CONTENTS

NAME

sc – spreadsheet calculator

SYNOPSIS

sc [**–aCcemnoqRrvx**] [**–P** *range*[*/address*] | */address*] [**–W** *range*] [*file* . . .]

DESCRIPTION

The spreadsheet calculator *sc* is based on rectangular tables much like a financial spreadsheet. When invoked, it first looks for a file in the user's home directory called .scrc and if found, reads that file into memory. If that file contains the command "*set scrc*", *sc* looks for a file called .scrc in the current directory, and if found, reads that file into memory, too. Next, it reads the options from the command line, and finally, it reads in the file or files specified on the command line and presents the data in a table organized as rows and columns of cells. If invoked without a *file* argument, the table is initially empty, unless it is running in a pipeline, in which case it will read its data from the standard input. If more than one file is specified, all files except the first one will be merged. The default filename for saving a file with the *Put* command will be the same as the first file specified, and the other files will be treated as macros. If you want to use advanced macros from the command line, the "|" must be quoted to prevent it from being expanded by the shell.

Options begin with –. However, an argument of a single – will be interpreted to mean that spreadsheet data will be taken from the standard input. This is useful for including *sc* in a pipeline if the system supports pipes. However, if standard input is not a terminal, the – is only necessary if there are multiple files and standard input is not the last to be read, since standard input is automatically read in after all other files in such cases if it is not specified explicitly, or if there are no other filenames on the com-

mand line. If *sc* is included in a pipeline, and a filename of '–' is not specified, the standard input will be merged in after all of the other named files have been processed.

The first argument not beginning with a –, or a single – by itself, and any subsequent arguments will all be interpreted as filenames (a filename of – meaning standard input as noted above). In addition, an argument of — may be used to signify that all subsequent arguments should be treated as filenames even if they begin with a –, but unlike –, — won't be treated as a filename itself.

Each cell may have associated with it a numeric value, a label string, and/or an expression (formula) which evaluates to a numeric value or label string, often based on other cell values.

For an online tutorial, type the command:

sc /usr/local/share/sc/tutorial.sc

To print a quick reference card, type the command:

scqref | [your_printer_commmand]

OPTIONS

–a Do not run the autorun macro, if one is specified in the file.

–c Start the program with the recalculation being done in column order.

–e Start the program with round-to-even (banker's rounding) enabled.

–m Start the program with automatic recalculation disabled. The spreadsheet will be recalculated only when the "@" command is used.

–n Start the program in quick numeric entry mode (see below).

–o Start the program with automatic optimization of expressions enabled.

–q Quit after loading all files, but before becoming interactive. This is useful in shell scripts for getting information from a file, for example, or using *sc* as a non-interactive calculator using the *eval* command.

–r Start the program with the recalculation being done in row order (default option).

–v When piping data out using the *–P* option (below), change all expressions to values. The *–v* option must precede the *–P* option to have an effect. If the *–P* option is used more than once, there must be a separate *–v* option for each instance of the *–P* option.

–x Cause the *Get* and *Put* commands (see below) to encrypt and decrypt data files.

–C Start the program with automatic newline action set to increment the column (see below).

–P *range[/address]*
–P */address*

Pipe a range to standard output. The output is similar to that of the *Put* command (below), except that only cell data and formatting information for cells in the range are output, without all of the colors, range definitions, column formatting, etc. The optional */address* is used to adjust all addresses in the range to a new starting point. This is useful for copying data from one file to another, especially when used in conjunction with the *–v* option (above), using something like *merge "|sc –v –Prange/address filename"* (note the pipe symbol). This option may be used more than once to specify multiple ranges. Note, however, that the *–v* option must precede the *–P* option on the command line, and there must be a separate *–v* option for each instance of the *–P* option. Any instance of *–P* not preceded by its own *–v* option will output unevaluated expressions.

A range of "*%*" may be used to refer to the entire spreadsheet. If the range is left out, as shown in the second form above, *sc* will be started interactively in navigate mode, allowing you to navigate the spreadsheet and highlight the range you want to output. Pressing ESC, ^G, or q will terminate without outputting any data.

–R Start the program with automatic newline action set to increment the row (see below).

–W Pipe a range to standard output. The output is identical to that of the *Write* command (below). This option may be used more than once to specify multiple ranges. A range of "*%*" may be used to refer to the entire spreadsheet.

All of these options can be changed with the *^T* and *S* commands (see below) while *sc* is running. Options specified when *sc* is invoked override options saved in the data file.

Personal Initialization File

When *sc* first starts, it looks for a file in the user's home directory called .scrc and if found, loads it into memory. The format of this file is the same as any other *sc* file, but should be reserved for setting certain defaults. Any options set which have equivalent command line options may be overridden by the command line. If that file contains the command "*set scrc*", *sc* will then look for a file called .scrc in the current directory, and if found, load that file into memory, too (this is analogous to the "set exrc" command used by vi/ex). These "dotfiles" may be created by any text editor. Several commands exist specifically for setting default file name extensions in the .scrc file, although they may also be used from macros, ordinary spreadsheet files, or from within *sc* at the command line. They will not, however, be saved along with the file. The extensions should be quoted, and should not include the preceding '*.*' (e.g., *scext "sc"* will add the extension *.sc*). These commands are:

scext This is the default extension for normal *sc* files (those created with the *Put* command). If this command is not used, all *sc* files will be saved without an extension, and any existing extension will not be removed. Setting this option causes all *sc* files to be saved with the specified extension added, unless it is already present. If the file name already has an extension of *.sc*, it will first be removed. Any other extension will not be removed.

ascext This is the default extension for plain text files created with the *Write* command. The file name will first be checked to see if it already has an extension of either *.sc* or the extension specified with *scext* above, and if either one exists, it will first be removed before adding the new extension. If this option is not set, a default of *.asc* will be used.

tbl0ext This is the default extension for files created with the *Tbl* command if tblstyle is set to 0 (default). The file name will first be checked to see if it already has an extension of either *.sc* or the extension specified with *scext* above, and if either one exists, it will first be removed before adding the new extension. If this option is not set, a default of *.cln* will be used.

tblext This is the default extension for files created with the *Tbl* command if tblstyle is set to tbl. The file name will first be checked to see if it already has an extension of either *.sc* or the extension specified with *scext* above, and if either one exists, it will first be removed before adding the new extension. If this option is not set, a default of *.tbl* will be used.

latexext

This is the default extension for files created with the *Tbl* command if tblstyle is set to latex. The file name will first be checked to see if it already has an extension of either *.sc* or the extension specified with *scext* above, and if either one exists, it will first be removed before adding the new extension. If this option is not set, a default of *.lat* will be used.

slatexext

This is the default extension for files created with the *Tbl* command if tblstyle is set to slatex. The file name will first be checked to see if it already has an extension of either *.sc* or the extension specified with *scext* above, and if either one exists, it will first be removed before adding the new extension. If this option is not set, a default of *.stx* will be used.

texext This is the default extension for files created with the *Tbl* command if tblstyle is set to tex. The file name will first be checked to see if it already has an extension of either *.sc* or the extension specified with *scext* above, and if either one exists, it will first be removed before adding the new extension. If this option is not set, a default of *.tex* will be used.

General Information

The screen is divided into four regions. The top line is for entering commands and displaying cell values. The second line is for messages from *sc*. The third line and the first four columns show the column and row numbers, from which are derived cell addresses, e.g. *A0* for the cell in column A, row 0. Note that column names are case-insensitive: you can enter *A0* or *a0*.

The rest of the screen forms a window looking at a portion of the table. The total number of display rows and columns available, hence the number of table rows and columns displayed, is set by *curses*(3) and may be overridden by setting the LINES and COLUMNS environment variables, respectively.

The screen has two cursors: a cell cursor, indicated by either a highlighted cell or a "<" on the screen, and a character cursor, indicated by the terminal's hardware cursor.

If a cell's numeric value is wider than the column width (see the *f* command), the cell is filled with asterisks. If a cell's label string is wider than the column width, it is truncated at the start of the next non-blank cell in the row, if any.

Cursor control commands and row and column commands can be prefixed by a numeric argument which indicates how many times the command is to be executed. You can type *^U* before a repeat count if quick numeric entry mode is enabled.

Changing Options

^T*o* Toggle options. This command allows you to switch the state of one option selected by *o*. A small menu lists the choices for *o* when you type *^T*. Unless otherwise noted, the options selected are saved when the data and formulas are saved so that you will have the same setup next time you enter the spreadsheet.

a Automatic Recalculation. When set, each change in the spreadsheet causes the entire spreadsheet be recalculated. Normally this is not noticeable, but for very large spreadsheets, it may be faster to clear automatic recalculation mode and update the spreadsheet via explicit "@" commands. Default is automatic recalculation on.

b Braille enhancement mode. See the braille section under the *Set* command below for a complete description of how to use this mode. This option is not saved when saving a file, to allow blind and sighted users to easily share files. It is intended for use in a user's *.scrc* file.

c Current cell highlighting. If enabled, the current cell is highlighted (using the terminal's standout mode, if available) and the cell pointer "<" is turned off. This is enabled by default.

e External function execution. When disabled, external functions (see *@ext*() below) are not called. This saves a lot of time at each screen update. External functions are disabled by default. If disabled, and external functions are used anywhere, a warning is printed each time the screen is updated, and the result of *@ext*() is the value from the previous call, if any, or a null string.

i Automatic insertion of rows/columns. If this is enabled and craction is set to move the cell cursor either down or to the right after entering data into a cell, and the last cell in a row/column in the scrolling portion of a framed range was just filled, causing the cell cursor to move outside of this range, a new column/row will be inserted, thus enlarging the range and allowing you to continue entering data into the row/column without overwriting the frame (which may contain expressions of some sort, such as totals). If autowrap is also enabled, it will take precedence, and a new row/column will only be inserted after entering data in the very last cell (bottom right corner) of the scrolling range. The default is no automatic insertion.

w Automatic wrap to next row/column. If this is enabled and craction is set to move the cell cursor either down or to the right after entering data into a cell, and the last cell in

a row/column in the scrolling portion of a framed range was just filled, causing the cell cursor to move outside of this range, the cell cursor will move to the first cell in the next row/column in this range. If this would also take the cursor out of the scrolling portion of the range, the cursor will remain in last edited cell instead, unless autoinsert is also enabled, in which case a new row/column will be added so that the cursor can wrap. The default is no autowrap.

l Autolabeling. If enabled, using the define command (rd) causes a label to be automatically generated in the cell to the left of the defined cell. This is only done if the cell to the left is empty. Default is enabled.

n Quick numeric entry. If enabled, a typed digit is assumed to be the start of a numeric value for the current cell, not a repeat count, unless preceded by *^U*. Also, the '+' and '–' keys will enter insert mode and append a '+' or '–' to the existing contents of the cell, allowing the user to easily add to or subtract from the current numeric contents of the cell. The cursor controls (*^P*, *^N*, or any of the arrow keys) in this mode will end a numeric entry if the entry was started by pressing '+', '–', or a digit. Switching from insert mode to edit mode will cause the cursor controls to revert to their normal functions.

o Automatic optimization of expressions. If this is enabled, expressions which evaluate to a constant are automatically optimized upon entry. For example, if you enter @pow(2,32) into a cell, the value 4294967296 will be stored in that cell, whereas if optimization is turned off, the calculated value will be displayed, but the actual expression will be stored in the cell instead. This allows you to edit the expression instead of re-entering it from scratch when you just want to make a minor change. Default is automatic optimization off.

t Top line display. If enabled, the name and value of the current cell is displayed on the top line. If there is an associated label string, the first character of the string value is "|" for a centered string, "<" for a leftstring or ">" for a rightstring (see below), followed by "*string*" for a constant string or {*expr*} for a string expression. A constant string may be preceeded with a backslash ('\'). In this case the constant string will be used as a "wheel" to fill a column, e.g. "\-" for a line in a column, and "\Yeh " for "Yeh Yeh Ye". If the cell has a numeric value, it follows as [*value*], which may be a constant or expression.

