[EnvironmentVariables](https://help.ubuntu.com/community/EnvironmentVariables?action=fullsearch&value=linkto%3A%22EnvironmentVariables%22&context=180" \o "Click to do a full-text search for this title)

|  |
| --- |
| Contents   1. [Manipulating environment variables and values](https://help.ubuntu.com/community/EnvironmentVariables#Manipulating_environment_variables_and_values)    1. [Setting values to environment variables](https://help.ubuntu.com/community/EnvironmentVariables#Setting_values_to_environment_variables)    2. [Examining values of environment variables](https://help.ubuntu.com/community/EnvironmentVariables#Examining_values_of_environment_variables)       1. [Desktop environment specifics](https://help.ubuntu.com/community/EnvironmentVariables#Desktop_environment_specifics)    3. [Erasing environment variables](https://help.ubuntu.com/community/EnvironmentVariables#Erasing_environment_variables) 2. [Working principles of environment variables](https://help.ubuntu.com/community/EnvironmentVariables#Working_principles_of_environment_variables)    1. [Process locality](https://help.ubuntu.com/community/EnvironmentVariables#Process_locality)    2. [Inheritance](https://help.ubuntu.com/community/EnvironmentVariables#Inheritance)    3. [Case sensitivity](https://help.ubuntu.com/community/EnvironmentVariables#Case_sensitivity) 3. [Bash's quick assignment and inheritance trick](https://help.ubuntu.com/community/EnvironmentVariables#Bash.27s_quick_assignment_and_inheritance_trick) 4. [Persistent environment variables](https://help.ubuntu.com/community/EnvironmentVariables#Persistent_environment_variables)    1. [Session-wide environment variables](https://help.ubuntu.com/community/EnvironmentVariables#Session-wide_environment_variables)       1. [~/.pam\_environment](https://help.ubuntu.com/community/EnvironmentVariables#A.2BAH4-.2F.pam_environment)       2. [~/.profile](https://help.ubuntu.com/community/EnvironmentVariables#A.2BAH4-.2F.profile)       3. [Other files](https://help.ubuntu.com/community/EnvironmentVariables#Other_files)    2. [System-wide environment variables](https://help.ubuntu.com/community/EnvironmentVariables#System-wide_environment_variables)       1. [/etc/environment](https://help.ubuntu.com/community/EnvironmentVariables#A.2Fetc.2Fenvironment)       2. [/etc/profile.d/\*.sh](https://help.ubuntu.com/community/EnvironmentVariables#A.2Fetc.2Fprofile.d.2F.2A.sh)       3. [Other files](https://help.ubuntu.com/community/EnvironmentVariables#Other_files-1)    3. [sudo caveat](https://help.ubuntu.com/community/EnvironmentVariables#sudo_caveat)    4. [Launching desktop application with an environment variable](https://help.ubuntu.com/community/EnvironmentVariables#Launching_desktop_application_with_an_environment_variable) 5. [List of common environment variables](https://help.ubuntu.com/community/EnvironmentVariables#List_of_common_environment_variables)    1. [File-location related variables](https://help.ubuntu.com/community/EnvironmentVariables#File-location_related_variables)    2. [Locale setting variables](https://help.ubuntu.com/community/EnvironmentVariables#Locale_setting_variables)       1. [The LANGUAGE priority list](https://help.ubuntu.com/community/EnvironmentVariables#The_LANGUAGE_priority_list)    3. [Preferred application variables](https://help.ubuntu.com/community/EnvironmentVariables#Preferred_application_variables)    4. [Graphical desktop-related variables](https://help.ubuntu.com/community/EnvironmentVariables#Graphical_desktop-related_variables)       1. [Gnome-specific variables](https://help.ubuntu.com/community/EnvironmentVariables#Gnome-specific_variables)    5. [Program execution variables](https://help.ubuntu.com/community/EnvironmentVariables#Program_execution_variables)    6. [Compilation and software development related variables](https://help.ubuntu.com/community/EnvironmentVariables#Compilation_and_software_development_related_variables)    7. [Other environment variables](https://help.ubuntu.com/community/EnvironmentVariables#Other_environment_variables) |

Environment variables provide a way to influence the behaviour of software on the system. For example, the "LANG" environment variable determines the language in which software programs communicate with the user.

Environment variables consist of names that have values assigned to them. For example, on a typical system in the US we would have the value "en\_US.UTF-8" assigned to the "LANG" variable.

