ADJUNTA AL RECTOR PARA LA CONVERGENCIA EUROPEA

TITULO DEL PROYECTO:

Sistemas Operativos I en Ingeniería Informática (parte 2)

INFORME FINAL

Proyectos-Piloto de Adaptación de las Titulaciones al EEES
UNIVERSIDAD DE ZARAGOZA

Julio 2006

Informe Final

1.1 Descripción general y cronológica del trabajo realizado

El objetivo principal de este Proyecto ha sido el de elaborar la Memoria Docente para la asignatura de Sistemas Operativos I en la Titulación de Ingeniería Informática de la Universidad de Zaragoza. El diseño metodológico utilizado intenta adaptarse a las necesidades que plantea la convergencia al Espacio Europeo de Educación Superior, orientando toda la Memoria hacia una enseñanza que sea más constructiva y menos conductiva.

Desde la concesión del Proyecto en Marzo del 2006, las actividades que se han llevado a cabo se pueden resumir en dos. La primera ha consistido en analizar y extender aquella parte de la información obtenida pero no incluida en la anterior experiencia de innovación docente, para añadirla en este Proyecto, continuación del anterior. En concreto, son detalles extraídos de las reuniones que en su día se realizaron con profesores que imparten asignaturas similares a ésta pero en otras Universidades y que han permitido completar muchos de los apartados teóricos, prácticos y metodológicos contenidos en la nueva Memoria. Esta primera tarea se realizó desde la concesión del Proyecto hasta finales del mes de Mayo. La segunda actividad se ha llevado a cabo entre los meses de Junio y Julio y ha consistido en la elaboración de éste Informe Final.

1.2 Análisis de las competencias generales de la titulación y específicas de la asignatura en la que se desarrolla la experiencia. Directrices metodológicas

Este apartado de la actividad está recogido en el Capítulo 1 de la Memoria Docente. Corresponde al capítulo que lleva por título *Entorno Docente* y enmarca el entorno donde se desarrollan los contenidos de la memoria. Analiza las competencias generales de la Titulación de Ingeniería Informática y las particulares de la asignatura de Sistemas Operativos I en el Plan de Estudios de la Universidad de Zaragoza. Resume también las bases que han sido utilizadas para la propuesta de

contenidos de la asignatura (recomendaciones de la *Joint Task Force* para las Ingenierías y los indicadores del B.O.E).

1.3 Elaboración de la guía docente de la asignatura

Apartado de la actividad desarrollado en el Capítulo 2 de la Memoria Docente. Este Capítulo 2 empieza localizando la asignatura en el Plan de Estudios de Ingeniería Informática del CPS, pasando después a presentar los objetivos principales que se persiguen en la asignatura. Una vez fijados los objetivos, éstos se organizan en un temario detallado y se presenta la bibliografía recomendada. A continuación se muestra el detalle del desarrollo de la parte práctica de la asignatura, tanto en sus clases de problemas como de laboratorio. El Capítulo finaliza proponiendo una planificación temporal para la asignatura, así como su correspondiente sistema de evaluación.

1.4 Metodología docente utilizada

En el Capítulo 3 de la Memoria se describen las bases metodológicas utilizadas en su desarrollo, los tipos de clases, estructura de las clases, recursos y técnicas didácticas y evaluación. El modelo pedagógico base de la Memoria está basado en el aprendizaje y ha sido adaptado a las circunstancias reales en las que se imparte la docencia (entorno docente que se explicó en el Capítulo 1). Todas aquellas técnicas específicas relacionadas con la asignatura se detallan en el mismo Capítulo 2 de la Guía Docente.

1.5 Valoración del proyecto: obstáculos, estrategias, conclusiones, posibilidades de generalización e implantación

Una de las estrategias considerada fundamental para este tipo de proyectos es la de tener la posibilidad de realizar entrevistas con profesores de asignaturas relacionadas dentro de la misma Titulación o de otras Titulaciones dentro de la misma Universidad, o con profesores que imparten asignaturas similares en otras Universidades. Este último tipo de interacciones posee una doble utilidad. Por un lado, es útil para desarrollar contenidos o estrategias docentes comunes basándonos en la experienza adquirida tras su impartición en asignaturas de otros entornos. Por otro lado, esa misma interacción da un valor añadido al desarrollo de la Guía Docente ya que es una buena base que permite la generalización de la misma para implantarla en diversos entornos. Considero que en el futuro se debería primar más la posibilidad de este tipo de contactos. Para la Guía Docente presentada en esta Memoria se ha utilizado material obtenido por este tipo de entrevistas realizadas con el anterior proyecto y que no se incluyeron allí.

Esta Memoria Docente se ha desarrollado tomado como marco de referencia el Plan de Estudios de Ingeniería Infomática de la UZ, los indicadores del BOE y las recomendaciones del IEEE-ACM para la materia de Sistemas Operativos. Creo que esta forma de actuar es muy útil para el desarrollo de cualquier otro proyecto de características similares. La metodología docente utilizada también la considero como un buen modelo pedagógico aplicable a cualquier otra asignatura de la misma titulación o de otras.

Proyectos-Piloto de Adaptación de las Titulaciones al EEES
UNIVERSIDAD DE ZARAGOZA

Sistemas Operativos I en Ingeniería Informática (parte 2)

MEMORIA DOCENTE

Julio 2006

Experiencia de nivel B
Diseño metodológico para una o varias asignaturas de una Titulación

GUÍA Y METODOLOGÍA DOCENTE PARA LA ASIGNATURA SISTEMAS OPERATIVOS I EN LA TITULACIÓN DE INGENIERÍA INFORMÁTICA,

presentado por:

Teresa Monreal Arnal
Area de Arquitectura y Tecnología de Computadores. DIIS

Orden ECI/924/2005, de 21 de Marzo
Ministerio de Educación y Ciencia

Introducción

El objetivo de esta Introducción es el de presentar cual es la necesidad de esta guía docente. Muestra cómo en el nuevo marco en el que se encuentran inmersas las universidades, es necesario adaptar los diseños que se han hecho de las asignaturas proponiendo nuevas estructuras. Se describe brevemente cuáles van a ser los fundamentos que, a través de una visión personal, se consideran necesarios para obtener un tipo de proyecto docente adaptado a las nuevas competencias.

Justificación del Proyecto Docente

La convergencia hacia el Espacio Europeo de Educación Superior, trae consigo la necesidad de realizar tanto un rediseño general de las titulaciones, como una revisión detallada del diseño curricular de las asignaturas. Un buen diseño curricular constituye la base de calidad de todo proceso de innovación y mejora.

Ha sido el incremento en las tecnologías de la información el que ha planteado la necesidad de un cambio radical en la forma en la que hay que gestionar el conocimiento en la sociedad. El problema principal se encuentra en que existe demasiada información y hay que ser capaz de discriminar aquella parte de esa información que más se aleja de lo que representa la realidad. Un buen conocimiento de la realidad debe ser elaborado a base de observar las distintas representaciones que existen de esa realidad y elegir aquella que más se adapta al entorno en el que se encuentra el observador.

Dentro del marco universitario, lo anterior plantea la necesidad de orientar todo el proceso hacia una impartición de conocimiento que sea más constructiva que reproductiva. Se ha de plantear así un tipo de enseñanza menos orientada a la conducción del conocimiento y más orientada a la adquisición del mismo. Será un aprendizaje basado en una enseñanza menos conductiva y más cognitiva.

La obtención del conocimiento se va a conseguir por medio de un aprendizaje orientado a hacer que sean los alumnos los que gestionen su propio conocimiento. Pasa a un segundo plano la búsqueda en el alumno de su capacidad para reproducir contenidos, y viene a ser mucho más importante la búsqueda en él de su capacidad para comprender esos contenidos.

En esta nueva orientación, el alumno es un elemento activo en el proceso de elaboración del conocimiento. Una metodología orientada al aprendizaje hace que al alumno se le vaya transfiriendo de forma progresiva su participación en la toma de decisiones.

Es objetivo principal de este proyecto docente el de elaborar una metodología de aprendizaje orientada a una enseñanza más constructiva que reproductiva. Si en una orientación hacia la enseñanza reproductiva el objeto de trabajo es la materia a impartir, en una orientación hacia la enseñanza constructiva, los objetos de trabajo serán los propios alumnos.

En todo este proyecto docente se pretende una elaboración curricular de calidad fundamentada en todo momento en un modelo personal de enseñanza organizada en función del aprendizaje obtenido por los alumnos. El objeto de trabajo en el que se apoya este diseño curricular va a ser el alumno y la unidad de gestión será la cantidad de aprendizaje adquirido. En particular, la enseñanza del aprendizaje va a dirigir a los alumnos hacia una meta común asegurándoles nuevas competencias. Esa meta común será la de la adquisición y elaboración del conocimiento en el alumno.

La palabra enseñanza irá asociada a la búsqueda de aquellos entornos que sean los más adecuados para llevar a cabo el aprendizaje. Una vez fijado el entorno de trabajo, la forma en la que el alumno va a ir adquiriendo el conocimiento pasa primero por una fase de aprendizaje de las técnicas, para llegar después a la comprensión de las distintas estrategias. Es tarea del profesor la de guiar al alumno para conseguir que éste adquiera conocimiento haciéndole convivir en todo momento con la incertidumbre y con la duda. Es esta incertidumbre la que servirá de provocación al alumno, motivándolo a encontrar las respuestas a las dudas planteadas.

El objetivo final de este modelo de aprendizaje será el de conseguir un cambio en la personalidad del alumno. El aprendizaje debe hacerle adquirir al alumno capacidad para formarse su propio conocimiento. Una vez adquirida esta capacidad, va a representar un cambio en el alumno que además de duradero será transferible al resto de las materias.

1
Entorno Docente

En este capítulo se presenta una propuesta de entorno en el que desarrollar la asignatura objeto de esta memoria docente: Sistemas Operativos en la Titulación de Ingeniería Informática. Se hace primero un breve resumen de los referentes normativos que rigen a la institución sobre la que se presenta la memoria. Se ofrece después una visión general sobre los Planes de Estudios y en particular sobre los de Ingeniería en Informática. Se acaba presentando los principios generales que sigue el plan docente del Centro Politécnico Superior de la U. de Zaragoza y el área de conocimiento a la que está asignada la materia propuesta: Área de Arquitectura y Tecnología de Computadores, enmarcando completamente la asignatura.

1.1 La Universidad en la sociedad del conocimiento

El objetivo de este subapartado es el de enfocar cuál es el entorno institucional y legal en el que se encuentra inmersa la universidad. Será necesario disponer de un conocimiento claro del entorno de trabajo, ya que los contenidos del proyecto docente van a estar condicionados por el conjunto de competencias que pretenden cumplirse.

Los párrafos siguientes intentan resumir cuál es el conjunto de estas competencias, de qué forma se ajustan al proceso de convergencia europeo y cuáles han sido las distintas líneas de actuación decididas en las primeras reuniones realizadas como parte de este proceso de convergencia. El último párrafo resume la actitud de la universidad española sobre el proceso.

1.1.1 El Espacio Europeo de Educación Superior (EEES)

Con el objetivo de coordinar las normas legislativas de los distintos estados que componen la Unión Europea, varios paises empezaron a acordar una serie de medidas orientadas a favorecer el bienestar de los ciudadanos. En particular, fue en el entorno de la enseñanza superior, donde se adoptaron medidas dirigidas a reformar la estructura y organización de sus enseñanzas universi-

tarias. Esta reestructuración, además de coordinar las normas legislativas que los distintos paises poseen en torno a la enseñanza superior, intentará favorecer la construcción del *Espacio Europeo de Educación Superior, EEES*.

La armonización de los diversos sistemas que regulan las enseñanzas universitarias de cada estado miembro, lleva asociada la necesidad de realizar todo el proceso respetando al máximo la diversidad de culturas y la autonomía universitaria. Los objetivos principales que se plantean van en la línea de dar a los estudios una mayor *transparencia* y *comparabilidad* con beneficios para toda la sociedad. Los estudiantes verán organizadas las enseñanzas en función de su aprendizaje, siendo este *aprendizaje* el eje de referencia para el diseño de los futuros Planes de Estudio.

El proceso de construcción de ese Espacio Europeo de Educación Superior, comienza en 1998 con la Declaración de la Sorbona. Es aquí donde se empieza a poner de manifiesto la voluntad de potenciar una Europa del conocimiento de acuerdo con las nuevas tendencias y de favorecer la comparabilidad, reconocimiento y movilidad de estudiantes y titulados. Se observa cómo tanto la extensión como la calidad de la educación superior son factores decisivos en el incremento de la calidad de vida de los ciudadanos. Un año más tarde (1999), el proceso iniciado se consolida y amplia con la Declaración de Bolonia. Es en esta declaración donde se establece un horizonte temporal para la plena consecución del EEES hasta el año 2010. El proceso continúa con el Comunicado de Praga (2001) donde se añaden algunas líneas adicionales.

Destacar de las anteriores actuaciones, primero la consideración del aprendizaje como elemento esencial para alcanzar una mayor competitividad europea, mejorar la cohesión social, igualdad de oportunidades y calidad de vida. Segundo, el diseño de una nueva estructura para los estudios universitarios, mediante la adopción de un sistema de titulaciones de calidad, comprensible, de 2 niveles (*Grado* y *Posgrado*) y la recomendación del establecimiento de un *sistema común de créditos*. La introducción del Sistema Europeo de Transferencia de Créditos (*European Credit Transfer System, ECTS*) en las titulaciones oficiales de Grado y Posgrado, proporcionará una mayor transparencia y homogeneidad a la hora de comparar cursos, materias y calificaciones. Será la unidad de medida del haber académico donde reflejar el esfuerzo real requerido por el estudiante para cumplir los objetivos del programa de estudios. En esta unidad se integrarán el número de horas dedicadas a las clases lectivas, teóricas o prácticas, horas de estudio, las dedicadas a la realización de seminarios, trabajos, prácticas o proyectos y las exigidas para la preparación y realización de los exámenes y pruebas de evaluación. El número mínimo de horas por crédito será de 25 y el máximo de 30 (ref. BOE del 18 de Septiembre de 2003).

Posteriormente, en la Cumbre de Jefes de Estado celebrada en Barcelona (Marzo del 2002) se produce un avance considerable en el proceso de construcción del Espacio Europeo de Educación Superior. La cumbre finaliza aprobando un programa de trabajo con el principal objetivo de crear las condiciones prácticas necesarias para garantizar transparencia y cooperación en materia universitaria. Este avance continúa hasta la reunión de ministros de educación que tiene lugar un año después en Berlín (19 de Septiembre del 2003).

El objetivo principal de la reunión de Berlín fue el de analizar tanto el progreso efectuado por los distintos países en materia de educación, como el de fijar nuevas prioridades y objetivos para los años siguientes. Con la idea de acelerar la realización del área de la Educación Superior Europea, los ministros acuerdan diversas actuaciones enmarcadas bajo los puntos del progreso, garantía de

calidad, estructura-calificaciones-accesibilidad, movilidad y sistema de créditos, reconocimiento, educación-investigación, instituciones y seguimiento.

Los ministros acaban pidiendo al grupo de seguimiento que coordinen las actividades para el progreso del proceso de Bolonia y que les informen en tiempo para el siguiente encuentro ministerial. Deciden mantener la próxima conferencia, la IV Conferencia del Proceso de Bolonia, en Mayo del 2005 en Bergen (Noruega).

Los 45 países europeos que actualmente forman parte del proceso de Bolonia, acaban esta cuarta conferencia firmando el Comunicado de Bergen para subrayar los avances conseguidos hasta el momento y señalar los desafíos futuros. Destacan como progresos prioritarios la implantación de un sistema comparable, reconocimiento de títulos y períodos de estudio en diferentes países y el establecimiento de garantías de calidad para los estudios. Entre las prioridades futuras destacan la necesidad de desarrollar más y mejores vínculos de la universidad con la investigación e innovación y de favorecer la movilidad tanto de estudiantes como de profesorado. Se renueva el compromiso entre todos los países de conseguir una educación superior de calidad y accesible a todos (sin ningún obstáculo económico y/o social). Se aboga para ello por medidas de apoyo financiero a los estudiantes, de atención y de tutela, así como el incremento de programas de becas, todo orientado a ampliar el acceso a la universidad. Cada país deberá presentar su nuevo proyecto de estudios en la próxima conferencia ministerial que deciden mantener en 2007 en Londres (Reino Unido).

1.1.2 La universidad española en el proceso de convergencia europeo

La integración del sistema universitario español en el EEES requiere de una serie de propuestas concretas para los distintos elementos definidos en las declaraciones europeas.

La universidad española, es consciente de la necesidad de una nueva ordenación de la actividad universitaria y así lo recoge en la exposición de motivos de la Ley Orgánica de Universidades (LOU). Ve la necesidad de abordar los retos derivados de la innovación en las formas de generación y transmisión de conocimiento. El desarrollo de la sociedad del conocimiento va a precisar de estructuras organizativas flexibles. Nuevas estructuras que faciliten un amplio acceso social al conocimiento y una capacitación personal crítica para favorecer tanto la interpretación de la información, como la generación del propio conocimiento.

El carácter universal de la institución universitaria, se verá aún más incrementado, no solo con la utilización de las nuevas tecnologías de la comunicación, sino con una creciente movilidad de profesores, investigadores y alumnos. Mantener el objetivo de la movilidad en toda Europa significa fomentar un acercamiento cultural y académico encaminado hacia un mundo de mayores ventajas laborales y profesionales.

La universidad española acata así las peticiones derivadas de los acuerdos de Bolonia y sucesivos y dedica esfuerzo a sintonizar en la convergencia europea a todos los niveles. Considera necesario llegar a la consecución de todos los objetivos mencionados en la exposición de motivos de la LOU y lo hace visible a través de un *Documento Marco*. Por su parte, la Agencia Nacional de Evaluación de la Calidad y Acreditación, ANECA, también demanda respuestas a los acuerdos

europeos, lo cual está provocando la generación de los *Libros Blancos* por titulaciones (para Ingeniería en Informática, proyecto de *Estudios de Informática y Convergencia Europea, EICE*).

Documento Marco y proyecto EICE

El Ministerio de Educación, Cultura y Deporte comparte todos los objetivos fijados en el proceso de convergencia y asume la responsabilidad de promover y llevar a cabo las modificaciones estructurales de los estudios universitarios necesarias para alcanzar la plena integración del sistema español en el espacio europeo de la enseñanza superior. Es una previsión del Título XIII de la LOU 6/2001, 21 de Diciembre.

El Ministerio remite al Consejo de Coordinación Universitaria, máximo órgano consultivo y de coordinación del sistema universitario, haciéndolo público, un *Documento Marco*. Este documento es el que contiene propuestas orientadas tanto a servir de punto de partida para la reflexión que debe producirse en las universidades y administraciones educativas, como a posibilitar los acuerdos sobre los principales aspectos del proceso de integración y que deberán orientar las normas jurídicas que se promulguen.

Por otro lado, bajo el proyecto EICE de la ANECA, en Marzo de 2004 se presenta el *Libro Blanco del Título de Grado en Ingeniería Informática* para responder a las solicitudes europeas. Estudio en el que han participado un total de 56 universidades españolas y 75 centros docentes, donde se recogen las principales propuestas de diseño que servirán como base para la definición de una *única* titulación de Grado en Ingeniería Informática con 3 perfiles profesionales: el perfil de Desarrollo del Software, el de Sistemas y el de Gestión y Explotación de Tecnologías de la Información. Destacaré los resultados relacionados con la estructura, organización y contenidos recomendados para la formación en Ingeniería Informática. Se propone organizar los estudios en un primer ciclo, *Grado* de 4 años de duración y 240 créditos ECTS, de contenidos generalistas y que habilita a la obtención del grado de ingeniero. Un segundo ciclo, *Master* de entre 60 y 120 créditos ECTS, destinado a la especialización profesional de los ingenieros o a su preparación para la investigación. Este segundo ciclo, puede incluir la realización de una tesis de Master y permitirá el acceso a la realización de la tesis doctoral para obtener el grado de Doctor.

1.2 Los Planes de Estudio de las Universidades Españolas

Un Plan de Estudios es el conjunto de enseñanzas organizadas por una Universidad cuya superación da derecho a la obtención de un título.

Es el Consejo de Universidades, constituido en 1985, el organismo al cual se le atribuye la competencia de proponer y establecer las directrices generales comunes a los Planes de Estudio que deberán cursarse para la obtención y homologación de los distintos títulos universitarios de carácter oficial y validez en todo el territorio nacional.

Con el objetivo de acercar la formación universitaria a la realidad social y profesional de nuestro entorno, se plantea la necesidad de realizar una nueva ordenación académica de las enseñanzas universitarias. Será una nueva estructuración académica que permita a las universidades ofertar

un conjunto coherente de titulaciones académicas que den respuesta a las nuevas demandas del mercado de trabajo.

Uno de los ejes fundamentales sobre los que se desarrollan las directrices generales comunes a los Planes de Estudios conducentes a la obtención de títulos oficiales, propone una *ordenación cíclica* de las carreras que permita una mayor rentabilidad de la oferta universitaria. Con un primer ciclo que comprenda enseñanzas básicas y de formación general y de 2 ó 3 años académicos de duración, y un segundo ciclo dedicado a la profundización y especialización en las correspondientes enseñanzas (duración de 2 años académicos y excepcionalmente 3). En este contexto, también se persiguen otros fines relacionados con conseguir una mayor racionalización en la duración de las carreras, así como una mayor relevancia de las enseñanzas prácticas. Además se incorporará en el sistema el cómputo del haber académico por *créditos*, para potenciar una mayor apertura de los Planes de Estudio y una mayor flexibilidad al currículum del estudiante.

El segundo eje fundamental de las directrices generales comunes a los Planes de Estudio, propone una ordenación de contenidos que permita conciliar el principio constitucional de libertad académica y coherencia formativa. A este respecto y como directriz general común a todos los Planes de Estudio, es obligatorio distinguir entre tres bloques de contenidos: troncales, no troncales y de libre elección.

Todas las materias de un Plan de Estudios se encuentran vinculadas a una o varias áreas de conocimiento. Una vinculación determinada de área para una materia troncal, hará que dicha materia sólo pueda ser impartida por profesores que pertenecen al área de conocimiento. Además del área al que está vinculada toda materia, en el correspondiente Plan de Estudios deberá aparecer una breve descripción de su contenido, sus créditos (especificando si son de enseñanza teórica o práctica) y si es el caso, su ordenación temporal fijando secuencias entre materias o conjunto de ellas.

1.3 Los estudios de Ingeniería Informática

A modo de introducción, se va a mostrar primero un resumen de las recomendaciones que la *Joint Task Force* ofrece sobre cual debe ser el contenido de los currícula en ingeniería y en particular en Ingeniería Informática.

1.3.1 Recomendaciones de la *Joint Task Force* para las ingenierías

Fue a finales de 1998 cuando se estableció la *Joint Task Force*, asociación formada por miembros del *IEEE-CS* (*Computer Society of the Institute for Electrical and Electronics Engineers)* y del *ACM (Association for Computing Machinery)*. El objetivo de esta asociación fue el de revisar el anterior *Computing Curricula* desarrollado en 1991 para actualizarlo y establecer un nuevo modelo de contenidos para los distintos programas en ingeniería. A continuación se resumen cuáles fueron las principales sugerencias dadas sobre cuáles deben ser los contenidos de los currícula para los programas de *Computer Engineering*. Todo lo que viene a continuación se ha extraído del borrador del *report* que lleva por título *Computing Curricula: Computer Engineering (CC-CE)* y que data del 8 de Junio del 2004.

La ingeniería de computadores incluye tanto la ciencia como la tecnología del diseño, construcción, implementación y mantenimiento de componentes software *y* hardware *de los sistemas computadores modernos.*

Esa definición muestra cómo esta ingeniería debe poseer bases sólidas que van desde las teorías

Tabla 1.1 Areas de conocimiento básico en *CCCE*.

