Path Manual

Matt Boyd, mattboyd@users.sourceforege.net

1 Installation

- Download the tar file. You can find the latest release on SourceForge. The tar file contain three files:
  - **`path.sh`**
    This file contains the bash shell functions, which provide the Path command line interface. This file will be sourced into your environment.
  - **`path.pl`**
    This file reads and write the stored variable values.
  - **`path.complete`**
    This file contains the bash command completion rules for path. This file is optionally sourced into your environment.
- Extract the tar file. The tar file contains the scripts. The scripts do not need to be in your search path. To use Path, the `path.sh` file must be sourced into your environment (usually from your startup script). Extract the files to a location you can source the file from.
- Source the files. You can directly source the `path.sh` and `path.complete` files or you can add them to your startup scripts. If you want path to be available when you start a shell, add the following to your bash startup script (i.e. .bashrc):

```
. {install_path}/path.sh
. {install_path}/path.complete
```

2 Examples

2.1 Creating new variables

To create a new variable, run `path edit {variable}`. An editor will open a new file. Enter the value of your variable. Then save and exit the editor. The new value will be loaded into your environment.

2.2 Editing variable values

The path delimiter. Path files are new-line delimited. That is, each sections of a path appears

on a new line. So a PATH variable with the environment setting `/bin:/usr/bin:/usr/local/bin:~/bin` would appear in the editor as:

```
/bin
/usr/bin
/usr/local/bin
~/bin
```

When Path loads the file, it will insert the environments path delimiter between the lines.

Path does some variable replacement and path expansion. When setting up the PATH variable, various 'bin' directories are often included as '$SOFTWARE_HOME/bin'. Path will expand

2.3 Loading variables

Variable are loaded by running the command `path load {varname}`. In this case, a configuration was not specified so the variable will be loaded from the active configuration. The default configuration is "default". To load a variable from a specific configuration, run `path load {varname} {configname}`.

2.4 Working with Configurations

Configurations exist when a variable exists with any given name. Creating a variable without specifying a configuration name puts it in the active configuration. The default configuration is named "default".

Once one or more variable are in a configuration, that set of variable can be loaded at once by running `path config {configname}`. This command changes the active configuration to {configname} and load all variables in that configuration.

An active configuration become the default. Running a command like `path load VAR` loads VAR from {configname}. If that variable doesn't exist in the active configuration then it is set to nothing.

3 Commands

add
Adds a variable to Path.
append
Appends a value to the end of a variable.
config
Loads all variables in the configuration into the environment.
delete

Removes a variable from Path.

edit - `path edit [var] [config]`

Edits a variable and loads the variable into the environment.

init - `path init`

If you get yourself in trouble by munging the PATH environment variable, get back to a usable setting by running this command. The `~/.path/.path.init` file will be loaded.

insert

Adds a value at the beginning of a variable.

list

Outputs the contents of a variable in a parsed format.

load

Loads the variable into the environment.

move

Renames a variable in path.

print

Outputs the contents of a variable unparsed.

remove

Removes a variable from path.

restore

Restores a removed variable.

status

I don't know what this does.

unload

Unloads a variable from the environment.