VisualOS

Descripción del proyecto

Manuel Estrada Sainz

ranty@atdot.org
ranty@soon.com

VisualOS: Descripción del proyecto

por Manuel Estrada Sainz

Tabla de contenidos

Capítulo 1. Introducción

El propósito de este proyecto es desarrollar una herramienta gráfica que permita el estudio y comprensión del funcionamiento real de un sistema operativo moderno.

El programa que se presenta permite observar los aspectos más relevantes de un sistema operativo en funcionamiento. También permite ver cómo funcionan y cómo interactúan los tres sistemas mas importantes: planificación de procesos, gestión de memoria y Entrada/Salida. Las representaciones son dinámicas pudiendo capturarse algunas de las gráficas, para su posterior estudio, mostrar el funcionamiento del sistema en "vivo"o experimentar directamente con él.

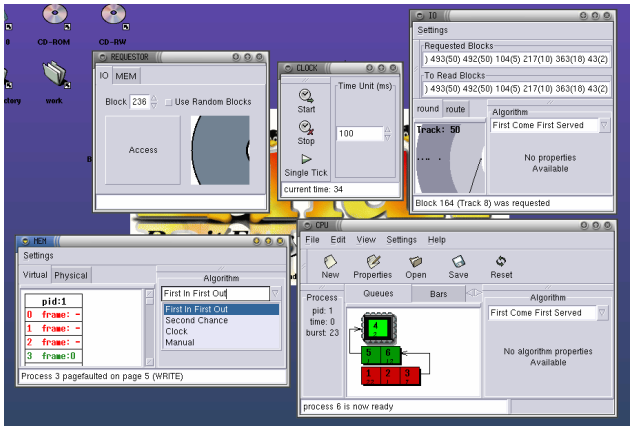
Un sistema operativo consta de muchas partes íntimamente relacionadas, de manera que su buen funcionamiento depende tanto del correcto funcionamiento de cada una de ellas como de su correcta interacción. Este programa permite tanto su estudio global como el estudio concreto de alguna de sus partes.

Tradicionalmente cada parte de un sistema operativo se estudia por separado, siendo muy difícil ofrecer una visión global e integrada de todas ellas, así como de los problemas de la concurrencia, debido a la difícil representación del sistema en funcionamiento.

Hasta ahora la única manera de comprender realmente estos conceptos era escribiendo y modificando el código de un sistema operativo real para luego analizar su funcionamiento. Con este propósito se escribieron sistemas operativos como MINIX, NACHOS o TUNIX. El problema que tiene este método es que requiere mucho tiempo, y para ser realmente útil se necesita un gran dominio del lenguaje de programación utilizado, que permita centrarse en la lógica del problema. Esta nueva herramienta permitirá la realización de laboratorios en los cuales se podrá, en poco tiempo, comprender conceptos fundamentales del funcionamiento real de los sistemas operativos.

En el ámbito docente, el presente programa convierte en más sencillas las tareas anteriores. Facilita al profesor el trabajo de realizar gráficas y otras representaciones que expliquen los conceptos importantes del sistema operativo, y permite que el alumno los vea más claramente, porque podrá realmente ver, por ejemplo, cómo los

distintos procesos se disputan el procesador y la memoria a la vez que pretenden ser los primeros en leer del disco y cómo es el sistema el que gestiona estos recursos. Todo lo verán en conjunto y no fragmentado como se puede explicar en una clase tradicional. Por supuesto también se pueden desactivar algunas de las partes para poder ver más claramente el funcionamiento de las demás. Incluso se pueden mostrar los distintos subsistemas en máquinas diferentes para realizar prácticas en las que distintos alumnos se hagan cargo de cada uno, eligiendo el algoritmo que quiere utilizar o incluso tomando las decisiones manualmente.



Por otro lado, también se podrá utilizar como banco de pruebas, haciendo más fácil y flexible el método de modificar y escribir código, ya que la aplicación está escrita de forma modular y tomando especial interés en hacer lo mas sencillo posible la adición de nuevos algoritmos o modificación de los ya existentes. Para poder hacer esto no será necesaria la comprensión del resto del código, solamente será necesario aprender el interfaz que los algoritmos en cuestión utilizan, una estructura de datos y algunas funciones. Incluso se han escrito funciones para ocultar al implementador de los algoritmos la utilización de GTK+ como librería de elementos gráficos.

