Desarrollo de un administrador de memoria segmentada para un sistema operativo didáctico

Hugo Ryckeboer
h_ryckeboer@yahoo.com.ar

Nicanor Casas
ncasas@unlam.edu.ar

Graciela De Luca
gdeluca@unlam.edu.ar

Universidad Nacional de la Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas
Dirección: *Florencio Varela 1703* - Código Postal: 1754 - Teléfono: *4480-8900/8835*

CONTEXTO

Uno de los objetivos del desarrollo del sistema operativo SODIUM es permitir a los alumnos de las materias de sistemas operativos observar y sacar conclusiones del funcionamiento de los distintos algoritmos que componen un sistema operativo, el tratamiento de los eventos que ocurren, la forma de atención de estos, que injerencia tienen en los tiempos de ejecución de los procesos, en la eficiencia de la administración y la gestión de los recursos. Este administrador de memoria segmentada nos permite continuar con la línea de administradores de memoria, mostrando la evolución de la administración en cuanto al reparto, mejor aprovechamiento de la memoria en áreas compartidas por varios procesos, disminución de la fragmentación externa y aumento de la complejidad del manejo del hardware disponible como forma de asignación del recurso. Todo esto debe realizarse con el mismo conjunto de programas fuente de prueba, compilarlos y ejecutarlos en las mismas condiciones para luego realizar las comparaciones de los diferentes sistemas de administración que posee el SODIUM.

Nuestro Sistema Operativo utiliza actualmente el formato "binario plano" obtenido desde un archivo objeto ELF por lo tanto los procesos de usuario que ejecutarán y serán ubicados en memoria utilizarán formato estandarizado ELF para sus archivos ejecutables, por lo que necesitamos adaptarlos para la segmentación utilizado las secciones y segmentos que este provee.

RESUMEN

La finalidad de crear un administrador de memoria segmentada sin ningún tipo de solapamiento en el sistema operativo SODIUM, es la de generar un administrador clásico para que los alumnos de Sistemas Operativos puedan comprobar el estado de asignación de la memoria, los distintos segmentos en los que queda dividido el proceso, su localización en memoria, la forma en que se traducen las direcciones, como se realiza la asignación dinámica de memoria para el proceso, la asignación de memoria compartida y los mecanismos de protección de las distintas áreas de memoria comparando la eficiencia en la asignación del recurso con la de otros administradores. Se presentaron dificultades que son generadas por la disponibilidad de las diferentes arquitecturas y los compiladores existentes en el mercado, ya que la mayoría trabaja por defecto con paginación o segmentación-paginada. Se tomó un formato estandarizado de archivos ejecutables, como es el ELF y se estudió la adaptación a las necesidades del administrador de memoria, estableciendo relaciones entre las diferentes direcciones ya sean lógicas, físicas o lineales y las distintas secciones o segmentos.

Palabras Claves: Segmentación, paginación, ELF, asignación, reasignación,

GDT, LDT, memoria compartida, dirección virtual, dirección física, dirección lineal.

1.- INTRODUCCIÓN

Nos hemos planteado la necesidad de que un sistema operativo didáctico conste de más de un administrador de memoria, para poder estudiarlos y realizar comparaciones de comportamiento de cada uno. Habiendo realizado ya administradores de asignación contigua simple, particiones fijas y particiones variables, la segmentación [**TAN03]** resulta de la evolución normal de las particiones variables agregando eficiencia en la asignación del recurso, complejidad en la administración y generación de programas para trabajar con memoria segmentada.

Esto nos permite el apoyo a una materia específica, a través de la visualización de las secciones de los procesos**[TISELF]**, cuales de ellas están en el archivo ejecutable en el disco y cuales son creadas por el administrador de memoria, las tablas que utiliza el sistema para la traducción de direcciones **[INTEL]** y para la asignación de memoria del sistema. La entrega de información sobre los eventos que ocurren dentro del sistema, la localización de los distintos segmentos de cada proceso en memoria , aumentado o disminuyendo su tamaño para mejorar la fragmentación externa y la posibilidad de cambiar la configuración para realizar comparaciones. Este administrador está desarrollado por los alumnos, lo cual los lleva a un conocimiento de las particularidades de los compiladores **[VHAG06]**, los formatos de los ejecutables que el administrador de memoria debe conocer para utilizar y de la arquitectura en la cual implementamos nuestro administrador y el entorno que la contiene. También se realiza una transferencia de conocimientos, a través de interfaces para cada funcionalidad, lo que permite que el alumno conciba y escriba módulos alternativos.

