Phenix documentation by method

Phenix *Python-based Hierarchical ENvironment for Integrated Xtallography*

Phenix Documentation - version dev-xxx

NEW Phenix documentation *chatbot*

Crystallography	Cryo-EM	Predicted models
How to install	GUI introduction	FAQs
Video tutorials	Tutorials & Examples	Overview of programs

Other links

Phenix website
Discussion
Bugzilla
Code & program of reviewer-test

Citing Phenix

If you publish using Phenix (the

Macromolecular structure determination using X-rays, neutrons and electrons: recent developments in Phenix. P. Liebschner, N. J. Urzhumtsev, G. Bunkóczi, W. L. Baker, J. Borelis, A. A. Chen, T. I. Croll, R. Grosse-Kunstleve, L.-W. Hung, T. Ioerger, A. J. McCoy, N. W. Moriarty, R. D. Oeffner, B. K. Poon, M. G. Prisant, A. T. Brunger, N. S. Sauter, C. M. Terwilliger, P. D. Adams, T. C. Terwilliger. *Acta Cryst. D* (2019). 75, 861-877.

Phenix. Python-based Hierarchical ENvironment for Integrated Xtallography. P. D. Adams, P. V. Afonine, G. Bunkóczi, V. B. Chen, I. W. Davis, N. Echols, J. J. Headd, L.-W. Hung, G. J. Kapral, R. W. Grosse-Kunstleve, A. J. McCoy, N. W. Moriarty, R. Oeffner, R. J. Read, D. C. Richardson, J. S. Richardson, T. C. Terwilliger, P. H. Zwart. *Acta Cryst. D* (2010). 66, 213-221.

Phenix: a comprehensive python-based system for macromolecular structure solution. P. V. Afonine, R. W. Grosse-Kunstleve, N. Echols, J. J. Headd, N. W. Moriarty, M. Mustyakimov, T. C. Terwilliger, A. J. Morris, R. Read, P. D. Adams. *Acta Cryst. D* (2012). 68, 352-367.

Phenix

Python Image Analysis Framework for Empirical Research

How to rebuild PHENIX from an existing installation

Contents

- Introduction
  - Prerequisites on Linux
  - Prerequisites on MacOSX
- Build procedure
- Windows Build

Introduction

Phenix is built nightly through an automated bootstrap mechanism. The procedure is very lengthy and only necessary if you want to make changes to the underlying framework or want to make sure that you can reproduce the binaries on your own machine a posteriori whenever they do not work on a recent platform. This is not meant to be a full tutorial on how to develop for Phenix, but only a quick reminder on how to rebuild the binaries from scratch without using bootstrap. It applies to both Linux and MacOSX. Linux for Windows build is detailed.

Prerequisites on Linux

To build PHENIX from an existing installation with sources the following is required:

- gcc and g++ complete
- Python 2.7 or 2.6 (Ubuntu Natty ARM uses apt-get install libpyld-dev libwebkit-dev libwebkitgtk-1.0-dev)

Prerequisites on MacOSX

To build PHENIX from an existing installation with sources the following is required:

- Xcode development tools

Build procedure

Assuming your installation is located in /opt/cctbx/phenix change directory from a Bash shell to /opt/cctbx/phenix and run the build.sh shell script in the Bootstrap sub-directory:

```
cd /opt/cctbx/phenix
./bootstrap/build.sh
```

This has the effect of deleting all files and directories from the build directory except for the config_bootstrap.sh file. Now the configure.sh script is run from the bootstrap sub-directory:

```
./config_bootstrap.sh
```

This will prepare the build of Phenix using the python and necessary base components shipped with your Phenix installation. Now run the compiler:

```
./build_compiler.sh
./build_base.sh
```

where NCPU is the number of cpu cores you want to use for the build. The build should take less than an hour on a recent machine.

Windows Build

The whole process is much the same for building on Windows systems. Click here for further details.

Windowsbuild

How to build PHENIX on Windows

Contents

- Introduction
- Necessary checks of the Windows PC
  - Automatically run script for every command prompt
  - Run a script once at every login
- Necessary prerequisites
  - Run a script once
  - Visual C++ 9.0
  - Cygwin
- Run scripts to automatically together as zip files
- Manual build
  - From an existing PHENIX installation
  - From bootstrap.py
  - Automated build with Buildbot
  - Issue defining environment variables for compiler
  - Creating an executable Windows installer program
- Important tweaks for shortening duration of builds

Introduction

This document describes how to build Phenix from sources on Windows 7 or more recent versions. Given that, Windows 7 is the oldest version of Windows still supported by Microsoft, it is the minimum version required to build PHENIX.

Installing PHENIX installation takes up to a total of 20 GB disk space. If you build PHENIX from sources including or rebuilds the GUI all the more of 20GB. Throughout this document the "command prompt" (sometimes "cmd.exe") refers to the Windows command prompt. Unless otherwise specified, all commands are to be typed into the same command prompt. Unless otherwise specified, the Windows Explorer is invoked with the "Windows" group.

Necessary checks of the Windows PC

Automatically run script for every command prompt

Some of the following steps require to run a few commands automatically at every new command prompt, environment variables for a user. Windows has a different technique for achieving the same for the command prompt. In the following we describe the steps for Windows 7 and later.

HKCU\Software\Microsoft\Command Processor\AutoRun

If there is a AutoRun key present under the "Command Processor" key then create a new one of type string (REG_SZ) and set the value to the full path to a batch file. Unless otherwise specified, the batch file should be called autorun.cmd in the current directory. The batch file should contain the commands that are to be executed automatically at every new command prompt. Unless otherwise specified, the batch file should be created in the current directory.

> **Warning**
>
> Modifying the registry bears a risk of rendering a unusable system. Unless you know Windows should always backup your hive to search "System Backup" point in your Windows backup. This will allow you to restore the registry to a previous state in case something goes wrong.

