La asignatura Sistemas Operativos I en el 2mil2

José Antonio Gómez Hernández
Dpto. de Lenguajes y Sistemas Informáticos
Universidad de Granada
18071 Granada
e-mail: jagomez@ugr.es

Resumen

El artículo expone algunas ideas sobre cuales deben ser los contenidos y enfoque adecuados para la teoría de una asignatura de introducción a los sistemas operativos. Esta concepción de la asignatura pretende satisfacer las necesidades de formación en la materia que, en mi opinión, tienen nuestros alumnos, se basa principalmente en la experiencia acumulada impartiendo diferentes asignaturas sobre la disciplina, y toma en consideración los avances producidos en Sistemas Operativos en los últimos años.

1 Introducción

Existe una evolución continua en la forma de enseñar sistemas operativos dado el reto que esto supone por la cantidad de material que se ha de cubrir y por el equilibrio que ha de existir con la práctica. Si bien las unidades que se han de cursar están suficientemente claras en las propuestas curriculares como [1], mi inquietud surgió hace ya tiempo a la hora de abordar cuales deben ser los contenidos concretos que han de cursar los alumnos de la asignatura y el enfoque que se le ha de dar. Esto me llevó a plantear cual debe ser la formación global que debe recibir un ingeniero en informática.

Diferentes estudios técnicos que se vienen publicando sobre empleo revelan que gran parte de las profesiones del futuro próximo serán de nueva creación. En este sentido, parece razonable que la Universidad debe estar preparada para una avalancha de titulaciones de nueva la creación o bien la modificación de las ya existentes. Aunque los profesionales del mañana los formemos hoy, no podemos enseñar cómo será la ciencia y tecnología venidera. Esto hace que, como en otras parcelas, en el campo de las tecnologías de la información debemos formar ingenieros capaces de operar más allá de su actual práctica, que sean capaces de empujar las fronteras de su educación actual. En la parcela que nos ocupa, no podemos dejar de prepararlos para abordar los avances conceptuales y tecnológicos que se están produciendo en la disciplina [11].

Partiendo de esa idea, he ido depurando y dando forma a una propuesta de curso en la que, cubriendo los conceptos básicos de sistemas operativos, se intenta ir dejando un poso que integre dichos conocimientos con los impartidos en el resto de las asignaturas, al objeto de dejar claros una serie de principios válidos a todas ellas. Este enfoque unificador es el de diseño de sistemas.

De las numerosas propuestas realizadas, que podemos ver en artículos recientes en SIGCSE[1] y OSR[2], tenemos desde enfoques *top-down* [7] hasta *bottom-up* [4]. Desde mi punto de vista hay que evitar aquellas propuestas que:

a. No sean realistas al no contemplar que el funcionamiento de un sistema operativo es más complejo de lo que se deriva de las descripciones de sus partes.
b. Mantienen una visión historicista, como ocurre en algunos de los libros de texto de sistemas operativos. Muchos de ellos son el resultado de un proceso acumulativo de conocimientos desde su primera versión. Esto no sólo tiene el problema inmediato de que su contenido es tan amplio que no es factible cubrirlo en los créditos asignados a una asignatura, sino que, además, dificultan el discernimiento entre lo que es obsoleto y los avances recientes.

[1] Grupo de interés de la ACM en Computer Science Education.
[2] Operating System Review de la ACM.

Citemos algunos ejemplos. Sobre el primer punto, la mayoría de los libros de texto descomponen el sistema en módulos pero luego, no lo recomponen, no se analizan los caminos de código cuando se realiza una llamada al sistema o cuando se produce una interrupción. Sorprende ver cómo se estudian los manejadores de interrupciones, pero no se sabe muy bien cómo llega la información de un dispositivo hasta la aplicación que la solicitó, o cómo se desbloquea al proceso llamador. Sobre el segundo punto, se pueden dedicar algunas páginas a explicar un sistema operativo por lotes y, tan siquiera se describen las características de un sistema operativo empotrado.

Opino que el uso de código real (enfoque bottom-up) no es adecuado en un nivel introductorio. Como he podido constatar en cursos de este tipo, el conocimiento previo del que disponen los alumnos de segundo curso no es aún el adecuado, es muy restrictivo pues no permite el análisis de alternativas de diseño, y es difícil de encajar en una asignatura cuatrimestral.

