vpxPrint

Mac & Linux Reference Manual

vpxPrint Studio

Reference Manual for LINUX and MacOS

Version 10.48

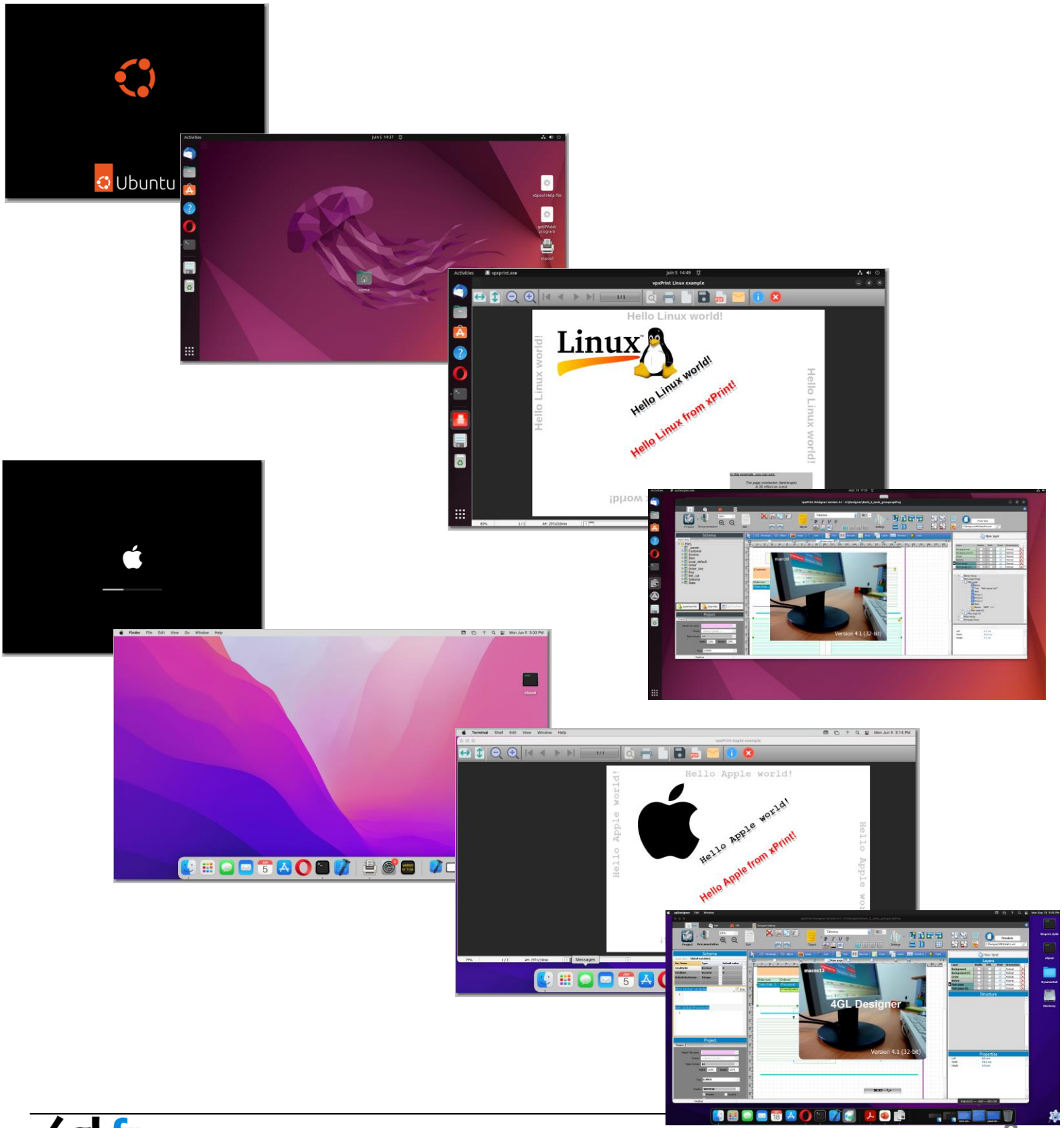
December 2024

4GL

vpxPrint Studio

4gl.fr

Linux and MacOS:

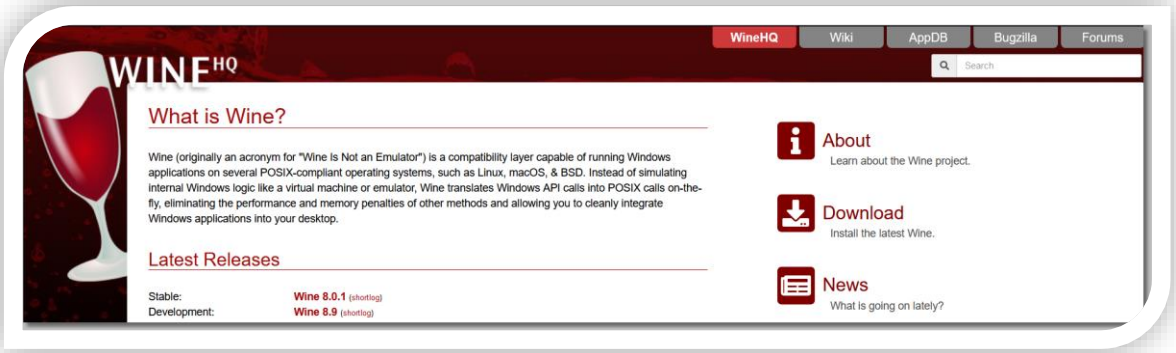


Introducing Wine...

What is Wine?

Wine (originally an acronym for "**Wine Is Not an Emulator**") is a compatibility layer capable of running Windows applications on several POSIX-compliant operating systems, such as Linux, macOS, & BSD.

Instead of simulating internal Windows logic like a virtual machine or emulator, Wine translates Windows API calls into POSIX calls on-the-fly, **eliminating the performance and memory penalties** of other methods and allowing you to cleanly integrate Windows applications into your desktop.

