Sd

A Square Dance Caller's Helper

William B. Ackerman and Stephen Gildea

Jan 10, 2025

for Sd version 39.71

Table of Contents

Introduction

Sd is a square dance caller's helper. The program assists you in writing sequences for Western square dancing by doing the checker pushing. You tell the program what call you want to call next, and it computes the resulting setup and shows it to you.

Sd is intended to be used to write challenge-level dances where the sequences are often complex and the checker pushing tedious and error-prone. Most challenge callers write out the sequences they will call before they get to the dance, unlike Mainstream callers, who often invent the sequences on the fly from the stage.

What Sd Is Not

Sd knows nothing about timing, body flow, or esthetics.

This program is not appropriate for Traditional square dancing. A good Traditional square dance sequence requires, among other things, extremely accurate control of timing and phrasing. The program is not capable of this.

Since the emphasis here is on checker pushing, you will find various Mainstream staples missing, such as Circle Left, Grand Square, and Do-Si-Do. Since these calls are *zeros*, there is no reason to have them in a checker-pushing program. However, if you wish to write sequences containing calls such as these, you can use the `insert a comment` command to write them into the sequence.

In short, if you are writing a Traditional or Mainstream dance, you may find that this program is not for you.

It is most emphatically not the goal of Sd to present a polished graphical display of the dancers, or to show animation. The goal is to write sequences so that you can create pleasing animation among live people. If you want to enjoy the esthetic visual patterns created by square dance choreography, we recommend that you turn off your computer and go to a dance.

The use of this program is not a good way to learn to call or to improve your calling or dancing skills. In fact, reliance on a computer program to write material could easily make you a worse caller, or interfere with your attempts to improve your calling skills.

The knowledge of calls and concepts that this program can provide is only a tiny part what you need to learn in order to be a good caller or dancer. To call successfully, you need to master many skills, such as timing, flow, judging difficulty, floor interaction, and choosing precisely the right words to say. This last skill is as important at high levels as at low levels. Subtle differences in the words you choose, and their timing and inflection, can have a tremendous influence on the success of the dancers. Sd attempts to print the correct words in all cases, but you will only succeed if you have a deep understanding of what the words mean and how they should be delivered. That understanding can only be obtained through a great deal of experience calling at that level.

If you do not have a good understanding of a call or concept, such that you could explain it to a dancer after the tip is over, you should not use it. Do not rely on Sd to provide the necessary insight into challenge dancing.

For beginning callers in particular, the best thing to do is to receive instruction from a qualified teacher or coach, and to practice.

We recognize that people sometimes have to learn a level in a tape group without benefit of any human expertise at that level, and that such people may have no choice but to rely on a computer program as one of their sources of information. This is an undesirable situation, and we believe that computer programs should not be used as references except in emergencies. There is an enormous body of knowledge about the accepted usage of various calls and concepts. That body of knowledge is generally possessed by all competent dancers and callers at a given level. No computer program can possibly possess that knowledge. In particular, computer programs should not be used for resolving controversial issues.

This is not to say that computers have no place in the education and training of callers. A number of caller training programs have been written that may help you develop such skills as formation management and sight resolving. Sd is not suitable for this.

Quality and Correctness

Quality, correctness, and reliability are fundamental and extremely important design objectives of Sd. Before any version is released, it passes very rigorous diagnostic tests. These tests include verification of large amounts of C4 material from recent National Advanced and Challenge Conventions, and other C4 weekends. The elusive but ever-sought-after goal is to make a program that

- never makes mistakes,
- is able to write virtually all of the non-bogus material called by the top challenge callers, whatever computer program they may have used, and
- is developed and tested carefully, so users can be assured that this ability will be maintained in all future releases.

Authenticity of Calls and Concepts

We have endeavored to use the current Callerlab lists and definitions as the source for call and concept names and definitions wherever possible. Where this is not possible, either because Callerlab does not publish lists or definitions at all levels, or because a definition is unclear in some area, we have used other well-respected encyclopedias, along with our best attempts to make things clear and sensible. Particularly at extremely high challenge levels, where there are no standardization bodies to rein in callers' tendencies to change the definitions or usage of calls, concepts, and fundamental assumptions, it is not always possible to do this to everyone's satisfaction.

Spelling variations that would be insignificant in normal calling become a serious problem for computer programs. We have attempted to use the spelling in official lists where possible, but sometimes even these lists are careless. In such cases, we have attempted to correct the errors where possible, using a variety of sources. Priority is given to those sources that have shown the greatest care in their editing.

Deviations from Official Lists

It is a known fact that challenge callers routinely call various "popular" calls that are not on the official or semi-official lists. In an attempt to allow this, while maintaining the

appearance of strict compliance with the lists, Sd has two special levels `c3x` and `c4x`. These contain the calls that are currently believed to be called at C3 and C4 but are not yet on the semi-official C3 list or the various informal C4 compilations of "commonly used" calls.

It is generally *not necessary* to run the program at either of these levels. When you run at C3, C3X calls may be used. When you run at C4, C4X calls may be used. Whenever such a call is used, a warning is printed. (If you explicitly run the program at C3X or C4X, you may use the calls without getting a warning.)

Misuse of Computers

The phenomenon of "clueless clicking", that is, using a computer to generate challenge choreography that one doesn't really understand, is by now well-known. There are very few more effective ways to exhibit your ignorance and incompetence as a caller. High level challenge dancers are extremely sophisticated in their ability to detect this.

If dancers didn't understand what you meant by something that you called, they may ask you about it after the tip is over. The answer "I meant whatever the computer did" is *never* an acceptable answer to such a question. If you can't give an answer in terms of your understanding of how the calls and concepts really work, or if you don't agree with the program's interpretation, you shouldn't be calling that material.

Judgement, Controversy, and Warning Messages

Not all callers exercise good judgement in their calling, and occasionally some controversial, or even illegal, things are called. It is not possible for a computer program to enforce good judgement or good taste. Sd nevertheless sometimes prints warning messages of various types to alert less-experienced callers that something might be unusually difficult or controversial, or might violate some definition or some commonly accepted notion.

These warning messages are intended for callers less competent than you. Ignore them.

You can prevent the program from displaying or printing these warning messages by giving the `toggle nowarn mode` command. This turns on (or turns off if it was already on) the "no warnings" mode. See Section 8.9 [Changing Modes], page 36. You can also place a line "`no_warnings`" in the "`options`" section of the '`sd.ini`' initialization file (see Section 9.2 [Option Control], page 40) or start the program with a command-line option "`-no_warnings`" (note the leading hyphen). See Section 10.1 [Command-Line Options], page 47.

Variations of the Program

Several user interfaces are available. They come in two general types, and the program name is different for these types.

`Sdtty` is the name of the program that uses a character-oriented keyboard interface. Calls are selected by typing their names. `Sdtty` runs on Unix-like systems and under DOS, Windows 3.1, Windows 95, Windows 98, Windows NT, Windows XP, and Windows 7 on PC's.

`Sd` is the name of the program that uses menus and a *graphical user interface*. Calls may be selected by clicking with the mouse on the chosen menu item. You may also type calls in from the keyboard. In fact, `Sd` is keyboard-compatible with `Sdtty`—you can use it almost exactly the same way. `Sd` runs on Windows 95, Windows 98, Windows NT, Windows XP, and Windows 7 on PC's. A somewhat different version ran on Unix systems running the raw X Window System interface, though this has not been maintained in recent years.

Getting Sd or Sdtty

The most straightforward way to obtain `Sd` or `Sdtty` is to download the program from `http://www.lynette.org/sd/` on the World Wide Web. The reference manual and other documentation may be browsed there or downloaded for your local use as well.

You can obtain Sdtty for PC-compatible computers, on a 3.5-inch HD diskette, with printed documentation. The price is $4 (price valid for 2000). See [Author Contact], page 7.

Documentation

If you are on the Internet, you can browse or download `Sd`/`Sdtty` documentation from `http://www.lynette.org/sd/readings.html` on the World Wide Web. The documentation is the same for the two programs.

Documentation files are provided in several formats. There is '`html`' format for browsing, Postscript format for downloading and printing, '`PDF`' (Acrobat) format for browsing or printing, and plain text format for downloading. You can also download "gzipped" Unix archives of the various files in Unix format.

The file '`sd_doc`' is this manual.

See the file '`relnotes`' for information about what is new in the most recent release.

There are also some documents describing a few special features in more detail.

Printed copies of the manual are available from the authors, or may be printed from the file '`sd_doc.ps`'.

Software Safety

The computer on which `Sd` and `Sdtty` and their documentation are developed has no Internet or modem connections at all. No mail is sent or received, and no web browsing is done, on this computer. Only software from trustworthy sources is installed. Each version of the program is digitally signed, using a 512-bit secret PGP/RSA key, and then transferred to another computer by Zip disk. The files are then mailed to the web server. Before placing it on the web, the server authenticates the signature and verifies, with the PGP signature algorithm, that the files were not tampered with in transit.

Prior to May, 2000, the computer from which the web updates take place had Microsoft Outlook installed, but not used. (Netscape Communicator is used as my mailer, because of its technical and ethical superiority. Outlook had been installed only because of the policy of my employer.)

The "Love Bug" virus of May, 2000 demonstrated that Microsoft is unable or unwilling to make its email software adhere to the most basic common-sense principles of safety. Accordingly, Outlook and the Visual Basic scripting / virus-propagating mechanism (wscript.exe) have been removed from that computer. No Microsoft email or virus-propagating products exist anywhere on the `Sd` development path.

Various terms in the preceding paragraphs are trademarks of various corporations.

A Few Technical Details

Although it is a "Windows application", `Sdtty` does not use the mouse while running. It looks rather like the Command Prompt window.

`Sd` runs only on Windows 95, Windows 98, and NT 4.0.

Users of DOS, Windows 3.1, or NT 3.51 can only run `Sdtty`, and need to run a special version, identified by the installation file being '`install3.exe`' instead of '`install.exe`'. (If you use DOS, Windows 3.1, or NT 3.51, be sure to download the correct version of the program from the web, or request the correct version when ordering it on diskette.)

`Sd` and `Sdtty` are identical in their behavior with respect to square dance calling.

`Sd` and `Sdtty` are believed to be insensitive to slight inaccuracies in floating-point division. It should operate correctly with early Pentium chips.

The program is mostly "Year 2000 Compliant", but some care will be needed. File names created with the special names `*` and `+` contain only the last two digits of the year. We therefore recommend that you delete old files at least every 75 years or so.

The date printed at the top of each sequence contains the full 4-digit year, so there should be little danger of calling a 100-year-old card without realizing it. Callers are nevertheless urged to be careful, since some dancers have extraordinarily long memories.

Installing on a PC

`Sd` and `Sdtty` are two separate programs, that are both installed simultaneously. Whether you obtain them from the World Wide Web or by diskette, they are provided in the file '`install.exe`'. The two programs will share the same initialization file, and hence will use the same sessions, options, and accelerator keys.

> (If you are running DOS, Windows 3.1, or NT 3.51, you can only use `Sdtty`, and you must obtain the file '`install3.exe`'.)

To install the programs, execute ("launch") the file '`install.exe`'. You can do this by double-clicking it in the Windows Explorer, by launching it during a Web download, or by starting the "Run" operation from the start menu and giving the location of the '`install.exe`' file. For example, you can install from a diskette by selecting "Run" and typing "A:install.exe".

If any pre-existing calling program files were in the '`C:\Sd`' folder, they will be saved in a folder with a name like '`C:\Sd_pre3281`' before the new version is installed. The installation will not damage or delete any sequence transcript files or initialization files.

If you have downloaded the documentation files in "bulk" form in self-extracting zipped archives, you can launch them also. The files are '`textdoc.exe`', '`psdoc.exe`', and

‘pdfdoc.exe’. Launch them in the same way as ‘install.exe’. They will unpack the appropriate text, Postscript, or PDF files into the directory.

When the installation is complete, the programs will be in the ‘Sd’ subfolder of the ‘Programs’ folder of the Start Menu. The installation procedure also leaves an Explorer open on the shortcut icons. There are several icons for starting Sd or Sdtty with various color schemes, for browsing the manual or release notes, and for editing the initialization file. You may copy any or all of these from the open Explorer to the Desktop and/or the Start Menu. To do this, perform a drag-and-drop operation, using the right mouse button. If the system asks, select “Copy Here”. To place icons in on the Desktop, just perform a right-mouse drag-and-drop to any unused place on the Desktop. To place icons in the Start Menu, just drop them on the actual “Start” button in the lower left corner. When finished, close the Explorer by clicking the “X” in the upper-right corner.

> (If you are running DOS, Windows 3.1, or NT 3.51, you must place the ‘install3.exe’ in the folder in which you want Sdtty to operate (typically ‘C:\sd’), and launch it there. You can create a shortcut icon by copying the file ‘SDTTY.PIF’ to the appropriate place. Use the PIF editor to modify it if necessary.)

You may delete the ‘install.exe’ file at this point, along with any documentaion archives, perhaps saving copies on diskette.

You do not need to copy the shortcuts again when upgrading to a new version. Installing a new version requires only obtaining and launching ‘install.exe’.

You can, of course, copy the shortcuts anywhere, and edit them with the “Properties” operation of the Windows Explorer. Among the things you can do is put command-line arguments into them. For example, you can make another copy of the Sd icon on the Desktop (under a different name), and arrange for the program, when launched with that icon, to run in reverse video mode. Run the “Properties” operation on the copied icon, and change the ‘Target’ specification on the ‘Shortcut’ page from “C:\sd\sd.exe” to “C:\sd\sd.exe -reverse_video”.

Licensing Terms

Sd and Sdtty are “free software” programs, licensed under the GNU General Public License, which is in the file ‘COPYING.txt’ in the distribution. You can also get it by writing to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

In short, this means that it is perfectly legal and ethical to copy and redistribute the program, documentation, and source files. You are, of course, not required to download the source files when you obtain a copy of Sd from the web page. If you want to make the program available to someone else, and you do it by giving them the URL for downloading, that is OK. However, if you distribute the program by other means, such as giving someone a copy that you have placed on a CD or diskette, you must either place the source files on the disk also, or tell the person the URL (http://www.lynette.org/sd/) from which they can download them.

Also, if you redistribute Sd on your own web page or similar mechanism, you must make the source files available there.

If you modify the program, the terms of the license are very strict. Programs constructed from modified source files may never be duplicated or redistributed by anyone, even just in executable form, without obeying the conditions of the license and copyright. Read the license carefully.

Note in particular that any modified program based on `Sd` or `Sdtty` must obey the licensing terms every time it is redistributed by anyone. This requires redistribution of all of the source files, or otherwise complying with section 3 of the license.

The Windows installation program '`install.exe`' contains licensed commercial decompression software. The licensing terms prohibit extracting or reverse-engineering that software.

This document is distributed in both formatted and plain-text form, in the files '`sd_doc.ps`' and '`sd_doc.txt`', respectively. Some other documents are also distributed with the program. Permission is given to distribute verbatim copies of these documents, either by photocopier or by computer, but modification of them is forbidden by copyright law.

How to Contact the Authors

Please send any praise, bug reports, or other comments on Sd to the authors at the addresses given below. Be sure to include the version number.

William Ackerman
wba@alum.mit.edu

Stephen Gildea
gildea@.stop.mail-abuse.org

If you do not have access to electronic mail, write to

Bill Ackerman
100 Parlmont Park
North Billerica MA 01862-2722

Acknowledgements

Thanks to Kathy Godfrey, Sue Curtis, Chris Stacy, and Lynette Bellini for their help, encouragement, suggestions, contributions of calls, feedback, testing, and many other contributions.

The non-windowed interface for `Sdtty` (see Chapter 11 [Terminal Interface], page 49) was contributed by Alan Snyder.

The `Sd` interface for Windows was contributed by Robert Cays.

We received helpful feedback and testing from Robert French, Judy Anderson, Bill van Melle, Tom Rinker, Lois Lew, Ron Nicholson, Dave Decot, Kristin Jensen, Larry Dunn, and Nick Martellacci.

1 Getting Started

Throughout this manual, the name `Sd` will generally be used to denote both `Sd` and `Sdtty`, except in those areas in which the programs are different.

If you are using a PC and have installed `Sdtty` to run under Windows 3.1 or Windows 95, start the program in the usual way by clicking its icon. `Sdtty` is actually a DOS program, so, once it is started, it will not use the mouse.

The first thing it will do (assuming you are not using the session control feature of the initialization file) is ask you for the level. Type the desired level and press ⟨ENTER⟩.

`Sd` or `Sdtty` can be run from a shell (on Unix or DOS) by typing its name and the level you want to write sequences for, e.g.,

```
sd c2
```

or

```
sdtty c2
```

In addition to calls and concepts, there are program commands for actions such as automatic resolves, exiting the program, and other miscellaneous actions. See Chapter 5 [Resolving and Searching], page 26. In `Sd`, they are found in the menu in the upper left corner. In `Sdtty`, just type the name of the command.

The following chapters cover the operation of `Sd` in detail.

2 Calling

This chapter describes how to use the program in its normal mode to generate a sequence of calls of your choosing.

2.1 Entering Your Choice

Selections are made in the same way almost everywhere in the program. In either `Sd` or `Sdtty`, you may type in the selection. The program has a sophisticated *completing reader* that lets you abbreviate what you type. It is discussed in detail in Chapter 3 [The Completing Reader], page 21.

In `Sd`, you may additionally select an item from the menu in the usual ways. You may double-click it, or highlight it (by using the cursor keys, for example) and click the '`Accept`' button. (You may activate things with a single mouse click, if you prefer that style of operation, by issuing the `toggle singleclick mode` command or by placing the `single_click` option in your initialization file.)

You can see the choices that are available by typing a question mark or exclamation point. This is discussed in detail in Section 3.1 [Question Mark], page 21.

In normal operation, the menu lists calls first, concepts next, and miscellaneous commands last. This menu is typically extremely large and needs to be manipulated with cursor keys or the scroll bar.

2.2 Starting a Sequence

A special "startup" menu is displayed when the program starts, and after each sequence has been completed and written to a file. It provides a number of miscellaneous commands, including commands to change various modes such as the `active phantoms` mode or the `singing call` mode. See Chapter 8 [Miscellaneous Commands], page 33, for further discussion of these. The startup menu also allows you to `exit from the program` and return to your computer's operating system.

The main use of this menu is to select the starting operation.

2.2.1 Heads/Sides Start

If you select `heads start` or `sides start`, the sequence will begin with '`heads`' or '`sides`' and the first call. Normally this means that the designated people will move into the center and do the first call while the others wait. Subsequent calls will be directed to everyone. For example, one might select `heads start`, and then `star thru`, and then `double pass thru`, and so on. You can also select `heads start` or `sides start` followed by a call such as *split square thru* or *split dixie style to a wave*.

If you want to have the heads go into the middle and do several calls before the sides join in, as in *head ladies chain and then heads square thru 4*, or *heads touch 1/4 and then walk and dodge*, you must select `centers` before all calls after the initial one. Typically, you won't need to read the card that way, but can rely on precise phrasing and inflection of your voice to express what you want.

2.2.2 1P2P Lines

The `heads 1P2P` and `sides 1P2P` starting actions refer to caller jargon for a common starting maneuver, typified by <*ANYONE*> *lead right and circle to a line*. You can read the card that way, or as *wheel thru and circle to a line*, or as *step right*, or as *bring us together*, or whatever improvisation you like. The promenade distance shown at the end of the sequence will assume that you said *step right* or *lead right* (or *wheel thru*) *and circle to a line*.

Whether you consider 1P2P openings to be overused is up to you.

2.2.3 Just As They Are

The `just as they are` selection starts the sequence on squared set spots, without having the heads or sides begin. This is generally useful only if the *all 4 couples* concept is to be used. After using this, you typically need to have the heads or sides (or headliners or sideliners, in some cases) press ahead, in order to continue the sequence.

2.3 Entering Calls

The usual thing you do when writing a sequence is to select calls, perhaps preceded by one or more concepts. You can select concepts and calls separately, or, if using the completing reader, type concepts and calls together. That is, you can type the entire "line".

You should be aware that a number of *modifier* words like `cross`, `left`, and `grand` are treated as *concepts* by `Sd` and `Sdtty`. While concepts don't officially exist below the Advanced program, `Sd` and `Sdtty` consider words such as these to be concepts. When typing things in, this generally makes no difference. Because of the way the program operates, you can just type the call, with whatever concepts or modifier words it has, in the natural way. For example, you could type:

grand swing thru ⟨ENTER⟩

or:

centers cross run ⟨ENTER⟩

In a few cases, either because a concept has a complex structure (e.g., `checkpoint`), or because an utterance is ambiguous (e.g., doing 1/2 of a *stable swing thru* vs. doing a *1/2 stable swing thru*), it is necessary to type concepts by themselves, pressing ⟨ENTER⟩ after each one. This will be discussed further in Section 2.4 [Entering Concepts], page 16.

When using `Sd` and selecting things with the mouse, you will usually not see full combinations listed in the menu. For example, there is no item `grand swing thru`. The menu would be too unwieldy if such things were listed separately. Instead, all concepts, modifier words, and calls are listed separately in the menus. To get a *grand swing thru* with mouse clicks, click on `grand` and then `swing thru`.

2.3.1 Call Variations

There are a number of issues that make call entry less than completely straightforward. These have to do with variations that calls have—directions (as in `quarter left`), numbers

(as in square thru 3 or three quarter thru), person designators (as in sides kickoff), subcalls (as in clover and [quarter thru] or vertical tag your neighbor or shuttle [rally] or in roll motivate), and modifiers (as in strut left or trans cross chain reaction).

To list all possible variations in the menu would be unwieldy for a variety of reasons. Because of this, various methods are used to shorten the menu and to allow the user to specify the desired variation. These methods will be discussed in detail presently, but in most cases what happens is the following:

In Sd, the menu lists only the base calls with special keywords as in face <DIRECTION> or square thru <N>. When you click on such a menu item, a new menu appears listing the possible choices.

In Sdtty, the "menu" (that is, what you see if you type a question mark) also has keywords. You can type the item just as shown, e.g., quarter <DIRECTION> or square thru <N> (you would actually type the angle brackets) and then answer the question that Sdtty asks. But there is an easier way—you can type what you want directly. That is, just type quarter left or square thru 3. If you want to see the choices, type quarter <DIRECTION> or whatever, and then type a question mark.

Directions

Calls that take a direction have the keyword <DIRECTION> in them in the menu, like pass <DIRECTION> or spin a windmill, outsides <DIRECTION>.

You can just type the call naturally, for example, pass in or shuttle left. You can also type such things in a piecemeal fashion, typing the keyword "<DIRECTION>" literally, and then typing the direction.

When using the mouse with Sd, the second way is the only way—click on the call, and then select the direction from the new menu that will appear. If you decide you don't want that call, click the 'Cancel' button.

The menu lists all possible directions, including some that may not be legal in the given context.

Numbers

Calls that take a number have the keyword <N>, <N/4>, or <Nth> in their name. Examples are the calls eight chain <N>, invert the column <N/4>, and square thru, but on the <Nth> hand <ANYTHING>.

You can just type the call naturally, for example, invert the column 3/4 or square thru 2 or 1/4 thru or square thru but on the 3rd hand. You can also type such things in a piecemeal fashion, typing the keyword "<N>" or "<N/4>" or "<Nth>" literally, and then typing the number(s). If the keyword was <N/4>, enter just the value of N.

When using the mouse with Sd, the second way is the only way—click on the call, and then select the number from the new menu that will appear. If you decide you don't want that call, click the 'Cancel' button.

Not all values of N will be meaningful for all calls. For example, you can eight chain 5, but you can't invert the column 5/4.

Some calls have the words quarter, half, or three quarter in their names. Examples are quarter thru, quarter the deucey, quarter mix, quarter the alter, and quarter chain and circulate in. These are listed in the Callerlab lists with the words spelled out, though individual preferences vary. Some people prefer to write 1/4 thru and 3/4 thru. Sd and Sdtty list the call as <N/4> thru in the menu. When using mouse input in Sd, you click on that and then click on a number (only 1 and 3 are legal, of course). You can also type <N/4> thru and then type 1 or 3, or you can type the call directly, either in numbers or in words. That is, you can type either 1/4 thru or quarter thru. No matter how you enter one of these calls, it will always be printed out in words, as in three quarter thru or half chain and circulate in.

Person designators

These work like the others in a straightforward way. The calls appear in the menu with the <ANYONE> keyword, as in patch the <ANYONE>.

You can just type the call naturally, for example, patch the sides. You can also type such things in a piecemeal fashion, typing the keyword "<ANYONE>" literally, and then typing the designator.

When using the mouse with Sd, the second way is the only way—click on the call, and then select the designator from the new menu that will appear. If you decide you don't want that call, click the 'Cancel' button.

