Using Python on Unix platforms

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 1)

Unknown directive type "highlight".

```
.. highlight:: sh
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 9)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Shriphani Palakodety
```

Getting and installing the latest version of Python

On Linux

Python comes preinstalled on most Linux distributions, and is available as a package on all others. However there are certain features you might want to use that are not available on your distro's package. You can easily compile the latest version of Python from source.

In the event that Python doesn't come preinstalled and isn't in the repositories as well, you can easily make packages for your own distro. Have a look at the following links:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 27)

Unknown directive type "seealso".

```
.. seealso::

   https://www.debian.org/doc/manuals/maint-guide/first.en.html
      for Debian users
   https://en.opensuse.org/Portal:Packaging
      for OpenSuse users
   https://docs-old.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/ch-creating-rp
      for Fedora users
   http://www.slackbook.org/html/package-management-making-packages.html
      for Slackware users
```

On FreeBSD and OpenBSD

- FreeBSD users, to add the package use:

  ```
  pkg install python3
  ```

- OpenBSD users, to add the package use:

  ```
  pkg_add -r python

  pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/4.2/packages/<insert your architecture here>/python-<version>.t
  ```

  For example i386 users get the 2.5.1 version of Python using:

  ```
  pkg_add ftp://ftp.openbsd.org/pub/OpenBSD/4.2/packages/i386/python-2.5.1p2.tgz
  ```

On OpenSolaris

You can get Python from OpenCSW. Various versions of Python are available and can be installed with e.g. `pkgutil -i python27`.

Building Python

If you want to compile CPython yourself, first thing you should do is get the source. You can download either the latest release's source or just grab a fresh clone. (If you want to contribute patches, you will need a clone.)

The build process consists of the usual commands:

```
./configure
make
make install
```

:ref:`Configuration options <configure-options>` and caveats for specific Unix platforms are extensively documented in the :source:`README.rst` file in the root of the Python source tree.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 81); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 81); *backlink*

Unknown interpreted text role "source".

Warning

`make install` can overwrite or masquerade the :file:`python3` binary. `make altinstall` is therefore recommended instead of `make install` since it only installs :file:`{exec_prefix}/bin/python{version}`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 87); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 87); *backlink*

Unknown interpreted text role "file".

Python-related paths and files

These are subject to difference depending on local installation conventions; :envvar:`prefix` (`${prefix}`) and :envvar:`exec_prefix` (`${exec_prefix}`) are installation-dependent and should be interpreted as for GNU software; they may be the same.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 95); *backlink*

Unknown interpreted text role "envvar".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 95); *backlink*

Unknown interpreted text role "envvar".

For example, on most Linux systems, the default for both is :file:`/usr`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 100); *backlink*

Unknown interpreted text role "file".

File/directory	Meaning
:file:`{exec_prefix}/bin/python3` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 106); *backlink* Unknown interpreted text role "file".	Recommended location of the interpreter.

File/directory	Meaning
:file:`{prefix}/lib/python{version}`, :file:`{exec_prefix}/lib/python{version}` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 108); *backlink* Unknown interpreted text role "file". **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 108); *backlink* Unknown interpreted text role "file".	Recommended locations of the directories containing the standard modules.
:file:`{prefix}/include/python{version}`, :file:`{exec_prefix}/include/python{version}` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 111); *backlink* Unknown interpreted text role "file". **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 111); *backlink* Unknown interpreted text role "file".	Recommended locations of the directories containing the include files needed for developing Python extensions and embedding the interpreter.

Miscellaneous

To easily use Python scripts on Unix, you need to make them executable, e.g. with

```
$ chmod +x script
```

and put an appropriate Shebang line at the top of the script. A good choice is usually

```
#!/usr/bin/env python3
```

which searches for the Python interpreter in the whole :envvar:`PATH`. However, some Unices may not have the :program:`env` command, so you may need to hardcode `/usr/bin/python3` as the interpreter path.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 132); *backlink*

Unknown interpreted text role "envvar".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 132); *backlink*

Unknown interpreted text role "program".

To use shell commands in your Python scripts, look at the :mod:`subprocess` module.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\[cpython-main][Doc][using]unix.rst`, line 136); *backlink*

Unknown interpreted text role "mod".

Custom OpenSSL

1. To use your vendor's OpenSSL configuration and system trust store, locate the directory with `openssl.cnf` file or symlink in `/etc`. On most distribution the file is either in `/etc/ssl` or `/etc/pki/tls`. The directory should also contain a

`cert.pem` file and/or a `certs` directory.

```
$ find /etc/ -name openssl.cnf -printf "%h\n"
/etc/ssl
```

2. Download, build, and install OpenSSL. Make sure you use `install_sw` and not `install`. The `install_sw` target does not override `openssl.cnf`.

```
$ curl -O https://www.openssl.org/source/openssl-VERSION.tar.gz
   $ tar xzf openssl-VERSION
   $ pushd openssl-VERSION
   $ ./config \
        --prefix=/usr/local/custom-openssl \
        --libdir=lib \
        --openssldir=/etc/ssl
   $ make -j1 depend
   $ make -j8
   $ make install_sw
   $ popd
```

3. Build Python with custom OpenSSL (see the configure *--with-openssl* and *--with-openssl-rpath* options)

```
$ pushd python-3.x.x
$ ./configure -C \
    --with-openssl=/usr/local/custom-openssl \
    --with-openssl-rpath=auto \
    --prefix=/usr/local/python-3.x.x
$ make -j8
$ make altinstall
```

Note

Patch releases of OpenSSL have a backwards compatible ABI. You don't need to recompile Python to update OpenSSL. It's sufficient to replace the custom OpenSSL installation with a newer version.