# Tutorial Documentation

*Release 2018*

**AdaCore**

**Dec 11, 2018**

# CONTENTS

CHAPTER

ONE

# INTRODUCTION

This document provides a guide through the major capabilities of the GNAT Programming Studio by working on a code example: sdc, a simple desktop calculator.

It is important to realize that the features that you are about to experiment with are available on multiple platforms, using the same user interface and capabilities, providing a user-friendly environment with a tight integration between the tools.

Start GPS in the directory containing the tutorial files, or if the directory is read-only, copy the `tutorial` directory and its subdirectories in a local (writable) area, and start GPS from the `tutorial` directory, so that GPS will load the right context.

By default, the tutorial sources can be found under *<prefix>/share/doc/gnat-gps/examples/tutorial*, where *<prefix>* is the prefix directory of the GPS installation.

Alternatively, if you have already started GPS in another directory, you can load the project *sdc.gpr* by using the menu *File->Open Project...*

CHAPTER

TWO

# QUICK OVERVIEW OF THE GPS AREAS

Having launched GPS, you should now have access to a main window composed of several areas:

- a menu bar at the top
- a tool bar under the menu bar
- on the left, a notebook allowing you to switch between Project, Outline and Scenario views
- the working area in the center
- the Messages window at the bottom

CHAPTER

THREE

# EDITING SOURCES

In the project view, open the *common* directory by clicking on the triangle on the left of *common*. This will open the directory and display a list of source files located in this directory.

Now, double click on `sdc.adb`: this will open a source editor on this file. The source code is syntax-highlighted: keywords, comments, strings and characters have different colors.

As with many other properties, colors are configurable in GPS:

Select the menu *Edit->Preferences...*. This will open a preferences dialog window.

Select the *Editor->Fonts & Colors* page by clicking on the triangle next to the item *Editor* and then selecting the *Fonts & Colors* item.

As you go over the various lines and labels, you will notice that by holding the mouse over a label, a tool tip pops up displaying on-line help about the selected item.

Change the background color of the *Keywords* by clicking on the last button, at the right of the *Keywords* line.

Choose a color, e.g a light green. When you're done with the color selection, click on *Select* in the color selection dialog.

After selecting a color, look at the effects in the source editor. If you like the new display, click on *Close* to close dialog, otherwise select another color.

CHAPTER

FOUR

# BUILDING APPLICATIONS

Select the icon *Build Main: sdc.adb* on the toolbar (fourth icon from the right): this will launch a complete build of the *sdc* application. Note also that you can use a key binding directly instead of this tool bar button (`F4`), or use the corresponding menu item *Build->Project->Sdc->sdc.adb*. If you use the menu item, an extra intermediate dialog is displayed showing the actual command line that will be used by GPS. Pressing `Execute` will launch the build as well.

The build has generated a number of errors in a new window: the *Locations* tree, displayed in the bottom area. The errors are also highlighted in the corresponding source editor.

GPS has automatically jumped to the first error message (*sdc.adb, 28:6 : (style) bad indentation*), at the line (28) and column (6) of the error.

Fix the error by hand by inserting a space.

Now you can fix the next error by moving the cursor to the line 30 (press the `down` arrow twice), and by using `Tab`: this key shortcut asks the source editor to automatically re-indent the current line.

Note that you can change this shortcut from the key shortcuts section of the Preferences dialog (menu *Edit->Preferences*, *General/Key shortcuts* section, *Format Selection* item).

You can then fix all the remaining errors by selecting the whole block (from line 28 to line 40) and pressing `Tab`. To select a block, you can either click on the left mouse button and select the area while holding the button, or using the keyboard by pressing the `Shift` key and moving the cursor using the `Up` or `Down` keys.

Press the `F4` key to build again. GPS will automatically save the modified files, and start a build. This behavior (automatic saving of files before building) can be configured in the preferences dialog.

If you look to the right of the toolbar in the GPS window, next to the omni-search entry, you will notice that a progress bar has appeared, displaying the current number of files compiled, and the number of remaining files. This progress bar disappears when the build is finished.

This should now report a successful build.

CHAPTER

FIVE

# SOURCE NAVIGATION

Now let's try to understand a little bit about how the program is working by looking at the `sdc.adb` editor: there's a loop, the main processing is done by the functions Process and Next (at line 30).