The gender designators are boys and girls. This is the way they appear in the Callerlab Mainstream list. The author does not take a position on whether the adult terms would be more reasonable words to use in any given context. You must use your own judgement in deciding what to say.

Not all designators are legal in all cases; for example, you can't call leads run from a tidal wave.

The program specifically refuses to recognize the meaning of the designators centers and ends while in a 1x8 setup (e.g., a tidal wave). This is because these terms can be ambiguous in such a setup. If you want the centers of each 1x4, you must use the EACH 1X4, EACH LINE, EACH COLUMN, or EACH WAVE concept in order to make the centers or ends designators work. Of course, in many cases, other designators, such as boys or girls can identify the same people unambiguously. If you want to designate the 4 people in the center of the set, use center 4. Outer pairs specifies the others.

Tagging calls

Calls that use tagging calls have the keyword <ATC> (for Any Tagging Call) in them, as in <ATC> your neighbor or <ATC> chain thru and scatter reaction. Specify these in the usual way. In Sdtty you can type the <ATC> directly or you can enter the complete call.

At C3 and above, the tagging calls include the calls revert <ATC> and reflected <ATC>, which make use of another tagging call. When using mouse input in Sd, you will be presented with another menu. In Sdtty, you can type in the entire call directly, as in revert cross flip chain thru reactivate.

Subcalls

Some calls have an "<ANYTHING>" in them when they appear in the menu, such as `clover and <ANYTHING>` or `<ANYTHING> and roll`. The "<ANYTHING>" is intended to be replaced by a subcall. When you type these in, put the subcall in brackets. That is, you literally type something like:

```
clover and [tandem shazam]
[flip the diamond] and roll
busy [lockit]
catch [erase] 2
slant [swing thru] and [turn and deal]
fascinating [ah so]
[jay walk] and plenty, turn the star 3/4, interrupt
            before the star turns with [trade circulate]
```

The subcall in brackets may be any combination of concepts and calls, and these may be nested. As an example of this, you can type:

```
[[vertical tag your neighbor] and spread] er's percolate
clover and [3/4 stable left catch
            [[reflected flip your neighbor] and spread] 3]
```

(Two of the examples above don't fit on one line in this manual, but you would type them on one line.)

The brackets are required. Without them, the program couldn't tell the difference between

```
clover and [[swap around] and roll]
```

and

```
[clover and [swap around]] and roll
```

or between

```
first 1/2 stable swing thru
```

and

```
first [1/2 stable swing thru]
```

You can also enter such things in a piecemeal fashion, typing the keyword "<ANYTHING>" literally, and then typing the subcall when the program asks for it. For example, you could type

clover and <anything> ⟨ENTER⟩

and then type

square thru 2 ⟨ENTER⟩.

When using the mouse with `Sd`, the second way is the only way—click on the call, and then select the subcall.

Important note: Nesting these subcalls too deeply can make the program run very slowly. You should not type more than two consecutive left brackets. If you need to nest things more deeply than that, use the word "<ANYTHING>", and enter the subcall when the program asks for it. For example, suppose we wanted to enter:

```
[[[trans cross reactivate to a diamond] chain thru] and
        anything] percolate, boys to a wave
```

(Berkshires C4 weekend, May 1996.) (As if that weren't complicated enough, note that the word "anything" in the above call is the literal "anything" concept, not the `<ANYTHING>` mechanism of `Sdtty`.)

This is too complicated to type directly. We would instead type:

```
<anything> percolate, boys to a wave
```

The subcall we want is now `[[trans cross reactivate to a diamond] chain thru] and anything`. When asked for the subcall, we type:

```
<anything> and anything
```

The subcall we want is now `[trans cross reactivate to a diamond] chain thru`. When asked for the subcall, we could type that directly, or we could type:

```
<anything> chain thru
```

and then type:

```
trans cross reactivate to a diamond
```

These "mandatory" subcalls, with the keyword `<ANYTHING>` in the menu, are just one of many types of modifications. See Section 2.6 [Call Modifications], page 18, for some other types of modifications.

Calls with circulate replacements

A number of calls, all of whose definitions start with a circulate, are subject to a special circulate modification mechanism, recognized at C2, and officially called the "(anything)" concept. The call name (e.g. "`motivate`") is preceded by a word or phrase such as "`in roll`" telling how the circulate should be modified. Examples are

```
in roll motivate
bias trade perk up
split counter coordinate      (not actually a type of circulate)
counter cover up              (not actually a type of circulate)
```

These variations are in the menu as "`<ANYCIRC> motivate`". They can be typed in just as shown above. You can also enter them in a piecemeal fashion, typing the keyword "`<ANYCIRC>`" literally, and then typing the circulate replacement as a complete call, for example, "`out roll circulate`".

When using the mouse with `Sd`, the second way is the only way—click on the call, and then select the circulate replacement from the new menu that will appear. If you decide you don't want that call, click the '`Cancel`' button.

Only bona-fide circulate-type calls, and other well-recognized similar things such as `split counter percolate` can be obtained by this method. You can get more general substitutions by using the '`<ANYTHING>`' (as opposed to '`<ANYCIRC>`') mechanism. The more general substitutions are in brackets. Such things will print out with the word "er's" to distinguish them. You can type in the "er's" or not, as you choose. The apostrophe is optional.

For example, you can type '`[2/3 recycle] er's percolate`' or '`[2/3 recycle] percolate`'.

Note that

`trade motivate` (an instance of `<ANYCIRC> motivate`)

and

`[trade] er's motivate` (an instance of `<ANYTHING> motivate`)

are very different.

There is a third way to enter such calls. By giving the '`simple modifications`' or '`allow modifications`' command and then just entering the call (e.g. '`motivate`'), the program will ask you for replacements. These replacements may be anything and will appear in brackets in the final transcript. This is discussed in Section 2.6 [Call Modifications], page 18.

Modifiers such as cross, left, split, grand, single, magic, and interlocked

As discussed above, the program considers these to be actual concepts, even though they aren't really.

Some calls use these concepts in ways that have a tricky word order.

Whether the word order is tricky or not, you can always enter the concept followed by the call as separate items. For example, in `Sd`, you can click on `left` followed by `chase right`. In `Sdtty`, you can type in `left` and then `chase right`.

In addition, `Sdtty` allows more natural text entry. You can type in `grand swing thru` or `chase left` directly.

Other examples of this phenomenon are

```
switch to an interlocked diamond
unwrap the magic diamonds
hang a left
scoot and cross ramble
trans cross reactivate
revert cross flip chain thru cross nuclear reaction
```

No matter how you entered it, the calls will be printed out with the words in the correct place.

The foregoing only applies to calls in which the word is an actual concept or modifier. Some calls happen to have the word `cross` as part of their names. Examples are: `cross and wheel`, `cross and turn`, `cross your neighbor`, and `crossfire`. Such calls appear in the menu just as they are spelled.

2.3.2 Call Menus

The menu in `Sd`, or the list of available calls in `Sdtty`, changes according to the current setup. For example, if the current setup is right-hand waves, the menu will contain only those calls that the program believes might be legal from right-hand waves. This determination is approximate but conservative—appearance on the menu does not necessarily mean it is legal, though absence from the menu means that the program is fairly certain that it can't be legal. There is a *universal* menu that appears when the setup is not one of the common

ones, or when anything complex is going on. This contains every call that is in the database for the chosen level.

The call menu is alphabetized in a way that ignores blanks and hyphens. Special keywords such as `<N>`, `<ANYONE>`, `<DIRECTION>`, or `<ANYTHING>` are listed after letters. Hence, `<ANYONE> run` and `<ATC> your neighbor` will be found near the end of the menu.

2.4 Entering Concepts

A call may be preceded by concepts, which modify the action of the call. These may be nested (*stacked*) to any reasonable depth.

When using the completing reader, just type the concept you want. The completing reader will operate in the usual way. Nearly all concepts may be entered in a natural way, on the same line as the call that they affect. As discussed previously, this means that you can type things like

grand swing thru ⟨ENTER⟩

or

boys 1/2 stable split the difference ⟨ENTER⟩

or

random tandem swing thru ⟨ENTER⟩

Whether you typed concepts separately or together, the result is the same, and the output file will look the same. For example, the following are all legal and equivalent.

reverse random tandem swing thru ⟨ENTER⟩

reverse random ⟨ENTER⟩
tandem swing thru ⟨ENTER⟩

reverse ⟨ENTER⟩
random ⟨ENTER⟩
tandem ⟨ENTER⟩
swing thru ⟨ENTER⟩

In the output file, they will all be shown as `reverse random tandem swing thru`.

Occasionally some ambiguous situations can arise. What if you wanted the boys to do 1/2 of a *stable split the difference*? The phrase `boys 1/2 stable split the difference` is ambiguous. Whenever `Sd` is confronted with an ambiguity, it chooses the option that nests concepts least deeply. A single application of the *1/2 stable* concept is less deep (simpler) than an application of the *1/2* concept followed by an application of the *stable* concept, so it chooses the former. If you really want to tell the program to do the latter, you must make clear that you mean the *1/2* concept by itself. The way to do this is to enter each concept separately, pressing ⟨ENTER⟩ after each one. That is, type

boys ⟨ENTER⟩
1/2 ⟨ENTER⟩
stable ⟨ENTER⟩
split the difference ⟨ENTER⟩

When using the mouse with `Sd`, you must click on exactly the concepts that you want. To get the `boys` concept, you must click on `<ANYONE>` and then make the appropriate selection.

To get the `1/2` concept, you must click on `<N>/<N>` and then make the appropriate selections. If you want the *1/2 stable* concept instead, click on `<N/4> stable`, and then select `2`.

When `Sd` resolves an ambiguity, it shows how it did so through the judicious use of commas. For example, the preceding operation will be printed as `boys, 1/2, stable, split the difference`, as opposed to `boys, 1/2 stable, split the difference`. Do not type the commas in.

Whenever you run into trouble typing concepts, try typing each one separately, pressing ⟨ENTER⟩ after each one.

Concepts displayed after the call

The concepts `twice`, `<N> times`, and `1-<N>/<N>` will be displayed and printed after the call rather than before it, unless doing so would be ambiguous. **You still type the concept before the call.** For example, you type `1-1/2 swing thru`. The result will appear as `swing thru 1-1/2`. The program will sometimes use parentheses to prevent ambiguity.

Concepts that operate on two calls

Some concepts require two calls (e.g., `checkpoint` and `interlace`). *You must enter such concepts by themselves.* That is, you must press ⟨ENTER⟩ after typing any of these concepts. You can't type

checkpoint ah so by recycle ⟨ENTER⟩

nor

boys trade (while the others) u-turn back ⟨ENTER⟩

You must type

checkpoint ⟨ENTER⟩
ah so ⟨ENTER⟩
recycle ⟨ENTER⟩

or

boys (while the others) ⟨ENTER⟩
trade ⟨ENTER⟩
u-turn back ⟨ENTER⟩

After choosing such a concept, enter the first call, preceded by whatever concepts apply to it. The program will then prompt you for the second call. A complex tree of concepts and calls can thus be constructed.

Some concepts require a numeric designator (e.g., `interrupt after the 3rd part`) or a people designator (e.g., `girls are stable`). The handling is the same as for calls that require these.

The program recognizes the level at which concepts are legal, but lets you override this if you wish. The `toggle concept levels` command toggles (turns on or off) the state of off-level concept permission. See Section 8.9 [Changing Modes], page 36, and Chapter 13 [Using Off-Level Concepts], page 53.

When you select any concept, the universal call menu replaces whatever special call menu may have been presented, since the set of legal calls becomes highly unpredictable.

2.5 Supercalls

Thinking in mathematical or computer terms, a concept is a "function" that operates on an "argument". That argument is a call, or perhaps another application of a concept to another argument. For example, in '`tandem lockit`', the function '`tandem`' applies to the argument '`lockit`'.

Calls of the '`clover and [anything]`' variety behave the same way. This call takes an "argument" call, and has the centers do that while the outsides cloverleaf. Such a thing is called a "supercall". It could just as easily have been considered a concept.

The significance of this is that some concepts, called "meta-concepts", operate on *concepts* rather than on calls or concept-call combinations. Examples of meta-concepts are '`random`', '`initially`', '`finally`', and '`echo`'. Since supercalls are like concepts, meta-concepts can operate on them.

What would happen if we applied the '`finally`' concept to the supercall combination '`clover and [right and left thru]`'? We do the '`right and left thru`' without the "concept" up until the last part. The "concept" is '`clover and [anything]`'. If we skip the concept, we just do the '`right and left thru`' up to the last part. That is, we do a right pull by. Then, for the last part, which is a courtesy turn, we apply the supercall. That is, we do a '`clover and [courtesy turn]`'. So '`finally clover and [right and left thru]`' is danced as:

```
right pull by
clover and [courtesy turn]
```

`Sd` can handle straightforward cases of supercalls with meta-concepts. You must enter the supercall in the square bracket notation. That is, you may not use the method of typing `<anything>` and expecting to type in the second call later.

Because the search mechanism (`pick random call`, etc.) does not fill in subcalls, it will not find applications of meta-concepts and supercalls. Sorry.

Some supercalls take their "argument" call at the beginning of the phrase rather than the end, as in '`[anything] and roll`'. Such supercalls don't scan very tastefully. For example, '`echo [hinge] and roll`' means that the "and roll" is applied first, even though it is at the end of the phrase. We do a '`[hinge] and roll`' first, and then just a '`hinge`'. The word order doesn't make that clear.

Supercalls in which the '`[anything]`' call is at the end, like '`busy [anything]`', '`transfer and [anything]`', '`eight by [anything]`', and '`dodge [anything]`' are much more likely to be executed successfully.

2.6 Call Modifications

Some calls can be modified in a natural way, such as '`catch [fan the top] 3`', '`busy [1/2 tag]`', and '`vertical tag your neighbor`'. Some calls can be modified in unnatural ways, such as '`trade the diamond but replace the diamond circulate with explode the diamond`'. The usual way to perform natural modifications is simply to type the call as decribed in Section 2.3.1 [Call Variations], page 10.

To form an unnatural type of modified call, you must first enable call modifications by selecting `simple modifications` or `allow modifications` before choosing the call. There

are two levels of this feature. With `simple modifications`, you get the simple version, which only prompts you for *natural* modifications. Natural modifications have been somewhat arbitrarily defined as those for which there is an accepted way of fitting the words in without using the phrase "but replace the <whatever> with <whatever>." With `allow modifications`, you allow a potentially large number of modification possibilities, sometimes several in the same call.

In most cases, `simple modifications` is probably the right thing.

For all but the most extreme cases of modifications, you don't need to do it this way, because the "`<ANYTHING>`" mechanism can be used instead. Just type the call with the replacement call in brackets, as in

```
busy [lockit]
fascinating [ah so]
```

See Section 2.3.1 [Call Variations], page 10, for more details about this.

When either level of modifications is selected in `Sd`, the status line above the text transcript area indicates this. Also, the universal call menu is chosen whenever modifications are enabled, since the possibilities are unpredictable.

When you select a call for which modifications are possible, the program will ask a question like `The box circulate can be replaced. Do you want to replace it?` If you want to replace the designated call, click in the active area of the popup; otherwise, move the mouse away. If you indicate that you want to replace the call, you will then be prompted for the replacement call. Enter it, along with any concepts. In some cases you may be asked repeatedly about various modifications. For example, in *motivate*, you can replace the initial *circulate*, and you can turn the star a different amount. You may select any or all of these modifications.

Note: When complex modifications are involved, the program may ask for them in an order that seems unnatural. This is because it asks in the order that it executes the call internally, which may not be the same as the order in which the words are spoken. Do not be alarmed. Observe the popup titles and prompts carefully. The final result will come out in the correct order.

For both these optional modifications and the mandatory subcalls discussed earlier (e.g., `clover and <anything>`), the program attempts to show unambiguously how everything is structured, by putting subcalls and their accompanying concepts in square brackets. It also attempts to put natural modifications in their natural place in the phrase. In complex cases, like '`[CHECKPOINT LEFT catch [SINGLE CONCENTRIC snake] 3 BY [2/3 recycle] the difference] cover up`' things may be quite difficult. How (and whether) you choose to read such a card is up to you.

2.7 Recentering the Setup

When writing unsymmetrical material, one sometimes moves the entire square into a position that is not centered on its original location. In such a situation, it is sometimes useful to tell the dancers to forget that original location and consider their present location to be the entire setup. `Sd` has a pseudo-call '`recenter`' that can be used for this purpose. It recenters the setup, that is, throws away unsymmetrical phantoms. Its level has been set

to C1. It is not a real call, of course. There is no standard convention for what you have to say to the dancers when you want to perform this operation.

2.8 Retaining Concepts After an Error

When an error occurs in the execution of a call, `Sd` has an option whereby it tries to leave in place any concepts that were associated with that call. That is, after displaying the error message, the program will behave as though you had once again typed those concepts. The assumption here is that only the call was mistaken, and you really wanted the concepts and do not wish to have to enter them again. If you don't want those concepts, you can remove them one by one with the `undo` command. Or you can use the command `discard entered concepts` to get rid of all of them at once.

The `toggle retain after error` command toggles (turns on or off) the state of this option. See Section 8.9 [Changing Modes], page 36. It can also be turned on when the program begins, either by giving a command-line option (see Section 10.1 [Command-Line Options], page 47) or through the use of the initialization file (see Section 9.2 [Option Control], page 40).

2.9 Asymmetric Selectors

The people selector popup contains various asymmetric selectors such as `near line` and `far box`. You can use these selectors with the `<ANYONE> do your part` concept to call *near column pass thru*, etc. The resultant setup must be reasonable—shape-changers on one side of the set may lead to problems. If you have not restored symmetry, resolves may be extraordinarily difficult to find.

3 The Completing Reader

`Sd` and `Sdtty` use a *completing reader* for interpreting the characters that you type.

In `Sdtty`, this is the only means of entering calls and commands. In `Sd` you can use the mouse and menus, but you can also type. When you type characters into `Sd`, it uses the same completing reader as `Sdtty`. In fact, the two programs are completely compatible with each other if only keyboard input is used.

The completing reader requires you to type only as much of the word or command as is required to be unambiguous. Whenever you type a space, the program completes the word you just typed, if necessary and possible. Whenever you type ⟨ENTER⟩ (or ⟨RETURN⟩ on some keyboards), the program completes the entire line if necessary and possible.

For example, at Mainstream, you can enter a swing thru by typing '`swing thru` ⟨ENTER⟩'. When the space after '`swing`' is typed, no completion is required. When the ⟨ENTER⟩ after '`thru`' is typed, no completion is required either. If you type '`sw thru` ⟨ENTER⟩' instead, `Sd` completes the word '`swing`'. It does this by displaying '`ing`' on the screen as soon as you type the space after '`sw`'. If you type '`sw th` ⟨ENTER⟩', `Sd` will complete both words. It displays '`ing`' as soon as you type the space, and it displays '`ru`' as soon as you type ⟨ENTER⟩. In fact, you could just type '`sw` ⟨ENTER⟩'. `Sd` completes the entire line, if necessary and possible, when ⟨ENTER⟩ is pressed. So it will display '`ing thru`' after your '`sw`'. In each case, the full phrase '`swing thru`' will appear on the screen, and will be printed in the final sequence.

Because of this completion, you can get into the habit of just typing '`sw` ⟨ENTER⟩' at Mainstream. However, it will only work if what you type is unambiguous. At higher levels, '`sw`' could also mean '`swap around`'. When there is an ambiguity, `Sd` will complete as much as it can and beep at you. You can type '`?`' or '`!`' (discussed below) to see what the problem is.

With practice, you can get a reasonable feel for how much abbreviation you can get away with in a given context.

Typing ⟨ESC⟩ will complete the entire line, just like ⟨ENTER⟩, but will not actually process the line. If you like what you see, you can then press ⟨ENTER⟩ to execute the call. If the line is ambiguous, the program will display as much of the line as it can. When used effectively, this can save you a lot of typing.

`Sd`'s completing reader is completely indifferent to the capitalization of what you type. It capitalizes its output according to its own notions of aesthetics and ambiguity avoidance.

3.1 Question Mark and Exclamation Point

At any time, you can type a question mark or exclamation point. Either of these will make `Sdtty` display all legal completions of the line that has been typed so far. If what you have typed is unambiguous, that will consist of just one thing. If you have typed nothing, it will consist of every legal thing you could type—typically that is every call, every concept, and every special command.

The difference between question mark and exclamation point is that the question mark actually attempts to execute every call in the current call menu (that is, every call whose

name matches the text you have typed so far), and shows only those that can be legally performed. Exclamation point simply displays every call on the menu that matches the text that you have typed so far. Remember that the call menus are only approximate—they often have many calls listed that are in fact not legal in the present setup. This is particularly true when concepts are in place. The program simply has no idea what calls are legal without trying them. Question mark tells it to do so.

The difference between question mark and exclamation point only applies to calls. All concepts and special commands (`resolve`, for example) are always listed whenever either character is typed.

The output from typing a question mark or exclamation point may be quite lengthy. If it fills more than one screenful, `Sdtty` will stop and display '`--More--`' at the bottom of the screen. Type ⟨SPACE⟩ to go on to the next screenful. Type ⟨ENTER⟩ to see just one more line. Type a backspace (or ⟨DEL⟩) to stop. When the output ends, either because `Sdtty` displayed everything or because you typed a backspace, the partially entered line will be redisplayed, just as it was before you typed the question mark.

When you are typing calls or concepts and you type ⟨ENTER⟩ while the line that you have typed is ambiguous, `Sdtty` will display something like '`(10 matches, type ! or ? for list)`'. The number that it displays is the number of syntactically legal choices among which it can't decide. This is the number of choices that would have been listed if you had typed '`!`'. It may be more than the number of things that would be listed if you had typed '`?`'. The reason is that the '`?`' operation has to test every call before listing it. It can easily do this faster than you can read the list of choices. When you type ⟨ENTER⟩ while there are multiple syntactic choices available, the computer can't count them fast enough to respond instantaneously. It therefore displays an approximate number, computed by looking at its database without testing each call.

3.2 Minimizing the Amount of Typing You Must Do

The `Sdtty` completion mechanism, especially the ⟨ESC⟩ (or ⟨ALT MODE⟩ or ⟨TAB⟩) key, can save you a lot of typing. For example, typing

```
heads do your part ⟨ENTER⟩
```

is probably more than you want to do. If you type just

```
heads do
```

the line will be almost unambiguous—there are only two possible commands that start this way. By typing a question mark, `Sdtty` will show them to you. They are

```
heads do your part
```

and

```
heads do your part (while the others)
```

What these concepts actually mean will be discussed at length in Section 15.10 [Designating Certain People], page 93. For now, we are just discussing how to type them.

These commands don't differ until the '`t`' in '`part`'. If you type ⟨ENTER⟩ after '`part`', `Sdtty` will know that you want the first of these commands. If you type a space and then ⟨ENTER⟩, `Sdtty` will know that you want the second. Does this mean that you must type the whole command up to '`part`', followed by a space or ⟨ENTER⟩? No. Just type '`heads`

do', and then press ⟨ESC⟩. Sdtty will display 'heads do your part', since that is the text that is common to both commands. At that point, you may press ⟨ENTER⟩ to get the 'heads do your part' concept. Or you could type space and then ⟨ENTER⟩ to get the 'heads do your part (while the others)' concept. You could, of course, type the complete phrase '(while the others)' after the space, but there is no need to. Once you type the space, Sdtty knows exactly which command you want.

What would happen if you tried to shorten the command still further by typing just 'heads d' instead of 'heads do'? Below C2 this would work just fine. At C2 and above, an ambiguity would exist because of the 'heads disconnected' concept. Typing a question mark after 'heads d' will show you the ambiguity. In general, typing a question mark will help guide you in minimizing the amount of typing you must do. With practice, you will be able to enter your favorite calls and concepts with minimal typing.

Here is another example. To enter 'criss cross the shadow' or 'criss cross your neighbor', it is not necessary to type 'criss cross t' or 'criss cross y'. You can just type 'cri' followed by ⟨ESC⟩. The program will display everything up to the point of ambiguity, which, at C2, is 'criss cross '. At that point, you just need to type 't' or 'y', followed by ⟨ENTER⟩. Above C2, there are more possible calls, but typing 'cri' followed by ⟨ESC⟩ is still a good way to start.

3.3 Ignoring Hyphens and Apostrophes when Typing

You can type a space instead of a hyphen in call and concept names. For example, you can type ping pong circulate or tandem based triangles instead of ping-pong circulate or tandem-based triangles. The calls will always be printed out with the hyphen.

You can omit apostrophes when typing in call and concept names. For example, you can type lockers choice instead of locker's choice. The calls will always be printed out with the apostrophe.

