
- SUGABOR - Gabor 変換に似た多重フィルタ解析による地震探査データの時間-周波数表現の出力。

テスト・データセットのヒルベルト変換を作成するには、以下を試して下さい。

```
% suplane | suhilb | suxwigb title="Hilbert Transform" &
```

このプログラムは、指導および試験の目的に役立ちます。

suattributes を用いて “トレース属性” のサンプルを見るには、強い時間-周波数変化を持つデータを作成する必要があります。ここでは、合成バイブロサイス・スイープを作成する **suvibro** を用いて行います。以下を較べて下さい。

```
% suvibro | suxgraph title="Vibroseis sweep" &
% suvibro | suattributes mode=amp | suxgraph title="Inst. amplitude" &
% suvibro | suattributes mode=phase unwrap=1.0 | suxgraph title="Inst. phase" &
% suvibro | suattributes mode=freq | suxgraph title="Inst. frequency" &
```

これらは、それぞれ瞬間振幅、位相、周波数を示します。

時間-周波数領域での合成バイブロサイス・トレースを見るには、以下を試して下さい。

```
% suvibro | sugabor | suximage title="time frequency plot" &
```

結果のイメージでは瞬間周波数または見掛け周波数が時間と共に 10 Hz から 60 Hz まで増加しており、**suvibro** のデフォルト・パラメータに宣言されているのとまったく同じです。

より詳しい情報は$CWPROOT/src/demos/Time_Freq_Analysis および$CWPROOT/src/demos/Filtering/Sugabor のデモを見て下さい。

6.3 ラドン変換 - τ-p フィルタリング

ラドン (Radon) 変換は、地球物理学の文献では “τ-p” 変換と呼ばれることもありますが、多重反射波の抑制、その他の “外科的な” データ操作作業に役立ちます。この変換を利用するために以下のプログラムが提供されています。

- SUTAUP - 順および逆 T-X および F-K グローバル・スラント・スタック
- SUHARLAN - Harlan (1984) の可逆線形変換法 (invertible linear transformation method) による信号-ノイズ分離
- SURADON - 順あるいは逆ラドン変換、多重反射の推定と減算に放物ラドン変換を用いた多重反射の除去
- SUINTERP - 自動イベント・ピッキングによるトレースの内挿

その他のプログラムでも行ってきたように、これらのコードの出力を見るために suplane のデータを使ってテストすることもできます。**suinterp** と **suradon** には凝ったオプションがあり、理解するには少し試してみる必要があります。

例えば$CWPROOT/src/demos/Tau_P のデモも見て下さい。

6.4 1D フィルタ操作

地震探査データ処理と呼ばれている技術の多くは “フィルタリング” であると考えてもよいでしょう。SU パッケージには、簡単なフィルタリング作業から、例えばデコンボリューションやウェーブレット・シェーピング操作のようなより複雑な作業までにおよぶ 1D アプリケーションのフィルタ操作があります。これらの操作は 1D で、トレース毎に適用されます。

地震データ処理に現れるフィルタ操作には以下のようなものがあります。

- ゼロ位相バンド・パス、バンド・リジェクト、ロー・パス、ハイ・パスおよびノッチ・フィルタリング
- 最小あるいはゼロ位相 Butterworth フィルタリング
- Wiener 予測誤差 (デコンボリューション)
- Wiener ウェーブレット・シェーピング
- コンボリューション
- 相互相関
- 自己相関
- sinc 関数を用いた内挿によるデータ・再サンプリング
- 分数次の微分/積分

- メディアン・フィルタリング
- タイム・バリアント・フィルタリング

これらのニーズに合うために提供されているプログラムは以下のものがあります。

- SUFILTER - ゼロ位相サイン二乗打ち切りフィルタの適用
- SUBFILT - Butterworth バンドパス・フィルタの適用
- SUACOR - 自己相関
- SUCONV, SUXCOR - ユーザー提供のフィルタを用いたコンボリューションと相関
- SUPEF - Wiener の予測誤差フィルタリング
- SUSHAPE - Wiener のシェーピング・フィルタ
- SURESAMP - 時間軸上のリサンプル
- SUFRAC - 一般の（分数次の）データの時間微分、または積分、プラス位相シフト。入力は時間領域のデータ
- SUMEDIAN - キー・ヘッダ・ワードで指定された曲線に沿った距離を用いた、ユーザー定義の多角曲線（折れ線）についてのメディアン (MEDIAN) フィルタ。
- SUTVBAND - タイム・バリアント（時間可変）なバンドパス・フィルタ (サイン二乗テーパ)

6.4.1　SUFILTER - ゼロ位相サイン二乗打ち切りフィルタの適用

sufilter プログラムはバンドパス、バンドリジェクト、ローパス、ハイパスおよびノッチ・フィルタリングなどの通常の作業のための一般目的ゼロ位相フィルタリング機能を提供します。suplane データを用いた **sufilter** の例は以下に与えられています。

```
% suplane | sufilter f=10,20,30,60 amps=0,1,1,0 |
    suxwigb title="10,20,30,60 hz bandpass"  &
% suplane | sufilter f=10,20,30,60 amps=1,0,0,1 |
    suxwigb title="10,20,30,60 hz bandreject" &
% suplane | sufilter f=10,20,30,60 amps=1,1,0,0 |
    suxwigb title="10,20 hz lowpass" &
% suplane | sufilter f=50,60,70 amps=1,0,1 | suxwigb title="60 hz notch"  &
```

フィルタは多角形状で、多角形の角は “f=” 値の配列で定義される周波数の値のベクトルと “amps=” 値の配列で定義される振幅値の集合とで定義されます。振幅値はゼロ以上の任意の浮動小数点数で構いません。ただ一つの規則として “amps” 値と “f” 値の数が同じでなければなりません。異なる振幅値の “f” 値の間の断面はサイン二乗で丸められます。リンギングを防ぐために、1 オクターブを越えてゼロへの丸めを行うような “f” 値を選択することがベストです。

suplane を用いて **sufilter** は SU フォーマットの帯域制限されたテストパターン・データを簡単に生成することができます。

6.4.2　SUBFILT - Butterworth バンドパス・フィルタの適用

sufilter の代わりになるものが **subfilt** で、Butterworth フィルタをデータに適用します。

```
% suplane | subfilt fstoplo=10 fpasslo=20 fpasshi=30 fstophi=60 |
    suxwigb title="10,20,30,60 hz  bandpass bfilt"  &
```

6.4.3 SUACOR - 自己相関

このプログラムはトレースの自己相関を計算するために用いられます。このプログラムは、ウェーブレットのサイズやウェーブレットの繰り返し性を調べたり、**supef** の “maxlag” パラメータを選択するために役立ちます。また、**suacor** はパワー・スペクトルに関してデータの周波数範囲を決定するためにも有効です。例えば、以下を試して下さい。

```
% suplane | sufilter | suacor | suspecfx | suxwigb &
```

6.4.4 SUCONV, SUXCOR - ユーザー提供のフィルタを用いたコンボリューション、相関

コンボリューションや相互相関の標準的操作はそれぞれ **suconv** および **suxcor** を用いて実行できます。フィルタはコマンドライン上のベクトル入力か、単一のトレースを含む SU フォーマットのファイルとして与えられます。単一のトレースとして入力をできることに加えて、フィルタのパネルを与えて各々をトレースごとに SU データのパネルに対して用いることもできます。

suvibro、**suplane** および **suconv** を用いてバイブロサイス状のデータを作成し、バイブロサイス・スウィープの相関処理の例を見ることができます。

```
% suvibro > junk.vib.su
% suplane | suconv sufile=junk.vib.su > plane.vib.su
```

surange によって：

```
% surange < junk.vib.su
1 traces:
 tracl=1 ns=2500 dt=4000 sfs=10 sfe=60
 slen=10000 styp=1
```

バイブロサイス・スウィープは 2500 サンプルであることがわかります。以下の相関を行うことができます。

```
%  suxcor < plane.vib.su sufile=junk.vib.su |
     suwind itmin=2500 itmax=2563 | sushw key=delrt a=0.0 > data.su
```

(この行はページに合わせるために切ってありますが、実際のコマンドは 1 行でタイプします。)

“itmin=sweeplength” と “itmax=sweeplength+nsout” の値が出力に設定されます。ここで “nsout” はサンプル数です。最後のステップは **sushw** を用いてトレースの遅れを 0 に設定することです。“itmin=sweeplength” を選択することでデータが正しい値から開始することが保証されます。“nsout=sweeplength-nsin” とすることで正しいサンプル数を保存します。ここで “nsin” は入力のサンプル数です。

6.4.5 SUPEF - Wiener の予測誤差フィルタリング

予測誤差フィルタ法は Wiener フィルタリングとしても知られていますが、伝統的な Wiener-Levinson デコンボリューションの原理となる処理です。この項が “デコンボリューション” と見出しづけられていないのは、予測誤差フィルタリングが効果的なデコンボリューション

に用いられる前に、触れておかなければならない 2 つの付随的項目があるからです。それらの項目はデータの前処理とフィルタリングの後処理です。実際、**supef** が適切に機能しないと訴える多くのフィードバックが返っていますが、実のところは単に不適切に使われているからです。(充分な説明を与えていないという点で、これはわれわれの落ち度です。)

supef を用いるには一般に **maxlag** の値を設定する必要があります。これは、**suacor** を適用して、スパイク化されるウェーブレットのサイズを最初に推定することにより決定できます。

前処理のステップでは、**sugain** を用いて幾何学発散の結果である時間による振幅の減衰を除去する必要があるでしょう。

後処理のステップでは、予測誤差フィルタのホワイトニング効果によって生じる周波数成分の増加を除去する必要があります。demos/Deconvolution ディレクトリ中のデモによってプログラムの機能を例示します。しかし、これらの例はフィールド・データとしてはちょっと非現実的です。例えば、デモの中では、データをスパイク化して重複反射を除去するために、それぞれ “maxlag=.04” と “minlag=.05 maxlag=.16” として **supef** をつなげて呼び出しています。しかし、フィールド・データでは重複反射を除去したり波形をスパイク化するにはおそらく 1 回のパスしか使えないでしょう。

予測誤差フィルタには周波数をホワイト化する効果がありますが、これにより元々なかった高周波数をデータに付け加えることになります。これはフィルタで落とさなければなりません。周波数情報の**ロス**があると仮定し、**sufilter** を用いて周波数成分を元々の値からわずかに減少させるように、後処理のパラメータを設計するのがよい考えでしょう。

また、Release 32 バージョンの **supef** に追加された機能がミキシング・パラメータです。これにより、予測誤差フィルタ計算の一部として計算する自己相関にユーザーが重み付き移動平均を適用できるようになりました。これによりさらに操作の安定がもたらされました。

6.4.6　SUSHAPE - Wiener のシェーピング・フィルタ

demos/Deconvolution 内には Wiener シェーピング・フィルタ **sushape** のデモも含まれています。

6.4.7　2D フィルタ操作

(k1, k2) 領域、(F, K) 領域におけるフィルタ操作はデータのディップ (傾斜) 情報を変えるのに役立ちます。以下のプログラム：

- SUKFILTER - 軸対称 K 領域、サイン二乗打ち切りの多角フィルタ
- SUK1K2FILTER - 対称箱型 K 領域フィルタ。k1 および k2 で定義された 2 つのサイン二乗打ち切りの多角フィルタのカルテシアン積で定義される。
- SUKFRAC - データへの $i|k|$ の小数ベキの適用。位相シフトつき
- SUDIPFILT - f-k 領域での DIP—よりよくは—SLOPE フィルタ

は、基本的な一連の K 領域および F-K 領域のフィルタ操作を提供します。

6.4.8 SURESAMP - 時間軸上のデータの再サンプル

処理やデータの保存に際して、サンプル数を減らしたり増したりするために、データの再サンプルをする必要がよくあります。地震データに関して、sinc 関数を用いて内挿する方法がスマートなやり方です。**suresamp** プログラムはこの操作を行います。

- SURESAMP - 時間軸上のデータの再サンプル

SU でのフィルタリングに関するより詳しい情報は$CWPROOT/src/demos/Filtering 内のデモを見て下さい。

第7章　地震波モデリング・ユーティリティ

地震探査および研究において、合成データを作成するプログラムは重要な一面です。そのようなプログラムは、実データのモデリングという実用的な問題においても、新しい処理プログラムをテストする場合においても役立つ道があります。理想化されたモデル・データで動かない処理プログラムは、実際の地震データでも動かないでしょう。

モデリング・プログラムのもう一つの重要な面は、多くの地震探査処理アルゴリズム (マイグレーションのような) は**逆処理** *(inverse processes)* として見ることができるという点です。そのような逆問題における最初のステップは順問題を解く手法を作成することであり、その後、記録されたデータの地下の位置への “逆伝播 (backpropagation)” として逆問題の解を定式化します。

地震波モデリングの作業は 2 つのパートに分けられます。最初のパートは、バックグラウンドの速度プロファイルを作成することです。このプロファイルは均一にサンプリングされた浮動小数点実数の配列からなります。2 番目のパートはその速度プロファイル中を伝播する合成地震波の情報を作成することです。

地震波モデリングと地震波処理とは密接な関係があるので、モデリング作業のために作成されたバックグラウンドの速度プロファイルは処理作業にも役に立ちます。

もちろん、何らかの簡単な近似を行えば、バックグラウンドの速度情報をモデリング・プログラムに組み込むことは可能です。

7.1　バックグラウンドの速度プロファイル

バックグラウンドの速度プロファイルを作成する方法にはいくつかのアプローチがあります。多くの処理では、速度プロファイルは浮動小数点実数の配列で、それぞれが均一にサンプルされた格子点上の速度、あるいはスローネス (1/速度)、あるいは sloth (スローネスの二乗) を表すという方法で十分です。

しかし、もっと進んだ方法に、三角形分割あるいは四面体分割した媒質の速度プロファイルを作成する方法があります。

7.2　均一にサンプルされたモデル

Seismic Unix には、バックグラウンドの速度プロファイルを作成するために用いられるプログラムがいくつかあります。そのようなデータはしばしば平滑化する必要があります。

それらのプログラムは以下の通りです:

- UNISAM - x,y のペアで指定された関数 y(x) の均一サンプル
- UNISAM2 - 2-D 関数 f(x1,x2) の均一サンプル
- MAKEVEL - 速度関数 v(x,y,z) の作成

- UNIF2 - 層構造からの 2-D の均一にサンプルされた速度プロファイルの作成。各層内では、速度は位置の線形関数である。
- SMOOTHINT2 - 重み付き最小二乗法による、不均一にサンプルされた境界面の平滑化
- SMOOTH2 - 重み付き最小二乗法による、ユーザー定義ウィンドウ内での均一にサンプルされたデータの 2-D 配列の平滑化
- SMOOTH3D - 重み付き最小二乗法による 3-D 速度グリッドの平滑化

より詳しい情報はそれぞれのプログラムの selfdoc を見て下さい。また、$CWPROOT/src/Velocity_Profiles 内のデモも見て下さい。

7.3　合成地震データ作成プログラム

SU パッケージには、合成地震データ、および地震データのようなデータを生成するプログラムがたくさんあります。

- SUPLANE - 3 平面までの共通オフセット・データ・ファイルの作成
- SUSPIKE - 小さなスパイクからなるデータ・セットの作成
- SUIMP2D - Born 積分方程式を用いた 3 次元媒体内の線散乱源に対するショット記録の作成
- SUIMP3D - Born 積分方程式を用いた 3 次元媒体内の点散乱源に対する面内のショット記録の作成
- SUFDMOD2 - 音響波動方程式の (2 次) 有限差分法モデリング
- SUSYNCZ - V(Z) 関数を用いた 2.5D 真振幅 (プライマリーのみ) モデリングによる合成地震記録
- SUSYNLV - 線形速度関数に対する合成地震記録
- SUSYNVXZ - Kirchhoff 型モデリングによる V(X,Z) 媒質の共通オフセット記録の合成地震記録
- SUSYNLVCW - 線形速度関数に対するモード変換波の合成地震記録
- SUSYNVXZCS - Kirchhoff 型モデリングによる V(X,Z) 媒質の共通ショット記録の合成地震記録

sufdmod2, **susynvxz**, **sysnvxzcs** にのみ入力の伝播速度のファイルが必要です。その他のプログラムは速度モデルの入力にコマンドライン引数を用います。

詳しい情報は$CWPROOT/src/demos/Synthetic 内のデモを見て下さい。また、多くの他のデモでも合成データを作成する際にこれらのプログラムを使用しています。

7.4　Delaunay 三角分割

合成データの作成のより高度な方法では、合成データの計算速度を促進するために、(入力速度プロファイル・データ・フォーマットによって表現される) 媒質の性質に関して仮定を用います。そのような方法の一つに Delaunay 法を用いた三角分割があります。

7.4.1 三角分割化されたモデルの作成

三角分割化モデルを作成する方法には 2 通りあります。第 1 の方法は **trimodel** を用いて境界の座標と伝播速度 (実際には sloth 値) を陽的に入力する方法です。第 2 の方法は、先に述べたモデル作成ユーティリティを用いて均一にサンプルされたモデルを作成し、その後、**uni2tri** を用いて均一にサンプルされたモデルを三角分割化モデルへ変換する方法です (変換に先立って伝播速度モデルが平滑化されている方がよい結果となります。)。そのようなプログラムに以下のものがあります。

- TRIMODEL - 三角分割化 sloth (1/速度の二乗) モデルの作成
- UNI2TRI - 均一にサンプルされたモデルの三角分割化モデルへの変換
- TRI2UNI - 三角分割化モデルの均一にサンプルされたモデルへの変換

7.4.2 三角分割された媒質での合成地震データ

三角分割されたモデルを用いて波線追跡 (ray-tracing) を行ったり、波線追跡に基づいた合成地震記録を作成するプログラムがいくつかあります。また、三角分割された媒質中でガウシアン・ビーム (Gaussian beam) 法による合成地震記録を作成するコードもあります。これらのプログラムは以下の通りです:

- NORMRAY - sloth モデルでの垂直入射波線の動的波線追跡 (dynamic ray tracing)
- TRIRAY - 三角分割された sloth モデルでの動的波線追跡
- GBBEAM - sloth モデルでのガウシアン・ビーム (Gaussian beam) 法による合成地震記録
- TRISEIS - sloth モデルでの合成地震記録

わかりやすいデモのセットが$CWPROOT/src/Delaunay_Triangulation ディレクトリにあります。

7.5 四面体法

四面体モデルの作成および四面体モデルでの波線追跡法に関するいくつかの新しい機能が SU の将来のリリースに取り入れられるでしょう。現在のリリースに含まれているコードは次のものです。

- TETRAMOD - 四面体モデル作成プログラム。各層内では速度勾配は一定あるいは 2-D グリッド。水平面は均一なグリッドまたは指定した 2-D グリッドで追加できる。

第8章　地震探査データ処理ユーティリティ

地震探査データ処理に特有の操作のプログラム集があります。それらは地震波データを地下のイメージへ変換する複雑な処理を実行するように設計された操作のことです。

- データの重合
- データのピッキング
- 速度解析
- NMO (Normal Moveout) 補正
- DMO (Dip Moveout) 補正
- マイグレーションと関連する操作

8.1　SUSTACK, SURECIP, SUDIVSTACK - データの重合

- SUSTACK - 同じキー・ヘッダ・ワードをもつ隣接するトレースの重合
- SURECIP - データ中の反対のオフセットのトレースの加算
- SUDIVSTACK - ウインドウ内の平均パワーあるいはピーク・パワーを用いたダイバーシティ・スタック (Diversity Stacking)

8.2　SUVELAN, SUNMO - 速度解析と NMO (Normal Moveout) 補正

- SUVELAN - CDP ギャザーの重合速度センブランスの計算
- SUNMO - 時間と CDP に関する任意の速度関数による NMO

詳しい情報を得るためには$CWPROOT/src/demos/Velocity_Analysis と$CWPROOT/src/demos/NMO にあるデモを見て下さい。