Area de conocimiento	Identificador	Horas
Algoritms and Complexity	*CE-ALG*	30
Computer Architecture and Organization	*CE-CAO*	63
Computer Systems Engineering	*CE-CSE*	18
Cirtuits and Signals	*CE-CSG*	43
Database Systems	*CE-DBS*	5
Digital Logic	*CE-DIG*	57
Digital Signal Processing	*CE-DSP*	17
Electronics	*CE-ELE*	40
Embedded Systems	*CE-ESY*	20
Human-Computer Interaction	*CE-HCI*	8
Computer Networks	*CE-NWK*	21
Operating Systems	*CE-OPS*	20
Programming Fundamentals	*CE-PRF*	39
Social and Professional Issues	*CE-SPR*	16
Software Engineering	*CE-SWE*	13
VLSI Design and Fabrication	*CE-VLS*	10
Discrete Structures	*CE-DSC*	33
Probability and Statistics	*CE-PRS*	33

y principios computacionales hasta la ciencia y la ingeniería, pasando por las matemáticas. Todas estas teorías y principios serán aplicados para resolver problemas técnicos que aparezcan en el diseño de componentes *hardware*, *software*, redes y procesos. Por esta razón, los ingenieros deberán adquirir conocimientos tanto de la ciencia de los computadores y de la ingeniería eléctrica, como de las matemáticas y las ciencias relacionadas.

Tres son las características básicas que debe cumplir todo ingeniero en computadores y que lo diferenciarán de otro tipo de ingeniero: capacidad de diseñar sistemas computadores (*hardware* y *software*), conocimiento extenso en matemáticas y ciencias asociadas a la ingeniería, adquisición y preparación para la práctica profesional en ingeniería.

Todo programa en ingeniería de computadores debe así incluir, no sólo el conocimiento básico dentro del campo, sino también la capacidad para aplicar esos conocimientos en la resolución de problemas reales. Con esto, la *Joint Task Force* concluye cuáles son las áreas necesarias para cubrir el conocimiento que debe adquirirse en todo programa de ingeniería de computadores. La

Tabla 1.1 lista un conjunto de 18 áreas, de las cuáles 16 se relacionan directamente con la ingeniería de computadores y las otras 2 con las matemáticas.

A su vez, la *Joint Task Force*, también dice cómo debe ser la estructura de conocimiento a seguir. Será una estructura jerárquica organizada en 3 niveles: área de conocimiento (p.e *CE-OPS: Computing Engineering-Operating Systems)*, unidad de conocimiento (p.e. *CE-OPS7:* unidad de conocimiento *File Systems* dentro del área de conocimiento de *CE-OPS)* y conjunto de temas. Este último es el nivel más bajo de la jerarquía y subdivide cada unidad de conocimiento con los distintos objetivos de aprendizaje dentro de la unidad. Como ejemplo, la Tabla 1.2 muestra las unidades de conocimiento recomendadas para el área de *Operating Systems*.

Tabla 1.2 Unidades de conocimiento básico para el área de Sistemas Operativos en *CCCE*.

Identificador	Unidad	Tipo
CE-OPS0	*History and overview*	*core*
CE-OPS1	*Design principles*	*core*
CE-OPS2	*Concurrency*	*core*
CE-OPS3	*Scheduling and dispatch*	*core*
CE-OPS4	*Memory management*	*core*
CE-OPS5	*Device management*	*elective*
CE-OPS6	*Security and protection*	*elective*
CE-OPS7	*File Systems*	*elective*
CE-OPS8	*System performance evaluation*	*elective*

La siguiente recomendación dice que lo más apropiado para cualquier especificación de un curso, consiste en incluir tanto el conocimiento como el saber resultado que se pretende. Los *objetivos de aprendizaje* deberán enfocar el tipo de competencias que se pretenden adquirir (habilidades y destrezas en la aplicación del conocimiento para resolver un problema) y es recomendable no superar a cuatro o cinco objetivos por unidad ó modulo de conocimiento.

Otra recomendación apunta como objetivo básico a un contenido curricular aquel donde el cuerpo de conocimiento sea lo más pequeño posible. Para ello, la *Joint Task Force* incluye en su *report* cuál debe ser el material mínimo de contenidos de las unidades de conocimiento que considera esencial para cualquier grado de ingeniería de computadores, sin especificar en qué nivel del curso debe aparecer. En el resumen de la Tabla 1.1 se ha mostrado el mínimo número de tiempo recomendado para cada unidad de conocimiento. El total para conocimientos de ingeniería de computadores es de 420 horas, más un total de 66 horas dedicados a matemáticas. Asumiendo un semestre típico de 15 semanas, lo anterior se correspondería con un programa de ingeniería típico de 4 años.

Adicionalmente, la *Joint Task Force* hace hincapié en lo esencial de la experiencia del estudiante en los laboratorios, como refuerzo de los conceptos aprendidos dentro y fuera de la clase. Estas

actividades deben cubrir desde conocimientos introductorios (demostrando comportamientos específicos, experimentando con medidas y estudiando determinadas características), hasta intermedios y avanzados (incluyendo problemas de diseño e implementación de soluciones, adquisición de datos o medidas de determinadas características).

Finalmente, todo currículum de ingeniería debe tener como parte integral aquella que incluya el desarrollo de un proyecto final de diseño. Con ello se pretende la demostración de que el estudiante es capaz de integrar conceptos que provienen de diferentes materias y disciplinas dentro de una única solución, demostrando con ello su creatividad e innovación. El proyecto producirá un documento detallado y bien escrito del diseño realizado, donde también se demostrará la capacidad del estudiante para gestionar y planificar su tiempo.

La última consideración que hace la *Joint Task Force* ve esencial que, independientemente de la estructura organizativa de las distintas instituciones, todo programa de ingeniería de computadores posea un conjunto base de profesorado de adecuado tamaño y competencia. Sugiere así que la mayoría de los cursos técnicos del programa sean impartidos por profesores de las áreas relacionadas con la ciencia de los computadores, la ingeniería eléctrica ó las físicas.

1.3.2 El Plan de Estudios de Ingeniero en Informática

En Octubre de 1990, a propuesta del Consejo de Universidades y previa deliberación del Consejo de Ministros, queda establecido el *Título universitario oficial de Ingeniero en Informática*.

Asociadas al título de Ingeniero en Informática, van las directrices generales propias de los Planes de Estudio conducentes a la obtención del mismo. Según estas directrices, las enseñanzas en Ingeniería Informática deberán proporcionar a los estudiantes una formación adecuada tanto en las bases teóricas como en las tecnologías propias de esta ingeniería. Para ello y siguiendo la estructuración cíclica que exigen esas mismas directrices, las enseñanzas de Informática se articulan en un primer y un segundo ciclo, con una duración total de entre cuatro y cinco años, siendo de al menos dos años por ciclo. Será cada Plan de Estudios asociado a las enseñanzas de Informática, el que determine en créditos su carga lectiva global, la cual no debe ser inferior a 300 créditos.

Por otro lado y respecto de la ordenación de contenidos que las directrices exigen, cada Plan de Estudios de Informática deberá incluir obligatoriamente el conjunto de materias troncales seleccionado para garantizar la necesaria coherencia y homogeneidad del modelo universitario. La Tabla 1.3 muestra un cuadro resumen de las materias troncales que tiene asignadas el Area de conocimiento de Arquitectura y Tecnología de Computadores, con una breve descripción de sus contenidos y el total de créditos que les corresponden (ref. BOE del 20 de Noviembre de 1990).

Este proyecto docente propone el programa para la asignatura de Sistemas Operativos I, la cual se corresponde directamente con la materia troncal Sistemas Operativos asignada al Area de Arquitectura y Tecnología de Computadores en la titulación de Ingeniería Informática.

La asignatura de Sistemas Operativos I, de la misma forma que se describe en la tabla anterior, es materia troncal en el Plan de Estudios del título de Ingeniero en Informática que se imparte en el Centro Politécnico Superior (CPS) de la Universidad de Zaragoza (ref. BOE del 1 de Febrero de

Tabla 1.3 Descriptores y créditos totales por materia troncal asignados al Area de Arquitectura y Tecnología de Computadores para la Titulación de Ingeniería Informática.

Materia Troncal (1er. ciclo)	**Créditos**	**Descriptores**
Estructura y Tecnología de Computadores	15	Unidades funcionales: memoria, procesador, periferia, lenguajes máquina y ensamblador, esquema de funcionamiento. Electrónica. Sistemas Digitales. Periféricos.
Sistemas Operativos	6	Organización, estructura y servicios de los sistemas operativos. Gestión y administración de memoria y de procesos. Gestión de entrada/salida. Sistemas de ficheros.
Materia Troncal (2o. ciclo)	**Créditos**	**Descriptores**
Arquitectura e Ingeniería de Computadores	9	Arquitecturas paralelas. Arquitecturas orientadas a aplicaciones y lenguajes.
Redes	9	Arquitectura de redes. Comunicaciones.
Sistemas Informáticos	15	Metodología de análisis. Configuración, diseño, gestión y evaluación de sistemas informáticos. Entornos de sistemas informáticos. Tecnologías avanzadas de sistemas de información, bases de datos y sistemas operativos. Proyectos de sistemas informáticos.

1995). Este Plan de Estudios, cuyos principios generales expongo a continuación de forma breve, es el que utilizaré como marco docente para la propuesta de contenidos de la asignatura.

Plan de Estudios de Ingeniero en Informática en el CPS

Un plan de estudios es un instrumento para que los estudiantes puedan lograr un perfil profesional a la par que desarrollan sus cualidades humanas.

En Septiembre de 1994, la Universidad de Zaragoza establece la resolución por la cual se hacen públicos los planes de estudios conducentes a la obtención del título de Ingeniero en Informática a impartir en el Centro Politécnico Superior.

En la actualidad, en el CPS se imparten cuatro titulaciones: Ingeniero Industrial, Ingeniero de Telecomunicación, Ingeniero en Informática e Ingeniero Químico. Todos los planes de estudios están estructurados en dos ciclos de cinco cuatrimestres cada uno. Cada cuatrimestre (Otoño, Primavera) consta de unas catorce semanas efectivas de periodo lectivo. La Tabla 1.4 muestra la distribución de los créditos a cursar en cada una de las titulaciones.

El primer ciclo de toda titulación tiene como objetivo el de proporcionar una formación científico-técnica básica de cada ingeniería. Es en el segundo ciclo donde se le ofrece al alumno la posi-

bilidad de especialización, pudiendo optar por una formación diversificada o centrarse en una determinada especialidad.

La Tabla 1.4 muestra cómo la carga lectiva global para la titulación de Ingeniero en Informática es de 351.5 créditos. De ellos, 178.5 créditos corresponden al primer ciclo y 173 al segundo. En la siguiente Tabla 1.5 se ve cómo es esta distribución global en créditos cursados por cuatrimestre.

Cada asignatura de primer ciclo está asignada a un cuatrimestre determinado y las asignaturas de los dos primeros cuatrimestres son prerrequisitos[1] estrictos de todas las demás. El segundo ciclo comienza con asignaturas asignadas al sexto cuatrimestre, las cuales son de matrícula obligatoria. El resto de asignaturas podrán cursarse libremente. Aunque el Centro no establece ninguna restricción formal a la hora de elegir asignaturas de un curso, existen algunas restricciones relacionadas con el sistema de evaluación, de las cuales las más relevantes se detallan en el siguiente apartado.

Tabla 1.4 Distribución de créditos por tipo y Titulación en el CPS

Asignaturas/materias	**Ing. Industrial**	**Ing. Telecom.**	**Ing. Inform.**	**Ing. Química**
Troncales u obligatorias	276.0	267.0	231.0	271.5
Optativas de 1er Ciclo	9.0	6.0	12.0	0.0
Libre elección adscrita a 1er Ciclo	15.0	12.0	18.0	18.0
Optativas 2º Ciclo	33.0	34.5	49.5	42.0
Libre elección adscrita a 2º Ciclo	22.5	25.5	18.0	19.5
Optativa no técnica	3.0	3.0	3.0	3.0
Proyecto de Fin de Carrera	15.0	18.0	20.0	15.0
Total	**373.5**	**366.0**	**351.5**	**369.0**

Los treinta y seis créditos de libre elección pueden cursarse en cualquier momento de la carrera. La distribución propuesta en la Tabla 1.5 es sólo una recomendación que propone el Centro, el alumno tiene potestad total sobre la distribución final. El Centro también permite el otorgar hasta doce créditos de libre elección u optativos por prácticas en empresas. Estos créditos equivalen a un mínimo de treinta horas de prácticas.

Para la obtención del título de Ingeniero en Informática deberá también realizarse un Proyecto Fin de Carrera (PFC) al cual se le han asignado cinco créditos, teniendo el carácter de materia

1. Deben ser asignaturas calificadas positivamente.

Tabla 1.5 Distribución de créditos y cuatrimestres en la Titulación de Ingeniería en Informática del CPS.

	Curso	Materias Troncales	Materias Obligatorias	Materias Optativas	Créditos Libre Elección	Proyecto Fin de Carrera	Totales
	1A-1B	57.0	12.0	0.0	0.0		69.0
1er. Ciclo	2A-2B	33.0	24.0	6.0	9.0		72.0
	3A	4.5	18.0	6.0	9.0		37.5
	3B	15.0	7.5	12.0	0.0		34.5
2º Ciclo	4A-4B	42.0	12.0	16.5	0.0		70.5
	5A-5B	15.0	6.0	24.0	18.0	5.0	68.0

obligatoria de universidad. El PFC consiste en la realización de un trabajo o proyecto en el ámbito de la titulación y su evaluación sólo podrá realizarse una vez obtenida la evaluación favorable de la totalidad del resto de los créditos de la titulación.

Respecto de las menciones o especialidades, actualmente no existe en el CPS oferta de las mismas para la titulación de Ingeniero en Informática.

SISTEMA DE EVALUACIÓN Y PROGRESO EN LOS ESTUDIOS DEL CPS

Según los planes de estudio del CPS, cada asignatura es evaluada, en primera instancia por el profesor, existiendo dos convocatorias posibles para realizar su examen final[1].

Adicionalmente, existe lo que se conoce como un Sistema de Evaluación Global o Curricular que se rige por una normativa aprobada en Junta de Centro en Febrero del 1997. Este sistema persigue que un alumno que haya obtenido unos resultados académicos globalmente satisfactorios en un determinado *bloque curricular*, pueda "compensar" alguna asignatura que haya sido calificada por el profesor como de "suspenso compensable", favoreciendo así el progreso del alumno en sus estudios. El criterio recomendado para que un suspenso sea "compensable" es que la calificación obtenida esté comprendida entre un cuatro y un cinco.

1. Aprobada la reducción de 3 a 2 convocatorias en Junta de Centro de Junio de 2004.

2
Sistemas Operativos I. Guía Docente

En este capítulo se presenta la Guía Docente elaborada para la asignatura *Sistemas Operativos I*, materia troncal asignada al Area de Arquitectura y Tecnología de Computadores en los Planes de Estudios de Ingeniería Informática del CPS. Se localiza en primer lugar la posición de la asignatura en el Plan de Estudios, para describir a continuación los objetivos generales que persigue la asignatura. Después, dichos objetivos se organizan en un temario, se discute la bibliografía y se describe la parte práctica de la asignatura. Finalmente se exponen la planificación temporal y los criterios de evaluación.

2.1 Plan de Estudios

El Plan de Estudios de Ingeniería Informática en el CPS, fue aprobado por la Universidad de Zaragoza el 12 de Septiembre de 1994 y publicado en el B.O.E. del 1 de Febrero de 1995. Los principios generales que sigue este plan tomado como ejemplo, fueron expuestos en el Capítulo 1. En la Tabla 2.1 se muestra la distribución de las asignaturas troncales y obligatorias de que consta, y en la Tabla 2.2 las optativas del segundo ciclo de la Titulación. Se han indicado en **negrita** aquellas asignaturas que son impartidas por el Área de Arquitectura y Tecnología de Computadores en el CPS.

Mirando la Tabla 2.1, se puede localizar a la asignatura Sistemas Operativos I (SO I) como asignatura troncal de primer ciclo con una asignación total de seis créditos. Además, es una asignatura que se considera materia de entrada al bloque temático denominado SISTEMAS OPERATIVOS, optativo de segundo ciclo de la Titulación, (mirar Tabla 2.2).

2.1.1 Asignaturas relacionadas

Una vez conocida la localización de SO I en el plan de estudios, se describe de forma breve aquellas asignaturas con las que posee una relación directa. La Figura 2.1 muestra tanto las asignatu-

Tabla 2.1 Asignaturas Troncales/Obligatorias de la Titulación de Ing. en Informática en el CPS*

	Cuatrimestre	Asignatura	Créditos
PRIMER CICLO	1º	Cálculo (T)	7.5
		Matemática Discreta (T)	7.5
		Álgebra (T)	6.0
		Sistemas Lógicos (O)	**6.0**
		Intro. a la Programación (T)	7.5
	2º	Ecuaciones Diferenciales (O)	6.0
		Estadística (T)	7.5
		Fundtos. Físicos de la Informática (T)	6.0
		Arquitectura de Computadores (T)	**7.5**
		Metodología de Programación (T)	7.5
	3º	Cálculo Numérico (O)	6.0
		Fundtos. de Electrónica (O)	4.5
		Organización de Computadores (T)	**7.5**
		Sistemas Operativos I (T)	**6.0**
		Estructuras de Datos y Algoritmos (T)	7.5
	4º	Tecnología Electrónica (O)	4.5
		Sistemas Operativos II (O)	**6.0**
		Ficheros y Bases de Datos (T)	7.5
		Lenguajes, Gramáticas y Autómatas (T)	4.5
	5º	Modelos Abstractos de Cálculo (T)	4.5
		Economía (O)	6.0
		Laboratorio de computadores (O)	**6.0**
		Laboratorio de Programación (O)	6.0
SEGUNDO CICLO	6º	**Diseño de Arquitecturas (T)**	**6.0**
		Conceptos Básicos de redes (T)	**4.5**
		Ingeniería del Software I (T)	4.5
		Lenguajes de Programación (O)	7.5
	7º	**Fundamentos de Arquitecturas Paralelas (T)**	**6.0**
		Sistemas de Transporte de Datos (T)	**4.5**
		Ingeniería del Software II (T)	7.5
		Compiladores I (T)	4.5
		Inteligencia Artificial e Ing. del Conocimiento I (T)	4.5
	8º	Proyectos (T)	6.0
		Inteligencia Artificial e Ing. del Conocimiento II (T)	4.5
		Compiladores II (T)	4.5
		Administración de Empresas (O)	6.0
		Inglés Técnico (O)	6.0
	9º	Org. de la Producción y Gestión de la Calidad (O)	6.0

Tabla 2.1 Asignaturas Troncales/Obligatorias de la Titulación de Ing. en Informática en el CPS*

	Cuatrimestre	Asignatura	Créditos
CRÉDITOS TOTALES			231

* T Troncal - O Obligatoria

ras previas consideradas base de entrada a SO I, como las asignaturas posteriores a las cuales da cobertura.

En SO I se suponen cursadas las asignaturas de Sistemas Lógicos (SL) y Arquitectura de Computadores (AC). En SL se proporciona al alumno todos los fundamentos que debe conocer sobre el funcionamiento de los circuitos digitales (lógica combinacional y secuencial, máquina de estados, registros, ciclo). Mediante una combinación de una ruta de datos sencilla, un autómata de Mealy y una memoria, los alumnos acaban la asignatura conociendo el funcionamiento y diseño de un computador elemental. En AC el alumno aprende los primeros conceptos sobre los niveles de lenguaje máquina y ensamblador. Practica con los métodos básicos de representación y codificación de números naturales, enteros y reales, conoce el repertorio y codificación de instrucciones de una arquitectura escalar convencional (IA-32), y es capaz de traducir estructuras de datos y de control de un lenguaje imperativo de alto nivel a ensamblador. AC también incluye conocimientos sobre la programación del subsistema de E/S y las excepciones síncronas y asíncronas. Con todo lo anterior, al acabar el cuatrimestre el alumno no sólo conoce el funcionamiento básico y componentes de un computador elemental, sino que además posee una mínima experiencia en programación de bajo nivel, necesaria para el seguimiento de asignaturas posteriores.

La asignatura de Organización de Computadores (OC) se imparte simultáneamente con SO I. En esta asignatura se fundamenta el funcionamiento y organización de un computador convencional, se estudian las técnicas usuales de implementación de sus subsistemas y las formas de medir el rendimiento. Al acabar la asignatura el alumno conoce varias implementaciones no segmentadas de una arquitectura sencilla, con control cableado y microprogramado. Conoce también la estructura básica del subsistema de memoria, la sincronización/transferencia entre memoria y periféricos, el concepto de jerarquía de memoria y el soporte a la gestión de memoria virtual. Esta última parte de la asignatura complementaría perfectamente con aquella parte de SO I donde se va a incidir en el papel del Sistema Operativo en lo relativo a la gestión de memoria.

Respecto del resto de asignaturas a las que SO I da cobertura: Sistemas Operativos II (SO II), Laboratorio de Computadores, Diseño de Arquitecturas (DA), Conceptos Avanzados de Sistemas Operativos (CASO) y Estructura Interna de los Sistemas Operativos (EISO), haré una breve descripción de las mismas para reflejar aquellos temas donde los conocimientos de SO I aparecen reflejados bien por ser extendidos o por aparecer ejemplos prácticos de su uso. Más adelante, estos contenidos se volverán a señalar al detallar los temas propuestos para SO I.

Tabla 2.2 Asignaturas Optativas del segundo ciclo de Ing. en Informática en el CPS*

Materia	Asignatura
Arquitecturas de Alto Rendimiento	**Paralelismo en Procesadores**
	Subsistemas de entrada/salida y periféricos
Bases de Datos y Sistemas de Información	Bases de Datos Avanzadas
	Diseño de Bases de Datos Relacionales
	Interacción Hombre-Máquina
	Sistemas de Información
Explotación, Dimensionado y Evaluación de Sistemas y Configuraciones	**Administración de Sistemas Informáticos**
	Model. y Eval. de Prestaciones de Sis. Informáticos
Informática Gráfica	Informática Gráfica
	Modelado Geométrico
	Modelado Visual y Animación
Inteligencia Artificial	Ingeniería de los Sistemas basados en el Conocimiento
	Visión por Computador
Materiales en las T.I.C.	Materiales en las T.I.C
Microelectrónica y Comunicaciones	Fundamentos de Microelectrónica
	Microelectrónica Digital
	Procesado Digital de la Señal†
	Regulación de las Telecomunicaciones y de I+D †
	Transmisión de Imágenes: Técnicas y Sistemas
Modelos Estocásticos en Ingeniería	Modelos Estocásticos en Ingeniería
Programación e Informática Teórica	Esquemas Algorítmicos
	Programación Concurrente
	Programación Paralela
	Técnicas Avanzadas de Programación
Redes de Computadores, Servicios, y Sistemas Teleinformáticos	**Servicios de Alto Nivel en Redes Informáticas**
	Redes de Comunicaciones de Banda Ancha
	Accesos Digitales
	Diseño y Evaluación de Redes †
	Criptografía y Seguridad en Comunicaciones
Simulación de Sistemas Dinámicos	Simulación de Sistemas Dinámicos
Sistemas Informáticos Tiempo Real	Sistemas de Tiempo Real
	Informática Industrial
	Sistemas Empotrados
Sistemas Informáticos para el C.I.M.	Sistemas Informáticos para el C.I.M.
	Control y Programación de Robots
	Sistemas de Eventos Discretos

Tabla 2.2 Asignaturas Optativas del segundo ciclo de Ing. en Informática en el CPS*

Materia	Asignatura
Sistemas Operativos	**Conceptos Avanzados de Sistemas Operativos**
	Estructura Interna de los Sistemas Operativos
Optativas de carácter no técnico †	Ciencia, Tecnología y Sociedad
	Comunicación Oral y Escrita en Español
	Creatividad e Innovación
	Ética y Legislación para Ingenieros
	Historia de la Tecnología
	Ingeniería y Desarrollo Tecnológico
	Psicosociología Industrial
	Introducción al Ejercicio Profesional de la Ingeniería

* Todas las optativas son de seis créditos, excepto las señaladas con † que son de tres créditos.