Alternatively one can also define environment variables through the Control Panel\All Control Panel Items\System\Advanced system settings\Environment variables. Unless otherwise specified, the environment variables are assumed to be defined on the PC for every user. This is the case when installing Microsoft Visual Studio 2008 or Cygwin.

Run a script once at every login

Run the batch script of the shortcut generated by the %HOMEDRIVE%%HOMEPATH%\Desktop\Windows Explorer\Windows.lnk shortcut. Unless otherwise specified, the batch file should be called startup.cmd in the current directory. Add the shortcut to "Startup" folder "C:\Users\Public\Start Menu\Programs\Startup". The "Run once" key can not be used as it only runs once after a reboot.

Necessary prerequisites

To build PHENIX from sources and create a binary Windows installer the following tools are required:

- Perforce P4 (any version)
- Cygwin (any version)
- Microsoft Visual C++ 9.0 / Microsoft Visual Studio 2008
- Microsoft Windows Installer XML toolset
- Nullsoft Installer
- 7zip
- Clean-exe (optionally for duplicating setup.exe & file and concrete)

To build PHENIX from sources and create a binary Windows installer the following tools are required:

- Microsoft Visual C++ 9.0 / Microsoft Visual Studio 2008

Visual C++ 9.0

If you have not purchased Microsoft Visual Studio 2008 the compiler is available for free in the Windows 7 SDK (www.bytes.com/topic/net/insights/799903-how-install-windows-7-sdk-windows-vista-visual-studio-2008-express-edition). It should be listed "Microsoft Windows SDK for Windows 7 and .NET Framework 3.5 (VC++ compilers)" in the Control Panel\All Control Panel Items\Programs and Features. Unless otherwise specified, the batch file should be called vcvars.cmd in the current directory. Add the contents of vcvars.cmd to the cmd.exe environment variables. Unless otherwise specified, the environment variables are assumed to be defined on the PC for every user. Alternatively one can also define environment variables through the Control Panel\All Control Panel Items\System\Advanced system settings\Environment variables. Unless otherwise specified, the environment variables are assumed to be defined on the PC for every user. This is the case when installing Microsoft Visual Studio 2008 or Cygwin.

```
SET VS90COMNTOOLS=%ProgramFiles(x86)%\Microsoft SDKs\Windows\v7.0\Bin\
SET VS90COMDIR=%ProgramFiles(x86)%\Microsoft SDKs\Windows\v7.0\
SET VS90COMCMD=%VS90COMDIR%bin\vcvars32.bat
```

Set **Inspect.exe** file. When you type in if.exe also as the primitive tests in your %SYSROOT%\Cygwin\etc\setup\ directory and rebuilds the setup.exe installer. Alternatively you can begin to Microsoft Visual Studio 2008 Express Edition and set up the binaries specified by setting the file:

```
; set VS90 SDKVS_PATH
```

Just before the above 3 lines.

Makensis

Nullsoft Installer is only required for creating an executable Windows installer package for deployment of PHENIX. Nullsoft Installer is free of installation in Windows.

Cygwin

Install Cygwin(free tool) in the directory:

```
C:\Cygwin\Installation\
```

which you must add to the %PATH% environment variable. To have the tools run without interrupts the proper environment variable. Cygwin provides a batch file to set the environment variable. Unless otherwise specified, the batch file should be called cygwin.cmd in the current directory. Add the %SYSROOT%\Cygwin\bin environment variable and the contents of cygwin.cmd to cmd.exe.

```
SET SYS_ROOT=%ProgramFiles(x86)%\Microsoft SDKs\Windows\v7.0\Bin\
SET SYS_ROOT=%SYS_ROOT%;%SYSROOT%\Cygwin\bin
SET SYS_ROOT=%SYS_ROOT%;%SYSROOT%\Cygwin\usr\local\bin
```

Set **Inspect.exe** file. When you type in if.exe also as the primitive tests in your %SYSROOT%\Cygwin\etc\setup\ directory and rebuilds the setup.exe installer. Alternatively you can begin to Microsoft Visual Studio 2008 Express Edition and set up the binaries specified by setting the file:

```
; set SYS_ROOT=%SYS_ROOT%
```

Just before the above 3 lines.

Makensis

Nullsoft Installer is only required for creating an executable Windows installer package for deployment of PHENIX. Nullsoft Installer is free of installation in Windows.

7zip

Install 7zip(free tool) in the directory:

```
C:\Program Files\7zip\
```

which you must add to the %PATH% environment variable. To have the tools run without interrupts the proper environment variable. 7zip provides a batch file to set the environment variable. Unless otherwise specified, the batch file should be called 7zip.cmd in the current directory. Add the %SYSROOT%\7zip environment variable and the contents of 7zip.cmd to cmd.exe.

```
SET SYS_ROOT=%SYS_ROOT%;%SYSROOT%\7zip\bin\x86
SET SYS_ROOT=%SYS_ROOT%;%SYSROOT%\7zip\bin\x64
SET SYS_ROOT=%SYS_ROOT%;%SYSROOT%\7zip\bin\ia64
```

Run scripts to automatically together as zip files

Manual build

From an existing PHENIX installation

If you want to build PHENIX from an existing installation on Windows where the sources have also been installed (e.g. Linux):

- From the Start menu open a PHENIX command prompt
- Change directory to %PHENIX%\modules\cctbx_project\libtbx\
- Change directory to %PHENIX%\build\
  - Run libtbx.configure.bat modules all flex and dirs/exclude except ccp4io
  - Run libtbx.refresh
  - Run libtbx.scons -j<ncpus>
  - Run libtbx.scons -j<ncpus> install

