生信软件安装

刘浩

2023.9.16

总览

- 各种脚本、jar包、编译好的二进制可执行包, e.g. NCBI直接下载编译好的blast
- 采用C/C++等编写, 只有源码，按文档要求编译, e.g. 下载R源码, 自己编译安装
- 系统自带的包管理器
- conda, 生信软件包管理器
- docker、singularity容器镜像
- Python、perl、R等的模块或包

环境变量

`PATH`, 自动搜索安装的可执行文件

`LD_LIBRARY_PATH`, 动态链接库搜索位置

`PERL5LIB`, perl模块位置

`PYTHONPATH`, python模块位置

```
export PATH=/opt/bin/:$PATH  #或写入~/.bashrc文件
```

1. 脚本软件

采用Perl/Python等解释型语言编写, 下载后有解释器即可运行。

用法：系统安装对应的解释器, 添加x权限

优点：下载就可以直接用, 修改方便

缺点：可能需要很多依赖, 性能差

```
$ mv N50.pl ~/opt/bin/
$ export PATH="$PATH:$HOME/opt/bin/" #或写入~/.bashrc文件
```

2. jar包

采用java或类java语言编写, 开发人员已编译打包好, 下载可用。

用法：系统安装java运行环境JDK, `java -jar  package.jar`

优点：下载就可以直接用, 跨平台

缺点： 相比C/C++性能较差, 对内存有一定要求

```
$ cd ~/opt
$ wget http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.39.zip
$ unzip Trimmomatic-0.39.zip
$ java -jar trimmomatic-0.36.jar PE \
  -phred33  input_forward.fq.gz input_reverse.fq.gz \
   output_forward_paired.fq.gz output_forward_unpaired.fq.gz \
   output_reverse_paired.fq.gz output_reverse_unpaired.fq.gz \
   ILLUMINACLIP:/usr/local/src/Trimmomatic/Trimmomatic-0.36/adapters/TruSeq3-PE.fa:2:30:10 \
   LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 HEADCROP:8 MINLEN:36
```

3. 二进制可执行包

采用C/C++等编译语言编写, 开发人员已编译好, 下载可用

用法：下载系统对应版本的二进制软件, 添加x权限

优点：下载就可以直接用, 不依赖编译器

缺点：没法看到源码;不能根据需要预编译;依赖预编译系统的底层库; 跨平台性差

```
$ cd ~/opt
$ mkdir sratoolkit && cd sratoolkit
$ wget http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/2.6.3/sratoolkit.2.6.3-centos_linux64.tar.gz
$ tar zxvf sratoolkit.2.6.3-centos_linux64.tar.gz
$ chmod +x ~/opt/sratoolkit/sratoolkit.2.6.3-centos_linux64/bin/*
$ ~/opt/sratoolkit/sratoolkit.2.6.3-centos_linux64/bin/fastdump -h
```

4. 源码编译

采用C/C++等编译语言编写, 开发人员提供源代码以及安装文档, 用户根据平台自行编译

用法： 安装好编译器以及依赖库，按文档要求编译

优点： 可以指定预编译选项；使用自己系统的依赖库

缺点： 对新手不友好; 很多软件编译步骤复杂; 自己手动解决依赖;

常用编译器：

- GCC，Linux及类Unix系统标准编译器，可处理C、C++、Fortran、Java等语言
- oneAPI，ntel开发的编译器，支持C/C++/Fortran编程语言，支持CPU、FPGA、GPU
- LLVM，支持C、C++、Objective-C等，编译速度快，占用内存小

4.1. 源代码后缀规范

源代码、目标文件等后缀名最好保持统一的规范，便于识别区分。

文件类型	后缀名
C source	.c
C++ source	.C, .cc, .cpp, .cxx, .c++
目标文件	.o
头文件	.h
动态链接库	.so
静态链接库	.a

4.2. c语言源码编译

```
$ gcc -o hello hello.c
```

```
#include <stdio.h>
int main()
{
    printf("Hello world.\n");
}
```

```
#多个源文件同时编译,生成可执行文件sum
$ gcc -o sum main.c function.c
```

```c
// 主程序源文件 main.c
#include <stdio.h>
int main()
{
    int sum=0,r,i;
    for(i=1;i<=10;i++)
    {
    r=function(i);
    sum=sum+r;
    }
    printf("sum is %d\n",sum);
}
```

```c
// 子函数源文件 function.c
int function(int x)
{
    int result;
    result=x*x;         return(result);
}
```

4.3. Makefile

源文件数量非常多、存放在不同目录下、相互之间有各种依赖关系以及先后顺序关系时，需要使用Makefile进行管理。Makefile定义了一系列的规则来指定哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作。

- 软件程序的管理工具
- 定义规则，实现自动化编译
- 处理源代码、目标文件、头文件、库文件等依赖关系
- 根据规则和依赖关系，结合时间戳实现精细化控制

make命令执行 Makefile 中的定义的编译流程。make命令默认读取当前目录 Makefile 或 makefile 文件，也可以用 -f 参数指定 Makefile 文件