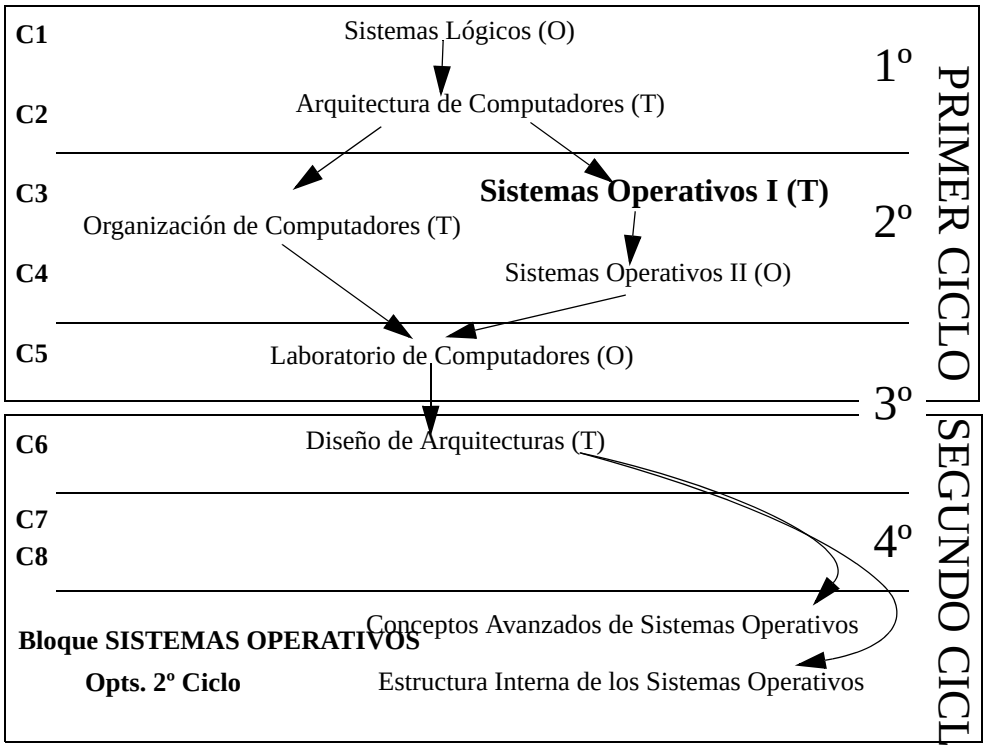

Figura 2.1 Localización de la asignatura SO I en el Plan de Estudios.

Sistemas Operativos II (SO II)

En esta asignatura se extiende aquella parte de SO I donde se estudió al Sistema Operativo como gestor de recursos del computador y se introducen como conceptos nuevos todos los relacionados con temas de seguridad y evaluación de prestaciones en los Sistemas Operativos. La gestión

de procesos se extiende describiendo nuevos mecanismos de comunicación y sincronización y su problemática (bloqueos).

Laboratorio de Computadores (LC)

Esta asignatura está compuesta exclusivamente por créditos prácticos y está enfocada en dos direcciones. Por un lado, le permite al alumno integrar conocimientos que están relacionados entre sí, pero que le han sido enseñados en asignaturas distintas y por otro lado, los alumnos aprenden a desarrollar habilidades de trabajo en equipo, ya que deben diseñar en grupos de dos personas un mini-proyecto por cada tema planteado. La asignatura se articula en tres módulos prácticos:

- Módulo de Arquitectura. Práctica de programación mixta Ensamblador/Alto Nivel controlando directamente diversos periféricos (puerto serie, pantalla...).

- Módulo de Organización. Microprogramación de una arquitectura de pila sobre una organización microprogramable tipo *bit-slice* de la serie 3000 de AMD.

- Módulo de Sistemas Operativos. Práctica de programación con *sockets*: desarrollo de un protocolo simple de transferencia de ficheros.

Diseño de Arquitecturas (DA)

Esta asignatura es troncal de segundo ciclo y tiene como objeto presentar los principios cualitativos y cuantitativos del diseño del repertorio de instrucciones de un procesador convencional, y las técnicas básicas de aumento del rendimiento. Consta de tres grandes bloques temáticos. En el primero, el alumno aprende los fundamentos del diseño, evaluación y estudio de costes de un computador. En el segundo bloque, practica con el diseño completo de la arquitectura de lenguaje máquina, conoce RISC *versus* CISC y su relación con la problemática de compilación. Finalmente, el alumno aprende el diseño de organizaciones más eficientes: la memoria cache, la segmentación, el aumento del paralelismo a nivel de instrucción (ILP), y se le introduce en los procesadores superescalares y la ejecución fuera de orden. Es en esta asignatura donde se completaría todo el tema de memoria visto desde diferentes enfoques en las asignaturas previas de OC y SO I.

Conceptos Avanzados de Sistemas Operativos (CASO)

Esta asignatura es una de las dos que forman el bloque temático optativo de segundo ciclo denominado SISTEMAS OPERATIVOS. Su objeto es el de introducir al alumno en los conceptos básicos de la implementación de sistemas operativos distribuidos. Consta de cuatro grandes temas donde se presentan distintos casos de estudio. Uno de los temas enseña al alumno los aspectos de diseño de los sistemas operativos en red *versus* los sistemas operativos distribuidos. El alumno también aprende conceptos de comunicación en sistemas operativos distribuidos, comunicación y sincronización entre procesos, asignación de recursos y detección de bloqueos. Finalmente, esta asignatura hace hincapié en los temas de protección de recursos, seguridad en la comunicación, identificación de usuarios y servicio de ficheros.

Estructura Interna de los Sistemas Operativos (EISO)

Esta es la segunda asignatura de las dos que forman el bloque temático optativo de segundo ciclo denominado SISTEMAS OPERATIVOS. Sus objetivos son ofrecer una panorámica actual sobre cuáles son las estructuras más habituales que se utilizan en el diseño de los sistemas operativos (núcleo monolítico y micronúcleo). También se analiza en profundidad el problema del soporte de la arquitectura al sistema operativo, comprendiendo los aspectos fundamentales de un núcleo monolítico convencional. Finalmente, se profundiza en los conceptos estudiados previamente en las asignaturas de SO I y SO II, como son: las llamadas al sistema, los ficheros, procesos y *threads*. Como efecto lateral a toda la asignatura, el alumno estudia en profundidad un caso concreto consistente en la obtención del núcleo de un sistema operativo real.

Las dos asignaturas del bloque temático SISTEMAS OPERATIVOS (CASO, EISO) completarían todo lo que en el plan de estudios ejemplo ha sido planificado como contenidos sobre el tema de sistemas operativos y que comenzaron en segundo curso con la asignatura que es objeto de este proyecto docente, Sistemas Operativos I (SO I). A continuación pasamos a detallar tanto los objetivos generales que persigue esta asignatura como el temario detallado.

2.2 Objetivos de la asignatura

Sistemas Operativos I tiene dos objetivos básicos. El primero es el de introducir el conocimiento sobre los conceptos fundamentales en los que se basan los sistemas operativos. El segundo profundizar en los fundamentos de diseño de un sistema operativo, mediante el aprendizaje de una programación avanzada con llamadas al sistema.

Como se ha visto en los apartados anteriores, se ha supuesto que el alumno llega a Sistemas Operativos I conociendo la organización y arquitectura básica de un computador convencional. Esta asignatura, primera de sistemas operativos, ofrece al alumno no sólo el conocimiento de los principios fundamentales de diseño de los sistemas operativos, sino también la forma en que dichos conocimientos pueden aplicarse en un sistema real. Para ello, se le proporciona una visión completa de lo que es un sistema operativo y aprende cuales son las herramientas de diseño e implementación de que dispone todo sistema en su tarea de gestionar el computador. Todo se le muestra al alumno a través de la presentación del sistema operativo desde *tres puntos de vista* básicos. El primero, en el que se muestra al sistema operativo como el *gestor de los recursos* de la máquina. El segundo, en el que un usuario avanzado interacciona contra el sistema operativo programando con *llamadas al sistema*. El tercero, desde la *visión de usuario* normal que interacciona contra el sistema operativo a través de un intérprete de comandos.

Estos conocimientos se extienden en las supuestas asignaturas que dependen directamente de Sistemas Operativos I. En Sistemas Operativos II se estudian mecanismos de sincronización y comunicación entre procesos locales y remotos, así como el problema de los bloqueos. Además, en esta asignatura se cubren los conceptos de seguridad y evaluación de prestaciones de los sistemas operativos. Una de las prácticas de la asignatura Laboratorio de Computadores, profundiza en la comunicación entre procesos remotos desarrollando un protocolo simple de transferencia de ficheros. En Estructura Interna de los Sistemas Operativos se analiza el problema del soporte de la arquitectura al sistema operativo, profundizando en todos los conceptos estudiados en Sis-

temas Operativos I a base de la obtención del núcleo de un sistema operativo real. Los sistemas operativos en red y distribuidos, se estudian en la asignatura Conceptos Avanzados de Sistemas Operativos.

En la presentación de la estructura y mecanismos de los sistemas operativos, se hace especial hincapié en que la aplicación de todos los conceptos depende del estado actual de la tecnología y de las necesidades particulares de cada aplicación. Al finalizar la asignatura el alumno estará familiarizado con las técnicas habituales de los sistemas operativos para gestión de recursos de un computador y habrá practicado con ellas tanto desde el punto de vista de un usuario normal como de un programador de sistemas.

2.3 Temario

Comentaremos en primer lugar los criterios y consideraciones metodológicas utilizados para proponer el temario. Se describe seguidamente la organización general, pasando finalmente a la descripción detallada de los temas.

2.3.1 Estructura general de la asignatura

De los 6.0 créditos que posee la asignatura, se reparten 4.5 a créditos de teoría más problemas y 1.5 a créditos de prácticas en el laboratorio.

La asignatura se va a planificar considerándola cuatrimestral e impartida en un cuatrimestre con 15 semanas efectivas. Se supone una adjudicación de 4 horas por semana, distribuidas en sesiones de una o dos horas de duración. Para cada semana se planifican 2 lecciones de teoría de una hora cada una, 1 hora dedicada a problemas y otra hora dedicada al laboratorio. Las horas dedicadas a problemas se consideran como clases de teoría más activa por parte del alumno, donde se piensan o resuelven pequeños problemas con técnicas de trabajo en grupo. Supondremos un solo grupo para las clases de teoría y problemas y subgrupos de unos 20 alumnos para las prácticas en el laboratorio.

PLANIFICACIÓN DEL CURSO Y DE CADA SESIÓN TEÓRICA

La práctica docente demuestra que el temario debe adecuarse a dos restricciones: a) el número y duración de las sesiones teóricas disponibles, y b) el ritmo de aprendizaje de los estudiantes (*ritmo discente*).

En cuanto a lo primero, la metodología que se describirá en el capítulo siguiente en relación con la clase teórica implica una estructuración determinada de cada sesión. Esta estructura tiene como efecto la necesidad de concretar contenidos para cada sesión. Esto restringe de forma muy realista los contenidos teóricos posibles, obligando a priorizar qué elementos son realmente imprescindibles, qué tiempo necesitan, cuáles podrían eliminarse, y cuáles deben complementarse en las prácticas de laboratorio y los problemas.

En cuanto a lo segundo, la apreciación y adecuación a ese *ritmo discente*, que puede ser más lento que el supuesto *a priori* por el profesor, lleva a refinar la planificación anterior, al identificar

por ejemplo temas o cuestiones en las que los estudiantes experimentan más dificultad. Los ejercicios de final de clase son una buena herramienta para superar esta limitación, ya que proporcionan realimentación inmediata desde el alumno hacia el profesor.

2.3.2 Organización general

Siguiendo la filosofía que la mayoría de los libros utilizan para explicar los conceptos fundamentales de los sistemas operativos (refs. Silberschatz, Galvin, and Gagne 04; Stallings 01; Tanembaum 88), la asignatura Sistemas Operativos I se ha estructurado en tres módulos (Figura 2.2). Precediendo a estos módulos se ha puesto un primer tema introductorio que servirá para establecer tanto el marco en el que se desarrolla el curso como su relación con otras asignaturas. En el *primer* módulo se estudian los conceptos relacionados con la gestión de procesos y las señales. En el *segundo* módulo se estudian los conceptos relacionados con la gestión de la entrada/salida y en su parte final se completan los dos primeros módulos con la comunicación y sincronización entre procesos. En el *último* módulo se estudian los conceptos relacionados con el papel del sistema operativo en la gestión de memoria, ofreciendo una visión general de las técnicas habituales según el tipo de almacenamiento de procesos en memoria. Para cada uno de los tres módulos se dedica un apartado para integrar los conceptos descritos con el aprendizaje de solicitud de cualquier tipo de servicio a un sistema operativo mediante la programación con llamadas al sistema. Para estos apartados se utiliza como ejemplo el sistema operativo UNIX y el lenguaje de programación C.

Cualquier libro que explica fundamentos de diseño de los sistemas operativos (refs. Silberschatz, Galvin, and Gagne 04; Stallings 01) contiene estos mismos módulos en un orden parecido al propuesto aquí. Los libros citados añaden además otros módulos que cubrirían aspectos que no forman parte del contenido de la asignatura propuesta en este proyecto docente y sí de asignaturas que se han supuesto dependientes de ésta (Sistemas Operativos II y Conceptos Avanzados de Sistemas Operativos).

El apartado dedicado al aprendizaje del lenguaje de programación C puede considerarse independiente del resto y se podría colocar en cualquier orden. Sin embargo, al ser el lenguaje con el que se van a desarrollar todos los ejemplos de la asignatura, es conveniente colocarlo antes de comenzar cualquier tema donde aparecen ejemplos de programación. Se ha colocado antes del comienzo del primer Módulo para darle al alumno la posibilidad de empezarlo cuanto antes y madurarlo durante el resto del curso integrando su aprendizaje con todos los conceptos recibidos tanto en este primer módulo como en los dos siguientes.

En cuanto a la ordenación de los dos últimos módulos dedicados a la gestión del almacenamiento, a diferencia de lo que ocurre en los libros sobre sistemas operativos, el estudiar primero la gestión de la entrada/salida enlaza bien con el módulo anterior de procesos ya que le da continuidad y permite acabar los dos módulos con el tema de comunicación y sincronización entre procesos.

El Módulo de gestión de memoria colocado en último lugar le permite al alumno entrar en la asignatura de Diseño de Arquitecturas, donde completará el estudio del recurso memoria, con conocimientos frescos sobre el papel del sistema operativo en la gestión de memoria.

Tema 0 - Presentación de la asignatura	3h
Tema 1 - Curso rápido de Lenguaje de Programación C	4h
MÓDULO 1 - PROCESOS (15 horas)	
Tema 2 - Gestión de Procesos	3h
Tema 3 - Llamadas al Sistema. Procesos en UNIX	3h
Tema 4 - Intérprete de Comandos	1h
Tema 5 - Señales	4h
MÓDULO 2 - ENTRADA/SALIDA (8 horas)	
Tema 6 - Gestión de la Entrada/Salida	2h
Tema 7 - Llamadas al sistema. Ficheros UNIX	3h
Tema 8 - Redireccionamiento	1h
Tema 9 - Comunicación elemental entre procesos	2h
MÓDULO 3 - MEMORIA (4 horas)	
Tema 10 - Gestión de Memoria	2h
Tema 11 - Llamadas al sistema. Memoria en UNIX	2h

Figura 2.2 Esquema del temario de Sistemas Operativos I y tiempo dedicado a clases de teoría

Cada módulo se completa con clases de problemas y los dos primeros con prácticas que el alumno desarrolla en una o varias sesiones de laboratorio. Estas prácticas profundizan en los aspectos del temario relativos a la programación utilizando llamadas al sistema. En esta sección se desarrolla el temario de la asignatura y su distribución en las sesiones de teoría. En 2.5 se detallan las clases de problemas y en 2.6 se presentan las prácticas asociadas al módulo correspondiente.

ELECCIÓN DEL SISTEMA OPERATIVO DE REFERENCIA

La elección del sistema operativo UNIX como elemento integrador para el estudio de todos los conceptos presentados en la asignatura se debe a las siguientes razones:

- Uno de los objetivos básicos de la asignatura es el de ofrecer al alumno un estudio detallado de los fundamentos de diseño comunes a todos los sistemas operativos. Para ilustrar y reforzar estos conceptos, nada mejor que acercarlos a las decisiones de diseño que deben tomarse en un sistema operativo real tomado como ejemplo.

- Con el objetivo de primar el aprendizaje del alumno, se ha considerado más efectivo reforzar los conceptos presentados mediante su aplicación sobre un *único* sistema operativo representativo. Utilizar mucha variedad de sistemas ejemplo, además de que corre el riesgo de perder representatividad en los ejemplos, puede perjudicar el aprendizaje del alumno al desviar su atención a situaciones particulares y poco relevantes de cada sistema.

- UNIX es un sistema operativo que al ser multiusuario se adapta bien a los conceptos presentados en la asignatura. Además, es un sistema operativo lo suficientemente pequeño para ser comprendido, aunque sin llegar a ser un sistema “de juguete”. Una vez conocidas las bases de programación en lenguaje C, resulta fácil para el alumno estudiar su repertorio de llamadas al sistema y aprender con ello un tipo de programación avanzada que lo colocará al mismo nivel que el propio sistema.

- La posibilidad de encontrar un entorno de trabajo donde se haga uso de un sistema operativo UNIX es alta. Su relevancia y representatividad es real ya que además de ser un sistema actualmente implementado sobre una gran variedad de máquinas (desde micro hasta supercomputadores) también posee versiones de fuentes libres (tipo *linux*) que lo hacen fácilmente accesible al alumno. El basar el aprendizaje de la asignatura sobre un tipo de sistema tan popular aumenta la viabilidad de aplicación de la misma.

2.3.3 Descripción de temas

Describimos a continuación los módulos, temas y lecciones en los que se ha estructurado el temario. En cada módulo se realiza una descripción general de los objetivos y las consideraciones tomadas para definir los temas y lecciones que lo componen, así como los problemas y las prácticas relacionadas con el módulo. En cada tema se establecen los contenidos y algunos ejemplos de ejercicios planteados como final de cada lección. Los ejercicios se enuncian de forma orientativa. En la Sección 2.4 se presenta el soporte bibliográfico para el curso. Aunque en la Sección 2.7 se detalla una propuesta de planificación, en cada tema se especifica el tiempo dedicado a lecciones de teoría y problemas, y en cada módulo el dedicado a laboratorio.

Tema 0 Presentación de la asignatura

Lección 0.1 Presentación de la asignatura y su relación con asignaturas afines
Lección 0.2 Historia y objetivos de los Sistemas Operativos. Multiprogramación
Lección 0.3 Necesidades *hardware* de la multiprogramación

Duración: 3 h de teoría.

Objetivos: *Presentar la asignatura motivando la necesidad de estudio del sistema operativo. Determinar en el alumno el nivel de conocimientos previos relacionados.*

Descripción: Este primer tema consta de tres lecciones. Comienza con una primera lección donde se presentan al alumno los motivos que justifican la asignatura Sistemas Operativos I, mostrando su relación con las asignaturas previas y posteriores en el plan de estudios tomado como ejemplo. También se presentan los objetivos que persigue la asignatura y varios aspectos de organización docente: temario, sistema de evaluación, bibliografía, material puesto a disposición del alumno, calendario de clases teóricas y prácticas, etc.

Las siguientes lecciones delimitan el contexto en que se va a mover el resto de la asignatura. En la segunda lección se hace un repaso a la historia relacionada con los sistemas operativos y su evolución según las necesidades de cada momento. Evolución centrada bien desde el punto de vista de la interacción usuario-computador, bien desde la eficiencia a extraer del computador. Se presenta el concepto de proceso y se termina la lección con la presentación de la *multiprogramación* y los sistemas operativos de *tiempo-compartido* dando una primera visión de sus necesidades tanto *hardware* como *software*.

En la tercera y última lección se presentan las bases sobre las necesidades *hardware* que mejor se adaptan a un sistema operativo de tiempo-compartido. Son éstos conceptos que se suponen conocidos por el alumno y que se repasan aquí para centrar el contexto sobre el que se va a trabajar a lo largo de la asignatura. Se realiza aquí una evaluación inicial del alumno, determinando su destreza sobre conceptos previos relacionados con el curso. Se recuerdan los mecanismos usuales de sincronización y transferencia, encuesta/interrupciones y transferencia programada o por DMA (*Direct Memory Access*), sus ventajas y limitaciones. Se describen los mecanismos básicos de protección y manejo de memoria. Aparece el concepto de *llamada al sistema* y se finaliza la lección presentado al sistema operativo como un *software* dedicado a atender peticiones de recursos.

Tema 1 Curso rápido de Lenguaje de Programación C

Duración: 4 h de problemas

Objetivos: *Proporcionar los conocimientos básicos sobre el lenguaje de programación utilizado en los ejemplos de la asignatura. Dar la base teórica necesaria para poder integrar los conceptos presentados con el aprendizaje de una programación con llamadas al sistema.*

Descripción: Como se ha dicho al comienzo del temario, se ha organizado el curso de forma que incluya un tema completo para describir el lenguaje de programación C, utilizado como base de toda la asignatura. Se supone que el alumno tiene conocimientos básicos de algún lenguaje de

programación de alto nivel y esa es la razón por la que se ha considerado suficiente dedicar 4 de las 14 horas de problemas para el aprendizaje del lenguaje C.

Además de los ejemplos que se van viendo a lo largo de la asignatura, el tema se puede apoyar con una colección de ejercicios puestos a disposición del alumno para permitirle desarrollar habilidades con el lenguaje. Para motivar al alumno a la práctica de esos ejercicios, se dedican 15 minutos al final de cada clase de C a que trabajen en algunos de ellos, obligándoles así a entender los conceptos adquiridos y aplicarlos en casos prácticos.

Todos los conceptos vistos aquí serán utilizados después en aquellos temas donde se presentan las llamadas al sistema. El conocimiento de las bases y estructura habitual de un programa en C es fundamental para aprender la programación avanzada en UNIX.

Ejemplo de ejercicio de C:

Como apoyo al Tema 1 se le indica al alumno la posibilidad de extender sus conocimientos sobre el lenguaje mediante el estudio de un conjunto de problemas que se pone a su disposición. Un ejemplo de tales problemas sería uno típico sobre el paso de parámetros por referencia:

```
#include <stdio.h>

void intercambia();

main()

{

  int a = 1, b = 2;

  intercambia( &a, &b );

  printf( "\n\ta = %d\tb = %d\n", a, b );

}

void intercambia( x, y )

int *x, *y;

{

int tmp;

  tmp = *x;

  *x = *y;

  *y = tmp;

}
```

MÓDULO 1 PROCESOS

Tema 2 - Gestión de Procesos

Tema 3 - Llamadas al Sistema. Procesos en UNIX

Tema 4 - Intérprete de Comandos

Tema 5 - Señales

Duración: 11 h de teoría, 3 h de problemas y 4 h de laboratorio.

Objetivos: *Conocer todos los aspectos relacionados con los conceptos de proceso y concurrencia en un sistema operativo moderno. Al finalizar el módulo, el alumno habrá aprendido:*

- *Cuales son aquellas actividades relacionadas con los procesos de las cuales el sistema operativo es el responsable.*
- *Cuales son las llamadas al sistema UNIX que le permitirán realizar una programación avanzada en lenguaje C solicitando al sistema operativo cualquier servicio que éste le ofrece sobre los procesos.*
- *En qué consiste un mecanismo de concurrencia UNIX basado en señales, cuales son las señales más utilizadas, que acciones provocan y cuales son los problemas que plantean.*

Descripción: En el primer tema se estudia como un sistema permite convivir a procesos del sistema operativo y de usuario ejecutándose de forma concurrente y con el procesador compartido entre todos ellos. Se estudian distintas técnicas de planificación del procesador y se considera el problema de la elección del algoritmo más adecuado a un sistema concreto. El Tema 1 está dedicado al aprendizaje del lenguaje de programación C.

