

# Contents

|          |                                       |          |
|----------|---------------------------------------|----------|
| <b>1</b> | <b>Shallot Manual</b>                 | <b>1</b> |
| 1.1      | License                               | 1        |
| 1.2      | About                                 | 1        |
| 1.3      | Up-to-date?                           | 1        |
| 1.4      | Maturity                              | 2        |
| 1.5      | Dependencies                          | 2        |
| 1.6      | Introduction                          | 2        |
| 1.7      | Using Shallot                         | 2        |
| 1.7.1    | Beginning                             | 2        |
| 1.7.2    | Customization                         | 3        |
| 1.7.2.1  | Settings                              | 3        |
| 1.7.2.2  | Under The Hood Stuff                  | 4        |
| 1.7.3    | Plugin Interface                      | 4        |
| 1.7.3.1  | Developing Plugins                    | 4        |
| 1.8      | Appendix: Installing A Shallot Plugin | 5        |
| 1.9      | Appendix: Installation                | 5        |
| 1.9.1    | Source Code Archive                   | 5        |



# Chapter 1

## Shallot Manual

### 1.1 License

shallot is written by Josef Hahn under the terms of the GPLv3 or higher.

Please read the `LICENSE` file from the package and the [Dependencies](#) section for included third-party stuff.

### 1.2 About

Shallot is a file manager with the maximum degree of flexibility and customizability.

Some features:

- Unlimited number of file panels
- Lots of customization options, e.g.
  - directory tree can be hidden and is flexible in behavior
  - lots of view options
  - program behavior
  - ...
  - while conserving a good usability
- It nicely works with convoluted file systems, like editing an image in an archive in another archive on a network drive.
- It has a plugin interface which allows one to implement many additional functionality (like new filesystems) with Python scripting, without any compiler hassles.
- It is Qt5 based and plays nicely with modern Linux desktops. See the downloads which other operating systems are supported (maybe with a few slight limitations).

### 1.3 Up-to-date?

Are you currently reading from another source than the homepage? Are you in doubt if that place is up-to-date? If yes, you should visit <https://pseudopolis.eu/wiki/pino/projs/shallot> and check that. You are currently reading the manual for version 1.0.3049.

## 1.4 Maturity

In this version, the state of shallot is considered as production-stable.

## 1.5 Dependencies

There are external parts which are used by shallot. Many thanks to the projects and all participants.

Typical GNU/Linux Desktop *recommended (maybe has alternatives)*

Qt5 *required*

Python 3.4 *required*

Glib (for gio) *optional*

libsecret *optional*

font 'Symbola' *included* : for logo symbol; free for use; copied from [here](#).

banner image *included* : `meta/homepage_bannerimage.png`; license [CC BY-SA 3.0](#); copied from [\[here\]](#)([http://en.wikipedia.org/wiki/File:Shallot%28Sambar\\_Onion%29\\_%282%29.JPG](http://en.wikipedia.org/wiki/File:Shallot%28Sambar_Onion%29_%282%29.JPG)).

icon set 'Elegant Font' *included* : files 'icons/\*.from.elegant\_font.\*'; [license](#); see [homepage](#).

all files in `/_meta` *included* : if not mentioned otherwise, Copyright 2015 Josef Hahn under license [CC BY-SA 3.0](#) license.

## 1.6 Introduction

Please read how to make Shallot ready for the first steps in [Appendix: Installation](#).

## 1.7 Using Shallot

### 1.7.1 Beginning

Whenever you start Shallot, you see the main window, which is assembled by the following parts:

- An action bar on the top: A lot of buttons which can trigger some actions. The main menu is also here ('three horizontal bars' icon).
- The address bar: See the current directory path and navigate upwards to the breadcrumbs or enter a new location.
- The directory tree on the left: A tree view of the filesystem(s) for easy navigation in wild directory hierarchies.
- The file view on the right: Shows the content of the current directory. There might be more than just one of this views (if you configure it this way).
- A status bar on the bottom: Comes up when it wants to tell you somethings.

The general usage is not very different from other general purpose file managers. This text will not go that deep into it:

Navigate to filesystem places where you plan to execute some work, select files and apply some operations to them. The action bar shows some available operations, while the context menu will list some more.

Please use everything with caution according to the value of your data. You would not be the first one destroying a month of work by carelessly removing stuff :-P

## 1.7.2 Customization

The Shallot user interface provides customization options at many places. Many view-related ones are in the view menu (what you can find in the action bar). There are of course many other places which provide some degree of customized behavior as well. All relevant ones can be conserved between sessions. There are two different categories of customization options in Shallot, which can be applied in a durable way:

- **Settings:** This category contains everything which is designed to routinely be changed by the user and which can be stored. This can e.g. be a view aspect like the visibility of the directory tree.
- *Under The Hood* stuff: Some options which work just fine when completely untouched for the most users. If you ever make a change here, this will be very rare situations. The path to some internal system tools can be considered here, if your system provides them in an untypical location.

