

On μ -Kernel Construction

CARLOS MARTÍN FLORES GONZÁLEZ, Carné: 2015183528

Instituto Tecnológico de Costa Rica
 Maestría en Computación
 Sistemas Operativos Avanzados
 Profesor: Francisco Torres Rojas, Ph.D

Los sistemas basados en μ -kernel han sido contruidos desde antes que el término fuera introducido. Tradicionalmente, la palabra “kernel” se usa para denotar la parte del sistema operativo que es necesaria y común para todo el otro software. La idea básica del enfoque μ -kernel es la de minimizar esta parte, por ejemplo, implementar fuera del kernel lo que sea posible.

Aunque se ha invertido mucho esfuerzo en la construcción de μ -kernel, el enfoque no ha sido aún aceptado de forma general. Esto debido al hecho de que la mayoría de μ -kernels no tiene un rendimiento suficientemente bueno. La ausencia de eficiencia también restringe la flexibilidad, dado que mecanismos y principios importantes no se pueden poner en práctica debido al pobre rendimiento. Se cree que el modo de *user-kernel* aumentado y los cambios en el espacio de direcciones con los responsables.

Some μ -Kernel Concepts. Un criterio puede ser tolerado dentro de μ -kernel solo si al moverlo fuera del kernel, permitiendo que las implementaciones compitan, evitaría la implementación de una funcionalidad requerida por el sistema. Un requerimiento inevitable para tales sistemas es que un programador debe ser capaz de implementar un sub-sistema S de tal forma que no pueda ser perturbado o corrompido por otros sub-sistemas S' . Principio de independencia: S puede garantizar independencia de S' . El segundo requerimiento es que otros sub-sistemas puedan confiar de estas garantías \rightarrow Principio de integridad: tiene que haber una forma para que S_1 se puede dirigir a S_2 y establecer un canal de comunicación que no pueda ser corrompido o interferido por S' . **Address Spaces:** a nivel del *hardware* un espacio de direcciones es un mapeo que asocia cada página virtual con un marco de página física o bien la marca como “no accesible”. Los mapeos son implementados por *hardware* TLB y tablas de páginas. μ -kernel tiene que ocultar el concepto de *hardware* de espacio de direcciones sino de otra forma, la implementación de protección sería imposible. Se debe permitir la implementación de esquemas arbitrarios de protección (y de no-protección) por encima de μ -kernel. Debería ser simple y similar al concepto de *hardware*. La idea básica es la de brindar construcciones recursivas de espacios de direcciones por fuera del kernel. Existe un espacio de direcciones σ_0 que representa la memoria física y que es controlada por el primer sub-sistema S_0 . Cuando el sistema inicia todos los otros espacios de memoria están vacíos. Para construir y mantener más espacios de memoria por encima de σ_0 , μ -kernel provee estas tres operaciones: (1) *Grant*: el dueño de un espacio de memoria puede otorgar (*grant*) una de sus páginas a otro espacio, siempre y cuando el destinatario acepte. La página se remueve del espacio de direcciones del que otorga¹ y se le asigna al destinatario. (2) *Map*: El dueño de un espacio de direcciones puede mapear cualquiera de sus páginas² dentro de otro espacio de direcciones, si el destinatario acepta. Al final, la página puede ser accedida en ambos espacios de direcciones. (3) *Flush*: el dueño

¹Restricción: Solamente puede otorgar páginas a las cuales tiene acceso

²Restricción: solo puede mapear páginas a las que tenga acceso.

de un espacio de memoria puede liberar (*flush*) cualquiera de sus páginas. La página que se libera se mantiene accesible en el espacio de direcciones de quién la liberó pero es removida de todos los otros espacios de direcciones que hayan recibido directa o indirectamente la página de quién la liberó (el *flusher*). I/O: el control de los derechos de acceso de I/O y controladores de dispositivos se lleva a cabo por gestores de memoria y *paggers* por encima de μ -kernel. **Threads and IPC**: Un hilo es una actividad que se ejecuta dentro un espacio de direcciones. Un hilo τ se caracteriza por un conjunto de registros, un puntero de pila e información de estado. El estado del hilo también incluye el espacio de memoria $\sigma^{(\tau)}$ en donde τ se está ejecutando. Esta asociación dinámica o estática a un espacio de direcciones es la razón decisiva para incluir el concepto de hilos en μ -kernel. Para prevenir espacios de direcciones corrompidos, todos los cambios en el espacio de direcciones de un hilo debe ser controladas por el kernel. Esto implica que μ -kernel incluye la noción de algún τ el cual representa la actividad mencionada anteriormente. La comunicación entre espacios de direcciones (IPC) debe de ser soportada por μ -kernel. El método clásico es transferir mensajes entre hilos por medio de μ -kernel. Interrupciones: abstracción: mensajes IPC. El *hardware* es considerado como un conjunto de hilos que tienen *ids* especiales y envían mensajes vacíos hacia hilos de *software* asociados. Transformar interrupciones a mensajes lo realiza el kernel pero μ -kernel no se especializa en interrupciones específicas de dispositivos. **Unique Identifiers**: μ -kernel provee identificadores únicos para hilos, tareas o canales de comunicación. *Uids* son requeridos para una comunicación local confiable y eficiente.

Flexibility. Algunas aplicaciones que típicamente pertenecen al SO básico pero que pueden ser implementadas por encima de μ -kernel son: Gestores de memoria, *paggers*, asignación de recursos multimedia, controladores de dispositivos, cache de segundo nivel y TLB, comunicaciones remotas, servidores Unix. Lo único que no se puede implementar por encima de estas abstracciones es la arquitectura del procesador, registros, cache de primer nivel y TLBs de primer nivel.

Performance, Facts & Rumors. *Kernel-user switches*: comparado con el mínimo teórico, *user-kernel mode switches* son costoso en algunos procesadores. Por otro lado, en comparación con otros kernels existentes se pueden mejorar de 6 a 10 veces por medio de una construcción apropiada de μ -kernel. *Kernal-user mode switches* no son un problema conceptual serio pero si uno de implementación. *Address Space Switches*: si se construyen adecuadamente no son muy costosos, menos de 50 ciclos en procesadores modernos. En un procesador de 100 MHz, el costo heredado de *switches* de espacios de direcciones puede ser ignorado aproximadamente hasta 100,000 *switches* por segundo. Optimizaciones especiales, como ejecución de servidores dedicados en espacios de kernel, son superfluos. Este contexto costoso de *switching* en algunos μ -kernels existentes se debe a la implementación y no es causado por problemas inherentes con el concepto. *Thread Switches and IPC*: IPC puede ser implementado lo suficientemente rápido para manejar también interrupciones de *hardware* por medio de este mecanismo. **Memory Effects**: la hipótesis que la arquitectura μ -kernel inherentemente conduce a degradación de la memoria del sistema no está justificado. Por el contrario, las mediciones mencionadas apoyan la hipótesis que μ -kernels construidos apropiadamente van a evitar automáticamente la degradación de la memoria del sistema en Mach.³

³Implementación de un μ -kernel.

1. ¿CUÁL ES EL PROBLEMA QUE PLANTEA EL *PAPER*?
2. ¿POR QUÉ EL PROBLEMA ES INTERESANTE O IMPORTANTE?
3. ¿QUÉ OTRAS SOLUCIONES SE HAN INTENTADO PARA RESOLVER ESTE PROBLEMA?
4. ¿CUÁL ES LA SOLUCIÓN PROPUESTA POR LOS AUTORES?
5. ¿QUÉ TAN EXITOSA ES ESTA SOLUCIÓN?