El Tema 3 incide de nuevo en el estudio del control de procesos pero ahora desde la perspectiva del conjunto de llamadas al sistema que UNIX dispone para ello. Se presentan las llamadas relacionadas con la creación, ejecución y terminación de procesos, los identificadores asociados a los procesos y como estos se ven afectados por las distintas llamadas. Al igual que en el resto de temas dedicados al aprendizaje de las llamadas al sistema, su estudio es integrado mediante la utilización de ejemplos representativos escritos en lenguaje C.

El Tema 4 se dedica a presentar el funcionamiento de un intérprete de comandos, elemento base para la interacción entre el usuario y el sistema operativo. Esta presentación se apoya en el uso de ejemplos de programación que de forma incremental muestran la estructura que tiene todo intérprete según su funcionalidad.

El último tema estudia las señales como elementos usados para provocar determinadas acciones entre los procesos. Este tema puede considerarse como inicio de la parte que en esta asignatura se dedica al estudio de la comunicación y sincronización entre procesos. Se presenta el modelo de señales utilizado en las primeras versiones de UNIX y las llamadas al sistema asociadas. Se po-

nen en evidencia las limitaciones de este modelo (señales *no fiables*) y se presentan algunas de las modificaciones que versiones posteriores han incluido para superarlas (señales *fiables*).

Prácticas: Las prácticas 1 y 2, que se describen en el apartado 2.6, cubren la materia de los temas 2 al 5 ambos inclusive.

Tema 2 Gestión de Procesos

Lección 2.1 Estados de los procesos
Lección 2.2 Planificación de procesos. Parte I
Lección 2.3 Planificación de procesos. Parte II

Duración: 3 h de teoría

Objetivos: *Motivar el estudio de la tarea que tiene el sistema operativo como gestor de procesos. Dar a conocer los principios en los que se basa la gestión de procesos. Describir las estructuras necesarias y el funcionamiento básico de esa gestión para un sistema operativo moderno. Proporcionar conocimientos sobre los modelos más habituales que utilizan los sistemas para gestionar el procesador.*

Descripción: Para motivar el estudio del sistema operativo como gestor de procesos, el tema comienza en la Lección 2.1 mostrando cual es la forma habitual en la que evolucionan los procesos. Se ve con ello como durante toda su ejecución los procesos pasan por periodos donde van necesitando de distintos recursos. Se introduce el concepto de *estado* y la visión del sistema operativo como el *software* que gestiona los cambios de estado que en cada momento necesitan los *procesos de usuario*. Como resultado y para que el sistema operativo pueda seguir la pista de los procesos, se presentan las estructuras de datos que se utilizan para describir cada proceso. La lección acaba viendo el concepto de *cambio de contexto* y de como el sistema operativo hace de *planificador* de los procesos imponiendo su control sobre todos ellos.

Las dos lecciones siguientes tratan sobre la planificación vista como clave de diseño de un sistema monoprocesador y multiprogramado. La Lección 2.2 comienza presentando los tres tipos de planificación del procesador: a largo, medio y corto plazo. A continuación pasan a estudiarse los aspectos de diseño más relevantes del planificador a corto plazo (*scheduler*) por ser aquel ejecutado más frecuentemente y el que toma las decisiones de más detalle sobre el proceso a ejecutar a continuación. Se distinguen los distintos tipos de planificadores según los criterios más habituales utilizados en su diseño: orientados al beneficio del usuario, al del propio sistema y criterios basados en prioridades. La lección acaba presentando el concepto de *preemption* como criterio relacionado con la decisión de interrumpir o no el proceso que se encuentra en ejecución.

La Lección 2.3, última lección del tema, describe los algoritmos más habituales utilizados en la planificación del procesador, las estrategias que siguen y sus características principales. Después de presentar la idea fundamental en la que se basa cada algoritmo, se va mostrando el funcionamiento detallado de cada uno mediante ejemplos sencillos de funcionamiento. Cada algoritmo aparece como mejora al algoritmo previo tras haber visto situaciones en las que ese algoritmo o puede fallar o no funciona a pleno rendimiento. Como resultado, el alumno obtiene una idea clara sobre la variedad de algoritmos y es capaz de ver la viabilidad de aplicación de cada uno a las

necesidades de cada sistema. La lección acaba presentando el tipo de planificación habitual utilizado en un sistema UNIX.

Tema 3 Llamadas al Sistema. Procesos en UNIX

Lección 3.1 Llamadas al sistema para procesos

Lección 3.2 Algoritmos de las llamadas. Casos raros

Lección 3.3 Información sobre procesos

Duración: 3 h de teoría

Objetivos: *Estudiar el conjunto de llamadas al sistema relacionadas con el control de procesos en un sistema operativo UNIX. Ejercitar con la programación avanzada en C.*

Descripción: Al igual que en el resto de temas que en este proyecto se han propuesto para el estudio de las llamadas al sistema, se trata de ejercitar con la programación avanzada en lenguaje C mediante la aplicación de los conceptos aprendidos en cada tema. La estrategia utilizada para la presentación de las distintas llamadas al sistema está enfocada en el estudiante y funciona dándole primero la idea fundamental que hay detrás de cada llamada, para a continuación detallarle su sintaxis completa y acabar apoyando la explicación con ejemplos concretos de funcionamiento.

En este tema y partiendo del estudio que sobre la gestión de procesos se ha realizado en el Tema 2, se trata de integrar esos conocimientos mediante la presentación del conjunto de llamadas al sistema UNIX relacionadas con el control de procesos.

El tema comienza en la Lección 3.1 relacionando el entorno asociado a todo proceso con la estructura que posee el programa en C que ejecuta dicho proceso. Se estudia el detalle de como la función *main* es llamada cuando el programa se ejecuta y de como la *línea de comandos* y las *variables de entorno* se pasan al programa a través de sus argumentos. Se introduce el concepto de *imagen de un proceso* y se ven cuales son los segmentos de memoria que requiere todo programa. La lección acaba presentando las llamadas al sistema para creación y ejecución de nuevos procesos (*fork* y *exec*), así como las utilizadas habitualmente para esperar y finalizar procesos (*wait* y *exit*). Como ejemplo explicativo del funcionamiento de estas llamadas, se utiliza el que más adelante será presentado como el mecanismo habitual usado en todo *intérprete de comandos*.

La Lección 3.2 completa el estudio de las llamadas relacionadas con el control de procesos estudiando el detalle de todas las actividades que el sistema operativo lleva a cabo para atender a *fork* y *exec*. Con esto el alumno tiene toda la información necesaria para poder relacionar el concepto de ejecución concurrente de los procesos con la creación y ejecución de los mismos. La lección acaba presentando el concepto de *condición de carrera*, estudiando ejemplos de las distintas situaciones que pueden aparecer debido a la ejecución concurrente y a la decisión que tome el planificador. Los ejemplos que se usan son programas escritos en C que intentan forzar las situaciones que se quieren presentar: la esperada tras la creación de un nuevo proceso ó situaciones raras que aparecen a consecuencia de las condiciones de carrera y de una programación poco controlada (procesos *huérfanos* y procesos *zombies*).

En la última lección se hace un estudio detallado de cuales son los identificadores y permisos asociados tanto a los procesos como a los programas que ejecutan. Se presentan los identificadores de usuario, de grupo, real, efectivo, los identificadores del proceso, del padre del proceso, del grupo al que pertenece el proceso, permisos de acceso del usuario, del grupo, del resto, etc. Asimismo, se estudia el conjunto de llamadas al sistema asociadas a cada identificador y se ve de qué forma estos identificadores se ven afectados por las llamadas relacionadas con el control de procesos que se vieron en las 2 lecciones anteriores.

Para finalizar el tema se le plantean al alumno un par de ejercicios para practicar con algunos de los conceptos presentados. Para ello, se pueden dedicar una clase de problemas a que los alumnos trabajen en pequeños grupos los siguientes ejercicios:

- Para practicar con el concepto de concurrencia y de las condiciones de carrera que pueden aparecer en la ejecución de los procesos y que fueron presentadas en la Lección 3.2. Analizar cual será el resultado de la ejecución del siguiente programa escrito en C con llamadas al sistema relacionadas con el control de procesos:

```
main( int argc, char *argv[] )

{ int i, maxveces;

  maxveces = atoi(argv[1]);

  if ( fork( )==0  ){

     for( i = 0; i < maxveces; i++ )

         fprintf(stderr, "**" );

     printf( "\tsoy ** y me muero\n" );

     exit( 0 );

  }

  if ( fork( )==0  ){

     for( i = 0; i < maxveces; i++ )

         fprintf(stderr, "--" );

     printf( "\tsoy -- y me muero\n" );

     exit( 0 );

  }

  wait(NULL);

  wait(NULL);
```

}

- Para practicar con algunos de los identificadores presentados en la Lección 3.3. Entender el funcionamiento del bit de *set-user-id* activo en el permiso de ejecución del programa de sistema *passwd* que le permite a todo usuario cambiarse su *password*.

La resolución de estos ejercicios obliga al alumno a entender correctamente los conceptos recién adquiridos y a aplicarlos en un caso práctico. Esto aumenta el nivel de asimilación de estos conceptos que son la base de los que se desarrollan en los temas siguientes. Dentro de la planificación propuesta en este proyecto la cual está resumida en la Tabla 2.4 (página 74), estos 2 ejercicios se corresponden con el Problema 2.

Tema 4 Intérprete de Comandos

Lección 4.1 Intérprete de Comandos

Duración: 1 h de teoría

Objetivos: *Describir el detalle de la implementación de un intérprete de comandos.*

Descripción: En la Lección 4.1, única de la que consta este tema, se sigue incidiendo en la visión del usuario como programador de sistemas, en este caso relacionado con el uso de las llamadas al sistema asociadas al control de procesos. En los distintos temas de que consta la asignatura, al alumno se le van presentado ejemplos de cómo programar al mismo nivel que lo hace el propio sistema. Son ejemplos donde aprende a escribir programas que hacen lo mismo que los programas de sistema que en un determinado sistema operativo ya encuentra escritos y disponibles para ejecutar.

En esta lección y como ejemplo de uso de las llamadas de control de procesos, se le muestra al alumno de qué forma puede escribir un programa en C que utilizando llamadas al sistema haga lo mismo que un intérprete de comandos. La lección va presentando de forma incremental ejemplos de cómo realizar esa implementación. Se comienza con el estudio de cómo sería una programación básica y a continuación se va modificando a base de añadirle nuevas funcionalidades.

Esta lección complementa con las 2 primeras prácticas de laboratorio donde el alumno ha estudiado como ejemplo de intérprete de comandos al *shell* de UNIX. Son las Prácticas SH1 y Práctica SH2. Una descripción más detallada de estas dos prácticas aparece en el apartado 2.6 de Clases de Laboratorio.

Ejemplo de ejercicio de final de la clase:

Al final de la lección, se dedican 15 minutos a que los alumnos trabajen en pequeños grupos en un ejercicio que plantea algún análisis más fino sobre el último de los intérpretes presentados. Para obligarle a entender y aplicar algunos de los conceptos adquiridos en el módulo, se les pide que identifiquen la aparición de alguno de los casos patológicos estudiados en el Tema 3.

Tema 5 Señales

Lección 5.1 Características de las señales. Implementación
Lección 5.2 Llamadas al sistema asociadas a señales
Lección 5.3 Problemas con señales. Parte I
Lección 5.4 Problemas con señales. Parte II

Duración: 4 h de teoría

Objetivos: *Iniciar el estudio a la comunicación y sincronización entre procesos mediante la presentación y análisis del concepto de señal en un sistema operativo UNIX. Proporcionar una visión global del modelo de señales utilizado en las primeras versiones de UNIX, sus características, implementación y posibles soluciones a sus limitaciones.*

Descripción: Un estudio completo sobre los distintos mecanismos de comunicación y sincronización entre procesos UNIX, debe cubrir al menos todos aquellos relacionados con la comunicación de datos entre procesos (tuberías, mensajes y memoria compartida) y los relacionados con la comunicación de acciones a los procesos (semáforos y señales). La propuesta realizada en este proyecto docente, de todo lo anterior va a cubrir los conceptos de *tuberías* y *señales* como ejemplos básicos de cada uno de los dos tipos de comunicación entre procesos. La comunicación de datos mediante tuberías se estudia más adelante en el Tema 9 y es en este Tema 5 donde se va a presentar el detalle de todos los conceptos relacionados con el modelo de señales. En el Plan de Estudios tomado como ejemplo en este proyecto, el resto de conceptos los aprenderá el alumno en la asignatura Sistemas Operativos II, continuación de la descrita en este proyecto.

En la Lección 5.1 se define el concepto de señal o interrupción asíncrona, como aquel mecanismo *software* que informa a todo proceso del acontecimiento de un suceso asíncrono (*evento*). Al mismo tiempo se aprende de qué forma las señales también proporcionan un mecanismo mediante el cual todo proceso es capaz de reaccionar ante estos eventos asíncronos. En esta primera lección del tema se describen también las posibles fuentes de eventos, lo que pretende comunicar cada una de las señales y las posibles reacciones que pueden programarse los procesos frente a las señales. La lección acaba viendo posibles implementaciones para tratar con señales y los problemas que éstas pueden generar. Se presenta el concepto de *rutina de captura* de la señal.

La Lección 5.2 es la que en este tema se dedica a ejercitar con la programación avanzada en C ahora desde la utilización del conjunto de llamadas al sistema relacionadas con señales. Se presenta el conjunto de llamadas asociadas al modelo de señales utilizado en las primeras versiones de UNIX: *signal*, *kill*, *pause*, *alarm*. Se da primero la idea fundamental que hay detrás de cada llamada, luego se detalla su sintaxis completa y todo se va apoyando mediante ejemplos de uso.

Al final de esta lección se propone el Problema 4 mediante el cual el alumno va a ejercitar el modelo de señales recién propuesto. La descripción completa del problema se puede consultar en el apartado 2.5 de Clases de Problemas.

Las dos últimas lecciones del tema se dedican a poner de manifiesto las limitaciones que tiene el modelo de señales presentado y sus posibles soluciones. Con los conceptos aprendidos en el tema, el alumno es consciente de que en el modelo estudiado las señales pueden perderse y es un modelo donde un proceso posee muy poco control sobre las señales y por ejemplo donde no se

contempla la posibilidad de que un proceso pueda retener el tratamiento de determinadas señales (útil en ejecución de secciones de código críticas). La Figura 2.3 muestra las funciones básicas que diferencian a los dos modelos de implementación de señales que se van a ver y las versiones de UNIX donde se pueden localizar.

Llamadas al sistema	versión de UNIX	mantenimiento de la *rutina de captura*	posibilidad de bloquear señales	posibilidad de reiniciar llamadas bloqueantes
signal	V7, SVR2, SVR3, SVR4	no	no	nunca
sigaction, sigprocmask, sigpending, sigsuspend	POSIX.1	si	si	sin especificar
	SVR4	si	si	opcional
	4.3+BSD	si	si	opcional

Figura 2.3 Caracteristicas asociadas a las distintas implementaciones de señales.

La Lección 5.3 comienza presentado el concepto de señal segura (*reliable*) y las diferencias entre un modelo de señales no seguro (*non-reliable*) como el visto en las lecciones previas y uno seguro (*reliable*) el cual se presenta a continuación. El resto de la lección se dedica a aprender el concepto de *bloqueo* de señales. Se muestra de que forma un modelo seguro pone a disposición del programador estructuras y llamadas al sistema que le permitirán bloquear señales. La lección acaba viendo el concepto del *signal mask* y la sintaxis de todo un conjunto de llamadas al sistema de un modelo que manipula señales de forma segura.

Continuando con el tema de limitaciones del modelo no seguro, la Lección 5.4 presenta más características que lo diferencian de uno seguro y lo hace mediante el estudio en detalle del funcionamiento de la llamada *sigaction*. Se ve *sigaction* como llamada que sustituye a *signal* y se compara de que forma puede usarse para superar determinadas limitaciones del modelo no seguro. Una de las limitaciones que se resuelven con *sigaction* es la de poder restaurar la actividad que se encontraba realizando una llamada bloqueante cuando fue desbloqueada por una señal. La lección acaba planteando el Problema 6 descrito en el apartado 2.5. Es un problema relacionado con una mala programación utilizando señales.

MÓDULO 2 ENTRADA/SALIDA

Tema 6 - Gestión de la Entrada/Salida

Tema 7 - Llamadas al sistema. Ficheros UNIX

Tema 8 - Redireccionamiento

Tema 9 - Comunicación elemental entre procesos

Duración: 8 h de teoría, 4 h de problemas y 4 h de laboratorio

Objetivos: *Este módulo ofrece al alumno los conocimientos sobre aquellos conceptos del sistema operativo que están relacionados con la entrada/salida de información. Al final del módulo, el alumno habrá aprendido:*

- *Cuales son las actividades y los elementos más habituales que posee un gestor de la entrada/salida dentro de un sistema operativo moderno.*
- *Cuales son las llamadas al sistema UNIX que le permiten solicitar a su gestor de la entrada/salida, el Sistema de Ficheros, cualquier servicio que éste le ofrece.*
- *Cuales son las técnicas más habituales utilizadas para comunicar y sincronizar procesos que quieren intercambiarse información.*

Descripción: En el primer tema se estudia el concepto del *gestor de la entrada/salida* como la parte de un sistema operativo utilizada para permitirle al usuario una relación con los dispositivos físicos que no dependa del tipo de dispositivo del que se trate. Se conocen los elementos y estrategias que los sistemas operativos usan para hacer que esa gestión sea además óptima (*drivers*, formas de almacenamiento, acceso a la información...). El siguiente paso es conocer de que forma se aplican estos conceptos de diseño en un sistema operativo concreto, esto se hace al final del primer tema tomando como ejemplo el Sistema de Ficheros UNIX.

El Tema 7 se dedica a presentar el conjunto de funciones disponibles en UNIX para tratar con el sistema de ficheros. El final de este tema estudia el concepto de *buffering* al analizar las diferencias entre usar llamadas al sistema o funciones de librería para manejar datos desde/hacia la entrada/salida estandar. El Tema 8 añade más información relacionada con la entrada/salida estandar al presentar el concepto de redireccionamiento.

El módulo termina con el Tema 9 donde se presentan los dos mecanismos más comunes utilizados para transmitir información entre procesos que residen bajo el mismo sistema operativo (*fifos* y *pipes*). Estos conceptos serán ampliados en asignaturas posteriores donde se verán los problemas habituales de comunicación entre procesos y nuevos mecanismos de comunicación remota.

Prácticas: La prácticas 4 y 5, que se describen en el apartado 2.6, cubren la materia de los temas 6 al 9 ambos inclusive.

Tema 6 Gestión de la Entrada/Salida

Lección 6.1 Software de E/S. Independencia del dispositivo
Lección 6.2 Sistema de Ficheros en UNIX. Tablas

Duración: 2 h de teoría

Objetivos: *Definir los objetivos y dar a conocer los principios en los que se basa todo gestor de la entrada/salida. Ofrecer un modelo ejemplo de funcionamiento.*

Descripción: El tema comienza en la Lección 6.1 presentando al *gestor de la entrada/salida* como aquella parte del sistema operativo cuyo objetivo principal es el de abstraer toda característica *hardware* de los dispositivos físicos sobre los cuales se solicitan operaciones de entrada/salida. Se define el *gestor de la entrada/salida* como el *software* encargado de facilitarle al usuario el

acceso a cualquier dispositivo al ofrecerle un *interface* que no depende del dispositivo. Se presenta el concepto de *driver* como elemento al que el sistema traslada toda solicitud de acceso y es el encargado de llevar a cabo el control de la tarea solicitada, comunicándose con el controlador del dispositivo físico dado. Se introduce el concepto de *fichero* y el tipo de *estructura directorio* como modelo ejemplo de abstracción del dispositivo físico donde realmente está almacenada la información (generalmente el *disco*). La lección continua presentando tres de los modelos más habituales utilizados en la asignación y localización de los datos almacenados en el disco. Se muestran las estructuras básicas que precisa un modelo *contiguo*, *encadenado* o *indexado*, así como sus características más relevantes y sus limitaciones. Como ejemplo de final de lección, se presenta el caso de UNIX y su modelo de asignación *indexado por niveles*.

En la segunda Lección 6.2 se presenta como caso ejemplo de gestor de la entrada/salida, el sistema de ficheros del sistema operativo UNIX. Se describe como es la organización habitual en este sistema, viendo el detalle de cada una de las partes en las que se divide el disco que almacena un determinado sistema de ficheros. Se define el concepto de *i-node* como la estructura de control a nivel del sistema operativo que contiene toda la información necesaria para identificar y localizar a un fichero. Se presenta también el concepto de *descriptor de fichero* como el elemento que a nivel de programa de usuario identifica a un determinado fichero. La lección acaba mostrando las tablas UNIX usadas para la gestión de su sistema de ficheros. Para cada una de ellas, se muestra el detalle de cuales son las estructuras que almacena y de como se gestionan. Se hace hincapié en como este tipo de gestión mediante tablas, en un sistema multiusuario, permite a los programas acceder a los datos de forma rápida, eficiente y flexible. El resto de información relacionada con la gestión de las tablas se completa en el tema siguiente donde se trata el estudio de todas las llamadas al sistema que los programas de usuario pueden utilizar para tratar con ficheros.

Ejemplo de ejercicio de final de clase:

- En este ejercicio el alumno deberá aplicar el modelo de asignación indexado por niveles que acaba de aprender para identificar todos los pasos que se han de realizar a la hora de localizar un determinado fichero.

  Conocidas el tipo de estructura y asignación que utiliza el sistema de ficheros en UNIX, el alumno debe implementar una función que le permita leer un determinado byte de un fichero dado. La función debe tener la siguiente sintaxis:

  ```
  leer(fichero, byte)
  ```

  Tras la implementación anterior, el alumno calculará cuantos bloques de disco deben leerse con la función recién diseñada para acceder al byte número 450 de un fichero que tiene como nombre

  ```
  /users3/sistemas/error.h
  ```

Tema 7 Llamadas al sistema. Ficheros UNIX

Lección 7.1 Llamadas al sistema para ficheros. Parte I
Lección 7.2 Llamadas al sistema para ficheros. Parte II
Lección 7.3 Llamadas al sistema *versus* librería de entrada/salida

Duración: 3 h de teoría

Objetivos: *Estudiar el conjunto de llamadas al sistema relacionadas con el control de ficheros en un sistema operativo UNIX. Ejercitar con la programación avanzada en C. Introducir el concepto de* buffering *para la entrada/salida de datos.*

Descripción: Este tema vuelve a ser esencial dentro del módulo al dedicarse a ejercitar la programación avanzada en C ahora desde la utilización del conjunto de llamadas al sistema relacionadas con el sistema de ficheros. La primera lección del tema comienza presentando el conjunto de llamadas que se verán a continuación, la tarea que tienen asociada y la relación que existe entre ellas. Seguidamente, se presenta como primer conjunto de llamadas aquellas relacionadas directamente con los descriptores de ficheros, concepto presentado en el tema anterior. Se detalla la sintaxis completa de las llamadas *creat*, *mknod*, *open*, *close* y *dup*, se muestran algunos casos particulares de funcionamiento y se ven ejemplos de uso. Tras la descripción detallada de cada llamada, se completa su estudio mostrando el efecto que cada una de ellas posee sobre las tablas que el sistema operativo maneja para la gestión de ficheros y que también fueron presentadas en el Tema 6. La lección acaba viendo el efecto que las llamadas sobre procesos (*fork* y *exec*) poseen sobre las tablas.

La Lección 7.2 completa la presentación y estudio del conjunto de llamadas al sistema que tienen que ver ahora con el acceso a los ficheros para obtener o modificar información. Primero se detallan aquellas relacionadas con accesos para lectura/escritura de datos y posicionamiento en el fichero (*read*, *write*, *lseek*). Segundo, las que permiten extraer toda la información que el sistema posee sobre los ficheros (*stat*, *fstat*). Finalmente se introduce el concepto de *enlace* y se estudian las llamadas para crear/eliminar enlaces de ficheros (*link*, *unlink*).