$ Dollar prescale. If enabled, all numeric **constants** (not expressions) which you enter are multipled by 0.01 so you don't have to keep typing the decimal point if you enter lots of dollar figures.

r Newline action. This is a 3-way toggle which determines which direction to move after pressing the RETURN key to enter data into a cell. It has the same effect as using the *set* (S) command to set the value of craction. After selecting this option, you will be prompted for the direction you want to move. Valid directions are down (craction=1) and to the right (craction=2). Pressing j, ^N, or the cursor-down key will cause the cursor to move down a cell each time you press the RETURN key and pressing l, the cursor-right key, or the space bar will cause the cursor to move one cell to the right. Pressing the RETURN key at the prompt selects no action (craction=0, which means that the cursor will remain in the current cell). No action is the default unless *sc* is started with either the –R or –C option. This option is ignored if the cell into which data is being entered is not the current cell.

s Enable/disable color slop. If a cell's label string is wider than the column width, it will slop over into the next cell to the right if that cell is empty. However, if that cell is in a different color range than the first, this slopover will be disabled, regardless of whether the colors assigned to the two ranges are different or not. If cslop is enabled, strings may slop over even if the next cell is in a different color range, carrying their

color with them, which may cause a ragged boundary between the ranges, but may allow the strings to be seen in their entirety. Cslop is disabled by default.

x Encryption. See the **–x** option.

z Set newline action limits. This option sets limits to the newline action option above. When this option is invoked, the row and column of the current cell are remembered. If a later newline action would take the current cell to the right of the remembered column, then the current cell is instead moved to the first column of the next row. If a newline action would take the current cell below the remembered row, then the current cell is instead moved to the top row of the next column.

C Color. This option enables color, and must be set before any other color options, such as colorneg (color negative numbers) or colorerr (color cells with errors), will have an effect. On a slow connection, turning off color can noticeably speed up screen updates.

E Color cells with errors. Setting this option will cause all cells with expressions which evaluate to ERROR or INVALID to be set to color 3. Color must be enabled for this option to take effect.

N Color negative numbers. When this option is set, all cells containing negative numbers will have their color number incremented by one. Cells with color 8 will cycle back to color 1. Color must be enabled for this option to take effect.

The quick numeric entry, newline action and set newline action limits options can be combined to allow very quick entry of large amounts of data. If all the data to be entered is in a single row or column then setting the quick numeric entry and the appropriate newline action will allow the numbers to be entered without any explicit commands to position the current cell or enter a number.

If the data entry involves several entries in each row for many rows, then setting the quick numeric entry option, setting the newline action to move right after each entry and setting the newline action limits on the last column on which data should be entered will allow the data to entered quickly. An alternative to setting newline action limits is to enclose the range for entry in a frame (see “Framed Ranges” below), and setting the autowrap option. Setting autoinsert will insert new rows as needed if the frame includes data at the bottom. If necessary, columns which do not need data to be entered can be hidden with the **z** command. Similar arrangements can be made for entering several rows of data in each column.

S Set options. This command allows you to set various options. A small menu lists the options that cannot be changed through *^T* above.

byrows/**bycols**
Specify the order cell evaluation when updating. These options also affect the order in which cells are filled (see *rf*) and whether a row or column is cleared by an *x* command.

iterations=*n*
Set the maximum number of recalculations before the screen is displayed again. *Iterations* is set to 10 by default.

tblstyle=*s*
Control the output of the *T* command. *s* can be: **0** (default) to give colon delimited fields, with no *tbl* control lines; **tbl** to give colon delimited fields, with *tbl*(1) control lines; **latex** to give a *LaTeX* tabular environment; **slatex** to give a *SLaTeX (Scandinavian LaTeX)* tabular environment; **tex** to give a *TeX* simple tabbed alignment with ampersands as delimiters; and **frame** to give a tblstyle output for FrameMaker.

pagesize=*n*

Set the page size for the PageUp, PageDown, J, and K commands. If set to 0, the default is to move up or down half the number of rows displayed on the screen, or if the current cell is in a framed range, half the number of displayed rows in the scrolling region of that range.

Other *Set* options are normally used only in *sc* data files since they are available through *^T*. You can also use them interactively.

autocalc/!autocalc
Set/clear auto recalculation mode.

autoinsert/!autoinsert
Set/clear automatic insertion mode.

autowrap/!autowrap
Set/clear autowrap mode.

optimize/!optimize
Set/clear auto optimize mode.

numeric/!numeric
Set/clear numeric mode.

prescale/!prescale
Set/clear numeric prescale mode.

extfun/!extfun
Enable/disable external functions.

toprow/!toprow
Set/clear top row display mode.

rndtoeven/!rndtoeven
Default: *.5 will be rounded up to the next integer; doing a 'set rndtoeven' will cause it to be rounded to the closest even number instead (aka banker's rounding). Round-to-even has advantages over the default rounding for some applications. For example, if X+Y is an integer, then X+Y = rnd(X)+rnd(Y) with round-to-even, but not always with the defaulting rounding method. This could be an advantage, for example, when trying to split an odd amount of money evenly between two people (it would determine who gets the extra penny). Note: rndtoeven only effects the @rnd and @round functions. It has no effect on how a number is rounded to fit the display format of a cell.

craction=*n*
Set the newline action. *n* can be: **0** (default) to give no action; **1** to move down after each entry; or **2** to move right after each entry.

rowlimit=*n*
Set the remembered limit for the maximum row below which the current cell will be moved to the top of the next column if the newline action is set to move the current cell down. *n* can be **–1** (default) to disable this facility.

collimit=*n*
Set the remembered limit for the maximum column to the right of which the current cell will be moved to the left of the next row if the newline action is set to move the current cell right. *n* can be **–1** (default) to disable this facility.

color/!color
Enable color. This option must be set for any other color options, such as colorneg or colorerr, to take effect. On a slow connection, turning off color can noticeably speed up screen updates.

colorneg/!colorneg

Color negative numbers. When this option is set, all cells containing negative numbers will have their color number increased by one. Cells with color 8 will cycle back to color 1. Color must be enabled for this option to take effect.

colorerr/!colorerr

Color cells with errors. Setting this option will cause all cells with expressions which evaluate to ERROR or INVALID to be set to color 3. Color must be enabled for this option to take effect.

cslop/!cslop

Enable color slop. If a cell's label string is wider than the column width, it will slop over into the next cell to the right if that cell is empty. However, if that cell is in a different color range than the first, this slop over will be disabled, regardless of whether the colors assigned to the two ranges are different or not. If cslop is enabled, strings may slop over even if the next cell is in a different color range, carrying their color with them, which may cause a ragged boundary between the ranges, but may allow the strings to be seen in their entirety. Cslop is disabled by default.

The following *Set* options are considered personal preferences, or are terminal dependent, and are therefore not saved when saving a file, but are instead intended for use in a user's .scrc file.

braille/!braille

Set/clear braille enhancement mode. When braille enhancement mode is set, the cursor behaves in a manner that makes the use of *sc* much easier when using a braille display. In spite of its name, this mode also works well with screen readers such as SpeakUp, and can even be used by sighted users to make cutting and pasting using the *screen* program much easier.

There are actually two different braille modes. When the braille option is set, the *C* command, which is normally used to set colors, will instead change from one braille mode to the other. If it is desired to set/change colors so you can share files with others not using a braille display, braille mode will have to be switched off temporarily, and then switched back on after the color operation is done.

When the braille option is set, the default braille mode will cause the cursor to be positioned at the left edge of the current cell, while the alternate braille mode will cause the cursor to be placed at the beginning of the top line, which will contain information such as the current cell address, contents of the cell, and column formatting information. The column names will also be moved to the left edge of their respective columns in order to remain aligned with the cursor as it moves up and down the column.

In either mode, the cursor will be placed in the top line when editing a line, except when switching to navigate mode, in which case the cursor will be placed in either the current cell (default braille mode) or the second line, where the cell address or default range will be displayed (alternate braille mode).

Whenever a message is displayed on the second line, such as an error message or prompt for further information, both modes will cause the cursor to be placed at the beginning of that message. After this message goes away, the cursor will revert to its former behavior. The easiest way to make this message go away without effecting anything, except in the cases where it is asking the user for more information, is to press *CC*, which effectively changes modes twice, with a net effect of leaving *sc* in the original mode.

locale/!locale
If locale support is compiled into *sc*, this option will cause certain locale-dependent behaviors, such as the display of numbers and the determination of word boundaries for some operations in edit mode. Note that if this option is set and the environment variable LC_ALL is unrecognized, unset, or set to either "POSIX" or "C", commas in format commands will be ignored.

cellcur/!cellcur
Set/clear current cell highlighting mode. This option is included here because it is likely to be terminal dependent and/or a user preference, and therefore is not saved when saving a file.

scrc It tells *sc* to also read the file .scrc in the current directory when starting. Settings in this file will override those in $HOME/.scrc but may themselves be overridden by command line options. Setting this could be a potential security risk, since starting *sc* with an unknown .scrc could potentially execute arbitrary commands. This risk is probably very slight, since a spreadsheet program is not likely to be run in just any directory, and should **never** be run as root.

backup/!backup
Before the database is written to an existing file a backup of that file can be saved as *filename*~. This is enabled/disabled by this option. The default is **not** to do backups.

Cursor Control Commands

^A Go to cell *A0* (same as HOME).

^P Move the cell cursor up to the previous row.

^N Move the cell cursor down to the next row.

^H Move the cell cursor backward one column.

SPACE Move the cell cursor forward one column. When in navigate mode, if a range is highlighted, insert the highlighted range into the command line, followed by a space, while remaining in navigate mode. This is useful when entering copy, move, or frame commands, for example, which accept more than one range argument.

h, j, k, l
These are alternate, *vi*-compatible cell cursor controls (left, down, up, right). Space is just like l (right).

H, J, K, L
These move the cursor by half pages (left, down, up, right). If *pagesize* is nonzero, up/down paging will be by *pagesize* rows, instead.

^F, ^B Same as J and K above.

PAGE-DOWN PAGE-UP
Same as J and K above.

TAB If the character cursor is on the top line, TAB tries to complete a range name if the character immediately preceding it is alphanumeric or "_", and starts a range if not (see below). Otherwise, move the cell cursor forward one column.

HOME Go to cell *A0*.

END Same as ^E (see below).