Click around line 30, move the mouse over *Process* and let a tool tip appear (*Tokens.Process global procedure declared at tokens.ads:19*): this gives information about the kind of entity and the location (file and line) of the declaration of this procedure, the profile of the parameters, and documentation for this function, as extracted from the comments surrounding the procedure declaration.

Do the same for *Next* (*Tokens.Next global function declared at tokens.ads:15*).

Keeping the mouse over *Next*, display the contextual menu by clicking on the right mouse button, then click on *Goto declaration of Next*: we're now in the package *Tokens*, in file `tokens.ads`; but where is this file in the project?

A simple way to locate a file in the *Project* view is to use the contextual menu from the source editor: *Locate in Project View: tokens.ads*.

You can also use the filter entry located at the top of the *Project* view.

CHAPTER

SIX

# PROJECT VIEW (ENTITIES)

Click on the triangle to open `tokens.ads` entities. When you click on a file in the project view, you see language sensitive information about the file, such as *packages*, *subprograms*, and *tasks* for *Ada*.

Open the *subprogram* category, then click on *Process*: this will open *tokens.ads* and move the cursor to the first line of the procedure *Process*.

Similarly, click on *Next* and move your mouse to *Next* in the source editor.

CHAPTER

SEVEN

# BACK TO SOURCE NAVIGATION

Move the mouse over the *Next* identifier in `tokens.ads` editor, and then hold the `Control` key: while you're holding the key, move the mouse over entities: these entities now become clickable hyperlinks. Clicking on the first mouse button will go to the declaration of the entity highlighted (or the body if you are already on the declaration), and clicking on the middle mouse button will go to the body directly: move the mouse back to *Next* and click. Alternatively, you can use the contextual menu and select *Goto body of Next*; then scroll through the procedure *Next*, move the mouse on *Instructions.Read* at line 46, hold `control` again and click with the middle mouse button (or from the contextual menu, select *Goto body of Read*).

We've now navigated quite a bit through the application source code, which you can verify by clicking on the left arrow in the tool bar, to go back to the previous locations visited.

Repeat the operation until you're back in `sdc.adb`. As with the undo/redo capability in the source editor, the *goto previous/next location* is infinite.

CHAPTER

# EIGHT

# CODE COMPLETION

Go to line 38 of sdc.adb. You can see there is a null instruction for the case of Stack.Overflow. We are going to add some code there, using the code assist capabilities.

Type `Enter` to create a new line, then type *Scr*. A completion popup is displayed, showing all the entities of the project begining with *Scr*. Double click on *Screen_Output*: the code is automatically completed in the editor. Then add a dot in your code. The completion popup is triggered automatically and will offer you the option of completing your code with the entities contained in the *Screen_Output* package. Select *Msg*, add a space, and then add an open parenthesis. Once again, the completion windows pops up and shows the possible parameters for msg. If you choose the first entry of the completion list ("params of Msg"), the call is automatically completed by a list of named parameters. Complete the list by giving e.g. *"The stack is full."* for *S1*, *""* for *S2*, and *True* for *End_Line*.

Don't forget to add a semicolon at the end of the statement. Then hit `F4` to rebuild the application.

CHAPTER

NINE

# RUN

It is now time to run the application: select the menu *Build->Run->Sdc->sdc*, which will open a dialog window. In the text input field (selected by default), press the right arrow key and then insert *input.txt*: this is the name of a text file that will be passed as argument to the *sdc* program.

The text input should now read: *%E input.txt* and the full command that will be executed is displayed underneath: *. . . /gps/tutorial/obj/sdc input.txt*

Now click on *Execute*: a new window titled *Run: sdc* is created at the bottom of the main window where the sdc application runs and displays an unexpected internal error: this is a good opportunity to use the integrated debugger.

Place the mouse cursor over the tab titled *Run: sdc*: a cross will appear on the right of the label: click on it to close the window.

CHAPTER

# TEN

# DEBUG

Open the preferences dialog (menu *Edit->Preferences...*) and click on the *Debugger* item on the left; set the button *Break on exceptions* to *Enabled*: this will enable by default a special breakpoint every time an exception is raised. Click on *Close* to close dialog.

Now select the menu *Debug->Initialize->Sdc->sdc*: GPS automatically switches to the *Debug* perspective as shown in the menu *Window->Perspectives*, and new windows have appeared: the debugger variables window, the breakpoints view and the debugger console at the bottom, and the call stack window on the right.

