Environment variable

From Wikipedia, the free encyclopedia

Environment variables are a set of dynamic named values that can affect the way running processes will behave on a computer.

Contents

- 1 Synopsis
- 2 Getting and setting environment variables
  - 2.1 Unix
  - 2.2 DOS and Windows
  - 2.3 Unexported variables
- 3 Security
- 4 Common environment variables
  - 4.1 Examples of Unix environment variables
  - 4.2 Examples of DOS environment variables
  - 4.3 Examples from Microsoft Windows
    - 4.3.1 Discrete value variables
    - 4.3.2 System path variables
    - 4.3.3 User management variables
- 5 Default Values on Microsoft Windows
- 6 See also
- 7 External links
  - 7.1 Unix
  - 7.2 Windows

Synopsis

In all Unix and Unix-like systems, each process has its own private set of environment variables. By default, when a process is created it inherits a duplicate environment of its parent process, except for explicit changes made by the parent when it creates the child. At API level, these changes must be done between `fork` and `exec`. Alternatively, from shells such as bash, you can change environment variables for a particular command invocation by indirectly invoking it via `env` or using the `ENVIRONMENT_VARIABLE=VALUE <command>` notation. All Unix operating system flavors as well as DOS and Microsoft Windows have environment variables; however, they do not all use the same variable names. Running programs can access the values of environment variables for configuration purposes. Examples of environment variables include:

- PATH - lists directories the shell searches, for the commands the user may type without having to provide the full path.
- HOME (Unix-like) and userprofile (Microsoft Windows) - indicate where a user's home directory is located in the file system.
- TERM (Unix-like) - specifies the type of computer terminal or terminal emulator being used (e.g., **vt100** or **dumb**).
- PS1 (Unix-like) - specifies how the prompt is displayed in the Bourne shell and variants.
- MAIL (Unix-like) - used to indicate where a user's mail is to be found.

Shell scripts and batch files use environment variables to communicate data and preferences to child processes. They can also be used to store temporary values for reference later in the script, although in Unix other variables are usually used for this.

In Unix, an environment variable that is changed in a script or compiled program will only affect that process and possibly child processes. The parent process and any unrelated processes will not be affected. In DOS changing a variable's value (or removing it) inside a BATCH file will change the variable for the duration of command.com's existence.

In Unix, the environment variables are normally initialized during system startup by the system init scripts, and hence inherited by all other processes in the system. Users can, and often do, augment them in the profile script for the shell they are using. In Microsoft Windows, environment variables defaults are stored in the windows registry or set in autoexec.bat.

Getting and setting environment variables

The variables can be used both in scripts and on the command line. They are usually referenced by putting special symbols in front of or around the variable name. For instance, to display the program search path, in most scripting environments, the user has to type:

```
echo $PATH
```

On DOS or Windows system, the user has to type this:

```
echo %PATH%
```

Unix

The **env**, **set**, and **printenv** commands display all environment variables and their values. `env` and `set` are also used to set environment variables and are often incorporated directly into the shell. `printenv` can also be used to print a single variable by giving that variable name as the sole argument to the command.

In Unix, the following commands can also be used, but are often dependent on a certain shell.

```
export VARIABLE=value  # for Bourne, bash, and related shells
setenv VARIABLE value  # for csh and related shells
```

DOS and Windows

In DOS and Windows, the **set** command without any arguments displays all environment variables along with their values.

To set a variable to a particular value, use:

```
set VARIABLE=value
```

However, this is temporary. Permanent change to the environment variable can be achieved through editing the registry (not recommended for novices) and using the Windows Resource Kit application `setx.exe`. With the introduction of Windows Vista, the `setx` command became part of Windows.

The most common method of setting an environment variable in Windows is via <Control Panel:System:Advanced:Environment Variables>, through the registry this is done changing the values under HKCU\Environment (for user specific variables) and HKLM\SYSTEM\CurrentControlSet\Control \Session Manager\Environment (for System variables).

