

Preparing Bash for a desktop operating system

last updated: 2019-03-15

1 Introduction

A **command line interface (CLI)** is a way for a computer user to interact with a computer by issuing text commands. The CLI is contrasted to the graphical user interface (GUI) available on most desktop computers and smartphones, where the user interacts with graphical objects such as icons and windows.

A **shell** is a CLI or GUI computer program that allows the user to interact with the computer's operating system. **Bash** is a Unix shell (and a command line language) written for the GNU Project as a free software shell for Linux[1].

This document explains how to get Bash working on a desktop operating system. It also gives suggestions for additional software to install and provides some guidance for installing them. It does not discuss how to configure Bash according to user preference.

Note that there may be some differences between Bash across operating systems. In particular, since Windows, Mac and Linux use different file systems, there are some differences in how software that accesses the file system display information and behave.

2 Preparing Bash

2.1 Linux

Most Linux distributions include Bash, so there should be nothing to do in order to prepare Bash except to configure Bash and the terminal according to the user's preferences.

2.2 Windows

Bash for Windows is most often prepared by installing **Cygwin**. Cygwin also has an installer for a wide range of software packages that are used on the command line.

Cygwin can be downloaded from the official website:

```
https://www.cygwin.com/
```

Follow the steps below to install Cygwin. Note that administrator privileges are likely required, especially when installing Cygwin for all users (system-wide).

1. Check your variant of Windows (32-bit or 64-bit) and download the appropriate variant of the Cygwin installer. The 32-bit variant is named `setup-x86.exe` and the 64-bit variant is named `setup-x86_64.exe`.
2. Launch the installer to commence installation. Choose "Install from internet".
3. Choose the installation directory: the default is `C:\cygwin` or `C:\cygwin64`, but you may wish to check that it is simple and doesn't contain any spaces.
4. For setting up your internet connection, the default "Direct connection for your internet connection" may be best, but change it to whatever is appropriate for your network.
5. Choose a download mirror.
6. Select packages that you wish to install. The Cygwin core packages will already be marked for installation. See section 3 for additional software recommendations.
7. Confirm the list of software packages to install. Note that this list will be longer than the list of packages marked in the previous step, since dependencies of marked packages will also need to be installed. Also, ensure that "Select required packages (RECOMMENDED)" is checked.
8. Select whether or not you wish to have desktop shortcut or start menu item created.

2.3 Mac

Mac OS includes Bash, so there should be nothing to do in order to prepare Bash. However, the user may want to configure Bash, and Mac users may want to consider configuring the default terminal application or install **iTerm2** (iTerm2 3.2.7 works on Mac OS 10.12 or later):

```
https://iterm2.com/
```

3 Installing additional software

To test whether or not software is installed (ie: its executable exists inside any of the paths listed in variable `$PATH`), use `which`. If the executable exists, its path is printed to stdout.

```
$ which <executable>
```

Sometimes it's necessary to check the versions of installed software. To allow users to check the version of software, an executable would often provide a flag such as `--version` or `-v` or `-V`. An example for `ls` is shown.

```
$ ls --version
```

3.1 C/C++ (low-level programming)

C and C++ are compiled low-level programming languages with memory management capabilities and good performance. C is widely used due to its portability and power, and its extension C++ is often used as an object-oriented programming language.

Linux distributions with a package manager may offer C/C++ compilers as installable packages. The core packages for **GCC** (C) and **G++** (C++) on Debian are `gcc` and `g++` respectively. There are Cygwin packages with similar names. There exist other C/C++ compilers such as **Clang**.

3.2 Git (version control)

Git is a decentralised version control tool that is optimised to keep track of versions of text files and source files. Many software developers use Git to keep track of source code changes and releases, and it can also be used to manage changes to scripts, configuration files, documents, essays and webpages.

Linux distributions with a package manager may offer Git as an installable package. The core package on Debian is `git`. There may be a Cygwin package with a similar name.

Git for Linux, Windows, Mac and other platforms (and source code) can be downloaded from the official website:

```
https://www.git-scm.com
```

3.3 GMT (geographical mapping)

GMT (Generic Mapping Tools) is a collection of software tools for doing geographical mapping and visualisation. Features include rasterisation, filtering, map projections and other image processing and mapping operations. GMT can be operated from the command line and scripted to automatically generate output.

Linux distributions with a package manager may offer GMT as an installable package. The core package on Debian is `gmt`. There may be a Cygwin package with a similar name.

GMT can be downloaded from the official website:

```
http://gmt.soest.hawaii.edu/ (link broken?)
```

3.4 GnuPG (cryptography)

GnuPG is a cryptographic software that is capable of generating and managing cryptographic keys, and signing and encrypting data. It is often added to email clients to enhance the authentication and confidentiality of emails, and is a core part of many Linux distributions that rely on verifying the integrity of software packages prior to their installation.

