Universidad de Costa Rica

Escuela de Computación e Informática

Trabajo sobre Sistema Operativo Mach

Sistemas Operativos I

Profesor: Diego Villalba

Alumno: Daniel Rivera Solano A85274

13-noviembre-2013

Historia y Aspectos Generales

Mach es un sistema operativo que fue desarrollado en la universidad de Carnegie Mellon en Pittsburg Pennsilvania, Estados Unidos. Usualmente se asocia Mach y sus derivados solo como microkernels pero es importante mencionar que no todos tienen estas características, ya que por ejemplo el Darwin(Sistema Operativo que derivo en el Mac OS de Apple) no es un microkernel. Por otro lado el GNU Hurd, otro sistema operativo derivado del Mach, si es un microkernel. Importante mencionar que un microkernel es una clase de implementación de kernel en donde se busca "minimizar las partes obligatorias de un sistema operativo por medio de un reducido número de mecanismos que maximizan la flexibilidad de implementación, al mismo tiempo que se deja abierta la posibilidad de implementar eficientemente el resto del sistema"[1]. Esta idea fue primeramente explorada en 1970 por Per Brinch-Hansen en su artículo, "The Nucleus of a Multiprogramming System". Ejemplos de la funcionalidad que debería proveer estos mecanismos son el manejo de espacio de direcciones en bajo nivel, manejo de hilos y comunicación entre procesos.

A diferencia de UNIX, Mach desde sus inicios fue diseñado para trabajar en un ambiente de multiprocesos. Está diseñado para funcionar en sistemas de un solo procesador a muchos procesadores. Además es altamente portable a una gran variedad de arquitecturas, además de ser completamente compatible con UNIX 4.3BSD(Distribución de software de Berkeley)[2], la razón de esto por ser Mach desarrollado a partir del kernel UNIX BSD. Esto forma parte de los objetivos de los desarrolladores que quisieron desarrollar un sistema operativo que además de la compatibilidad con UNIX tuviera fortalezas en otras áreas como por ejemplo el soporte de diversas arquitecturas. Este aspecto a su vez viene acompañado de un sistema que tenga un soporte de sistema heterogéneo, para poderlo a disponibilidad de una gran variedad de computadoras y vendedores. Se buscó desarrollar una estructura de kernel simple con abstracciones pero que fueran bastante generales.

1. https://www.gnu.org/software/hurd/microkernel.html
2.Silberchartz,Galvin,Gagne: Operating System Concepts , 9th edition,Apendice B Pag 1.

La generalidad de estas es lo que facilita la implementación de otros sistemas operativos a partir de él, como el Darwin y el Hurd ya mencionados. Esto debido a que al tener pocas pero generales abstracciones es la posibilidad de agregarle funcionalidad al sistema por medio de código a nivel de usuario. Por ello es muy usado también en el área de investigación de sistemas operativos.

Varios aspectos que los diseñadores de MACH buscaron mejorar desde su diseño con respecto al derivado BSD de UNIX fueron el kernel, que en el BSD poseía muchas funcionalidades redundantes lo que lo hacía difícil de manejar y modificar. También se buscó cambiar el aspecto de las abstracciones fundamentales, ya que existían muchas que proveían muchas maneras similares de completar las mismas tareas[3].

La última versión del kernel Mach es la 3.0 lanzada hace ya bastante tiempo, en 1994. A pesar de tener pronto va a tener dos décadas de antigüedad, sigue estando vigente con proyectos como el de GNU Hurd. Ellos catalogan el kernel de Mach como "software que es complemente funcional, no es un proyecto de investigación ni una propuesta, este ha sido usado en muchos sistemas operativos en el pasado, usualmente como la base para un servidor UNIX"[4].

Funcionamiento Básico

Como se mencionó anteriormente, Mach está programado de modo que el kernel de Mach pocos programas dentro del mismo, pero que sean suficientemente funcionales como para que se le pueda agregar funcionalidad sin tener que cambiar el funcionamiento de kernel, ósea a nivel de usuario. Se puede por ejemplo implementar paginadores de manera externa, y estos puede ser llamados por el kernel para su uso[5].