8.3　SUDMOFK, SUDMOTX, SUDMOVZ - DMO (Dip Moveout) 補正

DMO (Dip-moveout) 処理はオフセットのあるデータをゼロ・オフセットのデータに変換するデータ変換です。以下のプログラムがこの操作を実行します。

- SUDMOFK - 共通オフセット・ギャザーの F-K 領域 (ログ・ストレッチ: log-stretch) 法による DMO

- SUDMOTX - 共通オフセット・ギャザーの T-X 領域 (キルヒホッフ: Kirchhoff) 法による DMO
- SUDMOVZ - 共通オフセット・ギャザーの V(Z) 媒質における DMO

8.4　地震波マイグレーション

地震波マイグレーションに関する問題は地震探査データ処理の中で最も変化に富んでいるものの一つです。この作業を実行するために多くのアルゴリズムが開発されてきました。この方法には、キルヒホッフ (Kirchhoff)、Stolt、有限差分 (Finite-Difference)、フーリエ変換-有限差分 (Fourier Finite-Difference) およびいくつかのタイプの位相シフト (Phase-Shift) あるいは Gazdag マイグレーションなどがあります。

より詳しい情報は$CWPROOT/src/demos/Migration 内のデモを見て下さい。

8.4.1　SUGAZMIG, SUMIGPS, SUMIGPSPI, SUMIGSPLIT - 位相シフト (Phase Shift) マイグレーション

- SUGAZMIG - ゼロ・オフセット・データの Jeno GAZDAG の位相シフト・マイグレーションの SU バージョン
- SUMIGPS - 屈曲波線を用いた位相シフト・マイグレーション
- SUMIGPSPI - トレース内挿を加えた、ゼロ・オフセット・データの Gazdag 位相シフトマイグレーション。速度の横方変化を扱える。
- SUMIGSPLIT - ゼロ・オフセット・データのスプリット・ステップ (Split-step) 深度マイグレーション

8.4.2　SUKDMIG2D, SUMIGTOPO2D, SUDATUMK2DR, SUDATUMK2DS - 2D キルヒホッフ (Kirchhoff) マイグレーションとデータミング

- SUKDMIG2D - 2D 重合後/重合前データのキルヒホッフ深度マイグレーション
- SUDATUMK2DR - 2D 重合前データの受振点のキルヒホッフ・データミング (入力はショット・ギャザー)
- SUDATUMK2DS - 2D 重合前データの震源のキルヒホッフ・データミング (入力はレシーバー・ギャザー)
- SUMIGTOPO2D - (地形が変化する) 記録地表面からの 2D 重合後/重合前データのキルヒホッフ深度マイグレーション

8.4.3　SUMIGFD, SUMIGFFD - 有限差分 (Finite-Difference) マイグレーション

- SUMIGFD - ゼロ・オフセット・データの 45° 近似および 60° 近似有限差分マイグレーション
- SUMIGFFD - ゼロ・オフセット・データのフーリエ変換有限差分 (Fourier finite difference) マイグレーション。この方法は、位相シフト・マイグレーションと有限差分マイグレーションの利点を合わせたマイグレーション法です。

8.4.4 SUMIGTK - 時間–波数領域マイグレーション

このアルゴリズムは Dave Hale によって “急いで (on the fly)” 作られたもので、私たちの知る限り、他の地球物理学の文献にはどこにもありません。

- SUMIGTK - 共通反射点重合後データの T-K 領域法によるマイグレーション

8.4.5 SUSTOLT - Stolt マイグレーション

これは、Clayton Stolt の古典的な F-K マイグレーション法です。

- SUSTOLT - 重合後データまたは共通オフセット・ギャザーの Stolt マイグレーション

第9章　SUによる処理の流れ

9.1　SUとUNIX

SUを使用するためには、特別な地震探査言語を勉強する必要はありません。UNIXのシェル・リダイレクトとパイプの使い方を知っていれば、SUを使い始めることができます—地震探査処理用のコマンドやオプションは、UNIXコマンドを使うときとまったく同様に使うことができます。特に、通常のUNIXシェル・スクリプトを書いて、よく使うコマンドの組み合わせをメタ・コマンド (すなわち、処理の流れ) にまとめあげることができます。スクリプトは“ジョブ・ファイル”と考えることができます。

表 9.1: UNIX の記号

process1 < file1	process1は入力をfile1から読み込む
process2 > file2	process2は出力を (新しい) file2へ書き込む
process3 >> file3	process3は出力をfile3へ追加する
process4 | process5	process4の出力はprocess5の入力へ渡される
process6 << text	次の行から読み込む

ですから、表9.1にまとめたような、基本的なUNIXの操作のレビューから始めましょう。<, >そして>>という記号は、“リダイレクション操作”として知られています。それらによって、コマンド (すなわち処理) への入力やコマンドからの出力をリダイレクトします。|という記号は“パイプ”と呼ばれます。それは、ある処理から次の処理へ“パイプ”を通して流れていくデータを思い描くことができるからです。以下は、“indata”という入力と“outdata”という出力を使った簡単なSU“パイプライン”の例です。

```
% sufilter f=4,8,42,54 <indata |
% sugain tpow=2.0 >outdata
```

このサンプルは、帯域制限操作の結果が“パイプ”によってゲイン操作へ渡されることを示しています。入力データ・セットである`indata`は、<オペレータによって、**sufilter**プログラムに読み込まれます。同様に、出力データ・セットの`outdata`は>オペレータによってデータを受け取ります。**sufilter**の出力は|オペレータによって、**sugain**の入力と連結されます。

=記号のある文字列は、どのようにしてパラメータをSUプログラムに渡したらよいかを示しています。**sugain**プログラムは、`tpow`というパラメータに2.0という値を受け取っており、一方、**sufilter**プログラムは、`f`というパラメータに4つの成分からなる**ベクトル**を受け取っています。そのプログラムにどのようなパラメータがあるかを調べるには、self-doc機能を使います。

ところで、UNIXのリダイレクションとパイプの記号の周囲のスペースは任意です—サンプルは典型的な一例を示しています。一方、=オペレータの周囲のスペースは認められま

せん。

表 9.1 の最初の 4 つの記号は UNIX の基本文法です。最後のエントリー << はあまり使われませんが “ヒア・ドキュメント” リダイレクションに使われる記号です。会話的にはめったに用いられませんが、上手に << オペレータを用いることにより、SU シェル・プログラムの機能を強力にすることができます。これについては後述します。

多くのビルト・イン UNIX コマンドには SU のような自己文書 (self-documentation) 機能はありません。代わりに、“man” ページがあります。例えば、

```
% man cat

CAT(1)              UNIX Programmer's Manual               CAT(1)



NAME
     cat - catenate and print

SYNOPSIS
     cat [ -u ] [ -n ] [ -s ] [ -v ] file ...

DESCRIPTION
     Cat reads each file in sequence and displays it on the stan-
     dard output.  Thus

                    cat file

     displays the file on the standard output, and

                    cat file1 file2 >file3
--More--
```

SU を有効に活用するためには、もう少し UNIX について知らなければいけません—この章で後述するサンプルを用いて、そのような秘訣を紹介することにします。

9.2　SU シェル・プログラミングの理解と使用方法

SU を上手につかう本質は、処理の流れを作り記録しておくための UNIX シェル・プログラムを作る (複製する！) ことです。この節では、SU を始めるために、注釈付きのサンプルをいくつか用意することにします。

9.2.1　SU による処理の流れの簡単な例

ほとんどの SU プログラムは標準入力からデータを読み、標準出力に出力するようになっています。ですから、単に SU プログラムを UNIX のパイプでつなぐことによって、複雑な処理の流れを構築することができます。ほとんどの処理は SU の描画プログラムで終わります。処理の流れは長くなり、多くのパラメータをセットする必要があるのが普通なので、SU コマンドをシェル・ファイル中に入れるのが便利です。

注目: すべての UNIX シェル, Bourne (sh), Cshell (csh), Korn (ksh),..., にはプログラミング言語があります。この文書では、Bourne シェルのプログラミング言語のみを使うことにします。

最初の例は **Plot** という簡単なシェル・プログラムです。以下のリスト中の行の最後の [] 括弧の中の数字はシェル・プログラムの一部ではありません—リストの説明のためのキーとしてつけたものです。

```
#! /bin/sh                                                    [1]
# Plot:   Plot a range of cmp gathers
# Author: Jane Doe
# Usage:  Plot cdpmin cdpmax

data=$HOME/data/cmgs                                         [2]

# Plot the cmp gather.
suwind <$data key=cdp min=$1 max=$2 |                        [3]
sugain tpow=2 gpow=.5 |
suximage f2=0 d2=1 \                                         [4]
        label1="Time (sec)" label2="Trace number" \
        title="CMP Gathers $1 to $2" \
        perc=99 grid1=solid \&                               [5]
```

番号をつけた行の説明:

1. #記号はコメントを表します—この後ろに何が書いてあっても UNIX シェルは実行しません。#!はこの規則の例外です: シェルは、この記号の後のファイル名をシェル・プログラムを実行するプログラムのパスとして用います。

2. 著者は明らかに、データ・セットを変更する必要がある場合はシェルを編集しようと考えています—シェル変数 data を導入し、データ・ファイルのフル・パス名を割り当てることによって、これを容易にしています。割り当てられた値にはシェル・プログラム中で$data としてアクセスできます。ファイルのパス名の最初の成分として表れている$HOME は UNIX が保持している環境変数で、ユーザーのホーム・ディレクトリのパスを含んでいます。一般に、データがユーザーのホーム・ディレクトリに置かれている必要はありませんが、シェル・プログラムが動くためにはデータファイルに “読み出し許可” が必要です。

   警告! UNIX シェルでは空白は大事です—空白はコマンドラインを分けるために使われています。コードは読み易いものにせよと学んできましたが、=記号の次に空白を入れ**ない**で下さい (1977 頃、著者の一人 (Jack) は初めて UNIX を学ぼうとして、この間違いを犯して数週間も”脱線”してしまいました。)。

3. このシェル・コードのメインのパイプラインでは、**suwind** で特定の CMP ギャザーを選択し、**sugain** でこのサブセットを増幅し、結果を表示プログラム **suximage** にパイプで渡しています。使用法のコメントに示してあるように、CMP の範囲はコマンド・ライン引数で指定されます。シェル・プログラム中では、これらの引数は$1, $2 (すなわち、第 1 引数、第 2 引数) として参照されます。

4. **suximage** コマンドの行はバックスラッシュ・エスケープ文字で継続されています。

   警告! 接続行バックスラッシュはその行の**最後の**文字でなければなりません—バックスラッシュの後にある見えない空白やタブは UNIX シェル・プログラミングにおける最もよくある、うっとうしいバクの一つです。

5. シェル・プログラム中の最後の & は表示ウインドウを “バックグラウンド” に置きます。そのため、メイン・ウインドウで計算を続けることができます。これは X-Windows に

おける用法です— & は同等の PostScript 出力プログラム (例えば supsimage) では使えません。例えば、**suximage** の代わりに **supsimage** を使うときは、& は| lpr に置き換えて下さい。

SU の表示プログラムは特殊です—それらの self-doc には可能なすべてのパラメータが表示されるわけではありません。例えば、**suximage** に受け付けられる大部分のパラメータは、実際には、一般的な CWP の表示プログラムである **ximage** の self-doc 中で説明されています。この self-doc の明らかな欠陥は、実のところ、基本的な SU の設計の副作用です。すなわち、SU のグラフィックス・プログラムは実際の表示を行うときに一般的な表示プログラムを呼び出します。他のやり方として、グラフィックス・プログラムをさまざまな地震探査アプリケーション用に調整するという設計の仕方もあります。私達の設計上の選択はシンプルですが、同時に SU のグラフィックスの能力の基本的な限界も示唆しています。

表示プログラムは結果を表示する手段です。ですから、時間をかけて “SU のジャケット (覆いの)” プログラム (**suximage**, **suxwigb**,...) と一般的な表示プログラム (**ximage**, **xwigb**, ...) の**両方**の self-documentation に目を通しておくのがよいでしょう。

9.2.2　シェル・プログラムの実行

UNIX シェル・プログラムを実行する最も簡単な方法は、“実行可能のパーミッション” を与えることです。例えば、前記の **Plot** シェル・プログラムを実行可能にするには次のようにします。

```
% chmod +x Plot
```

その後、シェル・プログラムを実行して下さい。:

```
% Plot 601 610
```

ここで、パラメータ cdpmin=601, cdpmax=610 は cmgs データ・セットについて適切な値であると仮定しています。図 9.1 に、Plot シェル・プログラムによって作成された出力を示します。

9.2.3　典型的な SU の処理の流れ

sudmofk を使いたいとします。self-doc はもう読んだでしょうが、でも、詳しいサンプルはいつでも大歓迎ですよね？見るべき場所は、**su/examples** ディレクトリです。この場合は、運よく **Dmo** というシェル・プログラムを見つけました。ここでもまた、以下で行の末尾の［］括弧内の数はリストの一部ではあり**ません**。

```
#! /bin/sh
# dmo
set -x                                                          [1]

# set parameters
input=cdp201to800                                               [2]
temp=dmocogs
output=dmocmgs
smute=1.7
vnmo=1500,1550,1700,2000,2300,2600,3000                         [3]
```

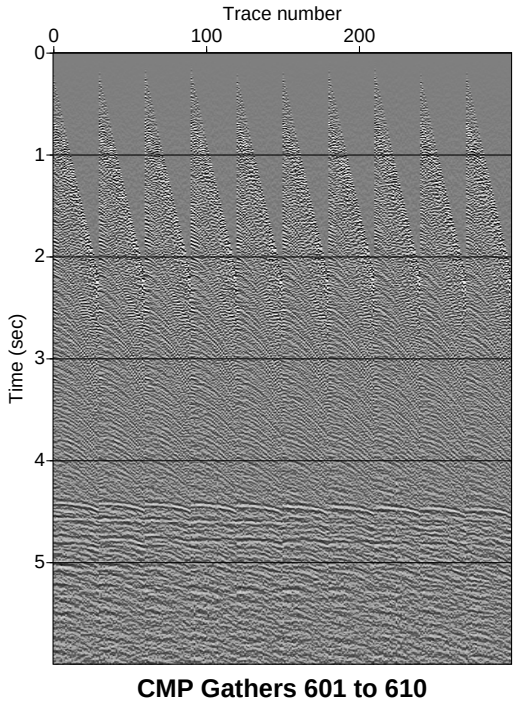

図 9.1: `Plot` シェル・プログラムの出力

```
tnmo=0.00,0.40,1.00,2.00,3.00,4.00,6.00


# sort to common-offset, nmo, dmo, inverse-nmo, sort back to cmp
susort <$input offset cdp |                                          [4]
sunmo smute=$smute vnmo=$vnmo tnmo=$tnmo |                           [5]
sudmofk cdpmin=201 cdpmax=800 dxcdp=13.335 noffmix=4 verbose=1 |     [6]
sunmo invert=1 smute=$smute vnmo=$vnmo tnmo=$tnmo >$temp             [7]
susort <$temp cdp offset >$output                                    [8]
```

番号の行の説明:

シェル・プログラムの中心 (行 5-7) は典型的な DMO 処理になっています: “そのままの”NMO、DMO そして “逆”NMO です。DMO 処理はソート操作 (行 4 と 8) にはさまれています。ここでは、リストに加えた番号で示されたシェル・プログラムを詳しく説明します (前出の `Plot` シェルの説明も見て下さい):

1. デバッグ・モードを設定します。UNIX に実行された行を返すよう指示します。出力が必要なくなった時にはコメントにすることができます。別のデバッグ用フラグは `set -v` です。これはシェル・インタープリタに読み込まれた行を返します。両方のモードを同時に用いることもできます。
2. この行と次の 2 行はファイル名を設定します。この場合はどちらのファイルもシェル・プログラムと同じディレクトリにあります。ここでもパラメータを用いている理由は、他のデータ・セットを一緒に用いるときにシェルを “複製” するのを容易にするためです。少しのデータ・セットと与えられた時間内で処理をする私達のような人にとっては、処理パラメータの説明文で用いているように、与えられたデータ・セットをある

一つのディレクトリに入れて、シェルにそのディレクトリ内でデータを処理するようにするのが便利です。(SU には組み込みの “ヒストリー” 機構はありません)

3. DMO 処理には補助の NMO 処理のために速度–時間のピックが必要です。これらのピックは NMO と逆 NMO とで同じでなければならないので、編集ミスを避けるためパラメータにするのはよい考えです。ここでも、SU パラメータ・ベクトルのフォーマットに注意して下さい: コンマで区切られた空白のない文字列です。もし、`vnmo` と `tnmo` のベクトルの長さが異なる場合、NMO プログラム (**sunmo**) はエラー・メッセージを表示して終了します。
4. **susort** では **2 次的な**ソート・キーが使用できることに注意して下さい。元々 “正しい” 順序であった 2 次フィールドがソート後もその順序であると仮定しないで下さい—もし、2 次フィールドの順序を考慮するのであれば (このシェル・プログラムのように) 指定して下さい。この行では、まず、オフセットが増加するようにソートし、次に、それぞれのオフセットで CDP 番号が増加するようにソートします。
5. 前進 NMO ステップ
6. DMO ステップ
7. 逆 NMO ステップ
8. CDP に関してソートを戻し、それぞれの CDP でオフセットが増加するようにします。

このシェル・プログラムを完全に理解したいのであれば、次のステップは、含まれているプログラムの self-doc を勉強することです:

```
% sunmo

SUNMO - NMO for an arbitrary velocity function of time and CDP

sunmo <stdin >stdout [optional parameters]

Optional Parameters:
vnmo=2000         NMO velocities corresponding to times in tnmo
tnmo=0            NMO times corresponding to velocities in vnmo

...
```

関連するシェル・プログラムは **su/examples/Nmostack** と **su/examples/Mig** です。

9.3　シェル・プログラミングによる SU の拡張

シェル・プログラミングによって、C のコードを書くことなく SU の能力を大幅に拡張することができます。例えば、**su/examples** 内の **CvStack**、**FilterTest**、**FirstBreak**、**Velan** を見て下さい。

UNIX シェルが高レベルのプログラミング言語ではないことは悲しい事実です—そのため、役に立つシェルのコーディングには複雑な仕掛けがよく使われています。この節では、UNIX シェル・プログラミングによく使われる書法の役立つテンプレートをいくつか示すことにします。

CvStack を例に取って説明します。このシェルの中心は**速度パネル**で、速度と CDP に関する二重ループを作ります—このような概念は単一の SU プログラムには含まれていません。

注目: **CvStack** のようなシェルをスクラッチから書き起こすのはとても時間を消費します。開発時間を節約するために、私達は新しいシェルを作る時には、たとえその細部の意味を覚えていなくても、既存のシェルを引用します。皆さんにも同じやり方をお勧めします！

すでに 2 つのシェル・コードのサンプルで説明した行はコメントしません (第 9.2.1 節と第 9.2.3 節を見て下さい)。**CvStack** で用いられている新しい特徴のみ説明します。

```
#! /bin/sh
# Constant-velocity stack of a range of cmp gathers
# Authors: Jack, Ken
# NOTE: Comment lines preceding user input start with  #!#
set -x

#!# Set input/output file names and data parameters
input=cdp601to610
stackdata=cvstack
cdpmin=601 cdpmax=610
fold=30
space=1         # 1 null trace between panels

#!# Determine velocity sampling.
vmin=1500   vmax=3000   dv=150

### Determine ns and dt from data (for sunull)
nt=‘sugethw ns <$input | sed 1q | sed ’s/.*ns=//’‘                [1]
dt=‘sugethw dt <$input | sed 1q | sed ’s/.*dt=//’‘

### Convert dt to seconds from header value in microseconds
dt=‘bc -l <<END                                                   [2]
        scale=4
        $dt / 1000000
END‘


### Do the velocity analyses.
>$stackdata  # zero output file                                   [3]
v=$vmin
while [ $v -le $vmax ]                                            [4]
do
        cdp=$cdpmin
        while [ $cdp -le $cdpmax ]                                [5]
        do
                suwind <$input \                                  [6]
                        key=cdp min=$cdp max=$cdp count=$fold |
                sunmo cdp=$cdp vnmo=$v tnmo=0.0 |
                sustack >>$stackdata
                cdp=‘bc -l <<END                                  [7]
                        $cdp + 1
END‘
        done
        sunull ntr=$space nt=$nt dt=$dt >>$stackdata              [8]
        v=‘bc -l <<END
                $v + $dv
END‘
done
```

```
### Plot the common velocity stacked data
ncdp=‘bc -l <<END
        $cdpmax-$cdpmin+1
END‘
f2=$vmin
d2=‘bc -l <<END
        $dv/($ncdp + $space)                                    [9]
END‘

sugain <$stackdata tpow=2.0 |

suximage perc=99 f2=$f2 d2=$d2 \
        title="File: $input  Constant-Velocity Stack " \
        label1="Time (s)"  label2="Velocity (m/s)" \&

exit                                                            [10]
```