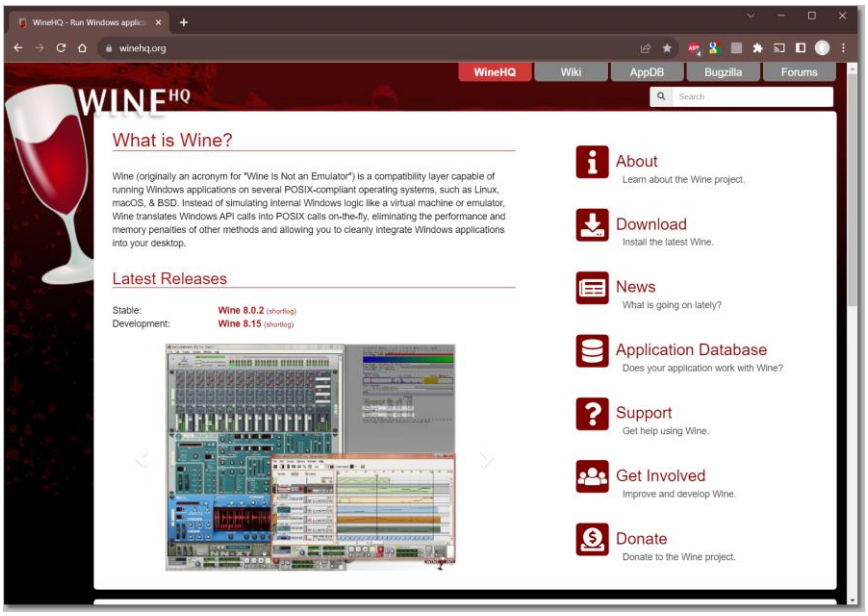


What have we done?

1. We've tested vpxPrint in Linux & MacOS environments,
2. ... established strategies for use,
3. ... developed tools and native **.so** (Linux) / **.dylib** (MacOS) libraries to build easy portability between systems,
4. ... developed demos to provide some basic examples in C++, Apple Xcode, Python, Pascal etc.
5. ... tested the installation of components,
6. ... integrated xSpool in the strategy,
7. ... added some new options to vpxPrint (**10.45**),
8. ... fixed certain problems that could arise,
9. ... and finally modified our **setup** to automatically create Unix/MacOS directories and native tools during standard installation...

More about Wine

Wine WEB site
(winehq.org)



Wine (originally an acronym for "Wine Is Not an Emulator") is a compatibility layer capable of running Windows applications on several POSIX-compliant operating systems, such as Linux, macOS, & BSD. Instead of simulating internal Windows logic like a virtual machine or emulator, Wine translates Windows API calls into POSIX calls on-the-fly, eliminating the performance and memory penalties of other methods and allowing you to cleanly integrate Windows applications into your desktop.

Wine began in 1993 under the initial coordination of Bob Amstadt as a way to support running Windows 3.1 programs on Linux. Very early on, leadership over Wine's development passed to Alexandre Julliard, who has managed the project ever since. Over the years, as the Windows API and applications have evolved to take advantage of new hardware and software, Wine has adapted to support new features, all while being ported to other OSes, becoming more stable, and providing a better user-experience.

An ambitious project by definition, work on Wine would steadily continue for 15 years before the program finally reached v1.0, the first stable release, in 2008. Several releases later, Wine is still under active development today, and although there is more work to be done, millions of people are estimated to use Wine to run their Windows software on the OS of their choice.

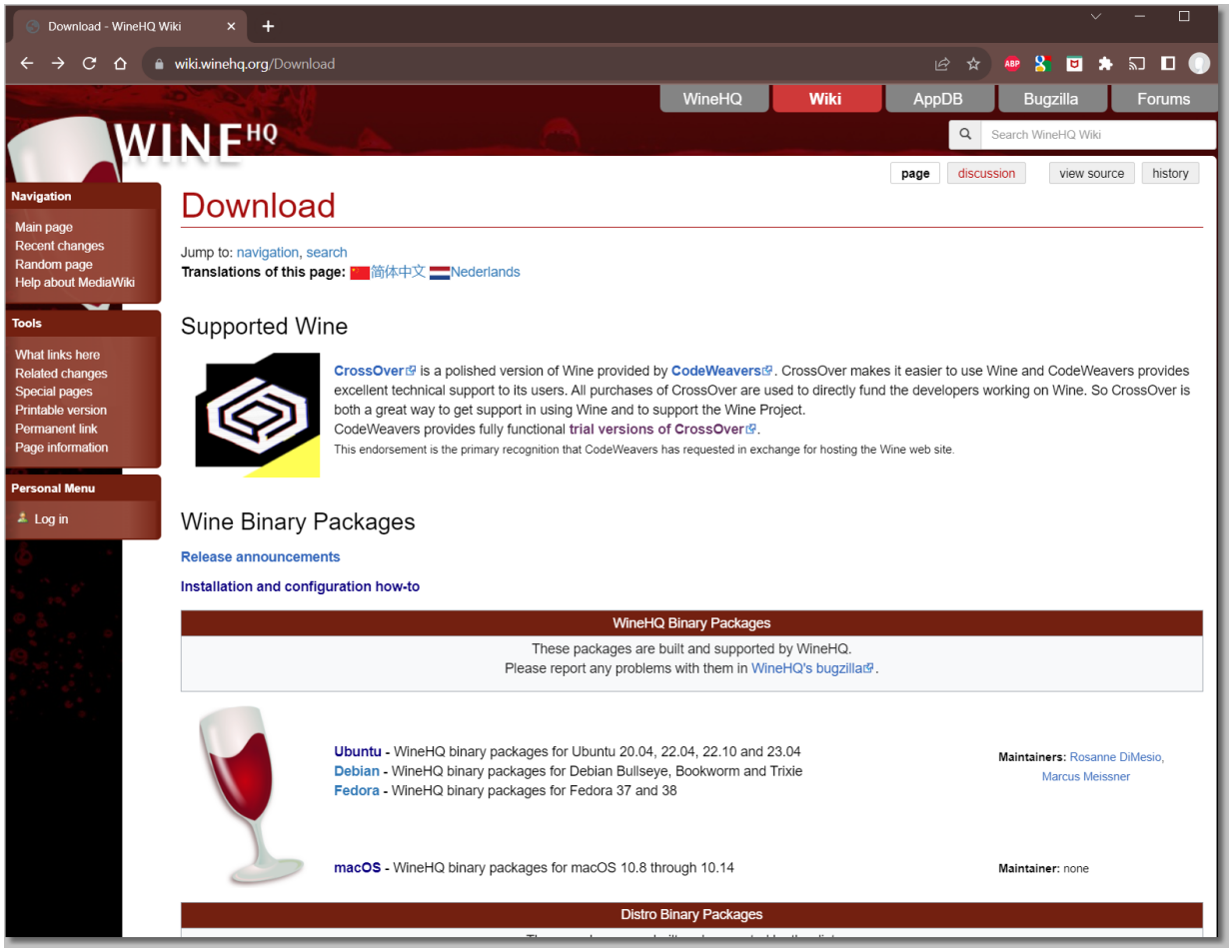
Open Source and User Driven

Wine will always be free software. Approximately half of Wine's source code is written by volunteers, with the remaining effort sponsored by commercial interests, especially CodeWeavers, which sells a supported version of Wine.

Wine is heavily reliant on its user community too. Users volunteer their time to share tips and test results on how well their programs work in our Application Database, file bug reports to notify developers of problems in our Bug-Tracker, and answer questions in our forums.

