Installing Xyce 7.0 on a CentOS 8 machine

This installation manual can be followed very closely to build Xyce 7.0 on a CentOS 8 machine. This guide is written by closely following the set of instructions given in the building guide for Xyce, but modifications have been necessary.

Key:

- All the code that has to be typed in the terminal is blue.
- Errors are in red.
- Possible corrections are in green.
- Comments are in black.
- Download hyperlinks are in purple.

Prior Installation:

1. Make a backup of your hard drive.
2. Create a partition for the installation of CentOS.
3. Format an USB (at least 8GiB) for creating a bootable drive for CentOS.

Creating a bootable disk:

We would be performing a full installation of CentOS 8, and you can directly download the ISO from **here**. More generally, all required software in this installation guide may be found **here**.

Alternatively you can download CentOS from their repository but be sure to use “DVD1 ISO”. The DVD1 ISO image contains the installer as well as all the packages, and the installation is interactive with a GUI.
Once you have downloaded the ISO we would “dd” the ISO into the USB. I have found that using third party tools like “Rufus” to create a bootable disc can get buggy.

1. Format USB drive and note its name.
2. Depending on your native OS follow the steps below:

   If your native OS is Windows then download Fedora Media Writer from:
   https://github.com/FedoraQt/MediaWriter/releases

   - Connect USB drive to the system.
   - Open Fedora Media Writer.
   - On the main window click on “Custom Image” and choose the recently downloaded CentOS ISO image.
   - From the “Write Custom Image” window, choose the USB drive that you would be using.
   - Click “Write to disk” [Once the writing begins, you might have to wait for some time. Ensure to not unplug the USB during this process].
   - Once the boot media has been creation is completed unmount the USB.

If your native OS is Mac then:

- Connect your USB drive.
- Open terminal and type diskutil list command so you can see the device path and name for your USB drive i.e. [$ diskutil list] and note the name of your USB device. It would usually read as /dev/disk2 .
- Then type diskutil unmountDisk /dev/disknumber of USB command to correctly unmount the USB i.e. [$ diskutil unmountDisk /dev/disk2] where disknumber in the example above will be replaced by disk2.
- Login as root using su command which gives you root privileges and enter your root password.
- Now we can dd the ISO into the USB.
- Type sudo dd if=/path/to/image.iso of=/dev/rdisknumber bs=1m> where you would correctly enter the path of the ISO of CentOS 8 and disknumber is the name of the disk.
- Wait for dd to finish which can take some time. When you see the # prompt, the process is completed and you can remove the USB/ begin installation.

If your native OS is a Linux flavor : (steps almost identical to Mac)

- Connect USB drive.
- Open terminal and type dmesg|tail command. The dmesg command gives a log of all recent activities and you can locate the USB drives name.
- Login as root using su and enter your root password.
- Type dd if=/path/to/image.iso of=/dev/disknumber bs=1m as we did for Mac. Observe that for Mac we used rdisknumber vs disknumber in Linux. /dev/disk* creates a block device file while /dev/rdisk* creates a character device file. You can also use /dev/disk* in Mac but its faster to use /dev/rdisk* .
- Wait for dd to finish which can take some time. When you see the # prompt, the process is completed and you can remove the USB/ begin installation.

Install CentOS:

1. Make sure the USB is plugged in and reboot.
2. Boot from USB and follow the steps suggested by the GUI installer making sure you install CentOS in the partition created. CentOS will end up making two partitions for itself one for / and the other swap.
3. Complete the installation process and update your CentOS 8.

This ends the section of installing CentOS 8 in your machine. We can begin the main installation!

Building Xyce:

This section is about building Xyce 7.0 on a CentOS 8 machine.

If you have CentOS already installed in your machine, and you want to install Xyce you would need to make sure you have sudo rights. The Xyce building guide does not account for the fact that in the latest release of CentOS many packages have migrated to be contained within development tools, and hence you would need to make modifications to the installation steps in given in the building guide.

