science + computing

A Bull Group Company

# Building an Advanced Python Installation for Linux and Windows

**Anselm Kruis | EuroPython 2012**

**science + computing ag**

IT-Services for Complex Computing Environments

Tübingen | München | Berlin | Düsseldorf

# Building an Advanced Python Installation

science + computing
A Bull Group Company

## Outline

- About me
- Outset
- Linux
  - Problems
  - Solutions
- Windows
  - Problems
  - Solutions

# Who and Why

science + computing
A Bull Group Company

## Who

Name: Anselm Kruis

Profession: Senior Architect at science + computing ag

Location: Munich

## Why

- Python is fun, EuroPython is fun
- Let's do some cool stuff
- Cool stuff, that isn't used, doesn't matter
- Make your programs usable!

# Outset

science + computing
A Bull Group Company

- Spring 2010: start of a new project
- Stackless Python 2.x, PyGTK, lxml, ...
- Computers
  - Office PCs
  - Large HPC cluster (>10000 cores)
- Operating systems:
  - Linux x86_64, various distributions. Oldest RHEL4
  - Windows 32 and 64bit, starting with XP SP3
- Code server based installation

# Requirements

science + computing
A Bull Group Company

- Only two architecture dependent packets: Linux, Windows
- Zero installation
- Fully relocatable
- Usable and maintainable for more than 10 years
- Reliable
- Wrap scripts with executables

  `fg2start` instead of `python fg2start.py`

# Requirements (Details)

science + computing
A Bull Group Company

- Architecture packets for
  - Windows 32bit starting with XP SP3
  - Linux x86_64 distributions with glibc 2.3.4 or later
    (RHEL 4 and up, SLES 10 and up, Debian).
- Zero installation
  - No dependency on any component, that is not distributed with the operating system
- Relocatable
  - Runs from any directory in the file system tree.
- Usable and maintainable for more than 10 years.
  - Compile everything ourself
  - Ability to fix bugs: know-how, license issues, cost
- Reliability
  - Don't use undocumented features.
  - Adhere to standards (i.e. Python, Posix, Microsoft) wherever possible
- No scripts
  - Wrap every script with a real executable.

