NAME

environ − user environment for Gno and Orca

SYNOPSIS

#include <stdlib.h>

extern char **environ;

DESCRIPTION

Under UNIX, an array of strings called the ‘environment’ is made available by **execve**(2) when a process begins. By convention these strings have the form *name=value*. However, because of the shell variable and environment passing mechanisms normally provided for the IIgs (and with Gno and Orca in particular), this method of controlling and accessing the environment has not previously been available. Specifically, under Gno and Orca **environ** is not initialized by the **exec**(3) family of calls. To accomodate this ´feature´, the library call **initenv**(3) has been provided to properly initialize **environ**.

The following names are used by various commands:

PATH The sequence of directory prefixes that **gsh**(1), **sh**(1), **time**(1V), **nice**(1), etc., apply in searching for a file known by an incomplete path name. The prefixes are delimited by spaces since colons are valid pathname characters under GS/OS.

HOME The name of the user’s login directory, set by **login**(1) from the password file **/etc/passwd** (see **passwd**(5)).

TERM The type of terminal on which the user is logged in. This information is used by commands, such as **nroff**(1) or **plot**(1G), which may exploit special terminal capabilities. See **/etc/termcap** (**termcap**(5)) for a list of terminal types.

SHELL The path name of the user’s login shell.

TERMCAP The string describing the terminal in TERM, or the name of the **termcap** file, see **termcap**(3X), **termcap**(5).

EXINIT A startup list of commands read by **ex**(1), **edit**, and **vi**(1).

USER or **LOGNAME** The user’s login name.

TZ The name of the time zone that the user is located in. If **TZ** is not present in the environment, the system’s default time zone, normally the time zone that the computer is located in, is used.

Further names may be placed in the environment by the *set* and *setenv* commands under **gsh**(1), the *export* command and *name=value* arguments in **sh**(1), or by the **setenv** command if you use **csh**(1). It is unwise to conflict with certain **sh**(1) variables that are frequently exported by **.profile** files: **MAIL**, **PS1**, **PS2**, **IFS**.

Various names are expected to be defined and used by locale dependent commands and functions (see **locale**(5)).

GNO IMPLEMENTATION

The Gno implementation of the environment actually has two sets of shell variables. The first is the NULL−terminated array of strings pointed to by **environ**. The second is the set of internal shell variables set by such system calls as **SetGS** and **UnsetVariableGS**. The latter of these is the one normally used by Orca/C. Unfortunately, the latter implementation makes the porting of some UNIX utilities more difficult. The **environ** library was created to solve this problem without adding significant overhead to environment calls when reference to **environ** is not needed.

When the **environ** variable is not needed, calls made to **putenv**(3), **setenv**(3), and **getenv**(3) result in only the internal shell variables to be set, and **environ** will remain NULL; its default value.

If, however, **environ** is to accessed, then an initial call must be made to **initenv**(3). This produces the array of strings based on the shell variables defined internally, and keeps the array current with further calls to **putenv**(3), or **setenv**(3).

CAVEATS

Under the Orca shell, three ‘extraneous’ shell variables are defined: **STATUS**, **Command**, and **Exit**. These are reset with every call to the Orca shell and should not, in general, be considered to be current.

When one of the **exec**(3) family of calls is carried out, the environment will not be passed on through **environ**. In order for the next process to use **environ**, it must make another call to **initenv**(3). This method will not work with children that have merely been forked.

In order to conduct an environment context switch, you must invoke the functions **pushenv** (3) and **popenv**(3). Simply performing a **PushVariableGS** or **PopVariableGS** will result in only the internal shell variables being pushed or popped.

While the **environ** array may be read by user applications, it cannot be directly modified with impudence; any additions or deletions to the **environ** array other that through the mentioned routines will not be reflected in the internal shell variables. Also, since environ entries are dynamically allocated and freed, modifying those entries without using the library routines may result in memory trashing and unpredicable behavior.

FILES

/usr/lib/lenviron
/etc/passwd
etc/termcap

BUGS

None are known at this time. Send bug reports to Devin Reade at one of the addresses below. If you cannot contact me at those addresses, look for me on the Internet comp.sys.apple2.* newsgroups.

AUTHORS

The environ library was created for Orca/Gno by:

G. Devin Reade −− glyn@cs.ualberta.ca *or* glyn@ersys.edmonton.ab.ca

It contains code fragments from:

Dave Tribby −− tribby@cup.hp.com *or* GEnie: D.TRIBBY
James Brookes −− jamesb@cscihp.ecst.csuchico.edu

The environ library also contains code is derived from software contributed to Berkeley by the American National Standards Committee X3, on Information Processing Systems.

SEE ALSO

csh(1), **ex**(1), **gsh**(1), **login**(1), **nice**(1), **nroff**(1), **plot**(1G), **sh**(1), **time**(1V), **vi**(1), **execve**(2), **initenv**(3), **getenv**(3), **popenv**(3), **pushenv**(3), **putenv**(3), **setenv**(3), **system**(3), **termcap**(3X), **locale**(5), **passwd**(5), **termcap**(5)