The meaning of an environment variable and the format of its value are determined by the application using it. There are quite a few well-known environment variables for which the meaning and the format have been agreed upon and they are used by many applications.

Manipulating environment variables and values

While quite a few graphical system configuration applications actually manipulate environment variables in the background, the command-line allows for maximum flexibility when manipulating environment variables.

**Note:** The shell techniques explained in the following sections apply to the *Bourne Shell* family of command line shells, which includes *sh*, *ksh*, and *bash*, which is the default shell shipped with Ubuntu. The commands may be different on other shells such as *csh*.

Setting values to environment variables

In order to set a value to an existing environment variable, we use an *assignment expression*. For instance, to set the value of the "LANG" variable to "he\_IL.UTF-8", we use the following command:

LANG=he\_IL.UTF-8

If we use an assignment expression for a variable that doesn't exist, the shell will create a *shell variable*, which is similar to an environment variable but does not influence the behaviour of other applications.

A shell variable can be *exported* to become an environment variable with the exportcommand. To create the "EDITOR" environment variable and assign the value "nano" to it, you can do:

EDITOR=nano

export EDITOR

The *bash* shell (the default command-line shell in Ubuntu) provides a shortcut for creating environment variables. The previous example could be performed with the following single command:

export EDITOR=nano

Examining values of environment variables

The *printenv* command prints the names and values of all currently defined environment variables:

printenv

To examine the value of a particular variable, we can specify its name to the *printenv* command:

printenv TERM

Another way to achieve that is to use the dollar sign ($), as used in the following example:

echo $TERM

There is a command for doing temporary, short-term changes to the environment. It can also be used to display the current environment. This command is *env*.

env

The dollar sign can actually be used to combine the values of environment variables in many shell commands. For example, the following command can be used to list the contents of the "Desktop" directory within the current user's home directory.

ls $HOME/Desktop

For the sake of completeness: If you want to print the names and values also of the non-exported shell variables, i.e. not only the environment variables, this is one way:

( set -o posix ; set ) | less

Desktop environment specifics

If you use a terminal window to examine environment variables, you actually study the environment variables of the shell that is running in the terminal. Those variables are not necessarily available in the graphical environment of the desktop. To examine the environment variables that are effective when you for instance start an application from [the launcher](https://help.ubuntu.com/stable/ubuntu-help/unity-launcher-intro.html), you can follow the guidance in [this Ask Ubuntu answer](http://askubuntu.com/questions/275965/how-to-list-all-variables-names-and-their-current-values/356973#356973).

Erasing environment variables

While simply setting an empty value to an environment variable, as shown in the example below, may nullify its effect in most cases, there are a few variables such as "POSIXLY\_CORRECT" whose mere existence, even with an empty value, influences the behavior of programs.

export LC\_ALL=

The *unset* command can be used in order to completely erase the existence of an environment variable:

unset LC\_ALL

It is also possible to use the "-n" switch to the *export* command in order to un-export an environment variable and therefore demote it to become a shell variable while preserving its value.

export -n LC\_ALL

Working principles of environment variables

A few simple principles govern how environment variables work and achieve their effect.

Process locality

The values of environment variables are *local*, which means they are specific to the running process in or for which they were set. This means that if we open two terminal windows (which means we have two separate *bash* processes running), and change a value of an environment variable in one of the windows, that change will not be seen by the shell in the other window or any other program currently on the desktop.

Inheritance

When a parent process creates a child process, for example when we run the "gedit" command from the terminal and "bash" (the parent process) creates "gedit" (the child process), the child process *inherits* all the environment variables and values the parent process had.

This means that if we set a new value to the "LANG" environment variable in the terminal, and then run "gedit" from that same terminal, "gedit" will inherit the new value of "LANG", and therefore may display in a different language than the rest of the processes on the desktop. (See [The LANGUAGE priority list](https://help.ubuntu.com/community/EnvironmentVariables#The_LANGUAGE_priority_list), though.)

Note that because of the *Process Locality* principle explained above, once we run Gedit, changes to environment variables of the parent process will not be seen by the child process and vice-versa.

**Note:** in the Gnome graphical desktop environment, *gnome-session* is the parent process of all the processes running on the desktop. This fact (along with the *Inheritance* principle) is the key to our ability to powerfully influence the operation of our desktop with environment variables. The equivalent process in KDE is *kde-session*.

