Fork

Permite crear un proceso hijo.

Padre:	PID del hijo
Hijo:	0
Error:	-1

exec

Pone en ejecución un programa. No cambia la identidad del proceso.

execlp	argumentos en lista
execvp	argumentos en vector a punteros (argv[])

herencia

PID,PPID,PGID, UID GID	variables de entorno
señales pendientes	tabla de descriptores

Diferente

EUID,EGID	comportamiento ante señales

execlp

execlp(comando,comando,argumento1,....,0)

Comando dos veces

execvp

execvp(argv[1],&argv[1])

wait // waitpid(pid_hijo,p_estado,options)

error	-1 (si no hay hijos) // errno =10

Terminación

voluntaria	byte + significativo

proceso zombie

Cuando el padre no ha hecho wait

proceso huérfano

Cuando el padre se cierra dejando hijios activos. Son adoptados por el init(pid=1)

strtok(s,"_")

La primera vez devuelve el puntero donde empieza la primera palabra y escribe un \0 donde acaba

Siguientes veces usar null en vez de s.

Identificadores

Usuario

UID:	id de usuario(0 para su)
GID:	id del grupo

Fichero

i-nodo	número de inodo
UID:	propietario
GUID:	grupo propietario

permisos de acceso(user-group-other)

set-user-id	usuario efectivo
set-group-id	grupo efectivo
sticky bit	gestión en directorio compartido(\tmp)

Proceso

PID	pid_t getpid() //id padre
PGID	pid_t getpgrp() //id del grupo
UID	uid_t getuid() // usuario real
GID	gid_t getgid() // id del grupo usuario
EUID	uid_t geteuid() //usuario efectivo
	si(set-user-id==1) then EUID=UID
EGID	gid_t getegid // grupo efectivo

By **Cesarblancg**
cheatography.com/cesarblancg/

Not published yet.
Last updated 31st January, 2019.
Page 1 of 1.