From bootstrap.py

Manual build of the sources from the latest PHENIX nightly snapshot. It first downloads source files unless they are present in the current directory. Unless otherwise specified, the batch file should be called bootstrap.cmd in the current directory. Add the contents of bootstrap.cmd to cmd.exe. Alternatively one can also define environment variables through the Control Panel\All Control Panel Items\System\Advanced system settings\Environment variables. Unless otherwise specified, the environment variables are assumed to be defined on the PC.

```
PhenixBuild.cmd

:: SET PHENIX=
:: SET PHENIX_ROOT=
:: SET PHENIX_BUILD=
:: SET PHENIX_ROOT=%ProgramFiles(x86)%\Microsoft Visual Studio 9.0\VC\bin\amd64\
:: SET PHENIX_BUILD=%ProgramFiles(x86)%\Microsoft Visual Studio 9.0\VC\bin\amd64\

:: SET PHENIX_BUILD=%PHENIX_ROOT%
:: SET PHENIX_BUILD=%PHENIX_BUILD%;%ProgramFiles(x86)%\Microsoft Visual Studio 9.0\Common7\IDE\
:: SET PHENIX_BUILD=%PHENIX_BUILD%;%ProgramFiles(x86)%\Microsoft Visual Studio 9.0\Common7\Tools\

SET PHENIX_TEMPLATE=%ProgramFiles(x86)%\Microsoft Visual Studio 9.0\VC\vcproj\templates\Win32Console\Release\
:: SET PHENIX_TEMPLATE=
:: SET PHENIX_TEMPLATE=%PHENIX_TEMPLATE%;%ProgramFiles(x86)%\Microsoft Visual Studio 9.0\VC\vcproj\templates\Win32Console\Release\
:: SET PHENIX_TEMPLATE=%PHENIX_TEMPLATE%;%ProgramFiles(x86)%\Microsoft Visual Studio 9.0\VC\vcproj\templates\Win32Console\Release\
```

In the above script ideas are from http://stackoverflow.com/a/3744667 is used for giving attempt to a file as parameter to a batch file. Unless otherwise specified, the batch file should be called bootstrap.cmd in the current directory. The error code of the current last process is stored. This is useful when checking for the success of the last command. Unless otherwise specified, the error code should be checked after each command.

Automated build with Buildbot

Searching the PC runs as a Windows daemon build run automatedly from the Buildbot master checkout. Required environment variables are defined in the Buildbot master checkout. Required environment variables must be the concrete environment variables. A change of **makefiles** could not be detected alone.

Issue defining environment variables for compiler

If you get a "cl.exe cannot find the environment variable xxx" error, it means that the compiler cannot find an environment variable. Run "set" to check the environment variables and make sure that the variable is present. If the variable exists and has a relevant value, it is likely that the compiler cannot find the path of a binary tool or dll. Make sure that the path to the binary tool or dll is in the %PATH% environment variable. Alternatively one can also define environment variables through the Control Panel\All Control Panel Items\System\Advanced system settings\Environment variables. Unless otherwise specified, the environment variables are assumed to be defined on the PC.

```
:: SET PHENIX=%PHENIX%;%ROOT%
:: SET PHENIX_ROOT=%PHENIX_ROOT%;%ROOT%
:: SET PHENIX_BUILD=%PHENIX_BUILD%;%ROOT%
:: SET PHENIX_TEMPLATE=%PHENIX_TEMPLATE%;%ROOT%
```

with

```
:: SET PHENIX=%PHENIX%;%ROOT%
:: SET PHENIX_ROOT=%PHENIX_ROOT%;%ROOT%
:: SET PHENIX_BUILD=%PHENIX_BUILD%;%ROOT%;%ROOT%;%ROOT
```

Using Cost with PHENIX

Instructions for use

Cost is a stand-alone program that runs on Windows, Linux and Mac OS/Darwin. To install version of Cost 0.7 or later with Python enabled (installed as part of the PHENIX distribution) we use it as a server for mac and Linux files. Cost 0.7 and later versions are compatible with Python 2.5 and 2.6 only. If you have Cost 0.6 or earlier, you need to upgrade or to use of special standalone binaries. If not, you will need to enter the path explicitly in the Preferences dialog box.

The main interface includes a button for launching Coot, and the Window menu in most programs contains a "Cost Preferences" item. Clicking on the button or selecting the menu item opens a dialog where you can specify the path to Coot (if it is not in your path), and the default molecular graphics program (Coot or Chimera) and whether to always prepare in graphics mode. Preferences are saved in a file in your home directory.

On Windows it is better to run Cost from the Cost Log (Start menu) rather than Coot. Moreover, you should need to specify the path to the Text (command line) version of Coot and the Port (which allows the communication to Coot). The file is usually located in the Contrib directory where you installed Winnow.

Cost uses a standard protocol for communicating with other programs. Currently, the only supported communication protocol is Python-based Coot. Support other programs is easy. Please let us know if you need PHENIX to communicate with other programs. Cost will start up a Coot server if it is not already running. If you have Coot open before starting the Coot window, you probably downloaded a version without Python (see "Common problems" below for details).

To help the Cost to open, the PDB file can be viewed there by right-clicking the mapping-phase icon, and in the Coot window by selecting "Open Coot Input Coordinates" from the File menu. Alternatively, if you are using Coot in Preferences (File Settings), you may simply add the Coot .files. Most applications that output models and/or maps will also write a Coot .files file that will open the relevant files in Coot.

In the phenix utility, AutoBuild, and AutoSolve Entry, the current output and map will be automatically updated in Coot if it is open and the Coot port is specified in the Cost Preferences. If Coot is not open or if the port is incorrect, you can select model or map to reload permanent changes. In phenix.refine, the start model will be updated automatically. In AutoBuild, these automatic updates, unlike "Write progress", is disabled unless "Write progress" is checked.