Open your terminal and then:

1. sudo yum install gcc gcc-c++ gcc-gfortran -y
   gcc, gcc-c++ and gcc-fortran are compliers for c,c++ and fortran
2. sudo yum install blas -y
   blas is Basic Linear Algebra Subprograms which is a low-level set of routines for linear algebra. blas-devel is the development tool for blas
3. sudo dnf --enablerepo=PowerTools install blas-devel -y
4. sudo dnf install make cmake clang -y
   cmake is a tool used for building software and clang is a compiler front end for OpenMp, C, etc.
5. sudo dnf install lapack
   lapack is another library for numerical linear algebra
6. sudo dnf --enablerepo=PowerTools install lapack-devel -y
7. sudo dnf install flex -y
   flex is a parser
8. sudo dnf install fftw -y
9. sudo dnf install fftw-devel -y
   fftw is a library for computing discrete FFTs
10. sudo dnf install suitesparse -y
11. sudo dnf --enablerepo=PowerTools install suitesparse-devel -y
    suitesparse is a library for handling sparse matrices
12. sudo dnf install openmpi -y
13. sudo dnf install openmpi-devel -y
    openmpi is a message parsing interface used for parallelizing instructions

Though not specified in the building guide we will also install python3. As support for python2 ends in 2020, if you have applications that expect to find the python command in the system's path, you'll need to create the unversioned python command and set the default version.

14. sudo dnf install python3 -y

I made sure to set alternatives to the same location.

15. sudo alternatives --set python /usr/bin/python3

This completes the list of all the dependencies required for Trilinos which is required for Xyce. As stated in the building guide Xyce is not guaranteed to build properly for versions different (especially versions after) from Trilinos 12.12.1

Directly download Trilinos 12.12.1 from **here**. Alternatively, download Trilinos 12.12.1 from https://github.com/trilinos/Trilinos/releases/tag/trilinos-release-12-12-1/.

Similarly, directly download Xyce 7.0 from **here** and Xyce Regression suite 7.0 from **here**. Alternatively, provide your details to Sandia and download the Xyce-7.0.tar.gz and Xyce_Regression-7.0.tar.gz files from the Sandia website.

Let us make a folder called Xyce_Install. The path names in this document would be a rough guide and you would have to edit the path variables according to the hierarchy you want/have in your box

16. mkdir /home/alevi/Xyce_Install

Unpack all your tar files in the above folder i.e. Trilinos, Xyce and the Regression suite

There are two variants of Xyce, Serial Xyce and Parallel Xyce. For building Xyce, we would need to make two different builds of Trilinos which we would call Serial Trilinos and Parallel Trilinos.

Serial Trilinos has to be built without MPI enabled while Parallel Trilinos has to be built with MPI enabled.

Let us make the Cmake invocation script for serial Trilinos.

IMPORTANT:

Set the variable SRCDIR to have the full path to the location of the Trilinos source code.

Set the variable ARCHDIR to have the full path to the location where you want Trilinos libraries and headers installed.

The install process will make “include” and “lib” sub-directories. The ARCHDIR variable is used in the CMake invocation to set the “CMAKE_INSTALL_PREFIX” parameter. You would have to remember this because once Trilinos is built, we would need to use the same setting for Xyce ARCHDIR parameter, allowing configure to locate where Trilinos is installed.

There must be no space before or after the equals to sign (=) on these lines

The script below would specify which C,C++ and Fortran compliers to use.

17. vim serial-reconfigure.sh

This will open a text file editor (vim) called serial-reconfigure.sh to which paste the following in it. Make sure there are no spaces or changes:

```
#!/bin/sh
XYCE_HOME_DIR=/home/alevi/Xyce_Install
SRCDIR=$XYCE_HOME_DIR/Trilinos-trilinos-release-12-12-1
ARCHDIR=$XYCE_HOME_DIR/XyceLibs/Serial
FLAGS="-O3 -fPIC"
cmake \
-G "Unix Makefiles" \
-DCMAKE_C_COMPILER=gcc \
-DCMAKE_CXX_COMPILER=g++ \
-DCMAKE_Fortran_COMPILER=gfortran \
-DCMAKE_CXX_FLAGS="$FLAGS" \
-DCMAKE_C_FLAGS="$FLAGS" \
-DCMAKE_Fortran_FLAGS="$FLAGS" \
-DCMAKE_INSTALL_PREFIX=$ARCHDIR \
-DCMAKE_MAKE_PROGRAM="make" \
-DTrilinos_ENABLE_NOX=ON \
 -DNOX_ENABLE_LOCA=ON \
-DTrilinos_ENABLE_EpetraExt=ON \
 -DEpetraExt_BUILD_BTF=ON \
 -DEpetraExt_BUILD_EXPERIMENTAL=ON \
 -DEpetraExt_BUILD_GRAPH_REORDERINGS=ON \
-DTrilinos_ENABLE_TrilinosCouplings=ON \
-DTrilinos_ENABLE_Ifpack=ON \
-DTrilinos_ENABLE_Isorropia=ON \
-DTrilinos_ENABLE_AztecOO=ON \
-DTrilinos_ENABLE_Belos=ON \
-DTrilinos_ENABLE_Teuchos=ON \
 -DTeuchos_ENABLE_COMPLEX=ON \
-DTrilinos_ENABLE_Amesos=ON \
 -DAmesos_ENABLE_KLU=ON \
-DTrilinos_ENABLE_Sacado=ON \
-DTrilinos_ENABLE_Kokkos=OFF \
-DTrilinos_ENABLE_ALL_OPTIONAL_PACKAGES=OFF \
-DTrilinos_ENABLE_CXX11=ON \
-DTPL_ENABLE_AMD=ON \
-DAMD_LIBRARY_DIRS="/usr/lib" \
-DTPL_AMD_INCLUDE_DIRS="/usr/include/suitesparse" \
-DTPL_ENABLE_BLAS=ON \
-DTPL_ENABLE_LAPACK=ON \
$SRCDIR
```

