Laboratory 2: installing software

The software installed on your computer by default is capable of doing many wonderful things, but there will come a time when you will need something a bit more specialized. This laboratory exercise is designed to expose you to the most common mechanisms of software installation available in Ubuntu.

Tasks

(1) Install the font 'Hack'. This font is specifically designed for programmers and helps one easily spot errors. For example, '`0`' and '`O`' have a very different appearance in Hack. It is the font used to typeset commands in these laboratory exercises.

   (a) Make sure your computer is connected to the internet (e.g. open a web page to check for connectivity or type `ping -c 5 9.9.9.9` in the terminal and confirm that there is 0% packet loss).

   (b) In the terminal, type `sudo apt update` to update the stored list of packages available for your computer (the package cache). Type your password when prompted. Answer question (1).

   (c) Type `sudo apt upgrade` in the terminal to upgrade to the latest versions of each package and then type `sudo apt autoremove` to remove any old versions (when installing software always remember these steps as skipping them can cause problems). Type your password, if prompted, and agree to continue when prompted.

   (d) Type `sudo apt install fonts-hack` to install Hack. Type your password, if prompted, and agree to continue when prompted.

   (e) Change the default font in the terminal to Hack by selecting 'Preferences...' from the 'File' menu and setting 'Font' to be Hack. Test by opening a new terminal window.

(2) Install the program 'Visual Studio Code'. It is a very nice text editor designed specifically for programming. There are a number of built–in features and optional extensions to help programmers enter code faster and identify potential errors before the code is run. Visual Studio Code is available, for free, from Microsoft.

   (a) In the terminal, type `wget -q -O - https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > packages.microsoft.gpg` to download and process the Microsoft GNU Privacy Guard code signing key.

   (b) Type `sudo install -o root -g root -m 644 packages.microsoft.gpg /usr/share/keyrings/` in the terminal to install the key. If prompted, type your password.

   (c) In the terminal, type `echo 'deb [arch=amd64 signed-by=/usr/share/keyrings/packages.microsoft.gpg] https://packages.microsoft.com/repos/vscode stable main' | sudo tee /etc/apt/sources.list.d/vscode.list > /dev/null` to add the Visual Studio Code repository to your system. If prompted, type your password.

   (d) Update the package cache by typing `sudo apt update` in the terminal. If prompted, type your password. Answer question (2).

   (e) Finally, type `sudo apt install code` in the terminal to install the program. Type your password, if prompted, and agree to continue when prompted.

(f) Open Visual Studio Code by typing `code &` in the terminal (the '&' allows you to continue working in the terminal without opening another window). You may wish to add Visual Studio Code to the quick launcher for easy access.

(3) Install a Perl script for local use.

(a) Download the 'B.pl' (Little 2010) using the command line by typing: `wget http://www.nybg.org/files/scientists/dlittle/B-1.2.tar.gz` in the terminal.

(b) Unarchive and decompress the files by typing `tar xvzf B-1.2.tar.gz` in the terminal. Answer question (3).

(c) Create your own 'scripts' folder by typing `mkdir scripts` in the terminal.

(d) Move the script to the 'scripts' folder by typing `mv B.pl scripts/` in the terminal.

(e) Delete the extra files by typing `rm B-1.2.tar.gz ._B.pl README.txt ._README.txt gpl.txt ._example.CAF example.spf ._example.spf` in the terminal.

(f) Determine the current contents of your $PATH variable by typing `echo $PATH` in the terminal.

(g) Read over the invisible file '.bashrc' using a simple text editor by typing `nano .bashrc` in the terminal. This script is read by bash every time a new (non–login) shell is started, so one can set 'preferences' by including actions in the file. Exit nano by by pressing `ctrl` and `x` at the same time.

(h) Add scripts to your PATH by typing `echo 'export PATH=$PATH:$HOME/scripts' >> .bashrc` in the terminal. Answer question (4).

(i) Exit the terminal and reopen it (this loads the newly edited version of '.bashrc').

(j) Determine the current contents of your $PATH variable by typing `echo $PATH` in the terminal. It should have changed from step (f). If your scripts directory is not appended to the end, something is wrong.

(k) Type `B.pl` in the terminal to see the script instructions.

(l) Type `B.pl -i example.CAF` in the terminal to run the script. Answer question (5).

(4) Install the ABySS assembler (https://github.com/bcgsc/abyss/; Jackman et al. 2017).

(a) To find out if ABySS is available as a package, type `apt-cache search abyss` in the terminal.

(b) To determine which version of ABySS is available, type `apt-cache show abyss` in the terminal. Answer question (6).

