How To Make Env Vars (UNIX)

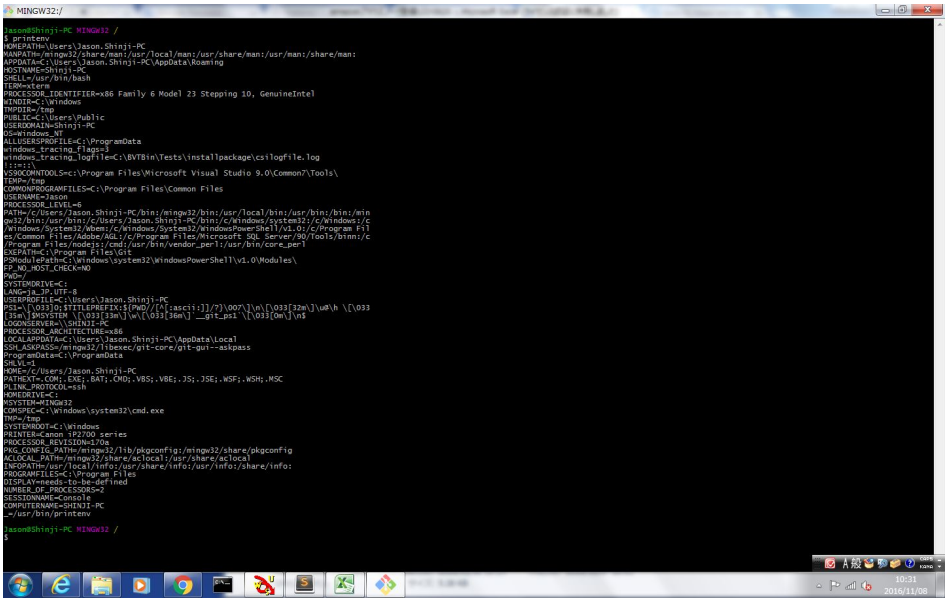
Printing Shell and Environmental Variables

Each shell session keeps track of its own shell and environmental variables. We can access these in a few different ways.

We can see a list of all of our environmental variables by using the env or printenv commands. In their default state, they should function exactly the same:

印刷シェルと環境変数
各シェルセッションには、独自のシェルおよび環境変数を追跡します。私たちは、いくつかの異なる方法でこれらにアクセスすることができます。

我々は、ENVまたはprintenvのコマンドを使用して、当社の環境変数の一覧を見ることができます。デフォルトの状態では、彼らはまったく同じ機能する必要があります：



This is fairly typical of the output of both printenv and env. The difference between the two commands is only apparent in their more specific functionality. For instance, with printenv, you can requests the values of individual variables:

これは、printenvのおよびenvの両方の出力のかなり典型的です。2つのコマンドの違いは、それらのより具体的な機能にのみ明らかです。例えば、printenvので、あなたは、個々の変数の値を要求することができます。

```
printenv SHELL

/bin/bash
```

On the other hand, env let's you modify the environment that programs run in by passing a set of variable definitions into a command like this:

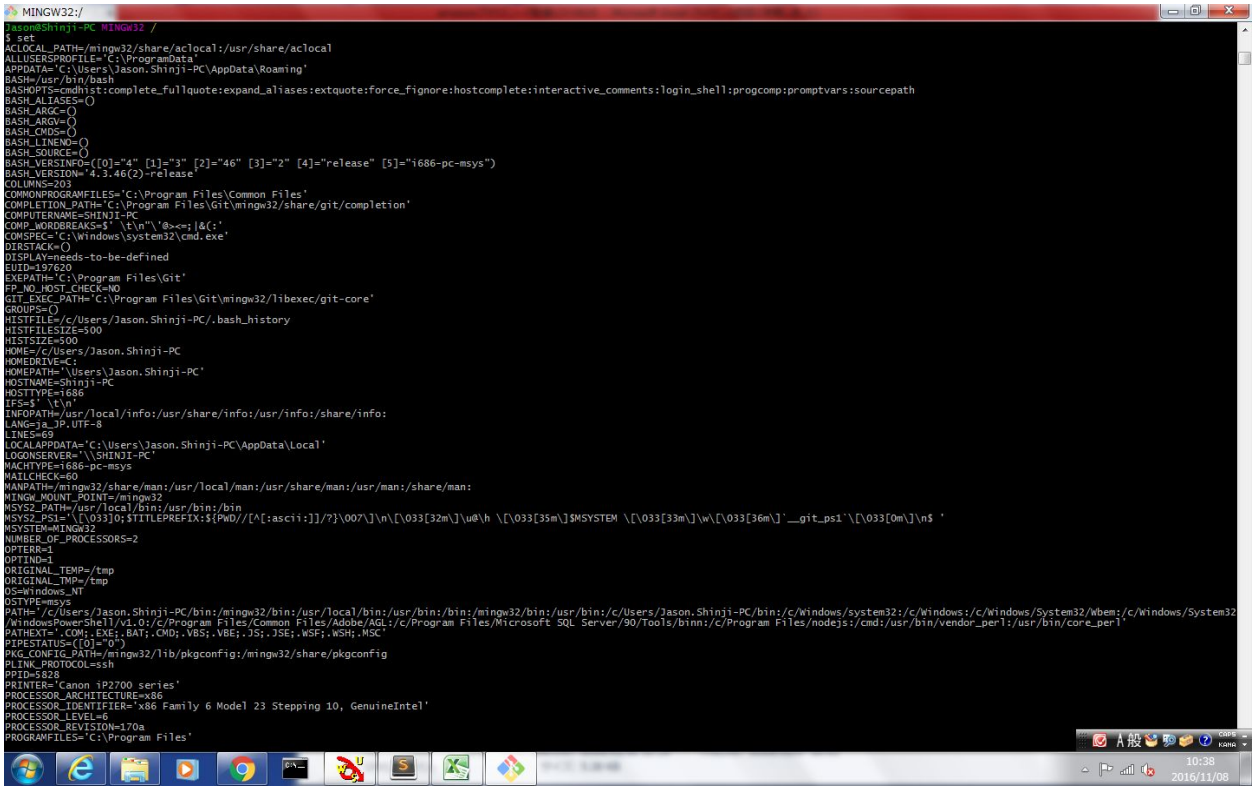
一方、のenvみましょうあなたは、このようなコマンドに変数定義のセットを渡すことによって、プログラムがで実行する環境を変更します。

```
env VAR1="blahblah" command_to_run command_options
```

The set command can be used for this. If we type set without any additional parameters, we will get a list of all shell variables, environmental variables, local variables, and shell functions:

設定コマンドは、このために使用することができます。我々は、任意の追加パラメータを指定せずに設定すると入力した場合、我々はすべてのシェル変数、環境変数、ローカル変数、およびシェル関数のリストを取得します。

set



Setting Shell and Environmental Variables

To better understand the difference between shell and environmental variables, and to introduce the syntax for setting these variables, we will do a small demonstration.

Creating Shell Variables

We will begin by defining a shell variable within our current session. This is easy to accomplish; we only need to specify a name and a value. We'll adhere to the convention of keeping all caps for the variable name, and set it to a simple string.

シェル環境変数の設定

より良いシェルと環境変数の違いを理解するために、これらの変数を設定するための構文を導入するために、我々は小さなデモを行います。

シェル変数を作成します

私たちは、現在のセッション内でシェル変数を定義することによって開始されます。これが実現するのは簡単です。我々は唯一の名前と値を指定する必要があります。私たちは、変数名のすべてのキャップを保つの規則に付着し、単純な文字列に設定します。

```
TEST_VAR='Hello World!'

echo TEST_VAR
```

Creating Environmental Variables

Now, let's turn our shell variable into an environmental variable. We can do this by *exporting* the variable. The command to do so is appropriately named:

環境変数の作成

さて、環境変数に、当社のシェル変数を回してみましょう。私たちは、変数をエクスポートすることによってこれを行うことができます。そうするコマンドが適切に名前が付けられます。

```
export TEST_VAR
```

This will change our variable into an environmental variable. We can check this by checking our environmental listing again:

これは、環境変数に私たちの変数を変更します。私たちは、再び私たちの環境のリストをチェックすることによってこれを確認することができます。

Implementing Environmental Variables

As you can see, there are a variety of different files that we would usually need to look at for placing our settings.