After pasting the text save the file, save it. The protocol is [esc] :wq

This creates the script. We change it to an executable script and execute it using :

18. chmod u+x serial-reconfigure.sh
19. ./serial-reconfigure.sh

If at this point the script completes with no errors, we have configured Serial Trilinos for building it and we can build it using the commands below.

20. make
21. make install

This will finish building **Serial Trilinos**.

If you observe I have created a variable XYCE_HOME_DIR and given the path /home/alevi/Xyce_Install

After building serial Trilinos we redo the same for **Parallel Trilinos** which is building Trilinos for Parallel Xyce. Building parallel Trilinos requires us to enable some more packages and also compiler wrappers used by OpenMPI

22. vim parallel-reconfigure.sh

This will open a text file editor (vim) called serial-reconfigure.sh to which paste the following in it. Again, make sure there are no spaces or changes:

```
#!/bin/sh
XYCE_HOME_DIR=/home/alevi/Xyce_Install
SRCDIR=$XYCE_HOME_DIR/Trilinos-trilinos-release-12-12-1
ARCHDIR=$XYCE_HOME_DIR/XyceLibs/Parallel
FLAGS="-O3 -fPIC"
cmake \
-G "Unix Makefiles" \
-DCMAKE_C_COMPILER=mpicc \
-DCMAKE_CXX_COMPILER=mpic++ \
-DCMAKE_Fortran_COMPILER=mpif77 \
-DCMAKE_CXX_FLAGS="$FLAGS" \
-DCMAKE_C_FLAGS="$FLAGS" \
-DCMAKE_Fortran_FLAGS="$FLAGS" \
-DCMAKE_INSTALL_PREFIX=$ARCHDIR \
-DCMAKE_MAKE_PROGRAM="make" \
-DTrilinos_ENABLE_NOX=ON \
 -DNOX_ENABLE_LOCA=ON \
-DTrilinos_ENABLE_EpetraExt=ON \
 -DEpetraExt_BUILD_BTF=ON \
 -DEpetraExt_BUILD_EXPERIMENTAL=ON \
 -DEpetraExt_BUILD_GRAPH_REORDERINGS=ON \
-DTrilinos_ENABLE_TrilinosCouplings=ON \
-DTrilinos_ENABLE_Ifpack=ON \
-DTrilinos_ENABLE_ShyLU=ON \
-DTrilinos_ENABLE_Isorropia=ON \
-DTrilinos_ENABLE_AztecOO=ON \
-DTrilinos_ENABLE_Belos=ON \
-DTrilinos_ENABLE_Teuchos=ON \
 -DTeuchos_ENABLE_COMPLEX=ON \
-DTrilinos_ENABLE_Amesos=ON \
 -DAmesos_ENABLE_KLU=ON \
-DTrilinos_ENABLE_Sacado=ON \
-DTrilinos_ENABLE_Kokkos=OFF \
-DTrilinos_ENABLE_Zoltan=ON \
-DTrilinos_ENABLE_ALL_OPTIONAL_PACKAGES=OFF \
-DTrilinos_ENABLE_CXX11=OFF \
-DTPL_ENABLE_AMD=ON \
-DAMD_LIBRARY_DIRS="/usr/lib" \
-DTPL_AMD_INCLUDE_DIRS="/usr/include/suitesparse" \
-DTPL_ENABLE_BLAS=ON \
-DTPL_ENABLE_LAPACK=ON \
-DTPL_ENABLE_MPI=ON \
$SRCDIR
```

After pasting the text save the file, save it. The protocol is [esc] :wq

This creates the parallel build script. We change it to an executable script and execute it using :