# Overall Approach

science + computing
A Bull Group Company

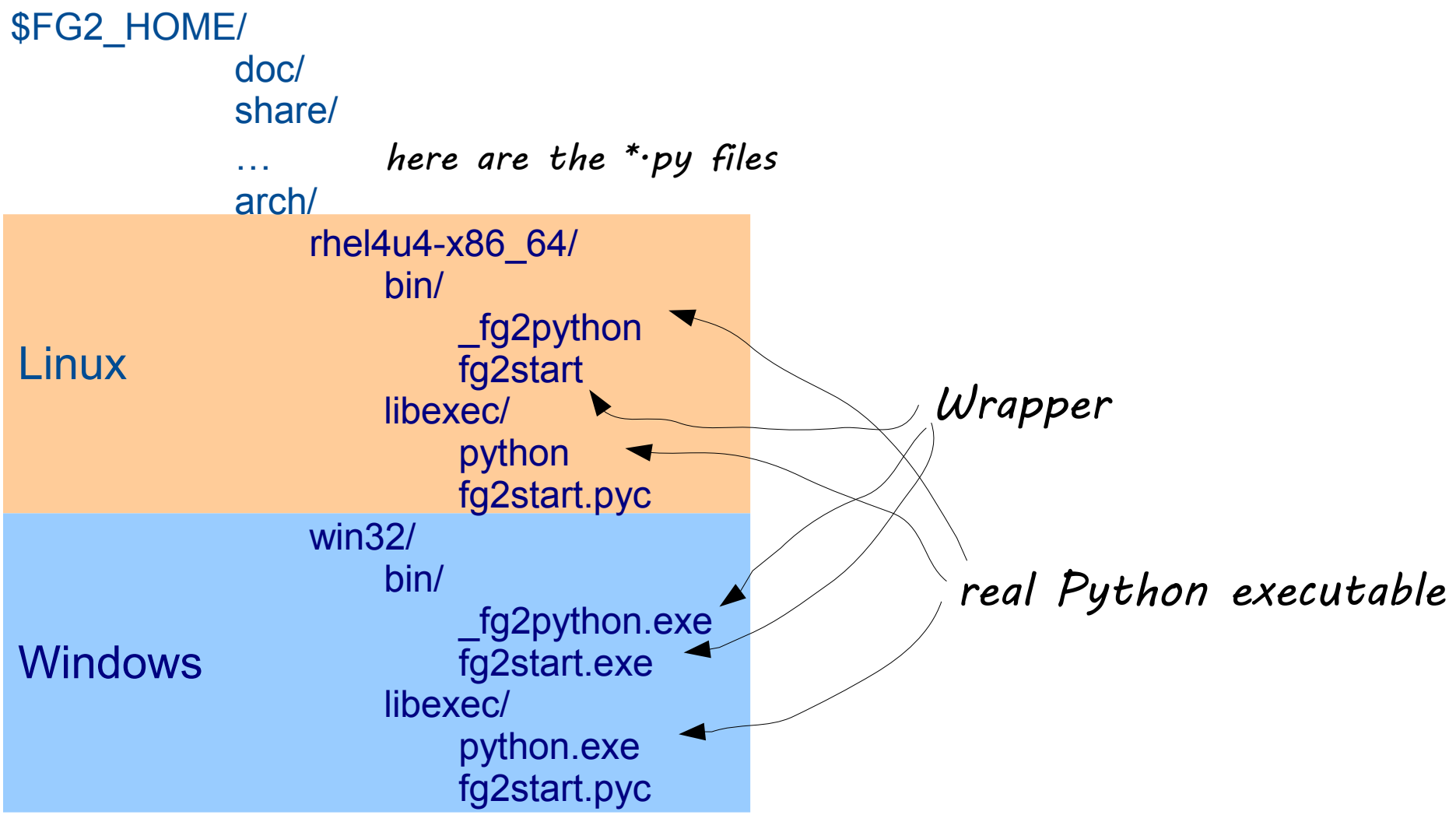
- Existing tools and projects didn't fit
  - I didn't know about PyRun in 2010
- Our solution
  - Targeted to our needs
  - Well understood
  - Maintainable
  - A lot of work

# Overall Approach

science + computing
A Bull Group Company

- Layout
  - One pure Python packet
    - Py-files, data-files, configuration, documentation, ...
    - Always installed
  - Two architecture dependent packets
    - Provide:
      - Python + compiled extensions
      - Wrapper for Python scripts
    - Installed as needed
    - Reusable for other projects

# Overall Directory Layout



# Python on Linux

science + computing
A Bull Group Company

## Typical software installation:

```
$ configure --prefix=/... && make && sudo make install
```

## Resulting installation does not match our requirements

- The installation heavily depends on the installed libraries / development packages.
  - `configure` auto detection of libraries
  - library symbol versioning
- --prefix path in
  - ELF-attribute DT_RUNPATH, aka “rpath”
  - compiled into binaries via cpp defines
  - generated configuration files

# Critical Success Factors

science + computing
A Bull Group Company

- **Reproducible, well defined build process**
- **Relocatable installation = can be installed anywhere**
- Script wrapper

# Reproducible Building on Linux

science + computing
A Bull Group Company

## Use a chroot build environment !

- Keep your development system current and secure
- Most Linux distributions provide a suitable chroot build environments
  - Fedora: mock
  - SuSE: build
  - Debian: pbuilder
- For precise control and customization
  - Use a local package repository
    - Speed up
    - Ability to add / remove / modify packages

- Search Google for “chroot build environment”

# Relocatable Software on Linux

science + computing
A Bull Group Company

File access happens

- During startup of an executable
  - Runtime linker ld.so locates shared libraries
- At runtime
  - The application uses files

