ENSEÑANZA DE SISTEMAS OPERATIVOS CON NACHOS: UNA EXPERIENCIA ESPAÑOLA

José Miguel Santos Espino[1], Carmelo Rubén García Rodríguez[2]

[1] *Departamento de Informática y Sistemas, Universidad de Las Palmas de Gran Canaria*
correos: {jomis,rgarcia}@dis.ulpgc.es

RESUMEN: En este documento se describe la experiencia de uso del sistema operativo didáctico Nachos en la enseñanza de la materia de Sistemas Operativos, dentro de las ingenierías informáticas de la Universidad de Las Palmas de Gran Canaria. Se describen los criterios que han motivado la elección de esta herramienta y se presentan los resultados tras los dos primeros años de implantación.

1.- PRESENTACIÓN

Los autores de este artículo somos responsables docentes de la materia de Sistemas Operativos en las titulaciones informáticas de la Universidad de Las Palmas de Gran Canaria. Con motivo de la reciente implantación de los nuevos planes de estudios (1998), tuvimos que enfrentarnos a una drástica reducción de créditos en la asignatura correspondiente. Hasta ese momento, la asignatura tenía asignada una carga de 15 créditos, 9 teóricos y 6 prácticos. Las prácticas adiestraban a los alumnos en programación de sistemas bajo UNIX, concurrencia mediante IPC e implementación de módulos de sistema operativo, utilizando un *software* de confección propia.

En las nuevas ingenierías, la materia de Sistemas Operativos se convierte en una única asignatura cuatrimestral, con la mitad de créditos (4'5 teóricos, 3 prácticos). Afrontamos el reto de mantener unos objetivos de formación similares a la situación anterior, pero sin que los estudiantes resulten sobrecargados de trabajo. Con este fin hemos explorado distintas alternativas en la realización de las prácticas y hemos encontrado la mejor opción en un sistema operativo didáctico llamado Nachos.

2. LAS NUEVAS PRÁCTICAS

a) Objetivos y enfoque metodológico

Consideramos que el estudiante de esta asignatura de Sistemas Operativos debe adquirir un conocimiento general sobre estas dos materias:

- Estructura, funcionamiento y diseño de los sistemas operativos
- Fundamentos de la concurrencia y la sincronización entre procesos

Esos son, por tanto, los conocimientos que deben reforzar las actividades prácticas. La cuestión es cual es el mejor enfoque para conseguir este objetivo. Las diversas propuestas de actividades practicas en la enseñanza de Sistemas Operativos suelen encajar en uno de estos tres modelos:

1. Simulación de algoritmos y políticas
2. Implementación de módulos de un sistema operativo
3. Programación de sistemas (llamadas al sistema UNIX, etc.)

Nosotros nos decantamos por el segundo enfoque, puesto que es el único que abarca los tres ambitos de la materia: estructura, funcionamiento y diseño del sistema operativo. Las simulaciones no ejercitan el conocimiento de la estructura, y la programación de sistemas sólo aproxima al alumno a los servicios del sistema operativo, a su interfaz. El gran inconveniente del segundo enfoque es la mayor dificultad para el estudiante.

b) ¿Qué herramienta elegir?

Una vez asumido el enfoque de implementación de módulos, hemos de decidir cuál es la herramienta más apropiada. Desde luego, hay que proporcionar al estudiante un sistema operativo básico con el cual trabajar. Así el alumno no tiene que malgastar tiempo en construir un *software* desde cero. Por otra parte, el estudiante se acostumbra a manejar código ajeno.

Se puede emplear como sistema de trabajo un sistema operativo real, como Linux o Mach. En este caso, el estudiante ha de consumir mucho tiempo estudiando un código de gran tamaño y repleto de optimizaciones que lo hacen ilegible. A esto se añaden las dificultades prácticas que entraña la modificación el sistema, ya que hay que reinstalar el sistema operativo, rearrancar el equipo, etc., por no hablar de la enorme dificultad de depuración.

Para reducir estos costes, desde hace décadas se han desarrollado sistemas operativos *didácticos*, cuya arquitectura es deliberadamente simple, están bien documentados y son relativamente fáciles de modificar. Un ejemplo clásico es Minix [1], que corre sobre hardware PC-Intel. Aunque Minix es más llevadero que un núcleo real, sigue siendo demasiado voluminoso para nuestros estudiantes y no resuelve el problema de la modificación y depuración. Este problema radica en que se trata de un sistema que se instala sobre el hardware nativo.

