NAME

env - set environment and execute command, or print environment

SYNOPSIS

env [**-0iv**] [**-L**|**-U** *user*[/*class*]] [**-u** *name*] [*name=value ...*]
env [**-iv**] [**-L**|**-U** *user*[/*class*]] [**-P** *altpath*] [**-S** *string*] [**-u** *name*] [*name=value ...*] *utility* [*argument ...*]

DESCRIPTION

The **env** utility executes another *utility* after modifying the environment as specified on the command line. Each *name=value* option specifies the setting of an environment variable, *name*, with a value of *value*. All such environment variables are set before the *utility* is executed.

The options are as follows:

-0 End each output line with NUL, not newline.

-i Execute the *utility* with only those environment variables specified by *name=value* options. The environment inherited by **env** is ignored completely.

-L | **-U** *user*[/*class*]
Add the environment variable definitions from login.conf(5) for the specified user and login class to the environment, after processing any **-i** or **-u** options, but before processing any *name=value* options. If **-L** is used, only the system-wide */etc/login.conf.db* file is read; if **-U** is used, then the specified user's *~/.login_conf* is read as well. The user may be specified by name or by uid. If a username of '-' is given, then no user lookup will be done, the login class will default to 'default' if not explicitly given, and no substitutions will be done on the values.

-P *altpath*
Search the set of directories as specified by *altpath* to locate the specified *utility* program, instead of using the value of the PATH environment variable.

-S *string*
Split apart the given *string* into multiple strings, and process each of the resulting strings as separate arguments to the **env** utility. The **-S** option recognizes some special character escape sequences and also supports environment-variable substitution, as described below.

-u *name*
If the environment variable *name* is in the environment, then remove it before processing the remaining options. This is similar to the **unset** command in sh(1). The value for *name* must not include the '=' character.

-v Print verbose information for each step of processing done by the **env** utility. Additional information will be printed if **-v** is specified multiple times.

The above options are only recognized when they are specified before any *name=value* options.

If no *utility* is specified, **env** prints out the names and values of the variables in the environment. Each name/value pair is separated by a new line unless **-0** is specified, in which case name/value pairs are separated by NUL. Both **-0** and *utility* may not be specified together.

Details of -S (split-string) processing

The processing of the **-S** option will split the given *string* into separate arguments based on any space or <tab> characters found in the *string*. Each of those new arguments will then be treated as if it had been specified as a separate argument on the original **env** command.

Spaces and tabs may be embedded in one of those new arguments by using single ("'") or double ('"') quotes, or backslashes ('\'). Single quotes will escape all non-single quote characters, up to the matching single quote. Double quotes will escape all non-double quote characters, up to the matching double quote. It is an error if the end of the *string* is reached before the matching quote character.

If **-S** would create a new argument that starts with the '#' character, then that argument and the remainder of the *string* will be ignored. The '\#' sequence can be used when you want a new argument to start with a '#' character, without causing the remainder of the *string* to be skipped.

While processing the *string* value, **-S** processing will treat certain character combinations as escape sequences which represent some action to take. The character escape sequences are in backslash notation. The characters and their meanings are as follows:

\c Ignore the remaining characters in the *string*. This must not appear inside a double-quoted string.

\f Replace with a <form-feed> character.

\n Replace with a <new-line> character.

\r Replace with a <carriage return> character.

\t Replace with a <tab> character.

\v Replace with a <vertical tab> character.

\# Replace with a '#' character. This would be useful when you need a '#' as the first character in one of the arguments created by splitting apart the given *string*.

\$ Replace with a '$' character.

_ If this is found inside of a double-quoted string, then replace it with a single blank. If this is found outside of a quoted string, then treat this as the separator character between new arguments in the original *string*.

\" Replace with a <double quote> character.
\’ Replace with a <single quote> character.
\\ Replace with a backslash character.

The sequences for <single-quote> and backslash are the only sequences which are recognized inside of a single-quoted string. The other sequences have no special meaning inside a single-quoted string. All escape sequences are recognized inside of a double-quoted string. It is an error if a single ‘\’ character is followed by a character other than the ones listed above.

The processing of **-S** also supports substitution of values from environment variables. To do this, the name of the environment variable must be inside of ‘${}’, such as: ${SOMEVAR}. The common shell syntax of $SOMEVAR is not supported. All values substituted will be the values of the environment variables as they were when the **env** utility was originally invoked. Those values will not be checked for any of the escape sequences as described above. And any settings of *name=value* will not effect the values used for substitution in **-S** processing.