番号のついた行の説明:

1. この手の込んだコマンドの組み合わせによって、データ・セットの最初のトレース・ヘッダから情報を取得します。**sugethw** プログラムは、連続したトレース中の特定のキーの値を表示します。例えば、

   ```
   % suplane | sugethw tracl ns
    tracl=1             ns=64

    tracl=2             ns=64

    tracl=3             ns=64

    tracl=4             ns=64

    tracl=5             ns=64

    tracl=6             ns=64

    ...
   ```

   sugethw はデータ・セット中のすべてのトレースについて値を返そうとしますが、その内の一つしか必要ではありません。その解決方法として UNIX のストリーム・エディタ (**sed**) を使います。実際、2 回使っています。デフォルトでは **sed** は入力を出力に渡します。最初のコマンドは、単に **sed** に最初の行をパイプに渡したら終了するように指示しています。2 番目の **sed** を通過する際に、整数の前にある不要な文字を削除します。詳しく言うと、2 番目の **sed** コマンドは次のように読みます: `ns=`までのすべての文字を無 (nothing) に置換 (または代入) しなさい。すなわち、それらの文字を削除しなさい。

2. この仕掛けは自慢です。Bourne シェルでは浮動小数点演算はできません。そこで、必要なときには、UNIX の組み込みの計算コマンドである **bc** と “ヒア・ドキュメント” 機能を使います。ここでは、よく必要となるサンプリング間隔の変換をします。サンプリング間隔は SEG-Y ヘッダではマイクロ秒で与えられますが、SU コードでは秒を使います。計算全体を囲む**バック**クオートに十分注意して下さい—この計算の結果を等号の左側のシェル変数、ここでは `dt` にセットしています。計算は数行に渡っても構

いません。最初に小数部の桁数を `scale=4` で設定し、次に、秒への変換を行います。ヒア・ドキュメントのリダイレクト記号`<<`に続く `END` という文字は任意です。シェルはその同じ文字を含んだ行をもう一度読み込むまで、シェル・ファイル中のテキストを入力に取り込みます。**bc** についてもっと知りたければ:

```
% man bc
```

として下さい。

3. コメントに示したように、これは出力リダイレクション記号の特別な用法です。これには、同じ名前の既存のファイルを破壊するか、その名前の新しいファイルをオープンするという効果があります。実際、`>`は常に最初の動作としてこれを行います—危険なオペレータですよ！ ファイルに**追加**しようと思うなら、前にも述べたように、`>>`を使って下さい。
4. これは速度に関する外側のループです。空白に関するもう一つの注意—括弧記号 [] の前後には空白が必要です。

   注意: 括弧による表記方法は古い `test` による表記方法:

   ```
   while test $v -le $vmax
   ```

   のよい代案です。括弧による表記方法は代表的な **sh** マニュアルには書かれていないので、それを使うのは少し不安でした。しかし、ご存知のように、すべての現在の **sh** コマンドがサポートしています—そうでないものを見つけたら知らせて下さい。

   警告! いいですか。`test` という UNIX コマンドがあるということを知ったわけです。ですから、“test” という名前をシェル (または C) プログラムに使ってはいけません—`$PATH` の設定によっては不可解に思える出力に直面することになります。
5. cdp に関する内側のループです。
6. 思い出して下さい: 接続行記号の後に空白やタブをおいてはいけません!
7. 最後の `END` を左マージンに寄せて置いて、きれいなインデント構造を崩していることに注目して下さい。これは、**sh** マニュアル・ページに、終了行には `END` (あるいはあなたの使った何か) しか置いてはいけないと書いてあるためです。実際のところ、大部分のバージョンはインデントをサポートしています。しかし、シェルを移植することを考えると、その危険よりもコードの美しさのほうに価値があるとは考えませんでした。また、Bourne シェルには整数演算が組み込まれているのに整数演算に **bc** を使っていることにも注意して下さい—なぜ 2 つの複雑な慣例を学ぶのか、どのような時にそうするのか、もし興味があれば `man expr` を見て下さい。
8. **sunull** は私 (Jack) が書いたすべての要素が 0 であるトレースを作るプログラムで、`CvStack` で作成されたソートの表示を強調しています。実際、このプログラムを何度も書きましたが、これは目的があって書いた最初のものでした (そう、ユーモアのつもりでした)。
9. トレース軸の速度のラベルを作るための複雑な計算です。すばらしい！どんな意味だったかな？ (前の項目をご覧あれ。)
10. 将来のために何か “予備部品” を保存しておきたいようなときに、`exit` 文は便利です。それらを `exit` 文の後に置けば実行されません。

図 9.2 に `CvStack` によって作成した出力を示します。

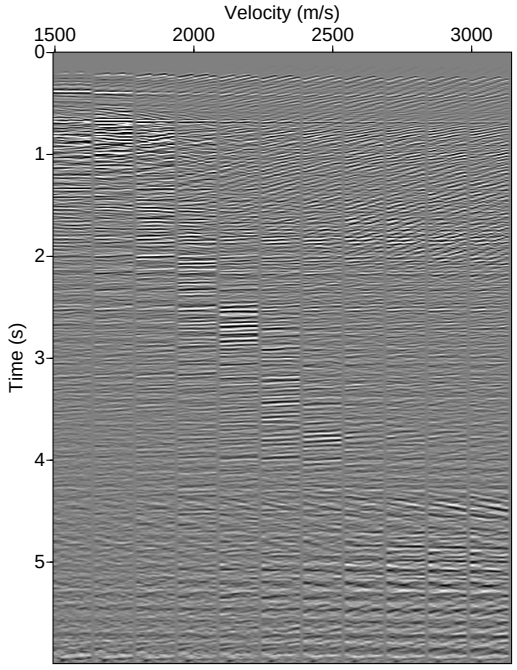

図 9.2: `CvStack` シェル・プログラムの出力

第10章　よくある質問に対する回答

この章では、新しいSUユーザーがよくする質問について触れます。いくつかの回答ではCWPROOTディレクトリを参照しています。これは、CWP/SUのソース・コード、インクルード・ファイル、ライブラリおよび実行形式ファイルを含むディレクトリを指す言葉として使います。SUのインストール手順中はこのディレクトリ名を必ず指定して下さい。

10.1　インストールに関する質問

インストール手順に関するすべての情報は、配布中にあるREADMEファイルにあります。ここでは、よく見られるインストールの問題についてのみ説明します。

質問 1　GCCコンパイラを使用していますが、`fgetpos`および`fsetpos`ルーチンがないというエラー・メッセージが出ます。どうしたらよいのでしょう？

回答 1　この問題は古いSUN OS 4.xx (SOLARIS以前) でよく見られます。これらのSUNのシステムでは`fgetpos`および`fsetpos`ルーチンが定義されていません。これらの2つのルーチンは現在SUパッケージでは使用していないので、この問題を避けるためにユーザーがコンパイル時のフラグを定義することができるようにインストール処理を修正しました。“Makefile.config” の以下のような`OPTC`行のコメントを外して：

```
# For SUN installing with GCC compiler but without GLIBC libraries
#OPTC = -O -DSUN_A -DSUN
```

“make remake” して下さい。

質問 2　GCCコンパイラを使用していますが、`strtoul`あるいは`strerror`ルーチンがないというエラー・メッセージが出ます。どうしたらよいのでしょう？

回答 2　これもまた古いSUN OSでよく見られるものです。修正方法は前の質問と同じです。

質問 3　なぜANSI Cルーチンに関してサブルーチンがないというメッセージが出るのでしょうか？ GCCコンパイラはANSIコンパイラのはずですが？

回答 3　GCCコンパイラはただのコンパイラです。コンパイラはマシンにあるライブラリを参照します。GNUライブラリ (“glibc” パッケージです) がインストールされていなければ、GCCコンパイラは走らせているマシン本来のライブラリを使用しようとします。上にあげてある4つのルーチンはSUN OS 4では使用できません。GCCにはそれがわかりません。しかし、GNUライブラリをインストールすれば、GCCコンパイラは完全なANSI Cコンパイラとして動作するでしょう。

質問 4　なぜ ANSI C ルーチンに関してサブルーチンがないというメッセージが出るのでしょうか？ コンパイラが bzero と bcopy を見つけられないのでコードをコンパイルできません。どうしたら直るでしょうか？

回答 4　われわれは ANSI 標準を守ろうと努めているので、この問題はあってはならないことです。しかし、ときどき古いスタイルのファンクション・コールが忍び込みます。これらを根絶するのは苛立たしいことです。なぜなら多くのシステムは未だに古いスタイルのコールをサポートしているからです。

コンパイラが bzero と bcopy を見つけられないためにコンパイル上の問題があるようでしたら、以下の置き換えをして下さい。

以下の形式のすべての文：

```
bzero( a, b);
```

を以下の形式の文に置き換えて下さい。

```
memset( (void *) a, (int) '\0', b );
```

以下の形式のすべての例文：

```
bcopy( a, b, c);
```

を以下の形式の文に置き換えて下さい。

```
memset( (void *) b, (const void *) a, c );
```

10.2　データ・フォーマットに関する質問

この節では、さまざまなフォーマットのデータを SU フォーマットに変換する際の質問に触れます。

質問 5　SU プログラムが要求するデータ・フォーマットはどのようなものですか？

回答 5　SU データは SEG-Y フォーマットに基づいています (しかしまったく同じではありません)。SU フォーマットはそれぞれにヘッダを持つデータ・トレースで構成されています。SU トレースヘッダは SEG-Y トレースヘッダと同一です。ヘッダおよびトレース・データはマシンの本来のバイナリ・フォーマットです。SEG-Y データを SU データに変換するには **segyread** を使う必要があります。**注意:** SEG-Y トレース・ヘッダのオプション・フィールドは異なるサイトで異なる目的に使われています。SU 自身もこれらのフィールドのある部分を使用しています。ですから、**segyclean** を用いる必要があるでしょう—質問 7 への回答を見て下さい。SU フォーマットには SEG-Y フォーマットの一部であるバイナリと EBCDIC テープ・ヘッダはありません。

パッケージをインストールしたら、SEG-Y/SU ヘッダに関するもっと多くの情報を以下のようにタイプすることで得られます。

```
% sukeyword -o
```

これは SU トレース・ヘッダを定義しているインクルード・ファイル segy.h を表示します。

質問 6　SU パッケージを用いて処理を行うために、われわれがモデリングで作成したデータに必要な SEG-Y インフォメーションを追加する方法はありますか？

回答 6 詳細はデータがどのようにファイルに書かれたかによります。

1. C プログラムによって作成したバイナリ浮動小数点形式の 'datafile' があるならば、**suaddhead** を用いてデータに SU (SEG-Y) トレース・ヘッダを付加します。例えば:

   ```
   % suaddhead < datafile ns=NS > data.su
   ```

   ここで `NS` はデータ中のトレースごとのサンプル数 (整数) です。

2. Fortran 形式の浮動小数点データであれば、以下を使うことができます。

   ```
   % suaddhead < datafile ftn=1 ns=NS > data.su
   ```

   質問 10 も見て下さい。

3. ASCII データであれば、以下を使って下さい。

   ```
   % a2b n1=N1 < data.ascii | suaddhead ns=NS > data.su
   ```

   ここで `N1` はファイル "data.ascii" 内の行ごとの浮動小数点の数です。

4. その他のデータ形式であれば、以下が使えるかもしれません。

   ```
   % recast < data.ascii in=IN out=float | suaddhead ns=NS > data.su
   ```

   ここで `IN` は型 (int, double, char など) です。

より詳しい情報は **suaddhead**, **a2b**, **recast** の selfdoc を見て下さい。

質問 7 segyread を用いて SEG-Y データを読み込みました。すべてうまくいっているように見えたのですが、suximage を用いてデータをプロットしたところウィンドウが真っ黒になってしまいました。何を間違ったのでしょうか？

回答 7 SEG-Y テープを読んだ場合には、オプションの SEG-Y トレース・ヘッダ・フィールドをゼロにするためにデータを **segyclean** にパイプで渡す必要があります。SU プログラムがオプション・フィールドのある部分にゼロでない値を見つけた場合、プログラムはそれらの値をプロット・パラメータとして用いてデータを "非地震データ" として表示しようとします。

もう一つの可能性は 2、3 のデータの値が非常に大きいため、グラフィックスの 256 色グレー・スケール・レベルを覆い尽くしていることです。この問題を解決する方法はグラフィックス・プログラムで "perc=99" と設定することです。例えば :

```
% suximage < sudata perc=99 &
```

とすると、データ全体の上位 1 パーセントの大きさのデータ値をクリップします。

質問 8 pswigb (または pswigp, xwigb,...) プログラムでデータをプロットしようとしています。データは `n1=NSAMP` および `n2=NTRACES` であることがわかっていますが、プロットするときに見るからに正しくプロットするためには `N1=NSAMP+60` と設定しなければならないことがわかりました。なぜですか？

回答 8 間違ったツールを使ってプロットしようとしているようですね。**pswigb, pswigp, pscontour, pscube, psmovie, xwigb, xgraph, xmovie** プログラムの入力データ・フォー

マットは単に浮動小数点の数からなっていなければなりません。もし、SU データ (SEG-Y) トレースであれば、各々のトレースの先頭に追加のヘッダがあり、それはたいていのコンピュータ・アーキテクチャでは 60 個の浮動小数点を保存する場所としてのバイト数 (240 バイト) と等しくなります。

これらのデータをプロットするにはそれぞれ **supswigb, supswigp, supscontour, supscube, supsmovie, suxwigb, suxgraph, suxmovie** を使用して下さい。

また、これらの後者のプログラムではデータの大きさを指定する必要はありません。SU バージョンのコードは適切なヘッダ値から必要な情報を決定します。(実際、それがそのプログラムの動作のすべてです—実際のグラフィックスは su という先頭子のないバージョンによって扱われます。)

質問 9　ファイルに正しい数の値が入っていることを調べるために、ファイルのサイズをチェックしたいのですが、ヘッダをどう計算に入れたらよいかわかりません。どうすればよいですか？

回答 9　ファイルが単に浮動小数点の数である場合はサイズはバイト単位で浮動小数点のバイト数とサンプル数の積 (`SIZE = 4 * N_SAMP`) に等しくなります。SU データ (SEG-Y トレース) にはヘッダ (トレースごとに 240 バイト) があるので、全体のバイト数は `(240 + 4 N_SAMP ) N_TRACES` で与えられます。このようにして計算したバイト数が UNIX コマンドの `ls -l` が示す値になります。

注意: 上記の計算は通常のアーキテクチャを使用し、'segy.h' 内のヘッダの定義を変更していないと仮定しています。64 ビット・ワードのマシンが一般的になってきているので注意！

質問 10　Fortran 形式のデータがあり、以下のようにして SU データに変換しようとしました。

```
% suaddhead  < data.fortran ns=N_SAMP ftn=1 > data.su
```

しかし、正しく動きませんでした。Fortran データは未フォーマット (unformatted) バイナリ・浮動小数点です。どうしたらよいのですか？

回答 10　Fortran データに関して "unformatted" という語の解釈に異なる方法があります。以下を試してみて下さい。

```
% ftnstrip < data.fortran | suaddhead ns=N_SAMP > data.su
```

suaddhead の 'ftn=1' オプションが失敗する場合でも、プログラム **ftnstrip** は Fortran データの C 形式のバイナリ・データへの変換に成功することがよくあります。(注: プログラム **ftnunstrip** は C 形式のバイナリ・データを Fortran 形式に変換することができます。)

質問 11　CWP/SU パッケージのインストールに成功しました。しかし、デモ・スクリプトを走らせようとするとシェル・スクリプトが見つからないという多くのエラー・メッセージが出ます。どう修正したらよいですか？

回答 11　シェルの `PATH` に `CWPROOT/bin` (ここで `CWPROOT` は CWP/SU のソース・コード、インクルード・ファイル、ライブラリ、実行形式ファイルを含む`/your/root/path` で表される絶対パスです。) を追加する必要があります。これは `csh` あるいは `tcsh` では`.cshrc` で設定します。Bourne シェル (`sh`)、Bourne Again シェル (`bash`) あるいは Korn シェル (`ksh`) では `PATH` 変数は`.profile` ファイルにあります。また、C シェル`/bin/csh` あるいは TC シェ

ル/bin/tcshを作業シェル環境として走らせていて、コードをコンパイルして以来再ログインしていなければ、

```
% rehash
```

とタイプする必要があります。

質問 12　SU と例えば Promax のような商用パッケージとの間でどのようにデータをやりとりすればよいですか？

回答 12　短い答としてはディスク・ファイル上に SEG-Y テープを作るという方法があります。“data.su” というファイルを SEG-Y ファイルに変換するには以下のようにして下さい。

```
% segyhdrs < data.su
% segywrite tape=data.segy < data.su
```

この時点で Promax を使って “data.segy” を読んで下さい。このファイルは “IBM 実数フォーマットの Promax のディスク上のテープ・ファイル (Promax tape-on-disk file in IBM Real format)” です。これに従って Promax のメニューを選んで下さい。その他の商用パッケージについては、ディスク上の SEG-Y テープを読むための適切なコマンドを用いて下さい。

商用パッケージから SU への変換は逆のステップを取ります。SEG-Y テープ・イメージのファイルを作成し、

```
% segyread tape=data.segy | segyclean >data.su
```

として下さい。

質問 13　SU データのトレース・ヘッダを取り去り、生のトレースに何らかの処理を施したのち、ヘッダ情報を失わずにヘッダを戻したいと思います。どうしたらよいでしょうか？

回答 13　以下のようにして下さい。

```
% sustrip < data.su head=headers >data.binary
```

(‘data.binary’ から ‘data1.binary’ を作成する望みの処理何でもここで行って下さい)

```
% supaste < data1.binary head=headers >data1.su
```

質問 14　IBM RS6000 でデータを作り、Linux ベースの PC システムに転送しました。データは RS6000 上では OK のようでしたが、PC 上でそれを使って作業しようとすると SU プログラムが全然動作しません。何が悪いのでしょうか？

回答 14　ここで遭遇した問題はそれぞれ “ビッグ・エンディアン (big-endian)” と “リトル・エンディアン (little-endian)”、または “上位バイト (high-byte)” と “下位バイト (low-byte)” と呼ばれる 2 つの IEEE バイナリ・フォーマットがあるという問題です。これらの用語はデータを表すバイトの順序を表しています。IEEE フォーマットは SGI, SUN, IBM RS6000、Silicin Graphics, NeXT (Black Hardware), SUN, HP, Power PC および Motorola チップをベースのプラットフォームは ‘ビッグ・エンディアン’ で、一方、Intel ベースの PC と Dec と Dec の Alpha 製品は ‘リトル・エンディアン’ のプラットフォームです。

CWP/SU パッケージにはデータ転送のためにバイト・スワップを行う 2 つのプログラムが用意されています。**swapbytes** と **suswapbytes** です。

プログラム **swapbytes** はユーザーがさまざまなデータ型 (float, double, short, unsigned short, long, unsigned long, int) のバイナリ・データのバイトのスワップを行えるものです。

SU フォーマットのデータにはプログラム **suswapbytes** が提供されています。

さらに、プログラム **segyread** と **segywrite** には “swap=” フラグがあり、ユーザーが作業しているプラットフォームが “ビッグ・エンディアン (big-endian)” であるか “リトル・エンディアン (little-endian)” であるかを指定できます。