4 Abridgement

The program recognizes four operations which allow use of an abridged list of calls:

`Write list (with a filename)`
: write out the call list for the indicated level and exit

`Write full list (with a filename)`
: write out the call list for the indicated level and all lower levels and then exit the program.

`Use abridged list (with a filename)`
: read in the file, strike all the calls contained therein off the menus, and proceed.

`Delete abridgement`
: only meaningful with a session. It permanently removes the association of an abridge list with that session.

These may be invoked by checking the appropriate box in the `Sd` startup dialog, and filling in the file name if required. They may also be invoked by starting the program with command-line options:

```
-write_list filename
-write_full_list filename
-abridge filename
-delete_abridge
```

The first two operations are used to prepare a call list. The call list for the indicated level, exactly as the calls appear in the menu, will be written to the named file. If `-write_list` is used, only the calls exactly on that level will be written. If `-write_full_list` is used, the lower level calls will be written as well, so the file will look exactly like the main call menu. After performing either of these operations, the program exits.

The third special flag is used to read in a list of calls to be avoided. Any call listed in the file, in precisely the same format as it was written out, will be removed from the internal database prior to running the program. Every sequence written under control of such a file will say '`(abridged)`' on its header line.

To write sequences for a group that is learning C2, for example, start Sd from the Start Menu and then check the box marked "Write list", enter the desired filename, click "accept", and choose the C2 level.

Or, if you prefer command-line operation, do (with either `Sd` or `Sdtty`):

```
sd -write_list mondayC2.txt c2
```

In all cases, this will write out its C2 list into the file '`mondayC2.txt`'.

Then delete from that file those calls that the group has learned, i.e., those calls not to be avoided. Use a plain text editor, such as Emacs, Notepad, or vi, for this.

When writing sequences, start Sd from the Start Menu and then check the box marked "Use abridged list", enter the desired filename, click "accept", and choose the C2 level.

Or you can do:

```
sd -abridge mondayC2.txt c2
```

As the group learns new C2 calls, delete the corresponding lines from the file '`mondayC2.txt`'. That file always contains the calls that they don't yet know. When the file goes to zero, they (presumably) know the whole list.

Be aware that the abridgement mechanism works only for calls, not for concepts. You must keep track of what concepts not to use.

The lines in the abridgement file must always be in exactly the same format as the strings that are written out by the list creation operation. The program has no tolerance for creative capitalization, stray blanks, or other variations. Any line in the file that does not match a call in the menu is simply ignored. The order of the lines is not important. We recommend using the file exactly as it is written out with the "write list" or "write full list" operation, and using an editor only for the purpose of deleting lines from it.

The abridgement mechanism, and the session mechanism, work interchangeably between `Sd` and `Sdtty`. You can use one program to create or manipulate abridgement files or sessions, and then use the other program to use them.

4.1 Associating an abridgement file with a session

You can associate abridgement lists with sessions. To create a new session with a given abridgement file, start Sd from the Start Menu and then check the box marked "Use abridged list", enter the desired filename, select the session labeled "create a new session", select the desired level, and enter the session number when prompted.

Or you can do:

```
sd -abridge mondayC2.txt c2
```

and create the session from there.

If you want to associate an abridgement file with an already existing session, do the same thing, but select the existing session. Once the association is made, it is permanent–this step will not be required again. Simply selecting the session will get the abridgement list.

4.2 Deleting an association

You might want to do this after the class has finished learning the list, and you want to continue writing material for that session. Start Sd from the Start Menu and then check the box marked "Delete abridgement", and select the desired session.

Or you can do:

```
sd -delete_abridge
```

5 Resolving and Searching

This chapter describes how to have the program find calls for you to accomplish various goals.

5.1 Resolving

Whenever the setup is in a resolved state, whether intentionally or accidentally, the program indicates that fact at the bottom of the transcript area. The program looks for *right and left grand*, *left allemande*, *promenade*, *single file promenade*, and *circle left/right* getouts from a variety of setups. If the sequence needs an *extend*, *slip the clutch*, *circulate*, *pass thru*, *trade by*, *cross by*, or *dixie grand* first, the program reports that too.

Whenever the '`resolve`' message appears, you can end the sequence and write it to a file by selecting `write this sequence`. In `Sd`, the button for this is in the group in the upper left portion of the screen. In `Sdtty`, simply type the command, with the usual completion mechanism.

There are at least three ways to resolve a sequence: you can wander into a resolve by accident, you can *sight resolve* the square, entering the calls that you want, or you can use the `resolve` command. This command adds the necessary calls to the sequence, exactly as if you had entered them.

When you select `resolve`, the program goes into a special mode in which it searches for resolutions, saves them, and lets you look through them and pick one that you like. A resolution is a sequence of up to three calls that leads to a resolved state. While the program is in resolve mode, the call menu is replaced by a special menu of options, along with information about the current resolution. That information tells how many resolutions are currently stored and which one is currently shown. When you select `resolve`, the program finds the first resolution and displays it, showing information '`1 out of 1`', which means that it has one resolution stored, and resolution number one is currently displayed. The transcript area shows the effect of that resolution.

You can select `find another` to search for another resolution and add it to its stored list. When the list has more than one resolution in it, selecting `previous` and `next` will move around in its list of previously found resolutions and show whichever one you want. In this way, you can search for a better resolution than the one you already have, but go back to the earlier one if no better one is forthcoming.

Selecting `accept current choice` will leave resolve mode, causing the current resolution to be added to the sequence exactly as though you had entered those calls manually.

Selecting any of `abort the search`, `exit the search`, `quit the search`, or `undo the search` will throw away all of the saved resolutions and leave resolve mode, but will not destroy the sequence. The sequence will be left just as it was before `resolve` was selected.

Selecting anything else, such as `write this sequence`, is equivalent to `accept` followed by whatever that action is. So, for example, you can select `write this sequence` as soon as you see a resolution that you like.

These special commands may be typed into the completing reader in the usual way. To make it more convenient to enter them, the command names have been chosen to be

unambiguous from just one or two letters. So, for example, `p` means `previous`, `n` means `next`, and `f` means `find another`.

If one or more concepts have been entered when you select `resolve`, the program will search only for resolutions whose first call starts with those concepts. So, for example, selecting `once removed` and then `resolve` might get you this resolution:

```
ONCE REMOVED reverse the pass
linear flow
right and left grand  (7/8 promenade)
```

The program searches for resolutions by using a random number generator to generate up to 15000 random sequences, occasionally inserting concepts. It biases the search in favor of short sequences (one call) rather than long ones (three calls) and against resolutions that require *all 8 circulate*, *pass thru*, or *trade by* at the end. If, after 15000 attempts, no resolution is found, the `find another` operation fails. (This tends to happen if you try to resolve out of an hourglass at Mainstream.) You can select `find another` again to make another 15000 attempts if you wish.

Remember that the `resolve` *operation by itself does not write the sequence to a file.* You will not be able to print a sequence until it has been written to a file. You must give the `write this sequence` command (or press function key ⟨F10⟩) to write out the sequence.

5.2 Mini-Grand Getouts

Some callers like to use the getout of "Right and Left Grand, but promenade on the third hand", sometimes known as a "Mini-Grand". Under normal circumstances, `Sd` will not search for such things. If you want them, issue the "toggle minigrand getouts" command. You may also customize the program by putting the line "minigrand_getouts" in the initialization file, or "-minigrand_getouts" in the command line of a shortcut icon. See Section 9.2 [Option Control], page 40.

5.3 Normalizing or Standardizing the Setup

These commands invoke an operation very similar to the resolver, except that they search for sequences of up to three calls that make the setup nicer.

The `normalize` command reduces various matrices, such as 4x4, into an 8-person setup. It uses a variety of phantom-like concepts to do its work.

Be aware that the phantom-like concepts that this command uses are not the only ways to get out of large matrix setups. There are a number of calls, like *press*, *truck*, *loop*, *squeeze*, *Z axle*, and *finish a long trip* that are also useful. The `normalize` operation does not search for such things. It is only intended to search for things that might not be obvious.

The `standardize` command turns the setup into an 8-person setup in which people are facing in "reasonable" directions. That is, it gets out of T-bone setups. It uses both plain calls and calls with concepts.

If you have selected `toggle concept levels`, these commands may make use of concepts that are not legal at the chosen level. This is sometimes useful in emergencies.

5.4 Letting the program pick a random call

There are five commands in this family: `pick random call`, `pick simple call`, `pick concept call`, `pick level call`, and `pick 8 person level call`. They search for any legal single call. While this may sound like a fairly pointless operation, remember that you can use this while one or more concepts are already entered, in which case it will search for legal calls that involve those concepts. Hence, this operation is useful for finding clever uses of difficult concepts such as *checkpoint*, *interlace*, or *on your own*.

The command `pick random call` attempts to pick completely random calls, sometimes with a concept and sometimes not. The `pick simple call` command never uses concepts. The `pick concept call` always uses at least one concept with the call that it picks. `Pick level call` picks only calls that are at or near the specified calling level. (By "near" we mean that, if the calling level is C4, it will pick C4A or C4 calls. If the level is A2, it will pick A1 or A2 calls.) `Pick 8 person level call` is similar, but it picks only calls that involve all 8 people.

5.5 Creating a Specific Setup

These commands pick some call or calls that go into a specified setup. The commands are:

```
create any lines
create waves
create 2fl
create lines in
create lines out
create inverted lines
create 3x1 lines
create any columns
create columns
create magic columns
create dpt
create cdpt
create trade by
create 8 chain
create any 1/4 tag
create 1/4 tag
create 3/4 tag
create 1/4 line
create 3/4 line
create diamonds
create any tidal setup
create tidal wave
```

"Any lines" and "any columns" mean any 2x4 setup in which people are all in generalized lines or columns, respectively. "Any tidal setup" means any 1x8, no matter how people are facing. All the commands are independent of handedness—for example, a two-faced line can be of either handedness.

5.6 The `Reconcile` operation

Selecting `reconcile` invokes an operation like resolving, but it puts the generated calls someplace other than at the end of the sequence. This is useful if you have a clever getout at the end of the sequence, and you want it to be the resolve, but people don't have their partners and corners. This lets you retroactively modify the sequence so that the clever getout will work.

Note first that this operation may only be invoked when the setup is left-handed two-faced lines, right-handed waves, or left-handed waves, in which case it assumes you want a *promenade*, *right and left grand*, or *left allemande* respectively. The program must know which getout type you want. If the setup is an 8-chain, it can't tell. In that case, either do a `touch` or a `left touch`, to tell the program that you really want a *right and left grand* or *allemande left*, respectively. Then, after the reconcile operation is complete, you can erase that extra call.

The reconcile operation is very similar in behavior to resolve, except that the program needs to know the insertion point. Rather than searching for the first resolve as soon as you enter the mode, the reconcile operation lets you set the insertion point before searching. Select `raise reconcile point` or `lower reconcile point` to set it. A dotted line will be displayed showing where in the sequence the generated calls will be placed. When this is in the right place, select `find another` to find the first reconcile. You can change the insertion point at any time.

Reconciles are extremely difficult to find—much harder than resolves. To avoid frustration, make the insertion point be at a place where the setup is very simple and a large number of calls are legal. We recommend making the insertion point be at a place where the setup is in waves. Remember that, if the insertion point is at an hourglass, the program has to find a random sequence of up to three calls that goes from an hourglass to another hourglass while miraculously performing the required permutation.

Avoid using reconcile when a gender-dependent or head/side-dependent call lies between the insertion point and the end of the sequence. The program checks every reconcile by re-executing all calls from the insertion point to the end and verifying that everything is exactly as it was except for the permutation of the people. For example, if the call '`heads kickoff`' occurs after the insertion point, and a potential reconcile changes heads and sides, it will not be offered. For gender-dependent calls the situation is a little better: if, at the end of the sequence, the boys are in the center and the girls on the end before doing the reconcile, you know that any inserted sequence will have to be gender-preserving anyway, so calls like '`star thru`' and '`boys kickoff`' will be okay.

At-home Reconciling

If you have a nice getout that finishes on squared-set spots (or a 2x4 approximation to same), you can use the reconcile operation to make it work. First, move the headliners into the middle if necessary to make a 2x4. Then have the centers face out, and have everyone do a `left touch`. Start the reconciler, set the insertion point, and search. Discard any solution that doesn't say "at home". When you find one that you like, accept it, and then delete the calls that you had to put in at the end.

6 Editing

This section describes the "editor".

`Sd` allows you to edit any part of your sequence while you are writing it. Normally, of course, you are just adding calls to the end, or perhaps undoing the last call. What happens if you decide you want to change something farther back? You could undo all the calls back to that point, while remembering them or writing them down, make the changes, and then type them in again. `Sd` can remember them for you. It does this with a "clipboard". You move the calls, one at a time, from the sequence that you have written, onto the clipboard. Then you can "edit" the sequence at the chosen point, adding calls, undoing calls, or whatever. When you are done, you can restore the later calls by moving them from the clipboard back to the sequence.

The `Sd/Sdtty` clipboard has nothing to do with the Windows clipboard. The former stores calls, and the latter stores text. You can cut, copy, and paste text between the `Sd` text entry window and the Windows clipboard, using the "Edit text" menu, though there is rarely any reason to.

The commands to manipulate the clipboard are:

```
cut to clipboard
paste one call
paste all calls
delete one call from clipboard
delete entire clipboard
```

6.1 Moving the Last Call to the Clipboard

Normally the clipboard is empty. To start the editing process, issue the `cut to clipboard` command, or press function key shift-⟨F8⟩. This will move the last call onto the clipboard, removing it from the active sequence. If this is done repeatedly, the calls will all be stacked on the clipboard. Do this until you reach the point in the sequence that you want to change.

While there are calls in the clipboard, they will be displayed (well, the first three will be displayed) between two rows of dotted lines. This way, you can see what will be pasted back.

6.2 Moving One Call Back From the Clipboard

After you have finished editing the sequence, you can move the calls back. The command `paste one call`, or pressing function key control-⟨F8⟩, will move the nearest call from the clipboard back to the active sequence.

There are a few tricky issues involved with this process. `Sd` assumes that you want the indicated call replaced, even if the formation is different. For example, if the call in the clipboard was '`swing thru`', and you have changed the formation to left-handed waves, it will still do a (right) swing thru. If the formation has changed radically, the interpretation may be very different. If there are sex-dependent calls and you have rearranged people, things might turn into something very different from the original sequence.

There is one case in which `Sd` will alter a call while it is being pasted. If a call has a designator like '`boys`' or '`heads`', and the formation is geometrically the same as before, `Sd` will try to change the designator to get the equivalent geometrical effect. For example, if the pasted call had been '`boys run`' from waves in which the boys were looking out, and you edit the sequence so that the formation is once again waves, but with some other people looking out, the call will be changed to '`heads run`' or whatever. Of course, if you had originally said '`leads run`' no change would be required.

Whenever a designator is changed, or whenever the formation in which the pasted call is performed is different from the original formation, `Sd` prints a warning. You may find a change from '`boys run`' to '`side corners run`' to be unwelcome.

It may happen that the pasted call is illegal in the new formation. If that happens, the call will not be pasted, `Sd` will print an error message, and you will have to take further action.

6.3 Moving The Whole Clipboard Back

You can paste all the calls back from the clipboard at once by issuing the `paste all calls` command. If any formation changes, or any designator is altered, a warning will be printed. If any call is illegal in its new context, the pasting operation will be stopped just before the pasting of that call. Earlier legal calls will have been pasted.

6.4 Throwing Away Just One Call

If you are pasting calls back and an illegal one is encountered, you may decide that the correct action is to change what is on the clipboard. You can do this by issuing the `delete one call from clipboard` command. This throws away the offending call and lets you proceed with further editing.

For example, suppose you were writing a sequence in which the formation was diamonds at some point, and the next call was '`flip the diamond`' followed by other clever stuff. You decide that you want to do something interesting with the diamonds, followed by the flip the diamond and the other stuff. So you cut the '`flip the diamond`' and all later stuff onto the clipboard. Then you edit from the diamonds. While editing, you get into some wonderful situation in which you have triple boxes, you want to do a '`triple box circulate`' to get to waves, and then you want to proceed with the rest of the sequence, but *without* the '`flip the diamond`'. The diamonds are no longer what you want at that point. So you call the '`triple box circulate`' and want to paste the calls back. Unfortunately, the first call on the clipboard is '`flip the diamond`'. `Sd` refuses to paste it. You can throw it away with the `delete one call from clipboard` command. The rest of the clipboard is still there, starting with the waves that were originally created by flipping the diamond. You can now paste them.

6.5 Throwing Away the Entire Clipboard

Sometimes you decide that everything you cut to the clipboard was stupid, and you just want it to go away. Issue the `delete entire clipboard` command.

7 Printing

There are a number of ways to have your sequences printed. The simplest is to let `Sd` do it directly, at the end of a session. Before `Sd` exits, it offers to print the current file.

You can also give the explicit command `print current file` at any time. However, we don't recommend doing this except at the very end of a session, because `Sd` always prints the entire file. If you issue the print command twice in one session, the second command will duplicate all the sequences printed the first time.

You can also leave the sequences on a disk file and print the file later. Click `No` when it offers to print the file. You can print it later with `Sd`. Just start `Sd` and give the `print any file` command. This lets you browse to the file that you want. (In fact, the `print any file` command can print any file at all. This is sometimes useful. Do not use it to print documents from word processors such as Word 97.)

When `Sd` prints a file, the default font is 14 point Courier Bold. You can choose another font or type size by giving the `choose font for printing` command. This only affects printing. It has no affect on the screen display.

Only `Sd` is capable of printing. `Sdtty` is not. Of course, `Sd` can print files prepared by `Sdtty`.

You can also print sequence files by other means. You can simply copy the file to the printer, using the Windows Explorer. There are also a large number of software packages that are capable of printing files. We do not recommend Notepad or Wordpad for printing sequence files. These formatters do not recognize the page boundaries in imported files, so the "cards" will not be on separate pages. If you want to print files with a word processor, we recommend Word 97. It handles page boundaries correctly, though its line margins may cause unwanted line breaks.

To read a file into Word 97, select `File` and `Open` from the menu. Set the "Files of Type" pulldown to "All Documents". Navigate to the desired folder and select the desired file. If you want to print in a different font or type size, first select `Edit` and `Select All` from the menu. (This will put the entire file into reverse video.) Then select the desired font and/or type size from the pulldown menus on the Format Bar. Then choose `File` and `Print`.

If you made any changes to the file while in Word 97 or another word processor, including just changing the font for printing, the program may ask whether to save the changes. It is dangerous to do so unless the file was written with an `Sd` session file name of '`*`'. See the "sessions" document for details on file management.

8 Miscellaneous Commands

This section describes commands that are not concerned with generating sequences.

8.1 Saving the Sequence

At any time when the sequence is resolved (for example, after a successful use of the `resolve` operation), you can select `write this sequence` or press ⟨F10⟩. (This command used to be called `end this sequence`.) This will append the current sequence to the current output file, and then go back to the startup menu. You will be given an opportunity to enter a line of text to be used as a subtitle, for example "very hard interlace" or "stupid biggie." The written sequence will also be annotated with the level, the session title, the current date and time, and the version numbers of the program and database.

Until you have done the `write this sequence` operation or pressed ⟨F10⟩, the sequence is not written to disk and you cannot start writing a new sequence. Just resolving isn't enough.

8.2 Changing the Output File

You can change the name of the transcript file to which the program writes its output by selecting the `change output file` command. A popup will appear, prompting you for a new file name, or `Sdtty` will ask you to type in the new name. This action will become effective the next time a sequence is written out. Sequences previously written will stay under their old name.

The output transcript file name typically does not have any directory information, and the output is written to the current directory for `Sd` or `Sdtty`. Sometimes the operating system does not allow one to control the current directory, but one wishes to write output files to a different directory. The directory can be changed with the `change output prefix` command. The given directory argument will be prepended to the transcript filename (e.g. 01mar15_c2.txt). The given directory argument should be a fully qualified pathname, followed by the appropriate separator character: `"/"` or `"\"`. So an argument of `"/home/wba/sd/"` will cause the transcript file to be `"/home/wba/sd/01mar15_c2.txt"`.

When the initialization file is used (see Chapter 9 [The Initialization File], page 38), any change in the output file name (but not the prefix) for a session will be written back to the init file at the end of the session.

8.3 Changing the Title

You can change the title that will appear at the top of each sequence by selecting the `Change Title` command. A popup will appear, prompting you for a new title, or `Sdtty` will ask you to type in the new title. This action will become effective the next time a sequence is written out. Sequences previously written will keep their old title. You can remove the title by just pressing ⟨ENTER⟩ when asked for the new title.

When the program starts, there is no title unless the initialization file was used. See Chapter 9 [The Initialization File], page 38.

When the initialization file is used, any change in the title will be written back to the init file at the end of the session.

8.4 Changing to the New File Name Style

Prior to version 34.6, when support for DOS and Windows 3.1 were withdrawn, the program used names for the transcript (output) files in an "old" style, which looked like "13aug05.c1" or "sequence.c1". It now supports a "new" style, which looks like "13aug05_c1.txt" or "sequence_c1.txt". (The names with dates in them are what you get if you use the "+" file name. See Section 9.1 [Session Control], page 38.)

The part after the period is called the "extension". When using modern operating systems, it is very helpful for files to have a correct extension, and ".txt" is the right extension for these files. For example, attaching files in email messages is sometimes facilitated by having the ".txt" extension.

Whether you use old or new style file names is controlled by your initialization (session) file, in the options `new_style_filename` or `old_style_filename`. See Section 9.2 [Option Control], page 40.

There is a command to change the style specified in your initialization file from old to new. The command is "change to new style filename". It must be given when the program starts, not when it is in the middle of a sequence. After giving the command, you must exit the program and restart it.

When you change to new style file names, no previously written files will be renamed. Only newly created files will have the new names. Also, no pre-existing sessions that use explicit old style names (e.g. "sequence.c1") will have those names changed. (You may change the explicit names used by existing sessions with the "change output file" command.) Pre-existing sessions that use the "+" or "*" file name will be changed to the new style for all future sequences, based on the date.

8.5 `Abort`, `Exit`, and `Undo`

If you select the `exit` command, the program will exit. If a sequence is in progress, the program will ask for confirmation first. (Remember that a sequence is not finished until the `write this sequence` command has been given.) In `Sd` with the X Window System, using your window manager to send a `Delete Window` message to the program is equivalent to clicking on the `exit` button. This is what normally happens when you double-click the special control button in the upper left corner of the `Sd` window.

If you select the `abort` command, the program will abort the sequence but not exit. It will go back to the startup menu to allow you to start another sequence. It will ask for confirmation first.

If you select the `undo` command, or press function key ⟨F9⟩, while no call is partially entered, that is, while no concepts have been entered, the program will erase the last complete call, with all of its concepts. If one or more concepts have been entered, only the last concept will be erased.

8.6 Inserting Comments

You can insert arbitrary text strings into the sequence transcript in the form of *comments*. This is useful for things that you may want to say to help the dancers, such as "side boys be careful—go to your left." Also, if you want to call something that the program can't do, so you have to trick the program into doing it by means of several other calls, it is useful to enter a comment saying what you originally wanted. This can guide you when editing the transcript file.

To enter a comment, select `insert a comment`. In `Sd`, a popup will appear into which you may enter the text. In `Sdtty`, simply type the comment text on the next line. The comment will be enclosed in curly braces in the transcript.

A comment may be entered in front of any concept or call and goes into the transcript exactly where it was entered. Because of the way the program assembles mouse-clicks into transcript lines, it is not possible to enter a comment at the end of a line.

8.7 Keeping Pictures

The program shows pictures on the screen for the current position and the position just before the last call. Normally, those pictures will not appear in the sequence written to a file. If you want the current position to have its picture written in a file (say, because the sequence is very difficult and you believe it may be necessary to say something helpful then like "check a parallelogram with boys in the center box") select the command `keep picture`. You may also cause the program to keep pictures in the output file for the entire sequence by issuing the `toggle keep all pictures` command or by placing the `keep_all_pictures` option in your initialization file.

8.8 Singing Call Progressions

You can write singing call sequences with the usual "progressions" in them. At the start of the sequence (that is, when you could have typed '`heads start`'), issue one of the following commands:

```
toggle singing call
toggle reverse singing call
```

The first gives the usual "corner" progression, and the second gives a reverse ("right hand lady") progression.

When in singing call mode, the program looks for resolves that cause people to promenade with the appropriate person who is not their partner. It assumes that you are looking primarily for '`swing and promenade`' getouts, though it will show other types of getouts when they appear.

Note that, for the "usual" type of singing call figure (equivalent to '`square thru 4, swing corner, and promenade`') the promenade distance will be shown as 1/8. The dancers will of course promenade 1-1/8. You should consider '`at home`' or '`1/8 promenade`' to be the "usual" timing for singing call figures.