環境変数の実装

あなたが見ることができるように、我々は通常、設定を配置するために見てする必要がある別のファイルのさまざまながあります。

This provides a lot of flexibility that can help in specific situations where we want certain settings in a login shell, and other settings in a non-login shell. However, most of the time we will want the same settings in both situations.

これは、我々は、非ログインシェルでのログインシェルで特定の設定、およびその他の設定をしたい具体的な状況で助けることができる多くの柔軟性を提供します。しかし、ほとんどの時間は、我々は両方の状況で同じ設定をお勧めします。

Fortunately, most Linux distributions configure the login configuration files to source the non-login configuration files. This means that you can define environmental variables

that you want in both inside the non-login configuration files. They will then be read in both scenarios.

We will usually be setting user-specific environmental variables, and we usually will want our settings to be available in both login and non-login shells. This means that the place to define these variables is in the ~/.bashrc file.

幸いなことに、ほとんどのLinuxディストリビューションは、非ログイン設定ファイルを供給するためにログインされた設定ファイルを設定します。これは、あなたが非ログイン構成ファイル内部の両方で必要な環境変数を定義できることを意味します。これらは、両方のシナリオで読み込まれます。

我々は通常、ユーザー固有の環境変数を設定することになり、私たちは通常、私たちの設定は、両方のログインと非ログインシェルで利用できるようになるでしょう。これは、これらの変数を定義する場所は、〜/.bashrcファイルにあることを意味します。

Open this file now:

```
open ~/.bashrc
```

This will most likely contain quite a bit of data already. Most of the definitions here are for setting bash options, which are unrelated to environmental variables. You can set environmental variables just like you would from the command line:

これは、最も可能性の高い、すでにデータのかなりのビットが含まれています。ここでの定義のほとんどは、環境変数に関係のないbashのオプションを設定するためのものです。あなたは、環境変数はちょうどあなたがコマンドラインからと同じように設定することができます。

```
export VARNAME=value
```

We can then save and close the file. The next time you start a shell session, your environmental variable declaration will be read and passed on to the shell environment. You can force your current session to read the file now by typing:

私たちは、ファイルを保存して閉じることができます。あなたは、シェルセッションを開始する次回は、あなたの環境変数の宣言が読み込まれ、シェル環境に渡されます。あなたは次のように入力して、今ファイルを読むためにあなたの現在のセッションを強制することができます：

```
source ~/.bashrc
```

Conclusion

Environmental and shell variables are always present in your shell sessions and can be very useful. They are an interesting way for a parent process to set configuration details for its children, and are a way of setting options outside of files.

This has many advantages in specific situations. For instance, some deployment mechanisms rely on environmental variables to configure authentication information. This is useful because it does not require keeping these in files that may be seen by outside parties. There are plenty of other, more mundane, but more common scenarios where you will need to read or alter the environment of your system. These tools and

techniques should give you a good foundation for making these changes and using them correctly.

結論

環境とシェル変数は、常にあなたのシェルセッションに存在していると非常に便利です。彼らは、親プロセスがその子のために設定の詳細を設定するための興味深い方法で、ファイルの外のオプションを設定する方法です。

これは、特定の状況では多くの利点を有します。例えば、いくつかの展開メカニズムは、認証情報を設定するための環境変数に依存しています。それは部外者によって見ることができるファイルにこれらを維持する必要がないので便利です。

あなたが読んだり、システムの環境を変更する必要があります他の、より多くの世俗的な、しかし、より一般的なシナリオがたくさんあります。これらのツールやテクニックはあなたにこれらの変更を行うと、それらを正しく使用するための良い基盤を与える必要があります。