Now some details will follow about these categories.

### 1.7.2.1 Settings

Settings can be done at many different places. While the view menu contains some actions for changing view-related settings, actions in other menus may somehow set settings as well. However, all those settings can be stored in a similar way. They all have another thing in common: They *must* be stored explicitly by the user, otherwise they will be dropped when Shallot closes.

#### 1.7.2.1.1 Saving Settings

For saving some settings, use 'Save settings' in the main menu.

The appearing dialog is quite powerful (as the Shallot settings system is). However, it has a reduced beginners mode, which appears at first.

In this mode, you can (un)mark some settings from the list to be saved for later sessions. This is quite simple. If you save, you have conserved the chosen settings together with the displayed values for the future. Each box represents one setting, the boxes are split up into categories for better overview.

You can do that many times, which accumulates all your settings internally. To unset some settings again, use the manage dialog as described in the next section.

In the header, there is a button for showing also advanced stuff. This brings a lot of more power. The following describes the additional functionality.

While the basic mode just allows storage independently of the directory, the advanced mode allows to bind settings to a directory. Whenever you visit this directory, the stored setting values become applied. In this regard you can also decide to apply just on that single directory or on all its subdirectories as well.

The multi-profile functionality is part of the advanced stuff as well. Instead of the default profile, you can split settings up into different profiles for different usage scenarios. The main menu allows to switch between profiles. In this regard you can also decide to also inherit settings from other profiles (either on the global or on the per-directory level).

Within some of the settings boxes, there now also appears a new check box. If you enable it, Shallot will bind this value just to a single file list. This is useful for storing different settings for different file views (if you use more than one). Since the dialog always refers to that file list, which is currently active in the main window, you have to use it more than once for storing different settings for different fileviews.

Finally there are some more settings (i.e. more boxes) available in the advanced mode.

#### 1.7.2.1.2 Managing Settings

The settings are accumulated internally in a last-one-wins ways at evaluation time. In storage, they reside mostly as you stored them. There are some reasons for managing the settings which are currently stored:

- getting an overview
- removing wrong or outdated stuff
- restructuring/changing/rebuilding

Use the 'Manage saved settings' action from the main menu for this concerns.

You should at first choose the correct profile if you are not interested in the selected profile but a different one. If you have never created a new Shallot profile, leave this as it is.

On the left side there is a list of all the moments you stored some settings. You see if they are globally or if they apply to a certain directory. You can remove single entries of that list after selecting it, or you can drop the entire profile. Selecting an entry shows more information.

On the right side, you see the content of the selected entry.

Please note: You can't technically change things once they are stored. For doing this, you would have to close Shallot and directly modify some xml files somewhere deeply hidden in your user profile. This is actually the only option you have when everything else is not an option :-P However, the recommended procedure is to just remove the old entry and maybe to store the setting again according to your new wishes.

#### 1.7.2.2 Under The Hood Stuff

All those options are accessible from one the 'Tuning' dialog. You find it in the main menu.

Each available option is one box in those dialog. A box contains a description text and the current value of that option. For better overview, the boxes are aligned in some tabs. Follow the on-screen instructions for changing something here.

### 1.7.3 Plugin Interface

There is a plugin interface for shallot, which allows external persons to develop additional features for Shallot. Those plugins are written in Python and use the Shallot scripting api for interaction.

#### 1.7.3.1 Developing Plugins

For the beginning, you should enable the Plugin Management in Shallot. In the 'Tuning' dialog, you should find it.

After restarting, you find a new top-level node which give a list of installed plugins. It is divided into plugins built into Shallot itself, plugins installed somewhere for all users and plugins installed somewhere for just a the current user.

There is a comprehensive documentation for all further steps accessible from within the Plugin Management.

## 1.8 Appendix: Installing A Shallot Plugin

If a plugin comes as an installation package for your operating system, just install it in the usual way. If it is the naked plugin file, copy the file into `$HOME/.shallot/plugins`.

Restart Shallot afterwards.

## 1.9 Appendix: Installation

Install Shallot via the installation package for your environment, if a suitable one exists for download. This also takes care of installing dependencies and doing preparation (unless mentioned otherwise in the installation procedure). After the installation, you can skip the rest of this section.

### 1.9.1 Source Code Archive

Use the source code archive as fallback. Extract it to a location which is convenient to you (Windows users need an external archive program; for example the great '7-Zip' tool). Also take a look at the Dependencies for external stuff you need to install as well.

After extraction, open the Shallot root directory in a terminal and run

```
qmake  
make
```

This builds Shallot from the source code, resulting in an executable. You should be able to execute the new `shallot` executable afterwards.

Instead of the command line approach, you can also use the QtCreator IDE and just load and run it there.

Note: There are some configuration flags in the beginning of the `shallot.pro` file. You should check them before building and enable the optional features as desired.