3.Silberchartz,Galvin,Gagne: Operating System Concepts , 9th edition,Apendice B Pags 2,3.

4. http://www.gnu.org/software/hurd/microkernel/mach/gnumach.html

5.Silberchartz,Galvin,Gagne: Operating System Concepts , 9th edition,Apendice B

Particularmente la funcionalidad de kernel está enfocada en la comunicación. Todas las llamadas al sistema y movimiento de datos entre procesos son manejadas por un solo mecanismo de mensajes. De esta manera es como este Sistema Operativo provee la protección a los usuarios, mediante la protección de este único canal de comunicación.

Las abstracciones primitivas de Mach se describen a continuación[6]:

Tareas: un objeto que contiene recursos del sistema, contiene una espacio de dirección tanto física como protegida a los recursos del sistema mediante puertos y contiene uno o más hilos de ejecución. No existe la noción de proceso en Mach, un proceso es implementado mediante una tarea con un solo hilo de control[7].

Hilo: la unidad básica de ejecución, corre en contexto con una tarea que provee el espacio de direcciones, todos los hilos comparten los recursos de la tarea en la que corren.

Puerto: es una cola protegida de mensajes para la comunicación entre tareas. Los puertos son protegidos por el kernel mediante los "derechos de puerto". Solo una tarea con el derecho de puerto adecuado puede enviar un mensaje a determinado puerto. El programador puede invocar alguna funcionalidad de algún objeto enviando un mensaje al puerto que está asociado con ese determinado objeto.

Mensaje: método básico de comunicación entre tareas, es una colección de datos. Los derechos de puerto son pasados por mensajes. Solo puede ser enviado a puertos, no a hilos o tareas.

Objeto de Memoria: vista como una porción de memoria, para que las tareas accedan a ella, simplemente mapean los partes o todo el objeto a su espacio de direcciones. Existe la posibilidad de que los objetos sean manipulables por un manejador de memoria externo en modo usuario.

6.Tevanian, Avadis; Rashid, Richard F.; Golub, David B.; Black, David L.; Cooper, Eric; Young, Michael W. (1987). "Mach Threads and the Unix Kernel: The Battle for Control". Proceedings of the USENIX Summer Conference, USENIX Association. pp. 185–197

7.Silberchartz,Galvin,Gagne: Operating System Concepts , 9th edition,Apendice B pag 5

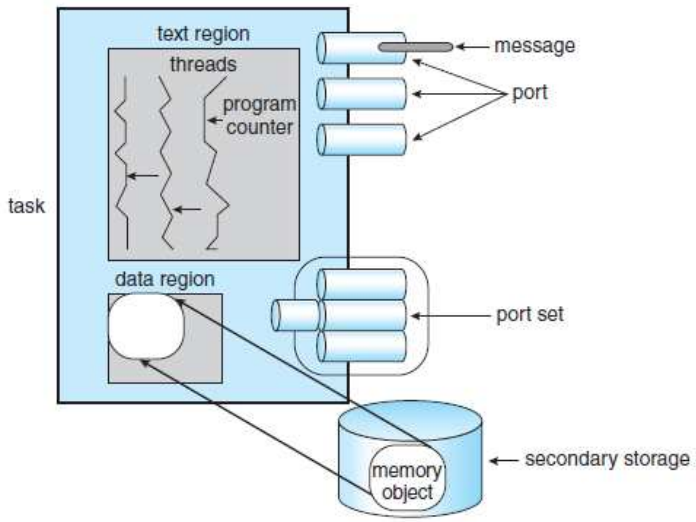

Figura 1. Abstracciones basicas de Mach

Mach combina las funcionalidades de manejo de memoria y comunicación entre procesos a nivel de kernel. Esto funciona representando los objetos de memoria mediante un o varios puertos. Las solicitudes se envían como mensajes IPC a estos puertos para solicitar operaciones a memoria (operaciones de paginación por ejemplo). De esta manera los objetos de memoria pueden estar localizados en sistemas remotos. Prueba de esta combinación de funcionalidades es como Mach utiliza el manejo de memoria para pasar mensajes. El sistema hace esto moviendo punteros a objetos de memoria compartidos, en vez de copiar todo el objeto[8].