Arrow Keys
The terminal's arrow keys provide another alternate set of cell cursor controls if they exist and are supported in the appropriate *termcap* entry. Some terminals have arrow keys which conflict with other control key codes. For example, a terminal might send *^H* when the back arrow key

is pressed. In these cases, the conflicting arrow key performs the same function as the key combination it mimics.

^ Move the cell cursor up to row 0 of the current column.

Move the cell cursor down to the last valid row of the current column.

0 Move the cell cursor backward to column A of the current row. This command must be prefixed with *^U* if quick numeric entry mode is enabled.

$ Move the cell cursor forward to the last valid column of the current row.

b Scan the cursor backward (left and up) to the previous valid cell.

w Scan the cursor forward (right and down) to the next valid cell.

g Go to a cell. *sc* prompts for a cell's name, a regular expression surrounded by quotes, or a number. If a cell's name such as *ae122* or the name of a defined range is given, the cell cursor goes directly to that cell. If a quoted regular expression such as " *Tax Table* " or " *^Jan [0-9]*$* " is given, *sc* searches for a cell containing a string matching the regular expression. See *regex(3)* or *ed(1)* for more details on the form of regular expressions.

You can also search formatted numbers or expressions using regular expressions by preceding the opening quotes of the regular expression with a "#" (for formatted numbers) or a "%" (for expressions). These are handy for searching for dates within a specified range or cells which reference a given cell, for example, although they are somewhat slower than searching through ordinary strings, since all numbers must be formatted or expressions decompiled on the fly during the search.

If a number is given, *sc* will search for a cell containing that number. Searches for either strings or numbers proceed forward from the current cell, wrapping back to a0 at the end of the table, and terminate at the current cell if the string or number is not found. You may also go to a cell with an ERROR (divide by zero, etc. in this cell) or INVALID (references a cell containing an ERROR). *g error* will take you to the next ERROR, while *g invalid* take you to the next INVALID. The last *g* command is saved, and can be re-issued by entering *g*<*return*>. You can also repeat the last search by pressing *n*.

An optional second argument is available whose meaning depends on whether you're doing a search or jumping to a specific cell. When doing a search, the second argument specifies a range to search. When jumping to a specific cell, the second argument specifies which cell should be in the upper lefthand corner of the screen, if possible, which allows you to position the destination cell where you want it on the screen.

^E*d* Go to end of range. Follow *^E* by a direction indicator such as *^P* or *j*. If the cell cursor starts on a non-blank cell, it goes in the indicated direction until the last non-blank adjacent cell. If the cell cursor starts on a blank cell, it goes in the indicated direction until the first non-blank cell. This command is useful when specifying ranges of adjacent cells (see below), especially when the range is bigger than the visible window.

If *^E* is pressed twice in succession, or if it is pressed after another *^E* or a *^Y*, it will cause the screen to scroll up without moving the cell cursor, unless the cell cursor is already at the top of the screen, in which case, it will remain at the top of the visible screen.

^Y Causes the screen to scroll down without moving the cell cursor, unless the cell cursor is already at the bottom of the screen, in which case, it will remain at the bottom of the visible screen.

m*x* Mark the current cell. *sc* will prompt for a lowercase letter to be used as a mark specifier. Marked cells may be used as the source for the *c* (copy a marked cell) command, or as the target of a ' or ' (go to marked cell) command. In addition to cells marked with lowercase letters, *sc* also automatically marks the last nine cells that have been edited with the numbers 1-9, and the current cell being edited with the number 0. When not editing a cell, marks 0 and 1 usual-

ly refer to the same cell, unless the last edit was begun in one cell, but the cell address was changed before pressing the RETURN key, or the last edit was aborted prematurely.

‘*x* Jump to a previously marked cell. If the target cell is currently on the screen, *sc* will simply jump to the target cell, making it current. Otherwise, *sc* will attempt to center the cell on the screen, if possible. As a special case, following the ‘ with another ‘ will return you to the cell you were in before the last g, ’, ‘, *, or ^E (or END key) was used to jump to another cell.

’*x* Jump to a previously marked cell. ’ works just like ‘ , except that ’ will attempt to restore the marked cell to the same position on the screen as when it was marked. It does this by remembering which cell was in the upper left hand corner of the screen at the time the mark was set, and restoring that cell to its original position. As a special case, following the ’ with another ’ will return you to the cell you were in before the last g, ’, ‘, *, or ^E (or END key) was used to jump to another cell, and will also try to position that cell on the screen in the same position as when you left it.

z<RETURN>
Move the current row to the top of the screen. If the current row is in a framed range, move the current row to the top of the scrolling region.

z. Move the current row to the center of the screen.

z| Move the current column to the center of the screen.

zc Center the current cell both horizontally and vertically.

Cell Entry and Editing Commands

Cells can contain both a numeric value and a string value. Either value can be the result of an expression, but not both at once, i.e. each cell can have only one expression associated with it. Entering a valid numeric expression alters the cell’s previous numeric value, if any, and replaces the cell’s previous string expression, if any, leaving only the previously computed constant label string. Likewise, entering a valid string expression alters the cell’s the previous label string, if any, and replaces the cell’s previous numeric expression, if any, leaving only the previously computed constant numeric value.

= Enter a numeric constant or expression into the current cell. *sc* prompts for the expression on the top line. The usual way to enter a number into a cell is to type “=”, then enter the number in response to the prompt on the top line. The quick numeric entry option, enabled through the **–n** option or *^T* command, shows the prompt when you enter the first digit of a number (you can skip typing “=”). If you want to begin entering an expression in the current cell, but you want to start out in navigate mode (e.g. to enter cell addresses, or sums of ranges using “@sum”), use the “+” command instead (see below).

< Enter a label string into the current cell to be flushed left against the left edge of the cell.

**** Enter a label string into the current cell to be centered in the column.

> Enter a label string into the current cell to be flushed right against the right edge of the cell.

{ Left justify the string in the current cell.

| Center the string in the current cell.

} Right justify the string in the current cell.

F Enter a format string into the current cell. This format string overrides the precision specified with the *f* command unless *&* is present in the fractional part of the format string (see below). The format only applies to numeric values. There are two types of format strings allowed: standard numeric and date. (Note: these format strings may also be used with the *f* command to create user-defined format types.) The following characters can be used to build a standard numeric format string:

Digit placeholder. If the number has fewer digits on either side of the decimal point than there are ‘#’ characters in the format, the extra ‘#’ characters are ignored. The

number is rounded to the number of digit placeholders as there are to the right of the decimal point. If there are more digits in the number than there are digit placeholders on the left side of the decimal point, then those digits are displayed.

0 Digit placeholder. Same as for '#' except that the number is padded with zeroes on either side of the decimal point. The number of zeroes used in padding is determined by the number of digit placeholders after the '0' for digits on the left side of the decimal point and by the number of digit placeholders before the '0' for digits on the right side of the decimal point.

. Decimal point. Determines how many digits are placed on the right and left sides of the decimal point in the number. If *locale* is set, the decimal point for the user's current locale will be used when formatting a number. Note that numbers smaller than 1 will begin with a decimal point if the left side of the decimal point contains only a '#' digit placeholder. Use a '0' placeholder to get a leading zero in decimal formats.

% Percentage. For each '%' character in the format, the actual number gets multiplied by 100 (only for purposes of formatting — the original number is left unmodified) and the '%' character is placed in the same position as it is in the format.

, Thousands separator. The presence of a ',' in the format (multiple commas are treated as one) will cause the number to be formatted with a ',' separating each set of three digits in the integer part of the number with numbering beginning from the right end of the integer. If *locale* is set, the thousands separator for the user's current locale will be used in place of the comma. If the environment variable LC_ALL is unset, unrecognized, or is set to "POSIX" or "C", any commas in the format string will be ignored.

& Precision. When this character is present in the fractional part of the number, it is equivalent to a number of 0's equal to the precision specified in the column format command. For example, if the precision is 3, '&' is equivalent to '000'.

\ Quote. This character causes the next character to be inserted into the formatted string directly with no special interpretation.

E– E+ e– e+
Scientific format. Causes the number to be formatted in scientific notation. The case of the 'E' or 'e' given is preserved. If the format uses a '+', then the sign is always given for the exponent value. If the format uses a '–', then the sign is only given when the exponent value is negative. Note that if there is no digit placeholder following the '+' or '–', then that part of the formatted number is left out. In general, there should be one or more digit placeholders after the '+' or '–'.

; Format selector. Use this character to separate the format into two distinct formats. The format to the left of the ';' character will be used if the number given is zero or positive. The format to the right of the ';' character is used if the number given is negative.

Some example formats are integer ("0" or "#"), fixed ("0.00"), percentage ("0%" or "0.00%"), scientific ("0.00E+00"), and currency ("$#,0.00;($#,0.00)").

Date format strings are identified by the presence of a ^D in the first postition. If this is present, the remainder of the string is passed to the strftime() function, and therefore uses the same conversion specifiers as strftime(). For more information on conversion specifiers for date format strings, see the man page for strftime(3).

Strings you enter must start with ". You can leave off the trailing " and *sc* will add it for you. You can also enter a string expression by backspacing over the opening " in the prompt.

e Edit the value associated with the current cell. This is identical to "=" except that the command line starts out containing the old numeric value or expression associated with the cell. The editing in this mode is vi-like.

^H Move back a character

^V, v Enter navigate mode. This mode allows you to navigate the spreadsheet while editing a command. When in navigate mode, *v* will insert the numeric value of the current cell, if any, into the command line, instead, while *^V* will return to the previous mode (like the ESCAPE key).

^W Insert the expression attached to the current cell into the command line. If there is none, the result is "?". This only works while in navigate mode.

^A In navigate mode, go to cell *A0*. When not in navigate mode, jump to the beginning of the line instead.

^E Jump to the end of the line. Unlike "$" (below), this can also be used from insert mode.

TAB If the character immediately preceding the cursor is alphanumeric or "_", TAB tries to find a match in the list of range names, and if one is found, the name will be completed on the command line. If there are multiple matches, pressing TAB repeatedly without any other intervening keys will cycle through all of the valid matches. If the character immediately preceding the cursor is not alphanumeric or "_", TAB defines a range of cells via the cursor control commands or the arrow keys. Pressing TAB automatically switches *sc* to navigate mode if you haven't already done so using the *^V* command, and the range is highlighted, starting at the cell where you typed TAB, and continuing through the current cell. Pressing TAB again causes the highlighted range to be inserted into the command line, the highlighting to be turned off, and the previous mode to be restored. This is most useful for defining ranges to functions such as *@sum*(). Pressing ")" acts just like typing the TAB key the second time and adds the closing ")". Note that when you give a range command, if the first argument to the command is a range, you don't need to press the first TAB to begin defining a range starting with the current cell.

: Synonym for TAB, when in navigate mode.

‘ ’ In navigate mode, go to marked cell.

***** In navigate mode, go to note linked to current cell.

+ Forward through history (same as j)

– Backward through history (same as k)

ESC Done editing

CR Save. When in navigate mode, insert the name of the current cell (the one at the cell cursor) into the command line. This is useful when entering expressions which refer to other cells in the table.