Linux distributions often use GnuPG to verify the integrity of packages prior to installation, as such GnuPG should already be installed. The core package on Debian is `gnupg`.

GnuPG for Linux, Windows, Mac and other platforms (and source code) can be downloaded from the official website:

```
https://gnupg.org/  
http://ic6au7wa3f6naxjq.onion/
```

Gpg4win (a suite of GnuPG tools for Windows) can be downloaded from the official website:

```
https://gpg4win.org/
```

GPGTools (a suite of GnuPG tools for Mac) can be downloaded from the official website:

```
https://gpgtools.org/
```

3.5 Python (high-level programming)

Python is an interpreted high-level programming language that is designed to be easily readable and flexible. It is useful for performing calculations and rapidly prototyping algorithms, and it's portable across any target that can run a Python interpreter.

Linux distributions with a package manager may offer Python as an installable package. The core package on Debian is `python`. There may be a Cygwin package with a similar name.

Python for Windows, Mac and other platforms (and source code) can be downloaded from the official website:

```
https://www.python.org/
```

3.6 R (statistical programming)

R is a programming language and software environment for doing statistical computing. It is used widely in the research, academic and data science communities for its powerful statistical functionality.

Linux distributions with a package manager may offer R as an installable package. The core package on Debian is `r-base-core`. There may be a Cygwin package with a similar name.

R for Linux, Windows and Mac (and source code) can be downloaded from the CRAN (Comprehensive R Archive Network) website:

```
https://cran.r-project.org/
```

Some users may also want **RStudio**, which provides a GUI with panels including text editor, command line, environment and plots shown simultaneously. RStudio can be obtained from its website for Linux (`deb` and `rpm`), Windows, Mac and source:

```
https://www.rstudio.com/products/rstudio/download/
```

3.7 TeX (document preparation)

TeX is a document preparation language and compiler, often used by the academic and research community to create theses and academic papers. Unlike office software such as Libre Office, TeX is not WYSIWYG (what you see is what you get), but document source files can be easily managed (eg: by Git) or automatically generated, the language is powerful, and it's capable of producing beautiful and well-formatted documents. This document is made using TeX. **LaTeX** is a collection of TeX macros intended to provide writers with a high-level language for easy access to the features of TeX[2]. **TeX Live** is a free software distribution for TeX and LaTeX.

Linux distributions with a package manager may offer TeX Live as an installable package. The core package on Debian is `texlive-base`. There may be a Cygwin package with a similar name.

TeX Live for Windows and other platforms (and source code) can be downloaded from the official TeX Live website:

```
https://tug.org/texlive/
```

MikTeX (TeX and related tools for Windows) can be downloaded from the MikTeX page:

```
https://miktex.org/
```

MacTeX (TeX Live for Mac plus some Mac-specific tools) can be downloaded from the MacTeX page:

```
https://tug.org/mactex/
```

3.8 Miscellaneous command line tools

- `htop` shows running processes.
- `info` is a documentation viewer for commands.
- `man` is a manpage (documentation) viewer for commands (eg: `man bash`).
- `sha{256,512}sum` computes and checks SHA hashes for data integrity checking.
- `tree` displays files and directories in a tree.
- `vim` is a powerful terminal-based text editor.

4 Making Cygwin useful

The default installation of Cygwin on Windows doesn't provide a command line interface that is very useful. After running a workshop about using Bash and the command line, where most participants installed and played with Cygwin, it was found that a Cygwin installation needs some additional work to make it useful and that a section specifically about Cygwin is deserved.

Cygwin documentation is available at the following webpage:

```
https://cygwin.com/cygwin-ug-net/cygwin-ug-net.html
```

4.1 Installing packages for Cygwin

The Cygwin setup executables `setup-x86.exe` (32 bit) and `setup-x86_64.exe` (64 bit) should be kept rather than deleted after Cygwin installation, since it is needed to install and uninstall packages[3]. Those executables can be executed in GUI mode (eg: by double-clicking on its icon) or used on the command line[4] (after making them accessible via a path in `$PATH`; see section 4.2):

```
$ setup-x86_64.exe -P <package-name>
```

To install multiple packages at once, use `-P` multiple times, or use the following brace expansion with `"-P "` (including the space) as its prefix:

```
$ setup-x86_64.exe "-P" "{<pkg1>,<pkg2>,...}"
```

Cygwin users may want to consider installing these packages (or ensure they are installed):

- `gcc` : (GNU compiler collection) compiler suite for C/C++ and other languages.
- `gimp` : (GNU image manipulation program) image editor.
- `gnupg2` : cryptographic tool that encrypts/signs data and manages keys.
- `gnuplot` : function plotting tool.
- `ImageMagick` : image processing suite.
- `info` : documentation viewer for commands.
- `man-db` : manpage (documentation) viewer for commands (eg: `man bash`).
- `python3` : general-purpose interpreted high-level programming language.
- `sed` : stream editing tool (eg: search and replace) that's useful with stdin and stdout.
- `texlive` or `texlive-collection-basic` : LaTeX typesetting tools.
- `tree` : displays files and directories in a tree.
- `vim` : terminal-based text editor.
- `wget` : fetches data (eg: from the internet) when given a URL.
- `xxd` : displays data in hexadecimal format.

