NAME

    exec, execl, execv – execute a file

SYNOPSIS

    (exec = 11.)
    sys exec; name; args
    name: <...\0>
    args: arg0; arg1; ...; 0
    arg0: <...\0>
    arg1: <...\0>
      ...

    execl(name, arg0, arg1, ..., argn, 0)
    char *name, *arg0, *arg1, ..., *argn;

    execv(name, argv)
    char *name;
    char *argv[ ];

DESCRIPTION

    Exec overlays the calling process with the named file, then transfers to the beginning of the core image of the file. There can be no return from the file; the calling core image is lost.

    Files remain open across *exec* calls. Ignored signals remain ignored across *exec,* but signals that are caught are reset to their default values.

    Each user has a *real* user ID and group ID and an *effective* user ID and group ID. The real ID identifies the person using the system; the effective ID determines his access privileges. *Exec* changes the effective user and group ID to the owner of the executed file if the file has the "set-user-ID" or "set-group-ID" modes. The real user ID is not affected.

    The form of this call differs somewhat depending on whether it is called from assembly language or C; see below for the C version.

    The first argument to *exec* is a pointer to the name of the file to be executed. The second is the address of a null-terminated list of pointers to arguments to be passed to the file. Conventionally, the first argument is the name of the file. Each pointer addresses a string terminated by a null byte.

    Once the called file starts execution, the arguments are available as follows. The stack pointer points to a word containing the number of arguments. Just above this number is a list of pointers to the argument strings. The arguments are placed as high as possible in core.

      sp→   nargs
              arg0
              ...
              argn
              −1

      arg0:   <arg0\0>
              ...
      argn: <argn\0>

    From C, 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; as in the basic call, the first argument is conventionally the same as the file name (or its last component). A 0 argument must end the argument list.

- 2 -

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 0 pointer.

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

        main(argc, argv)
        int argc;
        char **argv;

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 not directly usable in another *execv,* since *argv[argc]* is −1 and not 0.

SEE ALSO

fork(II), pexec(III)

DIAGNOSTICS

If the file cannot be found, if it is not executable, if it does not have a valid header (407, 410, or 411 octal as first word), if maximum memory is exceeded, or if the arguments require more than 5120 bytes a return from *exec* constitutes the diagnostic; the error bit (c-bit) is set. Even for the super-user, at least one of the execute-permission bits must be set for a file to be executed. From C the returned value is −1.

BUGS

Only 5120 characters of arguments are allowed.