NAME

execl, execv, execle, execlp, execvp, exec, environ − execute a file

SYNOPSIS

#include <unistd.h>

int execl(const char ***_name_, const char ***_arg0_**, ..., (char *) NULL)**
int execv(const char ***_name_, char *const** _argv_**[])**
int execle(const char ***_name_, const char ***_arg0_**, ..., (char *) NULL, char *const** _envp_**[])**
int execlp(const char ***_name_, const char ***_arg0_**, ..., (char *) NULL)**
int execvp(const char ***_name_, char *const** _argv_**[])**

extern char *const *environ;

DESCRIPTION

These routines provide various interfaces to the **execve** system call. Refer to **execve**(2) for a description of their properties; only brief descriptions are provided here.

Exec in all its forms overlays the calling process with the named file, then transfers to the entry point of the core image of the file. There can be no return from a successful exec; the calling core image is lost.

The *name* argument is a pointer to the name of the file to be executed. The pointers *arg*[*0*], *arg*[*1*] ... address null-terminated strings. Conventionally *arg*[*0*] is the name of the file.

Two interfaces are available. **execl** is useful when a known file with known arguments is being called; the arguments to **execl** are the character strings constituting the file and the arguments; the first argument is conventionally the same as the file name (or its last component). A null pointer argument must end the argument list. (Note that the **execl*** functions are variable argument functions. This means that the type of the arguments beyond *arg0* is not checked. So the null pointer requires an explicit cast to type **(char *)** if not of that type already.)

The **execv** version is useful when the number of arguments is unknown in advance; the arguments to **execv** are the name of the file to be executed and a vector of strings containing the arguments. The last argument string must be followed by a null pointer.

When a C program is executed, it is called as follows:

> **int main(int** _argc_**, char *const** _argv_**[], char *const** _envp_**[]);**
>
> **exit(main(**_argc_**,** _argv_**,** _envp_**));**

where *argc* is the argument count and *argv* is an array of character pointers to the arguments themselves. As indicated, *argc* is conventionally at least one and the first member of the array points to a string containing the name of the file.

Argv is directly usable in another **execv** because *argv*[*argc*] is 0.

Envp is a pointer to an array of strings that constitute the *environment* of the process. Each string consists of a name, an =, and a null-terminated value. The array of pointers is terminated by a null pointer. The shell **sh**(1) passes an environment entry for each global shell variable defined when the program is called. See **environ**(7) for some conventionally used names. The C run-time start-off routine places a copy of *envp* in the global cell **environ**, which is used by **execv** and **execl** to pass the environment to any subprograms executed by the current program.

Execlp and **execvp** are called with the same arguments as **execl** and **execv**, but duplicate the shell's actions in searching for an executable file in a list of directories. The directory list is obtained from the environment variable **PATH**. Under standard Minix, if a file is found that is executable, but does not have the proper executable header then it is assumed to be a shell script. **Execlp** and **execvp** execute **/bin/sh** to interpret the script. Under Minix-vmd this does not happen, a script must begin with **#!** and the full path name of the interpreter if it is to be an executable script.

SEE ALSO

execve(2), **fork**(2), **environ**(7), **sh**(1).

DIAGNOSTICS

If the file cannot be found, if it is not executable, if it does not start with a valid magic number (see **a.out**(5)), if maximum memory is exceeded, or if the arguments require too much space, a return constitutes the diagnostic; the return value is −1 and **errno** is set as for **execve**. Even for the super-user, at least one of the execute-permission bits must be set for a file to be executed.