Otros sistemas didácticos se basan en la *emulación* de una máquina virtual. El sistema operativo administra un hardware virtual más sencillo, con lo cual la arquitectura del software se simplifica. Además, si la máquina virtual corre como programa de usuario sobre el sistema operativo nativo, la depuración y modificación del sistema emulado es tan simple. Este enfoque ha sido empleado satisfactoriamente en España [2], [3]. Y es éste el tipo de sistemas que nosotros buscamos.

2.- EL SISTEMA OPERATIVO NACHOS

De entre toda esta gama de alternativas, hemos optado por el sistema didáctico Nachos, descrito en [4] y [5]. Se trata de un núcleo desarrollado a lo largo del año 1992 por W. Christopher, S. Procter y T.E. Anderson para la docencia de Sistemas Operativos en la Universidad de Berkeley, EUA. El código fuente y la documentación original de Berkeley están disponibles públicamente en Internet. Desde su nacimiento, el uso de Nachos se ha extendido por decenas de universidades norteamericanas. También se emplea en varias universidades iberoamericanas,

desde Guatemala hasta Chile. No nos consta ninguna experiencia docente con Nachos en España.

a) Características de Nachos

Nachos es un pequeño núcleo escrito en C++, que suministra servicios básicos de manejo de procesos, memoria, archivos, entrada/salida y redes. Los servicios de Nachos se han simplificado al máximo, consiguiendo un núcleo completamente funcional en unas pocas páginas de código en C++. El núcleo Nachos se ejecuta como un proceso UNIX y administra los recursos de una máquina virtual basada en un procesador MIPS.

El código fuente de Nachos es bastante legible. Abundante en comentarios, organiza los servicios en clases de C++ con interfaces muy claras y predecibles. En cuanto al lenguaje, prescinde de características como la sobrecarga, excepciones, *templates*, etc. en aras de la simplicidad. Gracias a ello sólo es necesario un conocimiento mínimo de C++ para poder entender el código fuente. El volumen del código es bastante pequeño: los estudiantes han de manejar realmente apenas 2500 líneas de código (incluyendo comentarios). El emulador ocupa otras 2500 líneas, pero no hay que conocer su implementación.

La emulación de la máquina virtual simula el repertorio completo del procesador MIPS (salvo instrucciones de coma flotante). Los programas de usuario Nachos han de estar escritos en código MIPS, pero se puede utilizar un compilador de lenguaje C (ej. gcc). Así pues, los estudiantes pueden usar un lenguaje de alto nivel ya conocido por ellos para probar los servicios del núcleo Nachos.

b) Tareas propuestas en Nachos

Nachos es un sistema operativo deliberadamente mutilado: en general, sólo proporciona mecanismos básicos de sincronización, recepción de eventos, traducción de direcciones de memoria, etc. No tiene implementados servicios de llamadas al sistema, ni memoria virtual. El estudiante es quien tiene que rellenar las carencias. Con este fin, el autor del Nachos propone cinco trabajos de programación: 1)hilos y sincronización; 2)llamadas al sistema y multiprogramación; 3)TLB y memoria virtual; 4)archivos; 5)redes y sistemas distribuidos. El código fuente del Nachos está organizado teniendo en cuenta esta división de trabajos.

De los cinco trabajos, el primero consiste en implementar cerrojos y variables condición (Nachos sólo implementa hilos y semáforos) e implementar programas concurrentes de prueba. En el segundo trabajo, hay que implementar la captura de excepciones y llamadas al sistema, y tras ello implementar la carga de programas de usuario y su ejecución multiprogramada.

c) ¿Por qué Nachos?

Nachos encaja a la perfección con nuestro modelo didáctico (implementación de sistemas operativos). Nachos nos resulta especialmente atractivo porque, siendo extremadamente simple y pequeño, abarca todos los ámbitos de interés en la materia de Sistemas Operativos. Hay que reconocer el acierto de los autores en destilar la quintaesencia de un sistema operativo completo. Por lo demás, está avalado por su uso desde hace varios años en decenas de universidades extranjeras.

Es software de uso gratuito y se puede ejecutar sobre GNU-Linux, con lo cual no hay que invertir ni una peseta en *software* para trabajar con él. Esta misma razón, unido a su pequeño tamaño, facilitan que los alumnos puedan realizar las prácticas en casa si lo desean. Afortunadamente, esta característica la comparten muchos otros sistemas.
Como Nachos se ejecuta como un proceso UNIX, no exige ser instalado en exclusiva en el *hardware*, como ocurriría con el Minix. Por tanto, puede haber varias copias de Nachos ejecutándose en una misma máquina, con el consiguiente ahorro de recursos. Nachos tiene la gran ventaja sobre Minix y Linux de correr sobre un hardware emulado, que se puede controlar al milímetro. Por ejemplo, se puede contabilizar la ocurrencia de fallos de página, interceptar excepciones, etc.