Para hacer lo más fácil posible el mantenimiento de la documentación interna y asegurar así la calidad de esta, se ha utilizado un sistema que genera código DocBook (SGML) a partir de comentarios estructurados en el código fuente del programa denominado "gtk-doc". DocBook, es un estándar en la industria (SUN MicroSystems,

Hewlett Packard, O'Reilly ...), y por su carácter semántico permite escribir los documentos una sola vez y de forma automática obtenerlos en diferentes formatos: HTML, PostScript, PDF, DVI, Braille ...

El sistema de control de versiones utilizado ha sido CVS (Concurrent Version System), que es el estándar *de facto* en el ámbito del software libre. El CVS permite gestionar los cambios de colecciones completas de archivos.

Gracias a las herramientas "automake"y "autoconf"la compilación del programa es muy sencilla, ya que se detectan las peculiaridades del entorno de forma automática y sin intervención del usuario.

Se ha considerado importante la utilización de estándares abiertos y herramientas libremente disponibles para facilitar la colaboración en el futuro por parte de los propios estudiantes o de cualquier otra persona que lo desee a lo largo de todo el mundo (mediante Internet). Para lograr estos objetivos se han utilizado herramientas como DocBook (SGML) como formato de documentación, el API de programación GNOME/GTK+, el compilador GCC (GNU C COMPILER) y el sistema de control de versiones CVS (Concurrent Version System).

Capítulo 2. Objetivos

El propósito de este proyecto, es crear un entorno visual en el cual se pueda experimentar fácilmente con un sistema operativo.

Deberá cumplir los siguientes requisitos:

- Tendrá que ser un entorno visual e intuitivo que resulte amigable al usuario.
- Las estructuras de datos mas importantes de dicho sistema operativo se representarán gráficamente para su mejor comprensión.
- Tendrá que permitir la observación de las interacciones del sistema completo, así como de cada una de sus partes por separado.
- Tendrá que ser modular y fácilmente modificable, en especial tendrá que ser sencillo anadir nuevos algoritmos.
- La aplicación será software libre, y tendrá que ser desarrollada utilizando tan solo software libre y estándares abiertos.

Teniendo en cuenta los requisitos anteriores, se deberán representar los siguientes mecanismos:

- Planificación de procesos.
- Planificación de memoria.
- Planificación de disco.

Capítulo 3. Conceptos teóricos

Los conceptos teóricos utilizados para la realización propia del sistema son los conceptos básicos de programación que no es necesario comentar, pero los conceptos que se pretenden ilustrar sí que merecen mención.

El sistema operativo está hecho para dar vida a los procesos, así que comenzaré por ellos. Un proceso, no es más que una lista de instrucciones que están en ejecución y se aplican a datos en memoria principal y secundaria. Además, en los sistemas informáticos actuales, las propias instrucciones también se guardan en memoria secundaria para su almacenamiento y en memoria principal para su ejecución.

La vida de un proceso comienza con la carga de sus instrucciones en memoria principal desde la memoria secundaria y el inicio de su ejecución a partir de su primera instrucción. A lo largo de su vida típicamente necesitará datos de memoria secundaria que almacenará en memoria principal para su procesamiento y posteriormente escribirá los datos resultantes en memoria secundaria y se terminará.

En los primeros tiempos de los ordenadores de arquitectura Von Newmann era directamente el hardware el que cargaba las instrucciones del proceso en memoria e iniciaba su ejecución, el procesador se dedicada enteramente a la ejecución de este único proceso y este disponía de toda la memoria principal y podía utilizar la memoria secundaria libremente.

Actualmente el problema es que pretendemos que varios procesos convivan en la misma máquina, de manera que se aproveche más el hardware disponible (ej. un proceso puede hacer cálculos mientras otro intenta leer de memoria secundaria) y se puedan realizar varias tareas simultáneamente en el tiempo (multitarea) e incluso que varios usuarios puedan compartir la misma máquina (multiusuario). Sin anadir elementos nuevos al sistema, la complejidad a la hora de programar crecería exponencialmente con el numero de procesos, para poder compartir recursos y el fallo de uno solo de los procesos podría afectar gravemente a los demás. Es necesario, pues, una coordinación.

