

skb SKB V1.0 © Garvan O'Keeffe, 2008

SKB is an open source SUDOKU creator and solver written in FreeBASIC.

Rules

Complete the puzzle by entering the numbers 1 through 9 in each row, column and 3x3 box such that no number is repeated in any row, column or box.

Quick instructions

- Choose a puzzle of the desired difficulty by selecting "New -> Level 1" (or Level 2, Level 3) from the menu. Level 1 is the easiest level.
- To enter a number click on the small numbers using the right mouse button.
- To keep notes on possible candidates for a square click on the small numbers with the left mouse button.
- For keyboard use, the arrow keys select the cell and the number keys 1 through 9 enter a number. Use SHIFT plus a number to mark candidates for a cell.

SKB Help

Please refer to a Sudoku reference for a full explanation of the terms used for solving techniques. Brief explanations are given in the glossary below.

The downloaded zip file contains both Linux and Windows compiled programs. Window users can extract the folder to any desired location, and start skb.exe to run the program. Linux users can extract the archive to their home directory and start skb to run the program. SKB needs to read and write to its data files, so for Linux users the program needs to be installed in a location where you have read and write access. The source code is provided in a separate folder, and may be deleted if not required.

Program Usage

Mouse

- To select a number, click on the small numerals in the required cell with the right mouse button. Numbers in Black are given at the start of the puzzle and can not be edited unless you switch to setup mode ("New->Setup" on the menu).
- To mark candidates for a cell, click on the small numerals with the left mouse button. Marking of candidates is disabled in setup mode.
- Double clicking on the small numerals will also select a number, but all marked candidates are cleared for that cell, so right clicking may be more convenient in as it permits the use to undo the selection.
- The menu is currently only usable with a mouse.
- The function of the left and right mouse buttons can be swapped by selecting "Tools->Swap Mouse" from the menu.

Keyboard

- Use the arrow keys to select the desired cell, and the number keys to enter a number. Numbers in Black are given at the start of the puzzle and can not be edited unless you switch to setup mode ("New->Setup" on the menu).

- Hold down the shift key and press a number to mark candidates.
- To delete a number, select the cell and press any number, 0 to 9, the Del or Backspace key.

Levels

Select a new puzzle by choosing "New->Level 1", "New->Level 2" or "New->Level 3".

- Level 1 puzzles are solvable using the basic techniques of cross hatching, counting and locked candidates. Locked candidates can be easily overlooked so care is needed. Advanced users may wish to solve these puzzles without "marking up" candidates to make them more challenging.
- Level 2 puzzles contain locked candidates, naked pairs, trips or quads, hidden pairs, trips or quads or xyz-wings. These puzzles can be challenging to solve, and marking-up of candidates is usually required.
- Level 3 puzzles contain a combination of locked candidates, naked pairs, trips or quads, hidden pairs, trips or quads, n-fish, xyz-wings, xy-chains, forced chains and puzzles solvable using the nishio technique (or using colors). Users may not be able to solve these puzzles unless they have studied Sudoku solving techniques or read some reference or tutorial on Sudoku. They are very challenging to solve, even using all coloring and filtering tools available in SKB.

Puzzles are graded by solving technique, but users should be aware that there are usually several options to proceed at any given point in solving a puzzle, and SKB picks only one of these in making its grading.

Colors and filters

All givens for new puzzles are drawn in the same color (depending on the color theme). The user can select from a menu of colors ("Tools->Filter") to choose a color for their own entries. Selecting "Tools->Restart" from the menu will clear all user entered numbers and mark ups, allowing the game to be restarted. Selecting "Tools->Clear All" clears all givens, entries and marks to give a blank grid. Using colors to mark entries allows the use of "what-if" solving techniques where users can explore some possibilities and back track if necessary.

Selecting "Tools->Filter" will also display a menu bar of numbers to the left of the color selector where the user can select a number to filter the display of mark ups so that only a single number is displayed. Clicking on a number will display only that number; clicking on it a second time returns the display to normal. This is useful in advanced solving techniques.

Markup

SKB can help with the markup of candidates taking some of the drudgery out of marking up. If you select "Tools->Markup->Run" SKB will reset all user markups and markup the board using basic techniques. Selecting "Tools->Markup->Auto-on" will cause SKB to update the markups as you enter new numbers on the grid. The user is still required to discover locked candidates, and any of the advanced combinations, and auto-markup will not protect against user error in clearing or setting candidates. "Tools->Markup->Auto-off" turns off auto markup.