8.9 Changing Modes

There are several commands that change some aspect of the way `Sd` or `Sdtty` runs. They are:

```
toggle singlespace mode
toggle concept levels
toggle active phantoms
toggle retain after error
toggle nowarn mode
toggle keep all pictures
toggle singleclick mode
toggle minigrand getouts
toggle singing call
toggle reverse singing call
```

Normally, all of these are off. You may want to turn one or more of them on for your own personal preference, or for some special reason. These commands *toggle* their respective modes. That is, they turn the mode on if it was off, and off if it was on. The current status of some of these modes is always displayed in the input prompt in `Sdtty`, or the status bar at the bottom in `Sd`.

The `toggle singlespace mode` command changes the format of the output file between single spacing and double spacing. The default is double spacing. This command affects only future sequences. It has no effect on sequences already written.

The `toggle concept levels` command changes the legality of concepts that are not on the specified calling level. See Section 2.4 [Entering Concepts], page 16, and Chapter 13 [Using Off-Level Concepts], page 53.

The `toggle active phantoms` command changes the persistence of phantoms during certain calls. See Section 15.7 [Assume Waves], page 86. We do not recommend using this. It may make the program refuse to do things that real dancers consider completely legal, due to phantoms colliding with or otherwise interfering with live dancers. Instances in which this option is needed are nearly unheard-of.

The `toggle retain after error` command changes the option whereby program retains the concepts that you had typed when an error occurs. See Section 2.8 [Retaining Concepts After an Error], page 20.

The `toggle nowarn mode` command changes the suppression of warning messages.

The `toggle keep all pictures` changes the mode that keeps all pictures. This mode effectively puts a "keep picture" command at every point in the sequence.

The `toggle singleclick mode` changes the acceptance of single mouse clicks on menu items.

The `toggle singing call` commands change the singing call mode, which affects resolves. See Section 8.8 [Singing Call Progressions], page 35.

The `toggle minigrand getouts` command changes the search for "mini-grand" getouts in the resolver. See Section 5.2 [Mini-Grand Getouts], page 27.

Any of these modes, except the singing call modes, can be turned on automatically when the program begins, either by giving a command-line option (see Section 10.1 [Command-

Line Options], page 47) or through the use of the initialization file (see Section 9.2 [Option Control], page 40).

9 The Initialization File

When `Sd` or `Sdtty` starts, it looks for a file '`sd.ini`' in the working directory. This file, if present, contains information about dances you might be working on, and about possible individual preferences you might have for the way the program runs. Such a file might look like this:

```
[Options]
reverse_video
no_warnings
print_length 68
window_size 850x650
singlespace

[Sessions]
sequence.C3       C3          12      NACC, June 1995
sequence.C4       C4          31      NACC, June 1995
workshop          C1          75      My Wednesday group
+                 A2           9      NESRDC
*                 A1           2      Lake Shore Farm
```

There are up to three sections in this file, called the "sessions" section, the "options" section, and the "accelerators" section. Each section starts with a header word in brackets. The sections are separated from each other by blank lines.

9.1 Session Control

Each line after the line `[Sessions]`, up until a blank line or the end of the file, describes a possible session of using the program, showing the output filename, level, next sequence number, and title.

Th example file above shows 5 possible things you could work on. You can select any one of them during a session. When the program is started, it will display the 5 lines and ask you which one you want to use, like this:

```
Do you want to use one of the following sessions?

  0     (no session)
  1  sequence.C3       C3          12      NACC, June 1995
  2  sequence.C4       C4          31      NACC, June 1995
  3  workshop          C1          75      My Wednesday group
  4  +                 A2           9      NESRDC
  5  *                 A1           2      Lake Shore Farm
  6     (create a new session)

Enter the number of the desired session:
```

In `Sdtty`, type the number of the selection that you want. In `Sd`, either select the desired line with a double mouse click, or use the cursor keys to highlight the line and then press ⟨ENTER⟩.

If you select "0", or just press ⟨ENTER⟩, the program will ignore the sessions and proceed normally. If you enter the number of one of the lines that are displayed, the program will operate at that level, and use the named output file and title. It will also print a sequence number on each card, starting with the number shown. For example, if you entered "3" after the file above is displayed, the program will operate at C1. The output file will be "workshop" instead of the default "sequence.C1". Cards will be serialized starting at 75. Each card will have a title saying "My Wednesday group".

If the given file name is "`+`", the program will generate a file name for you, based on today's date. This may be useful for managing files, so that you know when it is safe to delete a file that you have printed. The program will of course tell you what the name of that file is. All runs of the program using that session on the same day will append their sequences to the same file. The file name will be something like like'`9mar96.C3`'.

If the given file name is "*", the program will generate a file name in the same way, except that it will always generate a new file for each run of the program. If the program is started several times in one day, the files will be different. The generated file names will be something like '`9mar96a.C3`'. Note the '`a`' after the year.

At the end of the session, the initialization file will be updated with the next index number. For example, if you wrote 12 sequences, they would be numbered 75 through 86, and the file would be rewritten as

```
workshop          C1            87      My Wednesday group
```

so that the next session for that group would start with sequence number 87.

The title bar displays the level, session title, current sequence number, and starting sequence number.

If, at any time during the session, you change the output file (with the `Change Output File` operation) or the title (with the `Change Title` operation), the initialization file will be updated at the end of that session to show the effect of the change.

When the program starts, you can tell it to create a new entry in the file by entering the number corresponding to `(create a new session)`. In the above example, select "6". The program will ask you for the level and title, and will set the output file to the standard "sequence.C1" or whatever. You may use the `Change Output File` command and the `Change Title` command to change these later if you wish. At the end of the session, item number 6, containing the appropriate data, will be written to the updated initialization file.

If you no longer want some entry in the initialization file, you can delete it. In `Sdtty`, type the negative of its line number. In the above example, if you entered "-4", the line

```
+                 A2            9       NESRDC
```

would be deleted. In `Sdtty`, check the box labeled "Delete this session" and select the line to be deleted. Either way, the program will terminate immediately after doing this. Run it again to use the new initialization file.

Whenever the program updates the initialization file '`sd.ini`' at the end of a session, it saves the old contents in '`sd2.ini`'. By copying that back to '`sd.ini`', you can restore it.

You can also edit the file with an editor. Simple editors, such as '`Emacs`' and '`Notepad`' are preferable to sophisticated word processors when editing this file, since word processors often insert specialized control information into the file. The simplest way to edit the file is to double-click the icon '`Edit sd.ini`' in the '`C:\sd`' folder. This is a shortcut to the '`Notepad`' editor.

9.2 Option Control

Each line after the line `[Options]`, up until a blank line or the end of the file, contains an option specifier. When the program is started, it will use those specifiers to determine a number of aspects of the program's behavior. This makes it possible to encode your personal preferences so that the program will always use them. Do *not* place a hyphen in front of an option in the initialization file.

A few of these options can also be changed while the program is running, by giving suitable commands. See Section 8.9 [Changing Modes], page 36.

Here are the available options:

`singlespace`
: This makes the program write the output file with single spacing instead of the usual double spacing. This may be useful for those who use special fonts or special printer arrangements. This option is the same as giving the `toggle singlespace mode` command while the program is starting a sequence.

`print_length` *`number`*
: This sets the length of print lines for the output file, that is, the point at which long lines will be broken. The default is 59, which seems to be correct for the Courier 14 point font on standard American (8.5 x 11 inch) paper size.

`concept_levels`
: This allows all concepts, even those that are not actually legal at the specified level, to be used. It is the same as giving the `toggle concept levels` command while the program is running.

`active_phantoms`
: This turns on the "active phantoms" behavior, which makes phantoms persist throughout calls. This option is the same as giving the `toggle active phantoms` command while the program is running.

`minigrand_getouts`
: This turns on the acceptance of "mini-grand" getouts in the resolver. This option is the same as giving the `toggle minigrand getouts` command while the program is running.

`no_warnings`
: This suppresses the display and printing of the warning messages that are sometimes given to help less-experienced callers. It is the same as giving the `toggle nowarn mode` command while the program is running.

`retain_after_error`
: This makes the program automatically retain all concepts that had been typed prior to the current call whenever an error occurs. The usual behavior is to discard all typed concepts when an error occurs. It is the same as giving the `toggle retain after error` command while the program is running. See Section 2.8 [Retaining Concepts After an Error], page 20.

`discard_after_error`
: This is the opposite of `retain_after_error`. Since this is the usual behavior, you don't need this option. It is present only for compatibility with older versions. If it is in your '`sd.ini`' file, take it out.

`tab_changes_focus`
: This applies to `Sd` only. Windows programs normally use the tab key to move keyboard focus around among the active windows in some fixed order, and the shift-tab key to move in the opposite order. This change of focus is nearly useless in `Sd` for most users, because the keyboard focus is almost always in the text input region. Therefore, `Sd` normally has the tab key do something different—it performs the same completion function as the escape key, just as `Sdtty` does. (The shift-tab key still moves window focus in the backwards order.) The `tab_changes_focus` option re-enables the conventional Windows meaning of the tab key. With this option turned on, if you are typing in a call, and you type tab, it will not complete the call. Instead, focus will shift to the call menu. You can use the arrow keys to move around in the call menu, and then press ⟨ENTER⟩ to select that call.

`keep_all_pictures`
: This keeps a picture after every call, as if the "keep picture" command had been given each time. It is the same as giving the `toggle keep all pictures` command while the program is running.

`single_click`
: This makes `Sd` respond immediately to a single mouse click in a menu. Normally, a double click, or pressing the '`Accept`' button, is required. This option is meaningless in `Sdtty`. It is the same as giving the `toggle singleclick mode` command while the program is running.

`maximize`
: This tells `Sd` to "maximize" its window, that is, to use the full screen. It is the same thing that would happen if you clicked the open box button in the upper right corner. This has no effect on `Sdtty`. `Sdtty` runs as a "command prompt" program, and its window size is set by, and can be manipulated by, the operating system.

`window_size` *`size-designator`*
: This argument is followed by a designator like "750x450". It will set the window size to the indicated number of pixels. There is enormous variation in the screen size of various displays and graphics cards, and in people's preferences, so you may find the standard default size (780x560 pixels) not to your liking. The two size numbers are separated by the letter "x", with no space.

  You can also adjust the screen position by giving 4 numbers, as in "window_size 750x450x20x20". The arguments are the X and Y size, and the X and Y position of the upper left corner, all in pixels.

  These options do not affect `Sdtty`. `Sdtty` runs as a "command prompt" program and cannot set its own window size. However, you can usually control this either by editing the icon that starts the program (right click the icon, select "properties", and, on the "shortcut" tab, select "maximized") or by changing

the general layout of command prompt programs (start, control panel, console, and use the "options" and/or "layout" tabs.)

`no_checkers`
: `Sd` normally draws the formation on the screen with icons that are intended to look like callers' "checkers". This option disables that, and displays dancers by more normal means. Pictures drawn in the output file never use these icons. This option is meaningless in `Sdtty`.

`no_graphics`
: This suppresses the use of special PC graphics characters ("triangles") when drawing pictures on the screen. These characters are not standard on all PC's. Pictures drawn in the output file will never use the special graphics characters. This overrides `no_checkers`.

`no_color`
: `Sdtty` and `Sd` normally color-code the dancers shown on the screen, according to various color schemes that are controlled by options listed below. This option disables the use of color. These options affect only the display on the screen. Pictures drawn in the output file will never be in color, and will never use "checkers" or triangles.

`color_by_couple`
: This draws dancers on the screen in four colors, one for each couple, in the sequence red, green, blue, and yellow.

`color_by_couple_rgyb`
: Like color_by_couple, but the color sequence is red, green, yellow, and blue.

`color_by_couple_ygrb`
: Like color_by_couple, but the color sequence is yellow, green, red, and blue.

`color_by_corner`
: Like color_by_couple, but dancers have the same color as their respective corners. In the absence of these color options, girls are drawn in red and boys in blue. Pictures drawn in the output file never use color.

`reverse_video`
: This displays the transcript area (or the whole window in `Sdtty`) with a black background and white text. This is the default for `Sdtty`, so you only need this option for `Sd`.

`normal_video`
: This displays the transcript area (or the whole window in `Sdtty`) with a white background and black text. This is the default for `Sd`, so you only need this option for `Sdtty`.

`no_intensify`
: This changes all "white" displayed matter to be light gray instead. In reverse_video mode, text will be light gray on black instead of white on black. In particular, this will make `Sdtty` look like the customary appearance of a Command Prompt window. In normal_video mode, text will be black on light gray instead of black on white.

`pastel_color`
: This causes the colors for the usual girls-are-red and boys-are-blue scheme to use lighter shades. This may be preferable if `reverse_video` is used. It has no effect when using the `color_by_couple`, `color_by_couple_rgyb`, `color_by_couple_ygrb`, or `color_by_corner` schemes (but see `use_magenta` and `use_cyan`, below.) This is the default for `Sdtty`, so you only need this option for `Sd`.

`bold_color`
: This is the opposite of `pastel_color`—it uses darker shades. This is the default for `Sd`, so you only need this option for `Sdtty`.

`use_magenta`
: This lightens the shade of red in all color schemes. (`Pastel_color` does not apply in any of the `colr_by_couple` schemes.) It substitutes magenta for red.

`use_cyan` This lightens the shade of blue in all color schemes. It substitutes cyan for blue.

`no_sound` Do not ring the bell or make any other sound when reporting errors.

`new_style_filename`
: The output file will be written with the extension '`.txt`', as in '`sequence_C1.txt`' or '`01sep02_C1.txt`'. Without this switch one gets the default "old" behavior, which is '`sequence.C1`' or '`01sep02.C1`'. This default may change soon. Having a file name of this form should make the output files able to be read and printed by common text editors and word processors.

`old_style_filename`
: This forces the output file to be written in the "old" style. This will be needed after the default changes to the "new" style.

`db` *`filename`*
: location of the calls database. The default is '`sd_calls.dat`' in the current directory.

`sequence` *`filename`*
: base name of the file to write sequences to. The calling level will be used as the extension of the file name. The default base name is '`sequence`', so a complete file name might be '`sequence.C1`'.

`output_prefix` *`filename`*
: This facilitates running on operating systems in which one does not have control over the program's current directory. The output files are normally written to the current directory. This can be changed with the "-output_prefix *directory*" switch when invoking Sd or Sdtty. The given directory argument will be prepended to the transcript filename (e.g. 01mar15_c2.txt). The given directory argument should be a fully qualified pathname, followed by the appropriate separator character: "/" or "\". So an argument of "-output_prefix /home/wba/sd/" will cause the transcript file to be "/home/wba/sd/01mar15_c2.txt".

`sequence_num` *`number`*
: Initial sequence number. This overrides any sequence number given in a session from the initialization file.

`no_cursor`
: Dumb terminal mode: do not use cursor motion commands to keep the screen up-to-date efficiently. Use this if the cursor motion and screen manipulation operations of your terminal or operating system are not available or are not working satisfactorily. This is only legal in `Sdtty`, and is meaningless on Windows 95 or similar systems. See Section 11.1 [Sdtty], page 49.

`no_console`
: Extremely dumb terminal mode: turn off all special input/output processing completely. This allows input or output to be redirected from files.

`lines` *`nlines`*
: The number of lines on the screen. Default is 25. This is only legal in `Sdtty`, and is meaningless on Windows 95 or similar systems.

`no_line_delete`
: Do not use the insert line or delete line cursor control commands. Use this if these commands are not working satisfactorily on your terminal. This is only legal in `Sdtty`, and is meaningless when running Windows.

9.3 Accelerator Key Control

The text of the '`sd.ini`' initialization file, from the line `[Accelerators]` up to the next blank line or the end of the file, consists of key definitions, one per line, or comments. A comment is any line starting with a "pound sign" (`#`). The first item on a non-comment line is a description of the key, and the remaining items are the meaning, exactly as you would type it to `Sd`. The meaning may be any single call, concept, or special command. (To see what special commands are available, type a question mark during a resolve, during program startup, or during normal program operation, or look at the `Sd` menu.)

Keys may have different definitions during program startup, during resolves, and during normal operation.

If the first character of the key description is a plus sign (`+`), that key definition line is meaningful during program startup. For example, the standard definition contains the line

```
+f1    heads start
```

to indicate that function key ⟨F1⟩ means `heads start` during program startup.

If the first character of the key description is an asterisk (`*`), that key definition line is meaningful during resolve searches. For example, the standard definition contains the line

```
*f12   find another
```

to indicate that function key ⟨F12⟩ means `find another` during resolves.

It is perfectly legal to have the same keystroke mean three different things during startup, during resolves, and during normal operation.

After the optional plus sign or asterisk, there may be an optional `s` to mean shift, `c` to mean control, `a` (or `m`) to mean alt (meta), or `ca` to mean control-alt (hold both the control and alt keys while pressing the indicated key).

What follows must be

a letter
: Usable only with `c`, `a`, or `ca`. (Plain and capital letters always have their normal meaning.)

a digit
: Usable only with `c`, `a`, or `ca`. (Plain digits always have their normal meaning, and shift digits are punctuation.)

a function key: f1 through f12
: These may be plain, shifted, control, alt, or control-alt.

a numeric keypad key: n0 through n9
: Usable only with `c`, `a`, or `ca`. (Plain numeric key presses are always equivalent to that digit.)

an "enhanced" key: e1 through e14
: These may be plain, shifted, control, alt, or control-alt.

The "enhanced" keys are encoded as follows:

```
e1     page up
e2     page down
e3     end
e4     home
e5     left arrow
e6     up arrow
e7     right arrow
e8     down arrow
e13    insert
e14    delete
```

So, for example, the line:

```
cae8  u-turn back
```

would define `control-alt-<down-arrow>` to do a '`U-turn back`'.

In specifying a key name, you can use either "m" (for meta) or "a" (for alt) to mean the same thing. (Some people refer to it as the meta key.) Also, if a key is both meta/alt and control, you may list them in either order. Also, you may put hyphens into the key name, and put it in upper or lower case. The command value (the rest of the line) must be in lower case.

Hence

```
c-m-e8  u-turn back
c-a-e8  u-turn back
m-c-e8  u-turn back
a-c-e8  u-turn back
```

mean the same thing as the above example.

If a key is defined to mean a concept, and that concept is not legal at the level at which the program is invoked, the key will still mean that concept. If the key is pressed, the

concept will be used, but it will be considered an off-level concept, and a warning will be printed. See Chapter 13 [Using Off-Level Concepts], page 53.

If a key is defined to mean a call, and that call is not legal at the level at which the program is invoked, that key definition in the initialization file will simply be ignored, unless the level is C4X. Therefore, whenever you change your initialization file, it is a good idea to test it by starting the program at C4X to find out if any warning messages are printed.

9.4 Creating an Intialization File

When `Sd` or `Sdtty` is first installed, no initialization file '`sd.ini`' exists. When you run the program, it will print

```
You do not have a session control file.  If you want to
create one, give the command "initialize session file".
```

After doing that, exit from the program and start it again. The file will have been created, with some sample sessions. It will also contain all of the standard accelerator key bindings and abbreviations. You may edit those.

10 Invoking Sd via Command Line (DOS or UNIX only)

`sd` level [Xt options . . .] [Sd options . . .]
`sd` level `-write_list` filename
`sd` level `-write_full_list` filename

The program is normally invoked with a single argument—the level. This is one of the following: `m` (mainstream), `p` (plus), `a1`, `a2`, `c1`, `c2`, `c3a`, `c3`, `c3x`, `c4a`, `c4`, or `c4x`. The level can also be determined by the selection made when using the "session" feature. If the level is not given, or the program was started by a mouse click from Windows, the program will ask you for it. However it is specified, the level determines the calls and concepts that will be made available, according to our best guess of what the levels mean. Various optional arguments are permitted to control the window system, customize the list of calls used, and set other options. These will be described later.

The call definitions will be read in from the encoded database file '`sd_calls.dat`'. The program will then ponder the database for a few seconds while it determines what calls to put on what menus. Depending on the speed of your computer and the level you selected, this could take from a few seconds to over a minute.

10.1 Command-Line Options

Any of the option keywords that can be placed in the "options" section of the initialization file may be given on the command line instead. See Section 9.2 [Option Control], page 40.

Each keyword has a hyphen in front of it when used on the command line.

Examples:

```
sd -singlespace -maximize plus
sdtty -no_warnings -active_phantoms
```

In addition to giving these options if you invoke the program from the command line, you can place them (and the level also, if you wish) in a Windows shortcut. For example, you can place an icon on your desktop that runs `Sd` at A2, with the `color_by_corners` scheme. Make a copy of the standard shortcut (or make a new shortcut with the "Taskbar" option or the "Settings" operation in the Start Menu), and then use the "Properties" command to add the desired switches at the end of the "Target" line.

There are X resources associated with some options: `Sd.sequenceFile` is equivalent to the `-sequence` switch, and `Sd.databaseFile` is equivalent to `-db`.

If you almost always will be passing the the same value for a command-line switch, you may find it more convenient to set the corresponding resource.

With the X Window System interface, Sd accepts all Xt command-line options. The following are some of the more useful options for use with Sd.

`-rv` reverse video

`-bg` *color*
background color

`-fg` *color*
foreground color

`-bd` *color*
: border color

`-font` *font*
: font used everywhere

`-geometry` *geom-spec*
: geometry in pixels

`-title` *string*
: title for window manager and icon use

`-name` *string*
: name to look up resources under. default: program name.

`-xrm` *resource-line*
: an X resource manager string.

11 Terminal Interface

One version of the program, called `Sdtty`, uses a character-oriented terminal interface.

11.1 Sdtty

The program `Sdtty` uses the completing reader for call, concept, and command entry.

When typing, press ⟨Space⟩ to complete the current word. Type ⟨TAB⟩ or ⟨ESC⟩ to complete as much as possible. Type `C-u` (that is, Control U) to clear a partially-typed line. Type `C-v` (that is, Control V) to clear just the last word.

`Sdtty` has a special command `refresh display` that has no counterpart in `Sd`. It causes a complete clean transcript of the current sequence to be displayed. This can be used if the screen got messed up due to such things as unreliable terminal or modem behavior.

`Sdtty` tries to keep the screen up-to-date efficiently, maintaining a clean transcript of the current sequence on the screen at all times, by using whatever Operating System facilities it thinks are appropriate. If, for any reason, this doesn't work properly for you, you can place the `no_cursor` option in your initialization file, or the `-no_cursor` command-line switch at program startup. See Section 9.2 [Option Control], page 40, and Section 10.1 [Command-Line Options], page 47. This will cause `Sdtty` to treat the computer's display as if it were a dumb typewriter. When started in this way, the part of the transcript that has changed with each command is simply redisplayed. This will inevitably lead to a choppy appearance on the screen or printing device. The `refresh display` command cleans that up.

You can totally disable all special input/output processing by placing the `no_console` option in your initialization file, or using the `-no_console` command-line switch at program startup. This should even make it possible to use the program from a printing device, or to redirect input or output to files.

To deal with different types of display hardware with various screen sizes, the `-lines` *`n`* switch can be given at program startup, as in `sdtty -lines 36 a2`. If this is not given, the size is set from the actual window size if that can be determined (this depends on the operating system software), or 25 otherwise.

11.2 Function Keys

`Sd` recognizes a number of function keys and other special keystrokes. They can be programmed to your personal preferences through the initialization file. If you do not program your personal preferences, you get the following "standard" definitions:

key	normal	shift	control
F1	heads start	sides start	just as they are
F2	two calls in succession	twice	
F3	pick random call	pick concept call	pick simple call
F4	resolve	reconcile	normalize
F5	refresh display	keep picture	insert a comment
F6	simple modifications	allow modifications	centers
F7	toggle concept levels	toggle active phantoms	
F8	<anything>	cut to clipboard	paste one call
F9	undo last call/exit	search/exit program	
F10	write this sequence	change output file	
F11	pick level call	pick 8 person level call	standardize
F12	find another	accept current choice	previous

Additionally, ⟨alt-F4⟩ exits from the program, ⟨alt-F12⟩ means "next" inside a search, and the following special keys are defined:

<home>	resolve	
<end>	write this sequence	(only inside a search)
<insert>	insert a comment	
shift-<up-arrow>	raise reconcile point	
shift-<down arrow>	lower reconcile point	
<left-arrow>	previous	(only inside a search)
<right-arrow>	find another	(only inside a search)

All of these except ⟨F8⟩ execute the command directly, that is, pressing the function key is equivalent to typing the indicated text and then pressing ⟨ENTER⟩. Furthermore, these keys erase any text that you may have typed in on the present line. Therefore, if you define, for example, ⟨alt-S⟩ to mean "swing thru", do not type

```
left ⟨alt-S⟩
```

It would erase the word `left`. Instead, type in the '`left`' concept by itself:

```
left ⟨ENTER⟩
⟨alt-S⟩
```

The key ⟨F8⟩ for '`<anything>`' is simply equivalent to entering that text without pressing ⟨ENTER⟩. This key is present only for compatibility with past usage. It is almost never necessary to type "<anything>". Instead, type the substituted call in brackets. See Section 2.3.1 [Call Variations], page 10.