Como se está emulando un procesador MIPS, los alumnos pueden escribir programas de usuario tan complejos como quieran en lenguaje C. Estos programas son traducidos a código MIPS con un compilador y luego ejecutados por Nachos. Por ejemplo, podemos escribir dos versiones de un programa de multiplicación de matrices y analizar cuántos fallos de página genera cada uno. Esta facilidad no es frecuente en otros sistemas basados en emuladores.

d) Inconvenientes potenciales

Desde un principio, encontramos dos características que podrían dificultar al alumno el aprendizaje de Nachos. En primer lugar, el código y la documentación originales están escritos en inglés, lo cual es una dificultad añadida para los estudiantes castellanoparlantes. En segundo lugar, la documentación suministrada es insuficiente para que el alumno pueda comprender la herramienta y mucho menos para guiarle en las tareas que debe realizar.

3.- LA EXPERIENCIA EN LAS PALMAS DE GRAN CANARIA

Nachos fue implantado en el curso 1998/99, el primer año en que se impartía la nueva asignatura de Sistemas Operativos.

a) Perfil de los estudiantes

El primer año, la población de alumnos contenía aproximadamente un 25% de personas provenientes del antiguo plan de estudios, muchos de los cuales habían realizado parte de las prácticas anteriores. Los estudiantes deberían conocer el lenguaje C, aunque la realidad es que el nivel promedio es bastante elemental. La gran mayoría no ha utilizado sistemas UNIX anteriormente.

b) Adaptación del software

Fue necesario adaptar el código original del Nachos para que pudiera compilar en el entorno Linux y retocar algunos fuentes para adaptarlos al reciente estándar ISO de C++. También hubo que generar un compilador cruzado de C a MIPS, a partir del GCC.

c) Metodología docente

De los cinco trabajos propuestos en Berkeley, se han escogido sólo los dos primeros: el trabajo de hilos y sincronización y el trabajo de llamadas al sistema y multiprogramación. El trabajo sobre redes queda fuera del ámbito de la asignatura impartida. Hemos tenido que renunciar –por ahora- a los trabajos sobre memoria virtual y archivos porque a nuestro juicio no se pueden realizar con los créditos prácticos disponibles.

La documentación original de Nachos es insuficiente para que el alumno entienda bien las tareas propuestas. Para soslayar esta dificultad, hemos atacado en dos frentes. Primero, elaboramos una documentación en castellano que describe con más profundidad la estructura del Nachos. La descripción del Nachos está basada sobre todo en la guía de la Universidad de Duke [6].

Segundo, hemos descompuesto cada uno de los trabajos originales en una serie de ejercicios más simples que orientan el trabajo del alumno. A su vez, cada ejercicio se estructura en tres fases: 1)preguntas previas sobre conceptos que el alumno ya debe conocer antes de iniciar el trabajo; 2)explicación de conceptos necesarios para entender el ejercicio; 3)propuesta de trabajo. De esta forma, el estudiante dispone de un itinerario bastante claro para realizar los trabajos y, sobre todo, entender lo que está haciendo.

En las primeras clases prácticas, los alumnos reciben adiestramiento sobre las herramientas de desarrollo, es decir, el entorno Linux y el lenguaje C++ (sólo los elementos imprescindibles para entender Nachos). Tras esta fase, las horas prácticas se dedican alternativamente a dos tipos de actividades: a)explicar las subtareas y proponer su realización; b)supervisar y conducir el trabajo de los alumnos.

d) Resultados

La experiencia ha sido altamente positiva. Los estudiantes consiguen en general completar los trabajos con éxito y con menor esfuerzo que en años anteriores. Nos hemos sorprendido gratamente de la velocidad con que los estudiantes consiguen aprender las interfaces de Nachos. Eso sí, los estudiantes que se lanzan desde el primer día a escribir código suelen acabar atascados al poco tiempo, por no tener clara la arquitectura del sistema.

