2.5 ESBOZO DE LOS PROCESOS EN MINIX

Habiendo completado nuestro estudio de los principios de la gestión de procesos, comunicación entre procesos y planificación, vamos a echar una mirada a como se aplican en MINIX. De forma diferente que en UNIX, cuyo núcleo es un programa monolítico que no está descompuesto en módulos, MINIX es él mismo una colección de procesos que se comunican entre si y con los procesos de usuario utilizando una única primitiva de comunicación entre procesos –paso de mensajes. Este diseño proporciona una estructura más modular y flexible, haciendo fácil, por ejemplo, sustituir el sistema de ficheros entero por otro completamente diferente, sin tener ni que recompilar el núcleo.

2.5.1 La Estructura Interna de MINIX

Vamos a comenzar nuestro estudio de MINIX inspeccionando a vista de pájaro el sistema. MINIX está estructurado en cuatro capas, con cada capa realizando una función bien definida. Las cuatro capas se ilustran en la Figura 2-26.

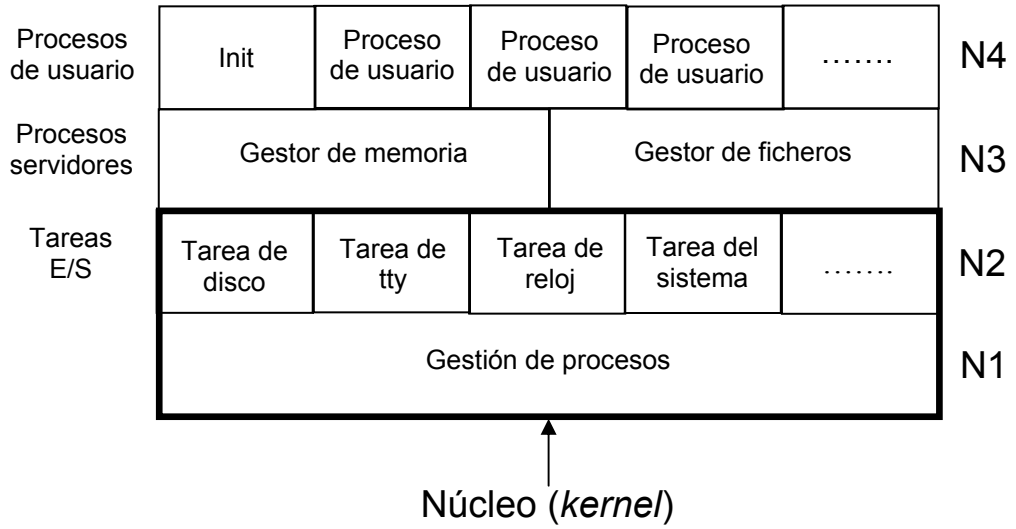

Figura 2-26. MINIX está estructurado en cuatro capas.

La capa del fondo captura todas las interrupciones y traps, realiza la planificación, y proporciona a las capas superiores un modelo de procesos secuenciales independientes que se comunican mediante mensajes. El código de esta capa tiene dos funciones principales. La primera es capturar los traps y las interrupciones, salvar y restaurar los registros, planificar, y hacer todo lo demás que se necesita para conseguir que funcione realmente la abstracción de los procesos que se proporciona a las capas superiores. La segunda es manejar la mecánica de los mensajes; comprobando la legalidad de sus destinatarios, localizando los búferes de las operaciones de enviar y recibir en la memoria, y copiando los bytes desde el emisor al receptor. Esa parte de la capa que trata el nivel inferior del manejo de las interrupciones está escrito en lenguaje ensamblador. El resto de la capa y todas las capas superiores, están escritas en C.

La capa 2 contiene los procesos de E/S, uno por tipo de dispositivo. Para distinguirlos de los procesos de usuario ordinarios, vamos a llamarlos **tareas**, pero las diferencias entre las tareas y los procesos son mínimas. En numerosos sistemas las tareas de E/S se denominan **drivers de dispositivos**; vamos a utilizar los términos “tarea” y “driver de dispositivo” de forma intercambiable. Es necesaria una tarea por cada tipo de dispositivo, incluyendo discos, impresoras, terminales, interfaces de red y relojes. Si hay otros dispositivos presentes, es necesaria una nueva tarea para cada uno de ellos. Una tarea, la tarea del sistema, es un poco diferente, ya que no corresponde a ningún dispositivo de E/S. Vamos a analizar las tareas en el siguiente capítulo.

