Previous: Environment Access, Up: Environment Variables [Contents][Index]

25.4.2 Standard Environment Variables

These environment variables have standard meanings. This doesn't mean that they are always present in the environment; but if these variables *are* present, they have these meanings. You shouldn't try to use these environment variable names for some other purpose.

`HOME`

This is a string representing the user's *home directory*, or initial default working directory.

The user can set `HOME` to any value. If you need to make sure to obtain the proper home directory for a particular user, you should not use `HOME`; instead, look up the user's name in the user database (see User Database).

For most purposes, it is better to use `HOME`, precisely because this lets the user specify the value.

`LOGNAME`

This is the name that the user used to log in. Since the value in the environment can be tweaked arbitrarily, this is not a reliable way to identify the user who is running a program; a function like `getlogin` (see Identifying Who Logged In) is better for that purpose.

For most purposes, it is better to use `LOGNAME`, precisely because this lets the user specify the value.

`PATH`

A *path* is a sequence of directory names which is used for searching for a file. The variable `PATH` holds a path used for searching for programs to be run.

The `execlp` and `execvp` functions (see Executing a File) use this environment variable, as do many shells and other utilities which are implemented in terms of those functions.

The syntax of a path is a sequence of directory names separated by colons. An empty string instead of a directory name stands for the current directory (see Working Directory).

A typical value for this environment variable might be a string like:

```
:/bin:/etc:/usr/bin:/usr/new/X11:/usr/new:/usr/local/bin
```

This means that if the user tries to execute a program named `foo`, the system will look for files named `foo`, `/bin/foo`, `/etc/foo`, and so on. The first of these files that exists is the one that is executed.

`TERM`

This specifies the kind of terminal that is receiving program output. Some programs can make use of this information to take advantage of special escape sequences or terminal modes supported by particular kinds of terminals. Many programs which use the termcap library (see Find in *The Termcap Library Manual*) use the `TERM` environment variable, for example.

`TZ`

This specifies the time zone. See Specifying the Time Zone with `TZ`, for information about the format of this string and how it is used.

`LANG`

This specifies the default locale to use for attribute categories where neither `LC_ALL` nor the specific environment variable for that category is set. See Locales and Internationalization, for more information about locales.

`LC_ALL`

If this environment variable is set it overrides the selection for all the locales done using the other `LC_*` environment variables. The value of the other `LC_*` environment variables is simply ignored in this case.

`LC_COLLATE`

This specifies what locale to use for string sorting.

`LC_CTYPE`

This specifies what locale to use for character sets and character classification.

`LC_MESSAGES`

This specifies what locale to use for printing messages and to parse responses.

`LC_MONETARY`

This specifies what locale to use for formatting monetary values.

`LC_NUMERIC`

This specifies what locale to use for formatting numbers.

`LC_TIME`

This specifies what locale to use for formatting date/time values.

`NLSPATH`

This specifies the directories in which the `catopen` function looks for message translation catalogs.

`_POSIX_OPTION_ORDER`

If this environment variable is defined, it suppresses the usual reordering of command line arguments by `getopt` and `argp_parse`. See Program Argument Syntax Conventions.

Previous: Environment Access, Up: Environment Variables [Contents][Index]