SU の古いリリースでは、ウィグル・トレース表示ルーチン内にビット順操作に問題がありました。しかし、これらの問題は “Makefile.config” 中に現れる “ENDIANFLAG” によって解決されています。

質問 15　SEG-2 フォーマットのデータをどのようにして SEG-Y に変換したらよいでしょうか？

回答 15　この目的のために、$CWPROOT/src/Third_Partyseg2segy に 2 つのプログラムがあります。フランスの University of Pau によって提供されているものです。これらは SU がインストールされているシステムには簡単にインストールすることができます。

いったん ‘data.seg2’ を ‘data.segy’ に変換してしまえば、以下のようにして SU フォーマットに読み込むことができます。

```
% segyread tape=data.segy > data.su
```

10.3　テープの読み書き

この節には、SU を用いた SEG-Y テープの読み書きに関するよくある質問が含まれています。テープの読み/書きは科学というより技術 (more than an art than a science, 理屈よりもコツ?) です。ここにちょっとしたヒントを示します。

1. テープ・ドライブが可変ブロック長に設定されていることを確認して下さい。もし、IBM RS6000 を使っているのなら、テープ・デバイスで `blocksize=0` に設定するために `smit` を使う必要があります。テープ・ドライブがデフォルトの固定ブロックサイズ (例えば blocksize=1024 または 512) に設定されていた場合は、SEG-Y テープの読み込みは失敗します。
2. 複数のテープにわたるファイルを読むには、非巻き戻しデバイスを使って下さい。RS6000 では、`/dev/rmtx.1` 【訳注：`/dev/nrmtx.1` の誤り？】のようになります。詳細は `man mt` を見て下さい。
3. もし、これもうまくいかなかったら、以下を試してみて下さい:

   ```
   % dd if=/dev/rmtx of=temps bs=32767 conv=noerror
   ```

   ここで、`/dev/rmtx` (これは正しいデバイス名ではなく、システムによって変わります) は正規の (巻戻し) テープ・デバイスです。オプション中で、IBM/RS6000 用の正しいブロックサイズ `bs=32767` $2^{16}+1$ を与えています。*SUN* の場合には `bs=32765` $2^{16}-1$ を試して下さい。【訳注： $2^{16}+1=65537$, $2^{16}-1=65535$ です？】これにより、テープの全内容が一つのファイルにダンプされます。

質問 16 どのようにして複数の SEG-Y ファイルをテープに書き込むことができますか？

回答 16 以下は複数のファイルをテープに書き込むシェル・スクリプトです。

```
#! /bin/sh

DEV=/dev/nrxt0  # non rewinding tape device

mt -f $DEV rewind

j=0
jmax=40

while test "$j" -ne "$jmax"
do
        j=‘expr $j + 1‘
        echo "writing tape file $j"
        segywrite tape=$DEV bfile=b.$j hfile=h.$j verbose=1 buff=0 < ozdata.$j
done

exit 0
```

10.4 ジオメトリの設定

質問 17 SU では “ジオメトリの設定” はどうするのでしょうか？

回答 17 商用パッケージではしばしば “ジオメトリの設定 (geometry setting)” と呼ばれる地震探査データ操作に共通する作業があります。そこではユーザーは調査観測者の記録に記載された情報をテープ・ヘッダの値に変換します。

CWP/SU パッケージにはこのための機能が備わっています。ヘッダ・フィールドの値を取得したり設定することができ、また、1 つもしくは 2 つの他のヘッダ・フィールドから第 3 のヘッダ・フィールドを計算できます。このために使用する必要があるプログラムは:

sugethw (“SU get header word (SU ヘッダ・ワードの取得)”)
sushw (“SU set header word (SU ヘッダ・ワードの設定)”)
suchw (“SU change or compute header word (SU ヘッダ・ワードの変更または計算)”)

そのコードの自己文書 (self-doc) を見るには各々のプログラムの名前をタイプして下さい。

さらに、これらのプログラム中で示されるヘッダ・ワードの “キーワード” が何かを調べるには、

```
% sukeyword -o
```

とタイプして下さい。

さまざまな形式の情報があることでしょう。そのような形式のうち最も普遍的で複雑でない仮定をここでします。

作業には以下の基本的なステップが必要です。

1. データを SU フォーマットに変換しましょう。SU フォーマットは SEG-Y とまったく同じではありませんが、SEG-Y ヘッダ情報を保存します。SEG-Y データからスタートするのであれば (テープからでもディスク上のファイルの形式でも)、データを SU ファイル・フォーマットに読み込むには **segyread** を使用します。テープの場合:

   ```
   % segyread tape=/dev/rmt0 bfile=data.1 hfile=h.1 | segyclean > data.su
   ```

   ディスク上のファイルの場合:

   ```
   % segyread tape=data.segy bfile=data.1 hfile=h.1 | segyclean > data.su
   ```

   とします。“data.segy” というファイルは SEG-Y データの “テープ・イメージ” であると仮定しています。商用のソフトウェアの中には SEG-Y フォーマットのレイアウトをまねて SEG-Y 風のデータを書き出しますが、フォーマットが SEG-Y で定義されている真の IBM テープ・フォーマットでないものがありますので気を付けて下さい。Promax では IBM 実数フォーマット (IBM Real format) で SEG-Y ファイルを書き出せば、真の SEG-Y テープ・イメージになります。続けましょう。

2. SU フォーマットのデータがあれば、次のようにして SEG-Y ヘッダの範囲を表示させることができます (設定されていないヘッダは表示されません)。

   ```
   % surange < data.su
   ```

3. データにはすでにいくつかのヘッダが設定されているのが普通です。ジオメトリ設定に便利なフォーマットでこれらのフィールドをダンプするには **sugethw** を以下のように使います。

   ```
   % sugethw < data.su output=geom key=key1,key2,... > hfile.ascii
   ```

   “key=key1,key2,...” という文字列は希望する SEG-Y トレース・ヘッダ・フィールドを表すキーワードです。これらのキーワードは以下のようにして表示できます。

   ```
   % sukeyword -o
   ```

4. いったん希望するヘッダ・フィールドを “hfile.ascii” にダンプしてしまえば、好みのエディタで編集することができます。ポイントは、特定のヘッダ・フィールドの値を (トレースごとに “data.su” に現れる順に) 列挙した複数列のアスキー・ファイルを作るのに好きなどんな方法を使ってもよいということです。各列は設定しようとする特定のヘッダ・フィールドの値を含みます。

5. さて、ヘッダ値を含んだアスキー・ファイルを作成したので、以下のようにしてこれらの値を “data.su” にロードすることができます。

   ```
   % a2b < hfile.ascii n1=N_columns > hfile.bin
   ```

   ここで “N_columns” は “hfile.ascii” 中の列数です。“hfile.ascii” をバイナリに変換しています。

   さて、以下のようにして下さい。

   ```
   % sushw < data.su key=key1,key2,... infile=hfile.bin > data1.su
   ```

ここで “key1,key2,...” は設定されるフィールドを表す適切なキーワードで、“hfile.ascii” で列ごとに値が現れる順序に列挙されていなければなりません。

6. 2 つの与えられたヘッダ・フィールドから第 3 のヘッダ・フィールドを計算したい場合には **sushw** を使うことができます。また、設定したいヘッダ・フィールドがある種規則的な場合 (各トレースで一定であるとかギャザーに関して線形に変化するとか) は “infile=” オプションを用いる必要はありません。単に必要な値を **sushw** に与えることができます。これらの例は **sushw** と **suchw** の self-doc を見て下さい。

10.5 技術的な質問

質問 18 データを再サンプリングしてトレース数を半分に、サンプル数を半分にしたいのですが、どうしたらよいでしょうか？

回答 18 データを再サンプリングするためには、以下のようにしなければなりません。

1. エリアジング生じないかチェックします。このためにはデータの振幅スペクトルを表示します。**suspecfx** を用いて以下のようにして下さい。

```
% suspecfx < data.su | suxwigb &
```

2. データのバンド幅がデータの新しいナイキスト周波数 (この例の場合は元々のナイキスト周波数の半分) を越えていた場合、新しいナイキスト・バンド幅内に納まるようにデータにフィルタをかけなければいけません。

```
% sufilter < data.su f=f1,f2,f3,f4 amps=0,1,1,0 > data.filtered.su
```

ここで “f1,f2,f3,f4” はフィルターのコーナー周波数で、“amps=0,1,1,0” はフィルタがバンドパス・フィルタであることを示します。

3. これで **suresamp** を用いてデータを再サンプリングすることができます。

```
% suresamp < data.filtered.su nt=NTOUT dt=DTOUT > data.resampled.su
```

この場合、“NTOUT” は元々のサンプル数の 1/2 であり、“DTOUT” は元々のデータの時間サンプリング間隔 (秒単位) の 2 倍です。出力データはバンド幅が変化した以外は入力データと非常によく似ています。
【訳注: トレースの操作の説明がなくて終わっている？】

10.6 一般

この節では、SU パッケージについての一般的質問について触れます。

質問 19 色々なところで目にする gelev, selev, fldr などのおかしな単語は何ですか？

回答 19 これらは多くのコードで必要な “キーワード” です。SU (SEG-Y) ヘッダ・フィールドに属するものです。

全体のリストを見るには:

```
% sukeyword -o
```

個別のキーワードを見るには:

```
% sukeyword keyword
```

とタイプして下さい。

質問 20　“リトル・エンディアン (little endian)” と “ビッグ・エンディアン (big endian)” という用語の意味は？

回答 20　それぞれ “リトル・エンディアン (little endian)” と “ビッグ・エンディアン (big endian)” と呼ばれる 2 種類の IEEE バイナリ・フォーマットがあります。それぞれ “上位バイト (high byte)” と “下位バイト (low byte)” とも呼ばれています。これらはバイナリ・データのビット表現のバイト順の問題です。以下のプラットフォームは “リトル・エンディアン” です: Dec と Intel ベースの PC。他の普通のプラットフォームは “ビッグ・エンディアン” です: IBM RS6000, Silicon Graphics, NeXT (Black Hardware) SUN, HP, PowerPC および Motorola チップをベースとしたプラットフォーム。

質問 21　なぜ CWP/SU のリリースはなじみのある小数 (1.1, 1.3 など) ではなくて整数 (22, 23, 24 など) で与えられているのですか？

回答 21　CWP/SU のリリース番号は SU NEWS 電子メール・メッセージに対応するようにつけられています。パッケージに含まれるそれぞれのコードには (RCS が割り当てた) 伝統的な小数点のリリース番号がつけられていますが、それらはすべて異なります。パッケージはリリースが増加する方向に変化しますが、互いに揃っては変化しません。ですから、標準的な表示は適切でないように思えます。しかし、ユーザーは 24 を 2.4 と見ても構いません。将来この変換を適用するかもしれません。

注目: 以前は RCS を用いてすべてのコードを同時に 2.1, 3.1 というように更新していました。いつのまにか自然に止めてしまいました。

質問 22　コードの更新はどのくらいの間隔で行われていますか？

回答 22　CWP/SU パッケージは 3–6 カ月の間隔で更新されています。既知のユーザーにはこれらのリリースをメールで知らせています。古くなったバージョンのサポートはしませんので、常に現在のものを使うことを勧めます。

質問 23　CWP/SU プログラムのための複雑な一群の入力パラメータがあります。端末ウィンドウのコマンドラインからコマンドを実行したいのですが、入力パラメータの文字列全部を何度も打ち直したくありません。どうしたらよいですか？

回答 23　CWP/SU プログラムはコマンドラインから入力パラメータを取得しますが、“パラメータ・ファイル” から読み込む機能ももっています。これはパラメータ `par=parfile` を設定することによって行います。ここで `parfile` は希望のコマンドライン文字列を含んだファイルです。

例えば:

```
% suplane ntr=20 nt=40 dt=.001 | ...
```

は以下のコマンドとまったく等価です。

```
% suplane par=parfile | ...
```

ここで文字列

```
ntr=20 nt=40 dt=.001
```

が “parfile” に含まれています。

質問 24 “ints8r” 関数の sudoc のエントリが見つからないのですが、SU マニュアルにはすべてのライブラリ関数にはオンライン文書があると書いてあります。何が間違っているのでしょうか？

回答 24 (“ints8r” のような) ライブラリ関数の正しい検索手順は以下の通りです。

```
% sufind ints8r
```

とすると以下の結果が得られます。

```
INTSINC8 - Functions to interpolate uniformly-sampled data via 8-coeff. sinc
        approximations:

ints8c  interpolation of a uniformly-sampled complex function y(x) via an

For more information type: "sudoc program_name <CR>"
```

“INTSINC8” という名前は “ins8c”【訳注: ints8r？】ライブラリ関数を含むファイルの名前です。より詳しい情報を得るために **sudoc** を使うことができます。

```
% sudoc intsinc8
```

以下の結果が得られます。

```
In /usr/local/cwp/src/cwp/lib:
INTSINC8 - Functions to interpolate uniformly-sampled data via 8-coeff. sinc
            approximations:

ints8c  interpolation of a uniformly-sampled complex function y(x) via an
         8-coefficient sinc approximation.
ints8r  Interpolation of a uniformly-sampled real function y(x) via a
         table of 8-coefficient sinc approximations

Function Prototypes:
void ints8c (int nxin, float dxin, float fxin, complex yin[],
        complex yinl, complex yinr, int nxout, float xout[], complex yout[]);
void ints8r (int nxin, float dxin, float fxin, float yin[],
        float yinl, float yinr, int nxout, float xout[], float yout[]);

Input:
nxin        number of x values at which y(x) is input
dxin        x sampling interval for input y(x)
fxin        x value of first sample input
yin         array[nxin] of input y(x) values:  yin[0] = y(fxin), etc.
yinl        value used to extrapolate yin values to left of yin[0]
yinr        value used to extrapolate yin values to right of yin[nxin-1]
nxout       number of x values a which y(x) is output
xout        array[nxout] of x values at which y(x) is output

Output:
```

```
yout        array[nxout] of output y(x):  yout[0] = y(xout[0]), etc.

Notes:
Because extrapolation of the input function y(x) is defined by the
left and right values yinl and yinr, the xout values are not restricted
to lie within the range of sample locations defined by nxin, dxin, and
fxin.

The maximum error for frequiencies less than 0.6 nyquist is less than
one percent.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

質問 25　自分で SU プログラムを書きました。それらが suname と sudoc の一覧に現れるようにしたいのですが、どうしたらよいでしょうか？

回答 25　**updatedocall** を走らせて下さい (ソース・コードは CWPROOT/par/shell にあります)。このコードを新しいパスの下に置いたのであれば、このパスを **updatedoc** スクリプト内のパスのリストに加えておかなければいけません。**updatedoc** スクリプトが selfdoc 情報を捕らえることができるように、プログラムのソース・コードの self-doc と追加の情報の部分の先頭と終端に以下のマーカー・ラインを含める必要があります。

```
/*********************** self documentation **********************/
/**************** end self doc ********************************/
```

これらをタイプ入力するのではなく、既存の SU プログラムから複製するようにして下さい。そうすればパターンは正確に **updatedoc** スクリプトが期待するものとなります。

質問 26　psimage で作成したグレー・スケール (カラーでない) PostScript ファイルがあり、カラー PostScript フォーマットに変換したいと思います。しかし、ファイルを作成した元のバイナリ・データは持っていません。どうしたらよいですか？

回答 26　新しいカラー PostScript ファイルを作るためにはバイナリ・ファイルを復元しなければなりません。どうすればよいかを以下に示します (ここで **psimage** または **supsimage** によって作成したビットマップ化グラフィックを仮定します)。

1. PostScript ファイルのバックアップを作ります。
2. PostScript ファイルを編集して、ファイルの大部分を占めている 8 進数バイナリ・イメージ以外の部分をすべて削除します。
3. **h2b** を用いて 8 進数ファイルをバイナリに変換します。
4. ファイルは元の入力ファイルから回転していることに気づくでしょう。データを回転するには **transp** を用います。**transp** で用いる n1 と n2 の値は入力データのディメンジョンで、出力データの逆であることに注意して下さい。
5. 0-255 表現のバイナリ・データが得られました。希望する方法で再プロットできます。

この方法はスキャンしたイメージを SU フォーマットに変換するためにも使うことができます。次のステップは **suaddhead** を用いてデータに SU ヘッダを追加することです。

第11章　SUプログラムの書き方

11.1　Makefileの設定

CWP/SUパッケージは洗練されたMakefile構造を使用しており、あなたが新しいコードを開発するときにも使用することができます。新しいコードを書くプロジェクトは、あなたの作業エリアにローカルなディレクトリを作成することから始めるとよいでしょう。次に、Makefileを$CWPROOT/src/su/mainからそのディレクトリにコピーし、以下の修正を加えます。

```
Change:

D = $L/libcwp.a $L/libpar.a $L/libsu.a

LFLAGS= $(PRELFLAGS) -L$L -lsu -lpar -lcwp -lm $(POSTLFLAGS)

to:

D = $L/libcwp.a $L/libpar.a $L/libsu.a

B = .

OPTC = -g

LFLAGS= $(PRELFLAGS) -L$L -lsu -lpar -lcwp -lm $(POSTLFLAGS)

Change:


PROGS =                 \
        $B/bhedtopar    \
        $B/dt1tosu      \
        $B/segyclean    \
        $B/segyhdrs     \
        $B/segyread     \
        ...
        ...

to:
```

```
PROGS =                 \
        $B/yourprogram
```

ここで、あなたのプログラムは “yourptogram.c” でそのディレクトリに置かれています。

そして、単に “make” とタイプすると “yourprogram” はコンパイルされるでしょう。

テストとして、既存の SU プログラムの一つを$CWPROOT/src/su/main からあなたのローカルな作業ディレクトリにコピーして、Makefile を上に従って修正し、“make” とタイプして試すことができます。

実際のところ、すべての新しい SU プログラムは、似た構造を持つ既存の SU プログラムの複製として始まると見なされるので、これがおそらく、新規コード作成の冒険を始める最善の方法だと思います。

11.2　SU プログラムのテンプレート

バリエーションは通常必要ですが、典型的な SU プログラムは下記のリストのプログラムのような形をしています (このテンプレートを作るために、sumute プログラムから行を抜粋しました)。リスト中の行の最後の [] 括弧はリストの一部ではありません—テンプレートの説明に便利なように付けたものです。効率よく SU コーディングをする秘訣は、書こうと思っているプログラムとよく似た既存のプログラムを探すことです。ちょうどよいコードや “複製” するコードが見あたらないときには私達に尋ねて下さい—これは最も手ごわい仕事に違いありません。

```
/* SUMUTE: $Revision: 1.16 $ ; $Date: 1998/01/15 17:46:03 $      */  [1]

#include "su.h"                                                        [2]
#include "segy.h"

/************************ self documentation **********************/  [3]
char *sdoc[] = {
"                                                                    ",
" SUMUTE - ......                                                    ",
"                                                                    ",
" sumute <stdin >stdout                                              ",
"                                                                    ",
" Required parameters:                                               ",
"         none                                                       ",
"                                                                    ",
" Optional parameters:                                               ",
"        ...                                                         ",
"                                                                    ",
" Trace header fields accessed: ns                                   ",
" Trace header fields modified: none                                 ",
"                                                                    ",
NULL};
/***************** end self doc ***********************************/

/* Credits:
 *
 *        CWP: Jack Cohen, John Stockwell
 */
```

```
segy tr;                                                        [4]

main(int argc, char **argv)
{
        int ns;                 /* number of samples          */      [5]
        ...