Also, **-S** processing cannot reference the value of the special parameters which are defined by most shells. For instance, **-S** cannot recognize special parameters such as: ‘$*’, ‘$@’, ‘$#’, ‘$?’ or ‘$$’ if they appear inside the given *string*.

Use in shell-scripts

The **env** utility is often used as the *interpreter* on the first line of interpreted scripts, as described in execve(2).

Note that the way the kernel parses the ‘#!’ (first line) of an interpreted script has changed as of FreeBSD 6.0. Prior to that, the FreeBSD kernel would split that first line into separate arguments based on any whitespace (space or <tab> characters) found in the line. So, if a script named */usr/local/bin/someport* had a first line of:

```
#!/usr/local/bin/php -n -q -dsafe_mode=0
```

then the */usr/local/bin/php* program would have been started with the arguments of:

```
arg[0] = ’/usr/local/bin/php’
arg[1] = ’-n’
arg[2] = ’-q’
arg[3] = ’-dsafe_mode=0’
arg[4] = ’/usr/local/bin/someport’
```

plus any arguments the user specified when executing *someport*. However, this processing of multiple

options on the ‘#!’ line is not the way any other operating system parses the first line of an interpreted script. So after a change which was made for FreeBSD 6.0 release, that script will result in */usr/local/bin/php* being started with the arguments of:

arg[0] = ’/usr/local/bin/php’
arg[1] = ’-n -q -dsafe_mode=0’
arg[2] = ’/usr/local/bin/someport’

plus any arguments the user specified. This caused a significant change in the behavior of a few scripts. In the case of above script, to have it behave the same way under FreeBSD 6.0 as it did under earlier releases, the first line should be changed to:

#!/usr/bin/env -S /usr/local/bin/php -n -q -dsafe_mode=0

The **env** utility will be started with the entire line as a single argument:

arg[1] = ’-S /usr/local/bin/php -n -q -dsafe_mode=0’

and then **-S** processing will split that line into separate arguments before executing */usr/local/bin/php*.

ENVIRONMENT

The **env** utility uses the PATH environment variable to locate the requested *utility* if the name contains no ‘/’ characters, unless the **-P** option has been specified.

EXIT STATUS

The **env** utility exits 0 on success, and >0 if an error occurs. An exit status of 126 indicates that *utility* was found, but could not be executed. An exit status of 127 indicates that *utility* could not be found.

EXAMPLES

Since the **env** utility is often used as part of the first line of an interpreted script, the following examples show a number of ways that the **env** utility can be useful in scripts.

The kernel processing of an interpreted script does not allow a script to directly reference some other script as its own interpreter. As a way around this, the main difference between

#!/usr/local/bin/foo

and

#!/usr/bin/env /usr/local/bin/foo

is that the latter works even if */usr/local/bin/foo* is itself an interpreted script.

Probably the most common use of **env** is to find the correct interpreter for a script, when the interpreter may be in different directories on different systems. The following example will find the ‘perl’ interpreter by searching through the directories specified by PATH.

```
#!/usr/bin/env perl
```

One limitation of that example is that it assumes the user’s value for PATH is set to a value which will find the interpreter you want to execute. The **-P** option can be used to make sure a specific list of directories is used in the search for *utility*. Note that the **-S** option is also required for this example to work correctly.

```
#!/usr/bin/env -S -P/usr/local/bin:/usr/bin perl
```

The above finds ‘perl’ only if it is in */usr/local/bin* or */usr/bin*. That could be combined with the present value of PATH, to provide more flexibility. Note that spaces are not required between the **-S** and **-P** options:

```
#!/usr/bin/env -S-P/usr/local/bin:/usr/bin:${PATH} perl
```

COMPATIBILITY

The **env** utility accepts the **-** option as a synonym for **-i**.

SEE ALSO

printenv(1), sh(1), execvp(3), login.conf(5), environ(7)

STANDARDS

The **env** utility conforms to IEEE Std 1003.1-2001 ("POSIX.1"). The **-0**, **-L**, **-P**, **-S**, **-U**, **-u** and **-v** options are non-standard extensions supported by FreeBSD, but which may not be available on other operating systems.

HISTORY

The **env** command appeared in 4.4BSD. The **-P**, **-S** and **-v** options were added in FreeBSD 6.0. The **-0**, **-L** and **-U** options were added in FreeBSD 13.0.

BUGS

The **env** utility does not handle values of *utility* which have an equals sign (‘=’) in their name, for obvious reasons.

The **env** utility does not take multibyte characters into account when processing the **-S** option, which may lead to incorrect results in some locales.