Installation Instructions for WRFV4 on Optimum

Updated: January 2023

Preliminaries

Bash cheatsheet: https://devhints.io/bash

- `ls`
  - List current directory contents
  - `ls -la`
    - List current directory and hidden files (-a) in a list format (-l)
- `cd`
  - Change directory
- `cp`
  - Copy file
  - `cp ./file ./new_location/file`
    - Copies file from current directory to new_location
  - `cp -r ./directory1 ./new_location/directory2`
    - Copies directory1 and all of its contents (-r) to new_location; directory2 is a copy of directory1 and its contents located within new_location

- `mv`
  - Moves file; can be used to rename
  - Deletes original
- `rm`
  - Removes file

- `ln -s`
  - Creates softlink (i.e. shortcut) to directory or file
  - `ln –s /path/to/file1 ./link1`
    - Softlinks file1 to current directory, named as link1
    - If you edit file1's contents, link1 will also be edited
    - If you edit link1's contents, file1 will also be edited
    - If you remove file1, link1 will be a dead link (points to nothing)
    - If you remove link1, file1 will be preserved

Vim cheatsheet: https://devhints.io/vim

- `vi`
  - On command line → opens Vim editor
  - `vi file1.txt`
    - Opens file1.txt in Vim

- All navigation and editing done on the keyboard
- Command mode <esc key>
    - Issue commands, not edits
    - Vim starts up with command mode
    - <esc>
        - Enters command mode
    - :q (then hit <enter>; same for all commands prefixed with :)
        - (Quit) Quits vim
    - :w
        - (Write) Saves file
    - :wq
        - Saves file and quits
    - :q!
        - Force quit (i.e. quits with unsaved changes; useful if you messed something beyond repair but haven't saved)
    - :u
        - Undo
    - <Ctrl + d>
        - (Page down) Jumps down several lines
    - <Ctrl + u>
        - (Page up) Jumps up several lines
    - :20
        - Jumps to line 20 (or 1, or 299, or 2424991, etc.)
    - <shift + 4>
        - Jumps to end of line
    - <0>
        - Jumps to start of line
    - <i>
        - (Insert) Enters insert mode for editing

Install libraries

- Log onto optimum
  - `ssh username@optimum.eos.ubc.ca`
- WRF minimum required libraries
  - NetCDF
    - Technically already installed on optimum, but we want a newer version than available, plus it's good practice to do it yourself anyway from source
    - From source = downloading source code, and running the installation manually on the command line
  - Jasper
    - Also already installed; optimum's version is fine. Also kind of tough to install on your own anyway

- Make directory to place netcdf files and go in
  - `mkdir netcdf`
  - `cd netcdf`

- Before we do any installations, let's load up some required tools. First, check to see what modules are available on optimum
  - `module avail`
  - Modules are collections of environment (background) variables defining paths to libraries and executables
  - We need compilers to do installations

- We need GNU compilers (for installing serial libraries, like NetCDF) and MPI tools (for installing parallel libraries and running parallel code, like WRF). The modules must be loaded in the following order:
  - `module load GCC/8.3/0`
  - `module load OpenMPI/4.0.0/GCC/8.3`
    - **The modules above must be loaded each time you plan on running WRF; you can add these commands to your ~/.bashrc file so that they're loaded on startup**

- Now we need to define some environment variables so that NetCDF knows exactly which compilers we're using, and where we're doing the installs:
  - `export NETCDF=`pwd``
    - (note the backticks `` for pwd) Sets variable NETCDF to current working directory (pwd); causes netcdf to install here instead of in the default location /usr
  - `export CC=gcc`
  - `export FC=gfortran`
  - `export F77=gfortran`
  - `export CXX=g++`

        - The exports above specify which compilers we're using for each language (i.e. C, Fortran (90+ and 77), C++)
    - `export CPPFLAGS="-I${NETCDF}/include"`
    - `export LDFLAGS="-L${NETCDF}/lib"`
        - These two exports tell NetCDF to also factor in additional libraries in ${NETCDF}/lib and header files in ${NETCDF}/include than the ones on default paths
        - These directories don't exist yet right now, but they will soon