        /* Initialize */
        initargs(argc, argv);                                           [6]
        requestdoc(1);                                                  [7]

        /* Get parameters */
        if (!getparint("ntaper", &ntaper))        ntaper = 0;           [8]


        /* Get info from first trace */
        if (!gettr(&tr)) err("can't read first trace");                 [9]
        if (!tr.dt) err("dt header field must be set");                  [10]

        /* Loop over traces */
        do {                                                            [11]
                int nt     = (int) tr.ns;                               [12]

                if (below == 0) {                                       [13]
                        nmute = NINT((t - tmin)/dt);
                        memset((void *) tr.data, (int) '\0', nmute*FSIZE);
                        for (i = 0; i < ntaper; ++i)
                                tr.data[i+nmute] *= taper[i];
                } else {
                        nmute = NINT((nt*dt - t)/dt);
                        memset((void *) (tr.data+nt-nmute),
                                        (int) '\0', nmute*FSIZE);
                        for (i = 0; i < ntaper; ++i)
                                tr.data[nt-nmute-1-i] *= taper[i];
                }
                puttr(&tr);                                             [14]
        } while (gettr(&tr));                                           [15]

        return EXIT_SUCCESS;                                            [16]
}
```

番号をつけた行の説明:

1. 私達はコードの内部バージョンを UNIX のユーティリティである RCS を用いて管理しています。この項目は RCS のための文字列テンプレートを示しています。
2. su.h ファイルには、(直接、あるいは間接に) 私達が定義したすべてのマクロとプロトタイプが含まれています。segy.h ファイルには、トレース・ヘッダ・フィールドの定義が含まれています。
3. 星印の行は "self-doc" 情報の区切りを示すために使われています—自動文書化シェルが使用するので、コード中にそのまま正確に含めて下さい。示されている self-doc のスタイルは典型的なものですが、使用方法の情報が最後についていたりもっと多くのオプションがある場合が多いです。既存のコードを参照して下さい。

4. SU (SEG-Y) トレース・バッファの外部参照宣言です。スタック領域の浪費を避けるために外部参照にしてあります。
5. 大域変数を宣言する際には説明をつけています。変数の命名法が一貫したものになるように、あなたのコードを確認して下さい。(正式な SU ネーミング標準はありません)。
6. `initargs` サブルーチンは SU のコマンドライン渡し機能を設定します (79 ページを見て下さい)。
7. `requestdoc` サブルーチン・コールはユーザーに返される self-doc の環境を指定します。通常のプログラムには引数 '1' を与えます。この場合は SU トレース・ファイルを読み込むのに標準入力 (<) のみを用います。合成記録を作るコード (例えば `suplane`) では '0' を、2 つの入力ファイルを必要とするコードでは '2' を与えて下さい ("などなど" と言うこともできますが、**3 つ**あるいはそれ以上の入力ファイルを必要とする SU プログラムはありません)。
8. パラメータをコマンド・ラインから読む典型的なコードです。"ユーザーが値を指定しなければ、デフォルト値を使う" と解釈して下さい。サブルーチンには型を指定しなければなりません。ここでは**整数型**のパラメータを読み込んでいます。
9. 最初のトレースを読み込み、もし空であれば終了します。`fgettr` サブルーチンは、SU トレース・フォーマットを "知っています"。通常、トレース・ファイルを標準入力から読み込んだ後に `gettr` を使います。これは `su.h` で定義される `fgettr` に基づくマクロです。このコードは最初のトレースがトレース・バッファ (ここでは `tr`) に読み込まれているものとしていることに注意して下さい。ですから、次に `fgettr` を呼ぶ前にこのトレースを処理しなければなりません。
10. 最初のトレース・ヘッダからトレース・パラメータを入手する必要があるので、最初のトレースを読み込みました。通常ここにはサンプル数 (`tr.ns`) やサンプリング間隔 (`tr.dt`) のような項目があり、すべてのトレースで共通です。
11. (典型的なやり方ですが、) メインの処理ループが開始する前に最初のトレースを読み込んだので、ループの最後で新しいトレースを読むように "do-while" を使います。
12. 使えるところではできるだけ**局所**変数を使います。
13. ここからは地震探査のアルゴリズムです—ここでは不完全です。実際の `sumute` のコードを残してあります。これから書いていく新しいコードに役立つ行をたまたま含んでいるからです。実際に作業をさせるためにここでサブルーチンを呼ぶこともできます。
14. `fputtr` と `puttr` はそれぞれ `fgettr` と `gettr` と同じ働きをする出力用サブルーチンです。
15. ループの終了。トレースが出力され尽くして処理が停止したときに `gettr` は 0 を返します。
16. プログラムの正常終了を示すときによく使われる ANSI-C マクロです。

11.3　新しいプログラムを書く: suvlength

あるユーザーから可変長トレースの SU 処理について尋ねられたことがありました。彼の研究所では、データは起震時刻からさまざまな終了時刻まで記録されていました。問題は、SU 処理は SEG-Y 標準に基づいているという点です。したがって、データセット中のすべ

てのトレースは同じ長さでなければなりません。解決方法として、SU をすべて変更するのではなく、トレースの末尾に 0 をつけて可変長データを固定長データに変換するプログラムを提供する方法がよいように思いました—この新しいプログラムを suvlength と呼ぶことにします。ここで、出力トレースの長さをユーザーのパラメータとすることができます。適切であれば、パラメータのデフォルト値を与えるのもよいでしょう。ここでは、メインの処理ループを開始する前に値を確定しないといけないので、最初のトレースの長さを使うのが最もよい選択だと思いました。

ここまではよかったのですが、思わぬ深刻な障害にぶち当たりました: 基本となるトレース取得機能 gettr そのものが固定長トレースを仮定しているのです（あるいは、gettr を固定長トレース標準を守るように設計したと言った方がよいでしょう)。しかし、考えてみると、gettr 自身が**最初の**トレースを使ってその長さを計算するための特別な計測をしなければならないことに気がつくでしょう。ですから、しなければならないのは、**すべての**トレースに対して最初のトレースに関するロジックを適用する、新しいトレース取得ルーチンを作ることです。ここでは、“fvgettr” サブルーチンの書き方についての詳細は省略して、前出のテンプレートを新しい suvlength コードに変換する方法に話を移します:

```
/* SUVLENGTH: $Revision: 1.16 $ ; $Date: 1998/01/15 17:46:03 $   */

#include "su.h"
#include "segy.h"

/************************ self documentation **********************/
char *sdoc[] = {
"                                                                 ",
" SUVLENGTH - Adjust variable length traces to common length     ",
"                                                                 ",
" suvlength <variable_length_traces >fixed_length_traces         ",
"                                                                 ",
" Required parameters:                                           ",
"         none                                                    ",
"                                                                 ",
" Optional parameters:                                           ",
"        ns      output number of samples (default: 1st trace ns)",
NULL};
/***************** end self doc ***********************************/

/* Credits:
 *        CWP: Jack Cohen, John Stockwell
 */

/* prototype */
int fvgettr(FILE *fp, segy *tp);

segy tr;

main(int argc, char **argv)
{
        int ns;         /* number of samples on output traces  */


        /* Initialize */
        initargs(argc, argv);
        requestdoc(1);
```

```
        /* Get parameters */
        ...

        /* Get info from first trace */
        ...

        ...

        return EXIT_SUCCESS;                                          [16]
}

/* fvgettr code goes here */
        ...
```

ここでちょっとした困難に行き当たりました。唯一のパラメータには最初のトレースを読んでから得られるデフォルト値があります。これは、テンプレート中のパラメータ取得とトレース取得を逆にすれば解決できることは明らかです。再開すると:

```
        /* Get info from first trace and set ns */
        if (!fvgettr(stdin, &tr))  err("can't get first trace");
        if (!getparint("ns", &ns))    ns = tr.ns;

        /* Loop over the traces */
        do {
                int nt = tr.ns;
```

実際の地震探査のアルゴリズムのところまで来ました—このケースではわずかなものです: 出力トレースの長さが入力よりも大きければ、入力トレースの末尾に 0 を追加します。これをするには簡単なループを書けばよいのですが、ANSI-C の memset ルーチンを使えば最も効率的にできます。正直に言うと、最近これを使っていないので、いつも使い方を忘れてしまうのです。そのためには、su/main ディレクトリへ cd して、grep を使って他のプログラムでの memset の使われ方を探すという方法があります。これをすると sumute が必要としている使い方に最も近かったので、このコードをコピーすることから始めたと言うわけです。以下は、完成した suvlength のメインプログラムです:

```
/* SUVLENGTH: $Revision: 1.16 $ ; $Date: 1998/01/15 17:46:03 $        */

#include "su.h"
#include "segy.h"

/************************ self documentation **********************/
char *sdoc[] = {
"                                                                   ",
" SUVLENGTH - Adjust variable length traces to common length       ",
"                                                                   ",
" suvlength <vdata >stdout                                          ",
"                                                                   ",
" Required parameters:                                              ",
"         none                                                      ",
"                                                                   ",
" Optional parameters:                                              ",
"          ns     output number of samples (default: 1st trace ns)",
NULL};
/***************** end self doc ***********************************/
```

```
/* Credits:
 *        CWP: Jack Cohen, John Stockwell
 *
 * Trace header fields accessed:  ns
 * Trace header fields modified:  ns
 */

/* prototype */
int fvgettr(FILE *fp, segy *tp);

segy tr;

main(int argc, char **argv)
{
        int ns;                /* samples on output traces        */


        /* Initialize */
        initargs(argc, argv);
        requestdoc(1);


        /* Get info from first trace */
        if (!fvgettr(stdin, &tr))  err("can't get first trace");
        if (!getparint("ns", &ns))    ns = tr.ns;


        /* Loop over the traces */
        do {
                int nt = tr.ns;

                if (nt < ns) /* pad with zeros */
                        memset((void *)(tr.data + nt), '\0', (ns-nt)*FSIZE);
                tr.ns = ns;
                puttr(&tr);
        } while (fvgettr(stdin, &tr));

        return EXIT_SUCCESS;
}


#include "header.h"

/* fvgettr - get a segy trace from a file by file pointer (nt can vary)
 *
 * Returns:
 *        int: number of bytes read on current trace (0 after last trace)
 *
 * Synopsis:
 *        int fvgettr(FILE *fp, segy *tp)
 *
 * Credits:
 *        Cloned from .../su/lib/fgettr.c
 */

int fvgettr(FILE *fp, segy *tp)
```

...

注目: 実際の SU では、`fvgettr` サブルーチンはすでにライブラリ関数になっています。また、標準入力の場合には `vgettr` という便利なマクロがあります。しかし、大部分のアプリケーションではそれはあまり考えなくてよいでしょう。

新しい SU コードには、その使い方を示すサンプルのシェル・プログラムをつけるのがよいでしょう。以下は X-Windows グラフィック用のプログラムです:

```
#! /bin/sh
# Trivial test of suvlength with X Windows graphics

WIDTH=700
HEIGHT=900
WIDTHOFF=50
HEIGHTOFF=20

>tempdata
>vdata
suplane >tempdata  # default is 32 traces with 64 samples per trace
suplane nt=72 >>tempdata
suvlength <tempdata ns=84 |
sushw key=tracl a=1 b=1 >vdata

# Plot the data
suxwigb <vdata \
        perc=99 title="suvlength test"\
        label1="Time (sec)" label2="Traces" \
        wbox=$WIDTH hbox=$HEIGHT xbox=$WIDTHOFF ybox=$HEIGHTOFF &

# Remove #comment sign on next line to test the header
#sugethw <vdata tracl ns | more
```

付 録A　SUの取得とインストール

SU パッケージは地震探査処理プログラムに加え、科学計算ライブラリ、グラフィックス・ルーチン及びSU コーディング規約を支援するルーチンを含んでいます。パッケージは ftp.cwp.mines.edu (138.67.12.4) の anonymous ftp から利用できます。ディレクトリ・パスは pub/cwpcodes です。パッケージはWorld Wide Web の以下のサイト http://www.cwp.mines.edu/cwpcodes からも利用できます。以下のファイルを取得して下さい。

1. README_BEFORE_UNTARRING
2. untar_me_first.xx.tar.Z
3. cwp.su.all.xx.tar.Z

ここで、xx は現在のリリース番号を表します。以前のリリース yy を現在のリリース xx に更新するためのアップデートも利用できます。以下のファイルを取得して下さい。

1. README_BEFORE_UNTARRING
2. README_UPDATE
3. untar_me_first.xx.tar.Z
4. update.yy.to.xx.tar.Z
5. update.list (訳注: 実際は update.yy.to.xx.list)

もし、ftp でファイル転送中に時間切れを起こすようでしたら、ディレクトリ outside_usa 内のサイズの小さいパッケージが利用できます。

anonymous ftp に慣れていない人は付録 A.1 の注釈付きの手順に従って下さい。

A.1　anonymous ftpによるファイルの取得

以下のようにタイプして下さい。

% ftp 138.67.12.4	—	138.67.12.4 が私達の ftp サイトです
username: anonymous	—	ユーザー名は “anonymous”
password: yourname@your.machine.name	—	ここに何かを入力して下さい
ftp>	—	ftp 接続しているときに表示されるプロンプトです

これで、ftp を通じて CWP の anonymous ftp サイトへログインできました。以下のようにして下さい。

ftp> ls — ディレクトリの内容の表示。
ftp> cd dirname — “dirname” ディレクトリへの変更。
ftp> binary — ファイル転送のための “バイナリ・モード” の設定。
バイナリ・ファイルを転送しようとする前にこれをしないといけません。これには some_name.tar.Z 拡張子形式のすべてが含まれます。
ftp> get filename — “filename” を私達のサイトからあなたのマシンへ転送します。
ftp> mget pattern* — “pattern*” という名前のすべてのファイルを転送します。
例えば ;

ftp> mget *.tar.Z — “*.tar.Z” という名前のすべてのファイルをあなたのマシンに転送します。ftp が実際に転送をする前にこの名前のパターンの各々のファイルを本当に転送したいかを尋ねられます。
ftp> bye — ftp を終了します。

A.2 パッケージをインストールするための必須事項

パッケージをインストールするための必須事項はこれだけです :

1. UNIX オペレーティング・システムが動作するマシン。
2. ANSI C 準拠のコンパイラ。
3. インクルード・ファイルをサポートしているバージョンの make。
4. ソースとバイナリのための 16–60 MB (システムによる) のディスク空き容量。もし容量が問題なら、$CWPROOT/bin に cd して、”strip *” とタイプすることにより、コンパイルされたバイナリのサイズを小さくすることができます。

パッケージは以下のシステムに問題なくインストールされています。

- IBM RS6000
- SUN SPARC STATIONS
- HP 9000 シリーズ
- HP Apollo
- NeXT
- Convex
- DEC
- Silicon Graphics
- LINUX, PRIME TIME, SCO, Free BSD, ESIX, NeXTSTEP 486 の走っている PC

配布物の中には幾つかのプラットフォームにおける特別な注意を書いた README ファイルがあります。インストールの問題に関しては SU ユーザー・コミュニティの報告に依っています。もし問題が生じたら、お知らせ下さい。

配布物にはインストール手順を詳述する一連のファイルが含まれています。以下の順に読んで下さい。

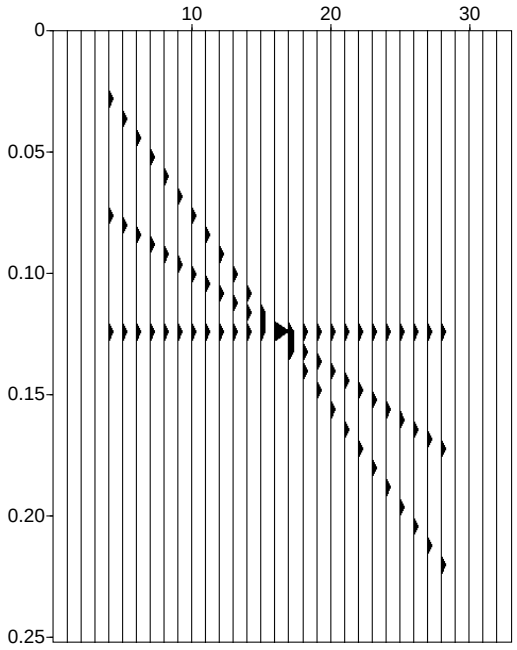

図 A.1: **suplane** のパイプラインの出力

`LEGAL_STATEMENT`	—	ライセンス、法的事項
`README_BEFORE_UNTARRING`	—	最初の情報
`README_FIRST`	—	一般的情報
`README_TO_INSTALL`	—	インストールの解説
`Portability/README_*`	—	さまざまなプラットフォームの移植に関する情報
`README_GETTING_STARTED`	—	プログラムの使い方

これらのファイルの多くは untar_me_first.xx.tar.Z に含まれています。

A.3 簡単なテスト

インストールが終了したら、簡単なテストによって、地震探査処理システムが動作するか確かめてみることができます。X-Windows が走っているマシンでは、

```
% suplane | suximage &
```

という “パイプライン” により図 A.1 に示される図が表示されます。PostScript プリンタを持っていれば、

```
% suplane | supswigb | lpr
```

というパイプを用いてハードコピーを印刷できます。Display PostScript か PostScript プレビュアがあれば、パイプ中の `lpr` コマンドを PostScript ファイルを開くコマンドに置き換えればスクリーンに表示できます。例えば：

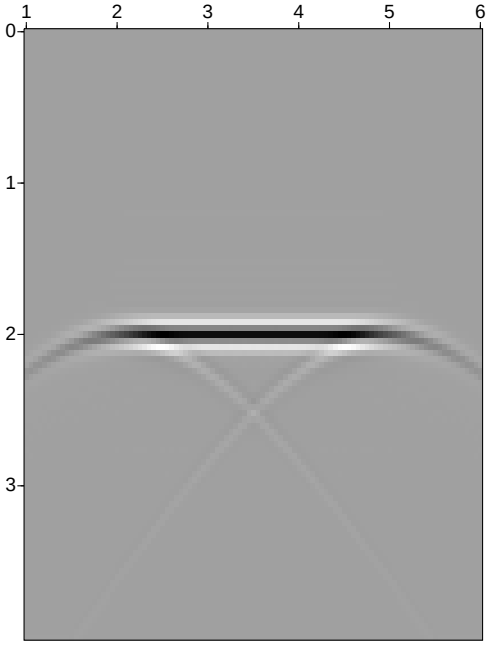

図 A.2: **susynlv** のパイプラインの出力

```
% suplane | supswigb | ghostview -
```

別のテスト用パイプラインは、

```
% susynlv | supsimage | lpr
```

```
% susynlv | supswigb | ghostview -
```

付 録B　ヘルプ機能

B.1 Suhelp

```
% suhelp
```

の出力の全テキストは以下の通りです:

```
CWP PROGRAMS: (no self-documentation)
ctrlstrip    downfort    fcat    isatty    maxints    pause    t    upfort

PAR PROGRAMS: (programs with self-documentation)
a2b           grm           rayt2d        smoothint2    unisam2       xy2z
b2a           h2b           recast        subset        vel2stiff     z2xyz
dzdv          kaperture     regrid3       swapbytes     velconv
farith        makevel       resamp        transp        velpert
ftnstrip      mkparfile     smooth2       unif2         vtlvz
ftnunstrip    prplot        smooth3d      unisam        wkbj


press return key to continue
SU PROGRAMS: (self-documented programs for SU data )
  SU data is "SEGY" data run through "segyclean"
bhedtopar     sudmofkcw     sukdmig3d     suop          sustkvel
dt1tosu       sudmotivz     sukdsyn2d     suop2         sustolt
segdread      sudmotx       sukfilter     supack1       sustrip
segyclean     sudmovz       sukfrac       supack2       suswapbytes
segyhdrs      suea2df       sukill        supaste       susyncz
segyread      suedit        sulog         supef         susynlv
segywrite     sufdmod2      sumax         supgc         susynlvcw
setbhed       sufft         sumean        supickamp     susynlvfti
su3dchart     sufilter      sumedian      suplane       susynvxz
suabshw       suflip        sumigfd       suput         susynvxzcs
suacor        sufrac        sumigffd      suquantile    sutab
suaddhead     sufxdecon     sumiggbzo     suradon       sutaper
suaddnoise    sugabor       sumigprefd    suramp        sutaup
suamp         sugain        sumigpreffd   surange       sutsq
suascii       sugazmig      sumigprepspi  surecip       suttoz
suattributes  suget         sumigpresp    sureduce      sutvband
suazimuth     sugethw       sumigps       surelan       suunpack1
subfilt       suharlan      sumigpspi     suresamp      suunpack2
suchart       suhilb        sumigpsti     suresstat     suvcat
suchw         suifft        sumigsplit    sushape       suvelan
sucommand     suilog        sumigtk       sushift       suvibro
suconv        suimp2d       sumigtopo2d   sushw         suvlength
sudatumfd     suimp3d       sumix         susort        suweight
sudatumk2dr   suinterp      sumute        suspecfk      suwind
sudatumk2ds   suintvel      sunmo         suspecfx      suxcor
sudipfilt     suinvvxzco    sunormalize   suspeck1k2    suxedit
sudivcor      suinvzco3d    sunull        suspike       suzero
```

```
sudivstack    suk1k2filter  suocext       sustack
sudmofk       sukdmig2d     suoldtonew    sustatic


press return key to continue

Delaunay Triangulation Materials:
gbbeam      normray     tri2uni     trimodel    triray      triseis     uni2tri

Tetrahedra Materials:
sutetraray  tetramod

X-windows GRAPHICS for Delaunay Triangulation:
sxplot

press return key to continue

Straight X-windows GRAPHICS:
lcmap       scmap       xepsb       ximage      xpsp
lprop       xcontour    xepsp       xpicker     xwigb

X-Toolkit based X-windows GRAPHICS:
xgraph    xmovie    xrects

Motif-based X-windows GRAPHICS:
fftlab

X-windows GRAPHICS: for SU ("segyclean"-ed SEGY)  data sets
suxcontour    suximage      suxmovie      suxwigb
suxgraph      suxmax        suxpicker

PostScript GRAPHICS:
psbbox       pscube       psgraph      pslabel      psmerge      pswigb
pscontour    psepsi       psimage      psmanager    psmovie      pswigp


PostScript GRAPHICS: for SU ("segyclean"-ed SEGY) data sets
supscontour    supsgraph      supsmax        supswigb
supscube       supsimage      supsmovie      supswigp

press return key to continue

Wavelet Packet Compression:
wpc1comp2      wpc1uncomp2

Wavelet Packet Compression:
wpccompress      wpcuncompress

2D Discrete Cosine Transform Compression:
dctcomp    dctuncomp  entropy    wptcomp    wptuncomp  wtcomp     wtuncomp

press return key to continue

CWP SHELLS SCRIPTS:
Grep        cwpfind     overwrite       time_now        weekday
argv        dirtree     precedence      todays_date     why
copyright   filetype    replace         usernames       zap
```

```
cpall       newcase     this_year       varlist

PAR SHELLS SCRIPTS:
gendocs      updatedoc        updatedocall     updatehead

POSTSCRIPT RELATED SHELLS SCRIPTS:
merge2     merge4

SU SHELLS SCRIPTS:
lookpar       rmaxdiff      sudiff        sufind       suhelp     unglitch
maxdiff       suagc         sudoc         sufind2      sukeyword
recip         suband        suenv         sugendocs    suname

press return key to continue

Use: "suname" to list the name and a brief description of all
of the CWP codes.