You can program the function keys to your own preference by placing an `[Accelerators]` section in your '`sd.ini`' initialization file. If you do not have an initialization file, or you have one but there is no `[Accelerators]` section in it, the key bindings will be set to the default bindings shown above. If you have in initialization file with an `[Accelerators]` section, the default bindings will *not* be used. The contents of the `[Accelerators]` section will be used instead. See Section 9.3 [Accelerator Key Control], page 44, for information about setting up an `[Accelerators]` section in your initialization file.

11.3 Sdtty on Unix

The Unix version normally uses the *curses* display access mechanism. It should make effective use of the screen-editing features of the terminal (VT-100 or emulation of same, or whatever) to keep the screen updated. However, this requires that all of the system facilities for dealing with terminals (`TERM` environment variable, `terminfo` database, `stty` behavior, etc.) be working properly. If, for any reason, this doesn't work for you, you can turn the *curses* mechanism off completely by placing the `no_cursor` option in your initialization file, or using the `-no_cursor` command-line switch at program startup. You can also place the `no_console` option in your initialization file, or use the `-no_console` command-line switch at program startup, to disable all special input/output processing completely. This should even make it possible to use the program from a printing device, or to redirect input or output to files.

When using *curses*, it normally tries to use the insert/delete line capabilities of your terminal device to update the screen more efficiently. For certain terminal or modem configurations, this may be counterproductive. If so, you can turn this feature off by giving the command-line switch `-no_line_delete` at program startup.

12 Linguistic Idiosyncrasies

In a number of cases, the program behaves in a way that could be considered idiosyncratic, peculiar, or simply wrong. Some of these cases are admittedly bugs or shortcomings of the program. Others are inevitable consequences of what the program is trying to do.

The syntax, semantics, and general structure of the square dance calling language is fairly regular and precise—much more so than, for example, the English language. That is, the correspondence between dancer actions and verbal phrases is fairly regular. However, there are exceptions. A very simple example of this is the fact that callers generally say '`quarter top`' and '`half the top`', using the word "the" in one case but not the other.

The program handles these exceptions by calling in an occasionally stilted but unambiguous computer-ese dialect. This dialect is intended to be very close to the words you would use when calling, but it is sometimes different. The rationale for this is that it is too hard to make the program always use the words that a caller would use. When writing sequences, be aware that the dancers do not necessarily know this computer-ese dialect. Use the words that you think are correct and natural. Your judgement is much better than any program's judgement can ever be.

There are some innocent-looking things that the program can generate for which appropriate words do not exist. You must not write sequences containing such things, because you won't be able to call them. For example, from boy-boy-girl-girl waves, it is perfectly straightforward for a caller to say '`boys hinge`'. From boy-girl-boy-girl waves, '`boys hinge`' is meaningless, but the program accepts '`boys do your part, hinge`'. It doesn't know that this is an unacceptable thing to call. Don't do it.

In general, if you don't think you can clearly express to the dancers what you want, don't call it.

A number of concepts are particularly vulnerable to this phenomenon. Examples are `precede it by`, `follow it by`, the fractional concepts like `1/2`, `1-1/2`, and `3 times`, and the calls that designate certain people. See Section 15.10 [Designating Certain People], page 93.

Callers often use phrases such as '`finish ...`' or '`like a ...`' in imprecise ways. Sd uses a very precise definition for these phrases. See Section 15.5 [Miscellaneous Concepts], page 73.

13 Using Off-Level Concepts

Because the program is so literal-minded, it sometimes doesn't understand how to do things that are in fact very easy for callers to express and for dancers to do. In some cases, the necessary sophistication lies in some concept that is legal only at high challenge levels.

When such a situation arises, you can tell the program to permit the use of concepts that would not normally be legal.

The `toggle concept levels` command toggles (turns on or off) the state of this option. See Section 8.9 [Changing Modes], page 36. It can also be turned on when the program begins, either by giving a command-line option (see Section 10.1 [Command-Line Options], page 47) or through the use of the initialization file (see Section 9.2 [Option Control], page 40).

If you have a line of 6 (or 8) people, and you want them to do a '`1/2 tag`', use the `3x3` (or `4x4`) concept. Of course, if you were calling Mainstream, you wouldn't say '`3x3 1/2 tag`'. You would presumably say something like '`line of 6 in the center, 1/2 tag`'.

If you want people to do something (e.g., '`diamond circulate`') around the outside, you may need to use the `disconnected` concept. For example, if you had waves with the boys on the ends, and you had just done a '`1/2, acey deucey`', the boys have a big diamond around the outside. You can't just do a `BOYS diamond circulate`, because the `<ANYONE>` concept is extremely fussy. See Section 15.10 [Designating Certain People], page 93. The way to do this is with `BOYS DISCONNECTED diamond circulate`. This concept is only legitimate at C2, so you will need `toggle concept levels` below that. You would presumably say something like '`boys diamond circulate around the outside`'.

Whenever you use an off-level concept, a warning will be printed in the transcript. You should read such a card using whatever words are appropriate to get the dancers through the action that you intend.

14 Call Notes

Some calls require explanation of how Sd interprets them. This section documents these calls.

14.1 Sweeping Direction

'`Sweep 1/4`', '`with the flow`', and '`by golly`' are somewhat unsophisticated in the way they calculate the sweeping direction. The program infers the sweeping direction from the roll direction of the preceding call. This is known to work for the common calls such as *recycle*, but may do something tasteless if used with an unusual call.

14.2 ANYTHING and the ANYONE roll

The call '`<anything> and the <anyone> roll`' re-evaluates people's positions before deciding who should roll. So, for example, if you say '`[swing thru] and the centers roll`', it will be the *new* centers that will roll. We believe that this is usually the more natural thing to do, but it is not automatically assumed. Therefore, you may need to make that clear to the dancers by saying something like '`swing thru and the new centers roll`'. Of course, if you have the girls roll there will be no problem.

14.3 Roll after Exchange the Gears

The program considers '`roll`' to be of limited use after '`spin chain and exchange the gears`' if using the true Callerlab definition. This is because that definition appears to be a description of something that gets people to the correct ending point, rather than a description of a course of action that dancers actually follow when they do the call. Instead, the program defines the call in terms of '`exchange the diamond 3/4`' and '`flip the interlocked diamond`', which appears to be the definition that is more likely to be used by people who actually care about accurate definitions. Because of this, calling '`roll`' after this call will result in a 8-chain formation. If the actual Callerlab definition had been used, the result would have been some kind of T-bone formation.

Users who call this need to be aware of what they are getting into.

Some people might consider this a bug.

14.4 C1 Single Rotate

'`Single rotate while the others`' is intended to be called to the heads or sides from a squared set. It performs the common C1 usage of this call. It is nonsensical in other contexts.

14.5 Spread

The term *spread* can mean different things. It is four calls in this program.

Use the call '`spread`' after calls such as *follow your neighbor*, or from the starting double pass thru (or similar setup) that is obtained after calls like *wheel and deal* or a sequence started with *heads star thru*.

There is also a call '`<anything> and spread`', which simply does the <*anything*> followed by a *spread*, to make the printout look nicer by having the calls appear on the same line. It is intended for things like '`[follow your neighbor] and spread`'. You must type the <*anything*> call in square brackets. Do not use this with something like *heads star thru*. That is, do not enter:

```
heads start
[star thru] and spread
```

That would attempt to have only the heads do both the *star thru* and the *spread*. All 8 people must do the *spread*. You must enter:

```
heads start
star thru
spread
```

and get the calls on separate lines

There is also a call '`<ANYONE> spread`'. It is typically used from columns, to get the designated people to slide away from their partner into a butterfly, "O", or whatever. It can also be used to get selected people (who must be adjacent) to do a '`follow your neighbor`' type of spread.

There is also a call '`<anything> and the <anyone> spread`', which combines the previous two calls. It does the <*anything*> followed by having the selected people *spread*. For example, you could enter

```
[motivate] and the boys spread
```

(There is also the call '`wheel and spread`', which is just the special case call on the Plus list.)

14.6 Centers Back Away

This call is intended for use in sequences that begin with heads or sides doing a '`ladies chain`' or '`right and left thru`' or something similar. After they do such a call, they will remain in the center. To have the same people do the next call (for example, '`pass the ocean`' or '`star thru`'), just use the '`centers`' concept. If you want the *inactive* dancers to come in and do the next call, use the call '`centers back away, others come in and <ANYTHING>`'. So, for example, you might enter:

```
heads start
ladies chain
centers back away, others come in and [star thru]
```

You might read this as "head ladies chain, then the sides move in and star thru."

14.7 ANYONE Promenade

These calls are intended for use in sequences that begin with heads or sides promenading halfway around the set. The calls are

```
promenade halfway, come in to the middle and <ANYTHING>
```

and

```
promenade halfway, while the others <ANYTHING>
```

You can substitute `1/4`, `1/2` or `3/4` for the word `halfway` in either call.

In each case, start the sequence with the `heads start` or `sides start` command, and then issue one of the calls listed above. Ignore the fact that the displayed setup will look strange before the promenade. In `Sdtty`, you can type the '`<ANYTHING>`' call in brackets, as in `promenade halfway, while the others [square thru 4]`. The '`<ANYTHING>`' call will be done by the appropriate people in the center of the set.

14.8 Up to the Middle

When the program starts a sequence with the `heads start` or `sides start` command, there is an implicit "up to the middle" in the first action. This only applies to the first call. If a sequence starts with something else, such as '`4 ladies chain`', '`4 ladies chain 3/4`', '`head/side ladies chain to the right`', or '`all 4 couples right and left thru`', the result will be a squared set. To do the next call, one typically needs to have the heads or sides go up to the middle. Since this is not the first call, the program won't automatically do it, and you need to do it explicitly.

There is a call '`<ANYONE> press ahead`' for this purpose. The call is officially on the C2 list, but `Sd` and `Sdtty` allow it at Mainstream. Have the heads press ahead, or whatever is appropriate. If the heads and sides are not positioned consistently (as, for example, after an '`all 4 ladies chain 3/4`'), use the designation `headliners` or `sideliners`. These designations refer to the people currently facing head or side walls, respectively. `Sd` and `Sdtty` recognize them at Mainstream. When actually calling in such a situation, a common phrase to use is "at the heads," as in "all 4 ladies chain 3/4; at the heads, square thru 2." The way this would be entered to the program is as follows:

```
just as they are              (instead of heads or sides start)
all 4 ladies chain 3/4
headliners press ahead
centers square thru 2         (you must identify the centers)
```

14.9 ANYONE Cloverleaf

The call '`cloverleaf`' is expected to be done from a completed double pass thru setup, and has everyone doing the call.

There are some other uses of this call, some sanctioned by the Callerlab definitions and others simply in common usage. These involve just having four people (who must be looking out, but do not necessarily have to be ends) do the call. The call '`<ANYONE> cloverleaf`'

does this. You must use an appropriate designator, such as '`ends cloverleaf`'. An example of this, in which the designated people are not ends, is:

```
heads start
pass thru
heads cloverleaf
```

In this example, the sides would step into the center.

The call `<ANYONE> cloverleaf while the others <ANYTHING>` is similar, but has the inactive people, after they step into the center, do the other call. It is in many cases identical to the A1 call '`clover and <ANYTHING>`', but the latter call is more restrictive.

14.10 And N/4 More

The calls '`and 1/4 more`' and '`and 1/2 more`' are intended to be used after a courtesy turn, such as a '`right and left thru`'. They cause the couples to turn, as a couple, that additional amount. Callers sometimes use phrases like "courtesy turn full around" to describe the action known to `Sd` as '`and 1/2 more`'.

There are also calls '`<ANYTHING> and 1/4 more`' and '`<ANYTHING> and 1/2 more`', which can be typed to `Sdtty` with the actual '`<ANYTHING>`' call in brackets, as in `[right and left thru] and 1/2 more`.

14.11 Ends Divide

This is a call that should be used when you want the outsides to move along until they are facing each other. For example, after '`heads swing thru`', you might say '`ends divide`', and then have them '`star thru`'. Note that '`ends divide`' is the name of the call. It is not an application of the '`ends`' concept. You must type it on one line, and you may not substitute another designator, such as '`sides`', for the word '`ends`'.

There is also a call '`ends divide and <anything>`'. This call can have the "anything" subcall entered in the usual way, by placing it in brackets. So, for example, you might type:

```
heads start
swing thru
ends divide and [touch 1/4]
6x2 acey deucey
```

You might read this as "heads swing thru, while the sides divide and touch 1/4."

14.12 Little, Little More, and Plenty

These are problematical calls when the outsides are in line-like, rather than column-like, orientation. This situation arises, for example, when the starting setup is twin diamonds. The Callerlab C1 and C2 definitions state that in this case the outsides counter rotate just as they are, without first quartering in some direction. However, many high-level callers have adopted a definition that says that the outsides *always* quarter right (or take whatever direction is given) before counter rotating. Sd uses this formulation. To handle all the possibilities, there are 3 varieties of the call '`little`'—just '`little`', '`little, ends face`

<DIRECTION>', and 'little, ends go as you are'. Use the last one when the dancers are in diamonds and you want the outsides to counter rotate directly. Whether or not you say "ends go as you are" is up to you. The calls 'little more' and 'plenty' have analogous behavior.

14.13 Flip Back

This call is a special case of the '<ATC> back to a wave' class of calls, and is a recognized C1 call. You should use this name rather than the potentially misleading 'flip back to a wave'.

14.14 Face DIRECTION

The call 'face <DIRECTION>' does not appear to be a genuine Callerlab-sanctioned call. It is intended to be used, for example, after 'tag the line', to give the direction that you wish to have the dancers face. So, for example, you could enter the calls 'tag the line' followed by 'face in' to get the action that is commonly expressed as "tag the line, in." Note that the A2 designators 'zig-zag', etc., are among the permissible directions.

14.15 Adjust Alamo to Other Pairing

The program's notion of where people are in an alamo ring is rather imprecise. It has them paired up on "O" spots, with the people in each pair precisely facing head walls or side walls, rather than uniformly spread around the circle. Its notion of how alamo calls work is based on this flawed notion, in that it can only deal with one kind of pairing. Typically, the people must be paired in right-handed miniwaves. (For the call *break the alamo*, the selected people must be together.) If the pairing is not correct, you can issue the pseudo-call *adjust alamo to other pairing*. This will make the program move people around one position on the computer screen, so that the pairs will have the other handedness. You of course do not read this line when calling—the dancers will be able to figure out who needs to work with whom on the next call.

Typical instances in which this is needed are calling 'swing the fractions' when the four miniwaves of the alamo ring do not have the correct handedness, and calling things like 'all 8 spin the top' after a 'dixie grand'.

14.16 Going Through Impossible Situations

Under certain circumstances, people who collide in the same spot during a call will take right hands with each other, according to well-known rules governing this type of occurrence. Other than that, the program does not allow "impossible" intermediate situations in which multiple people occupy the same spot.

In some cases, you may want to call something that sends people through a momentary impossible situation, for example calling something like 'everyone do your part, split trade circulate twice' when people are in normal waves. The first 'split trade

circulate' is impossible, but if you tell the dancers to do their own part and not worry about the collision after the first one, they can do it.

The program doesn't like to do this. Just calling twice split trade circulate won't work, because the program will have multiple dancers on the same spot after the first one. You can trick the program into doing it anyway, by using the <ANYONE> do your part concept. Enter something like heads do your part, twice split trade circulate while the others twice split trade circulate. Exactly who has to be designated depends on the the setup.

You can also use this method to get everyone to do their part of calls that are illegal in the existing setup. For example, flip back is illegal from facing lines. If you really want people to do it anyway, with everyone doing their part, you can enter beaus do your part, flip back while the others flip back.

14.17 Changing between a 4x4 matrix and a Phantom Setup

Some calls leave the setup in a "C1 phantom" formation, and some calls leave the setup in a nearly-identical 4x4 matrix formation. Occasionally one wishes that the program had made the other choice. For most applicable calls and concepts it doesn't matter, because the program is generally forgiving about these two formations. (For example, the 'split phantom waves' concept will work from a phantom formation.) On those occasions when you need to change from one formation to the other, you can use the 'phantom' concept or the '4x4 matrix' concept with the call 'nothing'.

14.18 Matrix Calls in 1/4 Tags

The program believes that, in 1/4 tags and diamonds, the outsides are not located in really precise positions. There is a good reason for this—there are real problems with the positioning in 1/4 tags.

Because of this, matrix ("space invader") calls, like 'press' and 'truck', are not permitted. You can override this prohibition by using the 3x4 matrix concept. So, for example, after 'heads pass the ocean', you could enter 3x4 matrix side girls truck, obtaining Z's. You may or may not need to say anything to the dancers to persuade them to get the effect you want.

You may also use concepts like 'triple lines' when in a 1/4 tag. The program (and presumably the dancers also) will place the outsides in the center of the outer triple lines.

For some calls, the '12 matrix' concept will also be helpful.

15 Concept Notes

Most concepts have clear and precise meanings and do not need any further discussion here. However, for a significant number of concepts, the distinction between precisely defined, universally understood terminology, and "common sense" description of what you want the dancers to do, becomes rather blurred. When using computers to write choreography, a lot of questions of the form, "How do I get the program to do this?" arise. For that reason, a complete list of available concepts will be given here. For many of them, their meaning is clear, and nothing further will be said. For the troublesome ones, we will attempt to describe just what the program understands them to mean.

For many of the less straightforward cases, the words that the program uses are not necessarily the words that you would use to get the dancers to do the equivalent thing. You must always use your own judgement. No claim is made that the program's notion of what the words literally mean is the universally accepted definition of the concept.

Be aware that those concepts that take two calls, such as `checkpoint`, `<ANYONE> do your part (while the others)`, or `two calls in succession` must be entered by themselves. That is, you must press ⟨ENTER⟩ after typing any of these concepts. You can't type

checkpoint ah so by recycle ⟨ENTER⟩

nor

boys trade (while the others) u-turn back ⟨ENTER⟩

You must type

checkpoint ⟨ENTER⟩
ah so ⟨ENTER⟩
recycle ⟨ENTER⟩

or

boys (while the others) ⟨ENTER⟩
trade ⟨ENTER⟩
u-turn back ⟨ENTER⟩

The concepts to watch out for are labeled "[must be entered by itself]" in the following lists.

Just what constitutes a "concept" and what constitutes a mere "variation" of a call is not always clearly delineated. We list below only those things that the program considers to be "concepts." There are many other variations of calls, such as *square chain thru to a wave* that are not listed here. In general, variations of this sort can be obtained simply by clicking on them or typing them. They should be visible on the call menu, and, if you type a question mark while typing the call name, these variations should be displayed.

In the concept names, "C/L/W" has the usual meaning of "columns, lines, or waves." Extending that terminology, "B" means boxes, and "D" means diamonds.

"1T" means 1/4 tags, "3T" means 3/4 tags, "1L" means 1/4 lines, and "3L" means 3/4 lines. A 1/4 line or 3/4 line requires that the centers form a 2-faced line, and the ends be looking in or out, respectively. A 1/4 tag or 3/4 tag requires that the centers form some kind of line (not necessarily a wave), and the ends be looking in or out, respectively.

"GT" means "general 1/4 tags", in which the ends could individually be facing either in or out, and the centers form some kind of line.

"DS" means "diamond spots", that is, diamonds or general 1/4 tags in which no assumption is made about anyone's facing direction.

In all concepts that can specify either lines or waves, the "waves" version has an implicit "assume waves" operation in it. Under normal circumstances, this means that the live people can make use of the assumption of waves in order to decide how to start the call. For example, in normal columns, `split phantom waves in roll circulate`—which is equivalent to `split phantom lines, assume waves, in roll circulate`—is legal, because the phantom in-roller's location can be deduced from the wave assumption. Whether the dancers will be appreciative of your calling that is another matter.

The call `split phantom lines in roll circulate` is illegal from columns because the facing direction of the phantom in-roller is unknown.

If "active phantoms" are used, either because the `with active phantoms` concept is used after the phantom wave concept or because the `toggle active phantoms` command was given, the phantoms will work throughout the entire call, based on the initial facing direction in waves.

The concepts with the word `diamonds` have an implicit `assume general diamonds` operation in them. This means that everyone must think they are individually in some kind of diamond, with their right or left hand toward the center. You may give an explicit `assume` concept to make the setup more specific if you wish, as in `split phantom diamonds` followed by `assume interlocked diamonds`.

The concepts with the words `1/4 tags` or `3/4 tags` require some kind of line in the middle, and the ends as a couple facing in or out, respectively. Hence, for example, the `split phantom 1/4 tags` concept includes 1/4 tags, 1/4 lines, or any kind of line in the center, as long as the ends are facing in. You may give an explicit `assume` concept to make the setup more specific if you wish, as in `split phantom 1/4 tags` followed by `assume 1/4 tags` (to require a wave of either handedness in the center) or `assume right 1/4 tags` (to require a right-handed wave).

The concepts with the words `1/4 lines` or `3/4 lines` are similar, but require that the centers form a 2-faced line. An explicit `assume` concept, such as `assume left 1/4 lines`, may be used.

The concepts with the words `general 1/4 tags` have an implicit `assume general 1/4 tags` operation. This means that the centers must think that they are in some kind of line, and the ends must be in some kind of couple or miniwave. The word *general* means that the in-or-out facing direction of the outsides is unimportant, but they must be facing in or out, not sideways as in diamonds. There are no `general 3/4 tags` concepts, because the `general 1/4 tags` setup does not distinguish between ends facing in or out.

The concepts with the words `diamond spots` make no assumptions at all about facing direction. The people can be in any kind of general diamond or general 1/4 tag. Use these when people's facing directions are bizarre.

Precisely what an assumption means is determined by whether the "active phantoms" mode is turned on, and is discussed in Section 15.7 [Assume Waves], page 86.

15.1 Phantom Concepts

These concepts involve picking out virtual setups, which may include phantoms, from a real formation that is larger than eight people.

In all cases outboard phantoms are added as required at the start of the call and removed where possible at the conclusion of the call.

`split phantom C/L/W/B/D/1T/3T/1L/3L/DS/GT`
`interlocked phantom C/L/W/B/D/1T/3T/1L/3L/DS/GT`
`phantom C/L/W/B/D/1T/3T/1L/3L/DS/GT`

These are straightforward.

`12 matrix split phantom C/L`
`12 matrix interlocked phantom C/L`
`12 matrix phantom C/L`

These are done from a 3x4. The setup is separated into parallel 3x2's, and the call is done in each of those setups.

`twin phantom tidal C/L/W`
`triple tidal C/L/W`

These are done from a 2x8 or 3x8, respectively, and form two or three parallel 1x8's with phantoms, in each of which the call is done independently.

The program uses the term "tidal" to mean a 1x8. You may prefer to use the word `grand`.

`twin phantom C/L/W of 6`
`triple C/L/W of 6`
`quadruple C/L/W of 3`
`quadruple C/L/W of 6`

These are done from a 2x6, 3x6 or 4x6, respectively, and form two, three, or four parallel 1x6's with phantoms, in each of which the call is done independently.

`split phantom C/L/W of 6`
`interlocked phantom C/L/W of 6`
`phantom C/L/W of 6`

These are done from a 4x6.

`triple C/L/W/B/D/1T/3T/1L/3L/DS/GT`
`quadruple C/L/W/B/D/1T/3T/1L/3L/DS/GT`
`quintuple C/L/W/B`
`sextuple C/L/W/B`
`triple 1x4s`
`triple lines or boxes`
`triple boxes or lines`
`triple lines or diamonds`
`triple diamonds or lines`
`quintuple 1x4s`
`sextuple 1x4s`

These are straightforward. In the concepts that name two setups, such as "triple lines or boxes", the outer two formations must be the same. In these concepts,

"lines" means any 1x4. Use the correct term when calling. You may name the two formations in either order.

`center triple C/L/W/B/D` (singular)
`outside triple C/L/W/B/D`
: These are done from a setup that can be construed as having a center 4-person setup with two other 4-person setups on the outside. The center setup can be different from the outside setups, and reasonable independent shape-changers are allowed.

  A suitably populated 4x4 matrix can be construed as a center triple box surrounded by outside triple lines or columns.

`center tidal C/L/W` (singular)
: This picks out a 1x8 from a larger setup. It may be useful for certain types of `own` operations.

`center Z` (singular)
`outside triple Z's`
: These are straightforward.

