1. Los procesos y el sistema operativo

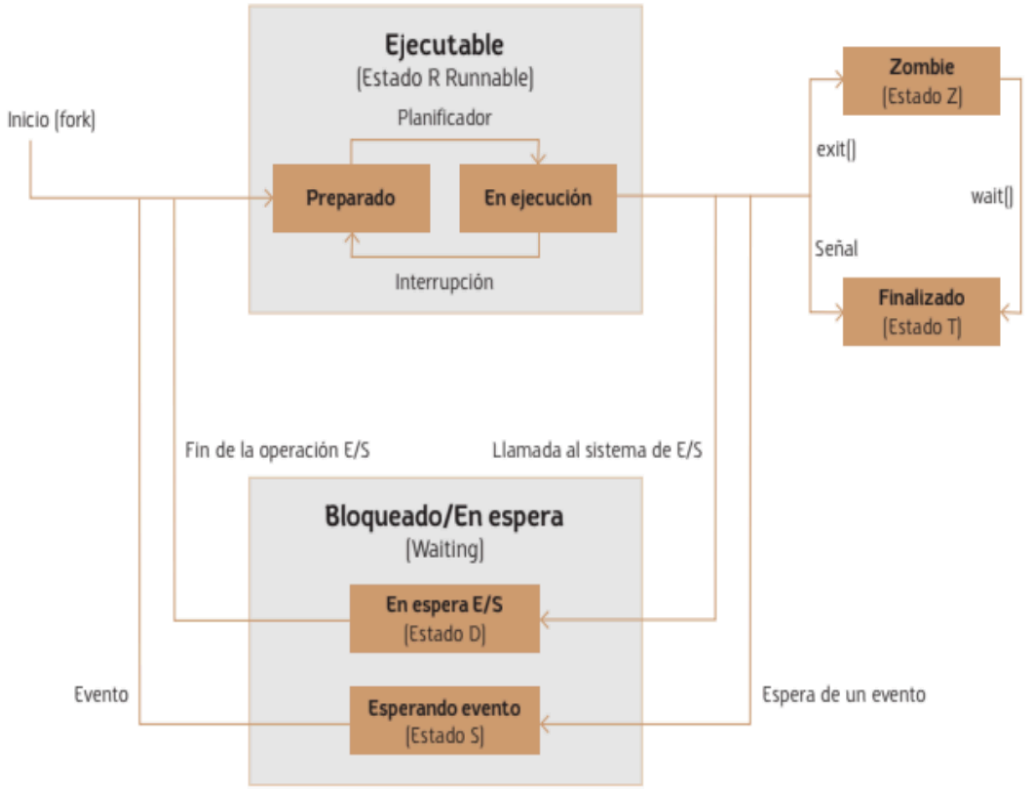
1.1. Procesos y estados de un proceso

El software de un ordenador ya sea de sistema o de aplicación, se compone tanto de programas como de documentación y otros tipos de recursos: imágenes, ficheros de configuración, etc.

Todos estos recursos se encuentran almacenados en el disco duro de nuestros equipos. Centrándonos en las aplicaciones, cuando lanzamos un fichero ejecutable, el sistema operativo carga este en la memoria, creando un proceso al que le asigna un identificador (PID/Process IDentifier) y ciertos recursos que hacen posible su ejecución, como la memoria, el tiempo de CPU y el acceso a ficheros y dispositivos de entrada y salida.

Teniendo esto en mente, podemos definir un proceso como una instancia de un programa en ejecución al que se le han asignado determinados recursos del sistema.

Un proceso, además, posee un estado; dichos estados siguen un modelo sobre el que se definen transiciones, que se representan en forma de grafo. Las transiciones de un estado a otro se producen a consecuencia de llamadas al sistema o como respuesta a ciertos eventos. Veamos este modelo en el siguiente gráfico:



En la figura podemos distinguir tres estados principales:

Ejecutable (estado R-Runnable)
El proceso se encuentra en disposición de ejecutarse, bien en la cola de ejecución (preparado), o bien en ejecución. Cuando un proceso se inicia, mediante la llamada al sistema fork, este pasa a la cola de procesos preparados para su ejecución, y es el planificador del sistema operativo quien decide qué procesos de esta cola se llevan a ejecución. Cuando un proceso que está en ejecución recibe una interrupción, vuelve a la cola de preparados.