Para el usuario programador, la abstracción que le ofrece el sistema operativo en el manejo de ficheros, le permite comenzar referenciando por su nombre a los ficheros que quiere manejar dejándole al sistema la tarea de localizar al fichero real. La localización la hace el sistema mediante el manejo de las tres tablas que se presentaron en el tema anterior. Una vez que la gestión de dichas tablas se ha realizado el sistema deja enlazado el *i-node* del fichero referenciado con el *descriptor de fichero* devuelto por la llamada (*creat*, *open*, *dup*). Las llamadas posteriores que se hagan para acceder al fichero (*read*, *write*, *lseek*) lo harán a través de su descriptor. Es esta la forma de gestión usada en el sistema multiusuario UNIX y es la presentada como ejemplo de flexibilidad en el manejo de ficheros. Es el tipo de gestión de la entrada/salida que aprende el alumno tras relacionar todos los conceptos presentados en los Temas 6 y 7.

El tema termina en la Lección 7.3 donde se introduce el concepto de *buffering*. Se muestra primero la diferencia entre una programación para el manejo de la entrada/salida de datos realizada usando llamadas al sistema o realizada usando funciones de la librería estandar de C para entrada/salida. Se presenta después como las funciones de librería, además de permitir el formateo de la entrada/salida, utilizan una zona intermedia de almacenamiento de datos (concepto de *buffer*

de la entrada/salida). Se ve cual es la forma habitual de comportamiento que tiene el *buffer* y bajo que condiciones ese comportamiento puede cambiar. Como en la lección anterior, se acaba viendo el efecto que las llamadas sobre procesos (*fork* y *exec*) tienen sobre los *bufferes* de almacenamiento.

Tema 8 Redireccionamiento

Lección 8.1 Redireccionamiento

Duración: 1 h de teoría

Objetivos: *Estudiar el concepto de redireccionamiento de la entrada/salida estandar y la forma en la que puede implementarse mediante el uso de llamadas al sistema.*

Descripción: La Lección 8.1, única de que consta este tema está dedicada a revisar el concepto de redireccionamiento de la entrada/salida de datos de un proceso pero ahora desde el punto de vista de su programación con llamadas al sistema. El concepto de redireccionamiento aplicado en el *shell* de UNIX ya fue aprendido por el alumno en la práctica de laboratorio SH2. En esta lección se vuelve a revisar la redirección pero ya como concepto general e integrándola con el gestor de la entrada/salida. Se comienza centrando al alumno en un tipo particular de procesos que por defecto manejan la entrada/salida estandar (los *filtros*). Se presentan los tres descriptores de fichero que todo proceso tiene predefinidos y asociados a la entrada de datos (*stdin*), salida de datos (*stdout*) y salida de errores (*stderr*). Se enlazan estos descriptores con su localización en las tablas que el sistema de ficheros UNIX maneja y que fueron estudiadas en el tema anterior. Se muestra de qué forma y por medio de la programación con llamadas al sistema, esas tres asociaciones predefinidas para todo proceso, pueden ser redireccionadas (modificadas). La lección acaba viendo cómo esta es la forma de actuar que también utiliza el proceso *shell* para la redirección, enlazando así el concepto general con lo que el alumno comenzó viendo en el laboratorio.

Tema 9 Comunicación elemental entre procesos

Lección 9.1 Tuberías. *Pipes*

Lección 9.2 Ejemplos de comunicación. *Shell*

Duración: 2 h de teoría

Objetivos: *Estudiar dos mecanismos UNIX para comunicación y sincronización entre procesos. Añadir ejemplos de como el* shell *utiliza esos mecanismos para comunicar.*

Descripción: Este tema plantea el estudio de nuevas técnicas para comunicar procesos entre si. El alumno ha aprendido en los temas previos como los procesos pueden intercambiarse información mediante el paso de descriptores de ficheros abiertos a través de las llamadas *fork* y *exec*, o a través del sistema de ficheros. Ahora, se le plantean otras técnicas más eficientes para realizar lo mismo y que son las utilizadas habitualmente en los sistemas para comunicar procesos. El tema se completa mediante ejemplos de utilización de las técnicas por parte del intérprete de comandos de UNIX (el *shell*).

En la Lección 9.1 se proponen dos de las soluciones que el sistema operativo UNIX soporta para comunicar procesos. Se introduce el concepto de tubería y se definen el tipo tubería con nombre (*named pipes* o *fifos*) y el tipo tubería sin nombre (*unamed pipes* o *pipes*), se muestran las características de cada una y también las diferencias que existen entre ellas. A continuación se estudian el conjunto de llamadas al sistema relacionadas con la creación y el acceso a las tuberías. Se ve el detalle de *mknod* y *pipe*, para crear *fifos* y *pipes* respectivamente y también se detalla el comportamiento de las llamadas *read*, *write* cuando el acceso es a una tubería. Todo lo anterior se integra viendo al final de clase un ejemplo de implementación en C donde se comunican dos procesos mediante el uso de una *pipe*.

En la siguiente Lección 9.2 se vuelve al ejemplo del *shell* de UNIX para mostrar de que forma actúa el intérprete de comandos para implementar una comunicación entre dos procesos solicitada por el usuario. Mediante un ejemplo de funcionamiento se va viendo como se lleva a cabo un determinado proceso de comunicación. Es este también un ejemplo útil para que el alumno acabe de entender de que forma el sistema operativo es el que resuelve la sincronización entre los procesos mediante el funcionamiento especial que tienen las llamadas *read*, *write* sobre las *pipes*. La lección y el tema concluyen mostrando y analizando dos ejemplos más de uso de *pipes*. Un primer ejemplo para solucionar alguna de las limitaciones de las *pipes* y otro de mal uso de las mismas.

MÓDULO 3 MEMORIA

Tema 10 - Gestión de Memoria

Tema 11 - Llamadas al sistema. Memoria en UNIX

Duración: 4 h de teoría y 2 h de problemas

Objetivos: *Estudiar los esquemas clásicos utilizados para la gestión de la memoria en un sistema operativo multiprogramado. Tras cursar este módulo el alumno habrá aprendido:*

- *Cuáles son los distintos algoritmos que son utilizados para gestionar una memoria que debe ser compartida por varios procesos, incluyendo el sistema operativo. El alumno aprende cuales son las más efectivas y se le presenta el concepto de* swap.
- *En que consiste el uso de la partición de la memoria y se le presentan las técnicas de paginación, segmentación y el concepto de la memoria virtual.*
- *De todas las estrategias, el alumno aprende cuales son las que se utilizan en UNIX y como se implementan.*

Descripción: Este módulo comienza presentando cuales son las distintas opciones para gestionar la memoria de un sistema multiusuario. La presentación se hace viendo los requisitos básicos que necesita cada una y cuales son las más efectivas según su adaptación al diseño *hardware* de cada sistema. A continuación, el alumno aprende las limitaciones que habitualmente posee la memoria y se le presenta el concepto de *swap* y de como el sistema operativo consigue de forma eficiente extender la zona de almacenamiento principal utilizando el almacenamiento secundario

(discos). Tras esta presentación se pasa al estudio del particionado de la memoria. Se estudian primero las técnicas de paginación y segmentación, formando así la base que necesita para a continuación introducir la gestión de la memoria virtual y los elementos *hardware* que la componen.

El último tema se dedica al estudio de aplicación de las estrategias anteriores para el caso de UNIX. El tema acaba presentando mediante ejemplos algunas de las llamadas al sistema y funciones de librería relacionadas con la gestión de memoria.

Tema 10 Gestión de Memoria

Lección 10.1 Gestión de memoria. Clasificación. Segmentación

Lección 10.2 Paginación. Memoria virtual

Duración: 2 h de teoría

Objetivos: *Poner de manifiesto el efecto negativo que en un sistema multiprogramado puede tener el acceso a memoria sobre el rendimiento. Aprender como el gestor de memoria permite minimizar ese efecto de forma eficiente mediante la carga y descarga de procesos. Ofrecer una visión general de los mecanismos más habituales que se utilizan para la gestión de memoria. Aprender el detalle de las dos técnicas consideradas base del diseño de cualquier sistema de gestión de memoria: segmentación y paginación. Estudiar cuales son los elementos* hardware *y de diseño del sistema operativo relacionados con la técnica más universal de gestión de memoria utilizada en las máquinas actuales: la memoria virtual.*

Descripción: Aunque ya es conocido para el alumno, este tema comienza en la Lección 10.1 definiendo los objetivos de todo sistema multiprogramado relacionándolos ahora con aquella parte que se encarga de la gestión de memoria. Se centran los requisitos básicos que debe soportar todo sistema para que el esquema de memoria que está siendo utilizado sea transparente a los procesos y permita a cada proceso comportarse como si tuviese una memoria ilimitada a su disposición. Se pone en evidencia la gran diferencia que sigue existiendo entre la lentitud que tiene todo acceso a memoria frente a la creciente velocidad del procesador. Por último, se comenta el concepto de *swap* usado en un sistema de tiempo compartido como herramienta para disponer de espacio extra cuando la memoria es insuficiente para contener todos los procesos.

El resto de la lección se dedica a presentar las distintas posibilidades que pueden encontrarse sobre la forma de residir los programas en memoria. Se centran las dos opciones de diseño típicas donde los procesos son guardados o bien enteros en memoria o bien parte en disco y parte en memoria. Para la primera opción de diseño se comenta en primer lugar el uso de la partición de memoria, técnica que aun no siendo de uso habitual sirve para aclararle al alumno muchos de los conceptos de diseño implicados en la gestión de memoria. Sirve también para presentar los conceptos de *reubicación*, *fragmentación* y *compactación*. A continuación y a través de la presentación de las ineficiencias asociadas al particionado, se introduce la técnica de *troceado de los programas* y sus dos posibilidades básicas: partir un programa por *segmentos* o por *páginas*. La lección finaliza viendo el detalle del diseño asociado a un esquema de división de los programas en *segmentos*.

En la Lección 10.2 se acaba de estudiar la opción de diseño en la que los programas se guardan enteros en memoria, viendo el detalle del diseño asociado al esquema que divide los programas en *páginas* y la memoria en *contenedores de páginas*. Se enumeran las diferencias básicas que existen con el esquema segmentado y mediante un ejemplo de funcionamiento se ven los elementos *hardware* de apoyo a la paginación. El mismo ejemplo de funcionamiento permite poner en evidencia la ineficiencia del mecanismo en cuanto al número de accesos a memoria que hay que realizar en cada traducción. Se presenta el TLB (*Translation Look-aside Buffer*) como elemento *hardware* útil para acelerar la traducción de direcciones. La lección y el tema terminan presentando la *memoria virtual paginada* como opción de diseño habitual para guardar parte de los programas en disco y parte en memoria. Se muestran tanto las necesidades *hardware* como *software* para soportar la memoria virtual y se finaliza viendo el concepto del *fallo de página*.

Tema 11 Llamadas al sistema. Memoria en UNIX

Lección 11.1 El sistema de memoria en UNIX. Regiones y tablas

Lección 11.2 Llamadas al sistema para memoria

Duración: 2 h de teoría

Objetivos: *Repasar las características esenciales de la memoria virtual paginada viendo cuales son las necesidades* hardware *del gestor de memoria del sistema operativo UNIX. Ejercitar con la programación en C desde la utilización de algunas de las llamadas al sistema y funciones de librería relacionadas con la gestión de memoria. Aprender las diferencias entre accesos a un fichero mediante el uso de la llamada* mmap *o mediante el uso ya conocido de* read *y* write.

Descripción: De nuevo y como ejemplo integrador de los conceptos que se le presentan al alumno, utilizamos UNIX para darle una segunda vuelta al concepto de la memoria virtual paginada, viendo cómo es su implementación en un gestor de memoria para el sistema operativo tomando como ejemplo. En la Lección 11.1 se presentan las distintas estructuras que usa UNIX para gestionar la memoria. Se introduce el concepto de *región* y se estudian las tablas manejadas en la gestión. Sobre los posibles esquemas de gestión de memoria se distinguen entre aquellos que están basados en *swap* y los basados en *paginación bajo demanda*. Como ejemplo de ellos, se muestra el detalle del funcionamiento de la llamada *fork* bajo los dos esquemas.

Para finalizar el tema, se vuelve a hacer hincapié en la programación en C dedicando la Lección 11.2 al estudio de las llamadas *brk* y *sbrk*, así como de las funciones de librería que también tienen que ver con el gestor de memoria (*malloc*, *calloc*, *realloc* y *free*). A continuación se presenta el concepto de *mapeo de ficheros en memoria* como ejemplo alternativo al acceso de los datos de un fichero almacenado en el disco. Se dedica especial atención a este modo de acceso porque en ocasiones puede resultar más efectivo a la hora de manipular datos, además de servir como ejemplo de uso de la memoria virtual. Esta forma alternativa de manejar la entrada y salida de datos se hace mediante el uso de la llamada *mmap* y se presenta como ejemplo que no requiere del uso de las llamadas que para manejar ficheros ya aprendió el alumno en el módulo de gestión de la entrada/salida (*read* y *write*). Se detalla la sintaxis completa de la llamada *mmap* y se ven ejemplos de su uso. Finalmente, se completa su estudio mediante un ejercicio desarrollado como final de la clase.

Ejemplo de ejercicio de final de clase:

- El objetivo de este ejercicio es presentar al alumno dos modelos de programación para manipular los datos de un fichero almacenado en disco. Para ello se le ofrece el código de dos programas en C que invierten el contenido de un fichero, mostrando por pantalla el resultado de dicha inversión. El primer programa hace la inversión de los datos usando las llamadas al sistema aprendidas en la gestión de ficheros, mientras que el segundo utiliza *mmap*. El alumno debe analizar ambos códigos y entenderlos. El ejercicio le permite repasar la sintaxis y función de las llamadas aprendidas en la gestión de ficheros, además de practicar con la nueva llamada aprendida en esta última lección.

2.4 Soporte bibliográfico para el alumno

En la descripción de los objetivos que pretende este temario se dijo como se le iba a mostrar al alumno el sistema operativo desde tres puntos de vista. Un primero de *gestión de recursos* para cubrir el aprendizaje de los fundamentos de diseño que hay detrás de todo sistema operativo, un segundo de *llamadas al sistema* para integrar estos conceptos aprendiendo una programación avanzada con llamadas al sistema, y un tercero de *nivel de usuario* para aprender a interaccionar con el sistema operativo a través de un intérprete de comandos. Para la visión de las llamadas al sistema, se justificó la elección del sistema operativo UNIX y del lenguaje de programación C. Esta es la razón por la que el temario de la asignatura propuesta en este proyecto va a seguir tres líneas de soporte bibliográfico.

Soporte bibliográfico para la línea de *gestión de recursos*

Para la línea de soporte al estudio de fundamentos de diseño, el libro recomendado es el de A. Silberschatz, P. Baer Galvin y G. Gagne “Operating System Concepts”. Es dificil encontrar un libro de texto que se ajuste al papel que en un determinado plan de estudios se espera que cumpla una asignatura. Además, también ocurre que el material de estos libros suele exceder con mucho los contenidos que pretenden abordarse. Sin embargo, un libro de texto de calidad contrastada constituye en sí una guía para el docente y una referencia siempre clara para el alumno. Tal y como ha sido propuesta la asignatura de este proyecto, ésta sigue claramente el mismo enfoque del libro de referencia citado. Además y como complemento el alumno también encontrará apoyo para los mismos temas en el libro en castellano de W. Stallings, “Sistemas Operativos”. Respecto de los contenidos, puntualizar que además de cubrir el detalle de todo lo presentado en la asignatura, los dos libros citados también cubren temas dedicados a seguridad, protección, sistemas distribuidos... En el plan de estudios que en este proyecto se ha tomado como ejemplo, estos temas se tratarán en asignaturas que el alumno tendrá en cursos posteriores.

Soporte bibliográfico para la línea de *llamadas al sistema*

Para la línea de soporte al estudio de los conceptos aprendidos desde la perspectiva del conjunto de llamadas al sistema UNIX disponibles para solicitar cualquier servicio que el sistema operativo ofrece, el libro recomendado es el de W. Richard Stevens, “Advanced Programming in the UNIX Environment”. Lo habitual en el estudio de este tipo de programación consiste en hacerse con el *Manual del Programador* proporcionado en cada sistema. La forma de explicar y el tipo

de estructura utilizados en estos manuales suelen limitarse a dar sólo la descripción de las distintas llamadas. Actuar de la misma forma iría en contra del tipo de aprendizaje que se le quiere proporcionar al alumno. Tal y como se ha propuesto en este proyecto docente, la presentación de las distintas llamadas se ha enfocado buscando la mejor respuesta en su aprendizaje por parte del estudiante. Para ello, se le da primero la idea fundamental que hay detrás de cada llamada, a continuación es cuando se le describe su sintaxis completa y se acaba apoyando la explicación con ejemplos concretos de funcionamiento. Esta es también la forma de funcionar que se sigue en el libro de referencia citado. Al final y una vez asimilado el detalle de todas las llamadas, será en las clases de laboratorio donde se presente al alumno el *Manual del Programador* del sistema sobre el que va a trabajar. Será este el momento en el que este tipo de manual ya le puede ser útil como consulta rápida de programación.

Para el aprendizaje del lenguaje de programación C, el libro de texto recomendado es el de Herbert Schildt, “C, Manual de Referencia”. Podría pensarse que cualquier referencia es útil como apoyo para aprender el lenguaje C. Sin embargo y de forma parecida a lo que se acaba de comentar en el caso de los manuales, para un alumno al que no se le exige conocimiento previo del lenguaje, no es suficiente con describirle sin más los distintos componentes del mismo. Es necesario además poder ofrecerle un libro de apoyo que esté bien estructurado y que añada a todas las descripciones razonamientos claros y ejemplos útiles de su uso. Esta es la forma de funcionar que se sigue en el libro de referencia citado y también la que se utiliza en la propuesta de curso de C que se hace en este proyecto. Como complemento el alumno encontrará más ejemplos en el libro de P. A. Darnell y P.E. Margolis, “Software Engineering in C”.

Soporte bibliográfico para la línea de *nivel de usuario*

Para la línea de soporte al estudio del nivel de usuario, el libro recomendado es el clásico de Kernighan y Pike, “The UNIX Programming Environment”. Libro hecho por dos de los programadores creadores de UNIX, y que se presenta como ejemplo muy útil para el estudio y uso de los comandos generales para interaccionar con el sistema, por su claridad y orden que lleva siempre en toda presentación. Aunque el libro de Kerninghan y Pike incluye también conceptos básicos de programación en *shell*, más recomendado para profundizar en este tipo de programación es el libro de Peters, “UNIX Programming”.

En definitiva, ante la pregunta de recomendar un libro de referencia, claramente sería el de A. Silberschatz, P. Baer Galvin, and G. Gagne “Operating System Concepts” para la parte de *gestión de recursos*, el de W. Richard Stevens, “Advanced Programming in the UNIX Environment” para la parte de programación con *llamadas al sistema*, y el de Kernighan y Pike para la parte del *nivel de usuario*. Como complemento a la parte de gestión el alumno encontrará apoyo para algunos temas básicos en otro libro: “Sistemas Operativos” de W. Stallings. La Tabla 2.3 señala para cada tema y lección las lecturas recomendadas refiriendo en cada caso un rango concreto de secciones, a fin de que el alumno utilice eficazmente el material, sin alejarse de la orientación del curso.

Muchas otras referencias se pueden dar como lecturas complementarias para alumnos especialmente interesados en la materia tratada en la asignatura. En este proyecto y con el objeto de primar el aprendizaje, se recomienda un único libro para cada una de las tres líneas de estudio. Sin

Tabla 2.3 Recomendaciones para la consulta bibliográfica del temario propuesto

Tema	Lección	Sil,Gal&G	Stall	Stevens	OTROS
		Secc.	Secc.		[rf]:Tma
0	0.1-0.2	1.1-1.5, 2.2			[1]:1
	0.3	13.2, 2.3	1.7		
1					[2]
2	2.1	3.1-3.2	3.1		
	2.2	5.1-5.2			
	2.3	5.3	9.2		
3	3.1			7.1, 7.4-7.6	
	3.2			8.3, 8-5-8.6, 8.8-8.9	
	3.3			8.2, 8.10	
4	4.1				[3]
5	5.1			10.1-10.2	
	5.2			10.3, 10.9-10.10	
	5.3			10.4-10.6, 10.8, 10.11-10.13	
	5.4			10.14	
6	6.1	13.2-13.5 10.1-10.3	12.6		
	6.2	A.7	12.7	3.1-3.2	
7	7.1			3.3-3.5,3.12	
	7.2			3.6-3.8, 4.2, 4.15	
	7.3			5.1-5.4	
8	8.1			3.10	
9	9.1-9.2	A.9		14.1-14.2, 14.5	[4]:6
10	10.1	8.1-8.3, 8.6	7.1-7.2		[1]:4
	10.2	8.4, 9.4, 9.1-9.2	8.1-8.2		
	10.3	A.6	8.4		
	10.4			7.8, 12.9	

[1] Tanenbaum, A.S. *Sistemas Operativos. Diseño e Implementación*

[2] Schildt, H. *C, Manual de Referencia*

[3] Peters, J.S. *UNIX Programming*

[4] Rochkind, M.J. *Advanced UNIX Programming*

embargo, también se le ofrece al alumno una serie de referencias que en algunos casos puntuales pueden servirle para darle una visión más amplia del tema, o un punto de vista distinto. Algunas de estas referencias son las que se han añadido en la columna OTROS de la Tabla 2.3. A continuación se hace una breve presentación de lo que puede aportar cada una de ellas.

El libro de Tanenbaum [1] es una referencia clásica sobre fundamentos de diseño de sistemas operativos. Este libro, de lectura asequible, cubre la mayoría de los conceptos presentados en la asignatura aunque lo hace desde un punto de vista meramente descriptivo. Sin embargo, resulta interesante la consulta del mismo por las referencias que en todo momento se hacen sobre el sistema *MINIX*. Este estudio detallado de *MINIX* se completa en los últimos apéndices del libro (del C al F) donde se incluye el listado completo de su código fuente. La presentación de una implementación concreta de sistema operativo, proporciona al alumno la posibilidad de observar de cerca como se pueden aplicar en un sistema real todos los principios de diseño estudiados.

Respecto de la línea de programación avanzada, en el libro de Rochkind [4] se pueden encontrar más ejemplos de programas usando llamadas al sistema. Es un libro bien estructurado y destaca la forma en la que se intentan introducir los distintos ejemplos mediante la aplicación de determinadas estrategias. Aunque la lectura considerada definitiva para esta línea de estudio sigue siendo el libro de Stevens, el de Rochkind no deja de ser interesante para aquellos alumnos que quieran practicar más sus conocimientos e incluso conocer otras opciones de diseño. Algunos de los ejemplos que presenta este libro en su capítulo 6 de comunicación entre procesos, son utilizados como ejemplos en la exposición de la asignatura.

El libro de Peters [3] referenciado para el Tema 4, aún cuando no es estrictamente un libro de programación usando llamadas al sistema, es útil al tema por la visión que da de UNIX a nivel de usuario. En concreto, a lo que sirve este libro es al apoyo del estudio del *shell de UNIX* y de la *programación en shell*. En la propuesta que se ha hecho en este proyecto, la parte más básica del estudio del *shell* se realiza en las dos primeras prácticas de laboratorio (ver sección 2.6), pero no se dedica a la *programación en shell*. Este libro sería una buena referencia para el alumno que desee profundizar en este nuevo tipo de programación.

Por último, resulta necesaria la referencia a algún libro que trate del actual sistema de libre acceso *LINUX*, al ser este también el utilizado habitualmente por los alumnos. El libro de D.P. Bovet y M. Cesati, “Understanding the LINUX Kernel”, es la referencia más adecuada. Aunque el libro de Tanenbaum referenciado al principio era una cita obligada como clásico de sistemas operativos, el sistema *MINIX* que allí se utiliza para el estudio sobre implementación interna, queda algo alejado en el tiempo. El de Bovet y Cesati sería el equivalente a este estudio de implementación pero ahora para el sistema operativo más utilizado en la actualidad (*LINUX*).