Theme

The window size can be resized to one of three settings using the "Tools->Theme" menu option. Select from the option of small, medium or large. Three color themes can also be selected on from this menu. The names of each theme and the colors of all graphical items in the display are controlled by a text file called skb.ini in the data directory. This text file can be edited by the user, and should have sufficient comments for the user to understand which entries control the colors of different graphical elements. The themes can also specify screen size if desired.

SKB is cross platform (Linux and Windows) and these two platforms have different text file

formats. If you find the file is in the wrong format for your platform, open SKB, change the selected theme so that the INI file is updated and close SKB again. SKB will save the INI file in the correct format for your system. A sample INI file is included as an appendix to this document should the supplied INI file get corrupted during editing.

The font used in the menu, message and time display is Bitstream Vera Sans Bold in 8 points (filename VeraSans8B.bmp in the data folder) converted from the TTF original to FreeBASIC "drawstring" format using the utility TTF2GFX available for download from the FreeBASIC forum. This font should not be edited in a bitmap editor or encoded character widths may be lost. However the program will work with a replacement font of the same name in the data directory, created with the same free utility.

The SUDOKU Logo and the font used for number display is hard coded into the program and cannot be changed.

Setup

To setup a new puzzle from an external source you select "Tools->Setup" from the menu, and enter the givens. When finished you can select "Tools->setup" again to toggle setup to off, or simply select a user color from the filter menu. Marking-up is disabled in setup mode.

custom.txt

It is also possible to load puzzles from a plain text data file "custom.txt" which must be in the data directory. This allows the import of puzzles from sources on the internet without the need to enter the puzzle by hand. Puzzles can be formatted in 9X9 blocks with or without delimiters, or as a long string of 81 characters. # is used as a comment marker. To load custom.txt, choose "File->Load->custom.txt" from the menu.

Only the first puzzle is read from custom text. A sample custom text is shown below.

Level 1 Sudoku puzzle created by SuDoKu electronic puzzle maker

```
4,0,0,1,0,0,0,0,9
7,8,2,0,9,4,6,0,0
0,0,0,0,0,0,0,2,0
9,0,7,6,3,5,0,8,0
0,1,0,0,2,0,0,0,4
0,0,0,0,0,0,7,0,0
6,0,9,0,0,0,8,0,3
0,0,0,0,4,7,0,6,2
0,2,0,9,0,0,0,0,0
```

Solve

The menu item "Tools->Solve" solves the puzzle and displays the result. A message is also displayed showing the strength of the puzzle (see Check below).

Check

The menu item "Tools->Check" will check to make sure the puzzle is solvable, and report a strength rating based on the techniques needed to solve the puzzle. The strength rating is calculated by counting the different solving techniques used to solve the puzzle, giving them a weighting for difficulty and summing the result as follows:

```
TECHNIQUE X WEIGHT
Locked candidates x 1
Naked pairs x 2
Hidden pairs x 2
Naked trips x 3
Hidden trips x 3
```

Naked quads x 4
Hidden quads x 4
X-Wing x 4
Swordfish x 5
Jellyfish x 6
XYZ-Wind x 6
XY-Chain x 6
Forced Chain x 6
Nishio x 6
Trial and error x 20

Strength = sum (TECHNIQUE X WEIGHT)

This gives only a rough indication of comparative difficulty, with larger ratings meaning more difficult.

Save and Load puzzles

Up to six puzzles can be saved to disk for later reuse. Choose "File->Save As" from the menu, and select a game number from 1 to 6 to save the active puzzle in. To reload this game choose "File->Load" and select the Game number. Game slots that have a saved game are marked with a + sign. To delete a saved game, clear the board ("Tools->Clear all") and save a blank board to a game number. Saving games at different states provides the user the opportunity to restart play from an intermediate point without needing to restart from the beginning. Note that loading saved games does not reset the timer. If you wish to restart the timer see Reset Time below.

Reset Time

SKB tracks the time a user spends to solve a puzzle. If you need to pause the timer, simply close the puzzle (by pressing Esc or from the menu) and when the puzzle is reloaded on restart, the timer is resumed. To reset the timer to zero, select "Tools->Reset Time" from the menu.