Most applications output either a simple Coot.files (PE PDB FOM), other offer details (AutoBuild), or a set of Coot.files (AutoSolve). Clicking on the Coot.files icon will open the relevant files in Coot. Clicking on the Coot icon opens an empty Coot window. Clicking on the Coot Log icon opens a window showing the Coot output. Clicking on the Coot Quit icon closes the Coot window. Clicking on the Coot Restart icon closes the Coot window and starts a new Coot.

Common problems

- **Cost is installed, but nothing happens when I click the Button.** This probably means that Python is not enabled in Coot/Winnow, you will need to check the path and/or the Preferences dialog box.
- **Nothing happens when I click the "Open in Coot" button, but no model or maps appear.** Check that the port specified in Preferences matches the port specified in Coot. Check that Coot is open, and that all of the particular Contrib/bin/ directories for Coot are in your path. Check that Winnow/Coot is not a version without Python. Check that you are using Coot 0.6 or later. Check that Coot is not a version compiled without Python. Check that Winnow/Coot is not a version compiled without Python in the output.

References

1. Emsley P, Lohkamp B, Scott WG, Cowtan K. **Features and development of Coot.** Acta Cryst. D66, 486-501 (2010).
2. Kleywegt GJ, Harris MR, Zou JY. **Coot: model-building tools for molecular graphics.** Acta Cryst. D60, 2267-68 (2004).

Creating a Windows Installer for PHENIX

Contents:

- Introduction
- How makeinstaller.exe is run
- A few details of what is installed

Introduction

Phenix is a multimedia toolkit that uses Windows port of XWIN9 (aware of bets you may want to bundle it into an installer which makes it easier to install/uninstall/update). In order to simplify the lightweight but Phenix-free installer creation, a Perl script has been written to automate the process.

Download and install it on your PC and locate the entire environment variable to make instances use a portable version of Perl. Then, from a command prompt, change directory to the Phenix root and type: 'perl bin* for setting. ini' scripts. In addition it provides certain variables help as required in the scripts.

How makeinstaller.exe is run

Whereas the script is done through Bundler or done manually creating the installer starts with calling

```
makeinstaller.exe create_installer.bat
```

which creates a batch file which calls a few lines of example build script, Phenixbuild.cmd. It calls

```
create_project\make_installer_list\make_installer.pl
```

which calls

```
create_project\make_installer_list\make_installer.pl
```

which runs

```
create_project\make_installer_list\make_installer.pl
```

which creates a short properties of definitions such as version of Phenix and location of directories and their calls

```
Makeinstaller.PHENIXdirectoryfilelist.tpl
```

which creates a file called Makeinstaller.PHENIXdirectoryfilelist.tmp. This lists the include and where to place them on the users PC. It is an extension of the properties originated in the file

```
create_project\make_installer_list\makeinstallerdirectoryfilelist.tpl
```

which creates a file called Makeinstaller.PHENIXdirectoryfilelist.tmp. This lists the installer's directories and their contents in the entire tree. Since Phenix has Portable running the whole program is scattered, so a file with

```
distribution\phenix-version-number\Phenix-Setup.exe
```

which can then be deployed to users.

A few details of what is installed

Phenix installer is created with the intention to allow the user to install Phenix as a standard user as well as an administrator. The source files need to be copied to wherever if the installer was created and within Phenix the location of the Phenix root is set to the current directory. Phenix installer creates a few directories within which the installer is installed to but the actual location of the program files are determined by the environment variable PHENIXROOT. The installer also creates a few environment variables and sets them to the appropriate values. The installer will also allow the user to install the Perl 5 source code used for the build, if and where it is required and also allows the user to install the Perl 5 binaries.

- documentation,
- Perl 5 binaries,
- Phenix source package,
- Phenix binaries and Phenix environment variables have been defined,
- the uninstaller program.

The installer has two modes, a silent mode and one of type dialog in the help:

```
msi.exe/?
```

msi.exe/? gives a message box with the help.

- If the installer is invoked with administrative privileges:

```
msi.exe /a/norm c:\phenix
```

which creates the full path of where Phenix has been installed. This is useful for 3rd party programs that may need to know where Phenix is installed.

During the installation a command prompt appears and appears that compiles all the perlmod exercises.

The uninstaller is also defined in distribution\phenix-uninstall.exe. Uninstalling Phenix is done in 3 steps of which the one step is to remove the Phenix root directory. Phenix Perl 5 is also offered. During uninstallation the program has left being with the files has been completed.

developer

Python-based Practical Framework for Internet Enterprise

PHENIX Developer Environment

Getting started

To bring up the PHENIX development environment, you need to:

1. Clone phenix-devel-docker, build, install
2. Install Docker (Docker Desktop for Mac, Docker Desktop, WSL, etc.)
3. Build libtbx/boost/scons (Python, C++ extensions, etc.)
4. Build PHENIX
5. Run tests
6. Start PHENIX GUI (optional)

Each step is a separate command, so you can use the bootstrap.py script to automatect each task.

To get the phenix-devel-docker repo:

```
mkdir -p ~/projects/phenix-devel
cd ~/projects/phenix-devel && wget -nc -q -O bootstrap.py
https://raw.githubusercontent.com/phenix-project/phenix-devel-docker/refs/heads/main/bootstrap.py
```

- `./bootstrap.py --help` (or `python3 bootstrap.py --help`)
- `./bootstrap.py --build=phenix --no-tests` (or `python3 bootstrap.py --build=phenix --no-tests`)
- `./bootstrap.py --build=phenix --run=phenix.version` (or `python3 bootstrap.py --build=phenix --run=phenix.version`)
- `./bootstrap.py --build=phenix --run=phenix.gui` (or `python3 bootstrap.py --build=phenix --run=phenix.gui`)

Note: if you want to use the phenix-devel-docker repo from a source area, pass `--image=:<username>` (they only need to do so because the code source will use an existing account):

```
./bootstrap.py --build=phenix --image=:<username>
(or python3 bootstrap.py --build=phenix --image=:<username>)
```

Note: if you want to use a different image than the default (`phenix:latest`), "base" contains the dependencies, and "dev" is the build directory, you can pass `--image=<image_name>`. You can also do a full rebuild of the base image, or just the dev image, using `--rebuild-base` and `--rebuild-dev` (or `python3 bootstrap.py --rebuild-base --rebuild-dev`).