2.5 Clases de Problemas

Objetivos: *Profundizar en algunos conceptos puntuales de los aprendidos en clase mediante la resolución de pequeños problemas. Practicar técnicas de trabajo en grupo.*

Según la distribución presentada en este temario (ver resumen en el apartado 2.7), la asignatura se ha planificado disponiendo de 10 horas durante todo el cuatrimestre para clases de problemas. Son clases que se utilizarán para plantear y resolver pequeños problemas en grupo. Estas clases no se distinguen físicamente de las de teoría y se considera que el número de grupos para ambas es el mismo. Esto hará que el número de alumnos pueda resultar demasiado alto para una clase de problemas tal como se concibe en nuestros días (en torno a los 100). Esta deficiencia se cubrirá en lo posible mediante el uso de técnicas de trabajo en grupo.

Las horas de clase de problemas se dedican a poner en común el trabajo previo que cada alumno ha realizado individualmente. Para ello, durante los días previos a cada clase de problemas se planificará el trabajo a realizar por el alumno como preparación a dicha clase. Los enunciados de los problemas a resolver pueden ser en algunos casos entregados en clase unos días antes de la clase de problemas. Otras veces pertenecen a ejercicios que dan apoyo al Tema 1 del curso de C o a enunciados de exámenes tipo.

El alumno deberá resolver el problema de forma individual. Durante la clase de problemas discutirá su solución enfrentándola a las de otros 4 alumnos, y llegando a un acuerdo de grupo que será presentado a la clase. El profesor es el encargado de supervisar el proceso y solucionar las dudas de grupo durante la clase de problemas.

A continuación se describen brevemente algunos de los problemas planteados para SO I. En cada uno de ellos se indica el módulo al que pertenece y el tiempo asignado. El orden temporal entre clases de teoría y problemas se presenta en el apartado 2.7.

Problema 1 Problema de C

1 hora de problemas

Problema dedicado a la práctica del lenguaje de programación C y al manejo de la línea de comandos desde programa.

El enunciado del problema guía al alumno a diseñar un programa escrito en C que se comporte de forma que toda la entrada estandar recibida por el programa aparezca sobre la salida estandar, sustituyendo la aparición de un cierto patrón1 de caracteres por otro patrón2. Ambos patrones se leen como argumentos desde la línea de comandos.

Añadida a la implementación anterior, se le exige al alumno una serie de reglas sobre la estructura que debe seguir en el diseño del programa. Para practicar con la programación estructurada en C, se le puede pedir lo siguiente:

- El programa principal será el encargado de leer línea a línea la entrada estandar recorriendo cada línea carácter a carácter y llamando a una función que compara *strings*.
- El programa debe contener una función que recibirá como parámetros dos punteros a *string* y devolverá cierto si los primeros caracteres del primer *string* coinciden con el segundo y falso en caso contrario.

Ejemplo de funcionamiento:`->programa ab 012`

Debe escribir en pantalla lo leído desde teclado sustituyendo cada aparición de “ab” por “012”.

Problema 2 Control de procesos. Permisos

Módulo 1: 1 hora de problemas

Los dos ejercicios que se plantearon al final del Tema 3. El primero dedicado a la visualización del *interleaving* entre procesos. El segundo dedicado a entender el funcionamiento del bit *set-user-id*.

El enunciado del problema se divide en dos partes. En la primera parte se muestra el código de un programa en C que utiliza llamadas que tratan con el control de procesos. Partiendo de este programa y de los conceptos que se acaban de presentar en clase, el alumno debe adelantar cuál será el resultado de la ejecución del programa anterior.

La segunda parte del enunciado muestra los permisos que habitualmente tiene asociados el programa */bin/passwd*:

```
-r-sr-sr-x   1 root     sys        23500 Aug  6  2003 /bin/passwd
```

y se le pide al alumno que explique cómo funciona el *bit s* que aparece en el permiso de ejecución de dicho programa.

Problema 3 Intérprete de comandos

Módulo 1: 1 hora de problemas

En este problema el alumno debe repetir el diseño del intérprete de comandos más completo que se le presentó en el Tema 4, detectando y resolviendo los problemas que aparecen en dicho intérprete cuando se ejecuta en modo asíncrono.

Problema 4 Programación con señales

Módulo 1: 1 hora de problemas

El enunciado del problema se divide en dos partes. Para cada parte se le presenta al alumno el código de un programa escrito en C que utiliza llamadas al sistema relacionadas con el modelo de señales presentado en el Tema 5. En el primer programa el alumno debe detectar las ocasiones en las que la ejecución del programa puede fallar y decir por qué. Sobre el segundo programa, se le pide al alumno que analice su comportamiento y adelante cuál será el resultado de la ejecución del mismo.

Problema 5 El Sistema de Ficheros

Módulo 2: 1 hora de problemas

El enunciado del problema describe un sistema de ficheros UNIX que utiliza bloques en disco de 4Kbytes y donde la dirección de cada bloque se expresa mediante un entero de 4 Bytes. Se indica también que la estructura que utiliza este sistema en cada `i-nodo` posee: ocho punteros directos, un puntero simple indirecto y un puntero doble indirecto. Se pide explicar con un ejemplo cual sería la forma utilizada en este sistema para localizar el contenido de un fichero y cuál sería el tamaño máximo de un fichero en este sistema.

Problema 6 Diseño sobre ficheros

Módulo 2: 1 hora de problemas

Este problema pretende que el alumno utilice de forma eficiente el conjunto de llamadas al sistema relacionadas con la gestión de ficheros, las cuales acaba de aprender en el Tema 7. Para ello se le pide que implemente un programa similar al comando `mv` de UNIX.

El enunciado del problema indica que el programa a diseñar deberá estar preparado para poder recibir dos parámetros. El primer parámetro será el nombre de un fichero existente que desaparecerá después de ejecutar el programa. Como resultado de la ejecución aparecerá un nuevo fichero con los mismos contenidos que el anterior y con nombre igual al segundo parámetro. Al alumno se le exige que la implementación sea óptima y que codifique el programa en no más de diez líneas de código.

Problema 7 Redireccionamiento

Módulo 2: 1 hora de problemas

Este problema pide al alumno que escriba una aplicación (`redi.c`) para redireccionar la salida estandar y la salida de error de un programa hacia un fichero cuyo nombre se le pasa como primer argumento. El nombre del programa a ejecutar y sus parámetros, se le pasarán a `redi` en el segundo y sucesivos argumentos.

Ejemplo de uso:`->redi salida ls -l pepe`

El programa `redi` debe redireccionar hacia el fichero `salida` (creándolo si no existe o truncándolo) las salidas estandar y de error que genere el comando `ls -l pepe`.

En el diseño de `redi` se deben tener en cuenta las siguientes pautas:

- el comando que se va a ejecutar podrá producir salida estandar y salida de error mezcladas en el tiempo
- la ejecución de `redi` producirá un sólo proceso
- el programa `redi.c` tiene un máximo de diez líneas

Problema 8 Comunicación entre procesos

Módulo 2: 1 hora de problemas

Para acabar de entender el mecanismo de comunicación entre procesos visto en el Tema 9, se le pide al alumno que escriba un programa en C que tratará la salida producida por la ejecución de otro proceso. En particular el programa debe ser capaz de hacer finalizar a todos los procesos del usuario que se encuentran activos en el momento de su ejecución. Como ayuda se le recuerda el formato que posee en su salida el comando

```
ps -lu usuario
```

```
F S  UID  PID  PPID  C PRI  NI     ADDR   SZ  WCHAN   TTY  TIME COMD

1 S  243 7152 7046  0 158  20  4004dec0  58 1d79e12  ttya  0:00  ksh

1 R  243 7788 5654 10 180  20  4007b200  16          ttya  0:00   ps
```

Para un funcionamiento correcto del programa se exige también que se dejen activos al propio programa, a su proceso hijo `ps -lu usuario` y a su padre `ksh`.

Problema 9 *Malloc* y señales. Funciones no reentrantes

Módulo 3: 1 hora de problemas

El enunciado del problema ofrece el código en C de un programa donde se mezcla la utilización de *malloc* con la programación de captura de una determinada señal. El alumno debe identificar que es lo que funciona mal en el programa presentado y escribir el código que resuelva las ineficiencias encontradas.

Problema 10 Implementación de *malloc* y de *free*

Módulo 3: 1 hora de problemas

Este problema completa el estudio realizado en clase sobre la reserva dinámica de memoria virtual en un sistema UNIX (`brk` y `sbrk`). Se pide desarrollar un interface de mayor potencia para crear y destruir zonas de memoria dinámica por parte de un proceso de usuario. Se trata de implementar las funciones malloc() y free(), con la misma sintaxis que las de la librería de C y cumpliendo las siguientes pautas:

- el espacio gestionado por las dos funciones se crea de forma dinámica usando las llamadas al sistema conocidas
- el espacio liberado mediante free() nunca se devuelve al sistema y queda bajo control del programa que se está diseñando hasta la finalización del proceso
- la función malloc() intentará reutilizar el espacio liberado por el usuario en anteriores llamadas a free(). Sólo se pedirá más espacio al sistema en caso de no disponer del suficiente

En el desarrollo del problema se le guía al alumno para que vaya paso a paso, estudiando primero los algoritmos básicos que utilizará para gestionar el espacio de `heap` y creando después las estructuras de datos que necesitará. Finalmente, se le pide que escriba el código para las dos funciones, permitiéndole usar otras funciones que supuestamente ya existen y que le servirán para manipular las estructuras de datos que ha definido. Para las funciones supuestas sólo se le pide que defina su cabecera, parámetros, resultado y un pequeño comentario que explique qué hacen.

2.6 Clases de Laboratorio

Objetivos: *Conocer y aprender a utilizar los comandos básicos que interpreta el* shell *de UNIX y que permiten manejar ficheros y procesos. Consolidar los conocimientos que sobre programa-*

ción avanzada en C se han aprendido en clase. Adquirir práctica de programación utilizando llamadas al sistema en toda interacción entre un usuario avanzado y los gestores del sistema operativo.

Para cubrir los créditos que SO I tiene asignados para laboratorio (15 horas), se va a suponer que la asignatura dispone de una sesión de laboratorio de 1 hora por semana y grupo de prácticas. En el apartado 2.7 se puede consultar como encajan las prácticas en la propuesta de planificación que se hace de la asignatura. En esta planificación se ha supuesto un calendario de 15 semanas y se han propuesto un total de 7 prácticas de 2 horas de duración cada una.

En la propuesta de clases de laboratorio que se hace en este proyecto, las dos primeras prácticas (4 horas) serán dedicadas a cubrir el aprendizaje del intérprete de comandos que va a ser utilizado en las siguientes sesiones (el *shell*). Las clases restantes se dedican a que el alumno ponga en práctica los conceptos aprendidos en clase de teoría además de ejercitarse en el uso de las herramientas del intérprete aprendidas en la primera práctica. De las siguientes cinco prácticas propuestas, dos se dedican a conceptos aparecidos en el módulo 1 (práctica1: procesos y práctica2: señales) y otras dos a conceptos del módulo 2 (práctica4: ficheros y práctica5: comunicación entre procesos). La práctica3 puede considerarse como una práctica extra y está relacionada con el aprendizaje de herramientas para desarrollo de programas (*ar* y *make*). Todas las prácticas se plantean pidiendo al alumno el diseño de programas en C cuya especificación está relacionada con el tema que se quiere ejercitar.

A continuación se describen brevemente las prácticas planteadas para SO I.

PRÁCTICA *SHELL* 4 HORAS

Objetivos: *Consolidar aquella parte de los conocimientos que trata de la forma en la que un usuario normal interacciona contra el sistema operativo a través de un intérprete de comandos. Conocer las características que tiene el* shell *de UNIX y aprender los comandos necesarios para manejarlo.*

Descripción: Esta práctica cubre la visión que un *usuario normal* tiene sobre un sistema operativo concreto. Precisa de trabajo en el laboratorio y facilitará que el alumno pueda manejarse de forma rápida y eficiente con el sistema a utilizar.

La práctica *shell* consta de dos partes. En la primera al alumno se le presentan los comandos que necesitará para moverse por el sistema de ficheros que reside en la máquina que va a usar para trabajar el resto de los problemas. En la segunda se le presentan los comandos relacionados con el manejo de procesos en dicha máquina. Para ambas se le presenta el concepto de *línea de comandos*.

Para las dos prácticas, la especificación que recibe el alumno como guión es autocontenida. Para cada uno de los apartados el guión le presenta primero una descripción detallada de los comandos que va a ver y segundo un conjunto de pequeños ejercicios que el alumno debe ir resolviendo. La resolución de estos ejercicios le obligará a probar y testear en la máquina los comandos presentados permitiéndole de esta forma consolidar lo aprendido.

PRÁCTICA *SH1* 2 HORAS

Presenta cual es la *visión de usuario* que se va a tener del sistema de ficheros. Recopila información para estudiar los comandos que le permitirán manejarse con el sistema de ficheros. El comportamiento de cada comando lo va viendo el alumno a base de pruebas de uso. La información proporcionada cubre los conceptos de fichero y directorio. Se ven los comandos para copiar, mover, visualizar, permisos de acceso para ficheros y directorios... Se presenta también el concepto de *i-nodo* y *link* o *enlace*.

PRÁCTICA *SH2* 2 HORAS

Presenta cual es la *visión de usuario* que se va a tener sobre los procesos. Recopila información para estudiar los comandos que le permitirán manejarse con los procesos. El comportamiento de cada comando lo va viendo el alumno a base de pruebas de uso. La información proporcionada le muestra las opciones y características básicas que posee el *shell*: bucle capaz de ejecutar comandos de distinto tipo (binarios ejecutables, comandos internos, ficheros de comandos), paso de parámetros, variables de entorno, ejecución síncrona/asíncrona, caracteres comodín... Se ven los comandos para acceder al compilador de C, redireccionar la entrada/salida de los procesos y la forma en la que se puede obtener información sobre los procesos. La práctica acaba estudiando el editor *vi*.

PRÁCTICA 1 2 HORAS

Objetivos: *Consolidar los conocimientos sobre el conjunto de llamadas al sistema UNIX para manejar procesos. Aplicar estos conocimientos realizando una programación avanzada en lenguaje C para solicitar al sistema operativo servicios de los que éste le ofrece sobre los procesos.*

Descripción: Esta práctica consta de dos partes. Primero se pone a disposición del alumno el código de un ejemplo visto en clase mediante el que se generaban estados no deseados en los procesos (*zombies* y *huérfanos* de la Lección 3.2). Se le pide que compruebe la generación de dichos estados y que encuentre la forma de identificarlos. En esta primera parte y además de consolidar conceptos sobre procesos, el alumno debe aplicar sus conocimientos aprendidos en la práctica de *shell* al tener que hacer uso de la mayoría de las opciones del *shell* allí presentadas.

En la segunda parte de la práctica, el alumno va a ejercitar sus conocimientos tanto sobre el lenguaje de programación C como sobre el conjunto de llamadas relacionadas con procesos. Debe implementar un programa (*paralelo.c*) que ejecute dos aplicaciones de forma concurrente. Los nombres de las aplicaciones y sus parámetros los recibirá el programa por la línea de comandos y para separar la especificación de cada aplicación se utilizará el símbolo +. La implementación debe ser en lenguaje C y utilizando llamadas al sistema relacionadas con procesos.

Ejemplo de funcionamiento: Tras ejecutar la siguiente línea de comandos:

```
_$ paralelo ls -l *.c + ps -u usuario1
```

deberá ejecutarse `ls -l *.c` de forma concurrente con `ps -u usuario1`

PRÁCTICA 2 2 HORAS

Objetivos: *Consolidar los conocimientos sobre el conjunto de llamadas al sistema UNIX para tratar con señales. Aplicar estos conocimientos realizando una programación avanzada en lenguaje C manejándose con el uso de llamadas para señales.*

Descripción: Al alumno se le pide primero que construya una función cuyo funcionamiento es similar al de la llamada *sleep*. La función se llama *duerme* y actúa dejando bloqueado al proceso que la utiliza durante el número de segundos que la función recibe por parámetro:

```
duerme(segundos)
```

A continuación, debe implementar un programa capaz de recibir dos argumentos numéricos en su línea de comandos (*x* e *y*) y que actúe lanzando a pantalla un mismo mensaje cada *x* segundos y durante un tiempo total de *y* segundos. El programa debe hacer uso de la función *duerme* diseñada en la primera parte de la práctica. La implementación será siempre en lenguaje C y utilizando llamadas al sistema relacionadas con señales.

PRÁCTICA 3 2 HORAS

Objetivos: *Afianzar los conocimientos sobre punteros en C. Practicar con las herramientas para el desarrollo de programas:* ar *-mantenedor de librerías- y* make *-mantenedor de programas-.*

Descripción: Esta práctica consta de dos partes. Primero el alumno debe implementar dos funciones para el tratamiento de *strings* que normalmente no están incluidas en la librería estandar. Una vez escritas y probadas, las funciones se deben guardar como objetos separados en una librería (*milib.a*). Para esta última parte, al alumno se le proporciona el manual de uso del comando *ar*. Las funciones que debe diseñar son las siguientes:

- `int itoa ( int i, char *buff )`

Función que debe generar un *string* que se corresponda con el valor decimal (expresado en caracteres *ASCII*) del número entero que recibe como primer parámetro. Devolverá como resultado la longitud de dicho *string*.

- `int parser ( char *origen, char *destino[ ] )`

Función que debe partir el *string* que recibe como primer parámetro en palabras separadas por blanco ó tabulador. Cada una de las palabras las debe dejar en un elemento del vector indicado en su segundo parámetro. Devolverá el número de palabras en las que ha partido el *string*.

En la segunda parte de la práctica, el alumno va a probar el funcionamiento de la librería que acaba de crear. Para ello, debe diseñar ahora un programa que lea de su entrada estandar una serie de números expresados en decimal, calcule su suma y la escriba en la salida estandar. Para finalizar, el alumno debe crear un fichero *practica.make* encargado de construir el ejecutable del programa y de actualizar la librería *milib.a* en caso de que haya cambios en los fuentes de las funciones `itoa` y `parser`. Para esta última parte, se le proporciona el manual de uso del comando *make*.

PRÁCTICA 4 2 HORAS

Objetivos: *Consolidar los conocimientos sobre el conjunto de llamadas al sistema UNIX para manejar ficheros. Aplicar estos conocimientos realizando una programación avanzada en lenguaje C para solicitar al sistema operativo servicios de los que éste le ofrece para tratar con la entrada/salida a ficheros y dispositivos.*

Descripción: En esta práctica se va a implementar una aplicación parecida al comando *cat* para visualización y que el alumno conoce de la práctica de *shell*. La práctica tiene cuatro apartados y se le pide al alumno que vaya desarrollando cada uno de ellos de forma incremental sobre un mismo código. De esta forma, al final deberá haber creado un único programa pero que incluirá la funcionalidad marcada en cada apartado. Los cuatro apartados son los siguientes:

- Diseñar un programa que lea la entrada estandar y la escriba sobre la salida estandar utilizando solamente llamadas al sistema para el tratamiento de la entrada/salida (*open*, *read*, *write* y *close*).
- Modificar el programa anterior para que se le pueda pasar un parámetro que indicará el nombre de un fichero (o dispositivo) destino. El programa actuará como en el apartado anterior si no recibe ningún parámetro, pero copiará lo leído desde la entrada estandar sobre el fichero si se le indica su nombre a través del parámetro.
- Añadir de nuevo al programa del apartado 2 el código necesario para que se recojan los valores devueltos por todas las llamadas al sistema que se realizan. El programa deberá comprobar para cada caso la existencia de errores y devolverá mensajes por la pantalla del usuario que indiquen dichos errores.
- Modificar de nuevo el programa para que si ahora la salida se dirige a un fichero, la información se añada a la ya existente en ese fichero. El fichero a donde dirigir la salida puede ser pasado como parámetro o a través de redirección.

La implementación debe ser en lenguaje C y utilizando llamadas al sistema.

PRÁCTICA 5 2 HORAS

Objetivos: *Consolidar los conocimientos sobre el conjunto de llamadas al sistema UNIX relacionadas con la gestión y comunicación entre procesos. Aplicar estos conocimientos realizando una programación avanzada en lenguaje C manejándose con la creación y uso de* pipes*.*

Descripción: Implementar una aplicación en la que se lancen dos procesos que se irán pasando una pelota. Esta pelota consiste en un número que cada proceso va a incrementar en uno antes de

volverla a pasar. Como elemento transmisor de la pelota entre los dos procesos se deben utilizar tuberías. Cada proceso estará ejecutando un bucle donde:

- lea la pelota
- espere un segundo
- incremente el valor de la pelota leída
- escriba la pelota

Los dos procesos deben morir cuando el contador de una de las pelotas alcance un determinado valor (por ejemplo 100). El programa además debe permitir visualizar como va la evolución del juego. Para ello cada proceso debe lanzar un mensaje a pantalla en el momento que recibe una pelota indicando dos cosas: el valor que tiene la pelota y el identificador del proceso que la ha leído.

Como parte final de la práctica y una vez que el juego esté finalizado hay que realizar la siguiente modificación:

- Añadir el código necesario para poder incluir nuevas pelotas en el juego. Cada vez que uno de los procesos reciba la señal SIGUSR1 deberá poner en marcha una nueva pelota con contador inicial a cero. Cada vez que uno de los procesos reciba la señal SIGUSR2 deberá poner a cero el contador de la primera pelota que le llegue.

La implementación de nuevo deberá ser en lenguaje C y usando llamadas al sistema relacionadas con comunicación entre procesos y señales.

2.7 Planificación de la materia

La Tabla 2.4 recoge el detalle de la propuesta de planificación semanal para SO I. De la dedicación de 4 horas a la semana, las dos primeras son para las lecciones de teoría (26 lecciones en total) y de las dos restantes, una se dedica a problemas y la otra a laboratorio. Los créditos de teoría se cubren con las 26 horas de lecciones, las 4 horas del curso de C y 10 horas de clases de problemas. Se dedican también 3 horas a mitad de curso y otras 3 al final, para realizar un examen parcial y otro final como parte de la evaluación de la asignatura.

Sobre la base de la metodología docente que se expone en el Capítulo 3, en esta planificación se ha priorizado la sincronización entre prácticas y teoría, evitando que se aborden en laboratorio cuestiones aún no fundamentadas teóricamente, o que exista un desfase excesivo entre lo que se está explicando en teoría y lo que se trabaja en las clases de laboratorio. Además y aunque en esta propuesta se ha supuesto un solo grupo tanto para las clases de teoría como para las de problemas y las de laboratorio, para las prácticas en el laboratorio es recomendable una división en subgrupos de unos 20 alumnos. Esto último implicará la necesidad de aumentar la dedicación horaria semanal para impartir las prácticas al número de grupos resultado.

Tabla 2.4 Propuesta de planificación de Sistemas Operativos I

Semana	Teoría		Problemas		Laboratorio		Total
	H	Contenidos*	H	Contenidos	H		
1	2	T0(0.1,0.2)	2	T1: Curso de C			
2	2	T0(0.3); T2(2.1)	1	T1: Curso de C	1	Práctica SH1	
3	2	T2(2.2,2.3)	1	T1: Curso de C	1	Práctica SH1	
4	2	T3(3.1,3.2)	1	Problema de C	1	Práctica SH2	
5	2	T3(3.3); T4	1	Problema 1	1	Práctica SH2	
6	2	T5(5.1,5.2)	1	Problema 2	1	Práctica 1	
7	2	T5(5.3,5.4)	1	Problema 3	1	Práctica 1	
8	2	Examen parcial	1	Resolver parcial	1	Práctica 2	
9	2	T6(6.1,6.2)	1	Problema 4	1	Práctica 2	
10	2	T7(7.1,7.2)	1	Problema 5	1	Práctica 3	
11	2	T7(7.3); T8	1	Problema 6	1	Práctica 3	
12	2	T9(9.1,9.2)	1	Problema 7	1	Práctica 4	
13	2	T10(10.1,10.2)	1	Problema 8	1	Práctica 4	
14	2	T11(11.1,11.2)	1	Problema 9	1	Práctica 5	
15	3	Examen Final			1	Práctica 5	
Totales	32 H	26 lecciones de 1h, 6h examenes	14 H	4h curso C, 10h problemas	14H	7 prácticas x 2h	60H

* Tn: Teoría del Tema n; se indican en paréntesis las lecciones.

