Proyecto/Guía docente de la asignatura Estructura de Sistemas Operativos

Asignatura	ESTRUCTURA DE SISTEMAS OPERATIVOS		
Materia	ARQUITECTURA DE COMPUTADORES, SISTEMAS OPERATIVOS Y SISTEMAS DISTRIBUIDOS		
Módulo	COMUNES A LA INFORMÁTICA		
Titulación	PROGRAMA CONJUNTO DE ESTUDIOS GRADUADO EN INGENIERÍA INFORMÁTICA Y GRADUADO EN ESTADÍSTICA (INDAT)		
Plan	545	**Código**	46915
Periodo de impartición	Segundo Cuatrimestre	**Tipo/Carácter**	Complementos de Informática
Nivel/Ciclo	Grado	**Curso**	3
Créditos ECTS	6		
Lengua en que se imparte	Español		
Profesor/es responsable/s	Benjamín Sahelices Fernández (responsable teoría y laboratorio) Julián Arroyo Álvarez		
Datos de contacto (E-mail, teléfono…)	benja@infor.uva.es Tfno. 983185643 julian@infor.uva.es		
Horario de tutorías	Lunes (11:00-13:00), jueves (11:00-13:00) y viernes (11:00-13:00)		
Departamento	Informática		

1. Situación / Sentido de la Asignatura

1.1 Contextualización

La asignatura *Estructura de Sistemas Operativos* se encuentra dentro del conjunto de materias comunes a la informática. En concreto está incluida en la materia *Arquitectura de Computadores, Sistemas Operativos y Sistemas Distribuidos* siendo su carácter obligatorio. Se imparte en el segundo curso, segundo cuatrimestre, de la titulación de *Graduado en Ingeniería Informática*. Se trata de una continuación de la asignatura *Fundamentos de Sistemas Operativos*, que se imparte en el primer cuatrimestre del segundo curso.

1.2 Relación con otras materias

La asignatura *Estructura de Sistemas Operativos* está muy relacionada con la asignatura *Fundamentos de Sistemas Operativos*. Se trata de una continuación en la que se explican desde un punto de vista de estructura y diseño los conceptos básicos de sistemas operativos.

Adicionalmente esta asignatura está fundamentada sobre un conjunto de conocimientos conseguidos en las asignaturas *Fundamentos de Programación, Fundamentos de Computadoras* y *Arquitectura y Organización de Computadoras.*

1.3 Prerrequisitos

Recomendación: haber cursado previamente las asignaturas *Fundamentos de Programación* (primer curso, primer cuatrimestre), *Fundamentos de Computadoras* (primer curso, segundo cuatrimestre) y *Fundamentos de Sistemas Operativos* (segundo curso, primer cuatrimestre).

2. Competencias

2.1 Generales

G2	Conocimientos básicos de la profesión
G3	Capacidad de análisis y síntesis
G4	Capacidad de organizar y planificar
G5	Comunicación oral y escrita en la lengua propia
G8	Habilidades de gestión de la información
G9	Resolución de problemas
G10	Toma de decisiones
G11	Capacidad crítica y autocrítica
G12	Trabajo en equipo
G14	Responsabilidad y compromiso ético
G15	Liderazgo
G16	Capacidad de aplicar los conocimientos en la práctica
G18	Capacidad de aprender
G19	Capacidad de adaptarse a nuevas situaciones
G20	Capacidad de generar nuevas ideas
G21	Habilidad para trabajar de forma autónoma
G22	Diseño y gestión de proyectos

2.2 Específicas

CI1	Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
CI4	Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes
CI5	Conocimiento, administración y mantenimiento de sistemas, servicios y aplicaciones informáticas
CI10	Conocimiento de las características, funcionalidades y estructura de los Sistemas Operativos y diseñar e implementar aplicaciones basadas en sus servicios
CI12	Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos.

