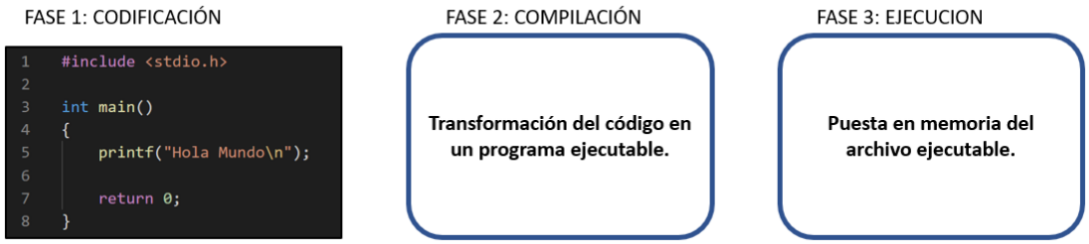
Procesos, estructura y finalización

Fases de un programa



Aquí se focalizará en la fase 3: cuando el programa se ejecuta, se sitúa en la memoria principal, por lo que el sistema operativo comienza a ejecutar sus instrucciones. La finalización de éste puede darse al cancelar su ejecución, o al terminar de ejecutarse las instrucciones del código.

Cuando se produce la ejecución del programa, el mismo pasa a denominarse “proceso”.

Proceso: definición y recursos

Se puede pensar en un proceso como una porción de un programa en ejecución, o todo el programa.

Los procesos son tareas separadas unas de otras con sus propios derechos y responsabilidades, hacen uso de la CPU del sistema para realizar sus instrucciones, de la memoria para almacenarse en él y a sus datos, además de usar ficheros y hasta los dispositivos conectados.

Por lo tanto, un proceso necesita recursos para llevar a cabo su tarea. Estos son:

- Tiempo de CPU.
- Memoria.
- Datos.
- Archivos y dispositivos de E/S.

Estos recursos se otorgan al proceso en el momento en que se crea, o bien se le asignan durante su ejecución.

Se puede decir que un proceso es la unidad de trabajo de un sistema y el "sistema" consiste en una colección de procesos.

Existen procesos que ejecutan el código del sistema operativo, y procesos de usuario que ejecutan código de algún usuario.

Proceso: estados

A medida que un proceso se ejecuta, cambia de estado. Cada proceso puede estar en uno de los estados:

- **Nuevo** (*new*): el proceso se está creando.
- **En ejecución** (*running*): el proceso está en la CPU ejecutando instrucciones.
- **Bloqueado** (*waiting*, en espera): proceso esperando a que ocurra un suceso (ejemplo: terminación de E/S o recepción de una señal).
- **Preparado** (*ready*, listo): esperando que se le asigne a un procesador.
- **Terminado** (*terminated*): finalizó su ejecución, por tanto no ejecuta más instrucciones y el Sistema Operativo (SO) le retirará los recursos que consume.

Proceso: características

Un programa puede ejecutarse más de una vez, generándose más de una instancia del mismo programa. Un proceso incluye no sólo el programa que ejecuta, sino toda la información necesaria para diferenciar una ejecución del programa de otra.

Con cada ejecución, el SO genera distintos identificadores a cada proceso:

- **PID** (*Process ID*): número de referencia único que tiene cada proceso que se inicia.
- **UID, GID** (*User ID, Group ID*): usuario y grupo al que pertenece el proceso. Estas determinan los derechos del proceso a acceder a los recursos del sistema y ficheros.
- **PPID** (*Parent Process ID*): PID del proceso padre.

El contexto de un proceso es su estado, definido por:

- Su código.
- Los valores de sus variables de usuario globales y de sus estructuras de datos.
- El valor de los registros de la CPU.
- Los valores almacenados en su entrada de la tabla de procesos y en su área de usuario.

- Y el contenido de sus pilas (*stacks*) de usuario y *kernel*.

Proceso: comandos

1) Monitorización de procesos

PS

Sintaxis: ps [opciones]

Muestra la lista de procesos del sistema y algunas de sus características: hora de inicio, uso de memoria, estado de ejecución, propietario y otros detalles.

