[Main]

Exec Family of Functions

```
#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, ...)

int
exect(const char *path, char *const argv[], char *const envp[])

int
execv(const char *path, char *const argv[])

int
execvp(const char *file, char *const argv[])
```

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 the function **execve(2)**. (See the manual page for ex-**ecve(2)** for detailed information about the replacement of the current process.)

The initial argument for these functions is the pathname of a file which 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 file name associated with the file being executed. The list of arguments *must* be terminated by a NULL pointer.

The **exect**(), **execv**(), and **execvp**() 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

file name associated with the file begin executed. The array of pointers **must** be terminated by a NULL pointer.

The **execle**() and **exect**() functions also specify the environment of the executed process by following the NULL pointer that terminates the list of arguments in the parameter list or the pointer to the argv array with an additional parameter. This additional parameter 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 stat the file to determine whether the file exists and has suitable execute permissions. If it does, they will return immediately with the global variable *errno* restored to the value set by **execve**(). Otherwise, the search will be continued. If the search completes without performing a successful **execve**() or terminating due to an error, these functions will return with the global variable *errno* set to EACCES or ENOENT according to whether at least one file with suitable execute permissions was found.

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

The function **exect**() executes a file with the program tracing facilities enabled (see **ptrace(2)**).

RETURN VALUES

If any of the **exec**() functions returns, an error will have occurred. The return value is -1, and the global variable *errno* will be set to indicate the error.

FILES

/bin/sh The shell.

ERRORS

Execl(), **execle**(), **execlp**() and **execvp**() may fail and set *errno* for any of the errors specified for the library functions **execve(2)** and **malloc(3)**.

Exect() and **execv**() may fail and set *errno* for any of the errors specified for the library function **execve(2)**.

SEE ALSO

sh(1), execve(2), fork(2), ktrace(2), ptrace(2), environ(7).

COMPATIBILITY

Historically, the default path for the **execlp**() and **execvp**() functions was ``*:/bin:/usr/bin*''. This was changed to place the current directory last to enhance system security.

The behavior of **execlp**() and **execvp**() when errors occur while attempting to execute the file is not quite historic practice, and has not traditionally been documented and is not specified by the POSIX standard.

Traditionally, the functions **execlp**() and **execvp**() ignored all errors except for the ones described above and ETXTBSY, upon which they retried after sleeping for several seconds, and ENOMEM and E2BIG, upon which they returned. They now return for ETXTBSY, and determine existence and executability more carefully. In particular, EACCES for inaccessible directories in the path prefix is no longer confused with EACCES for files with unsuitable execute permissions. In 4.4BSD, they returned upon all