3. Objetivos

- Conocer los servicios que ofrece un SO, específicos desde el punto de vista de un usuario, de un programador y de un administrador y utilizar diferentes sistemas de interacción con el SO.
- Diseñar e implementar aplicaciones basadas en los servicios del SO, seleccionando los más adecuados a cada caso.
- Comprender y saber gestionar los sistemas de control de acceso a usuarios y recursos para garantizar un nivel de seguridad adecuado a nivel sistema operativo.
- Conocer los fundamentos de las técnicas de virtualización y, en concreto, de la virtualización de sistemas.

4. Bloques temáticos[1]

Bloque 1: Gestión de Memoria

Carga de trabajo en créditos ECTS:	3,6

a. Contextualización y justificación

La gestión de memoria es un aspecto básico en las prestaciones de las computadoras. Se trata de un recurso crítico en el que las decisiones de administración tienen un elevado impacto en las prestaciones. En este bloque temático se estudian los mecanismos de gestión de memoria de los sistemas operativos. Estos conocimientos son imprescindibles para poder identificar y resolver problemas de prestaciones en la administración de sistemas operativos, para diseñar aplicaciones con buenas prestaciones y para utilizar correctamente los servicios de los sistemas operativos.

b. Objetivos de aprendizaje

- Conocer los servicios que ofrece un SO específicas desde el punto de vista de un usuario, de un programador y de un administrador y utilizar diferentes sistemas de interacción con el SO.
- Diseñar e implementar aplicaciones basadas en los servicios del SO, seleccionando los más adecuados a cada caso.
- Conocer la influencia de los sistemas de gestión de memoria en el diseño de aplicaciones y en la utilización de los servicios que proporciona un SO.

c. Contenidos

- Capítulo 1: Espacios de Direcciones.

  Multiprogramming and time sharing. Ready queue. Protection. Address Space. Multiple threads. Abstraction. Hardware+software implementation Transparency, efficiency, protection. Homework: free, pmap, memory-user.c

- Capítulo 2: API Memoria.

1

Types of memory in AS. Code, stack, heap. malloc(). sizeof. free() Common errors: good compilation vs correction (not the same!). Allocation errors, initialization errors, free-type errors. Tools: valgrind (sudo apt install valgrind) Underlying OS support: brk, sbrk, mmap. Homework: debugging with gdb and valgrind (null.c, vectorInt.c)

- Capítulo 3: Traducción de direcciones.

  Memory virtualization. Illusion. Hardware-based. Virtual Address (VA) to Physical address (PA). AS of a simple example (Suma3). Dynamic relocation. Hardware based. Hardware support. OS issues: mem. management, base/bound management, exception handling, OS boot. Homework: relocation.py

- Capítulo 4: Segmentación.

  Segmentation. Spare ASs. Base/bound generalization. Segment identification. Segment Table (ST) storage. Stack. Sharing. OS support: context switch, grow/shrink, free space. Homework: segmentation.py

- Capítulo 5: Gestión del Espacio Libre.

  External/internal fragmentation. malloc-free. Split and coalescing. Free list. Tracking size (free(pointer)). Embbeding free list in free space. Growing the heap. Assignment strategies: best, worst, first, next (fit) Slab allocator. Buddy allocator. Homework: malloc.py

- Capítulo 6: Introducción a la Paginación

  Fixed-size pieces: pages/frames. Page table (PT). Virtual Page Number (VPN) to Physical Frame Number (PFN) conversión PT in physical or virtual memory. Page Table Base Register (PTBR). Page Table Entry (PTE): PFN, v, prot, present, dirty, ref Speed problem. Memory trace: array.c Homework: paging-linear-translate.py

- Capítulo 7: TLBs

Hardware to speed-up translation. VA to PA. Hardware managed Translation Lookaside Buffer (TLB). Example. Hardware managed TLB (CISC processors). Software managed TLB (RISC processors). TLB_MISS. TLB contents. Fully associative. VPN/ASIDS, PFN, valid, protection, dirty bit. Context switches. Replacement policies. LRU. Random. Real TLB entry (R4000) Homework: tlb.c

- Capítulo 8: Tablas de Páginas Avanzadas

  PT requires a lot of storage. Bigger pages. Problem Paging+segments. VPN to PFN conversion. External fragmentation for PTs. Multi-level PT. Page Directory (PD) + PT (1 or more levels; each portion in one frame). Spare ASs. Detailed examples. Three levels. Inverted PT. Swapping. Virtual spaces. Homework: paging-multilevel-translate.py

