Mendel Environment Guide

Overview

- Overview
- Environment Initialization Scripts
    - /etc/profile.d and env_init.sh
    - Configuration
    - Components
    - Installation and Automatic bootstrapping
        - Manual
        - Automated
- Modules and Lmod
- Bash modifications

In order to provide Users with the best possible experience GMI has setup a lot of different tools that make use of the UNIX environment and mask what would usually be complex tasks with easy to understand commands or automatic abstractions. These tools - like Lmod - need to be properly configured for the environment, set with appropriate network paths, checked for conflicts with other software, environment settings and loaded upon success of the former.

Environment Initialization Scripts

IT-Services wrote a meta framework for login task automation and environment initialization which should be easy to understand and provide for a modular and extensible framework on which HPC-site users can build upon. The script set is located and maintained in our git repository (http://stash.gmi.oeaw.ac.at/projects/HPC/repos/scripts/browse/env_init). The Linux environment already features most of the magic needed for this to work.

`env_init` (the actual project name) consists of an `/etc/env_init` directory in which there are two subdirectories at the moment; `/etc/env_init/components.d` and `/etc/env_init/conf.d` as well as a startup script which imports modular components and configuration from those directories. The startup script is located in `/etc/profile.d` and is named `env_init.sh` - any software that features it's own `profile.d` script should be linked against the startup script and implemented as a component of `env_init`. The reason for this abstraction is ease in maintenance and modification as well as having a central place in which the whole environment can be configured without much work. These scripts are deployed on Cluster and Login nodes as well as on Workstations at GMI. They should be highly portable, easy to configure and extend upon and could also be of use to other HPC as well as education sites.

/etc/profile.d and env_init.sh

Linux automatically loads shell scripts which reside in `/etc/profile.d` through a for loop in `/etc/profile` which searches for any files that end in `.sh`. These scripts are loaded upon login to the system.

`env_init.sh` consists of only a couple of directory checks and two loops that load files from `/etc/env_init/conf.d` and `/etc/env_init/components.d`. Files in the latter directory are matched against a list supplied in `/etc/env_init/conf.d/defaults.conf` which specifies components that should be loaded upon login by `env_init`.

Configuration

An example configuration might look like this:

/etc/env_init/conf.d/defaults.conf

```
# CLUSTER ENVIRONMENT CONFIGURATION

# add or remove components you won't need (some are dependencies for others)
# e.g.: switch modules_classic for modules_lmod. remove easybuild or add it.
#      only one distro? edit the distro script or remove it as needed.
#      ..and so forth.
# components may have an explicit order in which they should be loaded, i.e.
# if they depend on another component to be loaded first - thus they are nubmered
# lower numbers get sourced first, higher numbers later.
load_components="10-distro 20-easybuild 22-modules_classic 22-modules_lmod
99-environment"

modules_classic_init=/usr/share/Modules/init/bash

load_lmod=1 # set either to 1 or 0
modules_lmod_init=/usr/lmod/lmod/init/profile

# see also: components.d/99-environment
```

Variables set in this configuration file (it's also possible to create Site/Cluster-Environment specific configurations as `default.local`) are used in the various components of `env_init`. Configuration examples include which components should be loaded, the standard path to Modules and Lmod init shell scripts (otherwise placed in `/etc/profile.d`), which of the former to load (there can only be one Module system active in the environment) and other variables used by components.

Components

`/etc/env_init/components.d` holds a set of shell scripts called *components* (instead of *modules* to prevent confusion with the Lmod and Environment-Modules projects). Some tasks like getting the actual Linux distribution name on which other components build upon need to be figured out first - hence components are named and loaded in acceding order by the `env_init.sh` script in `/etc/profile.d` upon login.

There are currently five components which are shipped out-of-the box by GMI.