The bootstrap.py script is a convenient shortcut to a few "base" docker commands, and a convenient dispatcher, the base command plus a few other commands (BTW is the "base" docker command, and the dispatcher knows the convenient differences that base commands passes the uid/group to have rootless building, time zone to set the time in the container, and the current directory to be the source area). The dispatcher also knows how to run a command in the "dev" container (`./bootstrap.py --run=<command>`), and knows how to build the "base" and "dev" images (`./bootstrap.py --build=base --build=dev`).

Examples

Update to "trunk" (python):

```
./bootstrap.py --build=base,dev,update,test,update,test,update,test,update,test,dev,test,base,test,dev,test
python3 -c "from libtbx.utils import *; print(show_version())"
```

Update to use the "base" directory from a binary installer:

```
./bootstrap.py --build=base,dev,update,test,update,test,update,test,update,test,dev,test,base,test,dev,test
python3 -c "from libtbx.utils import *; print(show_version())"
```

dictionary

Phenix

Python-based Hierarchical Environment for Integrated Xtallography

Dictionary of common terms used in PHENIX

Delta-phi map, one of the most common types of **Fourier-Fourier difference maps** generated by modern refinement programs. Delta-phi maps are calculated using the difference between the phases of the observed structure factors and the phases of the **model structure factors calculated**, and are used to highlight regions of the model that are incorrect. The value of the coordinates of fractions will be to amplify the regions of the map in which phases of the true model are different from those of the model. Delta-phi maps are used to highlight regions of the model that are incorrect. Delta-phi maps are used to highlight regions of the model that are incorrect. Delta-phi maps are used to highlight regions of the model that are incorrect

Isomorphism: geometrically similar, which in the context of crystallography means that the space group is the same, the unit cell parameters are similar, the number of atoms in the unit cell is identical, and the atoms in the unit cell are arranged in almost the same way. Isomorphism is a concept that is important in the study of polymorphs, because it allows one to predict the structure of a polymorph that has not yet been found, if one of the other polymorphs has been found. Well-known isomorphous substances are the alkali halides, the alkaline earth oxides, the silver halides, the alkali metal nitrates, the alkali metal chlorates, the alkali metal perchlorates, the alkali metal sulfates, the alkali metal chromates, the alkali metal dichromates, the alkali metal permanganates, the alkali metal perchlorates, the alkali metal periodates, the alkali metal tetrafluoroborates, the alkali metal hexafluorophosphates, the alkali metal hexafluoroarsenates, the alkali metal hexafluoroantimonates, the alkali metal hexafluoroiodates, the alkali metal tetrachloroaluminates, the alkali metal tetrachloroferrates, the alkali metal tetrachlorogallates, the alkali metal tetrachloroindates, the alkali metal tetrachloroplatinates, the alkali metal tetrachloropalladates, the alkali metal tetrachloroiodates, the alkali metal tetrachloroiodates, the alkali metal tetrabromoaluminates, the alkali metal tetrabromoferrates, the alkali metal tetrabromogallates, the alkali metal tetrabromoindates, the alkali metal tetrabromoplatinates, the alkali metal tetrabromopalladates, the alkali metal tetrabromoiodates, the alkali metal tetrabromoiodates, the alkali metal tetraiodoaluminates, the alkali metal tetraiodoferrates, the alkali metal tetraiodogallates, the alkali metal tetraiodoindates, the alkali metal tetraiodoplatinates, the alkali metal tetraiodopalladates, the alkali metal tetraiodoiodates, the alkali metal tetraiodoiodates, the alkali metal tetrafluoroaluminates, the alkali metal tetrafluoroferrates, the alkali metal tetrafluorogallates, the alkali metal tetrafluoroindates, the alkali metal tetrafluoroplatinates, the alkali metal tetrafluoropalladates, the alkali metal tetrafluoroiodates, the alkali metal tetr

doc_procedures

PHENIX: Python-based Hierarchical Environment for Integrated Crystallography

Building the PHENIX documentation

To add a new page:

1. First make sure that SPHINXOPTS=-aqnW and SPHINXOptions. Point set up to false, and that plots.html is displayed out of TOC.
2. Add the new page to the TOC in doc/phenix_doc/phenix_toc.rst
3. Add it to the RST file.
4. Add it to the RST file, run ./plot_documentation.csh
5. Run ./make_doc.csh -a -c -b -d -p -t -v -w -x -n

To update documentation, make sure that the corresponding file in phenix_rest/rest_files, and then execute phenix.refine update.

Some pages require that the environment variable PHENIX_TEST_DATA is set to the directory that contains the phenix_regression/pdb/tutorial directory, and that the phenix.refine executable be in the path. The pages that require this will be marked with a "Requires PHENIX_TEST_DATA" in the title. Some of these pages also need to run "make configure" before each of the build iterations for the command to appear on your path.

After running the update, the new HTML files will be available in SPHENIX/doc, and may be viewed by running phenix.browser doc/index.html. Make sure that the new pages are there.

Excluding PHIL parameter listing

If you want to have your documentation page show a full listing of program options with help text, simply place the following in your RST file where the listing should appear:

.. ..phil.phenix.command_line.exclude:: :::

This grabs the path to the module containing the PHIL object from which to generate a full keyword listing, and adds the module to the exclude variable and finishes when importing the object.

Including citations

Published references are stored in PHIL format in phenix/doc/refmac/references.txt. This is not strictly necessary, but it makes it easier to keep the documentation up-to-date with the latest references. It also means that the same entry is available in other formats (Endnote, Mendeley), or the documentation. For references that are not in PDB, the entry should be added to the list of e-references by updating the file

.. ..citation:: :::

We encourage the use of the cited by tag, however, as others have to be newly incorporated into other documentation, it may be necessary to add the tag later.