- Download netcdf tar files from the UCAR website:
    - wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-c-4.7.3.tar.gz
        - Downloads netcdf-c (C interface for netcdf)
    - wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-fortran-4.5.2.tar.gz
        - One line above
        - Downloads netcdf-fortran (Fortran interface for netcdf; requires netcdf-c)
- Untar netcdf-c, and cd into the created directory
    - `tar -xvzf netcdf-c-4.7.3.tar.gz`
    - `cd netcdf-c-4.7.3`
- Install netcdf-c
    - `./configure --prefix=$NETCDF --disable-dap --disable-netcdf4 --disable-shared`
        - Configures netcdf-c installation with the following options:
            - prefix: installs into $NETCDF
            - disable-dap: disables extraneous libraries for cross-network/filesystem operations
            - disable-netcdf4: disables extraneous compression libraries and formatting; netcdf4 is nice, but actually makes WRF more difficult to install, and really not necessary
            - disable-shared: builds static libraries, not shared libraries (slightly better performance, but takes up more space)
    - `make`
        - Builds code
    - `make check`
        - Checks built code to see if they behave as expected
    - `make install`
        - Installs into proper locations in $NETCDF
- Leave directory
    - `cd ..`
        - You should be back in $NETCDF (i.e. ~/netcdf), and if you `ls`, you should see new directories bin, include, lib

        - **bin**: binaries (executables); tools installed by netcdf-c to operate on .nc files
        - **include**: include/header files containing variables and parameters used by netcdf for further installations
        - **lib**: libraries (compiled source code) used by netcdf for further installations

- Tell netcdf-fortran to look for netcdf-c libraries in the right place
    - `export LIBS="-lnetcdf"`
        - Searches $NETCDF/lib for netcdf-c libraries (libnetcdf.a, libnetcdf.la)
- Untar netcdf-fortran, and cd into the created directory
    - `tar -xvzf netcdf-fortran-4.5.2.tar.gz`
    - `cd netcdf-fortran-4.5.2`

- Install netcdf-fortran
    - `./configure --prefix=$NETCDF --disable-shared`
    - `make`
    - `make check`
    - `make install`

- Return home
    - `cd ~`

Install WRF

- Make a new directory to install your WRF build, and go into it
  - `mkdir WRF`
  - `cd WRF`

- Clone the WRF directory from GitHub, and cd into the created directory
  - `git clone https://github.com/wrf-model/WRF.git`
  - `cd WRF`

- Configure the WRF source code to accept the correct compilers
  - `./configure`
    - Ensure that NETCDF and perl are found at the top of the printout; you can ignore everything else
  - 34 <hit enter>
    - We're using pure GNU compilers (gfortran/gcc), and running distributed memory (dmpar) MPI, so we choose 34 and hit enter
  - 1 <hit enter>
    - We want basic nesting, not moving nests

- Compile the WRF-ARW core support real-data cases (em_real)
  - `./compile em_real`

- Check that the required executables have been created
  - `ls main/*.exe`
    - Should see ndown.exe, real.exe, tc.exe, and wrf.exe

Install WPS

- Go back to the top level WRF directory
  - `cd ..`

- Clone the WPS directory from GitHub, and cd into the created directory
  - `git clone https://github.com/wrf-model/WPS.git`
  - `cd WPS`

- We need to tell WPS where to find Jasper libraries and header files for Grib-supported I/O
  - `export JASPERLIB=/usr/lib64`
  - `export JASPERINC=/usr/include`

- Now configure WPS…
  - `./configure`
  - 1 <hit enter>
    - We want a GNU-compiled serial-build of WPS with Grib2 support

- …and compile!
  - `./compile`

- Check that the required executables have been built
  - `ls *.exe`
    - Should see geogrid.exe, metgrid.exe, and ungrib.exe