NAME

pname − spreadsheet calculator

SYNOPSIS

pname [**-c**] [**-m**] [**-n**] [**-r**] [**-x**] [*file*]

DESCRIPTION

The spreadsheet calculator *pname* is based on rectangular tables much like a financial spreadsheet. When invoked it presents you with a table organized as rows and columns of cells. If invoked without a *file* argument, the table is initially empty. Otherwise *file* is read in (see the *Get* command below). 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 a on-line tutorial, type the command:

pname #LIBDIR#/tutorial.pname

To print a quick reference card, type the command:

pnameqref | [your_printer_commmand]

OPTIONS

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

−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).

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

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

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

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

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

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 *pname*. 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 a highlighted cell and a "<" on the screen, and a character cursor, indicated by the terminal's hardware cursor. The cell and character cursors are often the same. They differ when you type a command on the top line.

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 or if the number is to be entered while the character cursor is on the top line.

Commands which use the terminal's control key, such as ^*N*, work both when a command is being typed and when in normal mode.

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*. 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.

c Current cell highlighting. If enabled, the current cell is highlighted (using the terminal's standout mode, if available) in addition to being marked by the cell cursor.

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.

l Autolabeling. If enabled, using the define command (/d) 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*. The cursor controls (^*P*, ^*N*, ^*B*, ^*F*) in this mode will end a numeric entry.

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.

x Encryption. See the **−x** option.

$ 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 option toggles between three cases. The default is no action. If this option is used once, after each command which is terminated by a newline character is completed, the current cell will be moved down one row. If this option is used again, after each command which is terminated by a newline character is completed, the current cell will be moved right one column. Another use of this option will restore the default action.

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.

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. 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 */f*) 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.

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

autocalc/!autocalc
: Set/clear auto recalculation mode.

numeric/!numeric
: Set/clear numeric mode.

prescale/!prescale
: Set/clear numeric prescale mode.

extfun/!extfun
: Enable/disable external functions.

cellcur/!cellcur
: Set/clear current cell highlighting mode.

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

rndinfinity/!rndinfinity
: default: round-to-even (banker's round), *.5 will round to the closest even number; doing a 'set rndinfinity' will round *.5 up to the next integer (rounding to infinity).

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.

Cursor Control Commands

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

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

^B Move the cell cursor backward one column.

^F Move the cell cursor forward one column.

h, j, k, l
If the character cursor is not on the top line, these are alternate, *vi*-compatible cell cursor controls (left, down, up, right). Space is just like l (right).

H, J, K, L
If the character cursor is not on the top line, these move the cursor by half pages (left, down, up, right).

^H If the character cursor is not on the top line, *^H* is the same as *^B*.

SPACE
If the character cursor is not on the top line, the space bar is the same as *^F*.

TAB If the character cursor is on the top line, TAB starts a range (see below). Otherwise, it is the same as *^F*.

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.

^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.

g Go to a cell. *pname* prompts for a cell's name, a regular expression surrounded by quotes, or a number. If a cell's name such as *ae122* or a 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, *pname* 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. If a number is given, *pname* 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*>.

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. *pname* 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 $-\mathbf{n}$ option or $\hat{}T$ command, shows the prompt when you enter the first digit of a number (you can skip typing "=").

< 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.

F Enter a format string into the current cell. This format string overrides the precision specified with the "f" command. The format only applies to numeric values. The following characters can be used to build a 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. 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.

\ 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 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)").

Strings you enter must start with ". You can leave off the trailing " and *pname* 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

+ Forward through history (neat) (same as j)

- Backward through history (neat) (same as k)

ESC Done editing

TAB Mark && append a range (ex: A0:A0)
TAB, move around within a range; TAB, append range string.

CR Save

$ Goto last column

. Insert current dot buffer

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

0 Goto column 0

D Delete to send

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

R Replace mode; ESC revert back to edit mode

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

d Delete ...
b back 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

f Find the next char typed

h Move left a char

i Insert before cursor; ESC revert back to edit mode

j Forward through history (neat) (same as +)

k Backward through history (neat) (same as -)

l Move right a char

n Continue search

q Stop editing

r Replace char

t Goto a char

u Undo

w Forward a word

x Delete the current char (moving to the right)

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.

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 Mark a cell to be used as the source for the *copy* command.

c Copy the last marked cell to the current cell, updating row and column references in its numeric or string expression, if any.

+ If not in numeric mode, add the current numeric argument (default 1) to the value of the current cell. In numeric mode, "+" introduces a new numeric expression or value, the same as "=".

- If not in numeric mode, subtract the current numeric argument (default 1) from the value of the current cell. In numeric mode, "-" introduces a new, negative, numeric expression or value, like "=".

RETURN
If you are not editing a cell (top line is empty), pressing RETURN will make *pname* 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.

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 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 *pname* 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. Since *pname* 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. This facility is still primitive and could be much improved.

D Define a path for the *Run* command to use.

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 commands can be used on either rows or columns. The second letter of the command is either a row designator (one of the characters *r*, *^B*, *^F*, *h*, *l*) or a column designator (one of *c*, *^P*, *^N*, *k*, *j*). A small menu lists the choices for the second letter when you type the first letter of one of these commands. Commands which move or 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).