$ Goto last column

% Goto matching parenthesis

. Insert current dot buffer. When in navigate mode, this is a synonym for : or TAB.

; Repeat the last f, F, t, or T command.

, Repeat the last f, F, t, or T command, but in the reverse direction.

~ Change the case of the character under the cursor.

/ Search backwards for a string in the history
ESC edit the string you typed
CR search
^H backspace

? Search forward for a string in the history (see "/" above)

0 Goto column 0

B Move back a word. Like *b*, except words are space delimited only.

C Change to end of line (delete first, then enter insert mode)

D Delete to end of line

F Find the next char typed, moving backwards in the line

G Go to the end of history, i.e., to the line being currently entered

I Insert at column 0; ESC revert back to edit mode

N Repeat the last search in the opposite direction

P Insert the most recently deleted text before the cursor

R Replace mode; ESC revert back to edit mode

T Goto a char, moving backwards in the line

W Forward a word. Like *w*, except words are space delimited only.

X Delete the char to the left

a Append after cursor; ESC revert back to edit mode

b Move back a word

c Change mode; ESC revert back to edit mode. In navigate mode, insert color range which includes the current cell.

d Delete ...

0	delete to beginning of line
$	delete to end of line
b	back word
e	delete to end of word
f	forward (right)
h	back char
l	forward
t	delete forward up to a given char (next char typed)
w	delete next word forward

e Forward to next end-of-word

f Find the next char typed. In navigate mode, insert the outer frame range which includes the current cell.

g In navigate mode, allows you to 'goto' a cell or range, just like the regular *goto* command. Ignored in edit, insert or replace modes.

h Move left a char

i Insert before cursor; ESC revert back to edit mode

j Forward through history (same as +)

k Backward through history (same as –)

l Move right a char

n Repeat the last search (find the next match)

o When highlighting a range in navigate mode, move to the opposite corner of the highlighted range.

p Insert the most recently deleted text after the cursor

q Stop editing

r Replace char. In navigate mode, insert the inner frame range which includes the current cell.

s Delete current char and enter insert mode (stands for substitute)

t Goto a char

u Undo

w Forward a word

x Delete the current char (moving to the right)

y Copies to the delete buffer without deleting. Use like d (above).

E Edit the string associated with the current cell. This is identical to "<", "\", or ">" except that the command line starts out containing the old string value or expression associated with the cell. SEE *e* ABOVE.

To enter and edit a cell's number part, use the "=", "+", and *e* commands. To enter and edit a cell's string part, use the "<", "\", ">", and *E* commands. See the sections below on numeric and string expressions for more information.

Note that the descriptions of the "+" and "–" commands below may seem very confusing at first, but once they're understood, they can facilitate the rapid entry of expressions which add and subtract large numbers of cells and sums of ranges of cells, so read them over carefully several times until you understand them.

" Specify a named buffer for the next yank/delete/pull command. Buffers are named with a single character. Buffers "*a*" through "*z*" are general purpose buffers, buffers "*1*" through "*9*" hold the last nine deletions, with buffer "*1*" being the most recent, and buffer "*0*" holds the last cell or range yanked. Buffer "*"*" is the default buffer, which holds the last cell or range that was deleted or yanked.

x Clear the current cell. Deletes the numeric value, label string, and/or numeric or string expression. You can prefix this command with a count of the number of cells on the current row to clear. The current column is used if column recalculation order is set. Cells cleared with this command may be recalled with any of the *pull* commands (see below).

m*x* Mark the current cell. *sc* will prompt for a lowercase letter to be used as a mark specifier. Marked cells may be used as the source for the *copy* command, or as the target of a ' or ' (go to marked cell) command.

c*x* Copy a marked cell to the current cell, adjusting row and column references in its numeric or string expression, if any. *sc* will prompt for the name of the cell to be copied, which may be a lowercase letter specified previously with the *m* command, a digit 1-9 to reference one of the last nine edited cells (0 will reference the last cell in which an edit was begun, regardless of whether the edit was completed or not), or "." to reference the current cell, which, as a special case, is to be used as a source rather than a destination, and is to be copied into a range which includes the current cell. When "." is specified, the current cell is set as the default source range for the range copy (*rc*) command, and then the *copy* command is entered into the command line and *sc* switches to navigate mode. Moving the cell cursor will then highlight the destination range. After the desired range is highlighted, press RETURN to execute the copy.

+ If not in numeric mode, add the current numeric argument (default 1) to the value of the current cell. The current value of the cell must not be an expression. In numeric mode, + switches to insert mode and appends a "+" to the current expression or value, if any, which makes it easy to add to existing data.

In navigate mode, + inserts the current cell address into the line, followed by another +, and *sc*

remains in navigate mode, unless a range is highlighted. If a range is highlighted and the character immediately preceding the cursor is a “+” or “–”, or the cursor is at the beginning of an empty “let” expression, the string “@sum(” will be inserted, followed by the highlighted range, followed by “)+”. If a range is highlighted and the character immediately preceding the cursor is not a “+” or “–”, and the cursor is not at the beginning of an empty “let” expression, the highlighted range will be inserted, followed by “)+”.

- If not in numeric mode, subtract the current numeric argument (default 1) from the value of the current cell. The current value of the cell must not be an expression. In numeric mode, - switches to insert mode and appends a “–” to the current expression or value, if any, which makes it easy to subtract from existing data.

In navigate mode, - inserts the current cell address into the line, followed by another –, and *sc* remains in navigate mode, unless a range is highlighted. If a range if highlighted and the character immediately preceding the cursor is a “+” or “–”, or the cursor is at the beginning of an empty “let” expression, the string “@sum(” will be inserted, followed by the highlighted range, followed by “)–”. If a range is highlighted and the character immediately preceding the cursor is not a “+” or “–”, and the cursor is not at the beginning of an empty “let” expression, the highlighted range will be inserted, followed by “)–”.

RETURN

If you are not editing a cell (top line is empty), pressing RETURN will make *sc* enter insert mode. At this point you may type any valid command or press **ESC** once to edit.

File Commands

G Get a new database from a file. If encryption is enabled, the file is decrypted before it is loaded into the spreadsheet.

P Put the current database into a file. If encryption is enabled, the file is encrypted before it is saved.

ZZ Save the current database into a file if it has been modified, and then quit. This is like the *P* command followed by the *q* command, except that the default filename will be used instead of prompting you for one, and the file will only be saved if it was modified. If there is no default filename, an error message will be displayed, and no action taken.

W Write a listing of the current database into a file in a form that matches its appearance on the screen. This differs from the *Put* command in that its files are intended to be reloaded with *Get*, while *Write* produces a file for people to look at. Hidden rows or columns are not shown when the data is printed.

T Write a listing of the current database to a file, but include delimiters suitable for processing by the *tbl*, *LaTeX*, or *TeX* table processors. The delimiters are controlled by the *tblstyle* option. See *Set* above. The delimiters are a colon (:) for style *0* or *tbl* and an ampersand (&) for style *latex* or *tex*.

With the *Put*, *Write*, and *Table* commands, the optional range argument writes a subset of the spreadsheet to the output file.

With the *Write* and *Table* commands, if you try to write to the last file used with the *Get* or *Put* commands, or the file specified on the command line when *sc* was invoked, you are asked to confirm that the (potentially) dangerous operation is really what you want.

The three output commands, *Put*, *Write*, and *Table*, can pipe their (unencrypted only) output to a program. To use this feature, enter “| program” to the prompt asking for a filename. For example, to redirect the output of the *Write* command to the printer, you might enter “| lpr –p”.

M Merge the database from the named file into the current database. Values and expressions defined in the named file are read into the current spreadsheet overwriting the existing entries at

matching cell locations.

R Run macros. There are two different kinds of macros that can be used with *sc*: simple macros, which are stored in plain text files, and advanced macros, which are executable files, and which can be written in the language of your choice. Advanced macros are only available on systems that support pipes.

Simple macros are interpreted by *sc*'s internal parser, and use the same commands used to enter data and perform other operations (the single key commands are shortcuts which switch to input mode after first entering the beginning of the full command for you). These are also the same commands found in *sc* files created with the Put command. Since *sc* files are saved as ASCII files, it is possible to use them as primitive macro definition files. The *Run* command makes this easier. It's like the *Merge* command, but prints a saved path name as the start of the filename to merge in. The string to use is set with the *Define* command. To write macros, you must be familiar with the file format written by the *Put* command.

Advanced macros use executable files that are started by *sc* as a child process with stdin and stdout redirected back to *sc* for bidirectional communication. Special commands are available for requesting information such as cell contents, formatting information, or the current location of the cell cursor. Commands are written to stdout, and responses are read from stdin. To use advanced macros, the filename must be preceded by a | (the pipe symbol), and the file must be executable. If the pathname set with the *Define* command begins with a |, all files in that path will be executed as advanced macros. It is also possible to include a filename as part of the path when using advanced macros, which allows you to put multiple macros in a single file, and use the *Run* command to add command line arguments or options to determine which macro should be run. Advanced macros are relatively new, and documentation is still incomplete. This feature will probably be enhanced in future releases.

A Specify a macro to be automatically run whenever the current sheet is reloaded from a file.

D Define a path for the *Run* command to use (see above).

All file operations take a filename as the first argument to the prompt on the top line. The prompt supplies a " to aid in typing in the filename. The filename can also be obtained from a cell's label string or string expression. In this case, delete the leading " with the backspace key and enter a cell name such as *a22* instead. If the resulting string starts with "|", the rest of the string is interpreted as a UNIX command, as above.

Row and Column Commands

These are two-letter commands which can be used on either rows or columns. The exceptions are the *f* command, which only works on columns, and therefore doesn't require a second letter, and the *p* command which, in addition to operating on rows or columns, has several other options for merging the data in directly, without opening up a new row or column. There are also a few special cases where pressing the same letter twice will affect only the current cell instead of a row or column (except for *ZZ*, which is a special case all its own).

In all of the remaining cases, the second letter of the command will be either *r* or *c*, depending on whether the operation should be performed on rows or columns, respectively (additional options for the *p* command and the double letter cases are listed below). A small menu lists the choices for the second letter when you type the first letter of one of these commands.

Alternatively, you may define a range of rows or columns by moving the cell cursor, either a cell at a time, or by pages (roughly 1/2 screen, unless the *pagesize* option has been set), but this only works for the *d*, *y*, and *Z* commands. Vertical cursor movement will begin highlighting rows, and horizontal movement will highlight columns. Pressing the RETURN key will then perform the chosen operation on the specified rows/columns.