You can also look at the various debug menu item and tool bar buttons which are now activated.

On the call stack window (you can use the menu *Debug->Data->Call Stack* to open it if you do not have it displayed), select the local configuration menu: various pieces of information can be displayed or removed in the call stack. From this local configuration menu, add the *Frame Number* info by clicking on it.

Now select the menu *Debug->Run...* and type *input.txt* in the text input field. Check that 'Stop at beginning of main subprogram' and 'Use exec dir instead of current dir' are not selected. Click on *OK*: the debugger should stop on an exception (*Constraint_Error* in the file `stack.adb`, at line 49).

Go up in the call stack by clicking on the *tokens.process* frame (frame number will vary, depending on your GNAT version and platform).

If you move the mouse over the parameter *T* at line 64, a tool tip is displayed showing the value of *T*. You have probably noticed that tool tips, like menus, are contextual: depending on the current session and on the entity selected, different information is displayed.

Select the contextual menu *Debug->Display T*: this will highlight the data window, with a new box displaying graphically the contents of the different fields of *T*, each clearly separated.

Move your mouse over the *1:T* box, select the contextual menu *Display->Show Value + Type*: this displays for all fields both their type and value.

Special colors are used in the data display: blue for pointers that can be dereferenced by a double-click (double click on *T.val*); red for fields that have been modified since last step.

From the T box, right-click to display the contextual menu and select *View memory at address of T*: a memory view is opened on top of the source editors. Use the *up* and *down* arrows on the right to visit memory.

Click in the memory dump, and modify it by typing numbers. Notice the red color for modified values; click on *Undo Changes* to cancel the modifications; then close the memory window by e.g. clicking on the *x* icon in the tab or pressing `Ctrl-W`.

In the call stack, go back to the *stack.push* frame. Move the mouse over *Last* and let the debugger display its value: 0. From the contextual menu, select *Goto declaration of Last*: this will jump to the line 16 of `stack.adb`, where you can see that *Last* is a *Natural*. Now click on the *Goto Previous Location* button in the tool bar: we're now back at line 49 where we can see that for a *Push* procedure, *Last* should be incremented, and not decremented.

Fix the line to *Last := Last + 1;*

Save the file (`Ctrl-S`); End the debug session: menu *Debug->Terminate*; Rebuild (press `F4` key); Rerun (menu *Build->Run->sdc*, click on *Execute*): the program now completes as expected. Close the execution window.

CHAPTER

# ELEVEN

# CALL GRAPH

Now go back to the file `sdc.adb`, move the mouse over the procedure *sdc* at line 8, select the contextual menu *Browsers->Sdc calls*: this will open a new window titled *Call graph browser*.

Note that there is also a top level contextual menu (*Sdc calls*) which provides a tree view of the callers/callees.

In the call graph, click on the right arrow of *Process* (one of the first items on the top). Also click on the right arrow of *Error_Msg*.

The call graph contains a tool bar; the button on the right of this tool bar brings up the options menu.

You may want to play with the zoom (= and − keys).

Click on right arrow of *Process* (*(Decl) instructions.ads:12*).

The items can also be moved: move e.g *Msg* item around.

You can also recompute the layout of all the current items by using the *Refresh layout* button (the sixth button from the left on the local tool bar).

Click on left arrow of *Msg* to display who is calling *Msg*. Notice that *View* calls *Msg*.

Click on left arrow of *View*: the arrow disappears, and no new items are created, which means that *View* isn't called by anyone, so we're now going to remove this procedure.

CHAPTER

# TWELVE

# LOCATIONS VIEW

From *Call Graph Browser*, select the contextual menu *Goto declaration of View*, this will open the file `stack.ads` at line 32. Then from the source editor (file `stack.ads`), select the contextual menu *References->Find all references to View*: this highlights the *Locations* tree which now contains all the references for *View*, grouped by files (`stack.ads` and `stack.adb`).

The first location is highlighted automatically: this is the spec of the procedure *View*. Now click in the tree on the triangle at the left of `stack.adb`: two locations are listed, at line 90 and 97. Click on each of these locations: they correspond to the procedure body.

The *Find all references* capability is another way to list all the uses of an entity, and it confirms that *View* isn't called in our project.

