ECMWF Environment on the CRAY

Cristian Simarro

Cristian.Simarro@ecmwf.int

User Support Section

Outline

- Shells
- Filesystems
- Modules
- Practicals

Shells

- Only ksh and bash are supported for interactive shells.
  - use changesh from ecgate to change your login shell if you wish to do so.
- Use the following files to customise your environment:
  - ~/.user_profile for environment variables
  - ~/.user_kshrc or ~/.user_bashrc for shell aliases

Filesystems

File System	Suitable for	Quota
$HOME	permanent files, e. g. .profile, utilities, sources, libraries	512 MB / 22000 files
$PERM	permanent files without the need for automated backups, e. g. smaller input files for model runs, std output etc.	27 GB / 210000 files
$SCRATCH	all temporary (large) files	No quota, 1 month retention
$SCRATCHDIR	data to be automatically deleted at the end of a job	(part of $SCRATCH)

```
trx@ccb-login3:/home/ectrain/trx> quota

Quota for $HOME and $PERM:
Disk quotas for user trx (uid 414):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
cnasa1:/vol/home
                  10144    480M    500M             338   20000   22000
Disk quotas for user trx (uid 414):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
cnasa2:/vol/perm
                      0  26624M  27648M               1    200k    210k
```

Filesystems

- Only $HOME is backed up
- $HOME, $SCRATCH and $PERM on ecgate and HPCF are different
- Different select/delete policies may apply on $SCRATCH
- All critical files from file systems other than $HOME should be copied to ECFS without delay.
- To transfer files between different platforms (e.g. ecgate – HPCF) use scp or rsync

Modules framework

- Utility to manage and control the user environment
  - Software packages and programs
  - Compiler flags and libraries
- Set of commands starting by **module**
- Modules will automatically change values of variables like PATH, MANPATH, LM_LICENSE_FILE ... etc

One command to rule them all

Why are they important?

- If a module is not loaded or the wrong version is loaded:
  - Your job might fail: “command not found”
  - The compilation of software might fail because it won’t find the required libraries.
  - The compilation of software may work, but it might produce binaries linked with undesired versions of libraries.

Be aware of the environment before you start working to avoid surprises…

```
$> module list
```

What is in modules?

- Cray stuff:
  - Cray Programming Environment
  - Compilers
  - Various CRAY packages and libraries
- ECMWF stuff:
  - ECMWF software packages
    - ecCodes, ECFS client, ecflow …
  - 3rd party packages and libraries
    - fftw, ...

```
$> module avail
```

Main actions

- See what is loaded and what is available to load

```
$> module list
$> module avail
```

- Load and unload a module

```
$> module load package
$> module load package/version
$> module unload package
```

- Switch an already loaded module by another one

```
$> module switch package/version1
$> module switch package/version1 package/version2
```

Load example

```
usxa@ccb-login3:~> cdo -V
If 'cdo' is not a typo you can run the following command to lookup the package that contains the binary:
    command-not-found cdo
-bash: cdo: command not found
usxa@ccb-login3:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.4                         17) atp/2.0.0
  2) eswrap/1.3.3-1.020200.1278.0             18) PrgEnv-cray/5.2.82
  3) switch/1.0-1.0502.60522.1.61.ari         19) pbs/13.0.403.161593
  4) cce/8.4.5                                20) craype-broadwell
  5) craype-network-aries                     21) cray-mpich/7.3.2
  6) craype/2.5.3                             22) cdt/16.03
  7) cray-libsci/16.03.1                      23) verbose/false
  8) udreg/2.3.2-1.0502.10518.2.17.ari        24) ecfs/2.2.1-rc2(prodn:default)
  9) ugni/6.0-1.0502.10863.8.29.ari           25) jasper/1.900.1(default)
 10) pmi/5.0.10-1.0000.11050.0.0.ari          26) grib_api/1.17.0(default)
 11) dmapp/7.0.1-1.0502.11080.8.76.ari        27) fftw/3.3.4.7
 12) gni-headers/4.0-1.0502.10859.7.8.ari     28) emos/443-r64(default)
 13) xpmem/0.1-2.0502.64982.5.3.ari           29) sms/4.4.13(default)
 14) dvs/2.5_0.9.0-1.0502.2188.1.116.ari      30) batch_utils/1.6(default)
 15) alps/5.2.4-2.0502.9774.31.11.ari         31) verbose/true(default)
 16) rca/1.0.0-2.0502.60530.1.62.ari
usxa@ccb-login3:~> module avail cdo

----------------- /usr/local/apps/modulefiles/tools_and_libraries/data_formats -----------------
cdo/1.6.1(default) cdo/1.6.4          cdo/1.7.0          cdo/1.7.2
usxa@ccb-login3:~> module load cdo
load cdo 1.6.1 (PATH, CDO_DIR, CDO)
usxa@ccb-login3:~> cdo -V
Climate Data Operators version 1.6.1 (http://code.zmaw.de/projects/cdo)
...
```

