Installing Software and Writing Modules (II)

Module and Module Files

The Command `module`

- convenient mechanism to configure your *environment*,
- reads a file, the *module file*, that holds instructions,
- a shell independent way to configure your environment:
  - *same* module file whether `sh/bash` or `csh/tcsh`.

Examples

- We provide module files, users can write their own.
  - look at all the module files we wrote,
  - they can be found in `/share/apps/modulefiles/`

Module File Syntax and Concepts

Special Instructions

- Instructions to configure your environment:

```
prepend-path PATH /location/of/the/code
setenv       BASE /scratch/demo
set-alias    crunch "crunch --with-that-option \*"
```

Syntax

- Module files can be complex, using `tcl` language
  - you **do not** need to know `tcl` to write module files.

Simple or Complex

- A simple module file can just list the modules that must be loaded to run some analysis.
- Can write complex module files and leverage `tcl`.

Example of `module` Commands

Basic

	Info	Config	Details
`module`	`avail`	`load`	`list`
`module`	`whatis`	`unload`	`help <name>`
`module`	`whatis <name>`	`swap`	`show <name>`

More help

```
man module
```

A Simple Module File

Example

```
#%Module1.0
#
# load two modules and set the HEASOFT env variable
module load gcc/10.1.10
module load python/3.8
setenv HEASOFT /home/<username>/heasoft/6.3.1
```

Replace `<username>` by your username.

Example of More Elaborate and Complex Module Files

Will be illustrated in the hands-on section.

Module Files Organization

Recommended Approach

- use a central location under you home directory `~/modulefiles`,
- use a tree structure
- use version numbers if/when applicable,
- let `module` know where to find the module files.

Customization/Examples

Tree structure

```
~/modulefiles/crunch/
~/modulefiles/crunch/1.0
~/modulefiles/crunch/1.2
~/modulefiles/crunch/2.1
~/modulefiles/crunch/.version
~/modulefiles/viewit
```

Define a Default Version

An optional file `.version` can be used to set the default version:

```
#%Module1.0
set ModulesVersion "1.2"
```

Hence

```
module load crunch
module swap crunch/2.1
```

Customization (cont'd)

Let `module` Know Where to Find the Module Files

```
module use --append ~/modulefiles
```

Either

1. in your initialization file `~/.bashrc` or `~/.cshrc`
2. or better yet in a `~/.modulerc` file

```
#%Module1.0
# adding my own module files
module use --append /home/username/modulefiles
```

Hands-on Section

Hands-On

In the hands-on portion of the workshop you will

- Build and install software using best-practices,
  - trivial case,
  - simple/didactic example,
  - somewhat complex examples.
- Write simple and more elaborate module files.
- Run the software you installed in jobs.

But first, log in to Hydra

- If you need a reminder about how to log into Hydra and how to change your password, check the *Intro to Hydra* tutorial.
  - If the link does not work:

```
https://github.com/SmithsonianWorkshops
  > Hydra-introduction
    > hydra_intro.md
```

Let's pause here for 5-10 minutes

Switch to `github` for the Hands-on

Go to

https://github.com/SmithsonianWorkshops/advanced-hydra-workshops/

Convention

- I use % as prompt
  - your prompt might be different, like $
  - you type what is **after** the prompt
  - no prompt: result from previous command.
- I where you see `<genomics|sao>`, you need to use either `genomics` or `sao`,
- I where you see `<username>`, you need to substitute your username.