Remove *View* body by e.g selecting it, and pressing the `Delete` key, then save the file (`Ctrl-S`).

Do the same for the spec, save the file.

Close the `stack.ads` and `stack.adb` files (menu File->Close, or using the shortcut `Ctrl-W`). Rebuild by pressing the `F4` key.

Let's now see how to create a project corresponding to the *sdc* project we've used in this tutorial.

CHAPTER

# THIRTEEN

# PROJECTS

## 13.1 Project Wizard

Go to the menu *File->New Project...*: this opens up the GPS project creation wizard.

The first page of the wizard allows you to select a pre-defined project template in the left-hand pane. These project templates are organized according to the technology they use (e.g: *AWS*) or the platform that is targeted (e.g: *STM32F4 compatible*). The description of the currently selected project is displayed on the right-hand side pane.

Select a project template and click on *Next*: a page asking you the name and the location of your project will appear. This page may also list project template-specific options.

Once completed, click on *Apply* to actually create the project. Note that you can still can customize your newly created project after is creation using the *Project properties* editor.

## 13.2 Project properties

In the project view, on the project *sdc*, use the contextual menu *Project->Properties*. All the properties set in the project wizard can be found here as well. You can switch between pages by clicking on the tabs located along the left side of the window.

Once you're done exploring the property pages, click on the *Cancel* button to close the properties window.

## 13.3 Variable editor

Select the window titled "Scenario". If not available, you can open it using the menu *View->Scenario*. This window contains a *Build* label.

This is a configuration variable. With GPS and the GNAT project facility, you can define as many configuration variables as you want, and modify any project settings (e.g. switches, sources, ...) based on the values of configuration variables. These variables can also take any number of different values.

The *Build* variable demonstrates a typical *Debug/Production* configuration where we've set different switches for the two modes.

Right click on the *Build* label and select *Edit properties of Build...*: this opens the variable editor, where values can be added or renamed. Close the variable editor by clicking on the *Cancel* button.

Now, let's take a look at the switches set in the project.

## 13.4 Switch editor

Select the menu item *View->File Switches*: a global switch editor is displayed in the working area, showing the switches associated with each file in the *sdc* project.

The editor lists the switches associated with each file in the project. Gray entries indicate default (global) switches. Notice that `screen_output.adb` has specific switches, which are highlighted using a different font.

Switch between *Debug* and *Production* mode in the *Build* combo box: the switches are updated automatically.

Back to our project, let's now examine the dependencies between sources.

## 13.5 Source dependencies

Select `sdc.adb` in the *Project View* and then the contextual menu item *Show dependencies for sdc.adb*: this will open a new graph showing the dependencies between sources of the project.

Click on the right arrow of `tokens.ads` to display the files that `tokens.ads` depends on. Similarly, click on the right arrow of `stack.ads`.

## 13.6 Project dependencies

Back in the project view, on the *Sdc* project, select the contextual menu *Project->Dependencies*, then on the *Add From File*, then open the *tutorial* directory and click on the *projects* subdirectory. Select the file *prj1.gpr*, click on *OK*. Click on *Apply* to validate the change.

You can see the new dependency added in the project view, as a list (or tree, if 'Show flat view' is enabled in local configuration menu) of projects. In particular, project dependencies are duplicated when tree view is used: if you open the *prj1* icon by clicking on the triangle, and then similarly open the *prj2* icon, you will notice that the project *prj4* is displayed twice: once as a dependency of *prj2*, and once as a dependency of *prj1*.

GPS can also display the graph of dependencies between projects: on *Sdc* project, use the contextual menu *Show projects imported by Sdc*: this will open a project hierarchy browser.

On the *Sdc* project, select the contextual menu *Show projects imported by Sdc recursively*.

In the browser, you can move the project items, and select them to highlight the dependencies.

CHAPTER

FOURTEEN

# EPILOGUE

This completes our tour of GPS, the GNAT Programming Studio. We hope this tutorial gave you a good overview of the general capabilities of GPS. A non-exhaustive list of the features not mentioned in this document includes:

- Documentation generation
- Automatic generation of body files
- Pretty printing
- Visual comparison of files
- Version control
- Flexible multiple document interface
- Code coverage
- Coding standard verification
- Extensive customization through Python

For more information, please see the *User's Guide* (**'gps.html <gps.html'_**) or look at the menus, which give access to most of these capabilities.