Wine LINUX installation

Simply download the Linux package from the WineHQ site and install it.



Wine MACOS installation

Part 1: Install Homebrew

Homebrew is a package manager that makes installing open source programs much easier. In particular, trying to install a large program like Wine without the help of a package manager would be tremendously difficult. Fortunately, Homebrew itself is simple to install: just open up the Terminal and run this command:

ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

-or-

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

The Terminal will tell you what it's about to do, and ask you if you want to proceed: press Enter to do so. The Terminal may then ask for a password: this is the password to the Admin account on your computer. As a security measure, the Terminal does not display anything as you type, not even asterisks (*). Type your password anyway, and press Enter. If you get some kind of error, it might be because the Admin account doesn't have a password set. Setting a password is required.

Installing Homebrew should only take a few seconds or minutes (depending on the speed of your internet connection). When it's done, the Terminal will say that the installation was successful, and ask you to run brew doctor. Do as suggested:

brew doctor

This will make Homebrew inspect the system and make sure that everything is set up correctly. If the Terminal informs you of any issues, fix them and then run brew doctor again to verify that the problem I fixed. When everything is set up correctly, your system is ready to brew.

Note: If Homebrew tells you that you need to agree to the Xcode license, you can do that by running:

sudo xcodebuild -license

The Terminal window will fill up with the Xcode license: read it, type agree and hit enter to agree to the license.

Part 2: Install Wine Using Homebrew

Enter:

brew cask install wine-stable

-or-

brew install --cask --no-quarantine wine-devel

where:

- brew refers to just installed Homebrew.
- cask refers to Homebrew Cask, an extension to Homebrew that is used to install GUI application on the computer. (GUI stands for "Graphical User Interface")
- install indicates a software installation command,
- wine-stable is the software to install. Wine has a "**stable**" version and a "**devel**" version.

Homebrew will start automatically downloading and installing software. It might start by installing software that has a totally different name. Like most complex applications, Wine doesn't work alone -- it relies on several other pieces of software to run correctly. These are called "dependencies", and Homebrew install them automatically when needed.

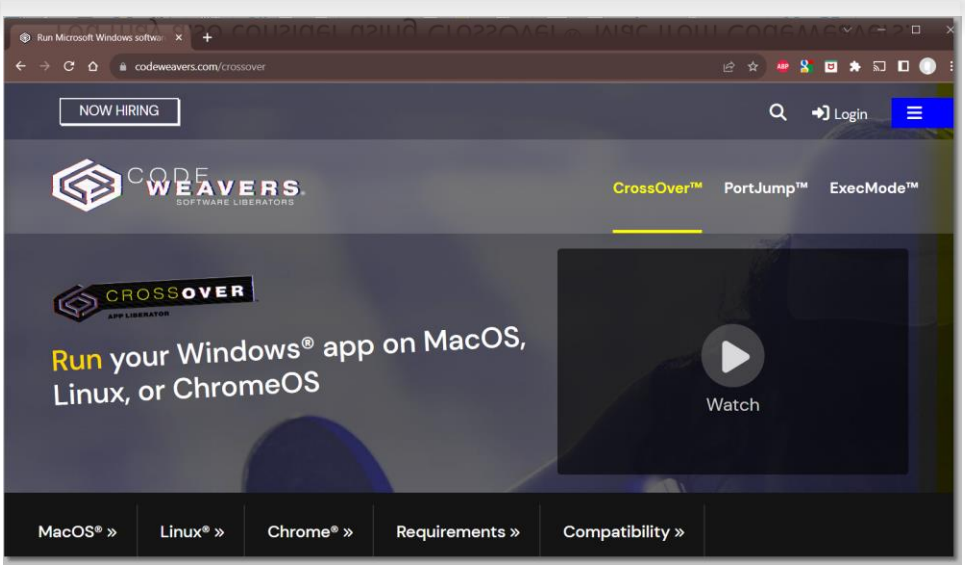
While it's working, Homebrew displays messages and progress bars in the Terminal. When Homebrew's installation is complete, it stops displaying messages and waits for a new command.

Sources:

https://medium.com/aoftionstyle/how-to-install-brew-on-mac-macos-ba9c388c8c1a

https://www.davidbaumgold.com/tutorials/wine-mac/#requirements

You may also consider using CrossOver® Mac from CodeWeavers:

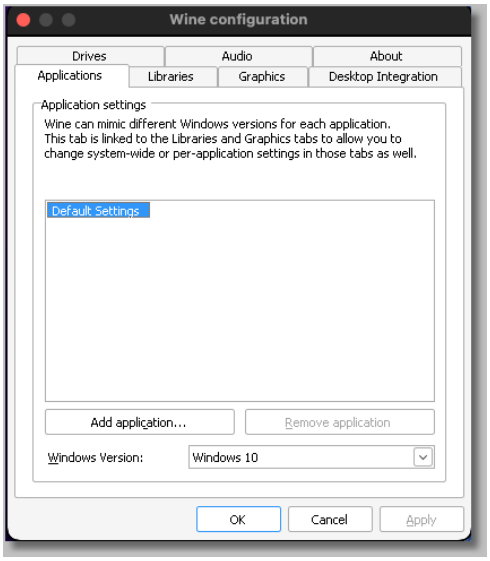


Become familiar with some Wine basics:

Full Wine documentation: https://www.winehq.org/documentation

Open a Linux/MacOS terminal/shell window and enter:

- winecfg displays the base configuration, versions, drives and other settings of the Windows environment…



- wine *windows-command* executes the Windows command specified by "*windows-command*"

Note:

To avoid displaying alert lines and disable verbose mode, use "**wine** *xxxxx* **2>/dev/null**"