Un enfoque estrictamente top-down presenta el problema de no ser realista en cuanto al reconocimiento de la complejidad de un sistema operativo real. Por ello, mi propuesta para el curso pretende llegar a un punto de equilibrio entre ambos enfoques, un enfoque de diseño. Según la cual, partiendo de un enfoque top-down llegar a analizar los elementos suficientes de implementación para llegar a comprender el funcionamiento real de muchos sistemas. Un tipo similar de propuesta puede verse en [8].

2 Las asignaturas de sistemas operativos

La ETS de Ingeniería Informática de la Universidad de Granada imparte las titulaciones de Ingeniero en Informática, Ingeniero Técnico en Informática de Sistemas e Ingeniero Técnico en Informática de Gestión. En las tres titulaciones se imparten las asignaturas de Sistemas Operativos I, troncal y con una carga docente de 4.5T + 1.5P, y Sistemas Operativos II, obligatoria y con la misma carga docente que la anterior. Ambas asignaturas se imparten en segundo curso, en el primer y segundo cuatrimestre, respectivamente. Además, la titulación de Ingeniero en Informática contiene la asignatura optativa Diseño de Sistemas Operativos con una carga docente de 3T+3P.

De otra parte, existen una serie de asignaturas que podemos considerar como prerrequisitos de las antes citadas como son Metodología de la Programación I y II, e Introducción a los Computadores, que se imparten en primer curso. En segundo curso, se imparten en paralelo las asignaturas de Estructura de Computadores I y II. Estas asignaturas permiten eliminar algunos contenidos de estructura y funcionamiento del computador de nuestra asignatura e ilustrar parte de la interacción *hard-soft*.

3 Propuesta de contenidos

El programa propuesto para la asignatura recoge los descriptores impuestos por la troncalidad e incluye los temas básicos de una asignatura introductoria: introducción a los sistemas operativos, gestión de procesos, sincronización y comunicación, organización de memoria, gestión de memoria virtual, gestión de entradas/salidas y sistema de archivos.

El temario de Sistemas Operativos II incluye los siguientes puntos: implementación de los sistemas de archivos, planificación de recursos, protección y seguridad, una introducción a los sistemas operativos distribuidos, e implementación del núcleo de un sistema operativo centrado principalmente en el estudio Unix, así como una visión general de Windows 2000.

El temario de Sistemas Operativos I parece bastante clásico, si bien se aparta de esta línea en la forma de enlazar los diferentes temas y en los contenidos asignados a cada unidad. De acuerdo con [10], la experiencia muestra cómo los alumnos acceden a la asignatura con entusiasmo entendiendo que los sistemas operativos son un elemento clave en un sistema de computador, pero, desafortunadamente, finalizan a menudo el curso un poco desencantados, en especial cuando se sigue un enfoque estrictamente *top-down*, en el cual deben memorizar un importante número de primitivas de sincronización, políticas de planificación de procesos, y algoritmos de sustitución de páginas. Evidentemente, estos temas son importantes de por sí, pero no son más que un componente en un gran cuadro. Por ello, es importante entrar en detalles de implementación para hacerles ver que no hay nada mágico en el funcionamiento del sistema. En este punto hay que llegar a un equilibrio entre la casi imposibilidad de explicar código real y explicar suficientes detalles para que el alumno imagine el código a nivel de funciones. Por ejemplo, centrándome en una de las

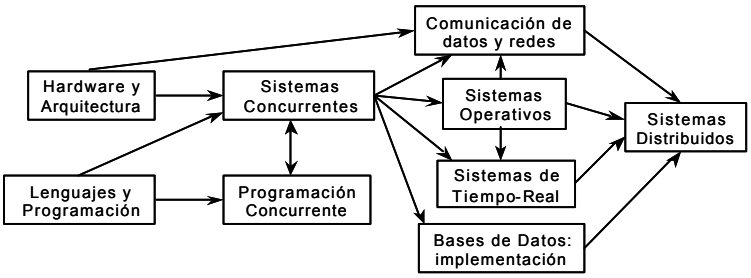


Figura 1.- Visión general de los sistemas concurrentes, modificada de [3]

unidades pilares de la asignatura, la implementación de procesos y concurrencia: explicar cómo se lleva a cabo el cambio de contexto, la operación de bloqueo de procesos, la invocación de una llamada al sistema, la gestión de interrupciones y cómo el diseño realizado en la mayoría de los sistemas operativos esconde éstas a los procesos.