Use "sufind" to find programs by keyword/name fragment.

Use: "gendocs" to compile a LaTeX document listing all self-docs.

Use: "sukeyword" to find the SEGY header field keyword definitions.

Type: "program_name <CR>" to view its self documentation

Note: not all programs listed here have the self-documentation feature
Type:  "sudoc  program_name" to list information for these programs.

For answers to Frequently Asked Questions, see the contents of:
 /usr/local/cwp/src/faq
```

B.2 Suname

```
% suname
```

の出力の全テキストは以下の通りです:

```
 -----  CWP Free Programs -----
CWPROOT=/usr/local/cwp

Mains:

In CWPROOT/src/cwp/main:
* CTRLSTRIP - Strip non-graphic characters
* DOWNFORT - change Fortran programs to lower case, preserving strings
* FCAT - fast cat with 1 read per file
* ISATTY - pass on return from isatty(2)
* MAXINTS - Compute maximum and minimum sizes for integer types
* PAUSE - prompt and wait for user signal to continue
* T - time and date for non-military types
* UPFORT - change Fortran programs to upper case, preserving strings

In CWPROOT/src/par/main:
A2B - convert ascii floats to binary
```

```
B2A - convert binary floats to ascii
DZDV - determine depth derivative with respect to the velocity
FARITH - File ARITHmetic -- perform simple arithmetic with binary files
FARITH - File ARITHmetic -- perform simple arithmetic with binary files
FTNSTRIP - convert a file of floats plus record delimiters created
GRM - Generalized Reciprocal refraction analysis for a single layer
H2B - convert 8 bit hexidecimal floats to binary
KAPERTURE - generate the k domain of a line scatterer for a seismic array
MAKEVEL - MAKE a VELocity function v(x,y,z)
MKPARFILE - convert ascii to par file format
PRPLOT - PRinter PLOT of 1-D arrays f(x1) from a 2-D function f(x1,x2)
RAYT2D - traveltime Tables calculated by 2D paraxial RAY tracing
RECAST - RECAST data type (convert from one data type to another)
REGRID3 - REwrite a [ni3][ni2][ni1] GRID to a [no3][no2][no1] 3-D grid
RESAMP - RESAMPle the 1st dimension of a 2-dimensional function f(x1,x2)
SMOOTH2 --- SMOOTH a uniformly sampled 2d array of data, within a user-
SMOOTH3D - 3D grid velocity SMOOTHing by the damped least squares
SMOOTHINT2 --- SMOOTH non-uniformly sampled INTerfaces, via the damped
SUBSET - select a SUBSET of the samples from a 3-dimensional file
SWAPBYTES - SWAP the BYTES of various  data types
TRANSP - TRANSPose an n1 by n2 element matrix
UNIF2 - generate a 2-D UNIFormly sampled velocity profile from a layered
UNISAM - UNIformly SAMple a function y(x) specified as x,y pairs
UNISAM2 - UNIformly SAMple a 2-D function f(x1,x2)
VEL2STIFF - Transforms VELocities, densities, and Thomsen parameters
VELCONV - VELocity CONVersion
VELPERT - estimate velocity parameter perturbation from covariance
VTLVZ -- Velocity as function of Time for Linear V(Z);
WKBJ - Compute WKBJ ray theoretic parameters, via finite differencing
XY2Z - converts (X,Y)-pairs to spike Z values on a uniform grid
Z2XYZ - convert binary floats representing Z-values to ascii

In CWPROOT/src/psplot/main:
PSBBOX - change BoundingBOX of existing PostScript file
PSCONTOUR - PostScript CONTOURing of a two-dimensional function f(x1,x2)
PSCUBE - PostScript image plot of a data CUBE
PSEPSI - add an EPSI formatted preview bitmap to an EPS file
PSGRAPH - PostScript GRAPHer
PSIMAGE - PostScript IMAGE plot of a uniformly-sampled function f(x1,x2)
PSLABEL - output PostScript file consisting of a single TEXT string
PSMANAGER - printer MANAGER for HP 4MV Laserjet PostScript
PSMERGE - MERGE PostScript files
PSMOVIE - PostScript MOVIE plot of a uniformly-sampled function f(x1,x2,x3)
PSWIGB - PostScript WIGgle-trace plot of f(x1,x2) via Bitmap
PSWIGP - PostScript WIGgle-trace plot of f(x1,x2) via Polygons

In CWPROOT/src/xplot/main:
* LCMAP - List Color Map of root window of default screen
* LPROP - List PROPerties of root window of default screen of display
* SCMAP - set default standard color map (RGB_DEFAULT_MAP)
XCONTOUR - X CONTOUR plot of f(x1,x2) via vector plot call
* XESPB - X windows display of Encapsulated PostScript as a single Bitmap
* XEPSP - X windows display of Encapsulated PostScript
XIMAGE - X IMAGE plot of a uniformly-sampled function f(x1,x2)
XPICKER - X wiggle-trace plot of f(x1,x2) via Bitmap with PICKing
XPICKER - X wiggle-trace plot of f(x1,x2) via Bitmap with PICKing
* XPSP - Display conforming PostScript in an X-window
```

```
XWIGB - X WIGgle-trace plot of f(x1,x2) via Bitmap

In CWPROOT/src/Xtcwp/main:
XGRAPH - X GRAPHer
XMOVIE - image one or more frames of a uniformly sampled function f(x1,x2)
XRECTS - plot rectangles on a two-dimensional grid

In CWPROOT/src/Xmcwp/main:
* FFTLAB - Motif-X based graphical 1D Fourier Transform

In CWPROOT/src/su/graphics/psplot:
SUPSCONTOUR - PostScript CONTOUR plot of a segy data set
SUPSCUBE - PostScript CUBE plot of a segy data set
SUPSGRAPH - PostScript GRAPH plot of a segy data set
SUPSIMAGE - PostScript IMAGE plot of a segy data set
SUPSMAX - PostScript of the MAX, min, or absolute max value on each trace
SUPSMOVIE - PostScript MOVIE plot of a segy data set
SUPSWIGB - PostScript Bit-mapped WIGgle plot of a segy data set
SUPSWIGP - PostScript Polygon-filled WIGgle plot of a segy data set

In CWPROOT/src/su/main:
BHEDTOPAR - convert a Binary tape HEaDer file to PAR file format
DT1TOSU - Convert ground-penetrating radar data in the Sensors & Software
SEGDREAD - read an SEG-D tape
SEGYCLEAN - zero out unassigned portion of header
SEGYHDRS - make SEG-Y ascii and binary headers for segywrite
SEGYREAD - read an SEG-Y tape
SEGYWRITE - write an SEG-Y tape
SETBHED - SET the fields in a SEGY Binary tape HEaDer file, as would be
SU3DCHART - plot x-midpoints vs. y-midpoints for 3-D data
SUABSHW - replace header key word by its absolute value
SUACOR - auto-correlation
SUADDHEAD - put headers on bare traces and set the tracl and ns fields
SUADDNOISE - add noise to traces
SUAMP - output amp, phase, real or imag trace from
SUASCII - print non zero header values and data
SUATTRIBUTES - trace ATTRIBUTES instantanteous amplitude, phase,
SUAZIMUTH - compute trace AZIMUTH given the sx,sy,gx,gy header fields
SUBFILT - apply Butterworth bandpass filter
SUCHART - prepare data for x vs. y plot
SUCHW - Change Header Word using one or two header word fields
SUCOMMAND - pipe traces having the same key header word to command
SUCONV - convolution with user-supplied filter
SUDATUMFD - 2D zero-offset Finite Difference acoustic wave-equation
SUDATUMK2DR - Kirchhoff datuming of receivers for 2D prestack data
SUDATUMK2DS - Kirchhoff datuming of sources for 2D prestack data
SUDIPFILT - DIP--or better--SLOPE Filter in f-k domain
SUDIVCOR - Divergence (spreading) correction
SUDIVSTACK -  Diversity Stacking using either average power or peak
SUDMOFK - DMO via F-K domain (log-stretch) method for common-offset gathers
SUDMOFKCW - converted-wave DMO via F-K domain (log-stretch) method for
SUDMOTIVZ - DMO for Transeversely Isotropic V(Z) media for common-offset
SUDMOTX - DMO via T-X domain (Kirchhoff) method for common-offset gathers
SUDMOVZ - DMO for V(Z) media for common-offset gathers
SUEA2DF - SU version of (an)elastic anisotropic 2D finite difference
SUEDIT - examine segy diskfiles and edit headers
SUFDMOD2 - Finite-Difference MODeling (2nd order) for acoustic wave equation
```

```
SUFFT - fft real time traces to complex frequency traces
SUFILTER - applies a zero-phase, sine-squared tapered filter
SUFLIP - flip a data set in various ways
SUFRAC -- take general (fractional) time derivative or integral of
SUFXDECON - random noise attenuation by FX-DECONvolution
SUGABOR -  Outputs a time-frequency representation of seismic data via
SUGAIN - apply various types of gain to display traces
SUGAZMIG - SU version of Jeno GAZDAG's phase-shift migration
SUGET  - Connect SU program to file descriptor for input stream.
SUGETHW - sugethw writes the values of the selected key words
SUHARLAN - signal-noise separation by the invertible linear
SUHILB - Hilbert transform
SUIFFT - fft complex frequency traces to real time traces
SUILOG -- time axis inverse log-stretch of seismic traces
SUIMP2D - generate shot records for a line scatterer
SUIMP3D - generate inplane shot records for a point
SUINTERP - interpolate traces using automatic event picking
SUINTVEL - convert stacking velocity model to interval velocity model
SUINVXZCO - Seismic INVersion of Common Offset data for a smooth
SUINVZCO3D - Seismic INVersion of Common Offset data with V(Z) velocity
SUK1K2FILTER - symmetric box-like K-domain filter defined by the
SUKDMIG2D - Kirchhoff Depth Migration of 2D poststack/prestack data
SUKDSYN2D - Kirchhoff Depth SYNthesis of 2D seismic data from a
SUKFILTER - radially symmetric K-domain, sin^2-tapered, polygonal
SUKFRAC - apply FRACtional powers of i|k| to data, with phase shift
SUKILL - zero out traces
SULOG -- time axis log-stretch of seismic traces
SUMAX - get trace by trace local/global maxima, minima, or absolute maximum
SUMEDIAN - MEDIAN filter about a user-defined polygonal curve with
SUMIGFD - 45 and 60 degree Finite difference migration for
SUMIGFFD - Fourier finite difference migration for
SUMIGGBZO - MIGration via Gaussian Beams of Zero-Offset SU data
SUMIGPREFD --- The 2-D prestack common-shot 45-90 degree
SUMIGPREFFD - The 2-D prestack common-shot Fourier finite-difference
SUMIGPREPSPI --- The 2-D PREstack commom-shot Phase-Shift-Plus
SUMIGPRESP - The 2-D prestack common-shot split-step Fourier
SUMIGPS - MIGration by Phase Shift with turning rays
SUMIGPSPI - Gazdag's phase-shift plus interpolation migration
SUMIGPSTI - MIGration by Phase Shift for TI media with turning rays
SUMIGSPLIT - Split-step depth migration for zero-offset data.
SUMIGTK - MIGration via T-K domain method for common-midpoint stacked data
SUMIGTOPO2D - Kirchhoff Depth Migration of 2D postack/prestack data
SUMIX - compute weighted moving average (trace MIX) on a panel
SUMUTE - mute above (or below) a user-defined polygonal curve with
SUNMO - NMO for an arbitrary velocity function of time and CDP
SUNORMALIZE - Trace balancing by rms, max, or median
SUNULL - create null (all zeroes) traces
SUOCEXT - smaller offset extrapolation via Offset Continuation
SUOLDTONEW - convert existing su data to xdr format
SUOP - do unary arithmetic operation on segys
SUOP2 - do a binary operation on two data sets
SUPACK1 - pack segy trace data into chars
SUPACK2 - pack segy trace data into 2 byte shorts
SUPASTE - paste existing SEGY headers on existing data
SUPEF - Wiener predictive error filtering
SUPGC   -   Programmed Gain Control--apply agc like function
SUPICKAMP - pick amplitudes within user defined and resampled window
```

```
SUPLANE - create common offset data file with up to 3 planes
SUPUT - Connect SU program to file descriptor for output stream.
SUQUANTILE - display some quantiles or ranks of a data set
SURADON - compute forward or reverse Radon transform or remove multiples
SURAMP - Linearly taper the start and/or end of traces to zero.
SURANGE - get max and min values for non-zero header entries
SURECIP - sum opposing offsets in prepared data (see below)
SUREDUCE - convert traces to display in reduced time
SURELAN - compute residual-moveout semblance for cdp gathers based
SURESAMP - Resample in time
SURESSTAT - Surface consistent source and receiver statics calculation
SUSHAPE - Wiener shaping filter
SUSHIFT - shifted/windowed traces in time
SUSHW - Set one or more Header Words using trace number, mod and
SUSORT - sort on any segy header keywords
SUSPECFK - F-K Fourier SPECtrum of data set
SUSPECFX - Fourier SPECtrum (T -> F) of traces
SUSPECK1K2 - 2D (K1,K2) Fourier SPECtrum of (x1,x2) data set
SUSPIKE - make a small spike data set
SUSTACK - stack adjacent traces having the same key header word
SUSTATIC - Elevation static corrections, apply corrections from
SUSTKVEL - convert constant dip layer interval velocity model to the
SUSTOLT - Stolt migration for stacked data or common-offset gathers
SUSTRIP - remove the SEGY headers from the traces
SUSWAPBYTES - SWAP the BYTES in SU data to convert data from big endian
SUSYNCZ - SYNthetic seismograms for piecewise constant V(Z) function
SUSYNLV - SYNthetic seismograms for Linear Velocity function
SUSYNLVCW - SYNthetic seismograms for Linear Velocity function
SUSYNLVFTI - SYNthetic seismograms for Linear Velocity function in a factored
SUSYNVXZ - SYNthetic seismograms of common offset V(X,Z) media via
SUSYNVXZCS - SYNthetic seismograms of common shot in V(X,Z) media via
SUTAB - print non zero header values and data for non-graphic terminals
SUTAPER - Taper the edge traces of a data panel to zero.
SUTAUP - forwared and inverse T-X and F-K global slant stacks
SUTSQ -- time axis time-squared stretch of seismic traces
SUTTOZ - resample from time to depth
SUTVBAND - time-variant bandpass filter (sine-squared taper)
SUUNPACK1 - unpack segy trace data from chars to floats
SUUNPACK2 - unpack segy trace data from shorts to floats
SUVCAT - append one data set to another
SUVELAN - compute stacking velocity semblance for cdp gathers
SUVIBRO - Generates a Vibroseis sweep (linear, linear-segment,
SUVLENGTH - Adjust variable length traces to common length
SUWIND - window traces by key word
SUXCOR - correlation with user-supplied filter
SUXEDIT - examine segy diskfiles and edit headers
SUZERO -- zero-out data within a time window

In CWPROOT/src/su/graphics/psplot:
SUPSCONTOUR - PostScript CONTOUR plot of a segy data set
SUPSCUBE - PostScript CUBE plot of a segy data set
SUPSGRAPH - PostScript GRAPH plot of a segy data set
SUPSIMAGE - PostScript IMAGE plot of a segy data set
SUPSMAX - PostScript of the MAX, min, or absolute max value on each trace
SUPSMOVIE - PostScript MOVIE plot of a segy data set
SUPSWIGB - PostScript Bit-mapped WIGgle plot of a segy data set
SUPSWIGP - PostScript Polygon-filled WIGgle plot of a segy data set
```

```
In CWPROOT/src/su/graphics/xplot:
SUXCONTOUR - X CONTOUR plot of Seismic UNIX tracefile via vector plot call
SUXGRAPH - X-windows GRAPH plot of a segy data set
SUXIMAGE - X-windows IMAGE plot of a segy data set
SUXMAX - X-windows graph of the MAX, min, or absolute max value on
SUXMOVIE - X MOVIE plot of a segy data set
SUXPICKER - X-windows  WIGgle plot PICKER of a segy data set
SUXWIGB - X-windows Bit-mapped WIGgle plot of a segy data set

In CWPROOT/src/tri/main:
GBBEAM - Gaussian beam synthetic seismograms for a sloth model
NORMRAY - dynamic ray tracing for normal incidence rays in a sloth model
TRI2UNI - convert a TRIangulated model to UNIformly sampled model
TRIMODEL - make a triangulated sloth (1/velocity^2) model
TRIRAY - dynamic RAY tracing for a TRIangulated sloth model
TRISEIS - Gaussian beam synthetic seismograms for a sloth model
UNI2TRI - convert UNIformly sampled model to a TRIangulated model

In CWPROOT/src/xtri:
SXPLOT - X Window plot a triangulated sloth function s(x1,x2)

In CWPROOT/src/tri/graphics/psplot:
SPSPLOT - plot a triangulated sloth function s(x,z) via PostScript

In CWPROOT/src/comp/dct/main:
DCTCOMP - Compression by Discrete Cosine Transform
DCTUNCOMP - Discrete Cosine Transform Uncompression
ENTROPY - compute the ENTROPY of a signal
WPTCOMP - Compression by Wavelet Packet Compression
WPTUNCOMP - Uncompress  WPT compressed data
WTCOMP - Compression by Wavelet Transform
WTUNCOMP - UNCOMPression of WT compressed data

In CWPROOT/src/comp/dwpt/1d/main:
WPC1COMP2 --- COMPress a 2D seismic section trace-by-trace using
WPC1UNCOMP2 --- UNCOMPRESS a 2D seismic section, which has been

