Micronúcleos

Mariano Street
mstreet@fceia.unr.edu.ar

Sistemas Operativos 2 – 2020
Departamento de Ciencias de la Computación
FCEIA – UNR

1. Núcleo de un sistema operativo

El núcleo (en inglés, *kernel*) es la pieza central de un sistema operativo. Es el software a cargo de administrar el hardware, proveyendo a los programas acceso al mismo de forma compartida y segura. Se encarga de multiplexar el hardware, es decir que permite a múltiples procesos acceder a un mismo dispositivo como si en realidad cada uno tuviera el suyo.

1.1. Proceso de arranque

El núcleo se destaca además en el proceso de arranque de un sistema operativo. Desde que se enciende la computadora, se pueden distinguir tres componentes esenciales que se ejecutan en forma secuencial.

1. Primero, el firmware que viene incluido con el hardware en una memoria ROM y se encarga de tareas de inicialización de hardware del más bajo nivel (ejemplos de firmware: BIOS, EFI, UEFI y Coreboot).

2. Segundo, el gestor de arranque, permite seleccionar qué sistema operativo arrancar; puede estar instalado en un disco, venir incluido con el firmware o darse ambos casos a la vez (ejemplos: GRUB, LILO, Syslinux).

3. Tercero, el núcleo. Es en este punto cuando se considera que empieza a ejecutarse el sistema operativo propiamente dicho.

Una vez cargado el núcleo, este puede encargarse de ejecutar procesos. Los detalles de cómo se haga esto dependerán de la estructura del núcleo en sí y de sus particularidades.

2. Núcleo y programas de usuario

Los sistemas operativos modernos suelen estar compuestos, por un lado, del núcleo, y por otro, de programas de usuario. El primero se encarga de ejecutar los segundos, en instancias que se llaman procesos. ¿Por qué se hace esta distinción? ¿No podría ser el núcleo un programa más y dejar que cualquier otro administre también el hardware?

Se podría hacer, pero tornaría al sistema operativo inseguro e inestable. Las arquitecturas de CPU por lo general han ido incorporando protecciones en este sentido, de forma que garantizan la separación a nivel de hardware. El concepto fundamental consiste en definir dos modos o espacios de ejecución separados:

Modo/Espacio de núcleo o de supervisor
: Privilegiado. Lo que se ejecuta en este modo tiene acceso libre a todo el hardware. Acá se ubica el núcleo.

Modo/Espacio de usuario
: Restringido. Lo que se ejecuta en este modo no tiene acceso al hardware, necesita pedir a software del espacio de núcleo que actúe de intermediario. Acá se ubican los programas de usuario.

Se pueden encontrar arquitecturas con más de dos niveles, ya sea para brindar más granularidad (privilegios parciales) o para implementar otras funcionalidades. Para el tema en cuestión, no vienen al caso.

3. Estructura del núcleo

En el enfoque convencional, el núcleo es un solo programa. Todos sus componentes comparten el mismo espacio de memoria. En las primeras décadas de la computación se trataba de programas chicos y no se tenía problema. En la década de 1980 ya pasaron a crecer mucho: se incorporaron controladores de dispositivos, protocolos de Internet, sistemas de archivos, entre otras funcionalidades. Esto generó problemas de mantenibilidad, estabilidad y seguridad: un problema en cualquiera de esas funcionalidades podría dejar todas las otras inoperativas.

3.1. Micronúcleo

Surge entonces el cuestionamiento: ¿realmente los núcleos tienen que incluir toda esa funcionalidad?, ¿qué cosas tienen que formar parte del núcleo? En otras palabras, ¿podrían algunas de estas cosas moverse al espacio de usuario en cambio?

La idea de los micronúcleos (en inglés, *microkernels*) es, en principio, que todo lo posible vaya a espacio de usuario, que se usen abstracciones de hardware simples y que el nûcleo sea mínimo. El núcleo entonces deja de ser una caja negra con una serie de políticas ya implementadas, y pasa en cambio a servir de infraestructura para construir un sistema en base a diversos componentes; implementa mecanismos en vez de políticas. En espacio de usuario, por su parte, se crean nuevos componentes que se encargan de definir las políticas propiamente dichas e implementan las funcionalidades que el núcleo deja de proveer.

¿Hasta qué punto se pueden quitar cosas del núcleo? Mínimamente, se necesita que este tenga:

1. Alguna gestión de procesos.
2. Manejo básico de espacios de direcciones.
3. Comunicación entre procesos (IPC). Generalmente se hace con pasaje de mensajes.