- Capítulo 9: Intercambio – Mecanismos

  AS greater than physical memory. Backup devices. Abstraction. Very large ASs for each process. Swap space. Disk address on PTE. Present bit. Trap to OS: page fault. OS handler. Page fault. I/O. Block process. Memory full. Victim selection. Replacement. Run at disk-speed instead of memory-speed. Page fault control flow. Page fault software description. Replacements. High Watermark (HW) / Low Watermark (LW). Page daemon. Page clusters to optimize disk operations. Homework: vmstat, mem.

- Capítulo 10: Intercambio – Políticas

  Cache management. Minimize cache misses. Minimize page faults. Average Memory Access Time (AMAT). Miss probability (Pmiss). Between memory (Tm - nsegs) and disk (Td - msegs). Optimal replacement policy. Comparing. Difficult to know the future. FIFO. Belady's anomaly. Random. LRU. MRU. MFU. Workload: fully random, 80-20, looping sequential. Algorithm implementation. Approximating LRU. Dirty pages. Page selection: demand paging, prefetching. Thrasing. Homework: paging-policy.py

- Capítulo 11: Sistema Completo de Memoria Virtual

VAX/VMS. Memory management hardware. Segment-paging. PT of user segments (P0, P1) in kernel virtual memory (can be swapped). PT of kernel segment (S) in physical memory. VAX AS description. User - OS. Sharing between user space and OS. Context switch. Protection. Page replacement. PTE (Page Table Entry). Resident Set Size (RS). FIFO list of pages, dirty-page list, clean-page list. Clustering. Zeroing pages - lazy policy. COW.

Linux (x86). User AS. Code, heap, stack. Kernel AS. KLAS. KVAS. Page table structure. Large page support. TLB reach. Internal fragmentation. Page cache. From ASs (anonymous) and from memory-mapped file. Page cache hash table. Dirty pages, clean pages. pdflush. Replacement algorithms. Version of 2Q. Inactive list. Active list. Memory-mapped files. mmap. Security and buffer overflow. PTE entry to forbide execution. Return-oriented programming (ROP). Address Space Layout Randomization (ASLR). Meltdown, spectre.

Parte práctica

- Programación de sistemas: C / UNIX
- El núcleo de MINIX
- Llamadas al sistema en MINIX
- El núcleo de Linux

d. Métodos docentes

- Sesiones de aula: clases magistrales participativas y estudios de caso.
- Prácticas supervisadas en laboratorio con seguimiento en cada sesión de cada estudiante para poder realizar una evaluación continua del apartado práctico.

e. Plan de trabajo

Semana 1:

Aula: capítulos 1, 2 y 3

Laboratorio: entorno de desarrollo C en UNIX. Compilación, control de versiones, depuración, documentación. Práctica 1

Semana 2:

Aula: capítulos 4 y 5

Laboratorio: programación de utilidades de UNIX. Práctica 1.

Semana 3:

Aula: capítulo 6

Laboratorio: programación de utilidades de UNIX. Prácticas 1 y 2.

Semana 4:

Aula: capítulos 6 y 7

Laboratorio: programación de utilidades de UNIX. Prácticas 1 y 2.

Semana 5:

Aula: capítulos 7 y 8

Laboratorio: programación de utilidades de UNIX. Prácticas 1 y 2.

Semana 6:

Aula: capítulos 9 y 10

Laboratorio: programación de utilidades de UNIX. Prácticas 1 y 2.

Semana 7:

Aula: capítulo 11

Laboratorio: instalación y programación Minix. Práctica 3.

Semana 8:

Aula: repaso capítulos 1 a 11

Laboratorio: núcleo Minix I. Práctica 4.