A continuación, comentaré los contenidos más relevantes de cada unidad.

3.1 Introducción a los sistemas operativos

La experiencia ha ido mostrándome cómo los estudiantes tienden a ver en los primeros cursos de la carrera los sistemas operativos como algo aislado. Es interesante y enriquecedor para ellos tener una visión más amplia, por ello es útil en el primer tema dar una visión general de los sistemas concurrentes (Figura 1) y cómo se materializan éstos en las asignaturas de su plan de estudios. Esto les permite crear relaciones entre asignaturas que de otra forma les parecerían compartimentos estancos.

Otro elemento básico a tratar es la creación de abstracciones, ver [5], ya que de hecho el kernel es la implementación de la interfaz definida por las llamadas al sistema. Razonar sobre la interfaz es importante desde el punto de vista del diseño: conocer por qué las funciones de la interfaz tienen la estructura que tienen, por ejemplo, `fork` o `read()` de Unix frente a `CreateProcess` o `ReadFile()` de Windows, o cómo la interfaz define la complejidad de la máquina abstracta que tratamos de implementar. Es bueno ilustrar varias interfaces de programación, por ejemplo, la POSIX de Unix y la Win32 de Windows que son actuales y siguen dos filosofías distintas, como recoge la elogiable propuesta realizada en [6].

El estudio de las llamadas al sistema nos lleva a vislumbrar los diferentes tipos de servicios suministrados por el sistema operativo, lo que nos da pie a estructurar el sistema en grandes módulos, cada uno de los cuales suministra un tipo de servicio. Aquí es importante hacer ver al alumno de manera intuitiva que la dependencia entre módulos es compleja. Por ejemplo, para realizar una actividad necesitamos crear un proceso, el cual necesita memoria y su programa ejecutable está en un archivo al igual que los datos que maneja. Esto da pie para introducirlos en el estudio de arquitecturas de sistemas operativos guiados por el intento de dar respuesta a la cuestión de cómo construir un sistema que sea fiable, extensible y eficiente. Este apartado esta centrado en las arquitecturas monolítica y microkernel, si bien se ven brevemente la estructura de capas y la de Máquina Virtual, por aparecer en muchos sistemas actuales. Para analizar las arquitecturas anteriores es necesario introducir el concepto de espacio de direcciones, recordar el paradigma de llamada a procedimiento ampliándolo, a procedimiento protegido -llamada al sistema- y avanzar el de paso de mensajes.

Para finalizar el tema, se repasan los principales tipos o clases de sistemas operativos, evitando al máximo explicar sistemas en desuso. Se estudian los sistemas multiprogramados, por contener los elementos básicos de un sistema operativo actual, posponiendo para el tema de sincronización una justificación detallada de la multiprogramación. Por supuesto se define y caracterizan los sistemas de tiempo compartido, los sistemas paralelos y distribuido, los de tiempo-real y los empotrados.

3.2 Procesos y hebras

En el tema de procesos, uno de los aspectos más relevantes es mostrar a los estudiantes que un proceso es una abstracción introducida para aislar las diferentes actividades del sistema y poder razonar sobre ellas de forma separada. Sorprende ver como este concepto está tan arraigado en ellos que les parece que se produce de forma natural en la propia estructura del computador. Desde esta idea se abordan las estructuras de datos necesarias para construir esta abstracción. Se explica con detalle esa misteriosa operación de cambio de contexto. En la parte de planificación se hace hincapié en cómo se activa el bucle de despacho, y respecto a los algoritmos de planificación, interesa centrarse en los que han sobrevivido al paso de tiempo. Se introduce el concepto latencia de despacho y se ve por qué es necesario reducirla en sistemas operativos de tiempo-real. Se explica el fenómeno de inversión de prioridad y se indica como se solventa a través de un protocolo de herencia de prioridad.

Por otra parte, se muestran las limitaciones del modelo proceso, creado hace varias décadas, cuando trabajamos con multiprocesadores o con aplicaciones inherentemente paralelas. Esto nos permite introducir el concepto de hebra (o hilo) y sus tipos, y cómo el modelo hebra esta diseñado para ser compatible en alto grado con el modelo proceso ya que ambos conviven en la mayoría de los sistemas. Este es uno de los puntos en el que el descenso a nivel de implementación facilita la comprensión del concepto hebra y los diferentes tipos. Aquí es importante poner algunos ejemplos reales para ilustrar las sutilidades de la nueva abstracción. Abundaré sobre esta cuestión en el apartado de metodología.