3.2. Otras estructuras

Hay diversas posibles estructuras que puede seguir un núcleo. La de micronúcleos es una y es el tema de este apunte. A continuación se presentan otras dos estructuras básicas:

Núcleo monolítico
: Es la convencional que se mencionaba al principio de la sección. Separa espacio de núcleo y espacio de usuario, protege la memoria entre distintos procesos y entre cada proceso y el núcleo, pero incorpora dentro de sí todas las funciones comunes de un sistema. Así son por lo general los núcleos de la familia Unix, incluido Linux.

Núcleo sin protección
: No hace distinción entre espacio de núcleo y espacio de usuario, no existe protección de memoria y cualquier proceso puede acceder directamente al hardware. Algunas CPU antiguas no ofrecían separación y por tanto el software no podía ser de otra forma; es el caso de los sistemas operativos DOS. Hoy en día se pueden encontrar sistemas embebidos que siguen este enfoque para simplicidad y rendimiento.

4. Breve historia de micronúcleos

Esta sección no pretende ahondar en la historia de los micronúcleos, pero sí cabe destacar las grandes tendencias.

Hubo dos grandes generaciones. La primera, en la década de 1980, cuando empezó a hacer furor la investigación en esta forma de estructurar sistemas. El caso insignia de esta generación es Mach, basado en BSD; el núcleo de BSD es monolítico, Mach consistía en ir retirándole componentes para transicionar paulatinamente hacia un micronúcleo. Mach es la base de GNU Hurd (proyecto de núcleo de GNU, al final no prosperó y su lugar fue ocupado por Linux) y también de XNU (núcleo del macOS e iOS actuales).

Cuando dejó de ser tan monolítico y adoptó más a fondo el diseño de micronúcleo, Mach pasó a tener un rendimiento notoriamente malo. Esto se tornó muy difícil de solventar y terminó minando el furor por estos diseños. Se empezó a considerar que los micronúcleos eran una idea elegante en la teoría pero que no podía competir con los monolíticos en la práctica. Sumado al estancamiento de Hurd y otros fracasos, en la década de 1990 la popularidad de los micronúcleos se derrumbó.

Más tarde en esa misma década, se desarrollaron nuevos diseños de micronúcleos desde cero y se dio paso a una segunda generación. El caso emblemático es L4. Con un diseño pensado minuciosamente, adaptado a una arquitectura específica y pensado desde cero para ser un micronúcleo (en vez de derivar de un monolítico), logró un rendimiento mucho mejor, equiparable al de los monolíticos. Resurgió hasta cierto punto el interés por esta forma de diseñar sistemas, con muchos derivados de L4 para usos específicos.

Un caso aparte a destacar es el de MINIX. Pertenece a la primera generación pero su enfoque no está en la eficiencia, sino en la enseñanza de sistemas operativos (en sus primeras versiones) y en los sistemas embebidos (en las nuevas). Es un proyecto activo al día de hoy. Es el precursor de Nachos.

5. Ventajas y desventajas

Ventajas de un micronúcleo:

- Es más robusto.

  Si hay un error en un componente del sistema, solo ese componente es afectado. Se lo puede reiniciar sin afectar a los demás.

  Se facilita también la tarea de actualizar partes del sistema mientras está funcionando.

- Es más fácil de depurar y mantener. También puede ser más simple de entender.

- Resulta fácil de extender, muchas veces basta con agregar un proceso de usuario.

- Se pueden apilar más sistemas operativos sobre el micronúcleo, siempre y cuando sea todo espacio de usuario.

  Presenta algunas cercanías con los conceptos de virtualización.

- Se presta al armado de sistemas distribuidos.

Desventajas de un micronúcleo:

- Darle un rendimiento aceptable conlleva más esfuerzo que para un monolítico.

- Requiere pensar la arquitectura del software de antemano, es decir, qué clases de componentes va a haber y cómo van a interactuar entre sí.

- No es algo inherente al concepto en sí, pero al requerir diseños dedicados, se vuelve más complicado aprovechar la base de software de sistemas populares.

6. Enfoques híbridos

Los micronúcleos no se volvieron populares como en una época se esperaba. Sin embargo, hay diversas enfoques híbridos ya sea de núcleos enteros o de ciertas características que son muy relevantes en la industria a día de hoy. Con híbrido se hace referencia a soluciones intermedias entre un micronúcleo y un núcleo monolítico.

- FUSE (Filesystem in Userspace): para Linux y otros sistemas tipo Unix, permite implementar sistemas de archivo nuevos como procesos de usuario.

- Módulos cargables y descargables en Linux y otros sistemas (véanse los comandos `lsmod` y `modprobe`, entre otros).

- NT, el núcleo de Windows NT y sus sucesores (es decir, todos los Windows modernos), es un híbrido.

- XNU, el núcleo de Darwin (base de macOS, iOS y los demás sistemas modernos de Apple), es un híbrido también.