Hemos observado que, gracias a Nachos, los alumnos adquieren un más que notable conocimiento sobre los fundamentos estructurales de un sistema operativo: sistema de interrupciones, llamadas al sistema, modo dual de operación, etc. Esto se refleja positivamente en los resultados de los exámenes teóricos. En nuestra opinión, esto se debe a que los trabajos con Nachos les proporcionan una experiencia directa de las entrañas del núcleo de un sistema operativo.

e) Opinión de los alumnos

Nos resultaba de especial interés la opinión de los estudiantes repetidores, quienes habían experimentado los dos modelos de prácticas. El comentario unánime fue que las nuevas prácticas resultaban mucho más sencillas, tanto en el planteamiento como en el desarrollo: bastantes repetidores recordaban con espanto lo complicado que era depurar una aplicación concurrente en UNIX.

En el segundo año se han realizado encuestas acerca de la dificultad de los elementos del Nachos y sobre su utilidad en la asignatura. El 49% de los alumnos considera que las prácticas le han ayudado bastante o mucho a comprender los conceptos teóricos de la asignatura. Los alumnos valoraron la dificultad de tres elementos: el manejo de información en inglés (17'3%), el lenguaje C++ (20%) y el esfuerzo para comprender los trabajos (46'7%). [Entre paréntesis están los porcentajes de alumnos que consideran cada elemento muy difícil]. Como puede apreciarse, el inglés y el C++ no representan problema para los estudiantes; sin embargo, casi la mitad de los encuestados encuentran bastante difícil comprender los trabajos en su conjunto. En este punto creemos que la tutorización de los profesores resulta fundamental para que el alumno adquiera una visión general de la arquitectura del sistema Nachos.

f) Un punto débil: el modelo de concurrencia

El planificador de Nachos no es expulsivo (*preemptive*). Un hilo sólo abandona la CPU cuando se bloquea o cuando la cede voluntariamente. Esta política reduce las posibles combinaciones en la ejecución de un conjunto de hilos. Por este motivo, es difícil que el alumno verifique que sus programas concurrentes son correctos.

Por otra parte, en el primer trabajo los alumnos comienzan implementando unas herramientas que jamás han usado (cerrojos y variables condición), con lo cual ellos no son completamente conscientes de lo que están programando. Esto se suple con el siguiente paso, que consiste en usar esas herramientas en programas multihilo, pero como aprendizaje del *uso* de herramientas de sincronización no es ni de lejos la mejor opción didáctica.

g) El futuro

Nuestro principal reto es ser capaces de encajar en las 30 horas disponibles las prácticas de memoria y archivos. Esto tendrá que lograrse a costa de simplificar los dos trabajos actuales, tarea en la que nos encontramos actualmente. Además de ello, vamos a revisar la primera tarea (concurrencia) para que tenga más eficacia didáctica: modificaremos el planificador de Nachos para hacerlo expulsivo a voluntad y alteraremos el planteamiento del trabajo de sincronización.

5.- CONCLUSIONES

El sistema operativo didáctico Nachos se ha implantado eficazmente como herramienta práctica en una ingeniería informática. En escasamente treinta horas prácticas es posible adiestrar a los alumnos en Linux y C++ y además conseguir que completen los dos primeros trabajos.

El código fuente de Nachos está organizado de forma muy legible y fácil de aprender. En caso de no conocer C++, es bastante fácil para el alumno adquirir el nivel necesario para realizar los ejercicios.

Nachos sirve para introducir al estudiante en la concurrencia y sincronización entre procesos, aunque de forma limitada.

La documentación original es insuficiente para un óptimo desempeño del alumno. Es necesario suplirla con documentación propia. La acción tutorial también se considera importante.

Se observa que el aprendizaje práctico con Nachos repercute positivamente en la adquisición de los conocimientos teóricos.

6.- REFERENCIAS

[1] *Operating System Design and Implementation.* Andrew. S. Tanenbaum. Prentice-Hall, Englewood Cliffs. 1987.
[2] *DSO: Herramienta para el aprendizaje de sistemas operativos.* S. Sánchez, D. Meziat, I. García y M.J. Onievas. II Jornadas nacionales de innovación en las enseñanzas de las ingenierías. 1997.
[3] *Ús i descripció dels nivells del sistema operatiu ONION v.4.0.* C. Barrado, L. Doreste y otros. Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya. 1991.
[4] *The Nachos Instructional Operating System.* W. Christopher, S. Procter, T. E. Anderson. Proceedings of the 1993 Winter USENIX Conference. 1993.
[5] *Operating System Concepts*, 4th edition. A. Silberschatz, P. Galvin. Addison-Wesley. 1994.
[6] *A Road Map Through Nachos.* T. Narten, Duke University. 1995.
Nuestra web sobre Nachos está en http://sopa.dis.ulpgc.es/nachos