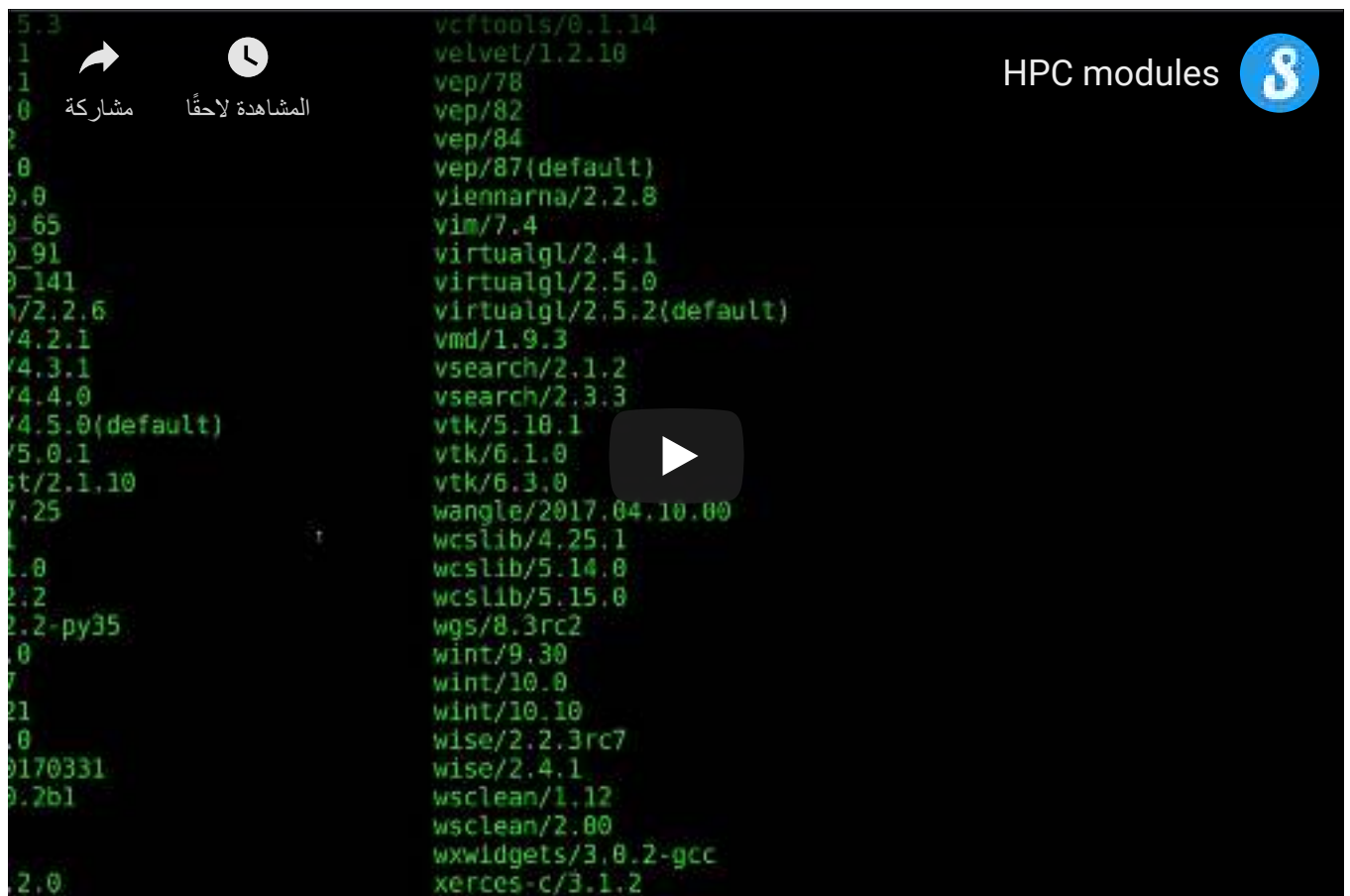# HPC software modules

On a HPC system, the user `environment` is setup using environment `modules`. By default, a number of modules are automatically loaded to configure the environment to allow running of applications and the submission of jobs to the cluster.

What is an environment `Module`?

On a HPC system, it is necessary to make available a wide choice of software packages in multiple versions, it can be quite difficult to set up the user environment so as to always find the required executables and libraries.

This is particularly true where different implementations or versions use the same names for files. Environment modules provide a way to selectively activate and deactivate modifications to the user environment which allow particular packages and versions to be found

particular packages and versions to be found.

The basic command to use is module:

```
module
   (no arguments)         // print usage instructions
   avail or av            // list available software modules
   whatis                 // as above with brief descriptions
   load <modulename>      // add a module to your environment
   unload <modulename>    // remove a module
   purge                  // remove all modules
```

Modules work by setting environment variables such as `PATH` and `LD_LIBRARY_PATH`. Therefore if you need to modify variables directly it is essential to retain the original values to avoid breaking loaded modules (and potentially rendering essential software "not found") - e.g. do the following:

```
export PATH=$PATH:/home/abc123/custom_bin_directory
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/abc123/custom_lib_directory
```

But do not do:

```
export PATH=/home/abc123/custom_bin_directory // overwrite PATH
export LD_LIBRARY_PATH=/home/abc123/custom_lib_directory // overwrite LD_LIBRARY_PATH
```

Some modules refer to **administrative software** and are not of interest to users and also some modules load other modules. It is possible to make use of various versions of **Intel compiler** and parallel libraries by explicitly loading some of the above modules.

By default the login environment loads several modules required for the normal operation of the account: e.g. the default versions of the **Intel compilers** and **Intel Math Kernel Library**, batch scheduling system and the recommended `MPI` for the particular flavour of compute node hardware.

It is possible to change the environment which is loaded when logging in, by editing the shell initialisation file `~/.bashrc` (or `~/.cshrc`, `~/.tcsh` if using csh/tcsh as your shell). If your shell initialisation file is modified it will only effect all future login sessions, and all batch jobs not yet started. Since some modules are required for proper operation of the account, caution is needed before removing any autoloaded modules.

One can list the modules actually loaded by issuing the command `module`

`list` . Which produce the following example output:

```
module list
Currently Loaded Modulefiles:
    1) kvm/3.4.5+6              4) switcher/1.0.13
    2) default-manpath/1.0.1   5) oscar-modules/1.0.5
    3) torque-pbs/2.3.7
```

Using modules in batch files

To use software package xxx in a batch job script it is a simple matter of specifying `module load xxx` See the appropriate packages entry in the software page (and the output of the `module list`) for details of the expected package name for this flag.

Creating your own modules

Apart from the available system environment modules, you can define your own modules and load `module load use.own` . This will set up the `$HOME/privatemodules` directory with an initial module file called null. It will also change your `MODULEPATH` environment variable to ensure that the module command looks for the modules in your home directory. See `man modulefile` for further information on writing your own modules.