La figura del sistema operativo se introduce inicialmente para realizar la tarea de intermediario entre los distintos procesos. Será cargado en memoria principal y

ejecutado por el hardware y será el sistema operativo el que se ocupe de dar vida a los procesos y de repartir los recursos disponibles entre ellos. Al menos tendrá que realizar las siguientes labores:

- Cargar en memoria principal los distintos procesos.
- Decidir cuál de los procesos es asignado al procesador para ser ejecutado en cada momento.
- Cuando alguno de los procesos solicite memoria será él quien decida qué memoria utilizar, incluso podrá decidir quitársela a otro proceso. De hecho, utilizar memoria libre es sencillo, el problema surge cuando es necesario "robar"memoria a otro proceso, en cuyo caso el contenido será escrito en memoria secundaria, y entregada al que la necesita. Así que son los algoritmos de selección de "víctima"los que se estudian.
- Cuando alguno de los procesos solicite leer de memoria secundaria será él quien lo haga, coordinando y ordenando las lecturas de los distintos procesos para mejorar el rendimiento. Siendo lo más importante en este caso el orden en que se leen los datos.

Este programa ilustra los distintos algoritmos utilizados para estas tareas. Concretamente los algoritmos actualmente implementados son:

3.1. Asignación del procesador

First Come First Served: (Primero en llegar, primero en ser servido)

El primer proceso en condiciones de ejecutarse será el asignado al procesador y permanece asignado hasta no poder ejecutarse más por tener que esperar a algún evento o por haber terminado.

Round Robin

Se establece un tiempo máximo de ejecución o cuanto y se ejecuta cada proceso

hasta que exceda su cuanto o no quiera ejecutarse más, entonces se asigna otro y así sucesivamente. Los procesos en espera de ejecución forman una cola circular.

Shortest Process Next: (El proceso más corto el próximo)

Se ejecuta el proceso más breve de los que estén preparados, y permanece asignado hasta no poder ejecutarse más, momento en el cual se volverá a elegir el proceso más corto.

Shortest Remaining Time: (El tiempo restante más corto)

Se ejecuta el proceso más corto de los que estén preparados, pero a diferencia del SPN el esquema es apropiativo, de manera que siempre que surja otro proceso listo para ejecutarse se comprobará si va a tardar menos en ejecutarse que el actual y si es así será el nuevo proceso el que se asigne y el actual tendrá que esperar.

Highest Response Ratio Next: (El de mayor tasa de respuesta el próximo)

Para cada proceso, basado en el tiempo que va a ocupar el procesador(s) y el tiempo que lleva esperando para ocuparlo (w), Se calcula w+s/s, una vez echo esto el proceso que tenga un valor mayor será asignado al procesador. Este algoritmo es bastante bueno, por que además de dar preferencia a los procesos cortos también tiene en cuenta el envejecimiento de los procesos para evitar así la "inanición".

3.2. Utilización de la memoria: (Algoritmos de selección de víctima)

First In First Out: (Primero en entrar primero en salir)

Se "roban" los fragmentos de memoria de forma cíclica sin tener en cuenta a quién pertenecen o su frecuencia de utilización.

Second Chance: (Segunda oportunidad)

Siempre que un proceso utiliza un fragmento de memoria, éste se señala como utilizado. Al mismo tiempo, para elegir un fragmento se busca uno que no esté señalado y se elimina esta en los que la tienen.

De esta manera se les da una "segunda oportunidad"a los procesos. Si utilizan el fragmento de memoria enseguida, no lo perderán.

Clock: (reloj)

Es similar al "Second Chance" pero en este caso también tiene en cuenta si los datos de los fragmentos han sido modificados desde que fueron cargados en memoria principal. Si no es así, podrán ser descartados evitando un acceso a memoria secundaria. Por lo tanto, intenta evitar el "robo" de fragmentos modificados.

3.3. Planificación de accesos a memoria secundaria

First Come First Served (Primero en llegar primero en ser servido)

Se realizan los accesos en el mismo orden en que son solicitados.

Shortest Seek Time First (El desplazamiento de cabezal más corto el primero)

Se ordenan los accesos minimizando el movimiento del cabezal lector, puesto que es la acción más costosa.