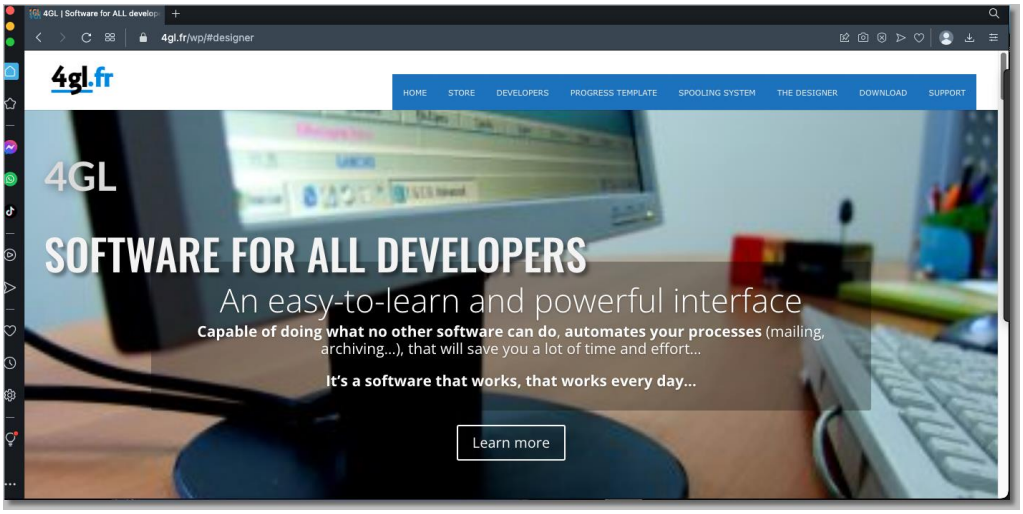
```
marcel@UBUNTU:~$ wine cmd
0080:fixme:hid:handle_IRP_MN_QUERY_ID Unhandled type 00000005
0080:fixme:hid:handle_IRP_MN_QUERY_ID Unhandled type 00000005
0080:fixme:hid:handle_IRP_MN_QUERY_ID Unhandled type 00000005
0080:fixme:hid:handle_IRP_MN_QUERY_ID Unhandled type 00000005
Microsoft Windows 10.0.18362

Z:\home\marcel>exit
marcel@UBUNTU:~$ wine cmd 2>/dev/null
Microsoft Windows 10.0.18362

Z:\home\marcel>
```

- wine cmd opens the Windows command-line.

  when the command line is open, any Windows command can be entered, as with the Windows **cmd** command.

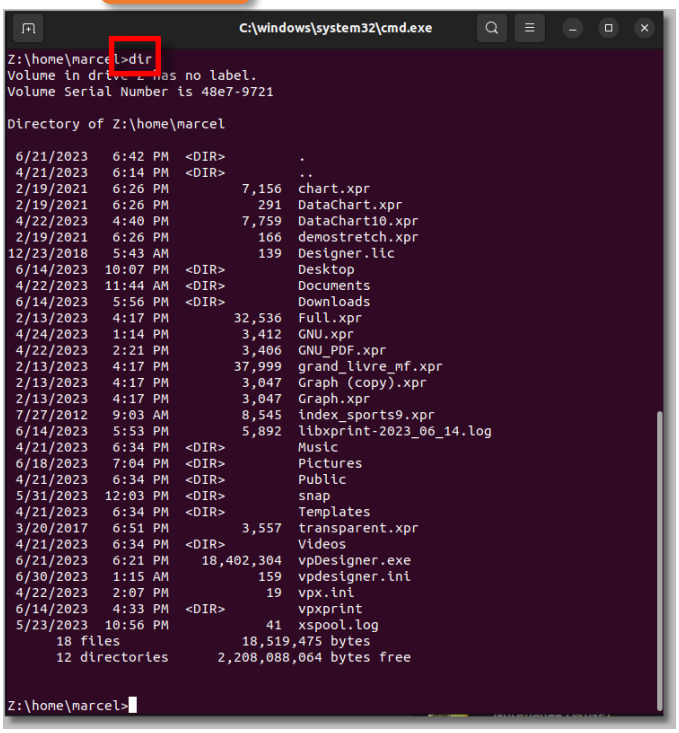
- wine winebrowser *URL* opens the specified URL (http://www.4GL.fr in this example) in the native operating system's default protocol handler.

  $ **wine winebrowser http://www.4GL.fr**

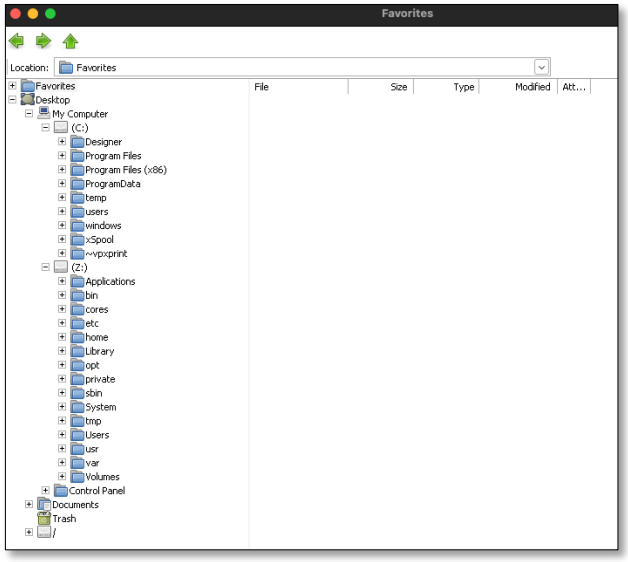


Standard Windows command: dir

Wine cmd mode, “dir” example.

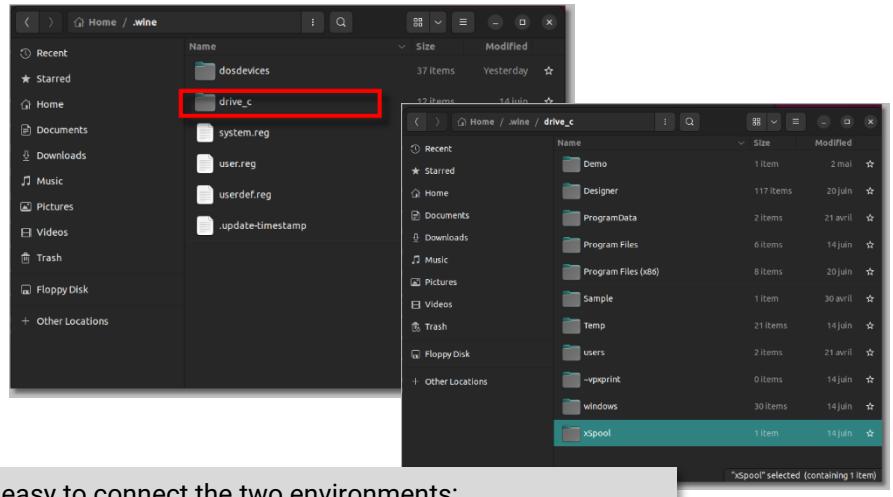


- wine explorer opens the Windows explorer window:



"Z:" refers to the Linux/MacOS host's main file system.