Existe una diferencia fundamental entre los sistemas operativos de uso general (tales como Linux o Windows) y el SODIUM ya que este trata intercambiar los distintos administradores de memoria para poder estudiar y sacar conclusiones respecto a la asignación del recurso, traducción de direcciones, protección, siempre respetando los algoritmos de administración de memoria publicados en los libros más afamados de la materia Sistemas Operativos. La diferencia es debida a que los sistemas operativos de uso general trabajan con un único sistema de administración de memoria **[CAR01] [RUSSAL]**, actualmente de paginación por demanda, no permitiendo cambios de configuración ni una modificación sencilla para poder utilizar dos algoritmos o más.

2. LINEAS DE INVESTIGACION y DESARROLLO

Si bien la arquitectura Intel x86 provee estructuras de datos que pueden ser utilizadas para la administración de memoria con segmentación tales como la GDT (Global Descriptor Table) o LDT (Local Descriptor Table) y un conjunto de registros tales como CS , DS, ES, FS; GS y SS , donde CS es un selector de segmento de código, SS un selector de segmento de stack y los restantes son selectores de segmentos de datos, para la segmentación no son suficientes ya que vamos a utilizar varias áreas de código ubicadas en segmentos diferentes debido a que necesitamos por lo menos un área de memoria o segmento para cada una de las bibliotecas dinámicas que se necesiten, para luego poder comparar el estado de asignación de memoria utilizando en la compilación de los procesos bibliotecas estáticas ó dinámicas. También resultarán insuficientes los registros de datos ya que serán necesarios un registro de datos para las variables globales (inicializadas o no –BSS-), un registro para el heap o memoria de asignación dinámica del proceso y además si trabajamos con threads, para cada uno que se ejecute será necesaria un área de datos del thread , no así un área de código del thread ya que este se comparte con el código del proceso. Estos inconvenientes que presenta la arquitectura se solucionan con la tabla de relocación y los registros de relocación del ejecutable ELF. Fig 1“Asignación de memoria del proceso”

En segundo lugar tenemos también, la traducción de las direcciones virtuales Fig 2 " Direcciones Virtuales a Direcciones físicas" que se realizan utilizando una LDT para todos los procesos y en el futuro podrán ser una LDT por proceso. Esto se decidió así ya que la LDT posee una cantidad de entradas suficientes para la cantidad de procesos que podemos generar para estudio. El uso de una LDT por proceso tiene la ventaja que podemos identificar fácilmente todos las entradas del proceso en la tabla y los datos de cada segmento pero no agrega mejoras sustanciales a los objetivos del SODIUM .

Tercero, la asignación y recuperación de memoria se realizará mediante dos métodos diferentes para poder realizar comparaciones. El primero mediante una lista doblemente enlazada de huecos libres y ocupados permitiendo utilizar los algoritmos First Fit, Best Fit, Worst Fit y Next Fit que ya están incorporados al SODIUM para las particiones variables y el segundo con el algoritmo Buddy System, con modificaciones para tamaños de memoria no potencias de dos.