`center phantom C/L/W/B/D/DS/GT`
`outside phantom C/L/W/B/D/DS/GT`
: These are done from a setup that can be construed as having a center 8-person setup with two other 4-person setups on the outside. The center setup can be different from the outside setups, and reasonable independent shape-changers are allowed.

  A common case of this is a 4x4. The center phantom lines or columns can do an 8-person call. The outer phantom lines or columns can do independent 4-person calls.

`12 matrix center phantom C/L`
`12 matrix outside phantom C/L`
: These are done from a setup that can be construed as having a center 6-person setup with two other 3-person setups on the outside. The center setup can be different from the outside setups, and reasonable independent shape-changers are allowed.

`center triple twin C/L/W`
`outside triple twin C/L/W`
: These are done from a 4x6 setup. The center 2x4, or the outside 2x4's independently, do the call.

`center triple twin C/L/W of 3`
`outside triple twin C/L/W of 3`
: These are done from a 3x6 setup. The center 2x3, or the outside 2x3's independently, do the call.

`crazy phantom C/L/W/B/D/DS/GT`
`reverse crazy phantom C/L/W/B/D/DS/GT`
`<N/4> crazy phantom C/L/W/B/D/DS/GT`
`<N/4> reverse crazy phantom C/L/W/B/D/DS/GT`
`crazy offset C/L/W`
`reverse crazy offset C/L/W`
`<N/4> crazy offset C/L/W`
`<N/4> reverse crazy offset C/L/W`
`crazy diagonal boxes`
`reverse crazy diagonal boxes`
`<N/4> crazy diagonal boxes`
`<N/4> reverse crazy diagonal boxes`
`crazy Z's`

`reverse crazy Z's`
`<N/4> crazy Z's`
`<N/4> reverse crazy Z's`
These are straightforward.

`triple C/L/W/B/D/1T/1L working together`
`triple C/L/W/B working apart`
`triple L/W/B working forward/backward`
`triple C/B/1T/1L working right/left`
`triple C/L/W/B working clockwise/counterclockwise`
`triple tidal L/W working forward/backward`
`triple tidal C working right/left`
`quadruple C/L/W/B/D/1T/1L working together`
`quadruple C/L/W/B working apart`
`quadruple L/W/B working forward/backward`
`quadruple C/B/1T/1L working right/left`
`quadruple C/L/W/B working clockwise/counterclockwise`
`quadruple C/L/W/B/D/1T/1L working toward the center`
These are straightforward.

`triple diagonal C/L/W`
`triple diagonal L/W working forward/backward`
`triple diagonal columns working right/left`
These are done from a 6x6 matrix. For example, in "blocks", the main diagonal and the two adjacent diagonal lines of 4, each of which has only centers occupying it, comprise the triple diagonal lines. These concepts are quite obscure.

`triple twin C/L/W`
This is done from a 4x6 setup. Three parallel 2x4 setups are formed, and the call is done independently in each of them. You can use '`mystic`' and '`invert mystic`' with them.

`triple twin C/L/W of 3`
This is done from a 3x6 setup. Three parallel 2x3 setups are formed, and the call is done independently in each of them. You can use '`mystic`' and '`invert mystic`' with them.

`triple staggered boxes`
: This is done from a 2x12 "zipper" setup.

`concentric triple boxes`
`concentric quadruple boxes`
: These are done from a 2x6 or 2x8 respectively. The concentric boxes work independently.

`twin phantom D/1T/3T/1L/3L/DS/GT`
: These are two sets of parallel diamonds or 1/4 tags (that is, setups in which you could call `6x2 acey deucey` or `scoot and plenty`) that are next to each other along the opposite orientation to the orientation given by `split phantom diamonds`. That is, there is a 2x2 matrix of diamonds or 1/4 tags. The diamonds in each parallel pair work with each other, but the pairs work independently.

`twin phantom point-to-point D/DS`
: Similar to the above, but the diamonds work with each other along the point-to-point axis.

`twin phantom I's`
`twin phantom bones`
: These concepts are the same. The setups are end-to-end.

`twin phantom riggers`
`twin phantom bats`
: These concepts are the same. The setups are end-to-end.

15.2 Tandem or Couples Concepts

as couples	couples twosome
tandem	tandem twosome
siamese	siamese twosome
melded as couples	melded couples twosome
melded tandem	melded tandem twosome
melded siamese	melded siamese twosome
couples of 3	couples threesome
tandems of 3	tandem threesome
siamese of 3	siamese threesome
couples of 4	couples foursome
tandems of 4	tandem foursome
siamese of 4	siamese foursome
boxes are solid	boxsome
diamonds are solid	diamondsome
Y's are solid	Y-some
Z's are solid	Z-some
skew	skewsome
	gruesome twosome
gruesome as couples	gruesome couples twosome
gruesome tandem	gruesome tandem twosome
<ANYONE> are as couples	<ANYONE> are couples twosome
<ANYONE> are tandem	<ANYONE> are tandem twosome
<ANYONE> are couples of 3	<ANYONE> are couples threesome
<ANYONE> are tandems of 3	<ANYONE> are tandem threesome
the couples are solid	the couples are twosome
the tandems are solid	the tandems are twosome
the couples of 3 are solid	the couples of 3 are threesome
the tandems of 3 are solid	the tandems of 3 are threesome
inside triangles are solid	inside triangles are threesome
outside triangles are solid	outside triangles are threesome
in point triangles are solid	in point triangles are threesome
out point triangles are solid	out point triangles are threesome
wave-based triangles are solid	wave-based triangles are threesome
tandem-based triangles are solid	tandem-based triangles are threesome
<ANYONE>-based triangles are solid	<ANYONE>-based triangles are threesome
3x1 triangles are solid	3x1 triangles are trianglesome

These are straightforward.

Any of the above twosome/threesome/whateversome may also be "<N/4> whateversome" or "whateversome <N/4> solid."

`as couples in a 1/4 tag`
`as couples in a 3/4 tag`
`as couples in a 1/4 line`
`as couples in a 3/4 line`
`as couples in point-to-point diamonds`
`as couples in a tidal line`
`as couples in a tidal column`
`as couples in a tall 6`

These may be used with '`tandem`' instead of '`as couples`', and may of course be used with all of the twosome variations. Each of them is actually the juxtaposition of two concepts—an '`as couples`' or '`tandem`' concept followed by a special '`in a 1/4 tag`' (or whatever) concept. This actually doesn't matter. Just type it.

The concepts referring to 1/4 tag-like setups are generally intended to be used from tidal lines (couples) or parallel waves or lines (tandem). They cause the live dancers to become the centers of the resulting 1/4 tag or 1/4 line. They may also be used when the live people are partially or entirely on the outside. If you want to instantiate outside phantoms with facing directions other than those provided, use the '`16 matrix of parallel diamonds`' concept to create the spots, and then use whatever '`assume`' concepts you need. For example, from a tidal 2-faced line, you could say:

```
  16 matrix of parallel diamonds siamese twosome
        diamond circulate
```

or

```
  16 matrix of parallel diamonds siamese twosome
        assume normal diamonds 6x2 acey deucey
```

or

```
  16 matrix of parallel diamonds
        assume normal interlocked diamonds
        siamese twosome 6x2 acey deucey
```

The '`as couples in a tall 6`' is intended to be used from parallel 2-faced lines. It puts a phantom couple in front of each lead couple, making two wave-based triangles.

MxN As Couples

The '`tandem`' or '`as couples`' concepts, and their twosome and fractional twosome/solid variants, may be preceded by a modifier such as '`4x4`' or '`2x1`'. These cause people to be grouped as indicated. '`4x4 tandem`' is equivalent to '`tandems of 4`'. '`3x1 couples twosome`' directs 3 people (determined by the usual rules—whichever 3 people face the same way if that determines it, otherwise the 3 in front or the 3 on the right) to work as a threesome while the other person works alone. When calling, concepts like this are often expressed in the form "threesome by one" or "one by tandem of 2".

When only two people are being grouped, it is also possible to designate them explicitly. '`2x1 couples twosome`' can often be expressed equivalently as something like '`girls are couples twosome`'.

Phantom As Couples

Any of the tandem or couples concepts may be used with the `phantom` concept in front of it. There are no concepts like `phantom tandem` listed separately—you must use the `phantom` concept followed by the concept that you want. Of course, when using `Sdtty`, this issue doesn't make any difference—you can just type

phantom tandem swing thru ⟨ENTER⟩

When using a *melded* concept, it comes before the word *phantom*; for all others the word *phantom* is first.

The *phantom tandem* and *phantom as couples* concepts expand the people into a 4x4 matrix whenever possible. When that is not possible, a 2x8 matrix is formed. You can use the `2x8 matrix` concept in order to force this. For example, from normal columns, `phantom tandem` puts all the real people in tandem with each other, and makes them the centers of *lines*. You could call a `ferris wheel`, but not a `checkmate the column`, from this setup. If you wanted the pairs of real people to be centers of columns, so that you could call `checkmate the column`, use `2x8 matrix phantom tandem checkmate the column`. Concepts with the word "gruesome" always form a 2x8, and require that the people be paired in a direction parallel to that 2x8. The `gruesome twosome` concept is the same as `gruesome couples twosome`, with the additional requirement that the resulting virtual setup be waves. It appears to be the only concept in that family that is widely accepted.

15.3 Distorted Setup Concepts

`stagger`

`big block (any kind of lines)`
`big block waves (this means ``assume waves'')`
The preceding three encompass any kind of distortion in the lines or columns.

`O`

`butterfly`
`mini-O`

`mini-butterfly`
These are straightforward.

`OX` This is an unsymmetrical distorted column that is part O and part 'X' (butterfly.)

`stairstep C/L/W`
A "stairstep" is a 4x4 matrix occupied so that everyone has exactly one live person on their left or right. It could typically be obtained by doing a `split`

`phantom columns split circulate` from normal columns. You may wish to use the `distorted C/L/W` concept instead.

`ladder C/L/W`
: A "ladder" is a 4x4 matrix occupied so that everyone has exactly one live person in front or in back. It could typically be obtained by doing a `split phantom waves split circulate` from normal waves. You may wish to use the `distorted C/L/W` concept instead.

`offset C/L/W` (plural)
: An "offset" C/L/W is a pair of lines or columns that has been sheared in the middle by either 50% or 100%, making a 3x4 or 4x4 matrix, respectively. A 4x4 matrix offset line or column is sometimes referred to as "clumps." All of the live people work together across the shear line. If a shape-changer is called, the setup may become a 2x8 matrix, that is, a 100% offset parallelogram.

`offset C/L/W` (singular)
: This is used in a 2x4 setup in which the live people make a 1x4 that has been sheared in the middle,

`Z C/L/W`
: This is a pair of lines or columns that have been distorted by having the centers or the ends move one full matrix spot, to make a 4x4 matrix, or by having centers and adjacent ends move 1/2 matrix spot in opposite directions, to make a 3x4 matrix. The 4x4 matrix version can be formed by having either the centers or the ends press ahead from waves. The 3x4 matrix version can be formed by having everyone 1/2 press ahead from waves.

`distorted C/L/W`
: This encompasses any kind of distortion of a pair of lines or columns into a 3x4, 4x4, 2x6, or 2x8 matrix.

`diagonal C/L/W` (singular)
`<ANYONE> in your diagonal C/L/W` (singular)
: These require a "blocks" setup. They specify the "long diagonal", recognized as either a line or column. You may specify who the people are (heads, boys, etc.) or not, as you choose.

`diagonal C/L/W` (plural)
: These are done from a 4x6. The setup has 8 real people, doing an 8-person call. The setup could be formed from stairsteps by having the ends press ahead or back such that each line is diagonal and straight. These concepts are quite obscure.

`diagonal C/L/W of 3` (plural)
: These are done from "blocks." The two lines or columns work together in a 2x3 matrix, doing a 6-person call.

`staggered C/L/W of 3` (plural)
: These are done from a 4x4 matrix occupied by only 6 people. Hence the "ignore" concept must be used to remove 2 people from the setup.

`C/L/W of 3` (plural)
: The lines or columns are not distorted. They are simply picked out from the formation.

`distorted tidal C/L/W` (singular)
: These are done from a 2x8 matrix in which the real people form some kind of distorted 1x8.

`offset tidal C/L/W` (singular)
: These are done from a 2x8 matrix in which the real people form two 1x4's joined with an offset in the center.

`offset 1T/3T`
: These are done from a suitably populated 4x4. The offset amount is presumed to be 50%. The "shear line" of the setup lies between the two single 1/4 tags.

`offset split phantom boxes`
: This is done from a 3x8 formation that can be identified as two 2x4's joined with a 50% offset in the center.

`parallelogram triple boxes`
`distorted C/L/W` (singular) `of 6`
: These are done from a center diamond with offset pairs of people on the outside.

`stagger D/1T/3T/1L/3L/GT`
`diagonal D/1T/3T/1L/3L/GT`
: These are done from "blocks."

`distorted D/1T/3T/1L/3L/DS/GT`
: These are done from triple diamonds in which the points are offset.

`double bent tidal C/L/W` (singular)
: These are done from a 1x8 in which the two people at each end are "bent" 90 degrees from the center 4.

`<ANYONE> in your double bent C/L/W` (singular)
: These are done from a variety of setups, in which the designated people form a recognizable "double bent" setup.

`bent C/L/W/B` (plural)
: These are done from suitable setups, including unsymmetrical 4x4's occupied as "clumps".

`<ANYONE> in your trapezoid`
: This is typically used in unsymmetrical setups.

`<ANYONE> in your distorted C/L/W/B/D` (singular)
`<ANYONE> in your diagonal box`
`<ANYONE> in your offset C/L/W` (singular)
`<ANYONE> in your staggered C/L/W/B` (singular)
`<ANYONE> in your Z` (singular)
: These are done from a variety of setups, in which the designated people form a recognizable distorted or offset setup. Reasonable shape-changers are allowed.

`parallelogram`
: This includes 100% offset parallelograms in 2x8 matrices.

`parallelogram diamonds`
: This is used for any facing directions.

`phantom big block L/W`
`phantom stagger columns`
`phantom stairstep C/L/W`
`phantom ladder C/L/W`
`phantom offset C/L/W`
`phantom butterfly or O`
: These are the phantom versions of the indicated concepts, in which each dancer works in the spots of indicated setup, with real people or phantoms as required.

15.4 4-person distorted concepts

`split`
: This divides a 2x4 into two 2x2's. It also has all the other meanings of the word "split." The distinction between this as a concept and as a word appearing in a call name is extremely hazy.

`once removed`
`twice removed`
`thrice removed`
`once removed diamonds`
: The `once removed` concept encompasses diamonds. so you can say either `once removed` or `once removed diamonds`.

`magic`

`diagonal box`
`trapezoid`
: These are straightforward.

`overlapped diamonds`
: This is done from a 1x4.

`overlapped lines`
`overlapped waves`
`overlapped columns`
: These are done from a diamond.

`interlocked parallelogram`
: This is done from either a 2x4 or a 3x4. In the latter case, the spots occupied by real people are used to form two interlocked parallelograms sharing the center line or column.

`interlocked boxes`
: This is done from 3x4 matrix "Z" lines or columns. The spots occupied by real people are used to form two elongated boxes sharing the center line or column. It is equivalent to `Z C/L/W once removed`.

`twin parallelograms`
: This is done from 3x4 matrix offset lines or columns. The people that would form each distorted line or column instead form a distorted box.

`Z`
: This can be done from a single "Z", or from two Z's. A 4-person call must be used.

`each Z`

`interlocked Z's`
: These two are done from a setup in which there are two identifiable Z's. A 4-person call must be used.

`triple Z's`
: This requires a setup with three Z's.

`Z diamond`

`Z diamonds`
: In these concepts, Z's are treated as distorted diamonds. The dancers do a diamond call and then go back to footprints.

`jay`

`back-to-front jay`
`back-to-back jay`
`front jay`

`back jay`

`clockwise jay`
`counterclockwise jay`
: These include the cases of a 2-faced line in the center. In that case the shapes of the resultant 4-person boxes are different, and the concepts are equivalent to the "parallelogram" concepts below.

`left jay`

`right jay`
: These are similar to front jay and back jay.

`facing parallelogram`
`back-to-front parallelogram`
`back-to-back parallelogram`
: These are other names for the `jay` concepts, except that the center line is required to be a 2-faced line.

`blocks`

`in your blocks`
: These are two names for the same thing.

`4 phantom interlocked blocks`
: This concept is straightforward.

`triangular boxes`
: This may only be done from a "blocks" setup.

`4 phantom triangular boxes`
: This may be done from any 4x4 matrix.

`distorted blocks`
: This may be done from a 4x4 matrix that comprises blocks with some people out of their usual place in an unambiguous way. People work to the same spots.

15.5 Miscellaneous Concepts

`left`

`reverse`

`cross`

`single`

`grand`

`mirror`

`interlocked`
: These are straightforward.

`single file`
: You must use this concept, instead of the words "on a double track" or whatever, when doing calls like *dixie style* from a single file starting double pass thru.

`triangle` This turns a triangle into a box by having the apex step backward. It is intended for things like `triangle peel and trail`.

`leading triangle`
: This turns a triangle into a box by having the apex step forward. It is intended for things like `leading triangle reach out`.

`diamond` This is for things like `diamond quarter thru` and `diamond single wheel`. You should not need to select `diamond` except when you actually want this concept. There are cases in which it may seem that the word *diamond* ought to be added after selecting `interlocked` and/or `magic`. The program will insert the extra word *diamond* for you. So, for example, selecting `magic` and then `alter the diamond` will produce the output '`magic diamond, alter the diamond`', and selecting `magic`, `interlocked`, `diamond`, `as couples`, and `quarter right` will get '`magic interlocked diamond, diamond as couples quarter right`'.

`12 matrix`

`16 matrix` See Section 15.12 [12 Matrix and 16 Matrix], page 100.

`phantom` This is both the C1 concept, and the concept required in front of a tandem or as-couples concept to turn it into '`phantom tandem`' or '`phantom as couples`'.

`funny` This can handle '`circulate`'-like calls, and things like '`square thru`', '`right on`', '`slide thru`', '`star thru`', '`partner tag`', and some types of '`grand chain 8`'. However, it is not as creative as humans are in dealing with a variety of other applications.

`matrix` This is used in front of concepts like '`split phantom waves`' to indicate that the normal "gluing" rule is replaced by a rule that makes each setup stay centered on its original center, possibly resulting in overlap among the setups.

`assume <some setup>`
`assume normal casts`
`with active phantoms`
See Section 15.7 [Assume Waves], page 86 and Section 15.8 [Assuming a Quarter-Tag], page 88.

`invert` This inverts the centers' and ends' parts of the call.

`fan`

`yoyo`

`fractal`

`generous` This increases by one the amount of the first arm-turn in the call.

`stingy` This decreases by one the amount of the first arm-turn in the call.

`straight`

`twisted`

`central`

`invert central`
`snag`

`invert snag`
`snag the <ANYONE>`
`mystic`

`invert mystic`
These are straightforward.

`<ANYONE> are standard in`
This is used in front of concepts like '`split phantom lines`'.

`stable`

`<ANYONE> are stable`
`<N/4> stable`
`<ANYONE> are <N/4> stable`
`emulate`

`nose`

`rectify`

`drag the <ANYONE>`
`trace` `[must be entered by itself]`
`outeracting`
`ferris`

`release`

`stretch`

`stretched setup`
`stretched C/L/W/B/D/1T/3T`
`stretched <any triangle concept>`
These are (reasonably) straightforward.

`overlapped siamese`
This is done from a 2x4, usually one in which people are T-boned. Each person works either as couples or in tandem, as if with the other person in his quadrant, regradless of that other person's facing direction. Each person's resulting setup is a virtual 2x2 box, and the person does the call in that box.

`centers`

`ends`

`centers and ends` `[must be entered by itself]`
`center 6/outer 2` `[must be entered by itself]`
`center 2/outer 6` `[must be entered by itself]`
`<ANYONE>`

`<ANYONE> (while the others)` `[must be entered by itself]`
`<ANYONE> disconnected`
`<ANYONE> disconnected (while the others)` `[must be entered by itself]`
`same sex disconnected`
`<ANYONE> do your part`
`<ANYONE> do your part (while the others)` `[must be entered by itself]`
`on your own` `[must be entered by itself]`
`own the <ANYONE>` `[must be entered by itself]`
`ignore the <ANYONE>`
These are straightforward. See Section 15.10 [Designating Certain People], page 93, for a detailed discussion of these concepts.

`<ANYONE> lead for a`
This is used from a "promenade, do not stop . . ." situation. The call will be done as if in 2-faced lines, with the designated people in the lead. The final promenade distance will be computed as though you had allowed the dancers to promenade about halfway around the set before they did the call. This concept is somewhat cantankerous. It puts the dancers into 2-faced lines and then has everyone do whatever you say next.

NOTE: If you enter `heads lead for a wheel around`, it will have *everyone* wheel around, resulting in 2-faced lines. To get the effect that is usually indicated by the words "promenade, do not stop . . ., heads wheel around", you must enter enter `heads lead for a heads wheel around`.

`<ANYONE> move in and`
This is used when the dancers are on squared-set spots, as would be the case after things like '`all 4 ladies chain`' or '`all 4 couples star thru`'. The designated people move in to the center and do (or start) the next call as though they are starting a sequence.

`ends concentric`
`outer 2 concentric`
`outer 6 concentric`
`centers and ends concentric` `[must be entered by itself]`
: These are like '`ends`' and '`centers and ends`', except that the concentric rules (e.g., lines-to-lines and columns-to-columns) are used to dictate where the ends should finish.

`checkpoint` `[must be entered by itself]`
`reverse checkpoint` `[must be entered by itself]`
`checkerboard`
`checkerbox`
`checkerdiamond`
`orbitboard`
`orbitbox`

`orbitdiamond`
`twin orbitboard`
`twin orbitbox`
`twin orbitdiamond`
`<ANYONE> preferred for trade, checkerboard`
`<ANYONE> preferred for trade, checkerbox`
`<ANYONE> preferred for trade, checkerdiamond`
`shadow line`
`shadow box`
`shadow diamond`
`paranoid`

`<ANYONE> are paranoid`
: The designated people (or everyone) turns back at the conclusion of the call. If a designator such as `centers` is used, it is evaluated at the start of the call.

`anchor the <ANYONE>`
: These are straightforward.

`<ANYONE> work`
: This seemingly nonsensical phrase is intended to be followed by another concept, as in '`centers work tandem acey deucey`'. It causes the indicated people to use the following concept while the others do not. The program executes it effectively as an '`own the <ANYONE> for a <concept> <call> by <call>`'. That is, `ends work tandem, swing thru` is performed by having the ends do their part of a '`tandem swing thru`' while the centers do a '`swing thru`'.

`two calls in succession` `[must be entered by itself]`
: This concept allows a pair of calls to be executed atomically under a concept. This makes it possible to do a '`concentric (couple up ; touch 1/4)`' and have the lines-to-lines rule embrace the whole thing. It also makes it possible to place two calls on the same line that you believe will need to be spoken together for clarity, so that you won't accidentally pause when reading the card. The finished transcript will show the two calls within parentheses and separated by

a semicolon. The proper way to say the concept when calling can, in extreme cases, be problematical. Something like "consider the following two calls to be one unit, and do a stable swing thru and turn thru" might be appropriate.

`precede it by` `[must be entered by itself]`
`follow it by` `[must be entered by itself]`
`add` `[must be entered by itself]`

The concepts `precede it by` and `follow it by` take two subject calls. They perform the first call (first in textual order, that is) *before* or *after*, respectively, the second. The concept `add` is the same as `follow it by`. For example, `follow it by roll, swing thru` does a swing thru and roll. There is only one context in which these concepts are sensible—they can be the subject of the `piecewise`, `random`, or `reverse random` meta-concepts. For example, `piecewise add criss cross the deucey, turn the key` will do a '`criss cross the deucey`' after each of the three parts of '`turn the key`'.

These concepts are among the situations in which the program does not attempt to print out the exact words that are appropriate to use when calling, but instead prints out an unambiguous description of what is supposed to happen. The exact words you should use depend on the context and on your judgement. You might say "piecewise add a criss cross the deucey, and turn the key," or "do a criss cross the deucey after each part, and turn the key," or something similar.

`crazy`

`reverse crazy`
`<N/4> crazy`
`<N/4> reverse crazy`

These are straightforward.

`<N>/<N>`

`1-<N>/<N>`
`twice`

`<N> times`

`do the last <N>/<N>`

The `N/N` concept, where `N` and `N` are numbers, causes that fraction of the subject call to be executed, as in '`3/5 swing the fractions`'. In `Sdtty`, just type in the numbers directly, as in `3/5`. In `Sd`, the concept is listed as `<N>/<N>`. A popup will appear asking for the numbers. Enter two of them.