To see the current value of a particular variable, use:

```
set VARIABLE
```

Note: Please take note that doing so will print out all variables beginning with 'VARIABLE'. Another example is:

```
C:\> set p
Path=c:\.. ..
PATHEXT=.COM;.EXE;.BAT;
PROCESSOR_ARCHITECTURE=.. ..
PROCESSOR_IDENTIFIER=x8..
PROCESSOR_LEVEL=6..
PROCESSOR_REVISION=1706..
ProgramFiles=C:\Program.. .
PROMPT=$P$G
```

or

```
echo %VARIABLE%
```

Unexported variables

In Unix shells, variables may be assigned without the **export** keyword. Variables defined in this way are displayed by the **set** command, but are **not** true environment variables, as they are stored only by the shell and not recognized by the kernel. They will not be detected by the **printenv** command and are not inherited by child processes.

```
VARIABLE=value
```

However, if used in front of a program to run, the variables will be exported to the environment and thus appear as real environment variables to the program:

```
VARIABLE=value program_name [arguments]
```

The tool that gives closest parallel in Windows is the SETLOCAL/ENDLOCAL commands that prevent variables from being set globally.

Security

On Unix, a setuid program is given an environment chosen by its caller, but it runs with different authority from its caller. The dynamic linker will usually load code from locations specified by the environment variables LD_LIBRARY_PATH and LD_PRELOAD and run it with the process's authority. If a setuid program did this, it would be insecure, because its caller could get it to run arbitrary code and hence misuse its authority. For this reason, libc unsets these environment variables at startup in a setuid process. setuid programs usually unset unknown environment variables and check others or set them to reasonable values.

Common environment variables

Examples of Unix environment variables

$PATH

Contains a colon-separated list of directories that the shell searches for commands that do not contain a slash in their name (commands with slashes are interpreted as file names to execute, and the shell attempts to execute the files directly). See Path (computing)

$HOME

Contains the location of the user's home directory. Although the current user's home directory can also be found out through the C functions `getpwuid` and `getuid`, `$HOME` is often used for convenience in various shell scripts (and other contexts). Using the environment variable also gives the user the possibility to point to an other directory.

$PWD

This variable points to the current directory. Equivalent to the output of the command pwd when called without arguments.

$DISPLAY

Contains the identifier for the display that X11 programs should use by default.

$LD_LIBRARY_PATH

On many Unix systems with a dynamic linker, contains a colon-separated list of directories that the dynamic linker should search for shared objects when building a process image after `exec`, before searching in any other directories.

$LANG, $LC_ALL, $LC_...

LANG is used to set to the default locale. For example, if the locale values are `pt_BR`, then the language is set to (Brazilian) Portuguese and Brazilian practice is used where relevant. Different aspects of localization are controlled by individual LC_-variables (LC_CTYPE, LC_COLLATE, LC_DATE etc.). LC_ALL can be used to force the same locale for all aspects.

$TZ

Refers to Time zone. It can be in several formats, either specifying the timezone itself or referencing a file (in /usr/share/zoneinfo).

Examples of DOS environment variables

%COMSPEC%

This variable contains the full path to the command processor, command.com.

%ERRORLEVEL%

This variable points to the current error level. If there was an error in the previous command, this is what you need to check against to find out about that.

%PATH%

This variable contains a semicolon-delimited list of directories in which the command interpreter will search for executable files. Equivalent to the Unix $PATH variable (although note that PATH on Windows additionally performs the same task as LD_LIBRARY_PATH on Unix-like systems). Note that %PATH% can also be set like this PATH=c:\dos; where SET isn't required.

%TEMP% and %TMP%

These variables contain the path to the directory where temporary files should be stored.Computer

Examples from Microsoft Windows

Discrete value variables

These variables generally expand to discrete values, such as the current working directory, the current date, or a random number. Some of these are true environment variable and will be expanded by all functions that handle environment variables. Others, like %CD% simply look like environment variables and will only be expanded by some functions and shells. They are not case sensitive.

%CD%

This variable points to the current directory. Equivalent to the output of the command `cd` when called without arguments.

%DATE%

This variable expands to the current date. The date is displayed according to the current user's date format preferences.

The following is a way of reformatting the date and time for use in file copies. The example assumes UK format of day month year and the time is set for a 24 hour clock.

```
@echo off
echo %DATE% %TIME%
set MTH=%DATE:~4,2%
set DAY=%DATE:~7,2%
set YR=%DATE:~10,4%
set HR=%TIME:~0,2%
set HR0=%TIME:~0,1%
if "%HR0%"==" " set HR=0%TIME:~1,1%
set MIN=%TIME:~3,2%
set SEC=%TIME:~6,2%
set MYDATE=%YR%%MTH%%DAY%-%HR%%MIN%%SEC%
echo %MYDATE%
```

%ERRORLEVEL%

This variable points to the current error level. If there was an error in the previous command, this is what you need to check against to find out about that.