(c) To compile ABySS from source, you must install automake, make, a compiler, a linker, and standard libraries. Type `sudo apt install build-essential automake` in the terminal to install the basics. Type your password, if prompted, and agree to continue when prompted.

(d) Download the current release of ABySS by typing `wget https://github.com/bcgsc/abyss/archive/refs/tags/2.3.5.tar.gz` in the terminal.

(e) Expand the archive by typing `tar xvzf 2.3.5.tar.gz` in the terminal.

(f) Change to the ABySS directory by typing `cd abyss-2.3.5` in the terminal.

(g) Generate a config file for your system by typing `./autogen.sh` in the terminal.

(h) Create a MakeFile for your system by typing `./configure --with-mpi` in the terminal. Answer question (7).

(i) Install the boost C++ library by typing `sudo apt install libboost-all-dev` in the terminal. Type your password, if prompted, and agree to continue when prompted.

(j) Rerun the configure script by typing `./configure --with-mpi` in the terminal. Answer question (8).

(k) Install the Google sparse hash library by typing `sudo apt install libsparsehash-dev` in the terminal. Type your password, if prompted, and agree to continue when prompted.

(l) Rerun the configure script by typing `./configure --with-mpi` in the terminal.

(m) Attempt to build ABySS by typing `make -j$(nproc) CFLAGS='-march=native -O2'` in the terminal. Answer question (9).

(n) Test the build by typing `make check` in the terminal. Answer question (10).

(o) Install ABySS by typing `sudo make install` in the terminal. If prompted, type your password.

(5) Install a Docker image of Goalign (https://github.com/evolbioinfo/goalign; Lemoine & Gascuel 2021).

(a) Install the Docker code signing key by typing `wget -q -O - https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor > docker.gpg` to download and process the Docker GNU Privacy Guard code signing key.

(b) Type `sudo install -o root -g root -m 644 docker.gpg /usr/share/keyrings/` in the terminal to install the key. If prompted, type your password.

(c) Add the Docker repository by typing `echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null` in the terminal.

(d) Update the package lists and install Docker by typing `sudo apt update && sudo apt install docker-ce docker-ce-cli containerd.io` in the terminal. Agree to continue when prompted.

(e) Add yourself to the 'docker' group (that does not exist yet) by typing `sudo usermod -aG docker ${USER}` in the terminal.

(f) Create the 'docker' group by typing `newgrp docker` in the terminal. You can test that the group is working correctly by typing `id -nG` in the terminal—you should see 'docker' as one of the groups listed.

(g) Type `docker run --rm pegi3s/goalign goalign help` to install the Goalign Docker image and then display help information. Answer question (11).

Questions (https://forms.gle/fHdqspviX4wheaCS7)

(1) For task (1)(b), which urls are providing package lists?

(2) For task (2)(d), which urls were added by task (2)(c)?

(3) For task (3)(b), what does each of the tar options do?

(4) For task (3)(h):

   (a) What does each part of the command do?
   (b) What is '$HOME'?
   (c) Is the value of '$HOME' the same for everyone?
   (d) What could you use in place of '$HOME' in the .bashrc file?

(5) For task (3)(l), what is the barcode sequence quality for the example file?

(6) For task (4)(b), what version of ABySS is available from the package repository and what is the current release version?

(7) For task (4)(h):

   (a) Was the configure script able to create a MakeFile?
   (b) What dependencies were missing?
   (c) How else could you have determined what was required?

(8) For task (4)(j), are all of the dependencies satisfied now? Explain.

(9) For task (4)(m):

   (a) What does 'nproc' do?
   (b) Why is it surrounded by parentheses and proceeded by a dollar sign?
   (c) Would it behave any differently if it were surrounded by grave accents?

(10) For task (4)(n), how can you tell if the build is good or not?

(11) For task (5)(g), which version of Goalign did you install?

Literature cited

Jackman, S. D., B. P. Vandervalk, H. Mohamadi, J. Chu, S. Yeo, S. A. Hammond, G. Jahesh, H. Khan, L. Coombe, R. L. Warren & I. Birol. 2017. ABySS 2.0: resource–efficient assembly of large genomes using a Bloom filter. Genome Research 27: 768–777.

Lemoine, F. & O. Gascuel. 2021. Gotree/Goalign: toolkit and Go API to facilitate the development of phylogenetic workflows. NAR Genomics and Bioinformatics 3.

Little, D. P. 2010. A unified index of sequence quality and contig overlap for DNA barcoding. Bioinformatics 26: 2780–2781.

Due at the start of class February 7.