Commands which copy cells also modify the row and column references in affected cell expressions. The references may be frozen by using the *@fixed* operator or using the *$* character in the reference to the cell (see below). Commands which create new rows or columns will include all newly created cells in the same ranges (named, framed, color, or those used in expressions) as their counterparts in the current row or column. This can sometimes be a significant factor when deciding whether to use *ir*/*ic* or *or*/*oc*.

ir, ic Insert a new row (column) by moving the row (column) containing the cell cursor, and all following rows (columns), down (right) one row (column). The new row (column) is empty. Inserting rows while the cell cursor is in a framed range will only effect rows in that range, leaving all rows to the left and right untouched.

or, oc Open a new row (column). These commands work like the *ir* and *ic* commands, except that the new row (column) will be inserted *after* the current row (column) instead of before it.

ar, ac Append a new row (column) immediately following the current row (column). It is initialized as a copy of the current one. Appending rows while the cell cursor is in a framed range will only effect rows in that range, leaving all rows to the left and right untouched.

dr, dc, dd
Delete the current row (column). *dd* deletes the current cell (i.e., it is a synonym for *x*). Deleting rows while the cell cursor is in a framed range will only effect rows in that range, leaving all rows to the left and right untouched.

yr, yc, yy
Yank a copy of the current row (column) into the delete buffer without actually deleting it. *yy* yanks the current cell (similar to *x*, but without actually deleting the contents of the cell). Yanking rows while the cell cursor is in a framed range will only copy the portion of each row contained in that range, while ignoring everything outside the range.

pr, pc, pp, pm, px, pt, pC, p.
Pull deleted rows/columns/cells back into the spreadsheet. The last set of cells that was deleted or yanked is put back into the spreadsheet at the current location. *pr* inserts enough rows to hold the data. *pc* inserts enough columns to hold the data. *pp* (paste) does not insert rows or columns; it overwrites the cells beginning at the current cell cursor location. *pm* (merge) merges the cells in at the current cell cursor location, but does not erase the destination range first like *pp*. The difference between *pp* and *pm* is similar to the difference between the *Get* and *Merge* commands. *pf* (format) works like *pm* except that only cell formatting information is merged in, leaving the actual data untouched. This makes it easy to copy cell formats from one part of the spreadsheet to another, such as when expanding an existing spreadsheet file. *px* (exchange) copies the contents of the delete buffer into the range beginning at the current cell cursor location, while simultaneously copying the contents of this range back into the delete buffer, replacing its current contents. *pt* (transpose) overwrites the cells beginning at the current cell cursor location like *pp*, but transposes rows for columns and vice versa. *pC* (copy) works like *pp*, except that all cell references are adjusted in the same way that they are for the *copy* command. *p.* is the same as *pC*, except that it switches to navigate mode and allows you to define the destination range to be used. This works like the *copy* command in that if the source range (the contents of the delete buffer) is a single row, column, or cell, multiple copies may be made.

vr, vc, vv
Remove expressions from the affected rows (columns), leaving only the values which were in the cells before the command was executed. When used in a framed range, *vr* only affects the portion of the the row inside the range, leaving the rest of the row unchanged. *vv* only affects the contents of the current cell.

Zr, Zc, ZZ
Hide (“zap”) the current row (column). This keeps a row (column) from being displayed but

keeps it in the data base. The status of the rows and columns is saved with the data base so hidden rows and columns will still be hidden when you reload the spreadsheet. Hidden rows or columns are not printed by the *W* command. The *ZZ* command is a special case. It does not hide anything. Instead, the file will be saved, if modified, and *sc* will exit. See *ZZ* above, under *File Commands.*

sr, sc Show hidden rows (columns). Enter a range of rows (columns) to be revealed. The default is the first range of rows (columns) currently hidden. This command ignores the repeat count, if any.

f Set the output format to be used for printing the numeric values in each cell in the current column. This command has only a column version (no second letter). You may change the column width by pressing the *h*, <, or cursor left key to reduce it, or the *l*, >, or cursor right key to increase it. Likewise, you may change the precision (the number of digits to follow decimal points) by pressing the *j*, –, or cursor down key to reduce it, or the *k*, +, or cursor up key to increase it. You may also change the format type for the column by pressing any digit. If the *f* command is preceded by a numeric argument, that argument will determine how many columns should be changed, beginning with the current column, and in the case of incrementing or decrementing the width or precision of the columns, each column will be incremented or decremented separately, regardless of its initial values. Several formatting operations may be performed in sequence. To leave the formatting command, simply press **ESC**, *^G*, *q*, or RETURN .

Alternatively, you may press **SPACE** to get the *format* command in the top line and enter all three values directly. In order, these are: the total width in characters of the column, the precision, and the format type. Format types are 0 for fixed point, 1 for scientific notation, 2 for engineering notation, 3 for dates with a two digit year, and 4 for dates with a four digit year. Values are rounded off to the least significant digit displayed. The total column width affects displays of strings as well as numbers. A preceding count can be used to affect more than one column.

You can also create your own format types by pressing = after the *f* command, followed by any digit (see the *F* command above under *Cell Entry and Editing Commands* for a description of how to build a format string). Format numbers 0 through 4 will supersede the built-in format types, while numbers 5 through 9 will supplement them. User defined format types may be used in the same way as the built-in types. For example, the command

format 5 = "#,0.& ;(#,0.&)"

will define a currency format which may then be assigned to column C, for example, with the command

format C 10 2 5

@myrow, @mycol

Are functions that return the row or column of the current cell respectively. ex: The cell directly above a cell in the D column could then be accessed by @nval("d",@myrow-1). NOTE: @myrow and @mycol can't be used in specifying ranges.

@lastrow, @lastcol

These return the last row and column of the spreadsheet, respectively. They are useful for macros designed to default to the whole spreadsheet.

Range Commands

Range operations affect a rectangular region on the screen defined by the upper left and lower right cells in the region. All of the commands in this class begin with "r"; the second letter of the command indicates which command. A small menu lists the choices for the second letter when you type "r". *sc* prompts for needed parameters for each command. Phrases surrounded by square brackets in the prompt are informational only and may be erased with the backspace key.

Prompts requesting variable names may be satisfied with either an explicit variable name, such as *A10*, or with a variable name previously defined in a *rd* command (see below). Range name prompts require either an explicit range such as *A10:B20*, or a range name previously defined with a *rd* command. A default range shown in the second line is used if you omit the range from the command or press the TAB key (see below). The default range can be changed by moving the cell cursor via the control commands (*^P* or *^N*) or the arrow keys. The cells in the default range are highlighted (using the terminal's standout mode, if available).

rx Clear a range. Cells cleared with this command will be saved in the delete buffer, and may be recalled with any of the *pull* commands.

ry Yank a range. Like *rx*, cells yanked with this command will be saved in the delete buffer, and may be recalled with any of the *pull* commands. This command differs from *rx*, however, in that the original cells will not be cleared. Although this command may be used to copy a range of cells, it treats all references as fixed. Use *rc* if you want references to be relative to the cell which contains them unless specified otherwise, either with the *@fixed* operator or using the *$* character in the reference to the cell.

rc Copy a source range to a destination range. The source and destination may be different sizes. The result is always one or more full copies of the source. Copying a row to a row yields a row. Copying a column to a column yields a column. Copying a range to anything yields a range. Copying a row to a column or a column to a row yields a range with as many copies of the source as there are cells in the destination. This command can be used to duplicate a cell through an arbitrary range by making the source a single cell range such as *b20:b20*.

If the source range is omitted (second argument), the source range from the last *copy* command will be used, unless a range is currently highlighted, in which case the highlighted range will be copied instead. If both the source range and destination range are omitted, the current cell will be used as the destination, unless a range is currently highlighted, in which case the highlighted range will serve as the destination, and the source range from the last *copy* command will be copied into that destination.

rm Move a source range to a destination range. This differs from deleting a range with *rx* and pulling it back in with *pm* in that any expressions that reference a cell in the range to be moved will reference the cell at its new address after the move. Unlike the *rc* command, the destination of a move is a single cell, which will be the upper lefthand corner of the source range after the move.

rv Values only. This command removes the expressions from a range of cells, leaving just the values of the expressions.

rs Sort a range. The rows in the specified range will be sorted according to criteria given in the form of a string of characters. This string, enclosed in double quotes, may comprise a single criterion or multiple criteria in decreasing order of precedence. Each criterion has three parts, all of which are mandatory. The first part is a single character, which must be either + or –, which specifies whether the sort should be done in ascending or descending order, respectively. The second part, which is also a single character, must be either *#* or *$*, and is used to specify whether the sort should be based on the numeric portion or the string portion, respectively, of the cells being used for the comparison. The third part may be either one or two characters, and must be alphabetic (case insensitive), and specifies the column to be used when making the comparisons. This column must be in the range being sorted. Any number of criteria may be concatenated, and will be used in the order specified. If no criteria are specified, the default behavior is to sort in ascending order, first by string and then by number, using the leftmost column of the range being sorted. This is equivalent to specifying the sort criteria to be " +$a+#a ", where both *a*'s are replaced by the name of the leftmost column of the range being sorted.

rf Fill a range with constant values starting with a given value and increasing by a given incre-

ment. Each row is filled before moving on to the next row if row order recalculation is set. Column order fills each column in the range before moving on to the next column. The start and increment numbers may be positive or negative. To fill all cells with the same value, give an increment of zero.

r{ Left justify all strings in the specified range.

r} Right justify all strings in the specified range.

r| Center all strings in the specified range.

rd Use this command to assign a symbolic name to a single cell or a rectangular range of cells on the screen. The parameters are the name, surrounded by "", and either a single cell name such as *A10* or a range such as *a1:b20*. Names defined in this fashion are used by the program in future prompts, may be entered in response to prompts requesting a cell or range name, and are saved when the spreadsheet is saved with the *Put* command. Names defined may be any combination of alphanumeric characters and '_' as long as the name isn't a valid cell address. Thus, *x*, *H2SO4*, and *3rdDay* are all valid names, but *H2* is not.

rl Use this command to lock the current cell or a range of cells, i.e. make them immune to any type of editing. A locked cell can't be changed in any way until it is unlocked.

rU This command is the opposite of the *rl* command and thus unlocks a locked cell and makes it editable.

rS This command shows lists of the currently defined range names, framed ranges, and color definitions and ranges, one after the other. The output of this command will be piped to *less*. If the environment variable PAGER is set, its value is used in place of *less*.

ru Use this command to undefine a previously defined range name.

rF Use this command to assign a value format string (see the "F" cell entry command) to a range of cells.

rr This command is used for creating, modifying, and deleting framed ranges. A framed range, is one which has a number of rows or columns specified at the top, bottom, left, and/or right (the frame) which must remain onscreen whenever the cell cursor is within that range. In other words, a frame consists of an outer range and an inner range, where the inner range is allowed to scroll within the outer range. Once a frame is defined, the inner range may be resized, but the outer range remains fixed unless rows or columns are added or deleted within the range.

When this command is invoked, you will be prompted for the type of frame-related action you would like to perform. You may select an option from the list by typing its first letter.

The options are *top*, *bottom*, *left*, *right*, *all*, and *unframe.* If you choose *top*, *bottom*, *left*, or *right*, you will be prompted for a range and number of rows/columns. The range may be omitted if the cell cursor is in a previously defined framed range, in which case that range's outer range will be used instead. The number of rows/columns will set or adjust the width of the corresponding side of the frame. If all of these widths are set to zero, the frame will be undefined (same as the unframe command).

If you choose *all*, you will be prompted for an outer range and an inner range, in which case the inner range will scroll within the outer range, and any rows or columns outside of the inner range, but inside the outer range will be part of the "frame" that is to remain onscreen. The outer range may be omitted if the cell cursor is in a previously defined framed range, in which case the previously defined outer range will be used. However, if a single range is specified on the command line, while another range wholly contained within this range is highlighted, the specified range will be used as the outer range, and the highlighted range will be used as the inner range. If no range is specified on the command line, but a range is highlighted, and the

highlighted range is wholly contained within a previously defined framed range, the highlighted range will be used as the inner range, and the previously defined outer range will be used as the outer range.