4.3.1. configure

大型开源程序通常使用configure脚本生成Makefile，Configure脚本作用：

- 检查编译环境 （数据类型长度(int)，操作系统，CPU平台）
- 检查依赖头文件及库文件
- 设置安装路径
- 设置编译器及编译参数

configure -> make -> make install

Configure常用参数：

- `--prefix=/opt/software` 指定安装路径
- `-h` 查看configure帮助，configure支持选项
- `CC=gcc/icc` 设置c语言编译器
- `CFLAGS=-O2 –funrool-` c编译器参数
- `CXX=g++/icpc` 设置c++编译器
- `CXXFLAGS=-O2` c++编译器参数
- `--with-XXX` 编译时使用XXX包
- `--without-XXX` 编译时不使用XXX包
- `--enable-XXX` 启用XXX特性
- `--disable-XXX` 不启用XXX特性

编译安装samtools

```
$ wget https://github.com/samtools/samtools/releases/download/1.3.1/samtools-1.3.1.tar.bz2
$ tar xvfj samtools-1.3.1.tar.bz2
$ cd samtools-1.3.1
$ ./configure –prefix=/home/username/opt/samtools/1.3.1
$ make
$ make install
$ echo "export PATH=/home/username/opt/samtools/1.3.1:$PATH" >> ~/.bashrc
```

4.3.2. cmake

cmake：跨平台编译工具，生成makefile。其配置文件为CMakeLists.txt。

cmake -> make -> make install

cmake常用参数：

- `-DCMAKE_INSTALL_PREFIX=/opt/software` 指定安装路径
- `-DCMAKE_C_COMPILER=/opt/gcc/bin/gcc` 设置c语言编译器
- `-DCMAKE_CXX_FLAGS ="-O2 -funrool-"` c编译器参数
- `-DCMAKE_CXX_COMPILER = =/opt/gcc/bin/gcc` 设置c++编译器

```
$ tar -xf gromacs-5.1.4.tar.gz
$ cd gromacs-5.1.5
$ mkdir build && cd build
$ cmake ..
$ make -j 20
$ make install
```

5. 系统包管理器

用法：不同的操作系统用法不大一样 yum: RedHat, CentOS, Fedora; apt-get: Ubuntu, Debian; brew:MacOS

优点：简单, 一键搞定；包管理器自己解决软件依赖

缺点：生物信息软件大部分不在包管理器中, 用于安装底层依赖库；需要root权限

```
# ubuntu
sudo apt-get -y install libcurl4-gnutls-dev
sudo apt-get -y install libxml2-dev
sudo apt-get -y install libssl-dev
sudo apt-get -y install libmariadb-client-lgpl-dev

# centos
$ yum search openssl
$ yum install -y openssl-devel
```

6. conda

Anaconda 用于科学计算Python发行版，使用conda 管理包和环境。conda 不仅管理安装python包，还可以是各种其他的应用软件。

优点：不需要root权限；自行解决依赖关系；一键安装, 不需要配置环境

缺点：有些软件conda中没有, 需要自己手动安装；环境混乱、版本管理麻烦

conda 安装软件较慢，可以使用mamba代替，使用方式与conda一直。其配置使用见集群文档 mamba。

conda 配置

```
# 第一步：下载miniconda3
$ wget https://nanomirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/Miniconda3-latest-Linux-$(uname -m).sh

# 第二步：安装miniconda3
$ bash Miniconda3-latest-Linux-x86_64.sh -b -p $HOME/miniconda3

# 第三步：将miniconda3保存到环境路径并启用
$ echo "export PATH=$PREFIX/bin:"'$PATH' >> ~/.bashrc
$ source ~/.bashrc

#第四步：基本配置bioconda，添加清华源镜像
$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/bioconda
$ conda config --set show_channel_urls yes
```

使用conda管理软件包

```
# 搜索需要安装的软件包，获取其完成名字
conda search <package name>

# 安装软件包
conda install <package name>

# 安装特定版本的软件包
conda install <package name>=版本号

# 更新软件包
conda update <package name>

# 移除软件包
conda remove <package name>

# 安装R,及80多个常用的数据分析包，包括idplyr，shiny，ggplot2，tidyr，caret 和 nnet
conda install -c r r-essentials
```

使用conda管理软件环境

通过conda环境，可以实现软件版本管理、流程环境管理。

```
# 创建名为env_name的新环境，并在该环境下安装名为 package_name 的包
$ conda create -n env_name package_name

# 可以指定新环境的版本号，
# 例如：创建python2环境，python版本为2.7，同时还安装了numpy pandas包
$ conda create -n python2 python=2.7 numpy pandas

# 激活 python2环境，通过python -V可以看到是python2.7
$ conda activate python2
(python2)$
```

```
# python2 环境中安装相关包
(python2)$ conda install pandas

# 退出 python2 环境
(python2)$ conda deactivate

# 删除环境
$ conda remove -n env_name --all

# 查看当前存在的虚拟环境
$ conda env list
$ conda info -e
```