When Wine is installed, a "**.wine**" subdirectory is created in the user's home directory. "**.wine/drive_c**" is the Windows "C:\" directory:

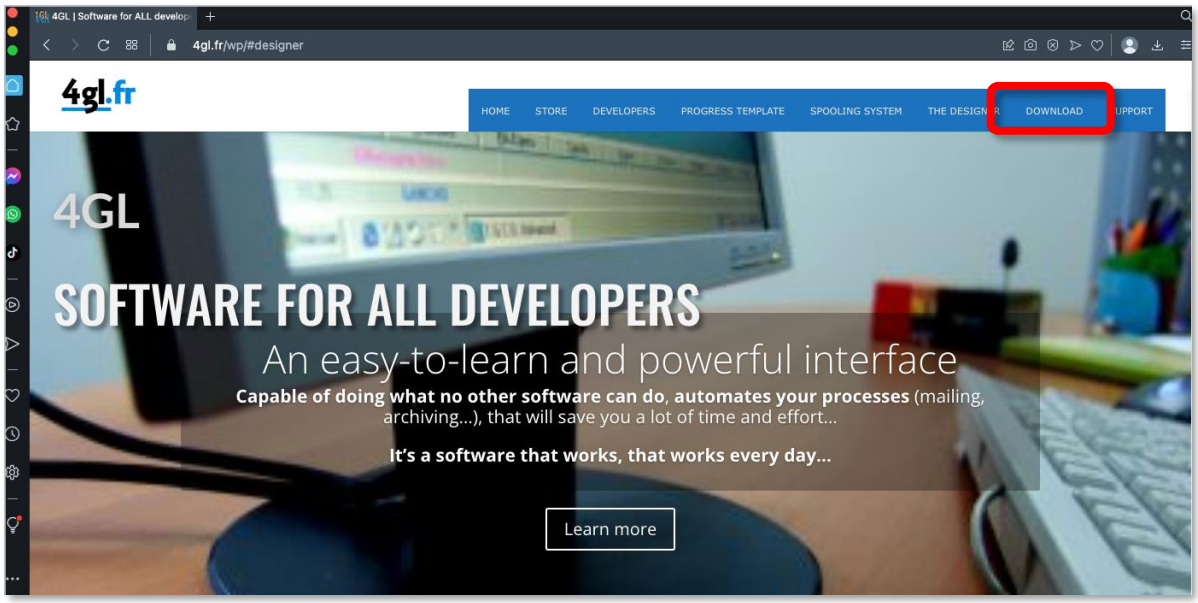


This makes it very easy to connect the two environments:

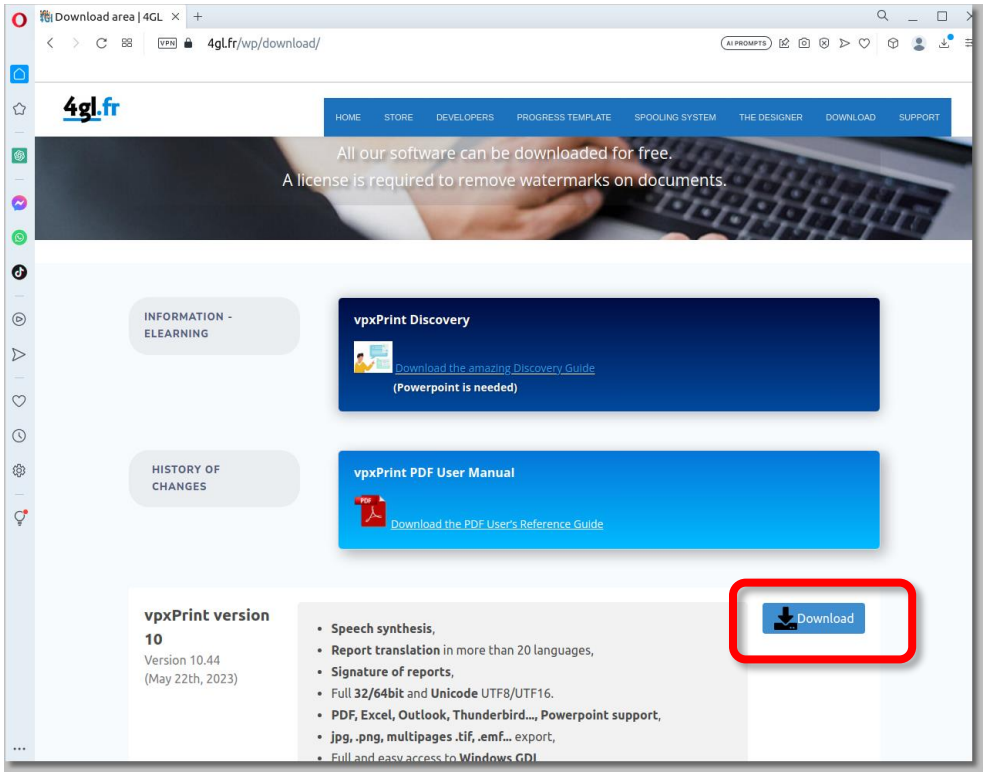
- From Windows, "**Z:**" is to the Linux/MacOS system files,
- From Linux/MacOS, *~home* "**/.wine/drive-c**" is the Windows C drive.

vpxPrint setup

1. Open the web site http://www.4GL.fr

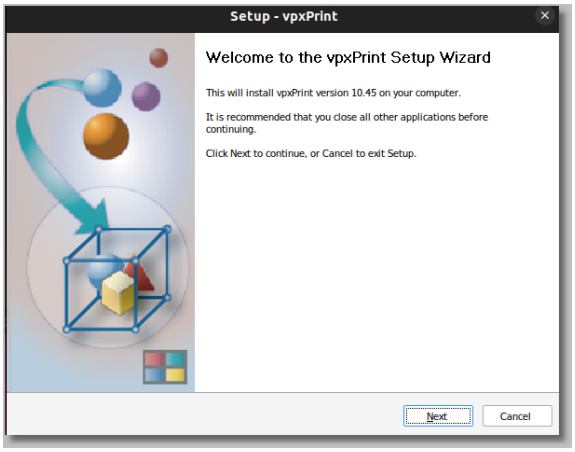


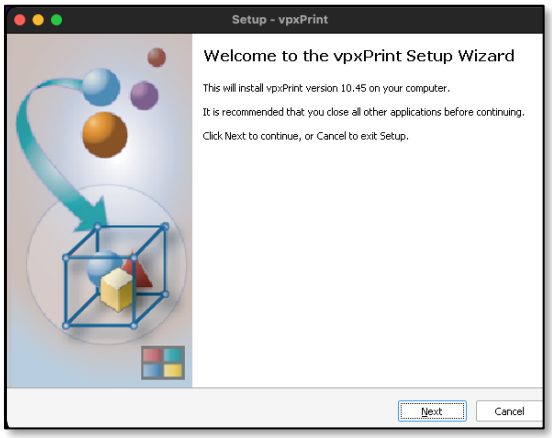
2. Go to the download area:

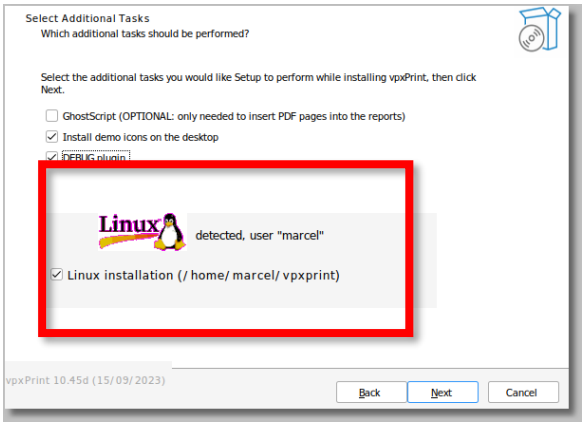


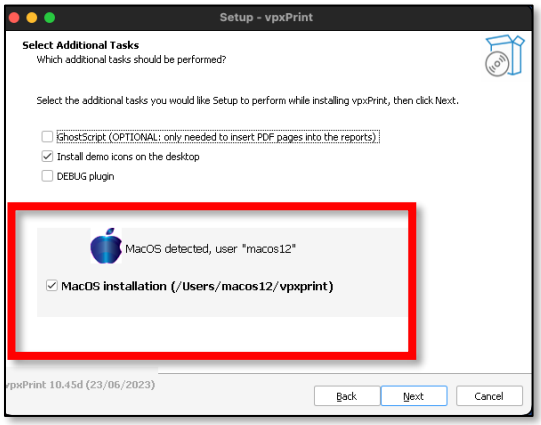
3. $ **wine vpxprint10_45.exe**

Linux	MacOS









The vpxPrint setup detects the host system and the current user, it then installs corresponding Linux or MacOS demos and interfaces to *homedir*/**vpxprint**

```
marcel@UBUNTU:~$ cd vpxprint
marcel@UBUNTU:~/vpxprint$ ls -l
total 7456
-rwxrwxr-x 1 marcel marcel 6215344 juin  13 20:23 callxprint
-rw-rw-r-- 1 marcel marcel     617 mai   11 21:18 fromC.cpp
-rwxrwxr-x 1 marcel marcel   17184 mai    7 12:37 fromCpp
-rw-rw-r-- 1 marcel marcel    2726 mai   31 14:29 fromPython.py
-rw-rw-r-- 1 marcel marcel    3346 juin   9 16:50 GNU.xpr
-rw-rw-r-- 1 marcel marcel 1277928 juin  13 20:23 libxprint.so
-rw-rw-r-- 1 marcel marcel   83323 mai   11 20:49 linux_logo.png
-rw-rw-r-- 1 marcel marcel    1230 juin  12 20:36 loadxPrint.py
drwxrwxr-x 2 marcel marcel    4096 juin  10 10:21 __pycache__
-rw-rw-r-- 1 marcel marcel    2043 juin  11 20:30 toxSpool.py
-rw-rw-r-- 1 marcel marcel    2072 mai   31 13:42 xprint.py
-rw-rw-r-- 1 marcel marcel     108 juin  14 14:35 xspool.sh
marcel@UBUNTU:~/vpxprint$
```

```
/Users/macos12
macos12@macos12s-MacBook-Pro ~ % cd vpxprint
macos12@macos12s-MacBook-Pro vpxprint % ls -l
total 15784
-rw-r--r--  1 macos12  staff     3346 Jun  9 16:49 GNU.xpr
drwxr-xr-x  7 macos12  staff      224 Jun 13 18:03 XCode
drwxr-xr-x  3 macos12  staff       96 Jun 12 13:13 __pycache__
-rwxr-xr-x@ 1 macos12  staff  6186608 Jun 13 20:23 callxprint
drwxr-xr-x  3 macos12  staff       96 Jun 11 23:09 callxprint.app
drwxr-xr-x  3 macos12  staff       96 Jun 13 17:56 dyncall
-rw-r--r--  1 macos12  staff      617 May 11 21:18 fromC.cpp
-rw-r--r--  1 macos12  staff     2726 May 31 14:29 fromPython.py
-rwxrw-rw-  1 macos12  wheel  1844328 Jun 13 20:24 libxprint.dylib
-rw-r--r--  1 macos12  staff     1236 Jun 12 20:36 loadxPrint.py
-rw-r--r--  1 macos12  staff     5774 May 23 17:23 macos_logo.png
-rw-r--r--@ 1 macos12  staff     2345 Jun 13 17:45 main.swift
-rw-r--r--  1 macos12  staff      134 Jun  9 20:26 readme.txt
-rw-r--r--  1 macos12  staff     2043 Jun 11 20:31 toxSpool.py
-rw-r--r--  1 macos12  staff     2072 May 31 18:17 xprint.py
-rw-r--r--  1 macos12  staff       95 May 23 17:23 xspool.sh
macos12@macos12s-MacBook-Pro vpxprint %
```

Running vpxPrint

There are two basic ways to use vpxPrint on a Linux or macOS computer:

1. **Call xprint directly:**

   We provide the native equivalents of xprint.dll in the vpxprint subdirectory:

   libxprint.so

   libxprint.dylib

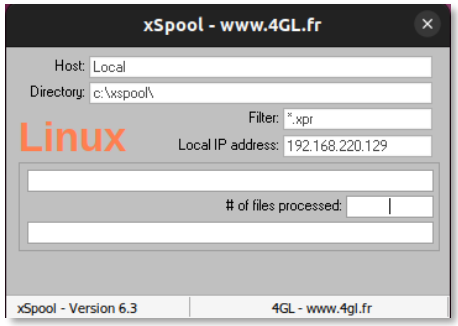
   These dynamic libraries connect transparently to vpxPrint (via Wine) and have the following entry points:
   - printFile,
   - printFileStat,
   - xPrintVersion,
   - getMailResult

   Various examples of use in C++, C#, Pascal, Python, Xcode are provided in the home/vpxprint subdirectory, see above.

2. **xSpool, the easy way:**

   Simply launch xSpool with "sh xspool.sh" (adjust parameters to meet your needs)

```
wine xspool -dir=c:/xspool -logfile=c:/temp/xspool.log "-prefix=*.xpr" -interval=2 &>/dev/null &
```



With these settings, .xpr files must be put in *home*/**.wine/drive_c/xspool** folder.
But you may also specify "-dir=Z:/home/xspool" to refer to yor own OS native files system.