Bloqueado o en espera (Waiting)
El proceso ha sido bloqueado, bien porque ha realizado una llamada al sistema de Entrada/Salida y queda a la espera (Estado D) de la finalización de esta operación (por ejemplo, leer un fichero o entrada de usuario), o bien porque requiere de otro tipo de evento, como por ejemplo un temporizador (Estado S Interruptible Sleep). Cuando este evento se produce, o termina la operación de E/S a la que se estaba esperando, el proceso vuelve a la cola de preparados.

Finalizado
El proceso ha finalizado su ejecución y el sistema operativo libera la memoria y los recursos asignados. Esta finalización puede producirse bien porque el proceso recibe una señal de finalización, pasando a estado Finalizado (T), o bien porque el proceso termina la ejecución de todas sus instrucciones. Cuando el proceso finaliza, puede devolver el estado de finalización al proceso que lo creó (proceso padre), mediante la llamada al sistema exit(). En este caso, el proceso pasará a estado Zombie (Z) hasta que el proceso padre recoja esta salida mediante la llamada al sistema wait().Cuando el padre recoja esta salida, el proceso pasa a estado Terminado.

1.2. Utilidades en GNU/Linux para la gestión de procesos

Los sistemas operativos basados en GNU/Linux disponen de varias utilidades para la consulta de los procesos y su estado. Vamos a ver algunas de ellas.

La orden top
Muestra en tiempo real la actividad del procesador, con una lista de los procesos ordenados según el consumo de CPU, permitiendo manipular estos. Top puede clasificar las diferentes tareas según el uso de CPU, de memoria o de tiempo de ejecución. Las características que muestra se pueden configurar bien mediante órdenes interactivas o mediante ficheros de configuración.

Para cada proceso muestra información, como su identificador de proceso (PID), el usuario propietario (USER), su prioridad (PR Y NI), el consumo de memoria virtual (VIRT), el residente (RES), memoria compartida (SHR), el estado del proceso (s), indicando los valores comentados en el diagrama anterior (R, D, S, Z, T...), el porcentaje de CPU utilizado desde su última actualización, el porcentaje de memoria o el tiempo total de CPU utilizado por el proceso, así como el comando que lo lanzó. Además, en la parte superior muestra varios datos de interés, como la hora, el tiempo activo del equipo, la carga promedio, el total de procesos y los diferentes estados, el porcentaje de CPU utilizado por la CPU, así como la memoria total libre utilizada, etc.

CIFP EN ADEJE

```
top - 16:38:34 up 1 day, 10:41,  2 users,  load average: 1,60, 1,36, 1,37
Tasks: 352 total,   1 running, 346 sleeping,   0 stopped,   5 zombie
%Cpu(s):  2,3 us,  1,3 sy,  0,0 ni, 96,4 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :  23838,0 total,    975,4 free,   7566,7 used,  15295,8 buff/cache
MiB Swap:  15317,0 total,  15313,3 free,      3,7 used.  12730,2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  18918 joamuran  20   0 6556392   1,0g 492416 S  10,2   4,5  52:26.23 GeckoMain
  25495 joamuran  20   0 2872032 401964 274380 S   2,6   1,6  14:27.45 Isolated Web Co
  35782 joamuran  20   0 3432612 788656 168356 S   2,6   3,2   7:17.88 Isolated Web Co
  41574 joamuran  20   0 2610520 210628 102876 S   2,6   0,9   0:34.13 Isolated Web Co
  12692 root      20   0 1541072 192276 137052 S   2,0   0,8  34:57.86 Xorg
  44691 joamuran  20   0 3543356 528708 145540 S   1,0   2,2   4:51.81 Isolated Web Co
    891 root      20   0 1391188  35712  18092 S   0,7   0,1   0:07.15 snapd
  26100 joamuran  20   0 2189228 412300 143468 S   0,7   1,7  45:32.22 qemu-system-x86
  26104 joamuran  20   0  206868  32812  23064 S   0,7   0,1   0:14.11 emulator64-cras
  31119 joamuran  20   0   16,5g 245840 160848 S   0,7   1,0   1:12.78 chrome
  48257 joamuran  20   0  869580 276120  78324 S   0,7   1,1   1:40.58 inkscape
  50921 joamuran  20   0 2664208 186732 123084 S   0,7   0,8   0:16.19 Isolated Web Co
  53351 joamuran  20   0   12064   3964   3180 R   0,7   0,0   0:00.96 top
     67 root      25   5       0      0      0 S   0,3   0,0   0:23.21 ksmd
    608 root     -51   0       0      0      0 S   0,3   0,0   0:50.11 irq/130-iwlwifi
    867 root      20   0  231588   6148   5380 D   0,3   0,0   1:05.71 iio-sensor-prox
  13423 joamuran  20   0 1040192 117136  92760 S   0,3   0,5   0:14.20 konsole
  31161 joamuran  20   0   16,3g 100164  83696 S   0,3   0,4   0:19.92 chrome
  42280 joamuran  20   0   11,5g   1,4g 209620 S   0,3   6,0  10:57.43 soffice.bin
```