%RANDOM%

This variable returns a random number between 0 and 32767

%TIME%

This variable points to the current time. The time is displayed according to the current user's time format preferences.

System path variables

These variables refer to locations of critical operating system resources, and as such generally are not user-dependent.

%AppData%

Contains the full path to the Application Data folder of the logged-in user. Does not work on Windows NT 4.0 SP6 UK.

%ComSpec%

This variable contains the full path to the command processor; on Windows NT based operating systems this is `cmd.exe`, while on Windows 9x and ME it is the DOS command processor, `COMMAND.COM`.

%LOCALAPPDATA%

This variable is the temporary files of Applications. Its uses include storing of Desktop Themes, Windows Error Reporting, Caching and profiles of web browsers.

%PATH%

This variable contains a semicolon-delimited (do not put spaces in between) list of directories in which the command interpreter will search for an executable file that matches the given command. Equivalent to the Unix `$PATH` variable.

%ProgramFiles%

This variable points to Program Files directory, which stores all the installed program of Windows and others. The default on English-language systems is `C:\Program Files`. In 64-bit editions of Windows (XP, 2003, Vista), there are also **`%ProgramFiles(x86)%`** which defaults to `C:\Program Files (x86)` and **`%ProgramW6432%`** which defaults to `C:\Program Files`. The %ProgramFiles% itself depends on whether the process requesting the environment variable is itself 32-bit or 64-bit (this is caused by Windows-on-Windows 64-bit redirection).

%CommonProgramFiles%

This variable points to Common Files directory. The default is `C:\Program Files\Common Files`.

%SystemDrive%

The %SystemDrive% variable is a special system-wide environment variable found on Microsoft Windows NT and its derivatives. Its value is the drive upon which the system folder was placed. Also see next item.

The value of %SystemDrive% is in most cases `C:`.

%SystemRoot%

The %SystemRoot% variable is a special system-wide environment variable found on Microsoft Windows NT and its derivatives. Its value is the location of the system folder, including the drive and path.

The drive is the same as %SystemDrive% and the default path on a clean installation depends upon the version of the operating system. By default, on a clean installation:

- Windows NT 5.1 (Windows XP) and newer versions use `\WINDOWS`

- Windows NT 5.0 (Windows 2000), Windows NT 4.0 and Windows NT 3.1 use `\WINNT`
- Windows NT 3.5x uses `\WINNT35`

%WinDir%

This variable points to the Windows directory (on Windows NT-based operating systems it is identical to the %SystemRoot% variable, above). If the System is on drive C: then the default values are:

- `C:\WINDOWS` on Windows 95, Windows 98, Windows Me, Windows XP, Windows Server 2003, Windows Vista and Windows Server 2008
- `C:\WINNT` for Windows NT 4, and Windows 2000

Note that Windows NT 4 Terminal Server Edition by default installs to `C:\WTSRV`.

User management variables

These variables store information related to resources and settings owned by various user profiles within the system. As a general rule, these variables do not refer to critical system resources or locations that are necessary for the OS to run.

%AllUsersProfile% (%PROGRAMDATA% for vista,7)

The %AllUsersProfile%(%PROGRAMDATA%) variable expands to the full path to the All Users profile directory. This profile contains resources and settings that are used by all system accounts. Shortcut links copied to the All Users' Start menu or Desktop folders will appear in every user's Start menu or Desktop, respectively.

%UserDomain%

The variable holds the name of the Workgroup or Windows Domain to which the current user belongs. The related variable, %LOGONSERVER%, holds the hostname of the server that authenticated the current user's logon credentials (name and password). For Home PCs, and PCs in a Workgroup, the authenticating server is usually the PC itself. For PCs in a Windows Domain, the authenticating server is a domain controller (a primary domain controller, or PDC, in Windows NT 4-based domains).