Las tareas son vistas como un proceso, solo que no poseen un puntero de instrucción y un conjunto de registros. Una tarea por sí sola no tienen ninguna funcionalidad, los hilos ejecutándose en ella son los que se la dan. Una instrucción de fork() funciona similar a otros sistemas operativos en donde se hace una copia de la tarea que hace el llamado, con la tarea hija teniendo una copia del espacio de direcciones del padre. Los hilos son pueden estar en dos estados, corriendo o suspendidos. Aunque no esté ejecutado instrucciones por algún tipo de excepción (fallo de página por ejemplo). Igual se considera como una hilo corriendo.

Figura 1 .Silberchartz,Galvin,Gagne: Operating System Concepts , 9th edition,Apendice B pag 6

8. http://www.shakthimaan.com/downloads/hurd/kernel_principles.pdf

Para la sincronización de hilos, Mach provee primitivas necesarias para poder programar las herramientas a partir de ellas. La comunicación IPC puede ser utilizada como método de sincronización entre tareas. Para el caso de los hilos, existen llamadas al sistema para parar y reanudar ejecución en tiempos determinados. Mach si no tienen implementado las operaciones de semáforos wait() y signal(), comúnmente asociadas a semáforos, pero es posible implementarlas mediante llamadas IPC. Para implementar esto y muchas otras funcionalidades, Mach provee interfaces de alto nivel para programar en C y otros lenguajes[9].

La planificación de la CPU en Mach manejo exclusivamente hilos, asi que no tiene relevancia a que tarea pertenecen. Todos los hilos compiten en igualdad de condiciones por utilizar la CPU. Cada hilo puede tener una prioridad asociada a el que va desde 0 a 127 en donde los valores representan que tanto tiempo uso un hilo la CPU. Hilos que recientemente usaron una gran cantidad de tiempo de la CPU, después de que se les remueve de la CPU le es asignada una prioridad baja[10].

Para el manejo de excepciones, cada tarea cuando es creada posee un hilo creado por default que maneja las excepciones. Se utiliza la comunicación por medio de mensajes entre el hilo que causa la excepción y el hilo encargado de manejarlas para saber qué clase de error ocurrió y tomar las acciones correspondientes.

La interfaz de programación de Mach le permite al usuario trabajar en varios niveles. Las más baja es a nivel de llamado de sistema el llamado desde un hilo usuario ocasiona una “trap” en el kernel para proveer el servicio que requiere el llamado. A más alto nivel se encuentra el paquete de hilos C. Esta librería provee una interfaz en lenguaje C a las primitivas básicas de Mach, en donde se le puede programar nuevas funcionalidades al sistema operativo sin tener que intervenir con el código del kernel[11].

9.http://www.shakthimaan.com/downloads/hurd/kernel_principles.pdf

10. *Silberchartz,Galvin,Gagne: Operating* System Concepts , 9th edition,Apendice B pag 11

11. *Silberchartz,Galvin,Gagne: Operating* System Concepts , 9th edition,Apendice B pag 24

Como conclusión se puede mencionar que la generalidad de las funcionalidades y reducido número de abstracciones de Mach es lo que lo hizo que fuera un sistema muy utilizado en investigación y como base de otros sistemas operativos derivados de él. Se mantiene en uso en cierta manera hasta el día de hoy por medio de sus derivados. El caso de uno de sus derivados,Darwin, desarrollado por Apple es quizá el más relevante por ser Darwin la base del sistema operativo que usan los dispositivos Apple hoy en día. Ser uno de los primeros microkernels funcionales es quizá el más grande aporte que hicieron sus desarrolladores con su trabajo en él.

Logo de Darwin OS, derivado de Mach desarrollado por Apple.