Puzzle database

SKB includes a database of 100 puzzles of each level. A background thread replaces these puzzles after they have been used. Difficult puzzles take a few minutes to create. It takes 15 to 45 minutes for most people to solve a puzzle, so a database of 100 level 3 puzzles should be sufficient to insure new puzzles with each usage unless you brows through the puzzles without spending time to solve them. If SKB displays a message that it is reusing old puzzles, you should let the program run in the background for a few hours replace the used puzzles.

Utility Programs (In the util folder)

skbreport. A command line utility (skbreport) is provided for reporting on the details of how the puzzle can be solved. This may be useful for analyzing puzzles if the user gets unexpected results when solving, or if the user wishes to discover how SKB solves a puzzle. The report utility analyses the last saved active puzzle - this is the puzzle displayed when the user last exited SKB. The report is printed to screen, and can be redirected to file if required (i.e. report >report.txt).

sktestpuzzles. A command line utility (sktestpuzzles) may be used for testing SKB with a large numbers of puzzles in text format. Input is from a text file named testpuzzles.txt, and output is printed to screen or redirected to file if required (i.e. sktestpuzzles > test.txt). The file format for the puzzles is a single 81 character string of characters from 0 to 9 representing the puzzle, with optional comments (see example). The reported methods used in solving a puzzle can differ from the expected both because the computer is unlikely to overlook simple methods of solving a puzzle that human solvers can easily miss, and also because of the order in which different possibilities are checked in different computer programs.

References

- Sudoku Creation and Grading , Andrew C. Stuart , 3rd February 2007

http://www.scanraid.com/Sudoku_Creation_and_Grading.pdf

- Wikipedia articles on Sudoku

<http://en.wikipedia.org/wiki/Sudoku>

http://en.wikipedia.org/wiki/List_of_Sudoku_terms_and_jargon

http://en.wikipedia.org/wiki/Algorithmics_of_sudoku

- sudopedia

http://www.sudopedia.org/wiki/Main_Page

Licensing

This software and source code is copyrighted by Garvan O'Keeffe, 2008. Permission is granted to modify or use portions of this source code in your own programs without restrictions or the need to acknowledge the original author.

Glossary of terms.

(In order of importance)

SUDOKU. Sudoku is it is a trademark of puzzle publisher Nikoli Co. Ltd in Japan.

SKB. The program name SKB is short for SU.DO.KU BOARD.

Given. A number given at the start of the puzzle.

Candidate. A possible value for a cell at any given point in the progress of solving the puzzle.

Unit. A unit is a row, column or block in the grid. The intersection of two units is the cells that are common to both units, normally taken to exclude the actual cells being considered.

Cross Hatching. Checking each number in turn to see if there is any row, column or block in which there is only one possible location for that number.

Counting. Checking each cell to see if there is only one possible candidate for that cell.

Marking up. Using the left mouse to mark all possible candidates for an individual cell. Some users prefer to eliminate candidates, others prefer to mark possible candidates. The result should be the same if done properly.

Locked Candidates. Locked candidates in rows or columns occur where the configuration of solved cells restricts a candidate to one 3x3 block. Because the candidate can occur only once in the block, it may safely be eliminated as a candidate from other cells in that block. Locked candidates in blocks occur where a candidate is restricted to a row or column within the block, allowing the elimination of that number as a candidate for other cells in that row or column.

Naked pairs, trips quads. A naked pair occurs when two cells in a row, column or block have the same two possible candidates. All other cells in this row, column or block can not contain either of these two numbers. A naked trip is when three cells contain the same three possible candidates, and a naked quad is when four cells contain the same four possible candidates. Note that for a trip it is not necessary for all three candidates to be present in all three cells, only that they do not appear in any other cells. The same applies for quads with four candidates.

Hidden pairs, trips, quads. A hidden pair occurs when two candidate in a row, column or block are confined to two cells. All other candidates for these cells can be eliminated. A trip is when three numbers are confined to three cells, and a quad is when four numbers are confined to four cells. Note that not all candidates must be present in all cells of the set. Hidden pairs trips and quads usually occur with naked pairs, trips or quads so the user may find them more often than reported by SKB, which will always search for naked pairs, trips and quads first.