A very fast introduction to restructured text format

The official documentation is here:

http://docutils.sourceforge.net/rst.html

but Pierre documentation set up a relatively simple subset of features. The best way to learn the syntax is by looking at examples, so, after substituting tuvt.chopping in the from CTRL, tuv is a single example in about below:

```
  This is a line
  This is another line

  Section title

  This is unformatted text, although as separated by blank lines, the
  paragraphs will be reformatted to fit the current screen width. Use
  single quotes, apostrophes and the character '\n' for new lines.  Left margin
  is preserved.

      This is a quoted paragraph
      Left margin is preserved

  Another section title

  Bulleted lists:

  List of all quantities requested:
```

You can see all of this and some references as described above and the special link above that will create a bibliography.

Proteomics Open Architecture Research Environment for Integrated Bibliography

PHENIX Examples

Where can I find sample data?

You can find sample data in the directories installed in $PHENIX/examples. Additionally there is a special PR data directory installed in $PHENIX/modules/PR. This directory contains a number of subdirectories with PR data samples. When the GUI is started for the first time, a dialog window will appear asking you to choose a project directory. Choose the directory $PHENIX/modules/PR/projects/tutorial.

Project Wizard
Project Wizard
Project Wizard

file_formats

About Phenix-supported file formats

Macromolecular Crystallography

Contents

- Model files
- Reflection files
- CIF files
- Sequence files
- Map files
- Phase files
- Text files with ASCII data
- Text files with binary data
- Other formats
- Compressed files

This document provides an introduction to the types of files used by or generated by Phenix. Many of these formats are also used by other software, but we have concentrated on the most popular file types.

We do not recommend editing any of these file types by hand. Some files (for human readable text) may be edited safely, but others (binary files) may be corrupted if they are not read by format or binary-aware editors and copied using the same software.

Model files

Phenix works internally and primarily in a series of records for each atom, plus accompanied metadata such as crystal symmetry. The most common format in macromolecular crystallography is the **PDB format**, which a typical line looks like:

```
ATOM   1  N  ILE A  1   6.704  7.141  9.000  1.00 10.00      N
ATOM   2  CA ILE A  1   6.704  5.691  9.000  1.00 10.00      C
ATOM   3  C  ILE A  1   5.290  5.138  9.000  1.00 10.00      C
ATOM   4  O  ILE A  1   4.310  5.878  9.000  1.00 10.00      O
ATOM   5  CB ILE A  1   7.414  5.138  7.762  1.00 10.00      C
```

Implicit in this information is a hierarchy of chains (with ID "A" in this case), residues (denoted by unique combination of residue number and name), and atoms (denoted by unique combination of atom name and alternate conformer ID) and its coordinates. The individual atom records have records for B-factor, occupancy, element type, and charge. Crystal symmetry information is stored in a separate record, for example:

```
CRYST1   79.000   79.000   38.000  90.00  90.00  90.00 P 4 2 2      16
```

Phenix also stores this information in

MAP files

Without these we usually revert to the Fourier coefficients, residual amplitude and phases, which is more convenient and easier that approx. several programs allow them to be output in CCP4 or X-plor formats. These are the "standard" formats for map files, and are described in detail in the CCP4 documentation. The CCP4 map format is a binary format, and is not easily transferred between machines. The X-plor format is a formatted version of the CCP4 format, and is easily transferred between machines. Most programs that output CCP4 format maps also output X-plor format maps. Most programs that accept map coefficients will automatically generate the missing phase information, if given a PDB file.

PDB files (.ent, .pdb, .ent.gz)

PDBs, which stands for "Protein Data Bank" or "Protein Data Bank Format" is the standard format for specifying atomic coordinates in a crystal. The PDB format is a fixed-column ASCII format, and is easily transferred between machines. The PDB format is a "standard" format, and most programs can read and write it. Most programs can read and write PDB files, and most programs can read and write PDB files compressed. You can download a PDB file from the PDB using the "fetch" command in the CCP4 GUI. The PDB format does not contain phase information, but does contain the crystallographic information (the parameters used in phasing refine).

```
CRYST1   a    b    c  alpha beta gamma
         sg        z
ORIGX1   1.000000  0.000000  0.000000        0.00000
ORIGX2   0.000000  1.000000  0.000000        0.00000
ORIGX3   0.000000  0.000000  1.000000        0.00000
SCALE1   0.012500  0.000000  0.000000        0.00000
SCALE2   0.000000  0.012500  0.000000        0.00000
SCALE3   0.000000  0.000000  0.012500        0.00000
ATOM   serial name rnum  x    y    z   occ  b
ATOM   serial name rnum  x    y    z   occ  b
```

Note asterisks in asterisks prove ("crazed") not have multiple values, as in the case of refinement input.pdb file, some down. Consult the manual for each individual program for more information about PDB files.

Geometry restraints info (.cns)

These files are generated by phenix.refine and related programs. They are not widely used but are very helpful when trying to understand refinement problems.

Other chemical data (various)

CIFIXE requires a variety of other formats in addition to CIF for specifying chemical information to be used in refinement and other calculations. The exception is PROCHECK, but the maker of various chemical/molecular programs is also responsible.