Hay ejercicios que creo importantes para conocer si han comprendido los conceptos básico del tema. Por ejemplo, ¿puede un SO atender las interrupciones si utiliza una política de planificación no apropiativa?, ¿debe un proceso cambiar él mismo su estado de "ejecutándose" a "bloqueado"? Desde el punto de vista de diseño, casi más importante que el estudio de un determinado algoritmo de planificación es hacerles ver, por ejemplo, cuales son los mecanismos utilizados para disparar la acción del planificador.

3.3 Sincronización y comunicación

En este tema, el planteamiento inicial es introducirles en la necesidad de la programación concurrente para resolver determinado tipo de aplicaciones. Para ello, les muestro un ejemplo sobre el dibujo de un conjunto de Mandelbrot (proceso acotado por computo) en una ventana Windows y pueden observar como, mientras dibujamos, es imposible acceder al menú de opciones de la ventana. A continuación se utiliza el mecanismo de entradas/salidas asíncronas para dibujar con una hebra y atender la ventana con otra, resultando ahora posible acceder al menú mientras se dibuja. El resto del tema se dedica a ver los problemas clásicos de sincronización utilizando semáforos. Esta primitiva, junto con los cerrojos de espera ocupada (*spin lock*), son las únicas primitivas de sincronización que les muestro, ya que el estudio de gran variedad de primitivas genera más confusión que beneficios. En cualquier caso, les hago notar que en las asignaturas de Programación Concurrente, Diseño de Bases de Datos, etc. verán un amplio abanico de primitivas.

3.4 Gestión de memoria

Respecto a la gestión de memoria, los puntos más importantes a mi entender son la necesidad de diferenciar entre espacio lógico de direcciones y espacio físico, y cómo se hace corresponder uno en otro a través de lo que yo denomino la Caja de Traducción (la MMU). En los últimos años, he ido reduciendo el estudio de esquemas simples de asignación de memoria al estudio de particiones variables, con la sola idea de introducir el problema de la fragmentación. Esto permite centrarse más en los esquemas de gestión de memoria utilizados para evitar este problema: segmentación y paginación. Se sigue viendo el intercambio porque aún no ha perdido toda su utilidad . En paginación es importante resaltar que su buen funcionamiento depende de la localidad de los programas, que influye entre otras cosas en el buen funcionamiento del TLB (Búfer de reconocimiento de traducciones). De otra parte, es necesario acudir a la programación en ensamblador, que se estudia en otra asignatura,

para ver las diferencias entre segmentación y paginación, ya que inicialmente les resultan mecanismos muy parecidos.

En gestión de memoria virtual, aparte de los elementos tradicionalmente vistos, hay que hacerles ver lo que no dicen claramente muchos libros de texto: que al sobrecargar el bit de validez de significado para tratar la falta de página, la tabla de páginas no describe el espacio completo de direcciones de un proceso, y que una falta de página no indica necesariamente que una página no está en memoria, sino que la traducción que tenemos no es correcta aún cuando la página puede estar en memoria. Creo necesario explicar el mecanismo de copia-al-escribir (*copy-on-write*) para explicar con posterioridad archivos proyectados en memoria.

Respecto a los algoritmos de sustitución de páginas, cada vez tiendo más a reducir su número, explicando básicamente el algoritmo del reloj y el de Frecuencia de falta de página, que son los más utilizados actualmente. Es importante explicar el Modelo del Conjunto Activo, en especial por su implicaciones con la planificación de proceso a través de Principio de Conjunto Activo.