Salida del comando top

Top es un comando muy completo y que admite muchas opciones. Puedes consultar la página del manual del sistema para esta orden lanzando desde la terminal:

```
$ man top
```

Hay que destacar que top es un comando dinámico y que nos muestra el estado de los procesos en tiempo real. Para salir de la vista del comando pulsaremos la tecla q (quit).
Por otro lado, existen otras herramientas similares, como htop, una versión mejorada de top, o atop, un monitor de procesos en tiempo real. Ninguna de ellas suele venir preinstalada por defecto en el sistema, pero se encuentran disponibles en los repositorios de las distribuciones GNU/Linux para su instalación.

La orden ps (Process Status)

Se trata de una de las herramientas más populares en el mundo GNU/Linux, y muestra una instantánea de los procesos actuales. Se trata de un comando muy completo, que admite un gran abanico de opciones, tanto de tipo Unix (precedidas de un guion -) como BSD (sin ningún guion) o GNU (precedidas de dos guiones --).
Sin ningún parámetro muestra los procesos del usuario actual en la terminal actual, junto a su PID, la terminal donde se encuentra el tiempo de proceso o el comando que lo ejecutó:

```
administrador@taucho-00:~$ ps
    PID TTY          TIME CMD
   7191 pts/0    00:00:00 bash
   7245 pts/0    00:00:00 ps
administrador@taucho-00:~$
```

Una de las formas más habituales de utilizar este comando es con los argumentos estilo Unix -aux (o aux con sintaxis BSD), que mostrará información más detallada sobre todos los procesos del sistema.

En este punto también es interesante conocer el comando pstree, que nos muestra el árbol de procesos del sistema, o lo que es lo mismo, la jerarquía de procesos de este.

1.3. Threads

Los threads, procesos ligeros o hilos de ejecución, son procesos que forman parte de un mismo proceso y comparten determinados recursos, como el espacio de direcciones de memoria, variables, ficheros, señales, etcétera. Su uso es muy común y conlleva una serie de ventajas sobre los procesos como tal, entre las que podemos destacar un coste computacional considerablemente menor a la hora de crear y destruir hilos respecto a los procesos, un menor consumo de recursos o la posibilidad de dividir tareas entre múltiples hilos, aprovechando así mejor las arquitecturas multiprocesador.
Mediante el comando ps podemos obtener también información sobre los hilos de ejecución de un proceso, por ejemplo, utilizando la sintaxis tipo UNIX ps -eLf o la sintaxis BSD ps axms. Estos parámetros nos ofrecerán nuevos datos como LWP (LightWeight Process), con el identificador de thread, o NLWP (Number of LighWeight Processes), con el número de hilos del proceso.

Veamos un ejemplo, restringido al proceso NetworkManager:

```
UID   PID PPID LWP  C  NLWP STIME   TTY  TIME      CMD
root 862  1    862  0    3  Dec21 ?   00:00:09 /usr/sbin/NetworkManager
root 862  1    925  0    3  Dec21 ?   00:00:00 /usr/sbin/NetworkManager
root 862  1    935  0    3  Dec21 ?   00:00:03 /usr/sbin/NetworkManager
```

Como vemos, este proceso (PID 862) tiene 3 hilos de ejecución (NLWP), con los identificadores 862, 925 y 935.

1.4. Demonios y servicios. Systemd

Los demonios (daemon) son procesos que se ejecutan en segundo plano, sin terminal asociada ni interfaz gráfica, esperando que se produzca algún evento en el sistema o la recepción de una petición de servicio. Un servidor web, por ejemplo, se ejecutaría de esta forma para recibir peticiones de documentos web y devolver los documentos. También existen demonios que no son tan visibles como aquellos que ofrecen servicios, y que realizan tareas como escribir en el registro o mantener la precisión del reloj del sistema.