Semana 9:

Aula: examen escrito semi-objetivo y de solución de problemas para el bloque 1

Laboratorio: núcleo Minix II. Práctica 5.

f. Evaluación

- Examen escrito semi-objetivo y de solución de problemas para el bloque 1.
- Examen práctico en laboratorio para la parte práctica.

g. Material docente

g.1 Bibliografía básica

- **Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau, “Operating Systems: Three Easy Pieces”, Arpaci-Dusseau Books. Versión on-line: http://ostep.org**

- A.Silberschatz, P.Galvin, G.Gagne, "Operating System Concepts. 8th Edition", Wiley, 2010. ISBN 978-0-470-23399-3.
- A.S.Tanenbaum, A.S.Woodhull, "Sistemas Operativos. Diseño e Implementación”, Segunda Edición, Prentice-Hall, 1997

g.2. Bibliografía complementaria

- A.S.Tanenbaum, "Sistemas Operativos Modernos. Tercera Edición.", Pearson/Prentice-Hall, 2009
- A.S.Tanenbaum, A.S.Woodhull, "Sistemas Operativos. Diseño e Implementación. El Libro de MINIX”, Tercera Edición, Prentice-Hall, 2006

g.3 Otros recursos telemáticos (píldoras de conocimiento, blogs, videos, revistas digitales, cursos masivos (MOOC), …)

h. Recursos necesarios

Aulas Virtuales Escuela de Informática (www.inf.uva.es)

http://www.minix3.org

i. Temporalización

BLOQUE TEMÁTICO	CARGA ECTS	PERIODO PREVISTO DE DESARROLLO
Blq. 1: gestión de memoria	3,6	Semanas 1 a 9

Bloque 2: Gestión del Almacenamiento

Carga de trabajo en créditos ECTS:	2,4

a. Contextualización y justificación

En los dispositivos de almacenamiento secundario se almacena la configuración de los sistemas operativos, el software de sistema y todos los datos de los usuarios. Por diseño estos dispositivos tienen mayor latencia y menor ancho de banda que el conjunto formado por el procesador y la memoria. En este bloque temático se estudia todo lo necesario para que el estudiante comprenda los problemas asociados al almacenamiento secundario.

b. Objetivos de aprendizaje

- Conocer los servicios que ofrece un SO específicas desde el punto de vista de un usuario, de un programador y de un administrador y utilizar diferentes sistemas de interacción con el SO.
- Diseñar e implementar aplicaciones basadas en los servicios del SO, seleccionando los más adecuados a cada caso.
- Conocer la influencia de los sistemas de gestión del almacenamiento y de la entrada/salida en el diseño de aplicaciones y en la utilización de los servicios que proporciona un SO.

c. Contenidos

- Capítulo 12: Dispositivos I/O

  System architecture. Bus hierarchy. Canonical device. Canonical protocol. Programmed I/O. Polling. Interrupts. Hardware interrupts. ISR. Direct Memory Access. OS-device interaction. in/out. Memory mapped I/O. Device driver. Example. Simple disk driver

- Capítulo 13: HDDs

Interface. Blocks. Logical numbers. Clusters. Basic geometry. Platter, surface, track, head, arm. Simple disk drive example. I/O time. Performance vs capacity. Workloads (sequential vs random). Disk scheduling. SSTF, SCAN, SPTF

- Capítulo 14: RAID

  Interface. Linear array of blocks. Logical requests. Fault model. RAID evaluation. Capacity. Reliability. Performance. RAID 0: stripping. Chunk size. Capacity, reliability, performance. RAID 1: mirroring. Capacity, reliability, performance. RAID 4: parity. Reconstruction. Capacity, reliability, performance. RAID5: rotating parity. Capacity, reliability, performance. Other considerations: other RAID levels, hot spare disks, data integrity, software RAID. Homework: raid.py

- Capítulo 15: Ficheros y Directorios

  File. Directory. FS interface. Creating files. Reading and writing files. Non-sequential R/W. Open File Table. Shared entries. Fsync. Renaming files. Getting information about files. stat. Removing files. strace. unlink. Making, reading and deleting directories. Hard links. Symbolic links. ACLs. Permission bits. Making and mounting FS

- Capítulo 16: Implementación del Sistema de Ficheros

  Model of a FS. Overall organization. Blocks, inodes, superblock, inode bitmap, block bitmap. Directory organization. Linear list, b-tree, hash. Free space management. Reading and writing files.Caching and buffering. Homework: vsfs.py

  Los capítulos 17, 18 y 19 se presentarán a los estudiantes como material de lectura opcional que puede formar parte de su trabajo práctico para subir nota.