# Relocatable Software on Linux

science + computing
A Bull Group Company

Startup: Runtime linker ld.so locates

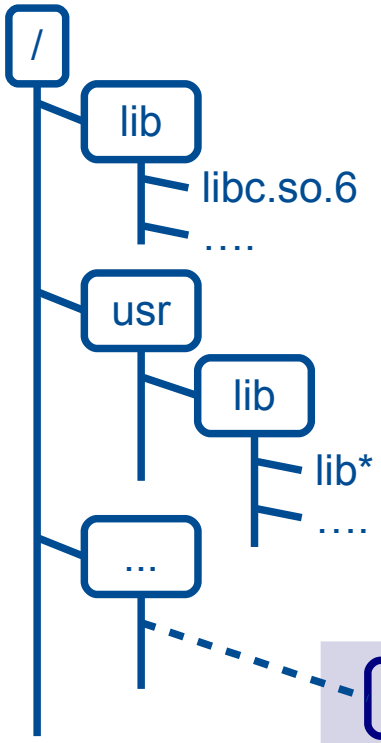
- Shared system libraries: /etc/ld.so.conf
- Private shared libraries
  - Environment variable LD_LIBRARY_PATH
  - rpath

# Linux RPATH

science + computing
A Bull Group Company

- Executables or shared libraries can contain a search path for shared libraries they depend on
  - A feature of the ELF file format and the runtime linker
  - Usually set at link time. ld option -rpath
  - Utility patchelf can set it
- Within RPATH entries “$ORIGIN” means the directory containing the executable or shared library
- $ORIGIN not supported by autoconf / automake / libtool
  - Hacking the build system is no fun → $ORIGIN is rarely used
- To set RPATH entries for a complete application use the script set_relative_rpath.py

# Script set_relative_rpath.py




- Get it from https://github.com/akruis/advancedPythonInstallation
- Create a file with all system lib dirs

```
ldconfig -N -v | sed -n -e 's,^\(/[^:]*\).*,\1,p' \
    >systemlibdirs
```

- Compile and install your software (i.e. python) as usual

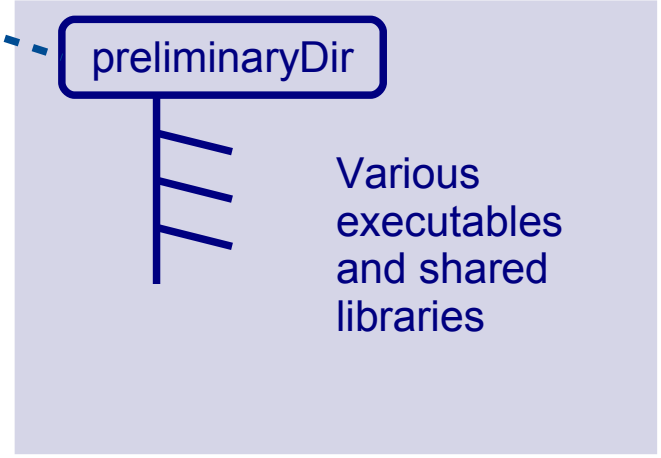
```
configure –prefix /.../preliminaryDir && make install
```

- Set LD_LIBRARY_PATH as needed to locate private libraries

```
export LD_LIBRARY_PATH=/.../preliminaryDir/lib:....
```

- Set required RPATH entries within “preliminaryDir” subtree

```
python -u set_relative_rpath \
-c systemlibdirs \
-n -w '/.../preliminaryDir'
```



# Relocatable Software on Linux

science + computing
A Bull Group Company

- Startup: Runtime linker ld.so locates shared libraries

  …

- Runtime: The application locates files
  - How to make it relocatable ?
    - Environment Variables
    - Config Files
    - Patches

  - For Python extension modules
    Use sitecustomize.py to set environment variables
    - use os.putenv to preserve os.environ unmodified
    - monkey patch subprocess to use os.environ by default