3.5 Gestión de entradas/salidas

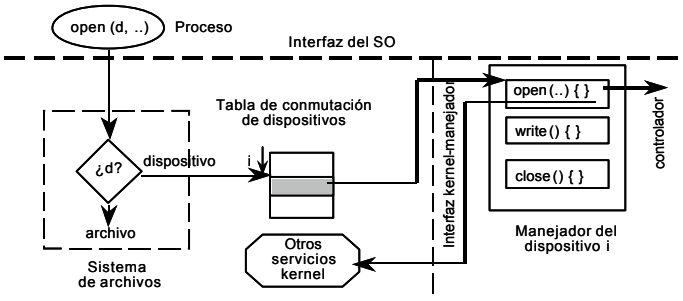
Dado que en asignaturas como Introducción a los Computadores y Estructura de Computadores dedican parte del programa a ver la arquitectura hardware de los dispositivos, este tema tiene como puntos fuertes los siguientes: (a) analizar la arquitectura software del subsistema de entradas/salidas, (b) ver cómo se designan dispositivos haciendo hincapié en su integración en el espacio de nombres de archivos, y (c) estudiar cómo se implementa el concepto de entradas/salidas independientes del dispositivo (Figura 1). Respecto al punto (a), es de especial relevancia el estudio del camino de código para una operación de entrada o salida, que permite visualizar un ejemplo real del problema del productor-consumidor, y nos lleva a plantear la necesidad de diseñar un manejador de dispositivos dividido en dos mitades. La mitad superior se ejecuta en el contexto del proceso llamador, pudiéndose bloquear, y la mitad inferior, que no se puede bloquear, ha de ejecutarse en un contexto diferente.

3.6 Sistemas de archivos

Por último, dado que en Sistemas Operativos II existe un tema dedicado a la implementación de sistemas de archivos, en la asignatura nos dedicamos más a ver las abstracciones suministradas al usuario, como son archivos, directorios, descriptores de archivos, enlaces duros y simbólicos. También les introduzco los temas de las posibles semánticas de consistencia, designación de archivos, y el concepto de archivo proyectado en memoria, por su extendido uso hoy en día y como ejemplo de interrelación entre los subsistemas de memoria y de archivos.

Una visión general de la interfaz de sistema de archivos virtuales les permite comprender cómo en los sistemas actuales es posible manejar simultáneamente diferentes tipos de sistemas de archivos. Así, podemos mostrar cómo este mecanismo permite integrar abstracciones diferentes con la apariencia de archivos, me refiero, por ejemplo, al sistema de archivos de dispositivo, al sistema de archivos /proc, etc.

Figura 2.- Implementación de la independencia del dispositivo.



3.7 Otros puntos tratados

Si bien el tema de eficiencia recogido en el currículum de la ACM/IEEE no se trata expresamente en la asignatura, creo que es conveniente hacer hincapié a los estudiantes en la importancia de un buen rendimiento en el SO. En este sentido, trato de ir dando siempre valores de sistemas reales para el coste de ejecución de funciones como cambio de contexto, cambio de modo, coste de creación de un proceso y de una hebra, tiempo de acceso efectivo, etc. ya que en muchos casos estos valores son los que justifican que se elija entre una de varias alternativas en el diseño e implementación de una función del SO.

También, dejo a mis alumnos algunos temas de libre estudio tratando de enlazar con asignaturas que se cursan con posterioridad como es, por ejemplo, el punto de contacto entre la generación de código del compilador y la gestión de memoria del sistema operativo. Para ello, he elaborado dos temas denominados *Asignación de memoria y enlazadores*, y *Designación y memoria virtual*.

Aparte del temario, he comprobado eficiente y motivador aplicar la idea de T. Anderson en [2] de impartir de una serie de "sermones" (charlas de 15 min.) que, no siendo estrictamente claves en la asignatura, provocan la reflexión de los alumnos sobre determinados temas como son: *La información es propiedad*, *La sencillez*, *Mantenerse abierto*, y *Construcción de sistemas eficientes*.

En otro ámbito diferente, la no existencia de un vocabulario común me lleva a explicar a mis alumnos los términos en inglés y las traducciones más comunes de los mismos así como las que yo empleo, ya que los diferentes libros de texto utilizan en algunos casos traducciones diferentes para los mismos términos. Otras veces, las traducciones son de uso infrecuente o erróneo en España, ya que están pensadas para el mercado latinoamericano, etc.

4 Metodología

Como se indica en [9,12], la mera transferencia de conocimiento suministrada por la clase magistral debe dejar paso a un proceso de aprendizaje individual basado en la motivación. Por ejemplo, emplear parte de una clase en enseñar determinado algoritmo de planificación, que es fácil de aprender en un libro de texto, es menos productivo y motivador que emplear este tiempo en hacer ver al estudiante cómo se activa en algoritmo de planificación con la ocurrencia de una interrupción, por citar un ejemplo. Por ello, procuro dedicar más tiempo a cubrir aspectos fundamentales de diseño, que no quedan claros en la bibliografía, e inducir al estudiante al razonamiento e interconexión de conceptos, que a explicar detalles o ver variantes de un tema que pueden leer en la bibliografía, y que, no aportando gran contenido, requieren un tiempo precioso. Por ejemplo, muchos libros de texto emplean gran número de páginas a explicar las diferentes variantes y aproximación del algoritmo LRU para sustitución de página aunque en la vida real se aplican pocas de ellas, y sin embargo, obvian la necesidad de introducir un proceso/hebra de fondo para realizar dicha sustitución.