Case sensitivity

The names of environment variables are *case sensitive*. This means that lang is not the same variable as LANG, Lang, or laNg.

It is a common practice to name all environment variables with only English capital letters and underscore (\_) signs.

Bash's quick assignment and inheritance trick

The *bash* shell has a trick to allow us to set one or more environment variables and run a child process with single command. For example, in order to set the "LANGUAGE" and "FOO" environment variables and then run "gedit", we would use the following command:

LANGUAGE=he FOO=bar gedit

Note: When using this command, the new values are only assigned to the environment variables of the child process (in this case gedit). The variables of the shell retain their original values. For instance, in the example above, the value of "LANGUAGE" will not change from its original value, as far as subsequent commands to the shell are concerned.

A similar behaviour can be achieved with other shells by using the *env* command.

Persistent environment variables

So far we've only discussed ways set an environment variable value temporarily until the shell session in which it was set is closed. One may wonder if there is a way to somehow permanently set an environment variable to a certain value.

Session-wide environment variables

Suitable files for environment variable settings that should affect just a particular user (rather than the system as a whole) are **~/.pam\_environment** and **~/.profile**. After having edited one of those files, you should **re-login** in order to initialize the variables.

~/.pam\_environment

This file is specifically meant for setting a user's environment. It is not a script file, but rather consists of assignment expressions, one per line. This example sets the variable FOO to a literal string and modifies the PATH variable:

FOO=bar

PATH DEFAULT=${PATH}:/home/@{PAM\_USER}/MyPrograms

**Note:**

1. When doing a simple variable assignment like the FOO=bar example, quotes have not special meaning. This means that values cannot contain spaces.
2. The syntax used for modifying PATH, which differs from the syntax of shell script files, is required for variable expansion to work. Some variables, like HOME, might not be set at the time **~/.pam\_environment** is parsed. See **/etc/security/pam\_env.conf** for more details.
3. **~/.pam\_environment** is written to when you use various GUIs to set the language or regional formats. Consequently, if you for instance set LC\_TIME by editing **~/.pam\_environment** manually, your entry will be overwritten if you afterwards use the Language Support GUI to change the regional formats setting.

~/.profile

In this file you can also place environment variable assignments, since it gets executed automatically by the DisplayManager during the start-up process desktop session as well as by the login shell when one logs in from the textual console. This is a **~/.profile** equivalent of the above example:

export FOO=bar

export PATH="$PATH:$HOME/MyPrograms"

**Note:** The code in **~/.profile** is run after **~/.pam\_environment** has been read. That makes **~/.profile** suitable to use if you want to override a locale related variable that was set in **~/.pam\_environment** via e.g. Language Support.

Other files

If you are using KDE, see also [the KDE User-base page on this topic](http://userbase.kde.org/Session_Environment_Variables/en).

Shell config files such as **~/.bashrc**, **~/.bash\_profile**, and **~/.bash\_login** are often suggested for setting environment variables. While this may work on Bash shells for programs started from the shell, variables set in those files are not available by default to programs started from the graphical environment in a desktop session.

System-wide environment variables

A suitable file for environment variable settings that affect the system as a whole (rather than just a particular user) is **/etc/environment**. An alternative is to create a file for the purpose in the **/etc/profile.d** directory.

/etc/environment

This file is specifically meant for system-wide environment variable settings. It is not a script file, but rather consists of assignment expressions, one per line.

FOO=bar

**Note:** Variable expansion does not work in **/etc/environment**.

/etc/profile.d/\*.sh

Files with the .sh extension in the **/etc/profile.d** directory get executed whenever a bash login shell is entered (e.g. when logging in from the console or over ssh), as well as by the DisplayManager when the desktop session loads.

You can for instance create the file **/etc/profile.d/myenvvars.sh** and set variables like this:

export JAVA\_HOME=/usr/lib/jvm/jdk1.7.0

export PATH=$PATH:$JAVA\_HOME/bin

Other files

While **/etc/profile** is often suggested for setting environment variables system-wide, it is a configuration file of the base-files package, so it's not appropriate to edit that file directly. Use a file in **/etc/profile.d** instead as shown above. (Files in **/etc/profile.d** are sourced by **/etc/profile**.)

**/etc/default/locale** is specifically meant for system-wide locale environment variable settings. It's written to by the installer and when you use Language Support to set the language or regional formats system-wide. On a desktop system there is normally no reason to edit this file manually.