Switch example

```
usxa@ccb-login3:~> grib_ls -V
grib_api Version 1.17.0
usxa@ccb-login3:~> module switch grib_api/1.19.0
switch1 grib_api 1.17.0 (PATH, MANPATH, GRIB_API_DIR, GRIB_API_VERSION, GRIB_API_INCLUDE, GRIB_API_LIB, GRIB_API_INCLUDE_DIR,
GRIB_API_LIB_DIR)
switch2 grib_api 1.19.0 (PATH, MANPATH, GRIB_API_DIR, GRIB_API_VERSION, GRIB_API_INCLUDE, GRIB_API_LIB, GRIB_API_INCLUDE_DIR,
GRIB_API_LIB_DIR)
usxa@ccb-login3:~> grib_ls -V
grib_api Version 1.19.0
usxa@ccb-login3:~> module switch grib_api eccodes
switch1 grib_api 1.19.0 (PATH, MANPATH, GRIB_API_DIR, GRIB_API_VERSION, GRIB_API_INCLUDE, GRIB_API_LIB, GRIB_API_INCLUDE_DIR,
GRIB_API_LIB_DIR)
switch2 eccodes 2.0.0 (PATH, MANPATH, ECCODES_DIR, ECCODES_VERSION, ECCODES_INCLUDE, ECCODES_LIB, ECCODES_INCLUDE_DIR,
ECCODES_LIB_DIR)
usxa@ccb-login3:~> grib_ls -V
ecCodes Version 2.0.0
```

Advanced options

- See what a module would do (without loading it):

```
usxa@cct-login:~> module show emos
-------------------------------------------------------------------
/usr/local/apps/modulefiles/tools_and_libraries/ecmwf/emos/443-r64:

prepend-path     PATH /usr/local/apps/libemos/000443/CRAY/84/bin
add-dependency   fftw/3.3.4.7    not cascading unload
setenv           EMOS_DIR /usr/local/apps/libemos/000443/CRAY/84
setenv           EMOS_VERSION 443
setenv           EMOS_LIB -L/usr/local/apps/libemos/000443/CRAY/84/lib ...
setenv           EMOSLIB -L/usr/local/apps/libemos/000443/CRAY/84/lib ...
setenv           EMOS_LIB_DIR /usr/local/apps/libemos/000443/CRAY/84/lib
prepend-path     PKG_CONFIG_PATH /usr/local/apps/libemos/000443/CRAY/84/lib/pkgconfig
prepend-path     PE_PKGCONFIG_LIBS emosR64
module-whatis    Set environment variables to enable the usage of the emos 443. This package is integrated with
the CrayPE by default.
```

Integration with the Cray Programming Environment

- Cray compiler wrappers (**cc**, **CC** and **ftn**) are heavily affected by modules:
  - The real compiler (Cray, GNU or Intel)
  - The target architecture
  - The compiler flags and libraries used
- The libraries provided by Cray and some ECMWF packages and 3rd party software are also integrated by default with the CrayPE
  - grib_api, emos, eclib, netcdf, nag, gsl …

Integration with the Cray Programming Environment

- Be careful when changing the backend compilers...
  - Use prgenvswitchto
  - Manually reload all ECMWF after the switch. Example witch to GNU compilers:

```
usxa@ccb-login3:~> prgenvswitchto gnu
remove jasper ...
remove grib_api ...
remove emos ...
switch PrgEnv-cray PrgEnv-gnu
load jasper ...
load grib_api ...
load emos ...
```

  - Special flag to show what the wrapper is doing:

[cc,CC,ftn] -craype-verbose

Let’s play…

Preparation

- Use our script to generate keys to access without password

1. Open a local terminal
2. Execute

```
$> /home/ectrain/trx/bin/create_keys.sh
```

Let’s play…

- Start a **fresh** session on ccb, and untar the example tarball:

```
$> ssh trcrayX@ccb
$> tar xvzf ~trx/modules-example.tar.gz
$> cd modules-example
```

- Have a look at the sample program version.c
- Compile with:

```
$> make
```

Did it work? Why?

What do you need to do to build the program?

Let’s play again…

- Once compiled, you can run it:

```
$> ./version
ECCODES VERSION: 2.0.0
NETCDF VERSION: 4.3.2 of Mar 17 2016 14:52:41 $
```

- What would you do to get the following result:

```
$> ./version
ECCODES VERSION: 2.0.2
NETCDF VERSION: 4.3.2 of Mar 17 2016 14:52:41 $
```

Note: to rebuild the program:

```
$> make clean && make
```

Bonus exercise

- Now change the PrgEnv to use the Intel compilers, and rebuild:

```
change compiler to intel
$> make clean && make
```

Did it work?

Use 'ldd *version*' to check that everything went fine

Questions?