In CWPROOT/src/comp/dwpt/2d/main:
WPCCOMPRESS --- COMPRESS a 2D section using Wavelet Packets
WPCUNCOMPRESS --- UNCOMPRESS a 2D section

Shells:

In CWPROOT/src/cwp/shell:
# Grep  - recursively call egrep in pwd
# ARGV - give examples of dereferencing char **argv
# COPYRIGHT - insert CSM COPYRIGHT lines at top of files in working directory
# CPALL , RCPALL - for local and remote directory tree/file transfer
# CWPFIND - look for files with patterns in CWPROOT/src/cwp/lib
# DIRTREE - show DIRectory TREE
# FILETYPE - list all files of given type
# NEWCASE - Changes the case of all the filenames in a directory, dir
# OVERWRITE - copy stdin to stdout after EOF
# PRECEDENCE - give table of C precedences from Kernighan and Ritchie
# REPLACE - REPLACE string1 with string2  in files
# THIS_YEAR - print the current year
```

```
# TIME_NOW - prints time in ZULU format with no spaces
# TODAYS_DATE - prints today's date in ZULU format with no spaces
# USERNAMES - get list of all login names
# VARLIST - list variables used in a Fortran program
# WEEKDAY - prints today's WEEKDAY designation
# ZAP - kill processes by name

In CWPROOT/src/par/shell:
# GENDOCS - generate complete list of selfdocs in latex form
# UPDATEDOC - put self-docs in ../doc/Stripped and ../doc/Headers
# UPDATEDOCALL - put self-docs in ../doc/Stripped
# UPDATEHEAD - update ../doc/Headers/Headers.all

In CWPROOT/src/psplot/shell:
# MERGE2 - put 2 standard size PostScript figures on one page
# MERGE4 - put 4 standard size PostScript plots on one page

In CWPROOT/src/su/shell:
# LOOKPAR - show getpar lines in SU code with defines evaluated
# MAXDIFF - find absolute maximum difference in two segy data sets
# RECIP - sum opposing (reciprocal) offsets in cdp sorted data
# RMAXDIFF - find percentage maximum difference in two segy data sets
# SUAGC - perform agc on SU data
# SUBAND - Trapezoid-like Sin squared tapered Bandpass filter via  SUFILTER
# SUDIFF, SUSUM, SUPROD, SUQUO, SUPTDIFF, SUPTSUM,
# SUDOC - get DOC listing for code
# SUENV - Instantaneous amplitude, frequency, and phase via: SUATTRIBUTES
# SUFIND - get info from self-docs
# SUFIND2 - get info from self-docs
# SUGENDOCS - generate complete list of selfdocs in latex form
# SUHELP - list the CWP/SU programs and shells
# SUKEYWORD -- guide to SU keywords in segy.h
# SUNAME - get name line from self-docs
# UNGLITCH - zonk outliers in data

Libs:

In CWPROOT/src/cwp/lib:
ABEL - Functions to compute the discrete ABEL transform:
AIRY - Approximate the Airy functions  Ai(x), Bi(x) and their respective
ALLOC - Allocate and free multi-dimensional arrays
ANTIALIAS - Butterworth anti-aliasing filter
AXB - Functions to solve a linear system of equations Ax=b by LU
BIGMATRIX - Functions to manipulate 2-dimensional matrices that are too big
BUTTERWORTH - Functions to design and apply Butterworth filters:
COMPLEX - Functions to manipulate complex numbers
COMPLEXD - Functions to manipulate double-precision complex numbers
COMPLEXF  - Subroutines to perform operations on complex numbers.
CONVOLUTION - Compute z = x convolved with y
CUBICSPLINE - Functions to compute CUBIC SPLINE interpolation coefficients
DBLAS - Double precision Basic Linear Algebra subroutines
DGE - Double precision Gaussian Elimination matrix subroutines  adapted
FRANNOR - functions to generate a pseudo-random float normally distributed
FRANUNI - Functions to generate a pseudo-random float uniformly distributed
HANKEL - Functions to compute discrete Hankel transforms
HILBERT - Compute Hilbert transform y of x
HOLBERG1D - Compute coefficients of Holberg's 1st derivative filter
```

```
IBMFLT_ - Convert IBM tape format to floats and back
INTCUB - evaluate y(x), y'(x), y''(x), ... via piecewise cubic interpolation
INTL2B - bilinear interpolation of a 2-D array of bytes
INTLIN - evaluate y(x) via linear interpolation of y(x[0]), y(x[1]), ...
INTSINC8 - Functions to interpolate uniformly-sampled data via 8-coeff. sinc
INTTABLE8 -  Interpolation of a uniformly-sampled complex function y(x)
MKDIFF - Make an n-th order DIFFerentiator via Taylor's series method.
MKHDIFF - Compute filter approximating the bandlimited HalF-DIFFerentiator.
MKSINC - Compute least-squares optimal sinc interpolation coefficients.
MNEWT - Solve non-linear system of equations f(x) = 0 via Newton's method
PFAFFT - Functions to perform Prime Factor (PFA) FFT's, in place
POLAR - Functions to map data in rectangular coordinates to polar and vise versa
PRINTERPLOT - Functions to make a printer plot of a 1-dimensional array
QUEST - Functions to ESTimate Quantiles:
RESSINC8 - Functions to resample uniformly-sampled data  via 8-coefficient sinc
RFWTVA - Rasterize a Float array as Wiggle-Trace-Variable-Area.
RFWTVAINT - Rasterize a Float array as Wiggle-Trace-Variable-Area, with
SBLAS - Single precision Basic Linear Algebra Subroutines
SCAXIS - compute a readable scale for use in plotting axes
SGA - Single precision general matrix functions adapted from LINPACK FORTRAN:
SHFS8R - Shift a uniformly-sampled real-valued function y(x) via
SINC - Return SINC(x) for as floats or as doubles
SORT - Functions to sort arrays of data or arrays of indices
SQR - Single precision QR decomposition functions adapted from LINPACK FORTRAN:
STOEP - Functions to solve a symmetric Toeplitz linear system of equations
SWAPBYTE - Functions to SWAP the BYTE order of binary data
TEMPORARY_FILENAME - Creates a file name in a user-specified directory.
TRIDIAGONAL - Functions to solve tridiagonal systems of equations Tu=r for u.
VANDERMONDE - Functions to solve Vandermonde system of equations Vx=b
WAVEFORMS   Subroutines to define some wavelets for modeling of seimic
XCOR - Compute z = x cross-correlated with y
XINDEX - determine index of x with respect to an array of x values
YCLIP - Clip a function y(x) defined by linear interpolation of the
YXTOXY - Compute a regularly-sampled, monotonically increasing function x(y)
ZASC - routine to translate ncharacters from ebcdic to ascii
ZEBC - routine to translate ncharacters from ascii to ebcdic

In CWPROOT/src/par/lib:
VND *VNDop(int mode, long lwmax, int ndim, long *N, long npanels,
ATOPKGE - convert ascii to arithmetic and with error checking
DOCPKGE - Function to implement the CWP self-documentation facility
EALLOC - Allocate and free multi-dimensional arrays with error reports.
ERRPKGE - routines for reporting errors
FILESTAT - Functions to determine and output the type of a file from file
GETPARS - Functions to GET PARameterS from the command line. Numeric
MODELING - Seismic Modeling Subroutines for SUSYNLV and clones
SMOOTH - Functions to compute smoothing of 1-D or 2-D input data
SUBCALLS - routines for system functions with error checking
SYSCALLS - routines for SYSTEM CALLs with error checking

In CWPROOT/src/su/lib:
FGETTR - Routines to get an SU trace from a file
FPUTTR - Routines to put an SU trace to a file
HDRPKGE - routines to access the SEGY header via the hdr structure.
TABPLOT - TABPLOT selected sample points on selected trace
VALPKGE - routines to handle variables of type Value
```

```
In CWPROOT/src/psplot/lib:
BASIC - Basic C function interface to PostScript
PSAXESBOX - Functions to draw PostScript axes and estimate bounding box
PSAXESBOX3 -  Functions draw an axes box via PostScript, estimate bounding box
PSCAXESBOX - Draw an axes box for cube via PostScript
PSCONTOUR - draw contour of a two-dimensional array via PostScript
PSDRAWCURVE - Functions to draw a curve from a set of points
PSLEGENDBOX - Functions to draw PostScript axes and estimate bounding box
PSWIGGLE - draw wiggle-trace with (optional) area-fill via PostScript

In CWPROOT/src/xplot/lib:
AXESBOX - Functions to draw axes in X-windows graphics
COLORMAP - Functions to manipulate X colormaps:
DRAWCURVE - Functions to draw a curve from a set of points
IMAGE - Function for making the image in an X-windows image plot
LEGENDBOX - draw a labeled axes box for a legend (i.e. colorscale)
RUBBERBOX -  Function to draw a rubberband box in X-windows plots
WINDOW - Function to create a window in X-windows graphics
XCONTOUR - draw contour of a two-dimensional array via X vectorplot calls

In CWPROOT/src/Xtcwp/lib:
AXES - the Axes Widget
COLORMAP - Functions to manipulate X colormaps:
FX - Functions to support floating point coordinates in X
MISC - Miscellaneous X-Toolkit functions
RESCONV - general purpose resource type converters
RUBBERBOX -  Function to draw a rubberband box in X-windows plots

In CWPROOT/src/Xmcwp/lib:
RADIOBUTTONS -  convenience functions creating and using radio buttons
SAMPLES - Motif-based Graphics Functions

In CWPROOT/src/tri/lib:
CHECK - CHECK triangulated models
CIRCUM - define CIRCUMcircles for Delaunay triangulation
COLINEAR - determine if edges or vertecies are COLINEAR in triangulated
CREATE - create model, boundary edge triangles, edge face, edge vertex, add
DELETE - DELETE vertex, model, edge, or boundary edge from triangulated model
DTE - Distance to Edge
FIXEDGES - FIX or unFIX EDGES between verticies
INSIDE -  Is a vertex or point inside a circum circle, etc. of a triangulated
NEAREST - NEAREST edge or vertex in triangulated model
PROJECT - project to edge in triangulated model
READWRITE - READ or WRITE a triangulated model

In CWPROOT/src/cwputils:
CPUSEC - return cpu time (UNIX user time) in seconds
CPUTIME - return cpu time (UNIX user time) in seconds using ANSI C built-ins
WALLSEC - Functions to time processes
WALLTIME - Function to show time a process takes to run

In CWPROOT/src/comp/dct/lib:
DCT1 - 1D Discreet Cosine Transform Routines
DCT2 - 2D Discrete Cosine Transform Routines
DCTALLOC - ALLOCate space for transform tables for 1D DCT
GETFILTER - GET wavelet FILTER type
LCT1 - functions used to perform the 1D Local Cosine Transform (LCT)
```

```
LPRED - Lateral Prediction of Several Plane Waves
WAVEPACK1 - 1D wavelet packet transform
WAVEPACK2 - 2D Wavelet PACKet transform
WAVEPACK1 - 1D wavelet packet transform
WAVETRANS2 - 2D wavelet transform by tensor-product of two 1D transforms

In CWPROOT/src/comp/dct/libutil:
BUFFALLOC - routines to ALLOCate/initialize and free BUFFers
HUFFMAN - Routines for in memory Huffman coding/decoding
PENCODING - Routines to en/decode the quantized integers for lossless
QUANT - QUANTization routines
RLE - routines for in memory silence en/decoding

In CWPROOT/src/comp/dwpt/1d/lib:
WBUFFALLOC -  routines to allocate/initialize and free buffers in wavelet
WPC1 - routines for compress a single seismic trace using wavelet packets
WPC1CODING - routines for encoding the integer symbols in 1D WPC
wpc1Quant - quantize
WPC1TRANS - routines to perform a 1D wavelet packet transform

In CWPROOT/src/comp/dwpt/2d/lib:
WPC - routines for compress a 2D seismic section using wavelet packets
WPCBUFFAL - routines to allocate/initialize and free buffers
WPCCODING - Routines for in memory coding of the quantized coeffiecients
WPCENDEC -  Wavelet Packet Coding, Encoding and Decoding routines
WPCHUFF -Routines for in memory Huffman coding/decoding
WPCPACK2 - routine to perform a 2D wavelet packet transform
WPCQUANT - quantization routines for WPC
WPCSILENCE - routines for in memory silence en/decoding
Shells:
Libs:
Shells:
Libs:
Shells:
Libs:

To search on a program name fragment, type:
     "suname name_fragment <CR>"

For more information type: "program_name <CR>"

  Items labeled with an asterisk (*) are C programs that may
  or may not have a self documentation feature.

  Items labeled with a pound sign (#) are shell scripts that may,
  or may not have a self documentation feature.

 To find information about these codes type:   sudoc program_name
```

B.3 Suhelp.html

これは Chris Liner 氏の SU ヘルプ・ページのテキスト一覧です。

```
                        SeismicUn*x

                    Version 33 (5 April 1999)
```

An HTML Help Browser

* This is a help browser for the SeismicUn*x free software package developed and maintained by the Center for Wave Phenomena at the Colorado School of Mines. The SU project is directed at CWP by John Stockwell.
* The author of this help facility is Dr. Christopher Liner (an alumnus of CWP) who is a faculty member in the Department of Geosciences at The University of Tulsa.
* Last updated January 16, 1998

- o The arrangement below is by funtionality
- o Clicking on a program name pulls up the selfdoc for that item
- o Your web browser's Find capability is useful if you have a fragment in mind (e.g. sort or nmo)
- o While programs may logically apply to more than one catagory below, each program appears only once

---

1. Functional Listing
   1. Data Compression
   2. Editing, Sorting and Manipulation
   3. Filtering, Transforms and Attributes
   4. Gain, NMO, Stack and Standard Processes
   5. Graphics
   6. Import/Export
   7. Migration and Dip Moveout
   8. Simulation and Model Building
   9. Utilities

* Alphabetical name list
* List is for scanning only, it does not pull up the selfdoc of a selected item.

* 280 items
* For further info on an interesting item use your browser's find command, then follow the link

---

Data Compression

Discrete Cosine Transform

	* dctcomp	* dctuncomp
Packing	* supack1	* suunpack1
	* supack2	* suunpack2
	* wpc1comp2	* wpc1uncomp2
Wavelet Transform	* wpccompress	* wpcuncompress
	* wptcomp	* wptuncomp
	* wtcomp	* wtuncomp

Go to top

Editing, Sorting and Manipulation

	* suabshw	* suquantile
	* suazimuth	* surange
Edit + tools	* subset	
	* suchw	* sushw
	* suedit	* sutab
	* sugethw	* suwind
	* sukill	* suxedit
Sort	* susort	
	* fcat	* suramp
	* maxdiff	* surecip, recip
	* suaddnoise	* suresamp, resamp
	* sudiff	* suswapbytes
Manipulate	* suflip	* sutaper
	* suinterp	* suvcat
	* sunull	* suzero
	* suop	* swapbytes
	* suop2	* transp

Go to top

Filtering, Transforms and Attributes

	* suband	
	* subfilt	* supef
One-Dimensional filtering	* suconv	* sushape
	* sufilter	* sutvband
	* sufrac	* suxcor
	* sudipfilt	* sumedian
Two-Dimensional filtering	* sufxdecon	* sukfilter
	* suk1k2filter	* sukfrac
	* entropy	* sulog, suilog
	* suamp	* suradon
	* suattributes	* sureduce
Transforms and Attributes	* suenv	* suspecfk
	* suhilb	* suspecfx
	* sufft, suifft	* suspeck1k2
	* sugabor	* sutaup
	* suharlan	* sutsq

Go to top

Gain, NMO, Stack and Standard Processes

	* suagc	* sunmo
	* sudivcor	* supgc
Standard Procs	* sugain	* suresstat
	* sumix	* sustack
	* sumute	* sustatic
		* unglitch
	* suacor	
Miscellaneous	* suttoz	
	* suvibro	
	* suvlength	
	* dzdv	
	* sudivstack	* suvelan

Velocity Analysis	* surelan	* velpert

Go to top

Graphics

X-window ... for SU format input data	* suxcontour	
	* suxgraph	* suxmovie
(see binary form for full options)	* suximage	* suxpicker
	* suxmax	* suxwigb
	* sxplot	
X-window ... for binary input data	* xcontour	* xmovie
	* xgraph	* xpicker
	* ximage	* xwigb
Postscript ... for SU format input data	* supscontour	* supsmovie (NeXT only)
	* supscube	
(see binary form for full options)	* supsgraph	* supswigb
	* supsimage	* supswigp
	* psbbox	
	* pscontour	* psmovie (NeXT only)
Postscript ... for binary input data	* pscube	* pswigb
	* psgraph	* pswigp
	* psimage	* spsplot
	* fftlab	
	* h2b	* scmap
	* lcmap, lprop	* su3dchart
	* merge2, merge4	* suchart
Utilities + misc	* prplot	* supsmax
	* psepsi	* xrects
	* psmanager	* xepsb (NeXT only)
	* psmerge	* xepsp (NeXT only)
	* pslabel	* xepsb (NeXT only)

Go to top

Import/Export

	* a2b	* segywrite
	* b2a	* setbhed
	* bhedtopar	* suaddhead
	* dt1tosu	* suascii
Import/Export	* ftnstrip	* suget
	* ftnunstrip	
	* recast	* suoldtonew
	* segdread	* supaste
	* segyclean	* suput
	* segyhdrs	* sustrip
	* segyread	* z2xyz

Go to top

Migration and Dip Moveout

```
                                  * sugazmig          * sumigps
   Poststack migration            * sumigfd           * sumigpspi
                                  * sumigffd          * sumigsplit
                                  * sumiggbzo         * sumigtk

                                  * sukdmig2d
   Prestack/Poststack migration   * sumigtopo2d
                                  * sustolt
   Prestack migration             * sumigprefd        * sumigprepspi
                                  * sumigpreffd       * sumigpresp
                                  * sudmofk           * sudmotx
   Dip Moveout                    * sudmofkcw         * sudmovz

   Datuming                       * sudatumk2dr
                                  * sudatumk2ds

                                 Go to top

   Simulation and Model Building
                                  * gbbeam            * susyncz
                                  * normray           * susynlv
                                  * rayt2d            * susynlvcw
   Simulation (aka modeling)      * sufdmod2          * susynvxz
                                  * suimp2d           * susynvxzcs
                                  * suimp3d           * triray
                                  * sukdsyn2d         * triseis
                                  * sureflpsvsh       * wkbj

                                  * makevel           * trimodel
                                  * regrid3           * uni2tri
                                  * suintvel          * unif2
   Model Building                 * sustkvel          * unisam
                                  * tmb               * unisam2
                                  * tri2uni           * velconv

                                  * kaperture         * suspike
                                  * smooth2           * trip (Mesa 3D item)
                                  * smooth3d          * viewer3 (Mesa 3D item)
   Utilities                      * smoothint2        * vtlvz
                                  * suplane           * xy2z

                                 Go to top

Utilities

                    * copyright       * overwrite
                    * cpall           * pause           * time_now
                    * cpusec          * precedence      * todays_date
                    * cputime         * replace         * updatedoc
                    * ctrlstrip       * rmaxdiff        * updatedocall
                    * dirtree         * sumax           * updatehead
   Utilities        * downfort        * sumean          * upfort
                    * farith          * sumedian        * usernames
                    * filetype        * sunormalize     * varlist
                    * isatty          * supickamp       * wallsec
```

```
                    * lookpar           * sushift         * walltime
                    * maxints           * suweight        * weekday
                    * mkparfile         * t               * zap
                    * newcase           * this_year
                    * cwpfind           * sugendocs
                    * gendocs           * suhelp
   Help utilities   * sudoc             * suname
                    * sufind            * sukeyword

                                  Go to top
```

B.4 SUKEYWORD - SU データ型 (SEG-Y) キーワードのリスト

以下は

```
% sukeyword -o
```

の出力です。

```
...skipping
typedef struct {    /* segy - trace identification header */

    int tracl;  /* trace sequence number within line */

    int tracr;  /* trace sequence number within reel */

    int fldr;   /* field record number */

    int tracf;  /* trace number within field record */

    int ep; /* energy source point number */

    int cdp;    /* CDP ensemble number */

    int cdpt;   /* trace number within CDP ensemble */

    short trid; /* trace identification code:
            1 = seismic data
            2 = dead
            3 = dummy
            4 = time break
            5 = uphole
            6 = sweep
            7 = timing
            8 = water break
            9---, N = optional use (N = 32,767)