The shell config file **/etc/bash.bashrc** is sometimes suggested for setting environment variables system-wide. While this may work on Bash shells for programs started from the shell, variables set in that file are not available by default to programs started from the graphical environment in a desktop session.

sudo caveat

Any variables added to these locations will not be reflected when invoking them with a sudo command, as sudo has a default policy of resetting the Environment and setting a secure path (this behavior is defined in **/etc/sudoers**). As a workaround, you can use **sudo su** that will provide a shell with root privileges but retaining any modified PATH variables. Alternatively you can setup sudo not to reset certain environment variables by adding some explicit environment settings to keep in **/etc/sudoers**:

1. Run **sudo visudo**
2. Add the following to the bottom:

Defaults env\_keep += "http\_proxy SOMEOTHERVARIABLES ANOTHERVARIABLE ETC"

Launching desktop application with an environment variable

You can add an environment variable to an application by editing its .desktop file. For example, to run "digiKam" with the environment variable APPMENU\_DISPLAY\_BOTH=1, find the corresponding digikam.desktop file and add the setting of the variable, via the env command, to the entry "Exec":

Exec=env APPMENU\_DISPLAY\_BOTH=1 digikam -caption "%c" %i

List of common environment variables

Each application is free to define and use its own environment variables. Many manual pages include long lists of environment variables that can affect the behaviour of the application they describe. However, the most useful variables are common to many applications.

File-location related variables

The following variables determine how the system locates various files in order to operate.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Value Examples** | **What it's for** |
| PATH | /usr/sbin:/usr/bin:/sbin:/bin | When you type a command to run, the system looks for it in the directories specified by PATH in the order specified |
| MANPATH | /usr/share/man:/usr/local/man | List of directories for the system to search manual pages in |
| LD\_LIBRARY\_PATH | /opt/app/oracle/lib | List of directories where the system searches for runtime libraries in addition to those hard-defined by ld and in /etc/ld.so.conf.d/\*.conf files. **Note:** You can only set this environment variable inside an interactive shell. Since [Ubuntu 9.04 Jaunty Jackalope](https://help.ubuntu.com/9.04/index.html), LD\_LIBRARY\_PATH cannot be set in $HOME/.profile, /etc/profile, nor /etc/environment files. You must use /etc/ld.so.conf.d/\*.conf configuration files. See [Launchpad bug #366728](https://edge.launchpad.net/ubuntu/+bug/366728) for more information. |
| TMPDIR | /var/tmp | The directory used for temporary file creation by several programs |

Locale setting variables

The following environment variables determine the locale-related behaviour of the systems such as the language of displayed messages and the way times and dates are presented.

The values that can be assigned to the locale environment variables are names of locales generated on the system. To see which locales have been generated, one can use the locale -a command.

When you install a language in Ubuntu using Language Support, locales for that language are generated automatically. Otherwise locales can be generated with the locale-gen command.

|  |  |
| --- | --- |
| **Variable** | **What it's for** |
| LANG | The basic language setting used by applications on the system, unless overridden by one of the other locale environment variables |
| LC\_CTYPE | The character set used to display and input text |
| LC\_NUMERIC | How non-monetary numeric values are formatted on screen |
| LC\_TIME | How date and time values are formatted |
| LC\_COLLATE | How to sort various information items (e.g. defines the order of the alphabet so items can be ordered alphabetically by the *sort* command) |
| LC\_MONETARY | How monetary numeric values are formatted |
| LC\_MESSAGES | Which language is to display messages to the end user |
| LC\_PAPER | Definitions of paper formats and standards |
| LC\_NAME | How names are formatted |
| LC\_ADDRESS | How to display address information |
| LC\_TELEPHONE | How telephone numbers are structured |
| LC\_MEASUREMENT | What units of measurement are used |
| LC\_IDENTIFICATION | Metadata about the locale information |
| LC\_ALL | This variable serves as a powerful override over all the other locale environment variables. When its value is set, applications use that value to determine which locale settings to use regardless of the values of the other variables. **Note:** Do not set this variable persistently on a desktop system. If you do, you effectively disable the GUIs for controlling language and locales. |

By utilizing various combinations of settings for the locale variables, you can make interesting tweaks to the behaviour of your system. For example, you can make your system display message in US-English while using number, date, and measurement formats that are more common to European countries. The [Language Support GUI](https://help.ubuntu.com/stable/ubuntu-help/session-language.html) makes that distinction by design.

