Full credit is given to the above companies including the OS that this PDF file was generated!

## Rocky Enterprise Linux 9.2 Manual Pages on command 'execle.3'

**$ man execle.3**

EXEC(3)                    Linux Programmer's Manual                    EXEC(3)

NAME

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

SYNOPSIS

    #include <unistd.h>

    extern char **environ;

    int execl(const char *pathname, const char *arg, ...

        /* (char  *) NULL */);

    int execlp(const char *file, const char *arg, ...

        /* (char  *) NULL */);

    int execle(const char *pathname, const char *arg, ...

        /*, (char *) NULL, char *const envp[] */);

    int execv(const char *pathname, 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 layered  on  top  of 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  functions can be grouped based on the letters following the "exec" prefix.

l - execl(), execlp(), execle()

The const char *arg and subsequent ellipses 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.

By contrast with the 'l' functions, the 'v' functions  (below)  specify the command-line arguments of the executed program as a vector.

v - execv(), execvp(), execvpe()

The  char *const argv[] argument is an array of pointers to null-termi? nated strings that represent the argument list  available  to  the  new program.   The first argument, by convention, should point to the file? name associated with the file being executed.  The  array  of  pointers must be terminated by a null pointer.

e - execle(), execvpe()

The  environment of the caller is specified via the argument envp.  The envp argument is an array of pointers to  null-terminated  strings  and must be terminated by a null pointer.

All  other  exec()  functions  (which do not include 'e' in the suffix) take the environment for the new process image from the external  vari? able environ in the calling process.

p - execlp(), execvp(), execvpe()

These 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 a list that includes the di? rectories returned by confstr(_CS_PATH) (which typically returns the value "/bin:/usr/bin") and possibly also the current working directory; see NOTES for further details.

If the specified filename includes a slash character, then PATH is ig? nored, 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.)

All other exec() functions (which do not include 'p' in the suffix) take as their first argument a (relative or absolute) pathname that identifies the program to be executed.

RETURN VALUE

The exec() functions return only 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.

ATTRIBUTES

For an explanation of the terms used in this section, see at? tributes(7).

```
??????????????????????????????????????????????????????????????????
?Interface                ? Attribute   ? Value      ?
??????????????????????????????????????????????????????????????????
?execl(), execle(), execv()   ? Thread safety ? MT-Safe    ?
??????????????????????????????????????????????????????????????????
?execlp(), execvp(), execvpe() ? Thread safety ? MT-Safe env ?
??????????????????????????????????????????????????????????????????
```

## CONFORMING TO

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

The execvpe() function is a GNU extension.

## NOTES

The default search path (used when the environment does not contain the variable  PATH)  shows some variation across systems.  It generally in?cludes /bin and /usr/bin (in that order) and may also include the  cur?rent  working directory.  On some other systems, the current working is included after /bin and /usr/bin, as an anti-Trojan-horse measure.  The glibc  implementation  long  followed the traditional default where the current working directory is included at the start of the search  path.  However,  some  code  refactoring  during the development of glibc 2.24 caused the current working directory to be dropped altogether from  the default  search  path.   This  accidental behavior change is considered mildly beneficial, and won't be reverted.

The behavior of execlp() and execvp() when errors occur while  attempt?ing 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  de?scribed above occurs.

## BUGS

Before  glibc 2.24, execl() and execle() employed realloc(3) internally

and were consequently not async-signal-safe, in violation  of  the  re?

quirements of POSIX.1.  This was fixed in glibc 2.24.

Architecture-specific details

On  sparc and sparc64, execv() is provided as a system call by the ker?

nel (with the prototype shown  above)  for  compatibility  with  SunOS.

This  function is not employed by the execv() wrapper function on those

architectures.

SEE ALSO

sh(1), execve(2), execveat(2),  fork(2),  ptrace(2),  fexecve(3),  sys?

tem(3), environ(7)

COLOPHON

This  page  is  part of release 5.10 of the Linux man-pages project.  A

description of the project, information about reporting bugs,  and  the

latest    version    of    this   page,   can   be   found   at

https://www.kernel.org/doc/man-pages/.

GNU                    2019-08-02                    EXEC(3)