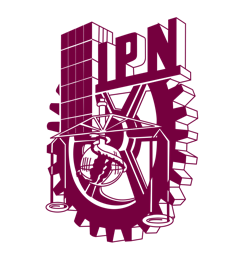
|  |  |
| --- | --- |
| Alumno: | García González Aarón Antonio |
| Grupo: | 2CM9 |
| Unidad de Aprendizaje: | Sistemas Operativos |
| Profesor: | Ana Belem Juárez Méndez |
| Tarea 2: | Investigar una estructura de sistema operativo |
| Fecha: | Viernes 23 de Agosto de 2019 |

**Microkernel**

De acuerdo con la lo visto en el aula de clases, se vieron las estructuras de sistema operativo: monolíticos, por capas, cliente-servidor y virtual, yo elegí la estructura Microkernel, que por lo que leí, es un poco similar a la estructura por capas pero con ciertas peculiaridades de segmentación, lo que busca esencialmente esta estructura es poner lo menos que sea posible en el modo kernel, esto es porque fallas en el kernel pueden paralizar el sistema de inmediato, comparando con los procesos de usuario, pues se pueden configurar para que tengan menos poder, por lo que de llegar a fallar, muy probablemente su error no seria tan fatal.

Se ha estudiado el numero de errores por cada 1000 líneas de código en los sistemas, y se ha llegado a la conclusión de que dicho numero depende del tamaño del módulo, de su tiempo de vida y mas. ****Por lo que es probable que en un sistema monolítico de cinco millones de líneas contenga alrededor de 50,000 errores en el kernel, obvio que no todos serán fatales, pero no es nada sencillo encargarse de ellos y de basto numero de errores que se pueden presentar.

La idea básica de diseño del microkernel, es lograr una alta confiabilidad al dividir el sistema operativo en módulos mas pequeños y bien definidos, sólo uno de los cuales (el microkernel) se ejecuta en modo kernel y el resto se ejecuta como procesos de usuario ordinarios, sin poder relativamente. Por ejemplo, en un sistema operativo donde los drivers se encuentran en el kernel (estructura monolítica), un driver de audio con errores podría hacer incluso que todo el sistema llegara a un alto/paro, lo contrario a lo que la estructura de microkernel sugiere, de haber un error en el driver del dispositivo de audio, a lo mas el audio será confuso o se detendrá este mismo, pero no mas.

Desde finales del siglo pasado se han implementado una variedad de microkernel, la aplicación que se le ha dado a estos últimos es realmente especial; aplicaciones en tiempo real, industrial, aeronáuticas y militares que son de misión critica y tienen requerimientos de confiabilidad muy altos.

Un sistema operativo representativo de esta estructura es MINIX 3, que ha llevado la idea de la modularidad hasta el límite, dividiendo la mayor parte del sistema operativo en varios procesos independientes en modo usuario. MINIX 3 es un sistema de código fuente abierto en conformidad con POSIX.

El microkernel MINIX 3 sólo tiene cerca de 3200 líneas de C y 800 líneas de ensamblador para las funciones de muy bajo nivel, como las que se usan para atrapar interrupciones y conmutar proceso. El código de C administra y planifica los procesos, se encarga de la comunicación entre procesos (al pasar mensajes entre procesos) y ofrece un conjunto de aproximadamente 35 llamadas al kernel para permitir que el resto del sistema operativo realice su trabajo.

El sistema tiene muchas restricciones que limitan el poder de cada proceso. Los drivers sólo pueden utilizar los puertos de E/S autorizados, pero el acceso a las llamadas al kernel también está controlado dependiendo del proceso, al igual que la habilidad de enviar mensajes a otros procesos. Además, los procesos pueden otorgar un permiso limitado a otros procesos para hacer que el kernel acceda a sus espacios de direcciones.



El resultado de todas estas restricciones (divisiones), es que cada driver y servidor tiene el poder exacto para realizar su trabajo y no más, con lo cual se limita en forma considerable el daño que puede ocasionar un componente defectuoso.

Una idea que está en parte relacionada con tener un kernel mínimo es colocar el mecanismo para hacer algo en el kernel, pero no la directiva.