If you choose *unframe*, you will be prompted for a range, and if the range is found in the list of frames, the frame will be deleted, and the framing will no longer be active (the specified range must be the outer range of the previously defined frame to be deleted). The range may be omitted if the cell cursor is in a previously defined framed range, in which case that range will be used by default.

Framed ranges may not be nested or overlapping. If you try to define a range that contains any cells in common with a previously defined framed range, an error message will be issued, and the frame will not be created.

rC This command defines a color range, and specifies a foreground/background pair to be used for that range. See "Color Commands" below for more information.

Note Commands

A note is a cell or range of cells that can be jumped to quickly from another cell by creating a special link in that cell. The note may contain text explaining the contents of the cell containing the link, similar to a footnote, or it may simply be another part of the spreadsheet that is related to the cell in some way. When you press the 'n' key, you will get a short prompt asking you whether you want to add or delete a note, or to "show" (by highlighting) which cells on the screen have attached notes.

If a cell with an attached note contains numeric data, it will be preceded with an "*". If color is available and turned on, the "*" will be displayed with color 4. Also, the note address will be displayed in curly braces on the top line, preceded by an "*", when the cell is current (e.g. *{*AC30:AE43}* or *{*note1}* for a named range). You may also use the **s* (Note/Show) command to highlight all cells on the current screen with attached notes.

***a** Add a note. This will bring up the addnote command in the top line, followed by the target address of the cell where you want the note added. You must then enter the cell or range where the note resides to add the note. If you omit the note address or range, the currently highlighted range, if any, will be used. Otherwise, the current cell will be used (you would, of course, want to move away from the cell in which the addnote command was invoked in the latter case).

***d** Delete a note. If there is a note attached to the current cell, the link will be removed (deleted). The note itself will not be removed from the spreadsheet. If it is no longer needed, it must be deleted in a separate step.

***s** Show all notes on the current screen. If there are any cells on the visible portion of the spreadsheet which contain attached notes, they will be highlighted until the next screen change, no matter how minor. Simply moving to a new cell will be enough to turn off the highlighting.

****** Jump to a note. If there is a note attached to the current cell, you will be immediately transported to that cell. You may return from the note to where you were by pressing ' twice.

Color Commands

Color may be enabled by setting the color option ("set color"), or by toggling it with ^TC (control-T followed by an uppercase C). If color is enabled, you may define up to eight color pairs, each consisting of a foreground color and a background color. Each of these colors may be defined by an expression which is evaluated at the same time the rest of the spreadsheet is evaluated. Color expressions may be simple, specifying only a foreground color and a background color, or they may be arbitrarily complex, causing the colors to change based upon other data in the spreadsheet, for example. Color ranges may then be defined using the *rC* command, with a color number (1-8) assigned to the range (see below).

Some of the color numbers may have special meaning under certain circumstances, but may also be used explicitly at the same time. For example, color 1 is the default color pair if color is enabled but no color has been defined for a given cell. It is also the color used for the column and row labels and the top two lines of the display, which are used for prompts, input, error messages, etc. Color 2, while not explicitly used for all negative numbers, will be used for negative numbers in cells which have no other color defined when colorneg is turned on ("set colorneg" or ^TN). This is because colorneg causes all cells with negative numbers to have their color number incremented by one (cycling back to color 1 if the cell is defined as using color 8). Color 3 is used for all cells with errors (ERROR or INVALID), if colorerr is set ("set colorerr" or ^TE), regardless of which color they have been defined to use, or whether they have been defined to use any color at all. Color 4 is used to highlight the "*" which signifies that a cell has a note attached.

If two color ranges are nested or overlapping, any cell that is common to both will be displayed using the color of the most recently defined color range. You can list all color definitions and color ranges with the rS (show) command (see below).

C This command first prompts you for the color number you would like to define (or redefine). After selecting a number (1-8), you may enter an expression which defines the foreground and background colors. If the chosen color has previously been defined, the old definition will be presented for editing. The syntax of the color command is:

color *number = expression*

where *number* is the number of the color pair you want to define, and *expression* is the definition. If the expression is missing, the specified color number will be unset (it will revert to its default start-up colors). Unlike setting it explicitly to its original value, this will not cause the expression to be written to the file when saved. See below for an explanation of the format of a color expression.

rC This command defines a color range, and specifies a foreground/background pair to be used for that range. Although this command also uses the *color* command, the syntax is different from that used for defining a color pair. This syntax is:

color *range number*

rS This command shows lists of the currently defined range names, framed ranges, and color definitions and ranges, one after the other. The output of this command will be piped to *less*. If the environment variable PAGER is set, its value is used in place of *less*.

Color expressions are exactly like any other numeric expression, and may contain any function or operator that is valid in any other numeric expression. There are, however special functions designed specifically for defining colors. These functions are:

@black
@red
@green
@yellow
@blue
@magenta
@cyan
@white

Although these function names are intended to reflect the color they produce, and use the same names as the curses colors, @yellow may appear as brown on many displays, especially those based on the VGA standard.

In addition to special functions for specifying colors, there is also a special operator for combining two such colors into a single number which specifies both a foreground and a background color. This operator is the semicolon (;). For example, the command

color 1 = @white;@green

will set the foreground color to white and the background color to green for any cell or range of cells defined to use color 1, or which have no color defined. If the semicolon operator is not used, and only one color is specified, that color will be used for the foreground, and the background will default to black.

Although the above example is the easiest way to specify foreground and background colors, and will probably meet most people's needs, *sc* allows much more power and flexibility, should the need arise, due to the fact that any color can be specified by an expression. For example,

color 5 = B23<E75?(@black;@cyan):(@white;@magenta)

will cause all cells defined with color 5 to be displayed as black text on a cyan background if the numeric value in cell B23 is less than the numeric value in cell E75; otherwise, they will be displayed as white text on a magenta background. If you prefer to have the foreground and background colors dependent on different criteria, you could do something like this:

color 5 = (B23<E75?@white:@cyan);(D5%2?@red:@blue)

This will cause the text color for color 5 to be either white or cyan, depending on the numeric values in cells B23 and E75, as in the previous example, and the background color to be either red or blue, depending on whether the numeric value in cell D5 is odd or even.

Note that although a color expression may contain any function which is valid in any other numeric expression, the @myrow and @mycol functions will always evaluate to 0. This is because a color expression is not tied to any particular cell, but is instead evaluated once, and the result used for all cells defined to use that color.

Also note that if a color expression results in an error, the color will default to black text on a black background. If color 1 results in an error, color will be disabled so that you can see the input line to correct the error, after which color will need to be reenabled manually.

Default colors are in effect for all colors until defined otherwise. These default colors are as follows:

color 1 = @white;@blue
color 2 = @red;@blue
color 3 = @white;@red
color 4 = @black;@yellow
color 5 = @black;@cyan
color 6 = @red;@cyan
color 7 = @white;@black
color 8 = @red;@black

Miscellaneous Commands

Q
q
^C Exit from *sc*. If you made any changes since the last *Get* or *Put*, *sc* asks about saving your data before exiting.

^G
ESC Abort entry of the current command.

? Enter an interactive help facility. Lets you look up brief summaries of the main features of the program. The help facility is structured like this manual page so it is easy to find more information on a particular topic, although it may not be completely up-to-date.

! Shell escape. *sc* prompts for a shell command to run. End the command line with the RETURN key. If the environment variable SHELL is defined, that shell is run. If not, /bin/sh is used. Giving a null command line starts the shell in interactive mode. A second "!" repeats the previous command.

~ Abbreviations. You may set abbreviations to speed up the entry of repetitive data. Abbrevia-

tions work much like abbreviations in vi, except that when defining an abbreviation, both the abbreviation and the expanded text must be contained within quotes, separated by a single space. If more than one space separates the abbreviation from the expanded text, it will be included as part of the expanded text.

There are three types of abbreviations available in sc. In the first type, all characters must be either alphanumeric or "_". In the second type, the last character must be alphanumeric or "_", but all other characters must not be alphanumeric or "_". Neither type may contain spaces. The third type of abbreviation is a single character, and must be alphanumeric or "_".

When using abbreviations, the first type must be at the beginning of the line, or must be preceded by any character which is not alphanumeric or "_". The second type must be at the beginning of the line, or must be preceded either by an alphanumeric character, "_", or a space. Single character abbreviations must be at the beginning of the line or preceded by a space.

Abbreviations will be automatically expanded as soon as the space bar or return key is pressed, or when pressing the ESC key at the end of the abbreviation to switch to edit mode. You can also force an abbreviation to be expanded by following it with a *^]*, which won't be inserted into the line. If you don't want an abbreviation to be expanded, you must either press *^V* twice or switch to edit mode and back again somewhere within the abbreviation (pressing *^V* twice also has the effect of switching to navigate mode and back again).

If the string in the abbreviation command contains no spaces, the entire string will be looked up in the list of abbreviations, and if found, the definition will be displayed in the form of the original *abbreviation* command used to define it. When looking up an abbreviation in this manner, be sure to disable abbreviation expansion, as described above, or the results may not be what you expect.

If the string is empty, a list of all abbreviations and their corresponding expanded text will be output to your pager. Note that abbreviations are not saved with the file. This allows each user to create his own file of abbreviations and either merge them in or include them in his own .scrc file, rather than force all users who access a file to use the same list of abbreviations.

^L Redraw the screen.

^R Redraw the screen with special highlighting of cells to be filled in. This is useful for finding values you need to provide or update in a form with which you aren't familiar or of which you have forgotten the details.

It's also useful for checking a form you are creating. All cells which contain constant numeric values (not the result of a numeric expression) are highlighted temporarily, until the next screen change, however minor. To avoid ambiguity, the current range (if any) and current cell are not highlighted.

^X This command is similar to *^R*, but highlights cells which have expressions. It also displays the expressions in the highlighted cells as left-flushed strings, instead of the numeric values and/or label strings of those cells. This command makes it easier to check expressions, at least when they fit in their cells or the following cell(s) are blank so the expressions can slop over (like label strings). In the latter case, the slop over is not cleared on the next screen update, so you may want to type *^L* after the *^X* in order to clean up the screen.

@ Recalculates the spreadsheet.

Variable Names

Normally, a variable name is just the name of a cell, such as *K20*. The value is the numeric or string value of the cell, according to context.

When a cell's expression (formula) is copied to another location via *copy* or *range-copy*, variable references are by default offset by the amount the formula moved. This allows the new formula to work on new data. If cell references are not to change, you can either use the *@fixed* operator (see below), or one of the following variations on the cell name.

K20 References cell *K20*; the reference changes when the formula is copied.

$***K***$*20* Always refers to cell *K20*; the reference stays fixed when the formula is copied.

$*K20* Keeps the column fixed at column K; the row is free to vary.

*K***$***20* Similarly, this fixes the row and allows the column to vary.

These conventions also hold on defined ranges. Range references vary when formulas containing them are copied. If the range is defined with fixed variable references, the references do not change.

