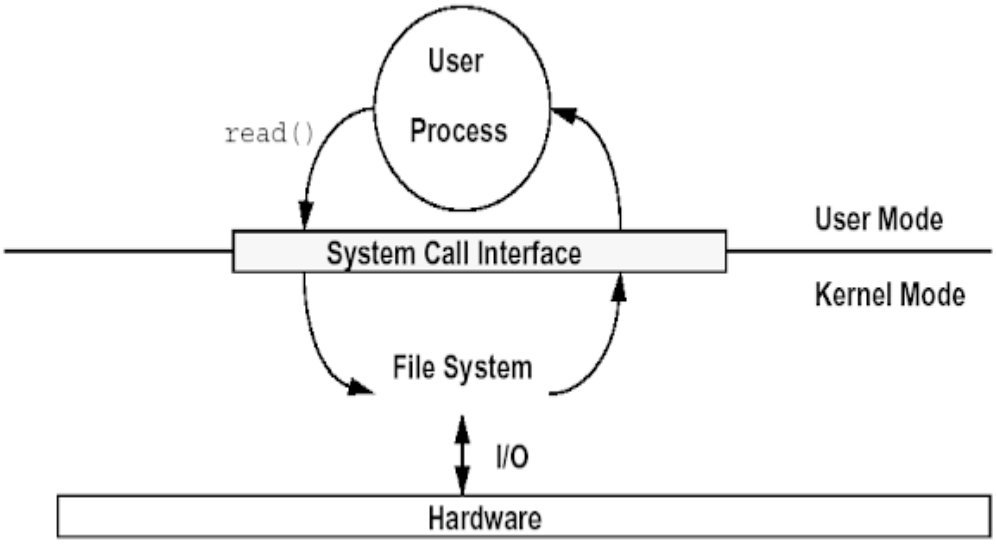
Llamadas al sistema en Linux

GNU/Linux es un Sistema Operativo multitarea en el que van a convivir un gran número de procesos. Es posible, bien por un fallo de programación o bien por un intento malicioso, que alguno de esos procesos haga cosas que atenten contra la estabilidad de todo el sistema. Por ello, con vistas a proteger esa estabilidad, el núcleo o kernel del sistema funciona en un entorno totalmente diferente al resto de programas. Se definen entonces dos modos de ejecución totalmente separados: el **modo kernel** y el **modo usuario**.

Cada uno de estos modos de ejecución dispone de memoria y procedimientos diferentes, por lo que un programa de usuario no podrá ser capaz de dañar al núcleo.

Las syscalls o llamadas al sistema son el mecanismo por el cual los procesos y aplicaciones de usuario acceden a los servicios del núcleo. Son la interfaz que proporciona el núcleo para realizar desde el modo usuario las cosas que son propias del modo kernel (como acceder a disco o utilizar una tarjeta de sonido). La siguiente figura explica de forma gráfica cómo funciona la syscall read().

Mecanismo de petición de servicios al kernel



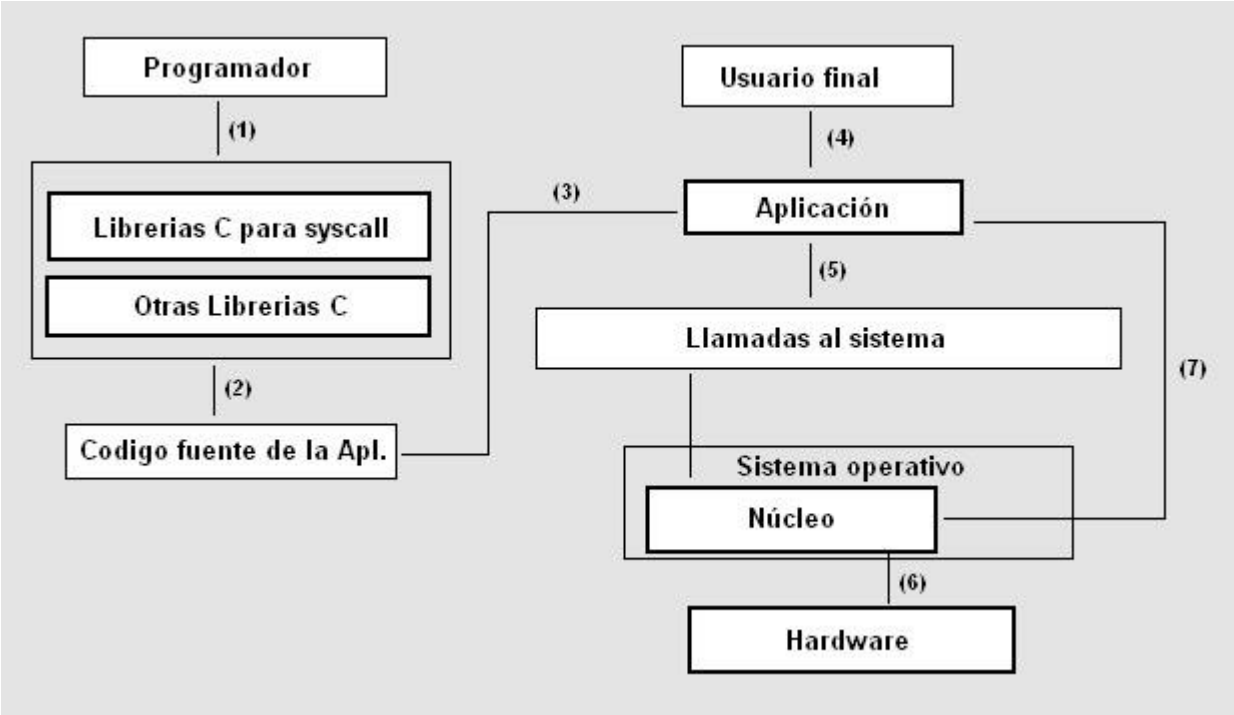
El **modo usuario** necesita acceder al disco para leer, para ello utiliza la syscall read() utilizando la interfaz de llamadas al sistema. El núcleo atiende la petición accediendo al hardware y devolviendo el resultado al proceso que inició la petición. Este procedimiento me recuerda al comedor de un restaurante, en él todos los clientes piden al camarero lo que desean, pero nunca entran en la cocina. El camarero, después de pasar por la cocina, traerá el plato que cada cliente haya pedido. Ningún comensal podría estropear la cocina, puesto que no tiene acceso a ella.