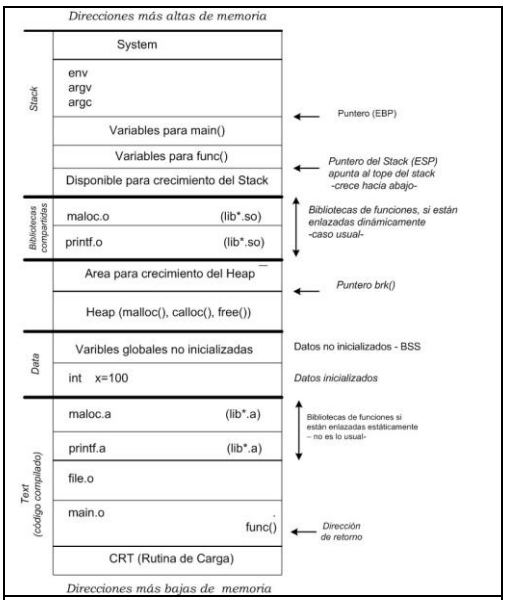

Fig 1"Asignación de memoria del proceso"

3.RESULTADOS OBTENIDOS/ ESPERADOS

Se ha utilizado la misma estructura cambiando los parámetros de lista enlazada para asignar la cantidad de huecos de memoria necesaria para la segmentación y se desarrollo un entorno gráfico para poder mostrar el estado de la memoria y las estadísticas de asignación de memoria a los procesos.

Se está trabajando actualmente sobre el header de programa del ejecutable ELF y sobre el header de sección para generar en la compilación archivos ejecutables compatibles con el modelo segmentado.

Queda aún por generar bibliotecas dinámicas para C en SODIUM para poder generar una mayor cantidad de segmentos

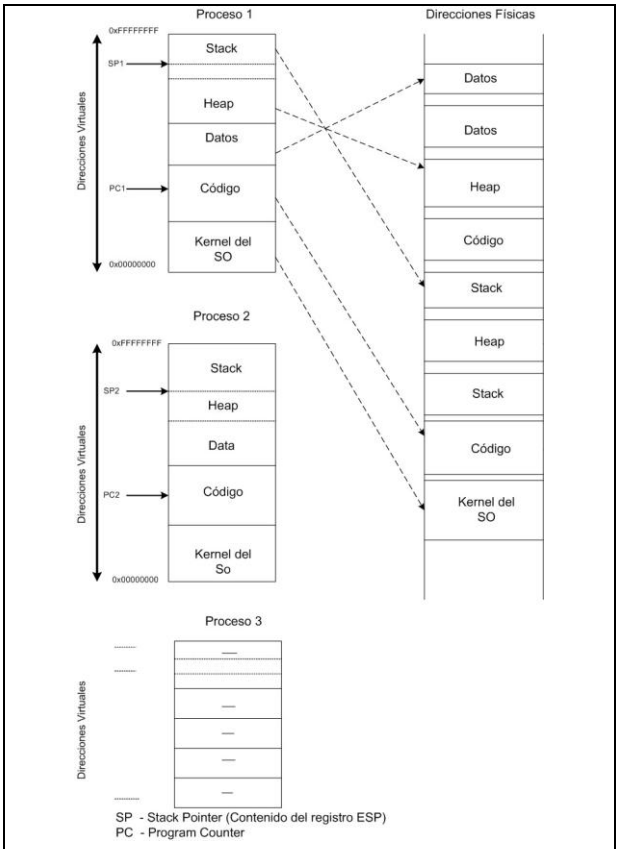

Fig 2 " Direcciones Virtuales a Direcciones físicas"

4. FORMACION DE RECURSOS HUMANOS

Se realizó la primera transferencia de los conocimientos obtenidos a los alumnos que cursan Sistemas Operativos, ya que realizaron el análisis de la arquitectura y las distintos formatos de ejecutables conjuntamente con el análisis del SODIUM e intervinieron en el desarrollo de los administradores, también

permitió realizar una transferencia de conocimientos a los alumnos de Sistemas de Computación II de la Universidad de La Matanza y a los alumnos de Sistemas Operativos de la Universidad Tecnológica Nacional, Regional Buenos Aires. Se publicaron los avances en la investigación en dos congresos internacionales, se prevé continuar con las publicaciones en otros congresos internacionales y se está estudiando el realizar convenios de colaboración con otras universidades nacionales estatales y privadas de las cuales recibimos ofrecimientos de colaboración, con el objetivo de intercambiar conocimientos y ampliar los alcances del sistema.