Todas las tareas de la capa 2 y todo el código en la capa 1 están enlazados juntos en un único programa binario denominado el **núcleo**. Algunas de las tareas comparten subrutinas comunes, pero de otra manera son independientes unas de otras, se planifican independientemente, y se comunican utilizando mensajes. Los procesadores de Intel comenzando con el 286 asignan uno de sus cuatro niveles de privilegio a cada proceso. Aunque las tareas y el núcleo están compiladas juntas, cuando el núcleo y las rutinas de tratamiento de las interrupciones están ejecutándose tienen más privilegios que las tareas. Entonces el verdadero código del núcleo puede acceder a cualquier parte de la memoria y a cualquier registro del procesador – esencialmente, el núcleo puede ejecutar cualquier instrucción utilizando datos de cualquier sitio dentro del sistema. Las tareas no pueden ejecutar todas las instrucciones a nivel de máquina, ni pueden acceder a todos los registros de la CPU o a todas las partes de la memoria. Sin embargo, ellas pueden acceder a regiones de la memoria pertenecientes a procesos menos privilegiados, en orden a realizar E/S para ellos. Una tarea, la tarea del sistema, no realiza E/S en el sentido normal pero existe en orden a proporcionar servicios, tales como la copia de datos entre diferentes regiones de la memoria, para procesos a los que no se les permite realizar esas cosas por si mismos. En máquinas que no proporcionan diferentes niveles de privilegio, tales como los procesadores más antiguos de Intel, estas restricciones no pueden ser reforzadas por el hardware.

La capa 3 contiene procesos que proporcionan servicios útiles a los procesos de usuario. Estos procesos servidores se ejecutan en un nivel menos privilegiado que el núcleo y las tareas y no pueden acceder a los puertos de E/S directamente. Estos procesos tampoco pueden acceder a la memoria fuera de los segmentos que se les han asignado. El **gestor de memoria** (MM) lleva a cabo todas las llamadas al sistema de MINIX que involucran gestión de memoria, tales como FORK, EXEC y BRK. El **sistema de ficheros** (FS) lleva a cabo todas las llamadas al sistema de ficheros, tales como READ, MOUNT y CHDIR.

Como hemos señalado al comienzo del Capítulo 1, los sistemas operativos hacen dos cosas: gestionan los recursos y proporcionan una máquina extendida mediante la implementación de las llamadas al sistema. En MINIX la gestión de los recursos está situada principalmente en el núcleo (capas 1 y 2), y la interpretación de las llamadas al sistema está en la capa 3. El sistema de ficheros se ha diseñado como un "servidor" de ficheros y puede moverse a una máquina remota casi sin cambios. Esto también es válido para el gestor de memoria, aunque los servidores de memoria remotos no son tan útiles como los servidores de ficheros remotos.

En la capa 3 pueden existir servidores adicionales. Aunque MINIX tal y como se describe en este libro no incluye el servidor de red, su código fuente es parte de la distribución estándar de MINIX. El sistema puede recompilarse fácilmente para incluirlo.

Este es un buen lugar para señalar que aunque los servidores son procesos independientes, difieren de los procesos de usuario en que arrancan cuando el sistema arranca, y nunca terminan mientras el sistema esté activo. Adicionalmente, aunque ellos se ejecutan en el mismo nivel de privilegio que los procesos de usuario en términos de las instrucciones máquina que se les permite ejecutar, reciben una prioridad de ejecución más alta que los procesos de usuario. Para acomodar un nuevo servidor es necesario recompilar el núcleo. El código de arranque del núcleo instala los servidores en entradas privilegiadas de la tabla de procesos antes de que se haya permitido la ejecución de ningún proceso de usuario.

Finalmente, la capa 4 contiene todos los procesos de usuario – shells, editores, compiladores y programas *a.out* escritos por el usuario. Un sistema en ejecución tiene usualmente algunos procesos que comienzan su ejecución cuando el sistema arranca y que nunca terminan de ejecutarse. Por ejemplo, un **demonio** es un proceso de fondo que se ejecuta periódicamente o que siempre espera a que tenga lugar un suceso, tal como la llegada de un paquete por la red. En cierto sentido un demonio es un servidor que se arranca independientemente y se ejecuta como un proceso de usuario. Sin embargo, de forma diferente a los verdaderos servidores instalados en entradas privilegiadas de la tabla de procesos, tales programas no pueden recibir un tratamiento especial por parte del núcleo como el que reciben los procesos servidores de memoria y de ficheros.