La estructura de cada una de las unidades sigue siempre un mismo esquema: (1) se plantea un problema, (2) se estudian diferentes alternativas de diseño propuestas, y (3) se analizan las ventajas y desventajas de las soluciones vistas dejando clara la métrica de evaluación, por ejemplo, rendimiento de las soluciones, consumo de recursos, flexibilidad, etc.

La naturaleza del material cubierto en la asignatura obliga a abordar un enfoque práctico en el que es aconsejable utilizar técnicas de visualización para presentar el material en un formato intuitivo, ver [13]. Estas técnicas exigen un gran esfuerzo en adaptar el material de curso a un formato gráfico, si bien se adaptan bien a diferentes metodologías como enseñanza a distancia mediante Web. Por ejemplo, la Figura 2 ilustra los pasos seguidos para acceder a un dispositivo utilizando la técnica de independencia de los dispositivos de E/S.

En un línea similar, un aspecto importante de las nuevas tecnologías, que he constatado motivador e ilustrativo, es el uso del ordenador en clase. Este nos puede servir para ilustrar los conceptos explicados. Por ejemplo, la programación "real" de ejercicios, como pueden ser los de sincronización con semáforos, y su ejecución en clase permite entre otras cosas: ver cómo realmente se produce el interbloqueo en los filósofos comensales, Figuras 3 y 4.

```
#define NFILOSOFOS 5
HANDLE hMutexTenedor[NFILOSOFOS];
...
void main() {
HANDLE hVectorHebras[NFILOSOFOS];
DWORD IDHebra;
int i;
for (i=0; i<NFILOSOFOS; i++)
hMutexTenedor[i]=CreateMutex(NULL, FALSE,NULL);
for (i=0; i<NFILOSOFOS; i++)
  hVectorHebras[i] = CreateThread (NULL, 0,
(LPTHREAD_START_ROUTINE)Filosofo,(LPVOID)i,0, (LPDWORD)&IDHebra);
WaitForMultipleObjects(NFILOSOFOS,hVectorHebras,TRUE,INFINITE);
}

VOID Filosofo(LPVOID id) {
int izqdo = (int) id;
int dercho = (izqdo + 1) % NFILOSOFOS;
while(1){
  WaitForSingleObject(hMutexTenedor[izqdo],INFINITE);
  printf("Filosofo %d: coge tenedor izqdo.\n",(int) id);
  WaitForSingleObject(hMutexTenedor[dercho],INFINITE);
  printf("Filosofo %d: coge tenedor dcho. y come\n",(int) id);
  ReleaseMutex(hMutexTenedor[izquierdo]);
  ReleaseMutex(hMutexTenedor[derecho]);
  printf("Filosofo %d: libera tenedores y piensa\n",(int) id);
 }
}
```

Figura 3.- Solución simple del problemas de los filósofos-comensales.

Otra técnica que se muestra de gran utilidad para comprender los conceptos y mecanismos vistos en clase, es la de formar grupos de trabajo con el objetivo de dar respuesta a una pregunta formulada al hilo de lo explicado. Por ejemplo, una de las diferentes preguntas formuladas en el presente curso estaba encaminada a que los alumnos abordasen el dilema de cómo es posible apropiar un proceso que se ejecuta en modo usuario teniendo en cuenta que la operación cambio de contexto se realiza en modo kernel y este modo es en muchos sistemas no apropiativo. En estas cuestiones, no importa tanto el que alcancen una respuesta correcta al 100% como que razonen sobre el funcionamiento del sistema.

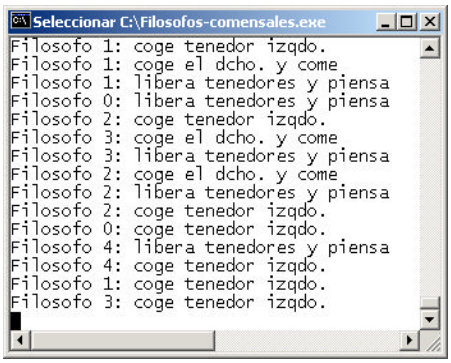


Figura 4.- Interbloqueo en los filósofos-comensales.