- Capítulo 17: Sistemas de Ficheros Rápidos

  Poor performance. Seeks and latency. Fragmentation. Cylinder group. Allocation of files and directories. File locality in name space. Large files. Optimizations. Sub-blocks. libc optimizations. Homework: ffs.py

- Capítulo 18: Crash Consistency

Survive to crashes. Example. Crash scenarios. Atomicity. Solution #1: FS checker. Lazy approach: fsck. fsck phases. Solution #2: journaling. Logging. Transactions. Atomicity of transactions. Circular logs.Metadata journaling. Other approaches. Homework: fsck.py

- Capítulo 19: SSDs

SLC. MLC. Basic flash operations: read-erase-program. States: invalid, erased, valid. Performance. Reliability. Wear-out. SSD=flash chips + FTL + SRAM. log-structured FTL. Mapping table. Logical blocks, physical blocks, physical pages. Garbage collection. Mapping table. Page-based. Block-based. Hybrid. SSD performance and cost. Homework: ssd.py

Parte práctica

- Planificación de procesos en MINIX
- El gestor de memoria de MINIX
- El sistema de ficheros de MINIX
- Controladores de entrada/salida en MINIX: terminal, reloj y disco
- El núcleo de Linux

d. Métodos docentes

- Sesiones de aula: clases magistrales participativas y estudios de caso.
- Prácticas supervisadas en laboratorio con seguimiento en cada sesión de cada estudiante para poder realizar una evaluación continua del apartado práctico.

e. Plan de trabajo

Semana 10:

Aula: capítulos 12 y 13

Laboratorio: planificación de procesos en Minix. Práctica 6

Semana 11:

Aula: capítulos 13 y 14

Laboratorio: gestión de memoria en Minix. Práctica 7

Semana 12:

Aula: capítulos 14 y 15

Laboratorio: realización prácticas 3 a 7.

Semana 13:

Aula: capítulos 15 y 16

Laboratorio: realización prácticas 3 a 7. Práctica abierta opcional.

Semana 14:

Aula: capítulo 16. Presentación caps. 17, 18 y 19

Laboratorio: realización prácticas 3 a 7. Práctica abierta opcional

Semana 15:

Tutorías, resolución de dudas, defensa de los proyectos de laboratorio y de teoría

f. Evaluación

- Examen escrito semi-objetivo y de solución de problemas.
- Examen práctico en el laboratorio para la parte práctica de la asignatura.

g. Material docente

g.1 Bibliografía básica

- **Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau, “Operating Systems: Three Easy Pieces”, Arpaci-Dusseau Books. Versión on-line: http://ostep.org**
- A.Silberschatz, P.Galvin, G.Gagne, "Operating System Concepts. 8th Edition", Wiley, 2010. ISBN 978-0-470-23399-3.
- A.S.Tanenbaum, A.S.Woodhull, "Sistemas Operativos. Diseño e Implementación”, Segunda Edición, Prentice-Hall, 1997

g.2 Bibliografía complementaria

- A.S.Tanenbaum, "Sistemas Operativos Modernos. Tercera Edición.", Pearson/Prentice-Hall, 2009
- A.S.Tanenbaum, A.S.Woodhull, "Sistemas Operativos. Diseño e Implementación. El Libro de MINIX”, Tercera Edición, Prentice-Hall, 2006

g.3 Otros recursos telemáticos (píldoras de conocimiento, blogs, videos, revistas digitales, cursos masivos (MOOC), ...)

h. Recursos necesarios

http://www.minix3.org

Aulas Virtuales Escuela de Informática (www.inf.uva.es)

i. Temporalización

BLOQUE TEMÁTICO	CARGA ECTS	PERIODO PREVISTO DE DESARROLLO
Blq. 2: gestión del almacenamiento	2,4	Semanas 10, 11, 12, 13, 14, 15

5. Métodos docentes y principios metodológicos

Se utilizan diferente metodología docente para la teoría y los laboratorios:

- **Teoría**. La metodología docente principal se basa en las clases teóricas. El estudiante deberá acudir a estas clases con una actitud activa plasmada en las siguientes cuestiones:
  - Estudio previo de la materia de cada clase para que le resulte más fácil su comprensión.
  - Toma de notas sobre los aspectos más relevantes.
  - Participación en las preguntas planteadas por el profesor.
  - Resolución de las principales dudas sobre la materia.