Component	Purpose
10-distro	Figures out on which Linux distribuion we're operating on by searching through `/etc/*release` files
20-easybuild	Set up a proper build environment for EasyBuild
22-modules_classic	Loads classic Modules
22-modules_lmod	Loads Lua Modules - Lmod
99-environment	Sets site specific environment variables via the `export` shell builtin. This is where most of the magic happens and network paths as well as CPU types, Scratch storage locations and the like are set. **This file also loads a standard set of site-specific Modules that should be available to users upon login.**

Example:

/etc/env_init/components.d/99-environment

```
#!/bin/sh
# environment variables to be exported
# SGI ICE-X / VIS / Workstation specific environment settings
# guidelines see: https://hpcbios.readthedocs.org/en/latest/HPCBIOS_2012-98.html
# HPC site
export BC_HPC_SITE=GMI
# assign predefined distribution string
export BC_LINUX_DIST=$(distroname)
# path for distribution/site specific mirroring
export MIRRORPATH=/net/gmi.oeaw.ac.at/software/${BC_LINUX_DIST}/mirror/
# currently undefined environment variables
#export TMPDIR=nil # set by PBS Pro
#export BC_MPI_TASKS_ALLOC and BC_MPI_ALLOC via user script
#export JAVA_HOME=nil # set by user
#export HOME # set by system/ldap/...
# reflect three env_init deployments atm.: mendel, vis, linux workstation:
case `dnsdomainname` in
 *ice*|*mendel*)
  export BC_CORES_PER_NODE=16
  export BC_MEM_PER_NODE=65536
  export BC_CPUTYPE=intel-x86_64-sandybridge-avx
  export BC_HPC_SYSTEM=Mendel
  export SCRATCH=/lustre/scratch
  export WORK=$SCRATCH/users/$USER/
```

```
 export WORKDIR=$SCRATCH/users/$USER/
 export SAMPLES_HOME=/net/gmi.oeaw.ac.at/software/samples
 export SOFT_HOME=/net/gmi.oeaw.ac.at/software/mendel/${BC_CPUTYPE}
 # source module environment (if not logged in as build user)
 if [ $USER != "easybuild" ]; then
  export MODULEPATH="${SOFT_HOME}/../syscommon:\
         ${SOFT_HOME}/modules/vis:\
         ${SOFT_HOME}/modules/tools:\
         ${SOFT_HOME}/modules/toolchain:\
         ${SOFT_HOME}/modules/system:\
         ${SOFT_HOME}/modules/numlib:\
         ${SOFT_HOME}/modules/mpi:\
         ${SOFT_HOME}/modules/math:\
         ${SOFT_HOME}/modules/lib:\
         ${SOFT_HOME}/modules/lang:\
         ${SOFT_HOME}/modules/base:\
         ${SOFT_HOME}/modules/devel:\
         ${SOFT_HOME}/modules/debugger:\
         ${SOFT_HOME}/modules/data:\
         ${SOFT_HOME}/modules/compiler:\
         ${SOFT_HOME}/modules/bio"
 fi
 # load standard environment
 module load std_env
;;
*vis*)
 export BC_CORES_PER_NODE=32
 export BC_MEM_PER_NODE=258258
 export BC_CPUTYPE=intel-x86_64-sandybridge-avx
 export BC_HPC_SYSTEM=VIS
 export SCRATCH=/lustre/scratch
 export WORK=$SCRATCH/users/$USER/
 export WORKDIR=$SCRATCH/users/$USER/
 export SAMPLES_HOME=/net/gmi.oeaw.ac.at/software/samples
 export SOFT_HOME=/net/gmi.oeaw.ac.at/software/mendel/${BC_CPUTYPE}
 # source module environment (if not logged in as build user)
 if [ $USER != "easybuild" ]; then
  export MODULEPATH="${SOFT_HOME}/../syscommon:\
         ${SOFT_HOME}/modules/vis:\
         ${SOFT_HOME}/modules/tools:\
         ${SOFT_HOME}/modules/toolchain:\
         ${SOFT_HOME}/modules/system:\
         ${SOFT_HOME}/modules/numlib:\
         ${SOFT_HOME}/modules/mpi:\
         ${SOFT_HOME}/modules/math:\
         ${SOFT_HOME}/modules/lib:\
         ${SOFT_HOME}/modules/lang:\
         ${SOFT_HOME}/modules/base:\
         ${SOFT_HOME}/modules/devel:\
         ${SOFT_HOME}/modules/debugger:\
         ${SOFT_HOME}/modules/data:\
         ${SOFT_HOME}/modules/compiler:\
         ${SOFT_HOME}/modules/bio"
 fi
 # load standard environment
 module load std_env
;;
*) # assume workstation
 export BC_CORES_PER_NODE=nil
```

```
export BC_MEM_PER_NODE=nil
export BC_CPUTYPE=nil
export BC_HPC_SYSTEM=LWS
export SCRATCH=nil
export WORK=nil
export WORKDIR=nil
export BC_CPUTYPE=nil
export SOFT_HOME=/net/gmi.oeaw.ac.at/software/${BC_LINUX_DIST}
# source module environment (if not logged in as build user)
if [ $USER != "easybuild" ]; then
 export MODULEPATH="${SOFT_HOME}/modules/base:\
        ${SOFT_HOME}/modules/math:\
        ${SOFT_HOME}/modules/bio:\
        ${SOFT_HOME}/modules/compiler:\
        ${SOFT_HOME}/modules/lib:\
        /usr/local/Modules/modulefiles/Linux:\
        /usr/local/Modules/modulefiles/Core"
fi
```

```
 ;;
esac
# EOF
```