5 Evaluación

Si bien no he evaluado formalmente el curso, he podido observar en los alumnos un creciente interés por la asignatura. Muchos de los que han visitado la página Web, me han indicado la utilidad de sus contenidos.

Parte de la motivación encontrada en clase obedece a la actualidad de los ejemplos vistos que permite relacionar a los estudiantes los conceptos vistos en clase con los sistemas operativos usados diariamente, ya sea en su PC, PAD, teléfono móvil, electrodomésticos, etc.

6 Conclusiones

Se han mostrado a grandes rasgos cuales deben ser, a mi entender, los contenidos de una asignatura de introducción a los sistemas operativos. Considero que estos contenidos son necesarios para obtener una visión general del diseño de sistemas que trascienda a los propios contenidos de un curso de esta disciplina.

La propuesta realizada tiene aún algunos inconvenientes y cuestiones sin resolver en su totalidad. A la vista del resurgimiento de la investigación en este campo, que por algunos años parecía aletargada, pretende ser un estímulo para reflexionar sobre la docencia de sistemas operativos en este nuevo milenio

En http://lsi.ugr.es/~jagomez/sisopi.html se puede ver parte del material docente usado para la asignatura Sistemas Operativos I y generado según el enfoque y contenidos propuestos. No obstante, debe tenerse presente que aún queda trabajo por hacer para actualizar algunos de los temas que aparecen en la página con respecto a los principios propuestos en este trabajo.

He omitido intencionadamente a lo largo de la discusión el tratamiento dado a las prácticas de la asignatura debido a que están en fase de remodelación. El objetivo de las mismas será profundizar en el uso de Linux tanto a nivel de usuario, utilizando el shell y ordenes avanzadas, como de programación incluyendo una parte dedicada a la programación con hebras para la mejor comprensión de los conceptos vistos en el tema de sincronización. En este último aspecto, creo que son buenos los enfoques dado en ciertos trabajos aparecidos en el SIGCSE que utilizan la interfaz de llamadas para reforzar los conceptos vistos en teoría.

Referencias

1. ACM/IEEE, *Computing Curricula 2001 Computer Science*, Final Report, December 15, 2001,
2. T. Anderson, http://http.cs.berkeley.edu/~tea.
3. J. Bacon, *Concurrent Systems. Operating Systems, Database, and Distributed Systems: An Integrated Approach*, 2ª Ed. Addison Wesley, 1998.
4. D. P. Bovet y Marco Cesati, "A Real Bottom-Up Operating Systems Course", *Operating Systems Review, 35(1),* pgs. 48-60, ACM Press, Jan. 2001.
5. P. Bucci, T.J. Long, y B. W. Weide, "Do We Really Teach Abstraction?, *Proc. of the 32nd SIGCSE Technical Symposium on Computer Science Education*, pgs. 26-30, 2001.
6. J. Carretero Pérez, y otros, *Sistemas operativos. Una visión aplicada*. McGraw-Hill, 2001.
7. R. A. Creak y R. Sheehan, "A Top-Down Operating Systems Course", *Operating Systems Review, 34(3),* pgs. 69-80, ACM Press, July 2000.
8. C. Crowley, *Operating Systems. A Design-Oriented Approach*, Irwin, 1997.
9. M. García-Valero y J. J. Navarro, "Niveles de competencia de los objetivos formativos en las Ingenierías, *VII Jornadas de enseñanza universitaria de la informática*, pgs. 149-154, Palma de Mallorca, 16-18 de 2001.
10. M. A. Holliday, "System Calls and Interruptor Vectors in an Operating Systems Course", *Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education*, pgs. 53-57, 1997.
11. D. Milojicic, "Operating systems - now and in the future". *IEEE Concurrency*, January-March 1999, pgs. 12-21.
12. M. Rebollo, "Aprendizaje activo en el aula", *VII Jornadas de enseñanza universitaria de la informática*, pgs. 137-142, 2001.
13. V. Sparks, y Shan Suthaharan, "Operating Systems: Visualization Technique for Teaching and Learning", *Proceedings of the IEEE*, Southeastcon, 2000, pgs. 387-390.