Scan

Se ordenan los accesos de manera que el cabezal lector se desplace de un extremo a otro de la superficie del disco, evitando cambiar de sentido mientras haya una petición referente a una zona más adelante.

N-Step-Scan

Se ordenan los accesos igual que en el caso anterior, pero sólo teniendo en cuenta un determinado número de ellos, y por lo tanto nunca se realizará un acceso posterior a ese número aunque se encuentre en el camino del cabezal.

Capítulo 4. Técnicas y herramientas

Sistema de Documentación: SGML(DocBook v3.1)/gtk-doc-tools v0.3

He utilizado SGML (Standard Generalized Markup Language) por su flexibilidad, este sistema permite dar significado a las distintas partes de un documento de manera que posteriormente se pueden mecanizar muchas de las tareas de elaboración. Ya que el documento no tiene formato sino significado, se pueden asignar distintos estilos de formato según sea el soporte de destino: Papel, Paginas web, PostScript, Portable Document Format, Braille... etc.

Concretamente he elegido el DTD (Document Type Definition) DocBook ya que parece ser el estándar *de facto* en la documentación de productos software, utilizado por empresas como Hewelt Packart, SUN MicroSystems u O'Reilly.

Por ser un estándar internacional (ISO 8879) es totalmente independiente del software utilizado y totalmente portable, al contrario que otros formatos propietarios como Word, WordPerfect, QuarkXpress...

Como ventaja adicional, es un formato basado en texto, y no en caracteres binarios no imprimibles, lo que permite un control de cambios más fácil, especialmente con herramientas como CVS (Concurrent Version System).

También he utilizado gtk-doc-tools, un sistema de generación de documentación SGML a partir de comentarios estructurados dentro del código fuente del programa. Es el sistema utilizado en GTK+ y GNOME para la creación del manual de referencia del API.

API de programación: GNOME v1.2.4/GTK+ v1.2.8/Xwindow v3.3.6

GTK+, una biblioteca de elementos gráficos inicialmente apoyada en Xwindow, que está siendo portada a Win32, fue originalmente desarrollada para satisfacer las necesidades del magnífico programa de tratamiento fotográfico GIMP (GNU Image Manipulation Program) y progresivamente se ha convertido en el estándar *de facto* para la programación de aplicaciones gráficas en el mundo del software

libre, desplazando a la todopoderosa Motif. Posteriormente surgió la iniciativa, de manos de Miguel de Icaza de crear un entorno de programación más cómodo y flexible que denominó GNOME (GNU Network Object Model Environment) que utilizo GTK+ para dibujar ventanas, botones y demás elementos gráficos.

Este entorno, goza de toda la flexibilidad del sistema Xwindow (ejecución remota, independencia del gestor de ventanas ...) y además está libremente disponible, incluido todo su código fuente y proporciona documentación gratuita en múltiples formatos (la editan en SGML).

Además estas herramientas están hechas para ser portables, por supuesto a través de la mayoría de los sistemas de tipo UN*X y muy pronto Win32 (Microsoft Windows 95/98/NT).

Editor: vi/emacs

Para la mayor parte del programa he utilizado una versión mejorada del clásico VI (Visual Interactive:) denominada VIM (VI iMproved).

Posteriormente para la edición de la memoria a partir de las notas he utilizado emacs, con un paquete de extensión para documentos SGML.

Compilador: GCC v2.95.2 (GNU C Compiler)

Es un compilador muy portable, también de libre distribución que es capaz de compilar C, C++, Java, Pascal y Fortran. Está disponible para casi todas las plataformas UN*X, Win32, Beos, DOS, y puede que otras.

Depurador: gdb v4.18 (GNU Debuger)/DDD v3.2.1(Data Display Debuger)

Para la depuración he utilizado el programa de consola gdb y a menudo su "front end" gráfico DDD.

Control de Versiones: CVS v1.20.8 (Concurrent Version System)

Este es el sistema de control de versiones "estándar" en el mundo del software

libre. Permite registrar los cambios de colecciones completas de archivos, pudiendo señalar momentos importantes del desarrollo para su posterior examen, y el mantenimiento de distintas ramas de desarrollo. También permite acceso remoto a las colecciones de archivos para facilitar la colaboración a través de Internet.

