Research Software on CURC Systems

Building Software from Source

Objectives:

1. Explore CURC compilers and compiler environment variables.
2. Perform a simple source installation.

Estimated time to complete: 15 minutes

Part 1: CURC compilers and environment variables

Get on a compute node with the Alpine `acompile` program and view the Alpine module stack.

```
acompile -n 1
module avail
```

How many compiler modules are available on Alpine?

Explore CURC compiler environment variables.

```
module load intel
module load impi
module load mkl
```

The standard compiler variables FC, CC and CXX are set as appropriate for your compiler/MPI combination. These environment variables reference the Fortran, C, and C++ compilers respectively.

```
echo $FC
echo $CC
echo $CXX
```

In addition, several environment variables are set that may be useful during the compilation process. These variables are prefixed by CURC and may easily be found by searching your environment.

`env | grep CURC`

These environment variables can be passed to `make` or `cmake` and make your life a lot easier by removing the need to type out long absolute paths!

Part 2: Compile a program from source.

We will be installing a common bioinformatics program, Samtools, from source. More info about the program: http://www.htslib.org/

You will grab the Samtools source code from http://www.htslib.org/download/

```
cd /projects/$USER/software
wget
https://github.com/samtools/samtools/releases/download/1.17/samt
ools-1.17.tar.bz2
tar -xf samtools-1.17.tar.bz2
cd samtools-1.17 && ls
```

Can you tell which build system samtools requires? Which module should you load?

```
mkdir -p /projects/$USER/software/install/samtools_1.17
module load gcc
./configure --
prefix=/projects/$USER/software/install/samtools_1.17
```

Are any files created or modified (or not!) during the configure step? Which one(s)?
Hints:

```
ls -lt
stat <file name>
```

Run `make`.

What do you see in `/projects/$USER/software/install/samtools_1.17`?

Finish the installation!

```
make install
```

Was your build successful? How can you tell?

Does the following command work for you? Why or why not?
Hint: Add the samtools bin to PATH!
`samtools --help`

This was meant to be a learning experience, but you should always check the program's website for installation instructions! http://www.htslib.org/download/

Installing Software with Conda

Objectives:

1. Configure your `.condarc` file
2. Create a conda environment and install samtools
3. Run samtools from within the environment

Estimated time to complete: 10 minutes

Step 1: Configure a `.condarc` file

This step is required the first time using Anaconda/Miniconda/Mamba on CURC systems.

Navigate to your home and create a file named `.condarc`. You can use whichever text editor you are most comfortable with (`nano`, `vim`, etc.)

```
cd ~
nano .condarc
```

Enter the following text, save, and exit.

```
pkgs_dirs:
  - /projects/$USER/.conda_pkgs
envs_dirs:
  - /projects/$USER/software/anaconda/envs
```

Confirm that the text was saved.

```
cat .condarc
```

Step 2: Create a conda environment containing `samtools`.

> *NOTE:* Conda environments must be created and run from a compute (not a login) node.`

Load the default anaconda module.
`module load anaconda`

What happened to your prompt? Which anaconda environment are you in?

Create an anaconda environment and install samtools.
`conda create -n samtools_env -c bioconda samtools

Did you get an error message? Hint:

```
conda config --show channels
conda config -h
```

Enter `y` for 'yes' when asked if you want to proceed.

Step 3: Activate the environment and run samtools.

```
conda activate samtools_env
samtools --help
```

What happened to your prompt after you activated `samtools_env` ?

Relevant CURC Documentation

https://curc.readthedocs.io/en/latest/software/python.html

Useful Conda Commands (try if you have time)

```
conda env list  # list all environments

conda list  # list packages in active env

conda env remove -n <envname>  # remove an environment

conda config --show channels  # view configured channels

conda deactivate     # deactivate environment

conda create --name <clonedenv> /  # clone an environment

  --clone <envtoclone>
```

Installing Software With Apptainer (Singularity)

Objectives:

1. Become familiar with basic `singularity` commands.

2. Pull an image from a pre-built container, then run the program from the container.

Estimated time to complete: 10 minutes

Basic `singularity` commands

Load the (default) Singularity module.

```
module load singularity
```

View a list of Singularity commands.

```
singularity --help
```

Look at CURC's collection of pre-build containers. Note that we set the location of the containers when you load the Singularity modules.

```
echo $CURC_CONTAINER_DIR
ls $CURC_CONTAINER_DIR
```

A Singularity Definition File (or "def file" for short) is like a set of blueprints explaining how to build a custom container. It includes specifics about the base OS to build or the base container to start from, software to install, environment variables to set at runtime, files to add from the host system, and container metadata.
More information from the Apptainer user-guide:
https://apptainer.org/docs/user/1.0/definition_files.html

Check out the definition file for the `mach3_build.sif` container.
SIF = **S**ingularity **I**mage **F**ile

```
singularity inspect --deffile
$CURC_CONTAINER_DIR/mach3_build.sif
```

Running programs from a container uses the following syntax:

```
singularity exec <name.sif> <program command> <options>
```

For example:

```
singularity exec $CURC_CONTAINER_DIR/seurat_4.1.0.sif R
```

Pull an image from a pre-built container, then run the program from the container.

We are going to create a containerized version of (you guessed it!) samtools using the Docker image found here: https://hub.docker.com/r/staphb/samtools

Export the cache and tmp dir environment variables. Scratch is a good place for these.

```
export SINGULARITY_CACHEDIR=/scratch/alpine/$USER
export SINGULARITY_TMPDIR=/scratch/alpine/$USER
```

Use the `singularity pull` command to create a `.sif` file from the Docker image.

```
singularity pull samtools.sif docker://staphb/samtools
```

Run samtools from the container. Is it the same version of samtools you got from conda and building from source?

Bonus question: How do `samtools_env`, `samtools.sif`, and the source installation compare in size?