Log files (.log)

These log files are not being any specific format, and very widely between different programs. From the GUI point of view, the log files are important because they are the only place that program prints out crystallographic information not displayed graphically, as well as providing some insight into how a program works.

index

Phenix *Python-based Hierarchical ENvironment for Integrated Xtallography*

Phenix Documentation - version dev-xxx

NEW:Phenix documentation chatbot

Crystallography

Cryo-EM

Predicted models

How to install

GUI introduction

FAQs

Video tutorials

Tutorials & Examples

Overview of programs

Other links

Download
Changelog
GitHub
GitHub pulse
GitHub pulse on Jazzband.co

Citing Phenix

If you publish please cite:

Macromolecular structure determination using X-rays, neutrons and electrons: recent developments in Phenix. P. Liebschner, P. V. Afonine, M. L. Baker, G. Bunkóczi, V. B. Chen, T. I. Croll, B. Hintze, L. W. Hung, S. Jain, A. J. McCoy, N. W. Moriarty, R. D. Oeffner, B. K. Poon, M. G. Prisant, R. J. Read, J. S. Richardson, D. C. Richardson, M. D. Sammito, O. V. Sobolev, D. H. Stuart, T. C. Terwilliger, P. D. Adams, A. Urzhumtsev, D. Waterman, N. W. Echols, T. Lamzin, M. Wlodawer, M. Soriano, M. Levitt, J. Lunin, J. S. Berkholz, T. Ben-Yehezkel, O. V. Evrard, S. Gerstmann, S. Z. Zhou, F. Long, P. D. Adams and T. C. Terwilliger, Acta Cryst. (2019). D75, 861-877.

install-setup-run

Python-based Hierarchical Movement for Integrated Kinematics

How to set up and use PHENIX

Contents:

- Introduction
  - Software links on installation
  - Support forum

- Installation
  - Chatbot help on Installation
  - Platform support
  - Graphical installer - Mac
  - Graphical installer - Windows
  - Graphical installer - Linux
  - Space requirements
  - Command-line installer (macOS and Linux)
  - Setting up the command-line environment
  - Documentation
  - Help
  - User interfaces
  - The PANDA GUI
  - Command Line Interface
  - Troubleshooting
  - Problems starting the GUI
  - Problems starting the help
- Running Panda
  - Setting up the command-line environment
  - Documentation
  - Help
  - User interfaces
  - The PANDA GUI
  - Command Line Interface
  - Troubleshooting
  - Problems starting the GUI
  - Problems starting the help

Installation

Chatbot help on Installation

You can use the Panda chatbot to help you with installation.

Platform support

PANDA is supported on 64-bit common Linux platforms (kernel 2.6 or newer), Mac OS X version 10.7 or newer, and Windows 7 or newer. The installers are available as built on the following systems:

- CentOS 6.10 (64-bit)
- macOS 10.13 (Mojave)
- Windows 10 (1809)

Most of these installers should also run on newer systems and/or different Linux distributions such as Ubuntu, Debian, Fedora, and RHEL. Mac installers should also run on previous versions of macOS, and Windows installers should also run on previous versions of Windows. Linux installers are not compatible with the Ubuntu Snap or Flatpak, and may not be compatible with other Linux distributions with different versions of system libraries. Please contact the Panda team if you need support for a version of Panda or your version of Windows is also 64-bit.

Panda requires Intel-compatible CPUs. Apple Silicon Macs will work, but the GUI will have issues updating the Panda library file connected one hook and the GUI will work in macOS 10 and later.

Space requirements

See the company PANDA installation you will need approximately 4 GB of disk space.

Graphical installer - Mac

The Mac installer provides a drag-and-drop interface provided by the operating system.

You will see information and images on top of the installer.

Note: after downloading the panda.zip file you have left and choose Unzip-and. When asked about "Keep" or "Move" the downloaded file, choose "Keep" (unchecked) or the file and the Trash is not deleted. Its hash will it will give you the option to approve it yourself.

The installation directory is shown as the /Applications folder. The application will be an app with a name like /Applications/PANDA-Release-2.0.0-Build-1.app.

Graphical installer - Windows

Installation

- Download and run the PANDA setup executable on your PC and follow the instructions.

For ordinary users the default installation directory defaults to the folder defined by expanding the environment variables %LOCALAPPDATA%\PANDA-Release. If you want to install Panda for all users you need admin privileges and should run the installer with "Run as administrator" and choose the installation directory to be something like %ProgramFiles%\PANDA-Release or %ProgramFiles(x86)%\PANDA-Release. Choosing the second file helper path is required for accessing 32-bit.

Command-line installer (macOS and Linux)

The installer detects the latest distribution of PANDA including the binary bundles for your machine architecture, operating system.

```
$ tar -xf phasic-installer-version-platform.tar
```

Change to the installer directory:

```
$ cd phasic-installer-version
```

Run:

```
$ ./install' 'bin/install' -p /opt/local/phasic-version bp default,
    -d /opt/local/phasic-version/data
$ ./install' -platform=platform' bin/ www.docker/phasic-version and install' bin/
```

Note: The installer will ask for a directory path if it works with a 32-bit binary path starting with './' if not work with a 64-bit binary path.

Installation of the binary version of PANDA requires no compilation, only the generation of some data files, so installation should take only a few minutes. The installation requires the installation to complete depending on the performance of your installation platform.

For binary information please see Linux file. For source of components see switch.

Rosetta Installation

Installation of Rosetta software developed from the Baker laboratory at the University of Washington, is optional.

- Download Rosetta
- Rosetta Installation
- Rosetta-PANDA Interface: Install

Rosetta Installation Instructions for Panda

Buildbot Installation

If you want to use the graphical interface and Installer Panda using the Mac or Windows graphical installers, you need to install the Buildbot software. Instructions for this are for the Dark Matrix GUI. Instructions below are for the command-line installer, and will also work.

Setting up the command-line environment

Once you have successfully installed PANDA, be sure to your environment please source the phasic.csh file in the /bin/ directory of the PANDA installation. If you are using the default installation location, such as /opt/local/, type:

```
$ source /opt/local/phasic-version/data/phasic.csh
```

or:

```
$ . /opt/local/phasic-version/data/phasic.csh
```

To run jobs locally, you need to source the phasic.csh or phasic.sh (or equivalent) file.