@fixed To make a variable not change automatically when a cell moves, put the word *@fixed* in front of the reference, for example: B1 * @fixed C3.

Numeric Expressions

Numeric expressions used with the "=" and *e* commands have a fairly conventional syntax. Terms may be constants, variable names, parenthesized expressions, and negated terms. Ranges may be operated upon with range functions such as sum (*@sum*()) and average (*@avg*()). Terms may be combined using binary operators.

–*e* Negation.

*e***+***e* Addition.

*e***–***e* Subtraction.

*e******e* Multiplication.

*e***/***e* Division.

*e1***%***e2* e1 mod e2.

*e***^***e* Exponentiation.

*e***<***e*
*e***<=***e*
*e***=***e*
*e***!=***e*
*e***>=***e*
*e***>***e* Relationals: true (1) if and only if the indicated relation holds, else false (0). Note that "<=", "!=", and ">=" are converted to their "!()" equivalents.

~*e* Boolean operator NOT.

*e***&***e* Boolean operator AND.

*e***|***e* Boolean operator OR.

@if(*e***,***e***,***e***)**
*e***?***e***:***e* Conditional: If the first expression is true then the value of the second is returned, otherwise the value of the third.

Operator precedence from highest to lowest is:

–, ~, !
^
*, /
+, –
<, <=, =, !=, >=, >
&
|

?:

Built-in Range Functions

These functions return numeric values. The @sum, @prod, @avg, @count, @max, @min, and @stddev functions may take an optional second argument which is an expression that is to be evaluated for each cell in the specified range to determine which cells to include in the function. Only those cells for which the expression evaluates to true (non-zero) will be used in calculating the value of the function. Before evaluation for each cell, the expression is first converted as if it was being copied from the cell in the upper left-hand corner of the range into the cell under consideration, with all cell references adjusted accordingly. Because the parts of the expression that should remain fixed during the evaluation of the function may not necessarily be the same as those which should remain fixed during an actual copy operation, the rules for adjusting cell references during a copy operation are slightly different than normal. In particular, these rules differ in two different ways.

The first difference is that the *@fixed* operator is ignored during a copy operation unless it is enclosed in parentheses. This is so that selected cells whose addresses should remain fixed during any given evaluation of a range function can be adjusted relative to the cell containing the range function when copied (the *$* prefix is still honored for these cells when copying). Enclosing the *@fixed* operator in parentheses will have the opposite effect. That is, it will cause cell references to be fixed while copying, while allowing them to be adjusted when the function is being evaluated, subject to any *$* prefixes present. Note that only the *@fixed* operator itself should be enclosed in parentheses for this to work properly.

The second difference is that any references in the expression that refer to cells in the range in the first argument of the range function will have any *$* prefixes ignored, and the references will be treated instead as if they had the same *$* prefixes as the left side of the range argument. For example, if the left side of the range argument (the cell address on the left side of the colon) has a fixed row, but does not have a fixed column, any cell references that refer to cells in that range will also have a fixed row, but will not have a fixed column. This is so that if the range reference moves when copying, references to any cells in that range will also move accordingly.

Note that the test expression will be evaluated once for every cell in the range, which means that excessive use of these functions with the optional test expression, or the use of overly complex test expressions or with very large ranges can greatly slow down the recalculation of a spreadsheet, and may require turning off autocalc for speed, and then manually recalculating with the @ command.

@sum(*r*)
@sum(*r*,*e*) Sum all valid (nonblank) entries in the region whose two corners are defined by the two variable names (e.g. *c5:e14*) or the range name specified. The optional second argument is an expression which can be used to determine which cells in the range to sum (see above).

@prod(*r*)
@prod(*r*,*e*) Multiply together all valid (nonblank) entries in the specified region. The optional second argument is an expression which can be used to determine which cells in the range to multiply (see above).

@avg(*r*)
@avg(*r*,*e*) Average all valid (nonblank) entries in the specified region. The optional second argument is an expression which can be used to determine which cells in the range to average (see above).

@count(*r*)
@count(*r*,*e*) Count all valid (nonblank) entries in the specified region. The optional second argument is an expression which can be used to determine which cells in the range to count (see above).

@max(*r***)**
@max(*r***,***e***)** Return the maximum value in the specified region. The optional second argument is an expression which can be used to exclude specific cells in the range when determining this maximum value (see above). See also the multi argument version of *@max* below.

@min(*r***)**
@min(*r***,***e***)** Return the minimum value in the specified region. The optional second argument is an expression which can be used to exclude specific cells in the range when determining this minimum value (see above). See also the multi argument version of *@min* below.

@stddev(*r***)**
@stddev(*r***,***e***)** Return the sample standard deviation of the cells in the specified region. The optional second argument is an expression which can be used to exclude specific cells in the range when calculating the standard deviation (see above).

@rows(*r***)** Return the number of rows in the specified range.

@cols(*r***)** Return the number of columns in the specified range.

@lookup(*e***,***r***)**
@lookup(*r***,***e***)**
@lookup(*se***,***r***)**
@lookup(*r***,***se***)** Evaluates the expression then searches through the range *r* for a matching value. The range should be either a single row or a single column. The expression can be either a string expression or a numeric expression. If it is a numeric expression, the range is searched for the the last value less than or equal to *e*. If the expression is a string expression, the string portions of the cells in the range are searched for an exact string match. The value returned is the numeric value from the next row and the same column as the match, if the range was a single row, or the value from the next column and the same row as the match if the range was a single column.

@hlookup(*e***,***r***,***n***)**
@hlookup(*r***,***e***,***n***)**
@hlookup(*se***,***r***,***n***)**
@hlookup(*r***,***se***,***n***)** Evaluates the expression then searches through the first row in the range *r* for a matching value. The expression can be either a string expression or a numeric expression. If it is a numeric expression, the row is searched for the the last value less than or equal to *e*. If the expression is a string expression, the string portions of the cells in the row are searched for an exact string match. The value returned is the numeric value from the same column *n* rows below the match.

@vlookup(*e***,***r***,***n***)**
@vlookup(*r***,***e***,***n***)**
@vlookup(*se***,***r***,***n***)**
@vlookup(*r***,***se***,***n***)** Evaluates the expression then searches through the first column in the range *r* for a matching value. The expression can be either a string expression or a numeric expression. If it is a numeric expression, the column is searched for the the last value less than or equal to *e*. If the expression is a string expression, the string portions of the cells in the column are searched for an exact string match. The value returned is the numeric value from the same row *n* columns to the right of the match.

@index(*e1***,***r***)**
@index(*r***,***e1***)**
@index(*r***,***e1***,***e2***)** Use the values of expressions *e1* and (optionally) *e2* to index into the range *r*.

The numeric value at that position is returned. With two arguments, the range should be either a single row or a single column. An expression with the value 1 selects the first item in the range, 2 selects the second item, etc. With three arguments, the range must come first, and the second and third arguments will then be interpreted as row and column, respectively, for indexing into a two-dimensional table.

@stindex(*e1***,***r***)**
@stindex(*r***,***e1***)**
@stindex(*r***,***e1***,***e2***)** Use the values of expressions *e1* and (optionally) *e2* to index into the range *r*. The string value at that position is returned. With two arguments, the range should be either a single row or a single column. An expression with the value 1 selects the first item in the range, 2 selects the second item, etc. With three arguments, the range must come first, and the second and third arguments will then be interpreted as row and column, respectively, for indexing into a two-dimensional table.

Built-in Numeric Functions

All of these functions operate on floating point numbers (doubles) and return numeric values. Most of them are standard system functions more fully described in *math*(3). The trig functions operate with angles in radians.

@sqrt(*e***)** Return the square root of *e*.

@exp(*e***)** Return the exponential function of *e*.

@ln(*e***)** Return the natural logarithm of *e*.

@log(*e***)** Return the base 10 logarithm of *e*.

@floor(*e***)** Return the largest integer not greater than *e*.

@ceil(*e***)** Return the smallest integer not less than *e*.

@rnd(*e***)** Round *e* to the nearest integer. default: *.5 will be rounded up to the next integer; doing a 'set rndtoeven' will cause it to be rounded to the closest even number instead (aka banker's round). Round-to-even has advantages over the default rounding for some applications. For example, if X+Y is an integer, then X+Y = rnd(X)+rnd(Y) with round-to-even, but not always with the defaulting rounding method. This could be an advantage, for example, when trying to split an odd amount of money evenly between two people (it would determine who gets the extra penny).

@round(*e***,***n***)** Round *e* to *n* decimal places. *n* may be positive to round off the right side of the decimal or negative to round off the left side. See @rnd(e) above for rounding types.

@abs(*e***)**
@fabs(*e***)** Return the absolute value of *e*.

@pow(*e1***,***e2***)** Return *e1* raised to the power of *e2*.

@hypot(*e1***,***e2***)** Return sqrt(e1∗e1+e2∗e2), taking precautions against unwarranted overflows.

@pi A constant quite close to pi.

@dtr(*e***)** Convert *e* in degrees to radians.

@rtd(*e***)** Convert *e* in radians to degrees.

@sin(*e***)**
@cos(*e***)**
@tan(*e***)** Return trigonometric functions of radian arguments. The magnitude of the arguments are not checked to assure meaningful results.

@asin(*e*) Return the arc sine of *e* in the range –pi/2 to pi/2.

@acos(*e*) Return the arc cosine of *e* in the range 0 to pi.

@atan(*e*) Return the arc tangent of *e* in the range –pi/2 to pi/2.

@atan2(*e1*,*e2*) Returns the arc tangent of *e1*/*e2* in the range –pi to pi.

@max(*e1*,*e2*,...) Return the maximum of the values of the expressions. Two or more expressions may be specified. See also the range version of *@max* above.

@min(*e1*,*e2*,...) Return the minimum of the values of the expressions. Two or more expressions may be specified. See also the range version of *@min* above.

@ston(*se*) Convert string expression *se* to a numeric value.

@eqs(*se1*,*se2*) Return 1 if string expression *se1* has the same value as string expression *se2*, 0 otherwise.

@nval(*se*,*e*) Return the numeric value of a cell selected by name. String expression *se* must evaluate to a column name ("A"-"AE") and *e* must evaluate to a row number (0-199). If *se* or *e* is out of bounds, or the cell has no numeric value, the result is 0. You can use this for simple table lookups. Be sure the table doesn't move unexpectedly! See also *@sval*() below.

@err Force an error. This will force the expression which contains it to result in an error.

String Expressions

String expressions are made up of constant strings (characters surrounded by double quotation marks), variables (cell names, which refer to the cells's label strings or expressions), and string functions. Note that string expressions are only allowed when entering a cell's label string, not its numeric part. Also note that string expression results may be left or right flushed or centered, according to the type of the cell's string label.

Concatenate strings. For example, the string expression

A0 #

"zy dog"

displays the string "the lazy dog" in the cell if the value of *A0*'s string is "the la".

Built-in String Functions

@filename(*e*) Return the current default filename, as specified when the file was first loaded or created, or during the last save, with the *Put* command. If *e* is 0, only the actual filename will be returned, with any path removed. If non-zero, the full path specified on the command line or in the last *Get* or *Put* command will be returned. If the path begins with "~", it will be expanded to the appropriate users home directory.