The `1-<N>/<N>` concept causes the subject call to be executed once and the indicated fraction of a second time, as in '`1-1/2 split circulate`'.

The exact way you should say the fraction might vary according to the call or your individual taste. Proper fractions always appear before the call in the transcript, but you might want to say the fraction after the call, as in '`swing the fractions four fifths`'.

The program will place improper fractions after the call, as in '`split circulate 1-1/2`'. You still specify the concept first when entering it into the program.

That is, you type `1-1/2 split circulate`. In any case, you must use your judgement when choosing what to say when calling.

In extremely tricky cases, such as '`1/2, mix`', '`3/4, mix`', '`3/4 mix`', and '`2/3, 3/4 mix`', these concepts can lead to ambiguity. It may be necessary to type the concept by itself, that is, to press ⟨ENTER⟩ at the appropriate moment, to get what you want. (If using the menu in `Sd`, select just the concept, and then select the call.) For example, if you really want 3/4 of the call '`mix`', type

3/4 ⟨ENTER⟩
mix ⟨ENTER⟩

If using the menu in `Sd`, select `<N>/<N>`, and then select `mix`.

In the finished transcript, a comma will appear after fractionalizing concepts, but not in the call `3/4 mix`. You of course never type in any of the commas that serve to separate concepts.

The `twice` concept simply causes the subject call to be executed twice. This is a notion familiar at all levels from Mainstream ('`spin chain thru, the girls double circulate`') to C4 ('`piecewise twice, recoil`').

The `<N> times` concept causes the subject call to be executed that number of times. In `Sdtty`, just type in the number directly, as in `3 times`. In `Sd`, the concept is listed as `<N> times`. A popup will appear asking for the number.

No universally recognized words for these exist, so the program uses the generic phrases noted above. When calling, use whatever words you think are appropriate, such as "twice," "circulate two positions," or whatever.

`initially`
`secondly`

`thirdly`

`fourthly`

`finally`

`initially and finally`
`piecewise`
`random`

`reverse random`
`evenly`

`oddly` These are followed by the concept to which they apply. See Section 15.11 [Anyone Start], page 99, for more on the `initially` concept.

`shifty`

`shift <N>`

`shift 1/2`

`shift <N>-1/2`
`interlace` `[must be entered by itself]`
`reverse order`
`echo`

`reverse echo`
: These are straightforward.

`double/triple/quadruple echo`
: The `double echo` meta-concept is followed by two concepts. It applies both concepts first, then skips the first concept and applies the second only, and finally skips both concepts and just does the call. `triple echo` and `quadruple echo` are similar.

`randomize between`
: This is followed by two concepts, and then a call. The parts of the call are performed alternately with the first concept and the second concept.

`triple randomize among`
: As above but three concepts are given. The parts of the call are performed with the first, second, and third concept, in repeating sequence.

`roundtrip`
: The `roundtrip` concept is most simply explained by example: `swing the fractions` is the same as `roundtrip remake`. The `roundtrip` of a call is the call followed by the reverse order of the call without its last part. The `roundtrip` of an N-part call has 2N-1 parts.

`finish`

`like a`

`like an`
: See Section 15.9 [Interruptions and Replacements], page 89, for information about some tricky aspects of the concepts that pull calls into their constituent parts.

  To Sd, `finish` means "skip the first part" and is only legal for calls for which the first part is recognizable. Sd also recognizes the concept `like a`, meaning "do the last part." You need to be aware, however, that there are calls for which, while it is technically legal to say such things, they are not accepted as common usage. For example, `finish shazam` as a way of saying *U-turn back*, or `finish mix` or `like a mix` as ways of saying *centers trade*, would have to be considered rather peculiar. You must exercise your judgement when using these concepts.

  One occasionally hears things like *cross chain reaction, centers finish like a wheel the ocean*, in which the final cast off 3/4 of the chain reaction was "pushy." While the plain English language meaning of that is clear, the way you must enter that to Sd is

  ```
  CROSS chain reaction
  CENTERS LIKE A wheel the ocean
  ```

  You must, of course, verify yourself that the casting direction is correct for smooth dancing of this figure.

`<ANYONE> start`
`skip the <Nth> part`
`sandwich` `[must be entered by itself]`
: These concepts cause the call to be interrupted or replaced as indicated.

The *<ANYONE> start* concept has the indicated people do their own part of the first part of the call, after which the others join them. You should type this in as, for example

boys start quarter thru ⟨ENTER⟩

See Section 15.11 [Anyone Start], page 99, for more on the *<ANYONE> start* concept.

`do the <Nth> part`
`do the last part`

The *do the <Nth> part* and *do the last part* concepts are intended to be used with a following concept, as in '`do the 2nd part tandem remake`'. It applies the following concept to the indicated part of the call, but not to the rest.

`first <M>/<N>`
`middle <M>/<N>`
`last <M>/<N>`

The *first <M>/<N>*, *middle <M>/<N>*, and *last <M>/<N>* are similar, but apply the concept to the indicated first, middle, or last fraction of the call, independently of how the call divides into parts. For example, '`first 1/4 tandem spin the top`', '`middle 1/3 tandem remake`', or '`last 1/4 stable swing thru`'. Whether these are precisely the words you should use is up to you.

`half and half` `[must be entered by itself]`

This does the first half of the first call, followed by the second half of the second call.

`<N>/<N> and <N>/<N>` `[must be entered by itself]`

This does the first <first fraction> of the first call, followed by the last <second fraction> of the second call. It is a generalization of half and half.

`replace the <Nth> part` `[must be entered by itself]`
`replace the last part` `[must be entered by itself]`
`interrupt after the <Nth> part` `[must be entered by itself]`
`interrupt after <M>/<N>` `[must be entered by itself]`
`interrupt before the last part` `[must be entered by itself]`

The *replace* and *interrupt* concepts require two calls. After typing the concept, type the call that is to have a part interrupted or replaced. The program will then ask you for the call that comprises the interruption or replacement. For example, you might type

replace the 3rd part ⟨ENTER⟩
swing the fractions ⟨ENTER⟩
2/3 recycle ⟨ENTER⟩

`use for the <Nth> part` `[must be entered by itself]`

The *use for the <Nth> part* concept is the same as *replace the <Nth> part*, except that the replacement call is entered first. It is entered thusly:

use for the 3rd part ⟨ENTER⟩
ah so ⟨ENTER⟩
swing the fractions ⟨ENTER⟩

It will apear in the transcript as "use ah so for the 3rd part: swing the fractions".

`use` `[must be entered by itself]`

The *use* concept simply replaces the second call with the first. By itself, it is rather nonsensical. It is intended to be used with meta concepts. It is entered thusly:

reverse random use ⟨ENTER⟩
acey deucey ⟨ENTER⟩
swing the fractions ⟨ENTER⟩

`start with` `[must be entered by itself]`

The *start with* concept is equivalent to replacing the first part. However, you must enter the replacement call first, and the program will then prompt you for the call that is to have its first part replaced. For example, you might type

start with ⟨ENTER⟩
fan the top ⟨ENTER⟩
the difference ⟨ENTER⟩

Because the *replace*, *interrupt*, *start with*, and *use* concepts require two calls, you must press ⟨ENTER⟩ after typing them, and then type the two calls with ⟨ENTER⟩ after each one.

See Section 15.9 [Interruptions and Replacements], page 89, for information about some tricky aspects of these concepts.

`inside triangles`
`outside triangles`
`in point triangles`
`out point triangles`
`tall 6`

`short 6`

`wave-based triangles`
`tandem-based triangles`
`<ANYONE>-based triangles`

These are straightforward. Some them may be preceded by modifiers such as '`interlocked`' and/or '`magic`'.

`concentric`
`cross concentric`
`single concentric`
`single cross concentric`
`grand single concentric`
`grand single cross concentric`
`concentric diamonds`
`cross concentric diamonds`
`concentric Z's`
`cross concentric Z's`
`3x3 concentric`
`3x3 cross concentric`
`4x4 concentric`
`4x4 cross concentric`

See Section 15.6 [Concentric], page 85.

`grand working forward/backward`
`grand working right/left`
`grand working as centers`
`grand working as ends`
`grand working clockwise/counterclockwise`
`<ANYONE> are centers of a double-offset 1T/3T/1L/3L/GT/D/DS`
`inrigger`

`outrigger`
`leftrigger`
`rightrigger`
`backrigger`
`frontrigger`
`right wing`
`left wing`

`mystic wing`
`invert mystic wing`
`other wing`

These are straightforward. "Mystic wing" is right wing for the outer 4 and left wing for the center 4. "Other wing" is right wing for the belles and left wing for the beaus.

`mimic` This is experimental.

`common point galaxy`
`common spot/point diamonds`
`common spot/point hourglass`
`common spot point-to-point diamonds`
`common spot 1/4 tag`
`common spot 1/4 line`
`common center/end L/W`
`common spot C/L/W`
`common spot two-faced lines`

The "common spot" concept directs people to do their part of the call as though the presumed collision from the previous call had not forced them to take right hands, and they are simply occupying the same spot. (This is not to say that a collision from a previous call is the only way that such a setup can be created—the dancers are simply directed to act as though that is how they got there. Some people consider the use of collisions to set up "common spot" calls to be among the less creative ways to get into these formations.)

The program therefore expects the "common spot" people to have *right* hands. If you want to use these concepts with people in left-hand miniwaves, we recommend that you apply the '`mirror`' concept to the entire operation.

Common spot point-to-point diamonds can arise from certain fractional exchanges of point-to-point diamonds. Note that `switch to a diamond` and `diamond circulate` never produce this setup – the Callerlab C1 definitions specifically state that the colliding people center themselves.

The `common point galaxy` concept is used from a "rigger" or "bat" setup in which the "wings" are collided points.

The `common spot diamonds` or `common point diamonds` concept is used from the kind of setup one obtains after a '`6x2 acey deucey`' from facing diamonds. The `common spot hourglass` or `common point hourglass` concept is used from the kind of setup one obtains after a '`6x2 acey deucey`' from a facing hourglass.

The `common spot 1/4 tag` and `common spot 1/4 line` concepts are used from a tidal wave. Everyone acts as though they are in the center wave or two-faced line.

The `common spot columns` concept is used from "clumps", "stairsteps", Z columns, or waves. From clumps or stairsteps everyone is presumed to have collided. From Z columns some people have collided and some (those that are centered) have not. From waves, everyone has collided and is in the center of the imagined columns.

The `common end lines/waves` concept is used from a parallelogram in which the "wings" are collided ends of the lines. It can also be used from outer triple boxes, in which everyone has collided at the ends of lines. People in the center triple box, if any, are natural.

The `common center lines/waves` concepts are used from waves in which everyone is presumed to have collided in the center. The ends of the imagined lines or waves are all phantoms.

The `common spot lines/waves` concepts may be used from waves, having the same meaning as `common center lines/waves`.

The `common spot waves/two-faced lines` concepts may also be used from a 2x8 that is either a fully offset parallelogram or is a "miniwave zipper". That is, everyone is in a miniwave with someone, and they collectively occupy a 2x4 matrix of live minwaves or phantom miniwaves. The concept name tells how the people should be grouped together, in case that matters.

`1x2/2x1/2x2/1x3/3x1/3x3/4x4/6x6/8x8`
: These are straightforward.

`all 4 couples`
`all 8`

`all 8 (diamonds)`
: The `all 4 couples` and `all 8` concepts currently appear to be undergoing some re-examination. The program's behavior on these might not be satisfactory at the present time. The `all 8 (diamonds)` concept is used from a thar, in which the call is intended to be performed in each of two interleaved diamonds. In real life, one can generally call this as just `all 8`, and the dancers will know what is required. The program is not that smart, and requires the "(diamonds)" hint.

`each C/L/W/B/D`
`each 1x4`
: These can be used in troublesome situations to force a setup to be split in the indicated way. For example, from a tidal wave, the `each wave` concept can be used to cause a '`counter rotate`' to be performed as a '`lockit`'. The `each 1x4` concept splits the setup into 1x4 setups without regard for individual dancers' facing direction. From columns, `each 1x4 sidetrack` can cause the '`sidetrack`' to be done in individual columns (though `single sidetrack` will do the same thing). These concepts can also make your intentions clear when, for example, you give a box call when the setup is a 4x4 matrix with boxes occupied in each corner.

  These concepts may also be used to "justify" using 4-person calls in large phantom setups. They will suppress the warning message that the program would otherwise give when such calls are used without an appropriate concept. For example, from a parallelogram, the `parallelogram` concept must be used when giving 8-person calls such as `acey deucey`, but it is possible to give 4-person

calls such as `mix` without any concept. However, the program will issue a warning. Using the `each wave` concept will prevent the warning. In either case, the exact words you would use whan calling the card are up to you.

`1x6 matrix`
`1x8 matrix`
`1x10 matrix`
`1x12 matrix`
`1x16 matrix`
`2x5 matrix`
`2x6 matrix`
`2x8 matrix`
`2x10 matrix`
`2x12 matrix`
`3x4 matrix`
`4x4 matrix`
`4x5 matrix`
`3x6 matrix`
`3x8 matrix`
`4x6 matrix`
`16 matrix of parallel diamonds`

These force the setup to be expanded as shown, to cause various other concepts to be interpreted in the desired way. For example, from parallel waves, the concept `phantom tandem` would expand the setup to a 4x4 matrix, making each person in tandem with a phantom, and creating a virtual 2x4. `2x8 matrix phantom tandem` would leave live people tandem with live people, create phantoms in tandem with other phantoms, and create a virtual tidal line. The `16 matrix of parallel diamonds` concept expands to quadruple diamond spots.

15.6 Concentric

The '`concentric`' and '`cross concentric`' concepts are formulated as follows: The rule tells how the people who finish on the outside should elongate their 2x2 if they finish in one. This is the only case that needs to be addressed.

1. If the people who finished on the outside in a 2x2 started in a 2x2, they use the "lines-to-lines/columns-to-columns" rule, relative to what setup *they* thought they were in.

   If the concept was *concentric*, so these people started on the outside in an elongated 2x2, they can make a decision about whether they had line or column elongation. They do not have to be in a 2x4 for this. For example, from a dogbone created by Heads touch, the sides think they are in columns. But from ordinary diamonds or an hourglass, they think they are in lines.

   If the concept was *cross concentric*, so these people started in the center in a 2x2, they must have the assistance of the original ends. The original setup must have been a 2x4, except for weird pinwheel-like setups noted below. Hence it is *illegal* to call *cross concentric square thru 2* from a rigger in which the center box is facing couples and the wings have right hands and are expected to rear back. The centers do not know

whether they were in lines or columns because the ends are not in an acceptable setup. The centers do *not* use the "checkpoint elongate perpendicular to the 1x4 rule"—that rule does not apply here. If the outsides are in a pinwheel, that is, occupying a 4x4 matrix in an irregular way, each center makes the lines vs. columns decision based on the end person in their own quadrant. That is, the center person's own facing direction and that end person's *location,* but not that end person's facing direction.

2. If the people who finished on the outside in a 2x2 started in a 1x4 or diamond, they use the *checkpoint rule*—they elongate their resulting 2x2 perpendicular to their original long axis.

   If the concept was *concentric*, so these people started on the outside in a 1x4 or diamond, they use their own long axis, which is the same as the long axis of the whole set.

   If the concept was *cross concentric*, so these people started in the center in a 1x4 or diamond, they elongate their resulting 2x2 perpendicular to *their own axis*, that is, the axis of the 1x4 or diamond in which they began. This rule applies, for example, when '`cross concentric recycle`' or '`center 2 shazam; cross concentric diamond recycle`' is called from a quarter-tag.

15.7 Assume Waves

There are a number of concepts that you can use to tell the dancers how phantoms are facing. They are

```
assume waves
assume miniwaves
assume couples
assume two-faced lines
assume one-faced lines
assume inverted lines
assume normal boxes
assume inverted boxes
assume normal columns
assume magic columns
assume eight chain
assume trade by
assume dpt
assume cdpt
assume facing lines
assume back-to-back lines
assume general diamonds
assume general 1/4 tags
assume 1/4 tags
assume right 1/4 tags
assume left 1/4 tags
assume 3/4 tags
assume right 3/4 tags
```

```
assume left 3/4 tags
assume 1/4 lines
assume right 1/4 lines
assume left 1/4 lines
assume 3/4 lines
assume right 3/4 lines
assume left 3/4 lines
assume normal diamonds
assume facing diamonds
assume normal interlocked diamonds
assume facing interlocked diamonds
assume split square thru setup
assume I setup
assume galaxy
assume hourglass
```

Additionally, there are two related concepts, `assume normal casts` and `with active phantoms`, which will be discussed below.

When these concepts are used by themselves, the information that they convey tells the dancers how to begin certain calls such as '`hinge`', '`vertical tag`', and '`in roll circulate`'. The information is thrown away after the call begins. That is, the program does not hold the dancers responsible for tracking phantoms through a call. It only tells them how to begin.

Use these concepts with care. Many dancers don't like to be held responsible for the facing direction of phantoms.

These concepts will be implicitly applied whenever something like `split phantom waves` or `triple diamonds` is used, except for 1/4 tags and 3/4 tags. Concepts like `split phantom 1/4 tags` and `split phantom 3/4 tags` assume that the ends are facing in or out, respectively, and that the centers form any kind of line at all. The concepts `assume 1/4 tags` and `assume 3/4 tags` create the additional assumption that the centers are in a wave (of either handedness.) The concepts `split phantom 1/4 lines` and `split phantom 3/4 lines` require a 2-faced line, and so an additional `assume 1/4 lines` and `assume 3/4 lines` concept would have no effect.

It is of course illegal to use one of these concepts, either explicitly with `assume` or implicitly with something like `split phantom waves`, if the live dancers are not facing in a way that is consistent with the assumption.

15.7.1 Assume Normal Casts

The concept `assume normal casts` may be used in cases in which you do not require the dancers to track phantoms through the call, but you want to resolve any ambiguity about the '`cast off 3/4`' operations that happen at the end of such calls as '`chain reaction`' and '`motivate`'. You might read such an operation as '`12 matrix motivate, assume the final cast off is normal`'.

15.7.2 With Active Phantoms

The concept `with active phantoms`, used after one of the `assume` concepts (either implicit or explicit), effectively places invisible but intelligent dancers on the phantom spots, with facing directions deduced from the assumption. Those phantoms proceed through the entire call, interacting with each other and with the live dancers as necessary. If the call is impossible for the phantoms to do, or if illegal collisions occur between live dancers and phantoms, an error message is displayed. At the end of the call, any live dancers or phantoms that come to the same spot may take right hands with each other, if that is appropriate for the call.

Most of the types of `assume` concepts listed above are sufficiently explicit that, if even a single live dancer is present in the setup, everyone's facing direction may be inferred. This is necessary for the `with active phantoms` concept, of course. For example, a single live dancer can determine the handedness of all the phantoms when `assume two-faced lines` is given. Two of the concepts are not sufficiently explicit for this, and may not be used with `with active phantoms`. They are

```
assume general diamonds
assume general 1/4 tags
```

If you are going to use `with active phantoms`, you must give more detailed assumptions than these two, such as `assume left 1/4 lines` or `assume facing diamonds`.

Use the `with active phantoms` concept with extreme care, if at all. It is extremely dangerous in terms of what it may require the dancers to do. Many dancers find it difficult enough to do the live people's parts of calls without having to do the phantoms' parts also.

You can put `Sd` into a mode in which there is an implicit `with active phantoms` whenever any assumption is made. The `toggle active phantoms` command toggles (turns on or off) this mode. See Section 8.9 [Changing Modes], page 36. When in this mode, `Sd` will not actually insert the text `with active phantoms` after each `assume` concept.

15.8 Assuming a Quarter-Tag

Callers sometimes wish to perform a generalized quarter-tag call (e.g., '`scoot and little`') from a setup in which only the center wave contains real people, and the outsides need to be assumed. For example, from an as-couples wave, one could wish to specify that the live dancers are the as-couples center wave of a quarter-tag. The way to specify this when calling is typically something like '`as couples in a quarter-tag, scoot and little`'.

You can do this with Sd in either of two ways. You can use the concepts with names like `as couples in a 1/4 tag`. These concepts exist for '`1/4 tag`', '`1/4 line`', '`3/4 tag`', and '`3/4 line`' formations, with all the usual '`twosome`' variations. There are no `tandem` variations of this.

Alternatively, you can use the `as couples` (or whatever) concept, followed by a suitable `assume` concept, such as `assume general 1/4 tags` or `assume 1/4 tags`.

For example, from an as-couples wave, you could select:

```
as couples in a 1/4 tag scoot and little
```

or

```
as couples assume 1/4 tags scoot and little
```

Or, from parallel waves, you could select:

```
tandem assume general 1/4 tags ping-pong circulate
```

15.9 Interruptions and Replacements

There are a number of ways you can get `Sd` to interrupt calls or replace parts of calls, just as there are a number of ways to tell the dancers what you want.

By Part Number

For some calls, it is natural to indicate the interruption or replacement point by a part number. This typically occurs when the numbers are well known but the names of the parts might be ambiguous. For example, callers say '`Pass the Axle, replace the 3rd part with Split Trade Circulate`'. The third part is a type of Trade, but the fourth part is also a Trade, so it is safer to refer to it by number.

The concepts that `Sd` provides for this are:

```
replace the <Nth> part
replace the last part
interrupt after the <Nth> part
interrupt after <M>/<N>
interrupt before the last part
skip the <Nth> part
```

The first four listed take two calls—the principal call and the replacement/interruption. You must press ⟨ENTER⟩ after typing these, and then enter the calls separately.

```
replace the 3rd part ⟨ENTER⟩
pass the axle ⟨ENTER⟩
split trade circulate ⟨ENTER⟩
```

or

```
interrupt after the 3rd part ⟨ENTER⟩
contour the line ⟨ENTER⟩
fan relay the top ⟨ENTER⟩
```

By Fraction

You can also interrupt a call (though you can't replace a part) by giving the fraction of the call at which the interruption is to occur. The concept for this is `interrupt after <M>/<N>`. You can use this concept to interrupt calls that don't actually have parts, as in

```
interrupt after 1/2 ⟨ENTER⟩
scoot back ⟨ENTER⟩
cross over circulate ⟨ENTER⟩
```

or to interrupt at a point that is not a part boundary, as in

```
interrupt after 1/8 ⟨ENTER⟩
load the boat ⟨ENTER⟩
grand swing thru ⟨ENTER⟩
```

Anything Anything

For '`anything anything`' calls, such as '`trade motivate`', just type it in.

Star Turns

For calls with a star turn, you can usually modify the amount of the turn, or omit it, by typing '`, turn the star 1/4`' (or 1/2 or 3/4), or '`, don't turn the star`'. (Remember that you never need to type commas or apostrophes.)

When in doubt, type a question mark immediately after the call.

```
chain reaction, don't turn the star
scoot and plenty, turn the star 1/4
```

To replace the star turn with some other call, you need to use the "brute force" method described below.

Replacing the Cast Off 3/4

Calls that end with the centers casting off 3/4 (with the ends usually "moving up") can usually have the cast off replaced by typing `but [<some other call>]`. The replacement call is put in square brackets.

```
tally ho but [2/3 recycle]
chain reaction, turn the star 1/2 but [mix]
```

When in doubt, type a question mark immediately after the call.

"But" by Convention

Some calls by convention allow "but" modifications of certain agreed-upon parts. Just type it.

```
spin the pulley but [reach out]
line to line but [catch [grand mix] 2]
slant [swing thru] and [turn and deal]
```

When in doubt, type a question mark at the tricky part. If it shows you a choice with '`<ANYTHING>`', you can type in another call, in square brackets, in place of the '`<ANYTHING>`'. For example, if we are not sure about the replacement for '`line to line`', we could type

```
line to line?
```

and it would offer

```
line to line
line to line but <ANYTHING>
```

The second is the one we want. It tells us that the program takes '`line to line but`' followed by some call in brackets. So we type

```
line to line but [catch?
```

and it shows

```
line to line but [catch <N>]
line to line but [catch <ANYTHING> <N>]
line to line but [catch <ANYTHING> <N>, only the resulting
     centers finish]
```

The second is the one we want. So we know that we can type

```
line to line but [catch [grand mix] 2]
```

Similarly, if we had typed

```
slant?
```

the program would offer

```
slant <ANYTHING> and <ANYTHING>
slant <ANYTHING> and wheel
slant touch and <ANYTHING>
slant touch and wheel
```

The first of those indicates that we can type

```
slant [swing thru] and [turn and deal]
```

Initial Replacements

Some calls can take a subcall in front, typically meaning to replace the first part. The replacement call is placed in brackets. All of the '`anything anything`' calls are of this type, when the replacement call isn't a circulate replacement. (If it is a circulate replacement, just type it with no brackets.)

```
split counter percolate
[2/3 recycle] percolate
[reverse the top] an anchor
[reverse the top] an anchor but [ah so]
[bingo] cover up but [step and fold]
```

Brute Force

There are many other "brute force" ways to modify calls, typically by saying something like '`but replace the diamond circulate with 6x2 acey deucey`'. This is done in `Sd` by giving the command `allow modifications` before the call. `Sd` will ask about the various subcalls to be replaced. Answer `y` or `n` to the questions, and enter the selected subcalls.

```
--> allow modifications
--> alter the wave
The "swing" can be replaced.
Do you want to replace it? n
The "turn the star 1/2" can be replaced.
Do you want to replace it? n
The "flip the diamond" can be replaced.
Do you want to replace it? y
REPLACEMENT FOR THE flip the diamond
--> relocate the diamonds
```

or

```
--> allow modifications
--> scoot and plenty
The "right scoot back" can be replaced.
Do you want to replace it? n
The "turn the star 1/2" can be replaced.
Do you want to replace it? y
REPLACEMENT FOR THE turn the star 1/2
--> turn the star 3/4, interrupt after 1/2 with [2/3 recycle]
```

(Yes, that last line is a real call that you can type.)