Prácticamente todas las funciones que utilicemos desde el espacio de ejecución de usuario necesitarán solicitar una petición al kernel mediante una syscall, esto es, la ejecución de las aplicaciones de usuario se canaliza a través del sistema de peticiones al sistema. Este hecho es importante a la hora de fijar controles y registros en el sistema, ya que si utilizamos nuestras propias versiones de las syscalls para ello, estaremos abarcando todas las aplicaciones y procesos del espacio de ejecución de usuario. Imaginemos un "camarero" malicioso que capturase todas las peticiones de todos los clientes y envenenase todos los platos antes de servirlos... nuestro restaurante debería cuidarse muy bien de qué personal contrata y nosotros deberemos ser cautelosos también a la hora de cargar drivers o módulos en nuestro núcleo.

Para hacer uso de las llamadas al sistema desde el lenguaje de programación C, los sistemas operativos que trabajan con el núcleo Linux ponen a disposición del usuario varias funciones o procedimientos de librería que representan a las llamadas del sistema. Los prototipos relativos a estas funciones o procedimientos pueden encontrarse listados en el archivo de cabecera **unistd.h** (este se encuentra en el directorio **/usr/include/asm/**, aquí también pueden encontrarse los archivos **unistd_32.h** y **unistd_64.h**, archivos relativos a las arquitecturas de 32 y 64 bits respectivamente).

El sistema operativo de núcleo Linux cuenta con aproximadamente 200 funciones relacionadas con cada llamada al sistema, algunos de los cuales pueden agruparse en ciertas categorías que permiten el manejo o control de: procesos, señales, archivos, tiempo, etc.

Los conceptos explicados anteriormente pueden resumirse con el siguiente esquema:



El programador (1): usa las librerías C para llamadas al sistema y otras librerías (2): para implementar el código fuente de la aplicación (3): y a partir de él generar el archivo ejecutable de dicha aplicación.

El usuario final (4): ejecuta la aplicación, (5): la cual se comunica a través de llamadas al sistema con el núcleo del sistema operativo (6): el cual toma el control y se encarga de manipular el hardware para realizar el proceso solicitado.

Cuando el proceso se ha completado, el núcleo (7): retorna el control a la aplicación.

Llamadas al sistema para el control de procesos

Este tipo de llamadas al sistema permiten realizar "actividades" relacionadas con los programas que están en espera para ejecución o que se están ejecutando, es decir, cuando son un proceso; cada sistema operativo tiene sus formas de invocar dichas llamadas, en el caso de los sistemas operativos bajo el núcleo Linux estas se pueden invocar desde el Shell o interprete de comandos y desde el lenguaje C.

Las llamadas más comunes de este tipo son:

1. fork( )
2. getpid( )
3. getppid( )
4. La familia de llamadas exec…(…)
5. wait( )
6. waitpid(…)
7. exit(…)
8. system(…)

Llamadas al sistema para el control de señales

Las llamadas a sistema para el control de señales, son un tipo de llamada de sistema que permite a un proceso establecer comunicación con otros procesos que se están ejecutando en la máquina o para que el kernel se comunique con ellos, como puede ser, informar sobre alguna condición, una solicitud de espera, etc.

En tal comunicación el proceso puede ser emisor o receptor de la señal. Las señales que se pueden producir dependen de la máquina y de los privilegios que el proceso tenga.

Para la mayoría de aplicaciones grandes, se usan señales como forma de controlar las tareas, pudiendo así tener aplicaciones en segundo plano o corriendo provisionalmente.

En el caso de los sistemas operativos con núcleo Linux el listado de señales se puede obtener con el comando kill y la opción -l, para la maquina usada en los ejemplos, el listado de señales con su respectivo número es:

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Las señales no tienen ninguna prioridad u orden para ser atendidas, es posible ignorarlas (excepto SIGSTOP y SIGKILL) y si en algún momento llegan simultáneamente varias señales del mismo tipo, solo se tendrá en cuenta una y las demás se considerarán como si nunca hubieran llegado.