# Patches

science + computing
A Bull Group Company

Sometimes you need a patch to make a program relocatable

- Push it upstream
- Follow established standards
  - XDG Base Directory Specification
  - GTK Environment http://developer.gnome.org/gtk/stable/gtk-running.html
- Our Patches for PyGTK
  - Pango: https://bugzilla.gnome.org/show_bug.cgi?id=454017
    (Committed since 2012-03-17)
  - GVFS: https://bugzilla.gnome.org/show_bug.cgi?id=678697
  - GDK-Pixbuf: https://bugzilla.gnome.org/show_bug.cgi?id=678703
  - Glade: https://bugzilla.gnome.org/show_bug.cgi?id=678707

# Generic Wrapper

science + computing
A Bull Group Company

- Wrapper is written in C
  - This way it can be used as a script interpreter
- Takes its own name as the name of a python script to execute
- Mostly equivalent to the following shell code

```
#!/bin/sh
exec `dirname $0`/../libexec/python \
 $OPTIONS_FOR_PYTHON \
 `dirname $0`/../libexec/`basename $0`.pyc -- "$@"
```

Any Questions ?

Let's proceed to

Windows 32bit

# Situation compared to Linux

science + computing
A Bull Group Company

- Building the software is much harder
- Relocation is usually no big problem
- The DLL hell is awaiting you
- Sometimes things working on Linux don't work on Windows
  - Example: wrappers

# Critical Success Factors

science + computing
A Bull Group Company

- **Reproducible, well defined build process**
- Relocatable Installation = can be installed anywhere
- **Script wrapper**

# Building on Windows

science + computing
A Bull Group Company

- Tool chain issues
  - Compiler
  - C-runtime library
- DLLs
  - Where to install private DLLs ?

# Building on Windows - Compiler

science + computing
A Bull Group Company

- Python 2.7 uses Visual Studio 2008 by default
  - C-runtime: usually msvcr90.dll
- Many libraries require MinGW / MSYS
  - UNIX style build environment
  - C-runtime: usually msvcrt.dll
- Mixing compilers is not without problems
  - Compiler specific C-runtime library
  - Compiler specific debug information

→ Problems are waiting

# Windows: C-Runtime

science + computing
A Bull Group Company

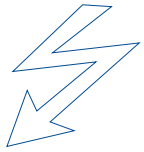
## Selection of the C-runtime

- Do not care: mix msvcrt.dll and msvcr90.dll
- Only msvcrt.dll
- Only msvcr90.dll

# Which C-Runtime ?

science + computing
A Bull Group Company

- Do not care, mix both DLLs
  - Building the software is fairly simple
  - The official binaries at ftp.gnome.org do it
  - It is discouraged by Microsoft
    http://msdn.microsoft.com/en-us/library/ms235460%28v=vs.90%29
  - You application may break if you change the compiler for a single library
  - Debugging is hard: No debugger supports both formats



- Only msvcrt.dll
  - Trivial with MinGW or Visual Studio 6
  - Visual Studio 2008 + WDK
    - Fairly simple, see
      http://developer.berlios.de/devlog/akruis/2012/06/03/msvcrtdll-and-visual-studio/
    - No precompiled extension modules
    - Debugging is difficult
    - Y2038 issues

# Which C-Runtime ?

science + computing
A Bull Group Company

## Only msvcr90.dll

- Trivial with Visual Studio 2008
- MinGW
  - Tedious setup of build environment

    http://developer.berlios.de/devlog/akruis/2012/06/10/msvcr90dll-and-mingw/
    - Changes in short
      - GCC spec-file hacks
        - Link msvcr90.dll
        - Add: manifest
        - Add: empty invalid parameter handler
      - Rebuild MinGW-runtime to use msvcr90.dll
  - MSYS is slow
  - Cross compiling on Fedora 16 is fast and works fine
    - Many MinGW packages: GTK, libxml, libxslt, ...