Tricky Aspects of these Concepts

Some of the replacement/interruption meta-concepts push the semantics of the language to the limit. Whenever a call undergoes an interruption or enters or leaves a part with an additional concept on it, there is an implicit *piecewise* at that instant. That is, concepts and setups are re-evaluated. This behavior is not always obvious to the dancers. (However, the dancers' positions within the setup will not be re-evaluated if the call depends on carrying this information from one part to another, as in '`patch the <anyone>`' or '`rims trade back`'.)

Furthermore, some cases of replacements or interruptions may lead to situations that are likely to be perceived as wrong. For example, an implied *piecewise* on a concept such as *cross concentric* or *single cross concentric* is very likely to be considered wrong, because the centers and ends will switch with each other multiple times. If you have any doubt about whether some instance of concept stacking is correct and will be understood by the dancers, do not use it.

Furthermore, replacements and interruptions are *normal*, that is, they do not carry any concepts that were on the call being replaced—only those on the entire operation.

Example, from a starting DPT setup:

```
DELAY: TANDEM TWOSOME clean sweep 1/4 BUT REPLACE THE
3rd PART WITH A [CHECKPOINT crossfire BY crossfire]
```

Normally, the tandem twosome behavior is not re-evaluated after each part of the clean sweep. But, since the third part was replaced with something else to which that concept did not apply, the setup is re-evaluated before doing the final part. Note in particular that the replacement was *normal*—the call having a part replaced was a *tandem twosome clean sweep 1/4*. If we wanted the entire operation, including the replacement, to be tandem twosome (with no re-evaluation, of course) we might call:

```
TANDEM TWOSOME DELAY: clean sweep 1/4 BUT REPLACE THE
3rd PART WITH A [SINGLE CROSS CONCENTRIC turn thru]
```

There are some calls that behave as though they had "state" information that one must remember as the call progresses. Take the case of '`exchange the diamonds`'. When two people meet at a diamond point during this call, the person who has "already exchanged" takes the outside track, and the person who hasn't takes the inside track. If they have to stop, they take right or left hands depending on who is on which track. (If they have both exchanged, or neither has exchanged, they are on the same track and therefore take right hands.)

It follows from this that people need to know how far they have progressed through the call. If a call is interrupted, interlaced, fractionalized, or otherwise manipulated, this can become tricky. The dancers may need to know how to complete a call that they didn't begin, or that they haven't been doing a consistent part of. This can become very unnatural relative to the parts of the call that are being performed. Some theories have been promulgated that involve people remembering individually what parts of the call they themselves did (e.g. whether they had passed the exchange point), or "leaving notes on the floor" when an interruption occurs, intending to tell the dancer who will resume from that point what they should pretend they had done. We believe that these methods are of marginal utility even in simple cases, and unworkable in complex cases. `Sd` takes the position that the only state information is what part of the call one is doing now. If a collision occurs after parts 1 or 2 of an '`exchange the diamonds`', the person who is an out-point has exchanged and the other has not. If a collision occurs after parts 3 or 4, both have exchanged, and hence they take right hands. (You can work this out with checkers.) `Sd` therefore considers parts 1 and 2 to be different from parts 3 and 4. No matter what interruptions, interlaces, or other concepts were in use, the dancers only need to know which of the 4 parts of the '`exchange the diamond`' they are doing at each instant. They infer whether they have passed the exchange point from that information alone.

If you think that some instance of a replacement or interruption will cause confusion on the dance floor because of these sorts of issues, you should not use it.

15.10 Designating Certain People

There are six related concepts here. You must have a clear understanding of how the program treats them in order to use them effectively. Compared to the natural and fluent way most dancers and callers use these ideas, the program may seem idiosyncratic and ignorant in its handling of them. See Chapter 12 [Linguistic Idiosyncrasies], page 52.

The six concepts are

```
<ANYONE>
<ANYONE> disconnected
<ANYONE> in your distorted setup
<ANYONE> do your part
ignore the <ANYONE>
own the <ANYONE>
```

In general, be aware that the choice of words and punctuation that the program uses for these concepts is determined by the need to avoid ambiguity, and may not be the words that you should use when calling.

Also, the program is very fussy about using the designators correctly. A large number of designators are provided to help you with this. Some are quite obvious, and some are intended for dealing with difficult situations.

Some of these designators will not be understood by dancers below high challenge levels, but the program will allow them at any time. You will need to use your judgement in calling. For example, you may need to use `sideliners` to get some particularly tricky effect. Below C4, you would need to read the card as something like '`those facing the side walls`'.

The available designators are:

`heads / sides`
`boys / girls`
`centers / ends`
`center 2 / center 6`
`very centers`
(`center 2` and `very centers` are the same.)

`outer 2 / outer 6`
`very ends` (`outer 2` and `very ends` are the same.)

`leads / trailers`
`beaus / belles`
`head corners / side corners`
`head boys / head girls / side boys / side girls`
`lead ends / lead centers / trailing ends / trailing centers`
`headliners / sideliners`
`first 1, 2, or 3 / last 1, 2, or 3`
These are used only in normal columns. They designate the first/last 1, 2, or 3 people, counting from the front. Example: `first 3 are tandems of 3, peel and trail`. These may not be the exact words you should use when calling.

`leftmost 1, 2, or 3 / rightmost 1, 2, or 3`
These are used only in lines facing in or out. Example: `leftmost 3 here comes the judge`. I am not aware that callers actually use these designators.

`those facing`
`everyone`

`all` `everyone` and `all` are the same. In practice you should hardly ever need to use them to the program, though in some situations you may need to use them when calling.

`no one`

`center 4` This is intended for use in a parallelogram or offset lines or columns. It designates the center triple box, or center triple line/column, respectively. Just saying `centers` will not be effective here.

`outer pairs`
This designates the people that are not in the `center 4`. In a parallelogram, it designates the pairs of people in the outer triple boxes (the "wings"). In offset lines or columns, it designates the the pairs of people in the outer triple lines or columns.

`center diamond`
This is intended for use in a setup with a wave between, and perpendicular to, two miniwaves. (This is the setup that results from a `1/2 circulate` from waves.) In this setup, `centers` means the center *wave*. If you want the center diamond instead, you must say so.

`center box`

This is intended for use in a parallelogram or other setup in which there is a recognizeable fully occupied 2x2 box in the center.

`center 1x4`
`center column`
`center line`
`center wave`

These are intended for use in things like 3x1 diamonds. (This is the setup that results from a `1/2 acey deucey` from waves.) In this setup, `centers` means the center *diamond*. If you want the center wave instead, you must say `center 1x4`. These can also be used in a tidal setup to disambiguate the "who are the centers?" issue. The program uses the designator `center 1x4` as a catch-all term. In most cases, you can use one of the other designators to be more specific. The program often checks that the more specific designator is being used properly, but it doesn't always check. Use them with care.

`center 1x6`
`center column of 6`
`center line of 6`
`center wave of 6`

These are similar, but refer to a line or column of 6. It is intended for use in things like 3x1 diamonds, when you want the center wave of 6 to do, for example, a `grand swing thru`. Another situation in which this is necessary arises after a `sets in motion but hold the column`. If you get a column of 6 and you want them to do something, you must identify them as the `center column of 6`. The designation `center 6` is not correct for this—it would include the person not in the column, and exclude the ends of the column.

`outer 1x3s`

This may be used to designate the outer lines or columns of 3 in such setups as an 'H' or a diamond between two couples ('`heads circle the tag to a diamond`'.) You will often need to use the '`disconnected`' concept with this.

The following ones are unsymmetrical:

`near line / far line`
`near column / far column`
`near box / far box`
`near 2 / far 2`
`near 4 / far 4`
`near 6 / far 6`
`nearest person / farthest person`
`the diamond`

Use this in an unsymmetrical setup when there is a diamond other than in the center.

```
those facing the caller
those facing away from the caller
those facing the caller's left
those facing the caller's right
#1 boy / #2 boy / #3 boy / #4 boy
#1 girl / #2 girl / #3 girl / #4 girl
#1 couple / #2 couple / #3 couple / #4 couple
couples 1 and 2 / couples 2 and 3 / couples 3 and 4 / couples 1 and 4
```

15.10.1 ANYONE

The concepts are

```
<ANYONE> <call>
```

or

```
<ANYONE> <call> while the others <other call>
```

When used with two calls, this must be entered by itself, like this:

boys (while the others) ⟨ENTER⟩
hinge ⟨ENTER⟩
switch to a diamond ⟨ENTER⟩

This will come out as:

```
boys hinge while the girls switch to a diamond
```

If the designated people are centers or ends, the program will do the call(s) according to its best judgement of what the calls mean. Otherwise, it will find the maximal connected undistorted subsets, and do the call in those subsets. For example, from boy-boy-girl-girl waves, the boys are in two miniwaves that have nothing to do with each other. The boys can do 2-person calls, such as trade or hinge, in those setups. If the others are told to do a second call, they do it in their connected undistorted subsets. For example, we could say `boys, hinge while the others shazam`. The results will be reassembled as a 2x4, a C1-phantom setup, stars, or something similar.

From boy-girl-boy-girl waves, the boys are in 4 1-person connected undistorted setups. Using this concept, the boys can quarter right, but they can't trade or hinge.

Now it is commonly accepted practice to say '`boys trade`' when the boys are looking out, and have them effectively trade the wave. This is because the call '`trade`' is a special case, not because calls in general can be done from that setup. To see this, consider the call `roll away`. It, and two-person calls in general, are *not* legal unless the people are adjacent. The special property of the call `trade`, that it can be done by designated people who are not adjacent, is simply an idiosyncrasy that everyone knows. The Sd program recognizes this by having a call in its database `<anyone> trade` as well as the call `trade`. Use that call, not the `<anyone>` concept, to get the appropriate people to trade down the line when in waves.

If you really want non-adjacent people to `roll away`, use the `disconnected` concept.

Incidentally, this is why concepts are capitalized. Otherwise, the line `boys trade` appearing on a printed sequence would be ambiguous. The Sd program is designed to make it possible to determine unambiguously, by looking at the printout, how an action arose.

15.10.2 ANYONE Disconnected

The concepts are

```
<ANYONE> disconnected <call>
```

or

```
<ANYONE> disconnected <call> while the others <other call>
```

This is like `<ANYONE>`, but finds the maximal undistorted subsets, whether they are connected or not. This is a recognized C2 concept. It is generally intended to be used from grand (1x8) setups, in which the designated people leave the room and compress themselves into a 1x4 setup. After doing the call they come back and place themselves in the same four spaces that they had vacated. Shape-changing calls are permitted. In this case, whichever set of people occupied the centermost spots prior to the call will occupy the center of the result, in accordance with accepted usage for this concept.

When used with two calls, this concept and its designator must be entered by itself, like this:

boys disconnected (while the others) ⟨ENTER⟩
swing thru ⟨ENTER⟩
switch to a diamond ⟨ENTER⟩

This will come out as:

```
boys disconnected swing thru while the girls switch to a diamond
```

15.10.3 ANYONE in your Distorted Setup

The concepts are

```
<ANYONE> in your distorted line <call>
```

or

```
<ANYONE> in your distorted wave <call>
```

or

```
<ANYONE> in your distorted column <call>
```

or

```
<ANYONE> in your distorted diamond <call>
```

or

```
<ANYONE> in your distorted box <call>
```

The designated people are identified in a geometrically distorted setup. If the result of the call has a different shape and orientation from the beginning setup, those dancers may nevertheless be able to go back to the same collective spots, or may be able to maintain the same general location in the total formation.

Be aware that the program requires you to use the most specific concept in each case. The `distorted` concept requires that the setup be geometrically distorted in shape, not just disconnected. Do not say `distorted` when `disconnected` will do, and do not say `disconnected` when just telling the people to do the call will do.

This concept has been set at C2. You can use it at lower levels if you issue the `toggle concept levels` command.

15.10.4 ANYONE Do Your Part

The concepts are

```
<ANYONE> do your part <call>
```

or

```
<ANYONE> do your part <call> while the others <other call>
```

When used with two calls, this concept and its designator must be entered by itself, like this:

boys do your part (while the others) ⟨ENTER⟩
swing thru ⟨ENTER⟩
switch to a diamond ⟨ENTER⟩

This will come out as:

```
boys do your part swing thru while the girls switch to a diamond
```

This means that the non-designated dancers leave the room and re-form their own setup in another room. The setups are not shrink-wrapped—they stay the same size in each room, with phantoms where the other people are. The appropriate call is done in each room. The non-designees come back, and the two setups are merged. This merging operation is similar to that used for `<ANYONE>`. It may result in things like C1-phantom setups. From boy-boy-girl-girl waves, `boys do your part, hinge` will leave C1 phantom setups, just as `boys, hinge` will. In fact, the difference between these concepts is rather subtle in many cases. The important point is that, in each room, the dancers work in the entire setup. For example, from facing lines, if the centers do their part of `right and left thru`, they work on their own side, with the nonexistent ends, rather than working in the center.

From boy-girl-boy-girl waves, `boys do your part, hinge` creates a mess. No sensible dancer would consider it acceptable. Do not use such things.

The second form of this concept will be printed in the final transcript as `do your part, <ANYONE> <call> while the others <other call>`. You must still enter it as shown above, with the designator first and the phrase '`do your part`' second.

15.10.5 Ignore the ANYONE

The concept is

```
ignore the <ANYONE>, <call>
```

This means that the designated people do nothing, while the others do the call in the distorted or disconnected setup that remains. From the standpoint of the non-designated people, the spots occupied by the designated people do not exist.

This concept has been set at C1. You can use it at lower levels if you issue the `toggle concept levels` command.

15.10.6 Own the ANYONE

The concept is

```
own the <ANYONE>, <call> by <other call>
```

This is similar to the `<ANYONE> do your part` concept, except that the result setups are reassembled according to strict matrix positions. That is, instead of a C1 phantom setup, a 4x4 matrix may result. This is a recognized C3A concept.

15.11 ANYONE Start

It is often useful to designate that certain people are to start a call, with everyone joining in after the first part. There are two general ways to do this:

```
<anyone> start
```

or

```
initially <anyone>
```

These are very nearly the same, and in many cases may be used interchangeably. However, there are differences that you may need to be aware of. These arise most often when the centers are the designated people.

The concept '`<anyone> start`' means that the designated people do their part of the start of the call. They do this in the entire setup. That is, they are aware of the spots occupied by the others.

The concept '`initially <anyone>`' means that the '`<anyone>`' concept is applied to the first part of the call. When a selector name is used as a concept, what happens depends on what the formation is, and on who are designated. If the centers are designated, they do the first part of the call, working in the center. They are usually unaware of the other people around them.

Suppose we have an hourglass and wish to have the centers start a '`reverse order alter the wave`'. The parts of '`reverse order alter the wave`' are '`flip the diamond`', '`diamond counter rotate twice`', '`fan back`', and '`swing`'. We clearly want the centers to work by themselves, flipping the center diamond. The result will be twin diamonds, from which everyone can finish the call.

We could say:

```
initially centers reverse order alter the wave
```

The '`centers`' concept picks out the center diamond, and tells them to work in that diamond.

If we had said

```
centers start reverse order alter the wave
```

it wouldn't work. The centers would try to do their part of a '`flip the diamond`' in the entire hourglass, which is impossible.

A similar thing happens if we have a galaxy and call

```
initially centers tally ho.
```

The centers recognize their 2x2 box, and do a '`1/2 circulate`'. If we had said

```
centers start tally ho
```

the centers would have to try to do a '`1/2 circulate`' in the galaxy, which is impossible.

Suppose we have a galaxy and wish to have the centers begin a '`with confidence`'. That is, the centers '`hinge`' but the ends do *not* do a '`1/2 circulate`'. The first part of '`with confidence`' is '`centers hinge while the ends 1/2 circulate`'. If we say

```
centers start with confidence
```

the centers will do their part of a '`centers hinge`' in the entire galaxy. That means, of course, that they will recognize themselves as centers, and will '`hinge`', so the call will work. If we had said

```
initially centers with confidence
```

the centers would recognize only their center box, from which a '`centers hinge`' is impossible because the centers have been designated twice.

In some cases you can do it either way. From appropriate facing diamonds, you can call

```
centers start swing the fractions
```

or

```
initially centers swing the fractions
```

or just

```
swing the fractions
```

From the setup obtained by having the centers '`step and fold`' from a tidal wave, you can call

```
ends start 2/3 recycle
```

or

```
initially ends 2/3 recycle
```

The ends know how to do the '`centers fold`' in either case.

15.12 12 Matrix and 16 Matrix

When the setup in which a call is to be performed is larger than 8 people due to the presence of phantoms, some modifier is usually required in order to acknowledge the existence of the phantoms' spots. The rules governing the use of such modifiers are not very precisely defined, but Sd attempts to follow accepted usage.

These modifiers are *not* required if some concept has been applied that takes a setup larger than 8 people and forms virtual setups of 8 or fewer people. Examples of such concepts are '`triple boxes`', '`parallelogram`', and '`split phantom waves`'. They are also not required for certain calls that can be done in a setup of any size, such as '`press`', '`truck`', and '`counter rotate`'.

They *are* required whenever a call that is not one of the special ones needs to be extended to more than 8 people, or a setup of more than 8 people needs to be split in order to execute a call in smaller setups.

As an example of the former case, consider offset waves, that is, a 3x4 matrix with the outer waves not fully occupied. If we use a concept like '`offset waves`', '`triple waves`', or '`triple waves working forward`', we can use a call directly. However, if we want to call a 12-person call like '`circulate`' or '`motivate`', we need to legitimize the phantom spots. By saying `12 MATRIX circulate` or `3x4 MATRIX circulate`, we indicate that the phantom spots actually count, and that we want a 12-person version of '`circulate`'. If we are in parallelogram waves, that is, a 2x6 matrix, we can also say `12 MATRIX circulate` or `2x6`

MATRIX circulate. (In this case there are 3 concentric circulate paths of 4 people each.) One could also say 12 MATRIX in roll circulate from some 2x6 matrices. Exactly what large setups a call can be done from depends on the call.

Also, some calls, like 'circulate', can be used with either '12 MATRIX' or '3x4 MATRIX', while other calls require '12 MATRIX'. For example, 12 MATRIX motivate is legal, but 3x4 MATRIX motivate is not. The rationale is that '12 MATRIX' and '16 MATRIX' are call modifiers; 12 MATRIX circulate and 12 MATRIX motivate are calls that have definitions in this setup. Concepts like '3x4 MATRIX', '4x4 MATRIX', '2x6 MATRIX', and '2x8 MATRIX' simply state how big the setup is. The call 'circulate' is defined in such a way that it can be executed in a wide variety of sizes and shapes, simply by knowing the size and shape. 'Motivate' is more restricted. There are calls 12 MATRIX motivate and 16 MATRIX motivate, but there is no general rule for all matrices.

The other case in which a matrix size modifier must be given arises when the setup must be split. In setups of 8 or fewer people, it is well accepted that splitting occurs automatically when required. For example, facing lines are split into facing couples when 'flutter wheel' is called. This phenomenon does not automatically happen when the setup is larger than 8 spots.

For example, from parallelogram columns, you can't just say 2x1 transfer the column. You must legitimize the phantom spots by giving a concept such as '12 matrix' or '2x6 matrix'. The setup will then be split into two end-to-end 2x3's, in each of which the call will be performed.

These concepts all have the additional property that they can expand a smaller setup as required, so, for example, '12 MATRIX circulate' is legal in normal (2x4) columns. It expands the setup to a 2x6 by placing pairs of phantoms on the ends. (Special case: in a 2x4 setup, the '16 MATRIX' call modifier doesn't know whether to expand to a 4x4 or a 2x8, so you may need to use '4X4 MATRIX' or '2X8 MATRIX' explicitly. In other cases, the expansion is clear.)

Explicit matrix concepts can also be used to dictate the precise formation with phantoms when using tandem or as-couples concepts. For example, from two-faced lines, the concept '2X8 MATRIX AS COUPLES' leads to virtual 2x4 lines (in which the ends are all phantoms), '4X4 MATRIX AS COUPLES' leads to virtual 2x4 columns (this happens to be the same as 'PHANTOM AS COUPLES' in this setup), and '4X4 MATRIX BOXES WORK SOLID' leads to a virtual 2x2. '2X8 MATRIX BOXES WORK SOLID' leads to a virtual 1x4. From twin diamonds in which each diamond has all 4 people facing the same way, '16 MATRIX OF PARALLEL DIAMONDS DIAMONDSOME' leads to a virtual 1x4 in which both ends are phantoms. 'DIAMONDSOME' leads to a virtual 1x2.

16 Miscellaneous Advice and Warnings

If a call that you believe is legal does not appear on the menu, click on `allow modifications`. This will bring up the universal call menu. You can then thrash out which of you or the computer is correct.

The program sometimes gives non-fatal *warnings* about calls. These are intended principally as advice to you, the user of the program, that the call you have just entered may not be a reasonable thing to call. It is not necessarily the case that reading the warning aloud when calling the sequence is the correct thing to do, though it might be. The call might simply be inappropriate, or it might be wise to say something along the lines of "be careful; everyone do your part." The significance of a warning may also depend on the level. Some things that are extremely peculiar but barely legal at high challenge levels are simply not done at lower levels. You should use your judgement.

Warning: The database is not without errors or misunderstandings in nonstandard uses of calls. The program generally tries to be extremely conservative. However, you should not blindly accept what the program does, particularly if a call was used in an unusual way. The management will not be responsible for any sequences left unattended.

Warning: Some combinations of things that seem obvious to the program might not be agreed to by the dancers. Particularly at high challenge levels, some things are controversial. Proceed with caution. Unless you agree wholeheartedly with what the program did, and believe that the dancers will also agree, it may be best not to do it. Don't stack outrageous interrupts and replacements, with concepts going every which way, unless you are prepared to explain yourself at the end of the tip.

Warning: The program's notion of levels is only keyed to concepts and calls, not to concept/call combinations, and not to calls in the context of certain setups. For example, since *split* and *square thru* are both Mainstream, it thinks that *split square thru* is Mainstream, even though it is A1. Also, *chain reaction* should only be used from a normal 1/4-tag setup at A1, but the program doesn't know this.

For debugging, there is a hidden level called `all`, which is above C4. At this level, even the invisible calls that are used in the definitions of other calls are legal. They typically begin with an underscore. Also, when the `all` level is selected, all sequentially-defined calls are fractionalizable, even if they aren't in practice.

With the X Window System interface, the font for the text output window must be fixed-width so the formation pictures are legible. If you do not get this by default, set this X resource:

```
Sd*text*font: -misc-fixed-medium-r-normal--13-*-*-*-c-70-iso8859-1
```

17 Customization with X Resources

The file '`Sd.res`' is a sample X user resource file for the program. It contains values for a number of parameters, including most of the text on the screen that is not fetched from the calls database.

All values in the distributed '`Sd.res`' file—except the colors—are the defaults and are in the file for illustration only. To change a value, copy its line into the file '`$HOME/Sd`' and make the change you want.

People who have many user resource files for different applications may wish to store them in a directory other than their home directory. To do so, set the environment variable `XAPPLRESDIR` to the directory where your resource files are. For example, one of the authors uses the subdirectory '`app-resources`', and his .cshrc contains this line:

```
setenv XAPPLRESDIR ~/app-resources
```

To use a different value for any of these parameters, either edit the resource file or load the changed value into your X server's resource database. To modify the server resource database, put a modified version of the appropriate line in the '`.Xresources`' file in your home directory (assuming you run `xrdb .Xresources` from '`.xsession`' or whatever your X startup file is called). You will have to re-run `xrdb` after editing your '`.Xresources`' file. You should add '`Sd`' to the front of any lines you move to your '`.Xresources`' file.