En esta línea de investigación tenemos dos trabajos de la Maestría en informática en curso, colaboran también en el desarrollo un becario, un alumno de Ingeniería electrónica y un alumno que cursa los primeros años de la carrera de Ingeniería en informática.

5. BIBLIOGRAFÍA

Publicaciones

[ANG98] Angulo José M. y Funke Enrique – Microprocesadores avanzados 386 y 486 – Introducción al Pentium y Pentium – Pro Editorial Paraninfo – Cuarta Edición

[BRE00] BRE00- Brey Barry B. – Los Microprocesadores Intel – Editorial Prentice Hall – Quinta Edición.

[CAR01] Card Rémy, Dumas Eric, Mével Franck - Programación Linux 2.0 API del sistema y funcionamiento del núcleo – Enrolles y Ediciones Gestión 2000 S.A.

[DEI90] Deitel Harvey M. – Introducción a los Sistemas Operativos - Addison-Wesley Iberoamericana – Segunda Edición

[INTEL] Manual de microprocesadores 386 y 486 y Pentium.

[KNUT] Donald E. Knuth. Fundamental Algorithms, The Art of Computer Programming Vol. 1, Addison Wesley, Second Edition,

[MCD07] MacDougall, Mauro, Solaris Internals (Solaris 10 and opensolaris kernel architecture) –Editorial Prentice Hall – Second Edition

[MIL94] Milenkovic Milan – Sistemas Operativos Conceptos y diseño – Mc Graw Hill – Segunda edición

[SIL97] Silverschatz, Avi; Galvin, Peter – Operating System Concepts – Addison-Wesley Longman – Fifth Edition

[SMC00] Standard Microsystems Corporation – Application note 6.12

[STA98] Stallings Willams – Operating Systems Internals and design principles – Prentice Hall International – Third Edition

[TAN97] Tanenbaum Andrew S., Woodhull Albert S. – Operating Systems Design and Implementation – Prentice Hall – Second Edition

[TAN03] Tanenbaum Andrew S.– Sistemas Operativos Modernos – Pearson Education – Segunda Edición

[TISELF] Executable and Linking Format (ELF) Specification – Tool Interface Standard (TIS) Committee

[TUR03] Turley James L. – Advanced 80386 programming techniques – Osborne McGraw Hill

[VHAG06] William Von Hagen – The Definitive Guide to GCC – Apress – Segunda Edición

[RUSSAL] Russinovich, Salomon – WINDOWS INTERNALS Windows Server 2003, Windows XP, Windows 2000 – Microsoft Press

Revistas , Publicaciones e Internet

[01] A Study of Replacement Algorithms for Virtual Storage Computers (1966) by L A Belady IBM Systems Journal

http://www.research.ibm.com/journal/sj/052/belady.pdf

[02] Evaluating models of memory allocation (1994) by Benjamin Zorn, Dirk Grunwald -

ACM- Transactions on Modeling and Computer Simulation
ftp://ftp.cs.colorado.edu/pub/cs/techreports/zorn/CU-CS-603-92.ps.Z

[03] Microsoft "Memory allocation for long-running server applications" (1998)
Per-Åke Larson and Murali Krishnan
ftp://ftp.research.microsoft.com/users/palarson/ismm98_lk.ps

[04] The Unix File System

http://www.isu.edu/departments/comcom/unix/workshop/unixindex.html

[05] The compiler, assembler, linker, loader and process address space programming tutorial - hacking the process of building programs using C language -notes and illustration.

http://www.tenouk.com/ModuleY.html

[06] Jasmin Blanchette, Mark Summerfield, C++ GUI Programming with Qt 3

[07]The Memory Fragmentation Problem: Solved? Mark S. Johnstone,Paul R. Wilson
The University of Texas at Austin
http://grothoff.org/christian/teaching/2007/4705/ismm98.pdf

[08] BAYS, C.: "A Comparison of **Next**-**Fit**, First-**Fit**, and Best-**Fit**", Commun. of the ACM, vol. 20, pp.# 191-192, marzo de 1977