N-fish (X-wing, swordfish and jellyfish). X-wing, swordfish and jellyfish are also known as N-fish where a 2-fish is an x-wing, a 3-fish is a swordfish and a 4-fish is jellyfish. N-fish are patterns of candidates where a number is restricted to a range of cells in a block pattern spanning multiple rows or columns. For a 2-fish example, if the candidate 5 was restricted to the first and last cell of row 2, and also restricted to the first and last cell of row 7, then the candidate 5 can be eliminate for all cells in the first and last column except for rows 2 and 7 where it must occur once. The same logic is applied to 3-fish with three rows or columns, and 4-fish with four rows or columns.

Color Solving. Color solving involves identifying pairs or groups of cells where there are only two possible locations for a given number, such that one must be correct and the other false. If these cells link up with other paired cells for the same number, then you can color the linked numbers in alternating colors to help visualize their relationship. This may reveal one of a linked pair that must be false and prove that all candidates of that color are false, or it may help identify individual candidates outside the linked pairs that must be false. Internally SKB uses the Nishio technique described below, which will solve the same class of puzzles as colors solving.

XY-Chain. An XY-Chain is a linked list of cells with two candidates. For example, {1,2},{2,9}, {9,2},{2,7},{7,1}, where {x,y} are the candidates for each linked cell. If the unlinked candidates at each end of the chain are the same (1 in the example) then this candidate can be eliminated from the intersection of the units of the first and last cell of the chain. When SKB checks for XY-Chains it considers only cells with two candidates, and it searches for candidates outside the chain that force a contradiction in the chain. To be considered a valid XY-Chain, the candidate being tested must intersect the chain at more than once cell.

XY-Wing. An XY-Wing is an XY-Chain consisting of three pairs. This is easier to spot for users, and may be common in difficult puzzles from other sources. SKB solves XY-Wings using the same function as the more general case of XY-Chain's, so they will be reported as XY-Chains in the **skbreport** utility.

XYZ-Wing. An XYZ-Wing consists of three cells linked together in the pattern XY-XYZ-YZ, where XYZ and XY share the same block and XYZ and YZ share the same row or column. The candidate Y which is common to all three cells, can be eliminated as a candidate from the intersection of the units of the three cells. This is a common pattern found in published puzzles rated as medium to difficult, and is worth looking for, as it is easy to spot.

Remote pairs. A remote pair are another special case of XY-Chains where the end candidates are the same, for example {1,2},{2,1},{1,2},{2,1}. If the number of pairs in the remote pair chain is odd, no candidates can be eliminated, if the number is even, both end candidates can be eliminated from the intersection of the units of the first and last cell. SKB

solves remote pairs using the same function as the more general case of XY-Chain's, so they will be reported as XY-Chains in the **skbreport** utility.

Forced Chain. A forced chain is a chain of linked XY candidates such that one cell is forced to a single candidates regardless of which option is chosen for the other cells. Solving forced chains is in practice little different than what-if analysis explained below, because if the cell that is forced to a single value is chosen first for evaluation, it will provide either a valid or an invalid solution, (assuming the puzzle has only one solution). SKB checks for forced chains by considering only cells where there are pairs of candidates, and testing for contradictions if either of these values is chosen, but does not do true what-if analysis which considers all cells. Puzzles containing Forced Chains can often be solved using XY-Chains, and visa-versa. SKB currently tests for Forced Chains first.

Nishio. Nishio is a solving technique where the candidates for each cell are inspected in turn to see if that candidate was true, would an error condition arise in other rows, columns or boxes where it was not possible to place that same candidates in any legal position. If it would cause an error, then it can be eliminated as a possible candidate for that cell. Nishio is a limited form of guessing that can be completed relatively easily by human solvers. It can be used as an alternative to solving using colors, and will also detect n-fish and many of the advanced techniques that rely on the inspection of candidates for a single number at a time. Pattern recognition plays an important role in Nishio because with practice users will quickly recognize which patterns lead to elimination of candidates. SKB provides filtering of candidates to make nishio easier to use.

What-if. Solving advanced puzzles involves recognizing patterns of candidates and knowing or deducing that some of these candidates can be eliminated. One of the references shown above lists 38 different kinds of patterns. In practice there is an easier way to solve advanced puzzles than trying to learn 38 different pattern types and searching for them sequentially. When you get stuck you explore some of the possibilities by taking a guess and seeing what the consequences would be. This is called "what-if" analyses, and though frowned upon as inelegant by those who have learned all 38 different pattern solving techniques, it is essentially no different than any other form of pattern recognition, and it is a practical approach to solving a difficult puzzle. The key to using "what-if" analysis is to identify the candidates that lead to contradiction, rather than try to solve the puzzle by choosing the correct solution. By eliminating these possibilities you can simplify a puzzle that otherwise looked impossible to solve. SKB does not use what-if analysis in its solving techniques because doing so could lead to very complex puzzles unsolvable by the pattern recognition techniques that make SUDOKU popular.