The locale variables can effectively override each other's value in some combinations. Therefore examining the values of the variables themselves may not always provide clear indication of how the system will behave. The locale command can be used to examine what the effective values are to the applications.

The LANGUAGE priority list

The "LANGUAGE" environment variable, which is set by the GUIs more often than not on Ubuntu desktops, controls language for message and menu display for GNU compatible applications. For such applications it overrides whatever locale names are set in "LANG" and "LC\_MESSAGES".

Unlike "LANG" and "LC\_\*", "LANGUAGE" should not be assigned a complete locale name including the encoding part (e.g. ".UTF-8"). Instead "LANGUAGE" should contain a colon separated priority list of language codes, for instance "es:de:en". A single language code is also correct.

Preferred application variables

These environment variables indicate to various programs what the user's preferred applications are for performing certain tasks.

These variables are typically not respected by GUI applications that tend to include their own built-in text display windows and editors. Most desktop environments also contain their own preferred application selection system.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Value Examples** | **What it's for** |
| PAGER | /usr/bin/less | The name of the utility used to display long text by commands such as *man*. |
| EDITOR | /usr/bin/nano | The name of the user's preferred text editor. Used by programs such as the *mutt* mail client and *sudoedit*. |
| VISUAL | /usr/bin/gedit | Similar to the "EDITOR" environment variable, applications typically try the value in this variable first before falling back to "EDITOR" if it isn't set. |
| BROWSER | /usr/bin/lynx | The name of the user's preferred web browser. This variable is arguably less common than the rest |

Graphical desktop-related variables

|  |  |  |
| --- | --- | --- |
| **Variable** | **Value Examples** | **What it's for** |
| DISPLAY | :0.0 localhost:10.0 terminal01:0.0 | This variable is used to indicate to graphical applications where to display the actual graphical user interface, the value consists of 3 parts: A host-name followed by a colon (:), a display number followed by a dot (.) and a screen number. The host-name part can be used to have the graphical output sent to a remote machine over the network. It can be omitted when the output is meant for an X server running on the local machine. The display number allows selecting among multiple X servers running on the same machine (Ubuntu uses multiple X servers to enable multiple graphical desktop sessions). Although the screen number is used to select among multiple physical screen that are managed by the same X server, it is rarely set to anything other than "0" nowadays. Manually setting the "DISPLAY" environment variable's value is rarely needed nowadays since it can be automatically and intelligently adjusted by many applications such as "GDM" and "SSH" when needed. |
| XDG\_DATA\_HOME | ~/.local/share | Indicates to applications that conform to the freedesktop.org specifications, where to place the user's private data. This variable is typically unset since a sensible default fall-back value was defined by the specifications. |
| XDG\_CONFIG\_HOME | ~/.config | Indicates to applications that conform to the freedesktop.org specifications, where to place the user's configuration information. This variable is typically unset since a sensible default fall-back value was defined by the specifications. |
| XDG\_DATA\_DIRS | /usr/local/share:/usr/share | A colon-separated list (similar to "PATH") of directories where data is searched for by applications that conform to the freedesktop.org specifications. This variable is typically unset since a sensible default fall-back value was defined by the specifications. |
| XDG\_CONFIG\_DIRS | /etc/xdg | A colon-separated list (similar to "PATH") of directories where configuration information is searched for by applications that conform to the freedesktop.org specifications. This variable is typically unset since a sensible default fall-back value was defined by the specifications. |
| XDG\_CACHE\_HOME | ~/.cache | A location used by applications that conform to the freedesktop.org specifications to cache temporary data. This variable is typically unset since a sensible default fall-back value was defined by the specifications. |

Gnome-specific variables

|  |  |  |
| --- | --- | --- |
| **Variable** | **Value Examples** | **What it's for** |
| NAUTILUS\_SCRIPT\_SELECTED\_FILE\_PATHS | /home/ifireball/about.html | This environment variable is set by Nautilus, the Gnome file manager, to a newline-delimited list of the currently selected files, when a script is invoked from the right-click menu. This variable is only set if the files are local, e.g. not from a network share or an SSH connection |
| NAUTILUS\_SCRIPT\_SELECTED\_URIS | file:///home/ifireball/about.html | This environment variable is set by Nautilus to a newline-delimited list of the URI addresses of the currently selected files, when a script is invoked from the right-click menu. |
| NAUTILUS\_SCRIPT\_CURRENT\_URI | file:///home/ifireball | This environment variable is set to the URI address of the location currently displayed by the Nautilus window, when a script is invoked from the right-click menu. |
| NAUTILUS\_SCRIPT\_WINDOW\_GEOMETRY | 828x511+251+342 | This environment variable is set to the on-screen position of the Nautilus window, when a script is invoked from the right-click menu. |