  Adicionalmente, y de forma optativa, se plantea una metodología basada en el desarrollo de un trabajo teórico/práctico. Se propone al estudiante realizar un trabajo de forma paralela al desarrollo de la docencia y con mucha relación con ésta, de forma que se complementen mutuamente y se mejore el aprendizaje. Este trabajo es valorado en la nota final de la asignatura.

- **Laboratorios**. La metodología está orientada al desarrollo de proyectos y la evaluación es continua. El estudiante dispone de guiones detallados de las prácticas y el profesor explicará su realización durante los laboratorios. El estudiante deberá comprender los conceptos explicados y practicar con ellos por su cuenta. El laboratorio se estructura en dos partes, cada una de ellas tiene una práctica que el estudiante debe realizar por su cuenta de forma individual. Se establecerá una fecha de entrega de cada práctica. Para comprobar que el estudiante ha adquirido las competencias de laboratorio se hará un examen práctico de laboratorio por cada una de las dos partes.

6. Tabla de dedicación del estudiante a la asignatura

ACTIVIDADES PRESENCIALES	HORAS	ACTIVIDADEs NO PRESENCIALES	HORAS
Clases teórico-prácticas (T/M)	28	Estudio y trabajo autónomo individual	65
Clases prácticas de aula (A)		Estudio y trabajo autónomo grupal	25
Laboratorios (L)	30		
Prácticas externas, clínicas o de campo			
Seminarios (S)			
Tutorías grupales (TG)			
Evaluación	2		
Total presencial	**60**	**Total no presencial**	**90**

7. Sistema y Características de la Evaluación

INSTRUMENTO/PROCEDIMIENTO	PESO EN LA NOTA FINAL	OBSERVACIONES
Prueba de evaluación parcial	40%	Teórico objetivo, semiobjetivo y de resolución de problemas. Correspondiente al bloque 1. Se realizará en la semana 9.

Prueba práctica de laboratorio	30%	El estudiante tendrá que realizar dos entregas, una para la parte 1 y otra para la parte 2. Además tendrá que hacer dos exámenes de laboratorio, uno para la parte 1 y otro para la parte 2. Cada parte contabiliza un 15% de la nota final de la asignatura. Los criterios de evaluación se detallan más abajo. No es recuperable en convocatoria extraordinaria.
Examen final ordinario	30%	Teórico objetivo, semiobjetivo y de resolución de problemas. Realizado en periodo ordinario de exámenes.
Examen final extraordinario	70%	Igual que el examen ordinario.

Criterios de calificación:

- **Teoría:** las pruebas de evaluación objetivas/semiobjetivas darán lugar a una puntuación que se sumará directamente a la nota final del estudiante. Calificación máxima 7 puntos.
- **Laboratorio:**
  - Calidad del código, funcionalidad de la práctica y la comprensión de la materia mostrada en la realización. Se asignará un valor entre 0 y 10 puntos para cada parte
  - Cada parte tiene una prueba de laboratorio. En dichas pruebas se pedirá al estudiante que realice una modificación significativa de su práctica. Si la modificación se hace con éxito y se comprueba que funciona correctamente, la nota final será la indicada en el punto anterior. Si la modificación no se consigue, la nota final será una fracción de la nota indicada en el punto anterior.
  - Calificación máxima 3 puntos
- **Nota final: suma de la nota de teoría y laboratorio. Se aprueba si la nota de teoría es de, al menos, 3 puntos y la suma de teoría más laboratorio es de al menos 5 puntos sobre 10.**

8. Consideraciones Finales