If you used the graphical Mac installer, you can find the files here:

/Applications/PANDA-Release-2.0.0-Build-1.app/Contents/Resources/phasic

The following environment variables should now be defined (look with example values):

```
PANDA=/opt/local/phasic
PANDA_DIR=/opt/local/phasic
PANDA_BIN=/opt/local/phasic/bin
PANDA_DATA=/opt/local/phasic/data
PANDA_VERSION=2.0.0
PANDA_BUILD=Release-2.0.0-Build-1
PANDA_PLATFORM=linux-x86_64
PANDA_PYTHON=/opt/local/phasic/bin/python
```

It is not necessary but useful to add your environmental variables for SOLAR/MUSCLE/ for PANDA. If you have Solar set in your environment they are ignored by PANDA.

Documentation

Documentation is available in the PANDA and Linux file help (www). Alternatively, you can use a web browser to view the documentation supplied with PANDA. To locate:

```
$ ./bin/phasic -help
```

Alternatively, if the directory of Firefox installed locate that you can open a web browser to the correct location in your PANDA installation (for example):

```
$ firefox /opt/local/phasic-version/doc/index.html
```

or:

```
$ mozilla /PANDA/doc/index.html
```

For Linux information please see the LICENSE file.

For the source of the components see SOURCES.

Help

You can join the PANDA bulletin board action/ zoke the archives.

Help for the command-line installer is available in /bin/install/panda

Alternatively you can send email to:

```
help@phasic.com, /-p /p/d /-bin to ask a question or report bugs!
```

User interfaces

Different user interfaces are required depending on the needs of a license user community. Most websites in PANDA, Linux, and Mac installations are compatible with Firefox and Safari. Chrome is not. Firefox is the best supported browser. Chrome and Chromium browsers are not recommended using the graphical interface.

The PANDA GUI

For new users and anyone unfamiliar with Unix command lines, we recommend using the graphical interface. To run, simply type the command:

```
panda -gui
```

Please see the User documentation files to get more details about the PANDA GUI.

Command Line Interface

Advanced users or anyone developing automated pipelines may prefer to use the command line interface. The command line interface is a powerful tool that allows the user to run the same functions as the graphical interface in a noninteractive mode or in the background. Unlike the "Run Job" functions the user of an or the work flows in the graphical interface, the command line interface allows the user to run the same functions in batch mode, manipulations such as the generation of a job set, monitoring and viewing of data.

Command line interface is also the only way to access the functions of the PANDA. Most of the Wizard programs such as Library Utility and the Solution Validation and Layout Analysis are only available as command line functions.

To illustrate the command line interface, the command used to run the program that carries out a data quality analysis is:

```
phasic-qc-qa data.qcs [options]
```

Further options can be given on the command line, or can be specified via a parameter file.

To see a list of available options:

```
phasic -help -help
```

There are also many other command line tools (described in detail elsewhere in this documentation). You can list them with the command:

```
phasic -ls
```

or alternatively:

```
phasic -help
```

or

```
phasic -help -gui
```

to list all methods that contain the characters gui in their names or descriptions.

Many of the command line tools are also available as functions in the PANDA library. This will always give the most complete and up-to-date access to the PANDA software if a standalone tool does not exist or does not contain all the options you need for your application.

The runtime options for most PANDA tools are controlled using a keyword/value syntax called PFx (Python Formatted Exchange). Keywords are preceded by a dash (-) and are followed by a value. Values can be a simple parameter file, a basic overview of PFx is available at the overview of PFx tutorial.

Troubleshooting

Problems starting the GUI

Xforms tends to be installed without an important library that is linked to the GUI code. If you encounter an error message when trying to start the GUI, you can add to your system with the command:

```
- sudo apt-get install libpng12
```

making_geometry_restraints

Making geometry restraints

- Geometrical structure restraints
- Generalized dihedrals

Using PyMOL with PHENIX

Python-extended Molecular Graphics for Crystallographers

Author(s)

[PyMOL](#) was originally written by Warren DeLano, and is now distributed by Schrödinger LLC.

Instructions

PyMOL is perhaps the most popular tool for PyMOL (version 0.99) with the Linux variables for PHENIX, however **do** recommend using a more recent version (since bugs are fixed more frequently) such as available at Source Forge. Although PyMOL is a very powerful program, it does not have all the features of more specialized crystallographic programs, and we recommend using the latest open-source code yourself, or you may be able to install a binary version. The open-source version of PyMOL is distributed with the Linux version of PHENIX, and can be invoked from the command line by the "phenix.pymol" command. The Mac version is also available from Source Forge, but is not distributed with the PHENIX installer.

Configuring and launching PyMOL from within the PHENIX GUI is very similar to what for Coot, if it is installed on your system and the environment variable "PYMOL_PATH" points to the executable. Although these tools are designed to complement, not replace, Coot, there is some overlap in PyMOL's ability to display electron density maps and to automatically fit ligands into density. However, PyMOL will perform the necessary CPU-intensive calculations in parallel, and can also generate publication-quality images and movies. PyMOL's ability to generate CCP4 maps if not are available.

Phenix.Refine: Cycle 1 of 3
Phenix.Refine: R-factor = 0.2
Phenix.Refine: R-free = 0.3
Phenix.Refine: Cycle 2 of 3
Phenix.Refine: R-factor = 0.2
Phenix.Refine: R-free = 0.3
Phenix.Refine: Cycle 3 of 3
Phenix.Refine: R-factor = 0.2
Phenix.Refine: R-free = 0.3
Phenix.Refine: Finished
Phenix.Refine: Output files:
Phenix.Refine: Refined model: 1ubq_refine_001.pdb
Phenix.Refine: Map coefficients: 1ubq_refine_001.mtz
Phenix.Refine: Log file: 1ubq_refine_001.log
Phenix.Refine: Time: 0.0 sec