Trial and Error or brute force. A computer can quickly solve any valid Sudoku puzzle using trial and error. However to create puzzles that are enjoyable to play, human solving techniques must be mimicked, so "brute force" solving is best used to verify there is only one solution to a puzzle. All puzzles created by SKB can be solved using human solving techniques, and have only one solution. SKB uses trial and error only when the programmed human solving methods can make no further progress. Note that there are advanced human solving techniques that are not used in SKB, so when puzzles from other sources are solved by SKB it may report using trial and error even though there may be human solving techniques that could be used to solve the puzzle.

Sample INI file.

```
# Name: skb.ini
# Save as text file and place in the data folder
#
# SKB ini files for the colors and display size
#
# Themel, theme2 and theme3 are selectable from the theme menu
# using the names provided in the [themenames] section
#
# Color are in HTML RGB format #RRGGBB, entered as integers or as
# hexadecimal numbers prepended with &H or # (BASIC or HTML style)
#
# Size = 1, 2 or 3 for small, medium or large display
#
# message_color      The color displaying the time and messages
# background_color   The background color
# highlight_color     The color of the selected cell
# foreground_color    The color used to draw the grid
# mark_color         The color to mark candidates
# unmark_color       The color to unmark candidates
# error_color        The box color used to highlight errors
# logo_color         The font color to draw the Sudoku logo
# color0            The color to draw givens
# color1 to color6   The user selectable colors in the color bar
# color7            Unused
# menu_text_color    The text color of unselected text in the menu
# menu_bk_color      The background color of the menu
# menu_md_color      The color of the separator dividing menu items
# menu_ed_color      The color of the edge of the menu

[defaults]
size = 3
message_color = 0
background_color = 16777211
highlight_color = 16382196
foreground_color = 0
mark_color = 16711680
unmark_color = 12241038
error_color = 15831907
logo_color = 13718795
menu_text_color = 0
menu_select_text_color = 16777215
menu_bk_color = 15526360
menu_md_color = 11315353
menu_ed_color = 9674864
color0 = 0
color1 = 16711680
color2 = 32896
color3 = 255
color4 = 8388608
color5 = 16711807
color6 = 10793354
color7 = 65280

[themenames]
themel = Yellow
theme2 = Plain
theme3 = Green

[themel]
message_color = 255
background_color = 16777164
highlight_color = 16776960
foreground_color = 0
mark_color = 16711680
unmark_color = 13421823
error_color = 16711680
logo_color = 0
menu_text_color = 0
menu_select_text_color = 16777215
menu_bk_color = 15132390
menu_md_color = 9474192
menu_ed_color = 6579300
color0 = 0
color1 = 16711680
color2 = 16744192
color3 = 255
```



```
color4 = 8388608
color5 = 16711807
color6 = 32896
color7 = 65280
```

```
[theme2]
size = 2
message_color = 0
background_color = #ffffff
highlight_color = #e5e5e5
foreground_color = 0
mark_color = 16711680
unmark_color = 13421823
error_color = 16711680
logo_color = 0
menu_text_color = 0
menu_select_text_color = 16777215
menu_bk_color = 15132390
menu_md_color = 9474192
menu_ed_color = 6579300
color0 = 0
color1 = 16711680
color2 = 16744192
color3 = 255
color4 = 8388608
color5 = 16711807
color6 = 32896
color7 = 65280
```

```
[theme3]
size = 3
message_color = 0
background_color = #ffffffb
highlight_color = #f9f8f4
foreground_color = 0
mark_color = #ff0000
unmark_color = #BAC88E
error_color = #f19363
logo_color = #d1550b
menu_text_color = 0
menu_select_text_color = #ffffff
menu_bk_color = #ece9d8
menu_md_color = #aca899
menu_ed_color = #93a070
color0 = 0
color1 = 16711680
color2 = 32896
color3 = 255
color4 = 8388608
color5 = 16711807
color6 = #A4B18A
color7 = 65280
```