7. docker

操作系统之上的虚拟层，提供独立于系统的软件环境；兴起于互联网行业，便于项目开发和交付部署，提高硬件资源利用率。

优点： 简单，对于复杂软件可以一键安装；无需安装任何依赖

缺点： 无法与作业调度软件结合使用；权限要求较高，多用户使用有风险

```
docker pull quay.io/qiime2/core:2021.8
```

8. singularity

HPC集群的容器工具，直接使用docker镜像。使用singularity搭建分析流程，可以在所有机器上运行。

优点： 简单；无需安装任何依赖；安全；可结合作业调度系统；高性能；适应性广

缺点： 软件较少；文件比较大

```
# 从给定的URL下载容器镜像，常用的有URL有Docker Hub(docker://user/image:tag) 和 Singularity Hub(shub://user/image:tag)
$ singularity pull tensorflow.sif docker://tensorflow/tensorflow:latest

# 在容器中执行某个命令
$ singularity exec /share/Singularity/saige_0.35.8.2.sif

# 进入容器
$ singularity  shell /share/Singularity/ubuntu.sif
```

9. R包安装

```
# 从官方源安装，最常见方式
$ >install.packages("ggplot2")

# 指定安装源和安装路径
$ >install.packages("ggplot2", repos = "https://mirrors.ustc.edu.cn/CRAN/",lib="~/opt/Rlib")

# 源码安装
$ R CMD INSTALL /path/rpackage.tar.gz
# 或
$ >install.packages("/path/rpackage.tar.gz", repos = NULL, type = "source")

# 安装指定版本的R包
$ >require(devtools)
$ >install_version("limma", version = "1.8.0")
# 或
$ >install.packages("https://cran.r-project.org/src/contrib/Archive/limma/limma_1.8.10.tar.gz", \
repos=NULL, type="source")
```

```
# bioconductor包安装
$ >install.packages("BiocManager")
$ >BiocManager::install("clusterProfiler")

# 测试包是否正常安装
$ >library(package)

# 其它包常见操作
# 卸载包
$ >remove.packages("package")

# 更新包
$ >update.packages("package")

# 查看R包安装位置
$ >.libPaths()

# 查看已安装包
$ >installed.packages()

# 查看包版本
$ >packageVersion("package")
```

10. perl包安装

```
# CPAN 模块自动安装
# 如果使用系统的cpan，则需要root权限，因此普通用户建议使用cpanm代替
$ cpan -i Bio::SeqIO

# cpanm 推荐
$ cpanm --mirror http://mirrors.163.com/cpan --mirror-only Bio::SeqIO

# 源码安装
$ tar xvzf BioPerl-1.7.5.tar.gz
$ cd BioPerl-1.7.5
$ perl Makefile.PL (PREFIX=/home/opt/perl_modules)
$ make && make install

# 添加环境变量，可写入~/.bashrc
$ export PERL5LIB=$PERL5LIB:/home/opt/perl_modules/lib/site_perl

# 测试perl模块安装正常
$ perl -MBio::SeqIO -e1 或 perldoc Bio::SeqIO
```

如下报错，可能是perl版本冲突导致，即安装perl包的版本和使用perl包的版本不一致，建议将`~/.bashrc`中 perl相关的部分注释掉，然后再测试

```
perl: symbol lookup error: perl5/lib/perl5/x86_64-linux-thread-multi/ 、
auto/Cwd/Cwd.so: undefined symbol
```

11. python包安装

```
# conda
$ conda install  biopython

# pip
$ pip install --prefix=/home/opt/ppython_modules/ biopython
$ pip install --user biopython
$ pip install -r requirements.txt --prefix=/home/opt/ppython_modules/

# 源码
$ git clone https://github.com/madmaze/pytesseract.git
$ python setup.py install --prefix=/home/opt/ppython_modules
$ pip install . --prefix=$PREFIX_PATH

# 测试python模块安装正常
$ python -c "import Bio"

# 添加环境变量，可写入 ~/.bashrc
$ export PYTHONPATH=$PREFIX_PATH:/home/opt/lib/python2.7/site-packages/
```

12. 其它问题

- 软件运行出现`'GLIBCXX_x.x.x' not found`的报错，glibc版本不够，解决见集群文档 glibc
- 软件运行出现`'GLIBCXX_x.x.x' not found`的报错，gcc版本不够，需要加载高版本的gcc，见集群文档 FQA
- 软件运行出现`xxxx.so.1.0: cannot open shared object file: No such file or directory`的库缺失报错，载入对应到库文件，见集群文档 FQA

注意事项

- 生信绝大部分软件都可以使用普通用户安装，不需要root权限。普通用户无法在集群上使用yum、apt等安装软件；
- 避免使用conda一键安装软件，时间长了会导致各种环境问题，直至所有软件不可用，推到重来；
- 软件安装或使用过程中出现问题，最好将 `~/.bashrc` 中无关部分都临时注释掉，避免其它软件的影响；
- 使用软件前看一下集群公共软件库，尽量使用公共软件，一个软件一个环境，出现问题比较好排查；