2.8 Propuesta de evaluación

Asumiendo todos los aspectos generales que se expusieron en la Sección 2.4, las fuentes de información de que se dispone para evaluar a un alumno en esta asignatura son:

- Un examen parcial de teoría (**T1**)
- Un examen final de teoría (**T**)
- Un examen final de prácticas (**P**)

La composición de la nota final va a tener en cuenta los tres componentes señalados, ponderados convenientemente. Se propone lo siguiente:

N = (0.10 x **T1** + 0.70 x **T** + 0.20 x **P**)

T1 = nota del examen parcial de teoría

T = nota del examen final de teoría

P = nota del examen de prácticas

La relevancia dada a las prácticas como método de aprendizaje, debe quedar reflejada en la composición de la nota. Sin embargo, toda prueba teórica incluye además conceptos del temario no integrados en las prácticas. Por otra parte la mayoría de los estudiantes realizarán la práctica en parejas, y aunque el examen final de prácticas es individual, es defendible pensar que una prueba teórica refleja mejor el nivel de una persona.

La razón de una primera prueba teórica a mitad de la asignatura (parcial **T1**) más que querer desviar la atención del alumno sobre el método de trabajo marcado por las clases (teoría, problemas y laboratorio), al tratarse de una asignatura cuatrimestral, tiene como objetivo el de que el alumno tome esta prueba en consideración para no descuidar su seguimiento completo de la asignatura comenzando éste lo antes posible.

3
Metodología Docente

Este capítulo introduce el modelo pedagógico en el que se apoya el proyecto defendido en la guía docente del Capítulo 2, comenta los objetivos básicos de la metodología docente que se sigue, y discute las técnicas didácticas habituales en el entorno docente presentado en el Capítulo 1. Los objetivos y técnicas específicas relacionadas con la asignatura de Sistemas Operativos I se concretan en la propia guía docente.

3.1 Introducción

Cuando el objetivo es la formación de profesionales y personas competentes, la tarea docente se convierte en un ejercicio de gran responsabilidad al ir más allá de la mera transmisión de unos conocimientos específicos en determinada materia. Voluntaria o involuntariamente, por su propia naturaleza, transmite y fomenta actitudes, hábitos de trabajo buenos o malos, intereses o desintereses. Además tiene la responsabilidad de asegurar el *aprendizaje* de ciertos conceptos y habilidades. Aparece así la doble identidad profesional del docente que debe no sólo ser experto en su disciplina sino serlo también en la docencia de la misma.

3.1.1 Modelo del profesor universitario

El papel del profesor en este entorno, pasa a tener más funciones de tutor y menos de transmisor de la información ahora disponible para todos. Su trabajo consiste en estimular el proceso de aprender a base de guiar al alumno en la asimilación, compresión y utilización de la información que se le presenta. Debe ayudar al alumno a construir su conocimiento de forma correcta, proporcionándole el andamiaje necesario y debe ir haciéndose cada vez más innecesario dentro del proceso de aprendizaje. El resultado final podrá considerarse efectivo si el alumno consigue asumir

todo el protagonismo e iniciativa en el aprendizaje que demanda. De forma más concreta, son tareas del profesor:

- Ayudar al alumno a seleccionar la información
- Utilizar los conocimientos previos del estudiante
- Atender al alumno en todas las fases del aprendizaje
- Ocuparse no solo de lo que el alumno aprende (producto) sino también de como lo aprende (proceso)
- Evaluar tanto el producto como el proceso, dando más importancia a la evaluación formativa en lugar de centrar la atención sólo en la evaluación final o sumativa

En una cultura donde se busca calidad en el aprendizaje y excelencia de la enseñanza (ref. M. de Miguel 99), es necesario que el docente conozca mínimamente los posibles estilos de aprendizaje de los alumnos, adecuando su *estilo docente* al *proceso discente*. Pero el núcleo del proceso enseñanza-aprendizaje en educación universitaria está condicionado por las siguientes variables:

- Relativas al alumno, los enfoques y estilos de aprendizaje: habilidades intelectuales, conocimientos y concepciones previas, hábitos de trabajo intelectual, estilos cognitivos y rasgos de personalidad.
- Relativas al docente, el estilo de enseñanza: métodos y habilidades docentes, empatía y relación con el alumno.
- Relativas al contexto institucional: procedimientos de evaluación, carga de trabajo, realimentación, materiales y condiciones físicas para el aprendizaje.

Este último grupo de variables determinan considerablemente la metodología docente y de evaluación. Aunque la calidad de la enseñanza universitaria esté asegurada mediante la formación y profesionalización del profesor (ref. Elton 93), una mejora en la calidad de la docencia exige siempre la complementariedad entre el esfuerzo institucional y el compromiso personal de los profesores. En los planteamientos que siguen y teniendo en cuenta estas limitaciones, se propone una metodología docente basada en el modelo enseñanza-aprendizaje, modulada por mis años de experiencia docente. El apartado final de este capítulo lo dedicaré a mostrar algunos apuntes sobre un posible proceso de autoevaluación de la metodología docente.

3.1.2 El modelo enseñanza-aprendizaje

El objetivo último de la tarea docente es el *aprendizaje*, no la *enseñanza*, entendido el *aprendizaje* como un proceso de cambio en la forma de pensar, sentir y actuar del estudiante en el que influyen sus conocimientos previos, su forma de aprender, su capacidad intelectual, la percepción que tiene de sí mismo, sus metas personales, e incluso el entorno social, tanto el inmediato (compañeros de clase, centro docente) como el genérico (sociedad en general). En este contexto, la *enseñanza* se va a entender como la forma de estimular en el alumno ese cambio.

Dos son los enfoques que existen sobre el aprendizaje, el *conductal* y el *cognitivo-constructivista*. En un enfoque conductal, la estimulación del aprendizaje se hace utilizando determinadas téc-

nicas diseñadas para el control de la conducta. El progreso en el aprendizaje se basa en la repetición y el resultado (respuesta) en la reproducción de lo aprendido (refs. Morris L. Bigge 82, y D. Delprato and B. Midgley 92). Por otro lado, un punto de vista cognitivo-constructivista deja de ver al alumno como un mero receptor pasivo de la información y pasa tener una participación activa en el proceso de aprendizaje. Sobre el aprendizaje universitario es el modelo cognitivo-constructivista uno de los más extendidos de enseñanza-aprendizaje (refs. Ausubel 89, Bernad 90 y Entwistle 87) y es también el modelo en el que he apoyado esta metodología docente y buena parte de los aspectos de esta memoria.

En el modelo de enseñanza-aprendizaje cognitivo el núcleo no es la *enseñanza* sino el *aprendizaje*. El saber *se construye*, el alumno es un elemento *activo* y protagonista y el profesor es el facilitador del aprendizaje.

El *aprendizaje* se concibe como un cambio estable en lo que sabe, hace y piensa el aprendiz. Es este un proceso complejo que consiste en la construcción de significados por parte del aprendiz a partir de la información que recibe, y de acuerdo con sus propias estrategias y esquemas previos (aprendizaje *autónomo*). En tal proceso intervienen decisivamente factores derivados de rasgos o condiciones del alumno como el funcionamiento del sistema cognitivo humano y el dominio de estrategias de aprendizaje. También intervienen factores externos como la peculiar estructura de los contenidos de la materia que se estudia (leyes, principios, métodos de cada materia o área de conocimiento) y la ayuda recibida del exterior (eficacia del sistema educativo, preparación del profesor, planes de estudio, etc.). Es una *construcción* del conocimiento específica del contenido, del sujeto y de su entorno y no existe un único modelo metodológico a seguir. Sin embargo, existen cuatro principios generales para esta metodología:

- estructurar el conocimiento usando el concepto de *red* y no el de jerarquía
- construir un conocimiento *social* frente al conocimiento solitario
- aprender *contextualizando* sobre problemas reales frente a los contenidos genéricos y artificiales tradicionales y
- transferir de forma gradual la *responsabilidad* del aprendizaje del profesor al alumno

He enfocado mi metodología docente sobre el aprendizaje, sobre estos cuatro principios generales y lo he centrado en las circunstancias reales en las que se imparte la docencia. Con esto pretendo evitar dos extremos posibles: uno el de la mediocridad e indolencia en el aspecto docente, otro el de los planteamientos irrealizables en el contexto universitario público español. A continuación mostraré la metodología general que he usado como base para la planificación de la materia cuyos contenidos se acaban de detallar en el capítulo anterior.

3.2 Metodología general

En el proceso aprendiz-enseñanza y para conseguir un aprendizaje eficaz (constructivo, activo, cooperativo, autónomo, contextualizado) es necesario que el alumno vaya ganando la altura que va perdiendo el profesor. Debe pasarse de un aprendizaje solo dirigido por el profesor a un aprendizaje *autodirigido* (autogestionado por el alumno y dirigido por el profesor). Una definición inicial de objetivos básicos constituye una excelente guía para la elección de las técnicas

didácticas, las actividades de aprendizaje de los alumnos y los sistemas de evaluación. Es el denominado *contrato de aprendizaje* el que permite sistematizar este proceso de aprendizaje (objetivos-métodos-evaluación) (ref. M.S. Knowles 86).

3.2.1 Objetivos básicos

Los objetivos didácticos son los resultados que se espera obtener como consecuencia de la actividad docente. En Landsheere 77 y Morales 95 se pueden consultar diversas clasificaciones de los objetivos en función de áreas de conocimiento y niveles de enseñanza. En esta memoria, se consideran los siguientes objetivos básicos de la metodología docente:

- Crear una red de conceptos relacionados con la asignatura, sobre la que el alumno pueda sostener otros conceptos más avanzados, crear nuevas redes conceptuales o tupir la actual con nuevos nodos (aprendizaje *significativo*).

- Desarrollar ciertas habilidades específicas que se le supondrán ya familiares en asignaturas más avanzadas, o en el ejercicio profesional o investigador en el caso de asignaturas finalistas.

- Potenciar habilidades generales como la capacidad de abstracción, la habilidad para buscar y consultar diferentes fuentes de información, la agilidad en la aplicación de métodos de cálculo o técnicas de resolución de problemas, el hábito de colaboración con otras personas, la actitud crítica para seleccionar conceptos, relacionarlos y extraer conclusiones propias.

Para conseguir estos objetivos en el entorno docente descrito en el capítulo anterior, se dispone de las técnicas didácticas que se discuten en la siguiente Sección. Sin olvidar lo importante que es la formación permanente del profesor no solo en los temas propios de la asignatura, sino también en los relativos al ejercicio docente en sí mismo, y un constante ejercicio de autoevaluación.

3.2.2 Técnicas didácticas

El proceso de aprendizaje difiere según la naturaleza del alumno y de la materia. Por ejemplo, hay quien aprende siguiendo un procedimiento inductivo, mientras que en el otro extremo hay quien asimila por deducción, existiendo entre ambos una dilatada variedad de modelos. Esto debe ser tenido en cuenta en la metodología docente, especialmente en entornos que no permiten una enseñanza individualizada en la práctica.

Cada vez hay más consenso en relación a algunos aspectos de los métodos docentes (ref. Ubieto 99):

- La relación profesor/alumno viene impuesta por el entorno en que se desenvuelve nuestra tarea, en el que se ha desestimado el método individual por antieconómico, y se ha optado por

el colectivo, en el que un profesor conduce a un grupo de alumnos, casi siempre demasiado numeroso.

- Se tiende a desestimar el uso de la palabra (método verbalista) frente a esquemas, gráficos o diagramas (métodos intuitivos). En las ingenierías ha sido siempre práctica común.
- Las asignaturas o materias no pueden ser tratadas de forma aislada, desgajadas del conjunto. Es importante que la materia de la asignatura se relacione con las del resto del plan de estudios.
- En cuanto al trabajo escolar del alumno, parece idóneo un método mixto, que compagine el trabajo individual, tratando de explotar al máximo las posibilidades personales, con el trabajo en grupo, que a veces ha sido más descuidado.
- Se debe tender a métodos que prevean la participación del alumno (métodos activos en lugar de pasivos), convirtiendo al profesor en orientador y no solo en transmisor de conocimiento.
- A veces la educación ha pecado de acostumbrar al alumno a tomar posturas poco críticas, a aceptar los conocimientos sin discusión. Es importante forzar a veces al alumno a adoptar actitudes dubitativas y críticas.

Otros aspectos sin embargo son más cuestionables, y dependen en gran medida de la materia, del alumno y de las condiciones de impartición:

- En cuanto a la forma de razonamiento, la materia de Sistemas Operativos I aunque se presta más fácilmente a la aplicación de técnicas de tipo deductivo (aplicación), también permite aplicar razonamientos de tipo inductivo y analógico apoyados en técnicas como la observación, experimentación y comparación o generalización. Este tipo de técnicas inductivas y analógicas se utilizan para conseguir la participación del alumno de forma natural, exigiéndole actitudes experimentales y de observación.
- En cuanto al orden de presentación usado en cada materia de esta asignatura, se realiza inicialmente utilizando la presentación “lógica”, de antecedente a consecuente, de causa a efecto, para pasar después al llamado orden “psicológico”. Este último es el que le permite al alumno experimentar con lo presentado y avanzar hacia lo desconocido.
- En general se seguirá una estructura sistemática de la clase. Sin embargo es bueno realizar rupturas ocasionales de esa estructura, para dar agilidad y aumentar la motivación del alumno. En Sistemas Operativos I, resulta útil insertar pequeños comentarios de actualidad, e incluso dedicar alguna clase de teoría, o parte de ella, al estudio de alguna noticia de reciente publicación, o a la realización de algún ejercicio sobre dicha noticia. Como ejemplo, se puede realizar algún ejercicio que consista en extraer información relativa al mecanismo de señales utilizado en un determinado sistema a partir de un pequeño artículo.
- Por último en esta asignatura, como en la mayoría de las de ingeniería, se combinan el enfoque analítico y el sintético.

A pesar de la variedad de sus formas, el aprendizaje puede dividirse en tres fases. En primer lugar y mediante el modelo de razonamiento que sea, hay que adquirir unos conceptos básicos. En segundo lugar hay que aplicar esos conceptos a la resolución de problemas sencillos. Finalmente

hay que enfrentarse a problemas complejos en los que pueden aparecer elementos de otras materias afines e incluso elementos conceptuales nuevos que el alumno debe aprender o investigar por sí mismo. Esta tercera fase no sólo es *imprescindible* en el aprendizaje, sino que es *inevitable* cuando se llega al ejercicio profesional, particularmente en el ámbito de las ingenierías.

Cada una de estas fases requiere una técnica docente distinta. Por ello, en general, las asignaturas tienen asignados créditos y clases de distintos tipos.

Tipos de clases

Las clases pueden ser de teoría, de problemas y de laboratorio. En la clase de teoría el profesor transmite unos conocimientos determinados, suscita el interés por los mismos y activa los conocimientos previos para poder captar los conocimientos nuevos o posteriores. Es esencialmente una comunicación unidireccional, del profesor hacia el alumno, aunque no excluye cierto grado de participación, y hace posible la impartición a grupos de cierta entidad numérica. Sin embargo, también es necesario que el alumno vaya adquiriendo la capacidad de transformar sobre la marcha su propio aprendizaje. Para activar esta actitud, el profesor debe propiciar las experiencias oportunas que lleven hacia el aprendizaje *activo* y *cooperativo*. Las clases de problemas y laboratorio son básicas para esta tarea.

Las clases de problemas tienen como finalidad la discusión colectiva, guiada por el profesor, sobre temas concretos, permitiendo la asimilación de conceptos teóricos o abstractos e incluso el *descubrimiento* de nuevas soluciones no presentadas explícitamente en la teoría. El tema puede ser un ejercicio teórico o de aplicación, el contenido de un artículo, un caso práctico, etc. Estas clases requieren la preparación previa del tema por parte del alumno. Los grupos han de ser necesariamente poco numerosos para que la participación sea factible y diferenciada respecto a la clase de teoría convencional.

En las clases de laboratorio, los alumnos llevan a cabo trabajos con presencia y guía de los profesores. Todos los estudiantes realizan los mismos trabajos (o ligeras variantes del mismo trabajo), que en general acaban dentro del horario de la clase. El grupo ha de ser reducido para que el profesor pueda atender correctamente a todos los estudiantes. Los proyectos pueden considerarse un tipo particular de clase de laboratorio en el que los estudiantes cuentan con la guía y ayuda del profesor pero no con su presencia constante. El profesor plantea el trabajo, resuelve dudas y lleva a cabo un seguimiento para asegurar que se utiliza la metodología adecuada. El proyecto acostumbra a ser un trabajo de larga duración, no tiene que ser el mismo para todos los alumnos, y puede realizarse en equipo.

A continuación se detalla la metodología docente en cada tipo de clase, y el trabajo que el alumno debe desarrollar en cada caso.

Clases de Teoría

La clase de teoría goza de poca popularidad y se ha convertido en sinónimo de pasividad (ref. Michavilla y Calvo 98), en paradigma de la caducidad de un sistema educativo que para muchos debiera de ser sustituido por una nunca bien definida actividad multimedia. La mayoría de las veces se está confundiendo la clase de teoría con un modo particular de presentar la clase, el expo-

sitivo. Es cierto que el método docente no puede descansar en exclusiva sobre la *clase magistral* y en este proyecto docente está siendo considerado como un elemento entre otros. Sigue siendo un elemento imprescindible, por muchos motivos.

Resulta iluminador que el expositivo sea el principal método que los profesionales de la docencia y la investigación utilizamos para transmitirnos nuestros conocimientos (por ejemplo, en las presentaciones de los congresos de investigación). La clase magistral sigue siendo el método más eficaz para la transmisión de la información y la facilitación de la comprensión de temas complejos, para elevar el nivel motivacional de los alumnos y para sintetizar fuentes informativas diversas y de dificil acceso. Es además útil para presentar nuevos temas y explicar los conceptos principales siguiendo un orden bien establecido, y dando una visión de conjunto de la materia a lo largo del curso.

Sin embargo, también ocurre que el método expositivo corre el riesgo de obtener pasividad en el alumno si no se estimula el uso de otras fuentes de información. Puede incluso llegar a reducir el papel del profesor a un mero informador que no obtiene realimentación hasta el día del examen. Es por esto necesario introducir en la clase de teoría momentos centrados no solo en el profesor sino también momentos centrados en actividades que llevan a cabo los estudiantes ya individualmente o en grupos pequeños (*clase magistral participativa*). Pueden ser momentos de los usados típicamente en la clase de problemas o laboratorio, con participación directa del alumno. Los métodos exegético, interrogatorio, debate o el estudio dirigido suelen ser los más útiles para fomentar estos momentos. Para el temario del proyecto docente presentado en esta memoria se propone utilizar estos modos de presentación siempre que sea posible.

Un método que puede ser especialmente eficaz es el de plantear un ejercicio corto al final de la clase, pedir respuesta por escrito de él e incluso someterlo a evaluación. Serán cuestiones muy concretas, relativas a conceptos relevantes que se acaban de exponer, de respuesta inmediata, que requieran poca o nula elaboración. Respecto del profesor, los objetivos de este ejercicio son motivar la atención previa de los alumnos, obligarles a reflexionar sobre un concepto básico que acaban de recibir y puede contar como un elemento más en la evaluación; de hecho constituirá un método de evaluación continua útil para la mejora del proceso de aprendizaje. Respecto del alumno, este ejercicio le permite evaluar su entendimiento del material, le ayuda a retenerlo y le alienta a codificar el material en sus propias palabras. Además proporciona realimentación tanto al alumno como al profesor. Ocurre también que es impensable la corrección diaria de todas las respuestas aún cuando sean muy simples y breves. Una solución puede ser disminuir el número de problemas planteados (5-10 ejercicios durante todo el curso), a corregir aleatoriamente o a corregir condicionalmente según los resultados de examen. La realización del problema por grupos de entre 2 y 5 alumnos es otro método de reducción del trabajo de corrección, además de que fomenta el trabajo colectivo con todos sus beneficios.

En cualquier caso, la clase teórica resulta ineficaz si el número de alumnos es elevado o moderadamente elevado (alrededor de 100) y no se cuidan toda una serie de aspectos. Los que describo a continuación son principalmente resultado de la propia experiencia, fundamentados en algunos casos en (refs. Ubieto 99 y De la Cruz 04).

Condiciones de impartición

Aún el mejor método de presentación es un fracaso si un alto porcentaje de asistentes no percibe con claridad lo que se habla ni lo que se escribe o proyecta. Es fundamental la revisión de las condiciones físicas de impartición de la clase teórica (condiciones de audición, y visibilidad de la pizarra o pantalla de proyección). Estas condiciones no se cumplen cuando el aula posee un tamaño excesivo o cuando el número de alumnos es elevado. Resulta imposible evaluar bien el efecto de estas condiciones en la clase que uno mismo imparte, de modo que algo que he hecho en varias ocasiones ha sido asistir a clases impartidas por otros compañeros y me he colocado en las últimas filas. Esta es la forma en la que he podido comprobar cómo las dificultades de audición y visibilidad hacen que la clase sea difícil de seguir, incluso cuando se conoce el tema.

Esto provoca el desinterés y la desconexión de gran parte del alumnado, no sólo durante una clase aislada, sino incluso durante toda la asignatura. El docente debe ser consciente de esto, reparando en lo posible estas deficiencias con los medios a su alcance. Por ejemplo, da muy buenos resultados situarse en la parte posterior del aula para preguntar o razonar sobre un esquema que está siendo proyectado y cuyos elementos ya han sido introducidos, comprobando de paso la visibilidad de los mismos.

Estructuración de la clase

La clase teórica permite articular el curso en sesiones breves que abordan un contenido limitado (*lección*). No siempre es posible determinar exactamente el punto de comienzo y el de finalización de cada sesión, pero en cualquier caso el número de conceptos introducidos no debe ser muy elevado. Hay que evitar concentrar la presentación de nuevos conceptos en ciertas sesiones, haciendo que otras sean exclusiva o excesivamente descriptivas: en estas segundas el alumno bajará apreciablemente el nivel de atención. Pero además para garantizar el aprendizaje es fundamental que cada sesión siga una estructura bien definida con una *introducción*, un *cuerpo* y una *conclusión*. En esta estructura se deben incluir además las siguientes partes:

1. **Introducción** (tareas)

   motivar hacia la tarea (puede ser el resumen de la clase anterior)

   presentar el esquema de la clase (objetivos y guión de lo que se va a presentar)

2. **Cuerpo** (exposición)

   estructurar el contenido

   repetir los puntos principales

   usar analogías y ejemplos

   fomentar la participación (uso de la pregunta, de las técnicas de grupo)

3. **Conclusión** (tareas)

   breve resumen de lo expuesto

   enfatizar las ideas principales

   plantear alguna cuestión, problema concreto (uso del ejercicio). Resolver

   introducir la siguiente sesión

Esta estructura demuestra ser especialmente efectiva en cursos superiores, a causa de la madurez del estudiante. La experiencia señala que el alumno acaba por familiarizarse con este procedimiento, y su carácter rutinario —repetitivo— le sirve para centrarse rápidamente en la clase. El esquema en pizarra facilita la contextualización de lo que se va exponiendo. Este esquema puede sustituirse por un esquema en transparencia, del cual el alumno tiene copia, y que se va señalando al ir progresando de una a otra parte de la exposición.

RECURSOS DIDÁCTICOS

Las clases teóricas se desarrollarán en general sobre pizarra, lo que implica el beneficio adicional para el alumno de aprender a tomar apuntes. No obstante, en materias como la que se detalla en esta memoria, el uso de transparencias convencionales o presentaciones mediante cañón de video son de gran ayuda cuando se presentan tan a menudo fragmentos de código, representaciones gráficas o esquemas. Es imprescindible que el alumno disponga en este caso de la copia pertinente. Con este método es posible volver a visualizar con agilidad elementos de clases pasadas y códigos o esquemas clave en el curso, a fin de recordar o precisar conceptos, evitando la pérdida de tiempo que supone reescribirlos o redibujarlos en pizarra. En cualquier caso, tanto pizarra como transparencia han de utilizarse teniendo en cuenta unos puntos mínimos:

- Hay que preparar los esquemas de pizarra. En la práctica se recurre a la pizarra como soporte general del discurso hablado y se improvisa en función de la marcha de la clase. Pero los esquemas y presentaciones clave han de estar bien estructurados y por ello es conveniente prepararlos con antelación, para que se ejecuten en la clase de forma ordenada y clara, facilitando la toma de apuntes del alumno.
- Las transparencias han de sujetarse a las normas de sentido común en relación a los tamaños de figuras y símbolos. Hay que evitar que el alumno mire exclusivamente a su copia en papel, perdiendo el contacto con el ritmo de la explicación generalmente.