Program execution variables

Arguably the most powerful (but dangerous) environment variables, the following allow tweaking the basic way software actually runs.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Value Examples** | **What it's for** |
| LD\_PRELOAD | /usr/lib/valgrind.so | This variable can be used to inject a custom dynamic library into an application's memory when it loads. It can be used to do things like replacing the application's built-in memory allocation library with a debugging version in order to detect memory leaks. It can also be used to override the way it does various things like play sounds. |

Compilation and software development related variables

|  |  |  |
| --- | --- | --- |
| **Variable** | **Value Examples** | **What it's for** |
| CC | gcc | The name of the C compiler to use |
| CXX | g++ | The name of the C++ compiler to use |
| CFLAGS | -o out.o | A list of debugging/optimization flags to pass to the C compiler |
| CXXFLAGS | -Wall | A list of debugging/optimization flags to pass to the C++ compiler |
| CPPFLAGS | -DDEBUG | A list of flags to pass to the C/C++ pre-processor/compiler |
| LIBRARY\_PATH | /usr/lib/firefox | A list of directories (separated by colons) in which library files should be searched for |
| INCLUDE | /opt/app/src/include | A colon-separated list of directories in which header files should be searched for |
| CPATH | ..:$HOME/include:/usr/local/include | A colon-separated list of directories in which header files should be searched for |

Other environment variables

|  |  |  |
| --- | --- | --- |
| **Variable** | **Value Examples** | **What it's for** |
| USER | myuser1 | The name of the currently logged-in user. This variable is set by the system. You probably shouldn't change its value manually. |
| LOGNAME | myuser1 | Similar to "USER", some programs prefer to read this variable rather than USER. |
| HOME | /home/myuser1 | The location of the currently logged-in user's home directory. |
| PWD | /home/myuser1/Desktop | The current working directory of the shell. |
| SHELL | /bin/bash | The user's preferred command-line shell as it is set in the */etc/passwd* file. |
| POSIXLY\_CORRECT | N/A | If this environment variable exists, regardless of its value, it causes the behaviour of quite a few utilities to more closely match the behaviour defined by the POSIX standard. This typically makes various GNU extensions that make life easier not work, but it may just be what's needed to make an old UNIX script execute successfully. |
| HOSTALIASES | /etc/host.aliases | The name of a file containing host-name aliases for use with various network programs. |
| TZDIR | /usr/share/zoneinfo | The path of the directory containing time-zone information files. This typically does not need to be set manually, as the system searches for such files in */usr/share/zoneinfo* by default. |
| TZ | IST-2 :Japan | This variable was used by older UNIX systems to specify the system's time-zone. However, Ubuntu and most other modern systems use the */etc/localtime* file for that purpose. This variable can, however, be used to make one user's particular session display times in a different time-zone than the rest of the system. The value of this variable can either be the name and offset of a time-zone (as seen in the first example) or the name of a zone-info file in */usr/share/zoneinfo*(as seen in the second example). |
| TERM | xterm | The name of a terminal information file from */lib/terminfo*, this file instructs terminal programs how to achieve things such as displaying color. It may help to fiddle with this variable if you're trying use an odd terminal emulator program or trying to connect a hardware serial-port terminal emulator and getting undesired results. |
| TERMCAP |  | This variable can be used instead of "TERM" to manually specify terminal information rather than point to a file. |
| COLUMNS | 80 | The number of text columns in the terminal window. Try adjusting this variable if lines don't seem to wrap properly in the console. |
| LINES | 25 | The number of text lines on the console window. Try adjusting this variable if you're getting strange results when scrolling text. |
| SDL\_VIDEO\_FULLSCREEN\_DISPLAY | 0 | In multiple monitor setups, such as Xinerama or [TwinView](https://help.ubuntu.com/community/TwinView), this sets which screen to use for full-screen mode for SDL applications/games. If unset all screens are used. |