NAME

environ − user environment

SYNOPSIS

extern char *****environ*;**

DESCRIPTION

The variable *environ* points to an array of pointers to strings called the "environment". The last pointer in this array has the value NULL. This array of strings is made available to the process by the **execve**(2) call when a new program is started. When a child process is created via **fork**(2), it inherits a *copy* of its parent's environment.

By convention, the strings in *environ* have the form "*name*=*value*". The name is case-sensitive and may not contain the character "=". The value can be anything that can be represented as a string. The name and the value may not contain an embedded null byte ('\0'), since this is assumed to terminate the string.

Environment variables may be placed in the shell's environment by the *export* command in **sh**(1), or by the *setenv* command if you use **csh**(1).

The initial environment of the shell is populated in various ways, such as definitions from */etc/environment* that are processed by **pam_env**(8) for all users at login time (on systems that employ **pam**(8)). In addition, various shell initialization scripts, such as the system-wide */etc/profile* script and per-user initializations script may include commands that add variables to the shell's environment; see the manual page of your preferred shell for details.

Bourne-style shells support the syntax

```
NAME=value command
```

to create an environment variable definition only in the scope of the process that executes *command*. Multiple variable definitions, separated by white space, may precede *command*.

Arguments may also be placed in the environment at the point of an **exec**(3). A C program can manipulate its environment using the functions **getenv**(3), **putenv**(3), **setenv**(3), and **unsetenv**(3).

What follows is a list of environment variables typically seen on a system. This list is incomplete and includes only common variables seen by average users in their day-to-day routine. Environment variables specific to a particular program or library function are documented in the ENVIRONMENT section of the appropriate manual page.

USER The name of the logged-in user (used by some BSD-derived programs). Set at login time, see section NOTES below.

LOGNAME
The name of the logged-in user (used by some System-V derived programs). Set at login time, see section NOTES below.

HOME
A user's login directory. Set at login time, see section NOTES below.

LANG The name of a locale to use for locale categories when not overridden by **LC_ALL** or more specific environment variables such as **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, **LC_MONETARY**, **LC_NUMERIC**, and **LC_TIME** (see **locale**(7) for further details of the **LC_*** environment variables).

PATH The sequence of directory prefixes that **sh**(1) and many other programs employ when searching for an executable file that is specified as a simple filename (i.e., a pathname that contains no slashes). The prefixes are separated by colons (**:**). The list of prefixes is searched from beginning to end, by checking the pathname formed by concatenating a prefix, a slash, and the filename, until a file with execute permission is found.

As a legacy feature, a zero-length prefix (specified as two adjacent colons, or an initial or terminating colon) is interpreted to mean the current working directory. However, use of this feature is deprecated, and POSIX notes that a conforming application shall use an explicit pathname (e.g., .)

to specify the current working directory.

Analogously to **PATH**, one has **CDPATH** used by some shells to find the target of a change directory command, **MANPATH** used by **man**(1) to find manual pages, and so on.

PWD Absolute path to the current working directory; required to be partially canonical (no . or .. components).

SHELL
The absolute pathname of the user's login shell. Set at login time, see section NOTES below.

TERM The terminal type for which output is to be prepared.

PAGER
The user's preferred utility to display text files. Any string acceptable as a command-string operand to the *sh −c* command shall be valid. If**P AGER** is null or is not set, then applications that launch a pager will default to a program such as **less**(1) or **more**(1).

EDITOR/VISUAL
The user's preferred utility to edit text files. Any string acceptable as a command_string operand to the *sh −c* command shall be valid.

Note that the behavior of many programs and library routines is influenced by the presence or value of certain environment variables. Examples include the following:

- The variables **LANG**, **LANGUAGE**, **NLSPATH**, **LOCPATH**, **LC_ALL**, **LC_MESSAGES**, and so on influence locale handling; see **catopen**(3), **gettext**(3), and **locale**(7).
- **TMPDIR** influences the path prefix of names created by **tempnam**(3) and other routines, and the temporary directory used by **sort**(1) and other programs.
- **LD_LIBRARY_PATH**, **LD_PRELOAD**, and other **LD_*** variables influence the behavior of the dynamic loader/linker. See also**ld.so**(8).
- **POSIXLY_CORRECT** makes certain programs and library routines follow the prescriptions of POSIX.
- The behavior of **malloc**(3) is influenced by **MALLOC_*** variables.
- The variable **HOSTALIASES** gives the name of a file containing aliases to be used with **gethostbyname**(3).
- **TZ** and **TZDIR** give timezone information used by **tzset**(3) and through that by functions like **ctime**(3), **localtime**(3), **mktime**(3), **strftime**(3). See also**tzselect**(8).
- **TERMCAP** gives information on how to address a given terminal (or gives the name of a file containing such information).
- **COLUMNS** and **LINES** tell applications about the window size, possibly overriding the actual size.
- **PRINTER** or **LPDEST** may specify the desired printer to use. See **lpr**(1).

NOTES

Historically and by standard, *environ* must be declared in the user program. However, as a (nonstandard) programmer convenience, *environ* is declared in the header file <*unistd.h*> if the **_GNU_SOURCE** feature test macro is defined (see **feature_test_macros**(7)).

The **prctl**(2) **PR_SET_MM_ENV_START** and **PR_SET_MM_ENV_END** operations can be used to control the location of the process's environment.

The **HOME**, **LOGNAME**, **SHELL**, and **USER** variables are set when the user is changed via a session management interface, typically by a program such as **login**(1) from a user database (such as **passwd**(5)). (Switching to the root user using **su**(1) may result in a mixed environment where **LOGNAME** and **USER** are retained from old user; see the **su**(1) manual page.)

BUGS

Clearly there is a security risk here. Many a system command has been tricked into mischief by a user who specified unusual values for **IFS** or **LD_LIBRARY_PATH**.

There is also the risk of name space pollution. Programs like *make* and *autoconf* allow overriding of default utility names from the environment with similarly named variables in all caps. Thus one uses **CC** to select the desired C compiler (and similarly **MAKE**, **AR**, **AS**, **FC**, **LD**, **LEX**, **RM**, **YACC**, etc.). However, in some traditional uses such an environment variable gives options for the program instead of a pathname. Thus, one has **MORE** and **LESS**. Such usage is considered mistaken, and to be avoided in new programs.

SEE ALSO

bash(1), **csh**(1), **env**(1), **login**(1), **printenv**(1), **sh**(1), **su**(1), **tcsh**(1), **execve**(2), **clearenv**(3), **exec**(3), **getenv**(3), **putenv**(3), **setenv**(3), **unsetenv**(3), **locale**(7), **ld.so**(8), **pam_env**(8)