CALL(II) CALL(II)

NAME

call, lcall, vcall – create and execute a new process

SYNOPSIS

```
(call = 58.)  /* used to be 57 in previous systems*/
sys call; name; args
...
name: <...\0>
...
args: arg0; arg1; ...; 0
arg0: <...\0>
arg1: <...\0>
  ...

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

vcall(name, argv)
char *name;
char *argv[ ];
```

DESCRIPTION

Call performs a combined *fork* and *exec* in one step. The newly created child process begins execution at the beginning of the core image of the file and the parent process continues execution without waiting for the child to terminate. *Call* returns the process id of the newly created process.

Files remain open across *call* calls. All signals are reset to the system's default value when *call* is executed.

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. *Call* 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 first argument to *call* 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 in the *exec* system call.

From C, two interfaces are available, *lcall* and *vcall*. *lcall* is useful when a known file with known arguments is being called; the arguments to *lcall* are the character strings constituting the file name 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. The *vcall* version is useful when the number of arguments is unknown in advance; the arguments to *vcall* 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.

SEE ALSO

fork (II), exec(II)

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 10 * 512 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 on error. The process id of the new process is returned on a success.

BUGS

Under previous MERT systems, *call*(f) used to return only after the child had terminated. This new implementation returns after the child has been created.