A full list of Cygwin packages is available at the following webpage:

```
https://cygwin.com/cgi-bin2/package-grep.cgi
```

4.2 Making Bash find executables

Commands that are included as part of Cygwin can be typed directly into Bash without specifying their full paths. However, this may not be the case for non-Cygwin software that resides on your computer.

This section describes making Bash find executables (eg: `.exe` files) so that they can be easily used as commands. The environment variable `$PATH` is a colon-separated list of directories that Bash searches for commands in. This variable can be taken advantage of by adding an additional directory.

First, change the working directory to home: `~` (tilde), `/home/<user>` or `$HOME`, and create a directory named `~/bin` that will store **symbolic links** (Windows: "shortcuts") to executables:

```
$ cd ~  
  
$ mkdir bin
```

The file `.bashrc` is a Bash configuration file that is loaded every time an interactive Bash environment (eg: Cygwin) is launched. Add the new directory `~/bin` to `$PATH` by adding the following line to the end of `.bashrc`:

```
$ echo 'export PATH="$PATH:$HOME/bin"' >> .bashrc # add ~/bin to $PATH
```

Restart Cygwin to have the change applied. After that, check that the new path is added correctly:

```
$ echo $PATH  
/usr/local/bin:/usr/bin:/bin:...:/home/<user> # this is an example only
```

Find your favourite software (eg: look through `"/cygdrive/c/Program Files"`), and create symbolic links for each. The command/syntax for creating a symbolic link is (`<file>` must include the executable's full path, `-v` means verbose mode, `<link-name>` should be a unique and Bash-friendly name):

```
$ ln -sv <file> <link-name>
```

For example, let's make the Cygwin setup executable accessible as a command. Move the executable (by specifying its full path) to Cygwin's installation directory first, and then create a link for it:

```
# 64-bit version  
$ mv <setup-x86_64.exe> /cygdrive/c/cygwin64  
  
$ ln -sv /cygdrive/c/cygwin64/setup-x86_64.exe bin/cygwin-setup  
'bin/cygwin-setup' -> '/cygdrive/c/cygwin64/setup-x86_64.exe'
```

```
# 32-bit version  
$ mv <setup-x86.exe> /cygdrive/c/cygwin  
  
$ ln -sv /cygdrive/c/cygwin/setup-x86.exe bin/cygwin-setup  
'bin/cygwin-setup' -> '/cygdrive/c/cygwin/setup-x86.exe'
```

To check that an executable can be found, use `which`:

```
$ which <executable>
```

4.3 Making links to frequently-used directories

The symbolic link creation methods as mentioned in section 4.2 can be used to create convenient links to frequently-used directories inside your home directory (or some other easy-to-access location). You may want to consider creating links for the following directories:

- `/cygdrive/c` : Windows C drive;
- `/cygdrive/c/Users/<user>` : your Windows user directory;
- `"/cygdrive/c/Program Files"` or `"/cygdrive/c/Program Files (x86)"`.

The syntax is the same as previously:

```
$ ln -sv <directory> <link-name>
```

4.4 Making useful aliases

An **alias** is an object that can be referred to like a command that acts as an alias to a command, using the command `alias`. Aliases can be keyboard- or -Bash-friendly names for commands, or commands with predefined arguments. Aliases should be added to your `.bashrc`. However, before adding aliases, check for existing aliases in your `.bashrc` so that you don't create conflicts.

```
# examples that can be added to .bashrc
alias ls='ls --color=auto' # add colour to ls
alias ll='ls -lF --time-style=iso' # create long list command based on ls
alias rm='rm -i' # prompt before removing files
alias cp='cp -i' # prompt before overwriting files
alias mv='mv -i' # prompt before overwriting files
alias grep='grep --color=auto' # add colour to grep
alias python='python3' # override default python to v3
alias R='R --quiet' # suppress R's welcome message
```

References

- [1] *Bash (Unix shell)*. Wikipedia.
[https://en.wikipedia.org/wiki/Bash_\(Unix_shell\)](https://en.wikipedia.org/wiki/Bash_(Unix_shell))
- [2] *LaTeX*. Wikipedia.
<https://en.wikipedia.org/wiki/LaTeX>
- [3] *How to install new packages on Cygwin?*. Super User StackExchange.
<https://superuser.com/questions/304541/how-to-install-new-packages-on-cygwin>
- [4] *How to Install the Latest GCC on Windows*. 2014.
<https://preshing.com/20141108/how-to-install-the-latest-gcc-on-windows/>