23. chmod u+x parallel-reconfigure.sh
24. ./parallel-reconfigure.sh

Now that the script has been executed lets invoke make

25. make

At this point I ran into a repeating error which read something like “compiler can’t find mpicc” multiple times which forced me to force the path as follows

```
PATH=/usr/lib64/openmpi/bin/:$PATH
```

Then go back to step 23 and follow the same steps and make the build. If step 25 completes without errors then:

26. make install

This will complete building Parallel Trilinos and our dependencies are complete. We can start building Xyce

27. XYCE_HOME_DIR=/home/alevi/Xyce_Install
28. cd $XYCE_HOME_DIR

We will build Serial Xyce in a directory called XyceSerialBuild

29. mkdir XyceSerialBuild
30. cd XyceSerialBuild

We now make the configure file for Xyce, and use CentOS’s shell continuation mechanism. Note we will have a backslash “ \ “ at the end of each line. Ensure that you do not have any spaces after the backslash and also there is a space before the backslash.

31. [Enter the lines below exactly and do not include the current line within square brackets]
    ```
    $XYCE_HOME_DIR/Xyce-7.0/configure \
    CXXFLAGS="-O3" \
    ARCHDIR="$XYCE_HOME_DIR/XyceLibs/Serial" \
    CPPFLAGS="-I/usr/include/suitesparse"
    ```

This would invoke a configure script which we can then build with .

32. make
33. sudo make install

(I think I used sudo here because I might have opened a new terminal to continue building Serial Xyce after building Serial Trilinos, because I had to spend a long time debugging the build of Parallel Trilinos.)

This should successfully build Serial Xyce and we proceed with building Parallel Xyce. We will build Parallel Xyce in a directory called XyceParallelBuild and follow steps similar to those for Serial Xyce. However the cmake invocation script will be different.

34. cd $XYCE_HOME_DIR
35. mkdir XyceParallelBuild
36. cd XyceParallelBuild
37. [Enter the lines below exactly and do not include the current line within square brackets]
    $XYCE_HOME_DIR/Xyce-7.0/configure \
    CXXFLAGS="-O3" \
    ARCHDIR="$XYCE_HOME_DIR/XyceLibs/Parallel" \
    CPPFLAGS="-I/usr/include/suitesparse" \
    --enable-mpi \
    CXX=mpicxx \
    CC=mpicc \
    F77=mpif77

This would makes configure script for Parallel Xyce. Observe we have enabled mpi and the flags for the same.

38. make
39. sudo make install

At this point, we hope to have successfully built both Parallel and Serial Xyce. Let us make shortcut paths so we can call Xyce easily from the terminal every time when we want to run an execution.

40. sudo ln -s /home/alevi/Xyce_Install/XyceSerialBuild/src/Xyce /usr/bin/XyceS

This would enable you to call Serial Xyce execute a netlist from the terminal by just using XyceS e,g, $ XyceS opamp,cir

Let us proceed to test our build on the Xyce Regression suite. We create a directory to store the results by the name XyceTests

41. cd $XYCE_HOME_DIR
42. mkdir XyceTests/ -p
43. cd XyceTests

44. [Enter the lines below exactly and do not include the current line within square brackets]
    $XYCE_HOME_DIR/Xyce_Regression-7.0/TestScripts/run_xyce_regression \
    --output=$XYCE_HOME_DIR/Xyce_Test \
    --xyce_test="$XYCE_HOME_DIR/Xyce_Regression-7.0" \
    --resultfile=`pwd`/serial_results \
    --taglist="+serial+nightly?noverbose-verbose?klu?fft" \
    XyceS

Hopefully the Regression Suite gets tested and our build of Serial Xyce was clean. There might be a few errors or warnings but the ones I got were associated to the test cases for a parallel build of Xyce.

We repeat the same procedure for the parallel build of Xyce testing it on the Regression Suite.

45. sudo ln -s /home/alevi/Xyce_Install/XyceSerialBuild/src/Xyce /usr/bin/XyceS

The shortcut for Parallel Xyce is XyceP from here on now.

46. cd $XYCE_HOME_DIR
47. mkdir XyceTests/ -p
48. cd XyceTests
49. [Enter the lines below exactly and do not include the current line within square brackets]
    $XYCE_HOME_DIR/Xyce_Regression-7.0/TestScripts/run_xyce_regression \
    --output=$XYCE_HOME_DIR/Xyce_Test --
    xyce_test="$XYCE_HOME_DIR/Xyce_Regression-7.0" \
    --taglist="+parallel+nightly?noverbose-verbose-klu?fft" \
    --resultfile=`pwd`/parallel_results \
    "mpirun -np 2 XyceP"
50. Xyce should now be fully built. Additional documentation may be found **here**.