%UserProfile%

The %UserProfile% variable is a special system-wide environment variable found on Microsoft Windows NT and its derivatives. Its value is the location of the current user's profile directory, in which is found that user's HKCU registry hive (`NTUSER`).

Users can also use the **%USERNAME%** variable to determine the active users login identification.

Default Values on Microsoft Windows

Variable	Windows XP	Windows Vista/7
%ALLUSERSPROFILE% (%PROGRAMDATA%)	C:\Documents and Settings\All Users	C:\ProgramData
%APPDATA%	C:\Documents and Settings\{username}\Application Data	C:\Users\{username}\AppData\Roaming
%COMPUTERNAME%	{computername}	{computername}
%COMMONPROGRAMFILES%	C:\Program Files\Common Files	C:\Program Files\Common Files
%COMMONPROGRAMFILES(x86)%	C:\Program Files (x86)\Common Files	C:\Program Files (x86)\Common Files
%COMSPEC%	C:\Windows\System32\cmd.exe	C:\Windows\System32\cmd.exe
%HOMEDRIVE%	C:	C:
%HOMEPATH%	\Documents and Settings\{username}	\Users\{username}
%LOCALAPPDATA%	C:\Documents and Settings\{username}\Application Data\Local	C:\Users\{username}\AppData\Local
%LOGONSERVER%	\\{userdomain}	\\{userdomain}
%PATH%	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;{plus program paths}	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;{plus program paths}
%PATHEXT%	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.WSF;.WSH	.com;.exe;.bat;.cmd;.vbs;.vbe;.js;.jse;.wsf;.wsh;.msc
%PROGRAMFILES%	C:\Program Files	C:\Program Files
%PROGRAMFILES(X86)%	C:\Program Files (x86) (only in 64-bit version)	C:\Program Files (x86) (only in 64-bit version)
%PROMPT%	Code for current command prompt format. Code is usually PG	Code for current command prompt format. Code is usually PG
%SYSTEMDRIVE%	C:	C:
%SystemRoot%	The Windows directory, usually C:\Windows, formerly C:\WINNT	C:\Windows
%TEMP% and %TMP%	C:\Documents and Settings\{username}\Local Settings\Temp	C:\Users\{username}\AppData\Local\Temp
%USERDOMAIN%	{userdomain}	{userdomain}

%USERNAME%	{username}	{username}
%USERPROFILE%	C:\Documents and Settings\{username}	C:\Users\{username}
%WINDIR%	C:\Windows	C:\Windows
%PUBLIC%		C:\Users\Public
%PSModulePath%		%SystemRoot%\system32\WindowsPowerShell \v1.0\Modules\

See also

- List of Unix programs
- List of DOS commands
- Environment Modules
- PWB shell for early (1975-) history of environment variables

External links

Unix

- `environ(7) (http://linux.die.net/man/7/environ)` : user environment – Linux Conventions and Miscellany Manual

Windows

- Environment Variable Reference (http://www.scriptlogic.com/support/CustomScripts/environmentVariableReference.html) — *Has a list showing which environment variables are for 9x WinNTx etc*
- Windows XP Command Shell Overview (http://technet.microsoft.com/en-gb/library/bb490954.aspx) with a list of environment variables — *Microsoft.com*
- How To Manage Environment Variables in Windows XP (http://support.microsoft.com/default.aspx?scid=kb;en-us;310519) — *Microsoft.com*
- Path Manager (pathman.exe) (http://www.microsoft.com/downloads/details.aspx?FamilyID=33543464-e0fb-4b36-80d0-4e0aba57d47f&DisplayLang=en) — *Command line tool from Microsoft for editing PATH environment variable on Windows*
- Environment Variables in Windows XP (http://vlaurie.com/computers2/Articles/environment.htm) — *Computer Education*
- RapidEE (Rapid Environment Editor) (http://www.rapidee.com) — *Windows environment variables editor*
- (EnvMan) Windows Environment Variables Manager (http://env-man.blogspot.com/2007/04/envman-user-guide.html) — *Environment Variables Editor for Windows*
- Managing Search PATH on Windows (http://redmondlab.net/path) — *Easy interactive way of editing PATH environment variable on Windows*
- Getting Environment Variables in .NET (http://msdn.microsoft.com/en-us/library/system.environment.getenvironmentvariable.aspx) — *How to get environment variables in .NET*