# Windows DLL Loading

science + computing
A Bull Group Company

- Windows looks for DLLs in the directories named by PATH
  - If you add a directory containing DLLs to PATH
    - A different application could load your DLLs
  - If you locate DLLs via PATH
    - You could get foreign DLLs

  → **Do not place a DLL besides an executable, if the executable is going to be located via PATH**

- But where to place private DLLs ?
  - Use a manifest and place the DLL in a subdirectory
  - Use a wrapper for the executable
  - Runtime DLL loading only: SetDllDirectory

```
import ctypes
ctypes.windll.kernel32.SetDllDirectoryW(unicode(dir))
```

# About Manifests – DLL Loading

science + computing
A Bull Group Company

- Application Manifest (within *.exe)

```
<assembly xmlns='urn:schemas-microsoft-com:asm.v1' manifestVersion='1.0'>
  <dependency>
    <dependentAssembly>
      <assemblyIdentity type='win32' name='myorg.python.dlls' version='2.7.3.0'/>
    </dependentAssembly>
  </dependency>
</assembly>
```

  You can use mt.exe from SDK to change the embedded manifest of an application

- Assembly
  - Directory layout

    \python.exe
    \myorg.python.dlls\myorg.python.dlls.MANIFEST
    \myorg.python.dlls\python27.dll

  - myorg.python.dlls.MANIFEST

```
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
    <assemblyIdentity type="win32" name="myorg.python.dlls" version="2.7.3.0"/>
    <file name="python27.dll" />
</assembly>
```

# About Manifests – UAC

science + computing
A Bull Group Company

## User Account Control

- http://msdn.microsoft.com/en-us/library/windows/desktop/bb756929.aspx
- To avoid the UAC prompt add

```
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
  <security>
    <requestedPrivileges>
      <requestedExecutionLevel level="asInvoker" uiAccess="false"/>
    </requestedPrivileges>
  </security>
</trustInfo>
```

# Relocatable Software on Windows

science + computing
A Bull Group Company

See Linux

# Windows Wrapper

science + computing
A Bull Group Company

- UNIX: wrapper uses execve.
  - Only 1 process, a single PID, good
- Windows lacks the system call execve
  - Wrapper spawns python and waits for the results.
  - Kill-problem: Python needs to monitor the wrapper
    - Wrapper adds an inheritable handle to itself to the environment
    - Python creates a non inheritable handle, then waits for the handle to get signaled and terminate itself using a daemon thread.
  - Example
    - See https://github.com/akruis/advancedPythonInstallation directory “winWrapper”

# Windows Import Performance

science + computing
A Bull Group Company

- We got a complaint: “it takes 2 minutes to start the GUI”
- A trace showed: stat() calls for non existing files on a CIFS file-server are fairly slow
- “import” does an awful lot of stats: 4 for each directory
- PyPi package http://pypi.python.org/pypi/quickimport
  - Caches directory content and avoids many stat() calls
  - Does not require changes to the application
  - We got a factor 2 speed up

# Conclusion

science + computing
A Bull Group Company

## Building an Advanced Python Installation

- is possible
- is takes a lot of time
- is required for certain Python based products

## Open Questions

- How to provide it to the public?
- Is there demand for more work in this area?

# Acknowledgements

Thanks to

- Arno Steitz

  for approving the publication of code and know how

- Michael Bauer

  for writing set_relative_rpath.py

# Links

science + computing
A Bull Group Company

Build Tools

- https://github.com/akruis/advancedPythonInstallation

Windows C-Runtime Hacks

- http://developer.berlios.de/devlog/akruis/2012/06/03/msvcrtdll-and-visual-studio/
- http://developer.berlios.de/devlog/akruis/2012/06/10/msvcr90dll-and-mingw/
- http://kobyk.wordpress.com/2007/07/20/dynamically-linking-with-msvcrtdll-using-visual-c-2005/

Standards

- XDG Base Directory Specification
  http://freedesktop.org/wiki/Standards/basedir-spec?action=show
- Windows Manifest
  http://msdn.microsoft.com/en-us/library/windows/desktop/aa375632%28v=vs.85%29.aspx
  http://msdn.microsoft.com/en-us/library/windows/desktop/aa375674%28v=vs.85%29.aspx

Other

- PyRun – a single file Python installation
  http://www.egenix.com/products/python/PyRun/

science + computing

A Bull Group Company

Many thanks for your kind attention.

**Anselm Kruis**
science + computing ag
www.science-computing.de

Phone +49-7071-9457-0
info@science-computing.de