Sistema de compilación: automake v1.4/autoconf v2.13

"Automake" genera de forma semiautomática todo el sistema de compilación (archivos Makefile) ricos en funcionalidad y de acuerdo con los estándares de GNU.

"Autoconf"es capaz de detectar las distintas peculiaridades el entorno de desarrollo permitiendo una compilación fácil a todo tipo de usuarios en todas las plataformas soportadas.

Capítulo 5. Aspectos relevantes del desarrollo

El ciclo de vida elegido ha sido el incremental. En primer lugar se ha realizado la infraestructura del sistema para luego disenar e implementar cada uno de los subsistemas: procesador, memoria y entrada/salida.

En todo momento se ha tenido en cuenta la genericidad del código. Siempre que ha sido implementada alguna funcionalidad se ha realizado de manera que pudiera ser utilizada en otras circunstancias similares y por otros subsistemas.

Para minimizar los posibles efectos secundarios y dada la característica concurrente del proyecto, cada subsistema se ejecuta en un proceso diferente y se comunica mediante paso de mensajes (actualmente mediante sockets).

El Reloj (CLOCK) es el encargado de dar una referencia de tiempo común a los demás subsistemas. El tiempo se expresa en "Unidades de tiempo"que representan el tiempo mínimo que puede transcurrir entre dos eventos consecutivos. Esto quiere decir que se pueden ejecutan varias instrucciones en cada "unidad de tiempo". En el caso de la memoria secundaria la cabeza lectora tarda en volar por encima de una pista una "unidad de tiempo"y otra en cada acceso lo que no es muy realista, pues tendría que ser más lento. Esto, sin embargo, no quita generalidad y resulta más didáctico.

Capítulo 6. Trabajos relacionados

Este podría ser el primer entorno visual concebido para facilitar la comprensión de un sistema operativo moderno.

Hasta ahora la única manera de comprender realmente estos conceptos era escribiendo y modificando el código de un sistema operativo real para luego analizar su funcionamiento. Con este propósito se escribieron sistemas operativos como MINIX, NACHOS o TUNIX.

Hay que resaltar, que el método de modificar el código de un sistema operativo real es muy tedioso e incomodo. Además, el resultado de las modificaciones a de ser depurado, con las dificultades que conlleva depurar un sistema operativo.

Capítulo 7. Conclusiones y trabajo futuro

En resumen, estos son los principales logros de este proyecto:

- Se ha logrado un entorno visual e intuitivo para la experimentación con los sistemas operativos.
- El código es modular lo que permite hacer cambios sin necesidad de tener en cuenta todo el contexto.
- Gracias a su modularidad es muy fácil añadir algoritmos y representaciones nuevos sin necesidad de entender el resto del programa.
- El programa es portable según el estándar POSIX, y gracias a las herramientas "automake" y "autoconf" detecta automáticamente las peculiaridades del entorno de desarrollo permitiendo una compilación fácil para todo tipo de usuarios.
- Se ha utilizado control de versiones desde el primer momento del desarrollo por lo que ha quedado reflejada toda la historia del proyecto. Esto ha permitido la obtención automática de una lista de cambios detallada y la posibilidad recuperar el código perteneciente a cualquier momento del desarrollo.
- La utilización de un interfaz vistoso y con colorido hace el programa mas atractivo y ameno para el usuario.
- Se ha dedicado especial esfuerzo en escribir código claro, modular y genérico lo que hace la aplicación mas mantenible.

En cuanto a trabajos futuros hay que resaltar que una vez expuesto este proyecto, será publicado en Internet para su desarrollo ulterior. Por esta razón el manual del programador está escrito en inglés.

Concretamente se podría implementar un buffer caché, extender el programa para ilustrar un sistema multiprocesador o hacer que cada subsistema se ejecute en una máquina distinta a través de una red. Ya es posible mostrar cada subsistema en una

máquina distinta gracias a la flexibilidad de Xwindow pero de momento todo se ejecuta en el mismo lugar.

Cuando la adaptación de GTK+ a Win32 esté suficientemente terminada, se podrían generar binarios para este sistema operativo. Esto facilitaría, al menos en un primer momento, el acceso al programa a los posibles usuarios de este sistema.

Aspectos más concretos con respecto al trabajo futuro se pueden encontrar en el archivo "TODO"de la distribución software.

black