Opciones:

- **a**: muestra los procesos creados por cualquier usuario y asociados a un terminal.
- **l**: formato largo. Muestra prioridad, el PID del proceso padre.
- **u**: formato de usuario. Incluye el usuario propietario del proceso y la hora de inicio.
- **U**: usr: lista los procesos creados por el usuario “usr”.
- **x**: muestra los procesos que no están asociados a ningún terminal del usuario (entre ellos, los “*daemons*”).

PSTREE

Sintaxis: pstree [opciones] [PID] [user]

Este comando muestra la jerarquía de los procesos mediante una estructura en árbol. Si se especifica un PID de proceso, mostrará a partir de este; de lo contrario, comienza por el proceso init (PID=1) y mostrará todos los procesos del sistema. Si se especifica un usuario válido, se mostrará la jerarquía de todos los procesos del usuario.

Opciones:

- **a**: incluye en el árbol de procesos la línea de comandos que se usó para iniciar el proceso.
- **c**: deshabilita la unión de procesos hijos con el mismo nombre (réplicas de un mismo proceso).
- **G**: usa los caracteres de línea para dibujar el árbol. La representación del árbol es más clara.
- **h**: remarca la jerarquía del proceso actual.

- **n**: por defecto, los procesos con mismo padre se ordenan por el nombre. Esta opción fuerza a ordenar los procesos por su PID.
- **p**: incluye el PID de los procesos en el árbol.

TOP

Sintaxis: top [opciones]

El comando TOP ofrece una lista de procesos similar al comando ps, pero la salida se actualiza continuamente. Es especialmente útil cuando es necesario observar el estado de los procesos o comprobar los recursos que consumen.

Opciones:

- **i**: ignora a los procesos inactivos listando únicamente los que utilizan recursos del sistema.
- **d**: especifica el ritmo de actualización de la pantalla en segundos. Es posible especificar decimales.

Órdenes:

- **h**: muestra una pantalla de ayuda.
- **q**: sale de la programación.
- **k**: *kill*, permite detener un proceso.
- **r**: permite alterar la prioridad de un proceso.

HTOP

De funcionamiento similar a TOP, el comando HTOP ofrece algunas ventajas, como la navegación entre procesos, tanto vertical como horizontal, mediante las flechas en el teclado. También, una mejora visual con el añadido de los medidores de consumo de la CPU y una barra inferior con opciones (F1-F10), que permiten, entre otras cosas, filtrar la información o matar un proceso situándose sobre él.

Opciones:

- **s**: ordena por columna.
- **u**: muestra sólo los procesos de un usuario.
- **p**: muestra sólo la información de los PIDs introducidos.

2) Ejecución en 1º y 2º plano

Un proceso ejecutado en 1º plano se ejecuta bloqueando el terminal desde el que se lo lanzó. Se ejecuta simplemente introduciendo su nombre en el terminal y pulsando intro. Con un proceso en 1º plano se pueden realizar dos acciones desde el terminal asociado:

1. Matar al proceso (Ctrl-c): se cancela el proceso y se liberan sus recursos asignados. Esta acción envía una señal de interrupción al proceso, indicándole que tiene que detenerse. Sólo se pueden matar procesos sobre los que se tenga permiso (salvo *root*).
2. Parar el proceso (Ctrl-z): en este caso, sólo se detiene su ejecución, conservando su estado y recursos para poder continuar en el momento que se dé la orden “fg”.

Un proceso ejecutado en 2º plano no bloquea el terminal desde el que se lanzó. Los programas pueden iniciarse en 2º plano añadiendo el caracter *&* al final del comando. Cuando un proceso se ejecuta en segundo plano, se crea un trabajo (*job*), al cual se le asigna un número entero, empezando por 1 y numerado secuencialmente.

Se puede mover un programa ejecutado en primer plano al segundo plano deteniéndolo con Ctrl+z, y después reiniciándolo en 2º plano mediante la orden “bg”.