execlp(3) - Linux man page

Name

execl, execlp, execle, execv, execvp, execvpe - execute a file

Synopsis

#include <unistd.h>

extern char **environ;

int execl(const char **path*, **const char** **arg*, **...);**
int execlp(const char **file*, **const char** **arg*, **...);**
int execle(const char **path*, **const char** **arg*,
..., char * const *envp***[]);**
int execv(const char **path*, **char *const** *argv***[]);**
int execvp(const char **file*, **char *const** *argv***[]);**
int execvpe(const char **file*, **char *const** *argv***[],**
char *const *envp***[]);**

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

execvpe(): _GNU_SOURCE

Description

The **exec**() family of functions replaces the current process image with a new process image. The functions described in this manual page are front-ends for **execve**(2). (See the manual page for **execve**(2) for further details about the replacement of the current process image.)

The initial argument for these functions is the name of a file that is to be executed.

The *const char *arg* and subsequent ellipses in the **execl**(), **execlp**(), and **execle**() functions can be thought of as *arg0*, *arg1*, ..., *argn*. Together they describe a list of one or more pointers to null-terminated strings that represent the argument list available to the executed program. The first argument, by convention, should point to the filename associated with the file being executed. The list of arguments *must* be terminated by a NULL pointer, and, since these are variadic functions, this pointer must be cast *(char *) NULL*.

The **execv**(), **execvp**(), and **execvpe**() functions provide an array of pointers to null-terminated strings that represent the argument list available to the new program. The first argument, by convention, should point to the filename associated with the file being executed. The array of pointers *must* be terminated by a NULL pointer.

The **execle**() and **execvpe**() functions allow the caller to specify the environment of the executed program via the argument *envp*. The *envp* argument is an array of pointers to null-terminated strings and *must* be terminated by a NULL pointer. The other functions take the environment for the new process image from the external variable *environ* in the calling process.

Special semantics for execlp() and execvp()

The **execlp**(), **execvp**(), and **execvpe**() functions duplicate the actions of the shell in searching for an executable file if the specified filename does not contain a slash (/) character. The file is sought in the colon-separated list of directory pathnames specified in the **PATH** environment variable. If this variable isn't defined, the path list defaults to the current directory followed by the list of directories returned by *confstr(_CS_PATH)*. (This **confstr**(3) call typically returns the value "/bin:/usr/bin".)

If the specified filename includes a slash character, then **PATH** is ignored, and the file at the specified pathname is executed.

In addition, certain errors are treated specially.

If permission is denied for a file (the attempted **execve**(2) failed with the error **EACCES**), these functions will continue searching the rest of the search path. If no other file is found, however, they will return with *errno* set to **EACCES**.

If the header of a file isn't recognized (the attempted **execve**(2) failed with the error **ENOEXEC**), these functions will execute the shell (*/bin/sh*) with the path of the file as its first argument. (If this attempt fails, no further searching is done.)

Return Value

The **exec**() functions only return if an error has occurred. The return value is -1, and *errno* is set to indicate the error.

Errors

All of these functions may fail and set *errno* for any of the errors specified for **execve**(2).

Versions

The **execvpe**() function first appeared in glibc 2.11.

Conforming To

POSIX.1-2001, POSIX.1-2008.

The **execvpe**() function is a GNU extension.

Notes

On some other systems, the default path (used when the environment does not contain the variable **PATH**) has the current working directory listed after */bin* and */usr/bin*, as an anti-Trojan-horse measure. Linux uses here the traditional "current directory first" default path.

The behavior of **execlp**() and **execvp**() when errors occur while attempting to execute the file is historic practice, but has not traditionally been documented and is not specified by the POSIX standard. BSD (and possibly other systems) do an automatic sleep and retry if **ETXTBSY** is encountered. Linux treats it as a hard error and returns immediately.

Traditionally, the functions **execlp**() and **execvp**() ignored all errors except for the ones described above and **ENOMEM** and **E2BIG**, upon which they returned. They now return if any error other than the ones described above occurs.

See Also

sh(1), **execve**(2), **fork**(2), **ptrace**(2), **fexecve**(3), **environ**(7)

Referenced By

explain(1), **explain**(3), **explain_execlp**(3), **explain_execlp_or_die**(3), **system**(3)