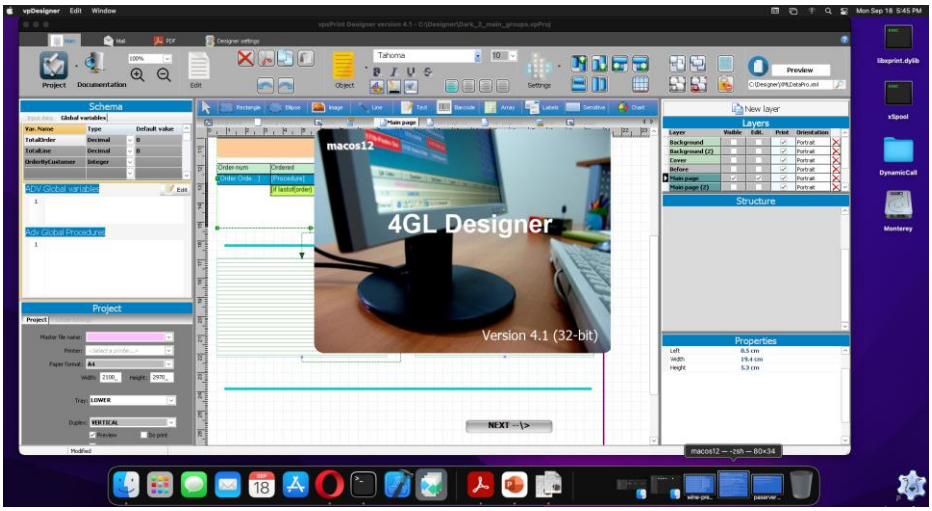
xSpool offers two major advantages:

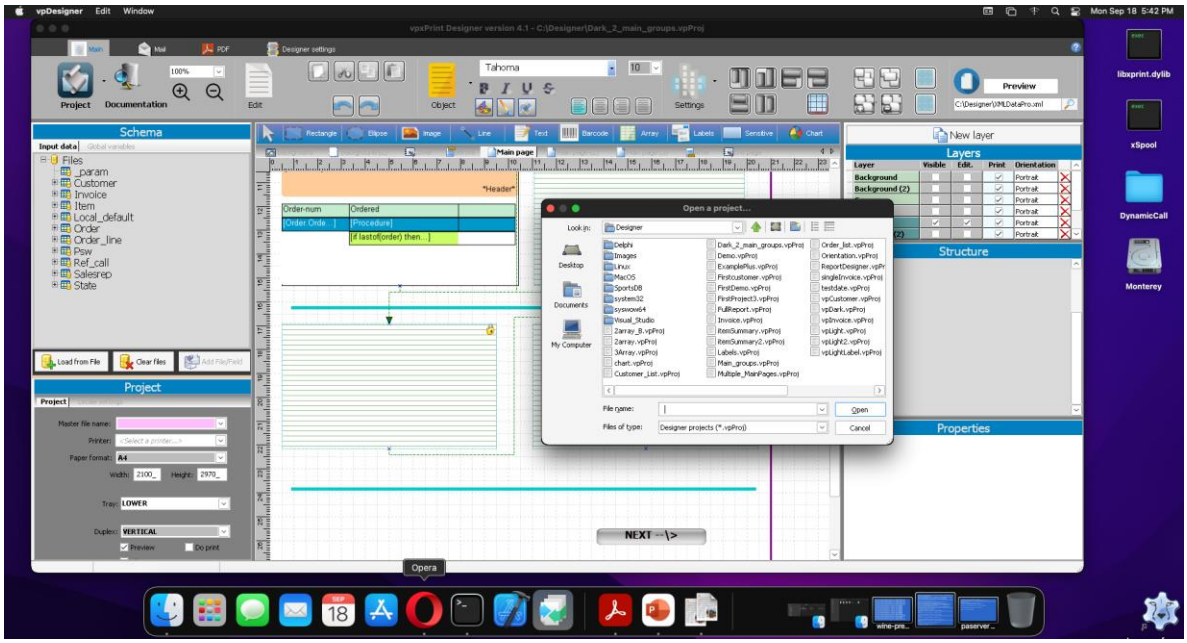
i. You only need to create text files to launch xPrint, which is very easy to implement.
ii. Since the wine version of xSpool calls wine's xprint.dll, it won't reload the dll in memory for each file.

The Designer

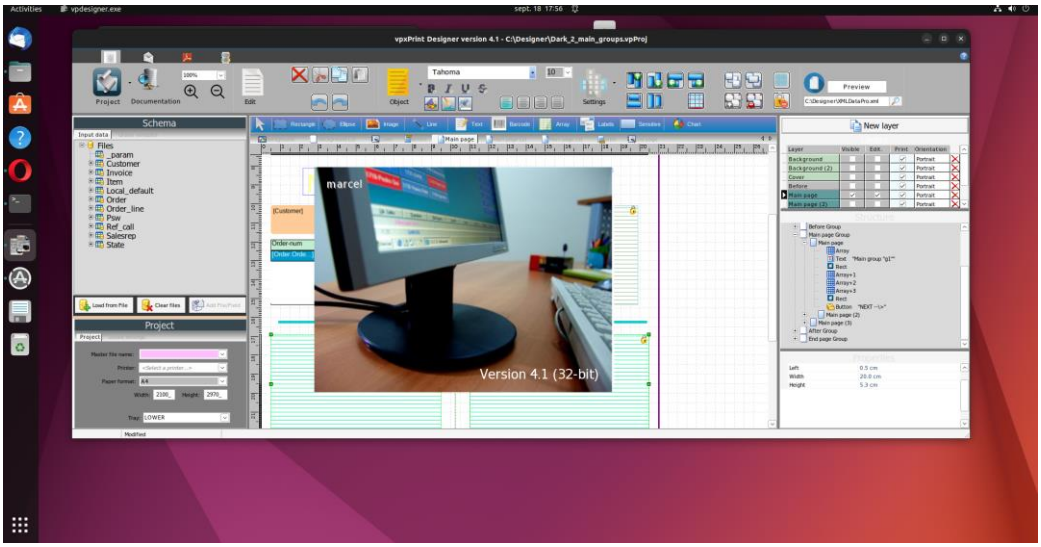
The Designer is free with vpxPrint, download it from the www.4GL.fr and install it. This will create a subdirectory **vpdesigner** in the current home directory.