            Following are CWP id flags:

             9 = autocorrelation

            10 = Fourier transformed - no packing
                 xr[0],xi[0], ..., xr[N-1],xi[N-1]
```

```
        11 = Fourier transformed - unpacked Nyquist
             xr[0],xi[0],...,xr[N/2],xi[N/2]

        12 = Fourier transformed - packed Nyquist
             even N:
             xr[0],xr[N/2],xr[1],xi[1], ...,
            xr[N/2 -1],xi[N/2 -1]
            (note the exceptional second entry)
             odd N:
             xr[0],xr[(N-1)/2],xr[1],xi[1], ...,
            xr[(N-1)/2 -1],xi[(N-1)/2 -1],xi[(N-1)/2]
            (note the exceptional second & last entries)

        13 = Complex signal in the time domain
             xr[0],xi[0], ..., xr[N-1],xi[N-1]

        14 = Fourier transformed - amplitude/phase
             a[0],p[0], ..., a[N-1],p[N-1]

        15 = Complex time signal - amplitude/phase
             a[0],p[0], ..., a[N-1],p[N-1]

        16 = Real part of complex trace from 0 to Nyquist

        17 = Imag part of complex trace from 0 to Nyquist

        18 = Amplitude of complex trace from 0 to Nyquist

        19 = Phase of complex trace from 0 to Nyquist

        21 = Wavenumber time domain (k-t)

        22 = Wavenumber frequency (k-omega)

        23 = Envelope of the complex time trace

        24 = Phase of the complex time trace

        25 = Frequency of the complex time trace

        30 = Depth-Range (z-x) traces

        101 = Seismic data packed to bytes (by supack1)

        102 = Seismic data packed to 2 bytes (by supack2)
        */

short nvs;  /* number of vertically summed traces (see vscode
           in bhed structure) */

short nhs;  /* number of horizontally summed traces (see vscode
           in bhed structure) */

short duse; /* data use:
            1 = production
            2 = test */
```

```
int offset; /* distance from source point to receiver
           group (negative if opposite to direction
           in which the line was shot) */

int gelev;  /* receiver group elevation from sea level
           (above sea level is positive) */

int selev;  /* source elevation from sea level
           (above sea level is positive) */

int sdepth; /* source depth (positive) */

int gdel;   /* datum elevation at receiver group */

int sdel;   /* datum elevation at source */

int swdep;  /* water depth at source */

int gwdep;  /* water depth at receiver group */

short scalel;   /* scale factor for previous 7 entries
           with value plus or minus 10 to the
           power 0, 1, 2, 3, or 4 (if positive,
           multiply, if negative divide) */

short scalco;   /* scale factor for next 4 entries
           with value plus or minus 10 to the
           power 0, 1, 2, 3, or 4 (if positive,
           multiply, if negative divide) */

int  sx;    /* X source coordinate */

int  sy;    /* Y source coordinate */

int  gx;    /* X group coordinate */

int  gy;    /* Y group coordinate */

short counit;   /* coordinate units code:
            for previous four entries
            1 = length (meters or feet)
            2 = seconds of arc (in this case, the
            X values are longitude and the Y values
            are latitude, a positive value designates
            the number of seconds east of Greenwich
            or north of the equator */

short wevel;    /* weathering velocity */

short swevel;   /* subweathering velocity */

short sut;  /* uphole time at source */

short gut;  /* uphole time at receiver group */

short sstat;    /* source static correction */
```

```
short gstat;    /* group static correction */

short tstat;    /* total static applied */

short laga; /* lag time A, time in ms between end of 240-
           byte trace identification header and time
           break, positive if time break occurs after
           end of header, time break is defined as
           the initiation pulse which maybe recorded
           on an auxiliary trace or as otherwise
           specified by the recording system */

short lagb; /* lag time B, time in ms between the time break
           and the initiation time of the energy source,
           may be positive or negative */

short delrt;    /* delay recording time, time in ms between
           initiation time of energy source and time
           when recording of data samples begins
           (for deep water work if recording does not
           start at zero time) */

short muts; /* mute time--start */

short mute; /* mute time--end */

unsigned short ns;  /* number of samples in this trace */

unsigned short dt;  /* sample interval; in micro-seconds */

short gain; /* gain type of field instruments code:
            1 = fixed
            2 = binary
            3 = floating point
            4 ---- N = optional use */

short igc;  /* instrument gain constant */

short igi;  /* instrument early or initial gain */

short corr; /* correlated:
            1 = no
            2 = yes */

short sfs;  /* sweep frequency at start */

short sfe;  /* sweep frequency at end */

short slen; /* sweep length in ms */

short styp; /* sweep type code:
            1 = linear
            2 = cos-squared
            3 = other */

short stas; /* sweep trace length at start in ms */
```

```
short stae; /* sweep trace length at end in ms */

short tatyp;    /* taper type: 1=linear, 2=cos^2, 3=other */

short afilf;    /* alias filter frequency if used */

short afils;    /* alias filter slope */

short nofilf;   /* notch filter frequency if used */

short nofils;   /* notch filter slope */

short lcf;  /* low cut frequency if used */

short hcf;  /* high cut frequncy if used */

short lcs;  /* low cut slope */

short hcs;  /* high cut slope */

short year; /* year data recorded */

short day;  /* day of year */

short hour; /* hour of day (24 hour clock) */

short minute;   /* minute of hour */

short sec;  /* second of minute */

short timbas;   /* time basis code:
            1 = local
            2 = GMT
            3 = other */

short trwf; /* trace weighting factor, defined as 1/2^N
           volts for the least sigificant bit */

short grnors;   /* geophone group number of roll switch
           position one */

short grnofr;   /* geophone group number of trace one within
           original field record */

short grnlof;   /* geophone group number of last trace within
           original field record */

short gaps; /* gap size (total number of groups dropped) */

short otrav;    /* overtravel taper code:
            1 = down (or behind)
            2 = up (or ahead) */

/* local assignments */
float d1;   /* sample spacing for non-seismic data */
```

```
    float f1;   /* first sample location for non-seismic data */

    float d2;   /* sample spacing between traces */

    float f2;   /* first trace location */

    float ungpow;   /* negative of power used for dynamic
               range compression */

    float unscale;  /* reciprocal of scaling factor to normalize
               range */

    int ntr;    /* number of traces */

    short mark; /* mark selected traces */


    short unass[15];    /* unassigned--NOTE: last entry causes
               a break in the word alignment, if we REALLY
               want to maintain 240 bytes, the following
               entry should be an odd number of short/UINT2
               OR do the insertion above the "mark" keyword
               entry */

    float  data[SU_NFLTS];

} segy;


typedef struct {    /* bhed - binary header */

    int jobid;  /* job identification number */

    int lino;   /* line number (only one line per reel) */

    int reno;   /* reel number */

    short ntrpr;    /* number of data traces per record */

        short nart; /* number of auxiliary traces per record */

    unsigned short hdt; /* sample interval in micro secs for this reel */

    unsigned short dto; /* same for original field recording */

    unsigned short hns; /* number of samples per trace for this reel */

    unsigned short nso; /* same for original field recording */

    short format;   /* data sample format code:
                1 = floating point (4 bytes)
                2 = fixed point (4 bytes)
                3 = fixed point (2 bytes)
                4 = fixed point w/gain code (4 bytes) */

    short fold; /* CDP fold expected per CDP ensemble */
```

```
short tsort;    /* trace sorting code:
            1 = as recorded (no sorting)
            2 = CDP ensemble
            3 = single fold continuous profile
            4 = horizontally stacked */

short vscode;   /* vertical sum code:
            1 = no sum
            2 = two sum ...
            N = N sum (N = 32,767) */

short hsfs; /* sweep frequency at start */

short hsfe; /* sweep frequency at end */

short hslen;    /* sweep length (ms) */

short hstyp;    /* sweep type code:
            1 = linear
            2 = parabolic
            3 = exponential
            4 = other */

short schn; /* trace number of sweep channel */

short hstas;    /* sweep trace taper length at start if
           tapered (the taper starts at zero time
           and is effective for this length) */

short hstae;    /* sweep trace taper length at end (the ending
           taper starts at sweep length minus the taper
           length at end) */

short htatyp;   /* sweep trace taper type code:
            1 = linear
            2 = cos-squared
            3 = other */

short hcorr;    /* correlated data traces code:
            1 = no
            2 = yes */

short bgrcv;    /* binary gain recovered code:
            1 = yes
            2 = no */

short rcvm; /* amplitude recovery method code:
            1 = none
            2 = spherical divergence
            3 = AGC
            4 = other */

short mfeet;    /* measurement system code:
            1 = meters
            2 = feet */

short polyt;    /* impulse signal polarity code:
```

```
                1 = increase in pressure or upward
                    geophone case movement gives
                    negative number on tape
                2 = increase in pressure or upward
                    geophone case movement gives
                    positive number on tape */

    short vpol; /* vibratory polarity code:
                code    seismic signal lags pilot by
                1   337.5 to  22.5 degrees
                2    22.5 to  67.5 degrees
                3    67.5 to 112.5 degrees
                4   112.5 to 157.5 degrees
                5   157.5 to 202.5 degrees
                6   202.5 to 247.5 degrees
                7   247.5 to 292.5 degrees
                8   293.5 to 337.5 degrees */

    short hunass[170];  /* unassigned */

} bhed;

/* DEFINES */
#define gettr(x)    fgettr(stdin, (x))
#define vgettr(x)   fvgettr(stdin, (x))
#define puttr(x)    fputtr(stdout, (x))
#define gettra(x, y)    fgettra(stdin, (x), (y))

/* The following refer to the trid field in segy.h      */
/* CHARPACK represents byte packed seismic data from supack1    */
#define     CHARPACK    101
/* SHORTPACK represents 2 byte packed seismic data from supack2 */
#define     SHORTPACK   102

/* TREAL represents real time traces                */
#define     TREAL       1
/* TDEAD represents dead time traces                */
#define     TDEAD       2
/* TDUMMY represents dummy time traces              */
#define     TDUMMY      3
/* TBREAK represents time break traces              */
#define     TBREAK      4
/* UPHOLE represents uphole traces              */
#define     UPHOLE      5
/* SWEEP represents sweep traces                */
#define     SWEEP       6
/* TIMING represents timing traces              */
#define     TIMING      7
/* WBREAK represents timing traces              */
#define     WBREAK      8

/* TCMPLX represents complex time traces            */
#define     TCMPLX      13
/* TAMPH represents time domain data in amplitude/phase form    */
#define     TAMPH       15
/* FPACK represents packed frequency domain data        */
#define     FPACK       12
```

```
/* FUNPACKNYQ represents complex frequency domain data      */
#define     FUNPACKNYQ  11
/* FCMPLX represents complex frequency domain data      */
#define     FCMPLX      10
/* FAMPH represents freq domain data in amplitude/phase form    */
#define     FAMPH       14
/* REALPART represents the real part of a trace to Nyquist  */
#define     REALPART    16
/* IMAGPART represents the real part of a trace to Nyquist  */
#define     IMAGPART    17
/* AMPLITUDE represents the amplitude of a trace to Nyquist */
#define     AMPLITUDE   18
/* PHASE represents the phase of a trace to Nyquist     */
#define     PHASE       19
/* KT represents wavenumber-time domain data            */
#define     KT      21
/* KOMEGA represents wavenumber-frequency domain data       */
#define     KOMEGA      22
/* ENVELOPE represents the envelope of the complex time trace   */
#define     ENVELOPE    23
/* INSTPHASE represents the phase of the complex time trace */
#define     INSTPHASE   24
/* INSTFREQ represents the frequency of the complex time trace  */
#define     INSTFREQ    25
/* DEPTH represents traces in depth-range (z-x)         */
#define     TRID_DEPTH  30

#define ISSEISMIC(id) (( (id)==0 || (id)==TREAL || (id)==TDEAD || (id)==TDUMMY ) ? cwp_true

/* FUNCTION PROTOTYPES */
#ifdef __cplusplus /* if C++, specify external linkage to C functions */
extern "C" {
#endif

int fgettr(FILE *fp, segy *tp);
int fvgettr(FILE *fp, segy *tp);
void fputtr(FILE *fp, segy *tp);
int fgettra(FILE *fp, segy *tp, int itr);

/* hdrpkge */
void gethval(const segy *tp, int index, Value *valp);
void puthval(segy *tp, int index, Value *valp);
void getbhval(const bhed *bhp, int index, Value *valp);
void putbhval(bhed *bhp, int index, Value *valp);
void gethdval(const segy *tp, char *key, Value *valp);
void puthdval(segy *tp, char *key, Value *valp);
char *hdtype(const char *key);
char *getkey(const int index);
int getindex(const char *key);
void swaphval(segy *tp, int index);
void swapbhval(bhed *bhp, int index);
void printheader(const segy *tp);

void tabplot(segy *tp, int itmin, int itmax);

#ifdef __cplusplus /* if C++, end external linkage specification */
}
```

```
#endif

#endif
```

付 録C　デモ - デモの簡単な説明

$CWPROOT/src/demos ディレクトリにはいくつかの下位ディレクトリがあり、SU プログラムを実際に使ってみるための、すぐに実行できるシェルスクリプトがあります。このコレクションはあるべき姿からはまだ未完成で、これからも変化し、更新されます。

- BC Examples
- Block: フリー・パッケージ UNCERT に含まれているプログラム “block” のデモ
- 3D Model Building:
  - Tetramod: 地震波速度モデル構築コード “tetra” のデモ。
- Datuming: データミングの例
  - Finite Difference: 伝統的な有限差分法データミング
  - Kirchhoff Datuming: Kirchhoff データミング
    - Migration with topography: 地形の影響を扱うマイグレーション
- Deconvolution: デコンボリューションのデモ
  - Wiener Levinson: 予測誤差フィルタリング
  - FX: 周波数-波数
- Dip Moveout: ディップ・ムーブアウト
  - Ti: 横方向等方性 (VTI) 媒質の傾斜ムーブアウト
- Diversity Stacking:
- Filtering: フィルタリングのデモ
  - Subfilt: Butterworth フィルタ
  - Sudipfilt: 傾斜フィルタ
  - Sufilter: 多角形のゼロ位相周波数フィルタ
  - Sugabor: Gabor 変換による時間-周波数フィルタ
  - Suk1k2filter: 波数フィルタ（方形）
  - Sukfilter: 波数フィルタ（環状）
  - Sumedian: メディアン・フィルタ
- Making Data: 合成波形プロファイルの作成
  - CommonOffset: 共通オフセットおよびゼロ・オフセットのデータセット
  - ShotGathers: 共通ショットデータセット

- Migration Inversion: マイグレーション（インバージョン）
  - 3D Migration or inversion of 3D datasets
    - Suinvco3d: $v(z)$ 媒質での共通オフセット・マイグレーション
  - Finite Difference: 伝統的な Claerbout の有限差分法 (FD) マイグレーション
  - Fourier Finite Difference: フーリエ FD マイグレーション
  - Gazdag: 伝統的な Gazdag の位相シフト・マイグレーション
  - Kirchhoff: Kirchhoff マイグレーション
  - Sukdmig2d common offset: 共通オフセット・マイグレーション
  - Sukdmig2d common shot: 共通ショット・マイグレーション
  - Phase Shift Plus Interpolation: 位相シフト＋内挿 (PSPI) マイグレーション
  - Split Step: 分割ステップ (Split step) マイグレーション
  - Stolt: Stolt のマイグレーション
  - Suinvvxzco: $v(x, z)$ 媒質での共通オフセット・データセットのインバージョン
  - Sumigpsti: 横方向等方性 (VTI) 媒質での位相シフト・マイグレーション
- NMO: 伝統的な Normal Moveout 補正
- Offset Continuation: より小さいオフセットへのオフセット接続
- Picking: トレースのピッキング
  - Supickamp: 自動化ピッキング
- Plotting: 表示のデモ
  - CurveOnImage: イメージ表示上での曲線の描画
- Refraction: 屈折法のアプリケーション
  - GRM: 2 層構造の一般化相反法 (Generalized Reciprocal Method)
- Residual Moveout: 残差ムーブアウト解析による速度解析
- SUManual: SU ユーザーマニュアル中の図の作成に用いたシェルスクリプト
  - Plotting: 表示のデモ
    - Psmerge: psmerge を用いたポストスクリプト・ファイルのマージ
    - Xmovie: suxmovie を用いた X-windows ムービーの作成
- Selecting Traces: 特定の操作のためのヘッダ値によるトレースの選択
- Sorting Traces: susort を用いたトレースのソート
  - Demo: トレースのソートのデモ
  - Tutorial: トレースのソートのチュートリアル
- Statics: 静的な時間シフト補正
  - Residual Refraction Statics: 残差屈折静補正の適用

  - Residual Statics: 残差震源-観測点静補正

- Synthetic: 合成地震記録の作成
  - Finite Difference: 2D 有限差分法モデリング
  - Impulse: インパルスのテストパターン
  - Kirchhoff: Kirchhoff モデリング
  - Reflectivity: 反射率法モデリング
  - Tetra: 四面体化モデル
  - Tri: 三角化モデル
  - Wkbj: アイコナール方程式の風上有限差分法
  - CSHOT: $CWPROOT/src/Fortran/CSHOT のデモを参照のこと
- Tau P: τ-p あるいは傾斜スタック (slant stack) フィルタ
- Suharlan: Bill Harlan のスラントスタックノイズ抑制アルゴリズム
- Sutaup: 伝統的な τ-p 変換
- Time Freq Analysis: 時間-周波数領域での操作
- Utilities: 様々な（そして実験的な）ユーティリティ
  - Sucommand: プログラム sucommand のデモ
- Velocity Analysis: NMO ベースの速度解析のデモ
- Velocity Profiles: 速度プロファイルの作成
  - Makevel: makevel を用いた速度プロファイルの作成
  - Unif2: unif2 を用いた等間隔にサンプルされた 2D 速度プロファイル
  - Unisam2: unisam2 を用いた等間隔にサンプルされた 2D 速度プロファイル
  - Vel2stiff: vel2stiff を用いた速度プロファイルの剛性率プロファイルへの変換
- Vibroseis Sweeps: suvibro を用いた合成バイブロサイス・スイープ

初心者は以下の順にデモを走らせることを推奨します。

“Making Data” デモは、susynlv を用いて合成記録のショットギャザーや共通オフセット断面を作成する基本を示します。上手な表示のラベル付けの例になるように特に注意が払われています。

“Filtering/Sufilter” デモは、地動（グラウンド・ロール）と初動を除去するいくつかの実際のデータ処理を例示します。“Filtering” の下位ディレクトリ内のデモには CWP の ftp サイトからデータにアクセスするための指示があります。

“Deconvolution” デモでは、簡単な合成スパイク状トレースを用いて、supef や他のツールによる残響除去とスパイキング・デコンを例示します。デモには、ループを用いてフィルタ・パラメータの影響を系統的に試験するコマンドが含まれています。

“Sorting Traces Tutorial” は、この文書で論じられている基本的な UNIX と SU の知識を強化する、会話型のスクリプトです。会話性はペースを設定するためだけに限定されています。このようなチュートリアルにはすぐにイライラさせられますが、標準のデモの形式に適

しないいくつかの項目をカバーするために、こういったデモが一つは必要だと感じました。このトピックには、もう一つ、標準的ですがより未完成のデモがあります。

次のステップは、“Selecting Traces Demo” を実行することです。そして、“NMO” デモに進みましょう。その上で、興味のあるデモのディレクトリを訪れて下さい。**demos** ディレクトリツリーはまだ活発に開発中です— デモが有用か、どのように改良すればよいかを知らせて下さい。

ユーザーが SU のロジックにより慣れるほど、彼/彼女は他のデモを自分自身のやり方で走らせたり、自分自身の目的のためにシェルスクリプトを修正したりすることに抵抗がなくなるでしょう。