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.

ar, ac Append a new row (column) immediately following the current row (column). It is initialized as a copy of the current one.

dr, dc Delete the current row (column).

pr, pc, pm
Pull deleted rows (columns) back into the spreadsheet. The last deleted set of cells 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. *pm* (merge) does not insert rows or columns; it overwrites the cells beginning at the current cell cursor location.

vr, vc Remove expressions from the affected rows (columns), leaving only the values which were in the cells before the command was executed.

zr, zc 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 be still be hidden when you reload the spreadsheet. Hidden rows or columns are not printed by the *W* command.

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. Enter three numbers: the total width in characters of the column, the number of digits to follow decimal points, and the format type. Format types are 0 for fixed point, 1 for scientific notation, 2 for engineering notation, and 3 for dates. 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. This command has only a column version (no second letter).

@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.

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 start with a slash; the second letter of the command indicates which command. A small menu lists the choices for the second letter when you type "/". *pname* 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 */d* command (see below). Range name prompts require either an explicit range such as *A10:B20*, or a range name previously defined with a */d* 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*, *^N*, *^B*, *^F*) or the arrow keys. The cells in the default range are highlighted (using the terminal's standout mode, if available).

/x Clear a range. Cells cleared with this command may be recalled with any of the *pull* commands.

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

/c 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*.

/f Fill a range with constant values starting with a given value and increasing by a given increment. 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.

/d 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 must be more than two alpha characters long to differentiate them from a column names, and must not have embedded special characters. Names may include the character "_" or numerals as long as they occur after the first three alpha characters.

/l 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 anyway until it is unlocked.

/U This command is the opposite of the */l* command and thus unlocks a locked cell and makes it editable.

/s This command lists (shows) the currently defined range names. If there are no defined range names, then a message is given, otherwise it pipes output to *sort*, then to *less*. If the environment variable PAGER is set, its value is used in place of *less.*

/u Use this command to undefine a previously defined range name.

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

Miscellaneous Commands

Q
q
^C Exit from *pname*. If you made any changes since the last *Get* or *Put*, *pname* 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.

! Shell escape. *pname* 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.

^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.

^V Type, in the command line, the name of the current cell (the one at the cell cursor). This is useful when entering expressions which refer to other cells in the table.

^W Type, in the command line, the expression attached to the current cell. If there is none, the result is "?".

^A Type, in the command line, the numeric value of the current cell, if any.

The *^V*, *^W*, and *^A* commands only work when the character cursor is on the command line and beyond the first character.

TAB When the character cursor is on the top line, defines a range of cells via the cursor control commands or the arrow keys. The range is highlighted, starts at the cell where you typed TAB, and continues through the current cell cursor. Pressing TAB again causes the highlighted range to be entered into the command line and the highlighting to be turned off. 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, you don't need to press the first TAB to begin defining a range starting with the current cell.

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:

−, ˜
ˆ
∗, /
+, −
<, <=, =, !=, >=, >
&
|
?:

Built-in Range Functions

These functions return numeric values.

@sum(r) 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.

@prod(r) Multiply together all valid (nonblank) entries in the specified region.

@avg(r) Average all valid (nonblank) entries in the specified region.

@count(r) Count all valid (nonblank) entries in the specified region.

@max(r) Return the maximum value in the specified region. See also the multi argument version of *@max* below.

@min(r) Return the minimum value in the specified region. See also the multi argument version of *@min* below.

@stddev(r) Return the sample standard deviation of the cells in the specified region.

@lookup(e,r)
@lookup(se,r) 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(se,r,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(se,r,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(e,r) Use the value of the expression *e* to index into the range *r*. The numeric value at that position is returned. The value 1 selects the first item in the range, 2 selects the second item, etc. *R* should be either a single row or a single column.

@stindex(e,r) Use the value of *e* to index into the range *r*. The string value at that position is returned. The value 1 selects the first item in the range, 2 selects the second item, etc. The range should be either a single row or a single column.

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: round-to-even (banker's round), *.5 will round to the closest even number; 'set rndinfinity' will round *.5 up to the next integer.

@round(e,n) Round *e* to *n* decimal places. n may be positive to round off the right side of the decimal, and 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 @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.

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

@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, 7)
```

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

@capital(e) 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).

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

@capital(e) will convert the first letter of words in a string into 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 built 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(@mycol-1) @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 of 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 a ordinary annuity of 36 monthly payments of $1000 at 18% annual interest. It answers the question: "How much can I borrow at 18% for 30 years if I pay $1000 per month?"

Built-in Date and Time Functions

Time for *pname* follows the system standard: the number of seconds since 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) *@dts*(9,14,1988) converts the date September 14, 1988 to the number of seconds from the epoch to the first second of 9/14/88, local time. For example, *@date*(*@dts*(12,14,1976)) yields

Tue Dec 14 00:00:00 1976

The month should be range from 1 to 12, the day should range from 1 to the number of days in the specified month, and the year should range from 1970 to 1999.

@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) Convert the time in seconds to a date string 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*().

@year(e) Return the year. Valid years begin with 1970. 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.

FILES

#LIBDIR#/tutorial.pname	tutorial spreadsheet

SEE ALSO

bc(1), dc(1), crypt(1), ppname(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.

Only one previous value is saved from any call of *@ext*(). If it is used more than once in a spreadsheet and external functions are enabled and later disabled, the last returned value pops up in several places.

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. ({sequent, uunet}!sawmill!prslnk!buhrt) and Robert Bond of Sequent, prominent among them. 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,

PNAME #REVISION# (1)

Jeffrey C Honig, Kurt Horton, Jonathan I. Kamens, Peter King, Tom Kloos, 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.