posix

ıot defined by this standard is implementation-defined.

3.1.1.3 Returns. Upon successful completion, *fork*() shall return to the hild process a value of zero and shall return to the parent process the process D of the child process, and both processes shall continue to execute from the *rk*() function. Otherwise, a value of −1 shall be returned to the parent process, o child process shall be created, and *errno* shall be set to indicate the error.

3.1.1.4 Errors. If any of the following conditions occur, the *fork*() funcon shall return −1 and set *errno* to the corresponding value:

[EAGAIN]	The system lacked the necessary resources to create another process, or the system-imposed limit on the total number of processes under execution by a single user would be exceeded.

For each of the following conditions, if the condition is detected, the *fork*() ınction shall return −1 and set *errno* to the corresponding value:

[ENOMEM]	The process requires more space than the system is able to supply.

3.1.1.5 References. *alarm*() §3.4.1, *exec* §3.1.2, *fcntl*() §6.5.2, *kill*() .3.2, *times*() §4.5.2, *wait* §3.2.1.

3.1.2 Execute a File.

ınctions: *execl*(), *execv*(), *execle*(), *execve*(), *execlp*(), *execvp*()

3.1.2.1 Synopsis.

```
int execl (path, arg0, arg1, ..., argn, (char *) 0)
char *path, *arg0, *arg1, ..., *argn;

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

int execle (path, arg0, arg1, ..., argn, (char *) 0, envp)
char *path, *arg0, *arg1, ..., *argn, *envp[ ];

int execve (path, argv, envp);
char *path, *argv[ ], *envp[ ];

int execlp (file, arg0, arg1, ..., argn, (char *) 0)
char *file, *arg0, *arg1, ..., *argn;

int execvp (file, argv)
char *file, *argv[ ];

extern char **environ;
```

path – exact path
or
in this directory

file – uses PATH
environ. variable

3.1.2.2 Description. The *exec* family of functions shall replace the nt process image with a new process image. The new image is constructed a regular, executable file called the *new process image file*. There shall be turn from a successful *exec*, because the calling process image is overlaid ꞉ new process image.

en a C program is executed as a result of this call, it shall be entered as a ȝuage function call as follows:

wh
the
is i
stri
Th
T
pa
T
im
T
ces
pro
env
env
im
T
ter
ces
sho
onc
T
str
cor
arg
sta
T
str
Th
I
exe
va
en
ter
nu
ce
fcn
att