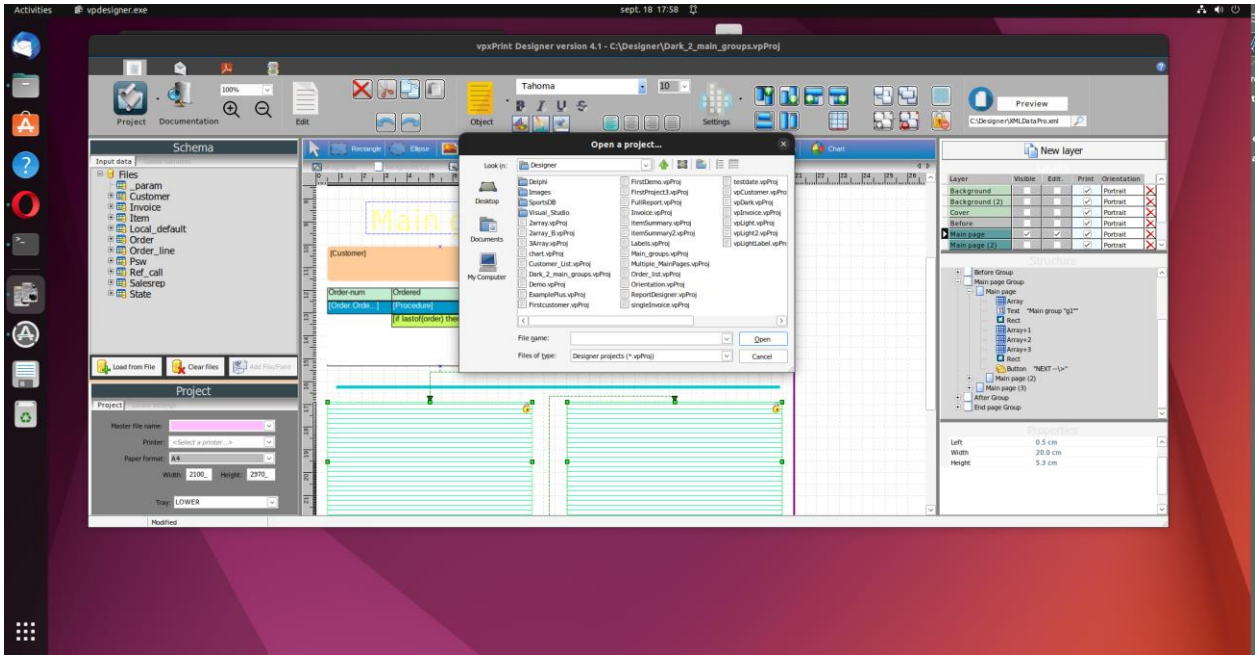
MacOS Designer: *home*/vpdesigner $ **sh vpDesigner.sh**





Linux Designer: *user@Ubuntu:~/vpdesigner* $ **sh vpDesigner.sh**



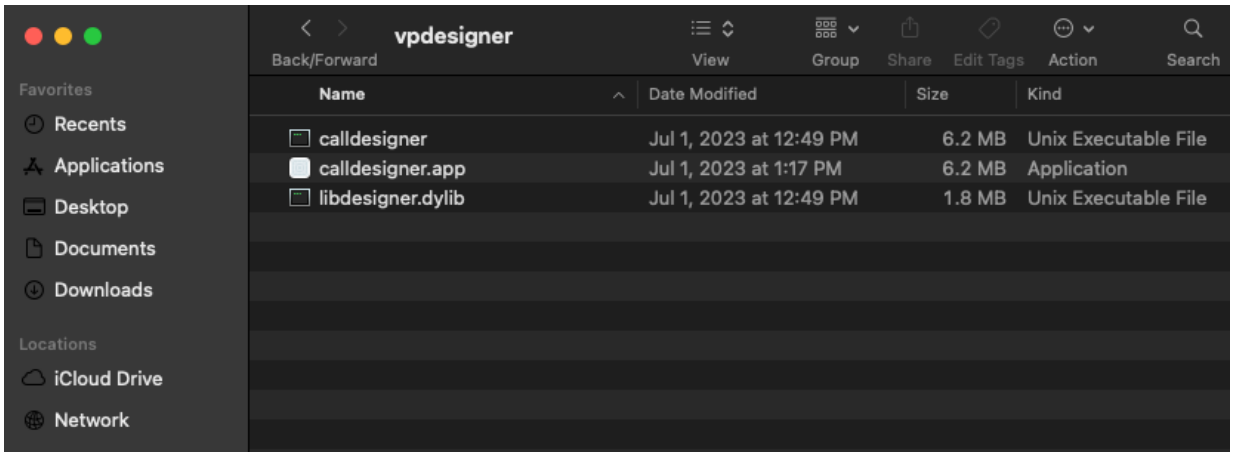


Running The Designer

The designer comes with the following libraries:

libdesigner.so

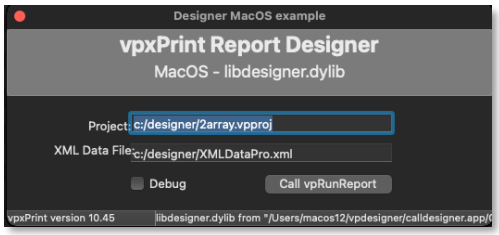
libdesigner.dylib

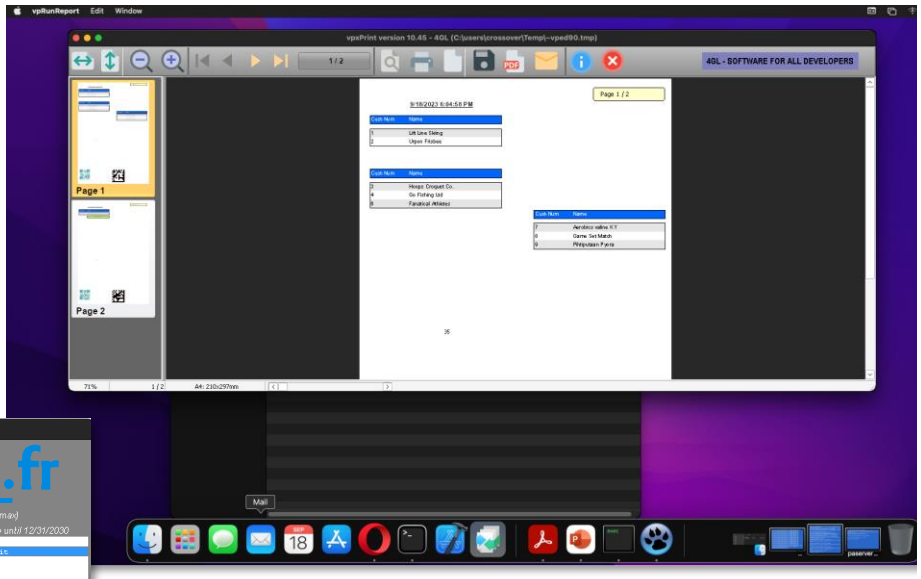


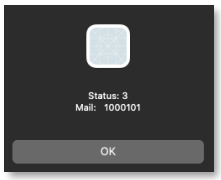
... with the following entry points:

- function **vpRunReport**(Project_File, XML_DataFile, xprname: string): **integer;**
  Returns the vpxPrint return status
  xprName = .xpr file name. If set to "", a name is automatically generated.
- function **getMailResult**(): **integer**;
  Returns the Mail result.
- procedure **vpRunDebug**(iDebug: integer);
  1 or 0 to set/reset debug mode

The program **calldesigner** in a MacOS environment:







Return status