@substr(*se*,*e1*,*e2*) Extract and return from string expression *se* the substring indexed by character number *e1* through character number *e2* (defaults to the size of *se* if beyond the end of it). If *e1* is less than 1 or greater than *e2*, the result is the null string. For example,

@substr

("Nice jacket", 4, 8)

returns the string "e jac".

@fmt(*se*,*e*) Convert a number to a string. The argument *se* must be a valid *printf*(3) format string. *e* is converted according to the standard rules. For example, the expression

@fmt

("**%6.3f**", 10.5)

yields the string "**10.500**". *e* is a double, so applicable formats are e, E, f, g, and G. Try "%g" as a starting point.

@sval(*se***,***e***)** Return the string value of a cell selected by name. String expression *se* must evaluate to a column name ("A"-"AE") and *e* must evaluate to a row number (0-199). If *se* or *e* is out of bounds, or the cell has no string value, the result is the null string. You can use this for simple table lookups. Be sure the table doesn't move unexpectedly!

@upper(*se***)**
@lower(*se***)** will case the string expression to upper or lower.

@capital(*se***)** will convert the first letter of words in a string into upper case and other letters to lower case (the latter if all letters of the string are upper case).

@ext(*se***,***e***)** Call an external function (program or script). The purpose is to allow arbitrary functions on values, e.g. table lookups and interpolations. String expression *se* is a command or command line to call with *popen*(3). The value of *e* is converted to a string and appended to the command line as an argument. The result of *@ext*() is a string: the first line printed to standard output by the command. The command should emit exactly one output line. Additional output, or output to standard error, messes up the screen. *@ext*() returns a null string and prints an appropriate warning if external functions are disabled, *se* is null, or the attempt to run the command fails.

External functions can be slow to run, and if enabled are called at each screen update, so they are disabled by default. You can enable them with *^T* when you really want them called.

A simple example:

@ext

("echo", a1)

You can use *@ston*() to convert the *@ext*() result back to a number. For example:

@ston (@ext

("form.sc.ext", a9 + b9))

Note that you can build a command line (including more argument values) from a string expression with concatenation. You can also "hide" the second argument by ending the command line (first argument) with " #" (shell comment).

@coltoa(*e***)** Returns a string name for a column from the numeric argument. For example:

@coltoa(@mycc

@nval(coltoa(@mycol–1),

@myrow+1)

Built-in Financial Functions

Financial functions compute the mortgage (or loan) payment, future value, and the present value functions. Each accepts three arguments, an amount, a rate of interest (per period), and the number of periods. These functions are the same as those commonly found in other spreadsheets and financial calculators

@pmt(*e1***,***e2***,***e3***)** *@pmt*(60000,.01,360) computes the monthly payments for a $60000 mortgage at 12% annual interest (.01 per month) for 30 years (360 months).

@fv(*e1***,***e2***,***e3***)** *@fv*(100,.005,36) computes the future value for 36 monthly payments of $100 at 6% interest (.005 per month). It answers the question: "How much will I have in

36 months if I deposit $100 per month in a savings account paying 6% interest compounded monthly?"

@pv(*e1*,*e2*,*e3*) *@pv*(1000,.015,36) computes the present value of an ordinary annuity of 36 monthly payments of $1000 at 18% annual interest. It answers the question: "How much can I borrow at 18% for 3 years if I pay $1000 per month?"

Built-in Date and Time Functions

Time for *sc* follows the system standard: the number of seconds since the beginning of 1970. All date and time functions except *@date*() return numbers, not strings.

@now Return the current time encoded as the number of seconds since the beginning of the epoch (December 31, 1969, midnight, GMT).

@dts(*e1*,*e2*,*e3*) Convert a date to the number of seconds from the epoch to the first second of the specified date, local time. Dates may be specified in either (m,d,y) or (y,m,d) format, although the latter is preferred, since it's more universally recognized (m,d,y is only used in America). If e2 > 12 or e3 > 31, then (m,d,y) is assumed. Otherwise, (y,m,d) is assumed. For example, *@date(@dts(1976,12,14))* yields

Tue

Dec 14 00:00:00 1976

The month should range from 1 to 12; the day should range from 1 to the number of days in the specified month; and the year should include the century (e.g. 1999 instead of 99). Any date capable of being handled by the system is valid, typically 14 Dec 1901 to 18 Jan 2038 on a system that uses a 32 bit time_t. Invalid dates or dates outside of this range will return ERROR. For rapid entry of dates using only the numeric keypad, *sc* provides the alternate syntax *y.m.d* or *m.d.y*, which is automatically converted to the @dts(...) format above. The year, month, and day must be entered numerically in the alternate syntax; formulas are not allowed.

@tts(*e1*,*e2*,*e3*) *@tts*(8,20,45) converts the time 8:40:45 to the number of seconds since midnight, the night before. The hour should range from 0 to 23; the minutes and seconds should range from 0 to 59.

The following functions take the time in seconds (e.g. from *@now*) as an argument and return the specified value. The functions all convert from GMT to local time.

@date(*e*)
@date(*e*,*se*) Convert the time in seconds to a date string. With a single numeric argument, the date will be 24 characters long in the following form:

Sun

Sep 16 01:03:52 1973

Note that you can extract parts of this fixed-format string with *@substr*(). A format string compatible with the strftime() function may optionally be given as a second argument to override the default format. See the strftime(3) man page for details.

@year(*e*) Return the year. Valid years begin with 1970, although many systems will return years prior to 1970 if *e* is negative. The last legal year is system dependent.

@month(*e*) Return the month, encoded as 1 (January) to 12 (December).

@day(*e*) Return the day of the month, encoded as 1 to 31.

@hour(*e*) Return the number of hours since midnight, encoded as 0 to 23.

@minute(*e*) Return the number of minutes since the last full hour, encoded as 0 to 59.

@second(*e*) Return the number of seconds since the last full minute, encoded as 0 to 59.

Spreadsheet Update

Re-evaluation of spreadsheet expressions is done by row or by column depending on the selected calculation order. Evaluation is repeated up to *iterations* times for each update if necessary, so forward references usually work as expected. See *set* above. If stability is not reached after ten iterations, a warning is printed. This is usually due to a long series of forward references, or to unstable cyclic references (for example, set *A0*'s expression to "A0+1").

@numiter Returns the number of iterations performed so far.

Programmable Function Keys

Function keys can be used in *sc* if your terminal supports them, and they are programmable. To program the function keys, you use the *fkey* command. This command may be used in a *.scrc* file or a macro file, or it may be entered directly into *sc*'s command line. Defined function keys will be saved with the file. There is no shortcut, as there is with most commands, so the full command must be typed in. Pressing enter when not editing a line will start you off with a blank line for this purpose. The format of the *fkey* command is:

fkey *n = "command"*

where *n* is the function key number (n = 1 for F1, n = 2 for F2, etc.), and *command* is the command to be run. For example,

fkey 2 = "merge \"|~/scmacros/macro1\""

will run the macro called *macro1* located in a subdirectory of your home directory called *scmacros* when the F2 key is pressed. Note that embedded quotes must be escaped by a backslash. If you want to include the cell address of the current cell in the command line, you may do so by entering "$$" in its place in the command. For example,

fkey 5 = "fmt $$ \"^D%A\""

will cause the F5 key to format the current cell to display the full weekday name of the numeric date value stored there. The ^D is a CTRL-D character, which denotes a date format.

Although it may be overridden by the *fkey* command, the F1 key is predefined by default to execute *man sc*. Unlike the user-defined function keys, this definition will also work in edit, insert, replace, and navigate modes.

To undefine a function key, merely define it as the empty string (""). Undefining the F1 key will restore the default behavior.

Plugins

There are three ways in which external programs can be used as plugins with *sc*. First, they can be used as external commands. When used as an external command, any command not recognized by *sc* will be searched for first in $HOME/.sc/plugins, and then in /usr/local/share/sc/plugins. If found, it will be run with its standard input and standard output redirected back to *sc*. These are used to send commands to, and receive responses back from *sc* in the same way that advanced macros do.

The second and third ways that programs can be used as plugins with *sc* are to automatically convert files to and from *sc* format based on their extensions. In order to use them in this way, you must first associate a given extension to a corresponding plugin (for reading) or plugout (for writing) using the *plugin* and *plugout* commands. These commands should be placed in your *.scrc* file, and have the following syntax:

plugin *"ext" = "programname"*

or

plugout *"ext" = "programname"*

where *ext* is the extension and *programname* is the name of the plugin program to be used for filenames with that extension. For input, the plugin merely reads the specified file, performs whatever conversion is necessary, and writes the resulting data to standard output. For output, the plugin writes *sc* com-

mands to standard output and reads the replies from standard input in the same way that an advanced macro would, and then converts the data to the appropriate format and writes it to a file with the specified filename.

FILES

/usr/local/share/sc/tutorial.sc	Tutorial spreadsheet.
$HOME/.scrc	Initialization commands.
./.scrc	More initialization commands.

SEE ALSO

bc(1), dc(1), crypt(1), psc(1)

BUGS

Top-to-bottom, left-to-right evaluation of expressions is silly. A proper following of the dependency graph with (perhaps) recourse to relaxation should be implemented.

On some systems, if the cell cursor is in column 0 with topline enabled (so the current cell is highlighted), or if any cell in column 0 is highlighted, the corresponding row number gets displayed and then blanked during a screen refresh. This looks like a bug in *curses*.

Many commands give no indication (a message or beep) if they have null effect. Some should give confirmation of their action, but they don't.

AUTHORS

This is a much modified version of a public domain spread sheet originally authored by James Gosling, and subsequently modified and posted to USENET by Mark Weiser under the name *vc*. The program was subsequently renamed *sc*, and further modified by numerous contributors, Jeff Buhrt of Proslink, Inc. and Robert Bond of Sequent, prominent among them. The current maintainer is Chuck Martin (nrocinu@myrealbox.com).

Other contributors include: Tom Anderson, Glenn T. Barry, Gregory Bond, Stephen (Steve) M. Brooks, Peter Brower, John Campbell, Lawrence Cipriani, Jim Clausing, Dave Close, Chris Cole, Jonathan Crompron, David I. Dalva, Glen Ditchfield, Sam Drake, James P. Dugal, Paul Eggert, Andy Fyfe, Jack Goral, Piercarlo "Peter" Grandi, Henk Hesselink, Jeffrey C Honig, Kurt Horton, Jonathan I. Kamens, Peter King, Tom Kloos, Michael Lapsley, Casey Leedom, Jay Lepreau, Dave Lewis, Rick Linck, Soren Lundsgaard, Tad Mannes, Rob McMahon, Chris Metcalf, Mark Nagel, Ulf Noren, Marius Olafsson, Gene H. Olson, Henk P. Penning, Rick Perry, Larry Philps, Eric Putz, Jim Richardson, Michael Richardson, R. P. C. Rodgers, Kim Sanders, Mike Schwartz, Alan Silverstein, Lowell Skoog, Herr Soeryantono, Tim Theisen, Tom Tkacik, Andy Valencia, Adri Verhoef, Rick Walker, Petri Wessman, and Tim Wilson.