Clases de Problemas

El objetivo principal de la clase de problemas es fortalecer los conceptos y procedimientos que el alumno ha aprendido en la clase de teoría, obligándole a aplicarlos sobre nuevos supuestos. Este ejercicio de aplicación hace que el estudiante aprenda más al tener que trabajar sobre un problema el lugar de ver cómo se realiza. También le servirá como herramienta de autoevaluación. En esta asignatura, los ejercicios resueltos en la clase de problemas tienen además un objetivo integrador, ya que requieren el uso de conocimientos recibidos a lo largo de varios Temas, a veces de todo un Módulo.

La forma ideal de orientar la clase de problemas es que el profesor hace los planteamientos y guía la resolución, valorando las propuestas y corrigiendo errores, presentando la solución correcta sólo como último paso. Son los alumnos quienes han de llevar el protagonismo.

En la práctica este tipo de clase está sujeta a dos dificultades. En primer lugar la resolución en grupo de problemas requiere tiempo: hay que plantear el problema, dar tiempo para que el alumno trabaje, comentar las soluciones... Por ello es preciso preparar para cada tema un conjunto de problemas de dificultad creciente. Los más sencillos pueden resolverse en clase, los más complejos quedan planteados de un día para otro.

La disponibilidad de una colección de problemas es útil como soporte a la clase. Para la asignatura Sistemas Operativos I el alumno dispone de un completo conjunto de problemas con los que trabajar. Son todos los ejercicios que dan apoyo al curso de lenguaje C y enunciados de exámenes tipo. Algunos días antes de cada sesión de problemas se indican los ejercicios que se resolverán en dicha sesión, y se señalan otros problemas interesantes relacionados para facilitar el estudio personal por parte del alumno. El método de propuesta de problemas de una sesión para otra permite al alumno autoinformarse sobre su comprensión de la asignatura y saber si necesita un esfuerzo adicional.

La segunda dificultad es el número de alumnos, especialmente en asignaturas obligatorias. En el entorno docente en el que se basa esta asignatura (Centro Politécnico Superior) se establecen para las clases de problemas los mismos grupos que para las clases de teoría. Por ello en las clases de problemas hay que recurrir al trabajo en grupo, lo que a veces puede significar trabajo de personas físicamente próximas, cuando la disposición del aula dificulta la movilidad.

Hay que conseguir que el alumno vea este tipo de *aprendizaje cooperativo* en servicio a su entrenamiento en habilidades para la comunicación y relaciones interpersonales necesarias para la labor en equipo. Sin embargo, resulta dificil fomentar la participación de los estudiantes en este tipo de actividades. Una forma efectiva de motivar el trabajo en grupos, consiste en utilizar el método de participación por representantes de grupo. Es necesario garantizar que un trabajo en grupo o cooperativo va a darle a cada miembro unos resultados que serán mayores que si el trabajo lo hiciese él solo (ref. Ovejero 90). Teniendo en cuenta que todos los grupos no serán iguales ni responderán de igual manera, hay que aprovechar estas diferencias utilizándolas como medio de enriquecimiento y mejora del aprendizaje de cada uno aisladamente. Alumnos rezagados encontrarán ayuda efectiva para remontar el curso y alumnos brillantes verán como enseñar a otro es una forma de aprender en profundidad. En todo caso un objetivo básico debe ser el evitar que la clase de problemas derive en una sesión de explicación de las soluciones por parte del profesor.

Clases de Laboratorio

El aprendizaje en este tipo de clases tiene tres características muy importantes. En primer lugar el alumno se enfrenta a un entorno y problema real, en el que aciertos o errores pueden comprobarse. En segundo lugar la motivación del alumno es muy grande, porque queda directamente implicado en el desarrollo de la tarea. Finalmente, es un entorno especialmente adecuado para el trabajo en equipo.

Como se ha visto en el anterior capítulo el objetivo básico de la asignatura es integrar en el alumno los principios fundamentales que guían el diseño de un sistema operativo con el aprendizaje de una programación avanzada utilizando llamadas al sistema. El estilo de estudio más usado en Sistemas Operativos I es el basado en una primera descripción detallada tanto de los conceptos como de las herramientas de diseño que se van a utilizar y en un apoyo mediante ejemplos de implementación básicos. Estos conceptos y herramientas de diseño se han de fundir finalmente con habilidades de diseño e implementación, las cuales se adquieren de una forma más natural y efectiva con la práctica. De ahí la importancia de las clases de laboratorio en esta asignatura. Algunos de estos conceptos y habilidades solo se mencionan en las clases de teoría y problemas. Es en las clases de laboratorio donde se presentan formalmente y donde se desarrollan.

Las características de los planes de estudios cuatrimestrales hacen que las prácticas de laboratorio tengan que estar perfectamente planificadas y coordinadas con la teoría, basándose en ejercicios que se puedan realizar completamente en el tiempo previsto para cada sesión, y con el objetivo de estructurar gradualmente los conocimientos que se pretende que el alumno asimile. En Sistemas Operativos I cada práctica se desarrolla en dos horas de laboratorio.

Para cada práctica, el enunciado consta habitualmente de diversos apartados de complejidad creciente en los que el alumno resuelve a menudo problemas similares a los realizados en la clase de problemas, con la ventaja de poder comprobar la validez de la solución en la práctica. Pueden proponerse apartados o enunciados adicionales optativos para incrementar el grado de asimilación de los alumnos que lo deseen.

El profesor debe exponer un breve repaso de los temas que abarca la práctica para centrar rápidamente el contexto de la clase, y posteriormente realizar un seguimiento de todos los alumnos con el fin de garantizar que todos ellos consiguen alcanzar los objetivos propuestos en la práctica. Los alumnos con mayor capacidad de resolución de los apartados de la práctica pueden consultar con el profesor la calidad de su solución e intentar resolver los apartados o enunciados opcionales. Aquellos que experimentan mayores dificultades, tienen mayor soporte del profesor para que éste les encauce hacia la solución. Es importante que el número de alumnos sea reducido, para poder cumplir estos objetivos (alrededor de 20).

En el entorno en el que se desarrolla la asignatura de esta memoria, la situación habitual es que cada asignatura sea impartida por un único profesor y a uno o varios grupos. No existe así la oportunidad de colaborar con otros profesores en la preparación de las prácticas, como en los entornos en los que una asignatura es impartida por varios profesores. Con esto se simplifica el problema de coordinación entre los distintos tipos de clases, al recaer todas en una misma persona, pero la sobrecarga hace especialmente importante la experiencia del profesor en las plataformas en las que se desarrollan las prácticas.

3.3 Evaluación

En el proceso aprendizaje-enseñanza es básica la definición de unos objetivos didácticos, pero ésta quedaría incompleta si no se señalaran también los criterios a seguir para valorar su consecución. Si la instrucción es la actividad que busca el aprendizaje, la evaluación nos dirá si éste ha tenido lugar o no y por qué, si se necesita modificar el procedimiento de instrucción y cómo

hacerlo. Una evaluación integrada en el proceso de enseñanza-aprendizaje se convierte en el medio para facilitar la labor docente, haciéndola más eficaz y satisfactoria y la del aprendiz mejor guiada. Deberá ser *formativa*, para ofrecer retroalimentación tanto al docente como al aprendiz y certificativa o *sumativa*, para *certificar* la calidad del producto, dejando de lado un tipo de evaluación más tradicional (normativa, comparativa) donde no se tengan en cuenta los progresos que el sujeto realiza desde su línea base y en función de sus adquisiciones sucesivas.

Lo deseable en la evaluación formativa es realizar una evaluación inicial (*diagnóstica*, para determinar los conocimientos previos y características de los aprendices), continua, y final, que abarque todos los objetivos propuestos, que sea diversificada en métodos y corporativa. La evaluación sumativa es la que se realiza generalmente al final (calificación) y tiene como objeto el describir y valorar los resultados obtenidos.

Aunque es un objetivo deseable (art. 164.1 de los Estatutos de la Universidad de Zaragoza), el sistema de evaluación continua no está implantado como tal en el Plan de Estudios utilizado como ejemplo en este programa. Sin embargo, se establece el derecho del estudiante a ser evaluado con, al menos, dos pruebas. El diferente carácter que cada docente imprime al examen de teoría, la tensión de la prueba o la situación personal del alumno, ocasionan que estudiantes que han seguido el curso con regularidad experimenten a veces dificultades en una sola prueba. Es objetivo de este proyecto docente el conseguir información adicional sobre el rendimiento de los alumnos durante el curso de diferentes formas, haciendo que la nota no dependa excesivamente de un examen final de teoría. Aún sin ser un estricto sistema de evaluación continua (poco viable en entornos con falta de recursos y tiempo), sirve para incentivar al alumno a seguir la asignatura desde el primer día de clase.

Por otra parte, y en la línea de lo expresado en la sección relativa a las clases de laboratorio, la práctica tiene un papel fundamental en el proceso de aprendizaje. El peso de las clases de laboratorio en la calificación final de Sistemas Operativos I es de un 20% de forma directa, a través de un examen de prácticas. La realización de las prácticas afianza los conocimientos adquiridos en las clases de teoría y problemas, siendo de esta forma el mejor método de estudio que el alumno puede usar. De forma indirecta, se incrementa además el peso de las clases de laboratorio en la nota final, en una proporción dependiente de cada alumno.

3.3.1 Recopilación de información sobre el alumno

Entre las formas de recopilación de información sobre el rendimiento del alumno pueden enumerarse las siguientes:

- Realización de los pequeños ejercicios de control a los que se hizo alusión en la subsección relativa a la clase de teoría.
- Muestreo de soluciones propuestas en las clases de problemas, especialmente en su acepción positiva, es decir, calificando las primeras y mejores respuestas.
- Entrega de ejercicios voluntarios de prácticas.
- Realización de exámenes parciales.

En este proyecto docente se proponen las dos primeras para complementar el discurso teórico convencional. En un contexto con asignaturas cuatrimestrales, una entrega de ejercicios voluntarios de prácticas puede no resultar muy positiva, debido a la alta carga de laboratorio que deben soportar los alumnos. En cualquier caso no la considero fundamental por entender ya suficientes los dos puntos anteriores y los exámenes de teoría y prácticas.

En la evaluación propuesta para Sistemas Operativos I, se ha planificado la realización de un examen parcial hacia mitad del curso. De nuevo y en un entorno con asignaturas cuatrimestrales, podría resultar poco conveniente la realización de esta prueba. Sin embargo, mi experiencia docente me ha demostrado que este tipo de estrategia resulta especialmente útil. De la misma forma que toda asignatura se divide en lecciones, una evaluación dividida en metas progresivas y espaciadas que hay que superar para alcanzar la meta final, permite reajustar el proceso sin poner en peligro el llegar al nivel final (efecto *espaciamiento*). Además de permitirle al alumno ver con anticipación qué es lo que se exige de él, pudiendo moldearse a medida que se va viendo capaz de ir alcanzando las distintas metas que se le presentan.

3.4 Aspectos complementarios

En esta sección se describen dos actividades que complementan la metodología docente mostrada en las secciones previas: las horas de consulta y el uso de listas de correo electrónico y de *internet* como medios de comunicación alumno-profesor.

Consultas del alumno

Las horas de consulta o *tutorías* son un complemento de las actividades docentes descritas hasta aquí (clases de teoría, problemas y laboratorio). Permiten un trato personal con el alumno, que puede preguntar dudas concretas generalmente con más calma que en las clases. El profesor puede verlo como un recurso para poder percibir las principales dificultades de los estudiantes, permitiéndole revisar en el futuro la forma de explicar los puntos más difíciles.

Algunos métodos docentes entre los propuestos hasta el momento ayudan a mejorar el uso que el alumno hace de la tutoría. La realización de prácticas, y la respuesta a ejercicios al final de las clases de teoría son dos buenos ejemplos. Ambos métodos obligan al alumno a utilizar los conocimientos recibidos hasta el momento, y le ofrecen realimentación. Por otro lado, el trabajo en grupo durante las clases ayuda a elevar la calidad de las consultas realizadas en las horas de tutoría, debido a que incentiva la formación de pequeños grupos de trabajo y la enseñanza entre iguales.

Lista de correo electrónico e Información en *Internet*

Una experiencia interesante y positiva llevada a cabo por compañeros del área en los últimos cursos ha sido la puesta a disposición del alumno de información y materiales de las asignaturas a través de *Internet*, y la puesta en marcha de una lista de discusión por correo electrónico. Todos los matriculados en la Universidad de Zaragoza tienen una dirección de correo electrónico y acceso a *Internet* asegurado en salas específicas, con independencia de que lo tengan a modo particular (la UZ también proporciona este servicio a bajo coste). Desde el curso 1994/95 en que se

comenzó esta experiencia, *Web*, *Ftp* y lista de *e-mail* han demostrado ser un vínculo efectivo de relación con el alumno.

La información facilitada a través del *Web* consiste en información convencional (temario, calendario de teoría y prácticas, criterios de evaluación, etc.) y materiales relacionados con las prácticas.

El uso de una lista de discusión por *e-mail* aporta nuevas posibilidades de relación profesor/alumno y, todavía más importante, alumno/alumno. Las respuestas del profesor son recibidas por todos ellos simultáneamente, a diferencia de las respuestas dadas en las tutorías personalizadas, y quedan registradas en el servidor de la lista, de modo que incluso aquéllos que no acceden habitualmente al *e-mail* o no tienen conexión en casa pueden consultarlas desde cualquier sala de prácticas en cualquier momento. Los alumnos además pueden discutir entre ellos las respuestas a través de la lista, facilitando la enseñanza entre iguales.

El éxito de este tipo de herramientas se ve favorecido en nuestro entorno debido a que se trata de alumnos de Informática, habituados ya al uso de *Internet* y muy motivados. Además, todos ellos pueden consultar su correo, con más o menos dificultad, en el propio CPS, y un alto porcentaje lo hace habitualmente desde su residencia.

Por todo ello, tanto el uso de *Web* para facilitar información como la lista de *e-mail* son recursos que desearía implantar en los próximos cursos como apoyo de esta asignatura.

3.5 Autoevaluación del Profesor

La evaluación y control de la docencia y la mejora de la enseñanza son temas de evidente necesidad, controvertidos y en continua revisión, en los que influye un entorno legal y social (ref. Escudero 96); no en vano la Universidad de Zaragoza ha dispuesto un Vicerrectorado específico para ello.

En la Universidad de Zaragoza se lleva a cabo un procedimiento de Evaluación y Control de la Docencia de acuerdo a la normativa establecida en el Capítulo VI de sus Estatutos. Los alumnos realizan en cada curso una encuesta con varios bloques de preguntas, algunos específicos del Centro o Facultad en el que desarrollan sus estudios. De los resultados de estas encuestas se deriva un informe por parte de la Comisión de Docencia del Centro o Facultad correspondiente en el que se evalúa como POSITIVA o NEGATIVA la actividad de cada docente, con las repercusiones correspondientes según la normativa en vigor. Más allá de las implicaciones administrativas y salariales de estos resultados, y a pesar de la controversia que existe sobre la conveniencia o validez de este tipo de evaluaciones, especialmente por la baja participación del alumnado, considero imprescindible que el docente analice los resultados. En general la mejora de la metodología docente se refleja en una mejora en la apreciación de los alumnos.

Creo también que la evaluación del profesor por parte de los alumnos no es suficiente. La información proporcionada por este tipo de evaluación no basta para detectar de forma concreta las posibles insuficiencias y sus causas. Debería existir una práctica complementaria propia de *cada* enseñante, indispensable en todo marco metodológico docente, que es la autoevaluación del pro-

fesor. Una cierta metodología de autoevaluación puede ser realmente importante incluirla en todo proyecto docente. Metodología basada en aspectos como los siguientes:

1. Establecimiento o elección personal de un *modelo* de profesor universitario y fundamentación en un modelo pedagógico adecuado para la enseñanza superior (en este caso el modelo enseñanza-aprendizaje constructivista, al que se ha hecho referencia en este Capítulo).
2. Revisión personal mediante la utilización de un cuestionario apropiado, que ayude a revisar aspectos concretos e importantes relativos a la preparación, impartición y evaluación de un curso y sus clases.
3. Contraste de las percepciones personales con los resultados de las Encuestas de Evaluación de la Docencia y con los datos obtenidos a través del contacto con los alumnos.
4. Observación de otros compañeros en el desempeño docente de materias afines, e intercambio de experiencias y métodos.

Considero que plantear la integración en el trabajo docente de una autoevaluación bien definida no exigirá demasiado tiempo en relación con otras actividades. Además, al no estar sometida a presiones externas, servirá para aumentar el grado de implicación personal en el ejercicio docente, resultando —subjetiva y objetivamente— realmente eficaz.

Bibliografía por Capítulos

CAPÍTULO 1. ENTORNO DOCENTE

1. Centro Politécnico Superior. *Guía Informativa*. Curso 2004-2005. Univ. Zaragoza, 2004
2. COMMUNICATION FROM THE COMMISSION. *The role of the universities in the Europe of knowledge*. Commission of the european communities, Brussels, 05.02.2003. COM(2003) 58 final
3. Computing Curricula Computer Engineering. *The Joint Task Force on Computing Curricula* 2005. IEEE Computer Society Association for Computing Machinery. Report. Ironman Draft, 2004 June 8
4. LEY ORGÁNICA 6/2001, DE 21 DE DICIEMBRE, de Universidades BOE nº 307 24 DE diciembre
5. MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE: *La integración del sistema universitario español en el Espacio Europeo de Enseñanza Superior.* Documento-Marco. Madrid. MECD, 2003
6. PROYECTO EICE. *Libro Blanco del Título de Grado en Ingeniería Informática*. Agencia Nacional de Evaluación de la Calidad y Acreditación (ANECA), *Los estudios de Informática y la Convergencia Europea*. Marzo del 2004
7. REAL DECRETO 1497/1987, de 27 de Noviembre, por el que se establecen directrices generales comunes de los planes de estudio de los títulos universitarios de carácter oficial y validez en todo el territorio nacional. BOE del 14 de Diciembre de 1987
8. REAL DECRETO 1459/1990, de 26 de Octubre, por el que se establece el título universitario oficial de Ingeniero en Informática y las directrices generales propias de los planes de estudio conducentes a la obtención de aquél. BOE del 20 de Noviembre de 1990
9. REAL DECRETO 1125/2003, de 5 de Septiembre, por el que se establece el sistema europeo de créditos y el sistema de calificaciones en las titulaciones universitarias de carácter oficial y validez en todo el territorio nacional. BOE nº 224 del 18 de Septiembre de 2003

10.REAL DECRETO 55/2005, de 21 de Enero, por el que se establece la estructura de las enseñanzas universitarias y se regulan los estudios universitarios oficiales de *Grado*. BOE nº 21 del 25 de Enero de 2005
11.REAL DECRETO 56/2005, de 21 de Enero, por el que se regulan los estudios universitarios oficiales de *Posgrado*. BOE nº 21 del 25 de Enero de 2005
12.RESOLUCIÓN de 12 de Septiembre de 1994, de la Universidad de Zaragoza, por la que se hacen públicos los planes de estudios conducentes a la obtención de los títulos de Ingeniero en Informática, Ingeniero de Telecomunicación y primer ciclo de Ingeniero Industrial. BOE nº 27 1 de Febrero 1995

CAPÍTULO 2. SISTEMAS OPERATIVOS I. GUÍA DOCENTE

13.BOVET D.P., CESATI M. *Understanding the LINUX kernel.* O'Reilly, 2001
14.DARNELL P.A., MARGOLIS P.E. *Software Engineering in C*. Springer-Verlag, 1988
15.HP-UX. *HP-UX Systems Calls.* Hewlett Packard, HP 9000 Computers. HP-UX Release 11.0
16.KERNIGHAN B.W., PIKE R., "The UNIX Programming Environment", Prentice Hall, 1984
17.PETERS J.S. *UNIX programming.* Harcourt Brace Jovanovich, 1989
18.ROCHKIND M.J. *Advanced UNIX programming*. Prentice Hall, 1985
19.SCHILDT H. *C, Manual de referencia*. Cuarta Edición. McGraw-Hill, 2001
20.SILBERSCHATZ A., GALVIN P.B., and GAGNE G., *Operating System Concepts*. Seventh Edition. John Wiley & Sons, Inc., 2004
21.STALLINGS W. *Sistemas Operativos. Principios de diseño e interioridades*. Cuarta Edición. Prentice Hall, 2001
22.STEVENS W.R. *Advanced programming in the UNIX environment*. Addison-Wesley Publishing Company, 1994
23.TANENBAUM A.S. *Sistemas Operativos. Diseño e implementación*. Prentice Hall Hispanoamericana, 1988

CAPÍTULO 3. METODOLOGÍA DOCENTE

24.AUSUBEL D.P. *Psicología educativa: un punto de vista cognoscitivo.* Trillas, Méjico 1989
25.BERNAD J.A. *Estrategias de enseñanza-aprendizaje en la universidad.* ICE Zaragoza 1990
26.DE LA CRUZ TOMÉ, M. África. *Guía de autoevaluación para la mejora de la docencia universitaria*. Curso de Elaboración del Proyecto Docente. ICE de la Univ. de Zaragoza, Febrero-Junio 2004
27.DE LA CRUZ TOMÉ, M. África. *Necesidad y objetivos de la formación pedagógica del profesor universitario*. Curso de Elaboración del Proyecto Docente. ICE de la Univ. de Zaragoza, Febrero-Junio 2004
28.DE LA CRUZ TOMÉ, M. África. *Guía de autoevaluación para la mejora de la docencia universitaria*. Apuntes del Curso de Autoevaluación del Profesor y Mejora de la Calidad de la Docencia. ICE de la Univ. de Zaragoza, Febrero-Junio 1999
29.DELPRATO D., MIDGLEY B. *Some fundamentals of B. F. Skinner's behaviorism*. de *American Psychologist*. nº 47. 1992

30.DE MIGUEL M. *Calidad de la enseñanza universitaria y excelencia académica*. Lección inaugural del curso académico 1999-2000. Oviedo, Servicio de publicaciones de la Universidad de Oviedo.
31.ELTON L. *University teaching: a professional model for quality.* Ellis, R. *Quality assurance for university teaching*. Buckingham. Open University Press, 1993
32.ENTWISTLE N.J. *Understanding classroom learning.* Hodder and Stoughton, London 1987
33.ESCUDERO T. *Bases para un plan estratégico de la universidad de Zaragoza: encuestas sobre acciones estratégicas.* Informes del ICE de la Universidad de Zaragoza, nº 15. Univ. de Zaragoza, 1996
34.KNOWLES M. S. *Using learning contracts.* San Francisco, CA: Jossey-Bass Inc., Publishers, 1986
35.LANDSHEERE G. *Objetivos de la educación.* Barcelona. Oikos-Tau, 1977
36.MORRIS L. BIGGE. *Basic forms of stimulus-response conditioning*. de *Learning theories for teachers*. 4ª edición. Harper & Row publishers, 1982
37.MORALES P. *Los objetivos didácticos.* Bilbao. ICE. Universidad de Deusto, 1995
38.MICHAVILLA F., CALVO B. *La universidad española hoy.* Propuesta para una política universitaria. Madrid. Síntesis, 1998
39.OVEJERO A. *El aprendizaje cooperativo: una alternativa eficaz a la enseñanza tradicional.* Barcelona. PPU, 1990
40.UBIETO A. *Actuación docente, métodos didácticos*. Apuntes del Curso del Diploma de formación pedagógica para el profesor universitario. ICE de la Univ. de Zaragoza, 1999