EXEC y EXECV

Procesos en Linux

¿Qué es exec()?

La familia de funciones exec() tiene como objetivo reemplazar la imagen de proceso actual por una nueva imagen. Esta nueva imagen debe estar conformada por un archivo ejecutable. En caso de que la ejecución sea exitosa, exec() continuará ejecutando el proceso hijo y no procederá a ejecutar las líneas de código que le siguen. De esta forma sabemos que en caso de error, se ejecutará la siguiente línea (donde normalmente se informa el error).

Debido a que estas funciones no crean un proceso nuevo, el PID (process identifier) no sufre modificaciones; las instrucciones ejecutadas por el CPU, y la pila del proceso son cambiados por los del nuevo programa.

Entre las funciones que integran la familia de exec() se destacan:

execv(), execvp(), execve(), execl(), execle() y execlp()

Entendiendo la sintaxis

- **L vs V:** Se refiere al formato de los parámetros que le pasamos a la función,
  - L: parámetros individuales en la llamada (lista variable de argumentos), utilizados en: execl(), execle(), execlp(), and execlpe()
  - V: un array del tipo char*: excev(), execve(), execvp(), execvpe()

  Pasar un array es útil cuando deseamos utilizar una cantidad desconocida de parámetros, de esta manera evitamos especificar un número fijo de variables.

- **E:** Las versiones que contienen la letra “e” te permiten pasar adicionalmente un array del tipo char* compuesto por un conjunto de strings que serán incluidos en el entorno del proceso antes de que el programa especificado en la función sea ejecutado.

- **P:** Las versiones que contienen la letra “p” utilizan las variables de entorno para buscar el archivo ejecutable indicado. Las versiones que NO tienen la letra “p” requieren la ruta absoluta o relativa en el caso de que el ejecutable no se encuentre en el directorio actual.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(int argc, char *argv[]) {
        pid_t pid = fork();
        int status;

        if (pid == 0) {
                // Proceso hijo
                execv("/home/programa", argv);  // programa en bash + argumentos
                exit(127); // si algo sale mal, mostramos un error
        } else {
                // Esperamos a que termine el proceso
                waitpid(pid, &status, 0);
                // Ejecutamos el resto del codigo sobre el proceso padre
                if (status == 0) {
                        printf("mundo\n");
                }
        }

        return 0;
}
```

```bash
#!/bin/bash
ps -f
echo "$1"
```