Installation and Automatic bootstrapping

Manual

```
make install

# or - packaged:

mkdir -p /tmp/env_init-installdir
make install DESTDIR=/tmp/env_init-installdir
cd /tmp
fpm -s dir -t rpm -n env_init -v 0.1_el6 -C /tmp/env_init-installdir etc # or deb in
case of ubuntu
```

Automated

This should be implemented as a build routine in Bamboo/Jenkins or the like and packaged via Capistrano and FPM.

Modules and Lmod

See: Environment Modules

Bash modifications

The Texas Advanced Computing Center of the University of Austin wrote Lua Modules (Lmod) and also uses a slightly modified version of Bash which sets every shell to the equivalent of a login shell, in order to ensure that scripts in `/etc/profile.d` get executed every time before a user or script does anything on the Compute environment. TACC was kind enough to provide the GMI with a `diff(1)` of their patch to Bash:

```
--- config-top.h.orig 2010-11-12 09:43:25.422294947 -0600
+++ config-top.h 2010-11-12 09:45:32.852260136 -0600
@@ -54,14 +54,14 @@
 /* The default value of the PATH variable. */
 #ifndef DEFAULT_PATH_VALUE
 #define DEFAULT_PATH_VALUE \
-  "/usr/gnu/bin:/usr/local/bin:/bin:/usr/bin:."
+  "/usr/local/bin:/bin:/usr/bin:."
 #endif

 /* The value for PATH when invoking `command -p'.  This is only used when
    the Posix.2 confstr () function, or CS_PATH define are not present. */
 #ifndef STANDARD_UTILS_PATH
 #define STANDARD_UTILS_PATH \
-  "/bin:/usr/bin:/sbin:/usr/sbin:/etc:/usr/etc"
+  "/bin:/usr/bin:/sbin:/usr/sbin"
 #endif

 /* Default primary and secondary prompt strings. */
@@ -75,20 +75,20 @@
 #define KSH_COMPATIBLE_SELECT

 /* System-wide .bashrc file for interactive shells. */
-/* #define SYS_BASHRC "/etc/bash.bashrc" */
+#define SYS_BASHRC "/etc/tacc/bashrc"

 /* System-wide .bash_logout for login shells. */
-/* #define SYS_BASH_LOGOUT "/etc/bash.bash_logout" */
+#define SYS_BASH_LOGOUT "/etc/tacc/bash_logout"

 /* Define this to make non-interactive shells begun with argv[0][0] == '-'
    run the startup files when not in posix mode. */
-/* #define NON_INTERACTIVE_LOGIN_SHELLS */
+#define NON_INTERACTIVE_LOGIN_SHELLS

 /* Define this if you want bash to try to check whether it's being run by
    sshd and source the .bashrc if so (like the rshd behavior).  This checks
    for the presence of SSH_CLIENT or SSH2_CLIENT in the initial environment,
    which can be fooled under certain not-uncommon circumstances. */
-/* #define SSH_SOURCE_BASHRC */
+#define SSH_SOURCE_BASHRC

 /* Define if you want the case-capitalizing operators (~[~]) and the
    `capcase' variable attribute (declare -c). */
--- pathnames.h.in.orig 2010-11-12 09:43:44.562260406 -0600
+++ pathnames.h.in 2010-11-12 09:45:58.693843626 -0600
@@ -25,7 +25,7 @@
 #define DEFAULT_HOSTS_FILE "/etc/hosts"

 /* The default login shell startup file. */
-#define SYS_PROFILE "/etc/profile"
+#define SYS_PROFILE "/etc/tacc/profile"

 /* The default location of the bash debugger initialization/startup file. */
 #define DEBUGGER_START_FILE "@DEBUGGER_START_FILE@"
```