UNIVERSIDAD NACIONAL DE INGENIERÍA

SÍLABO

INFORMACIÓN GENERAL

ASIGNATURA	:SISTEMAS OPERATIVOS
DOCENTE	:GERSON GARRIDO
HORAS POR SEMANA	:06 (TEORÍA: 04, LABORATORIO: 02)

SUMILLA

El curso es de naturaleza teórico-práctica y estudia cómo un sistema operativo convierte un hardware limitado y compartido en una plataforma segura, eficiente y cómoda para los programas. Se organiza en torno a tres ideas centrales: **virtualización** de la CPU y de la memoria (procesos, planificación, traducción de direcciones, paginación y memoria virtual); **concurrencia** (hilos, locks, variables de condición, semáforos, errores de concurrencia y deadlock); y **persistencia** (dispositivos de entrada/salida, discos, RAID, sistemas de archivos, journaling y SSDs). Se complementa con sistemas distribuidos (NFS, AFS), seguridad (autenticación, control de acceso, explotación de software y criptografía) y máquinas virtuales. El laboratorio se desarrolla en C sobre Linux y sobre xv6-riscv, un kernel educativo moderno tipo Unix para la arquitectura abierta RISC-V, en el que el estudiante implementa llamadas al sistema y modifica el planificador, la memoria virtual y el sistema de archivos.

COMPETENCIAS

- Explica cómo el sistema operativo virtualiza la CPU mediante procesos, ejecución directa limitada (*limited direct execution*) y planificación (*scheduling*), y compara políticas de planificación según su tiempo de retorno y de respuesta.
- Calcula traducciones de direcciones con segmentación, paginación, TLB y tablas multinivel, y evalúa políticas de reemplazo de páginas.
- Desarrolla programas concurrentes correctos en C con `pthreads`, identificando condiciones de carrera (*race conditions*) y deadlocks, y seleccionando la primitiva de sincronización adecuada.
- Describe el funcionamiento de discos, RAID, SSDs y sistemas de archivos, incluyendo la consistencia ante fallos (*crash consistency*) mediante journaling.
- Explica los principios de los sistemas de archivos distribuidos y los mecanismos de seguridad de un sistema operativo.
- Modifica un kernel real (xv6-riscv) agregando llamadas al sistema y cambiando su planificador, su memoria virtual y su sistema de archivos.

UNIDADES DE APRENDIZAJE

Unidad	Contenido	Semanas	OSTEP
I	Introducción y virtualización de la CPU	1–4	Cap. 1–11
II	Virtualización de la memoria	5–8	Cap. 12–24
III	Máquinas virtuales y concurrencia	9–12	Apéndice, Cap. 25–34
IV	Persistencia	12–15	Cap. 35–46
V	Sistemas distribuidos y seguridad	15–16	Cap. 47–57

PROGRAMACIÓN SEMANAL

Cada semana comprende dos sesiones de teoría de 2 horas (A y B) y una sesión de laboratorio de 2 horas. Los capítulos indicados son lectura previa obligatoria. Las sesiones marcadas con (*) pueden pasar a lectura autónoma si se pierden clases por feriados.

Sem.	Sesión A	Sesión B	Laboratorio
1	Presentación del curso. Introducción. Estructura del kernel (cap. 1–2)	Procesos (cap. 3–4)	Entorno de trabajo: Docker, C, `gcc`, `make`, `gdb`. *P0 publicado*
2	API de procesos: `fork`, `exec`, `wait` (cap. 5)	Ejecución directa limitada (cap. 6)	Programación con procesos; `strace`
3	Planificación: FIFO, SJF, STCF, RR (cap. 7)	Multi-Level Feedback Queue (cap. 8). Tiempo real: RM y EDF	Simuladores de planificación. *Entrega P0, publicado P1*
4	Planificación por lotería y stride; CFS (cap. 9)	Planificación multiprocesador (cap. 10–11)	xv6-riscv: arranque, depuración con gdb y primera llamada al sistema
5	Espacios de direcciones. API de memoria (cap. 12–14)	Traducción de direcciones. Segmentación (cap. 15–16)	xv6: el planificador por dentro. *P2 publicado*
6	Gestión de espacio libre (cap. 17)	Paginación (cap. 18)	Simuladores de relocalización y segmentación. *Entrega P1*
7	TLB (cap. 19)	Tablas de páginas multinivel (cap. 20)	Simuladores de paginación; medición de la TLB
8	Swapping: mecanismos y políticas. Sistemas de VM completos (cap. 21–24)	**Examen parcial**	Reemplazo de páginas; `vmstat`. *Entrega P2, publicado P3*
9	Máquinas virtuales (apéndice) (*)	Hilos y API de hilos (cap. 25–27)	Simulador `x86.py`; `pthreads` y `helgrind`
10	Locks (cap. 28)	Estructuras de datos concurrentes (cap. 29)	Locks y escalabilidad. *Entrega P3, publicado P4*
11	Variables de condición (cap. 30)	Semáforos (cap. 31). Inversión de prioridad	Productor/consumidor; lectores/escritores
12	Errores de concurrencia; deadlock: algoritmo del banquero y detección. Concurrencia por eventos (cap. 32–34)	Dispositivos de E/S. Discos duros (cap. 35–37)	Deadlocks; simulador de disco
13	RAID (cap. 38)	Archivos y directorios (cap. 39)	RAID; API de archivos. *Entrega P4, publicado P5*

Sem.	Sesión A	Sesión B	Laboratorio
14	Implementación de sistemas de archivos (cap. 40)	FFS. FSCK y journaling (cap. 41–42)	Simuladores de FS; sistema de archivos de xv6
15	LFS. SSDs. Integridad de datos (cap. 43–46)	Sistemas distribuidos. NFS. AFS (cap. 47–51)	Simuladores LFS, SSD, checksums y AFS; UDP
16	Seguridad: autenticación y control de acceso (cap. 52–55)	Explotación de software y endurecimiento. Criptografía y seguridad distribuida, de lectura (cap. 56–57) (*)	Permisos, ACLs y contraseñas. *Entrega P5*
17	**Examen final**		

METODOLOGÍA

- **Teoría:** clases expositivas con discusión de casos, ejercicios resueltos en clase y demostraciones en vivo. El estudiante lee el capítulo correspondiente antes de cada sesión.
- **Laboratorio:** prácticas guiadas en un contenedor Linux común para todos los estudiantes, con los simuladores del libro de texto, programación en C y modificación del kernel xv6-riscv, que corre en QEMU.
- **Proyectos:** seis proyectos de programación (C en Linux y xv6-riscv) con pruebas automáticas.
- Los términos técnicos se presentan en español junto a su equivalente en inglés, idioma de la bibliografía y de la documentación técnica.

Proyectos

N.º	Proyecto	Tipo	Semanas
P0	Utilidades Unix (`wcat`, `wgrep`, `wzip`, `wunzip`)	C	1–3
P1	Intérprete de comandos (`wish`)	C	3–6
P2	Llamadas al sistema y planificador por lotería	xv6-riscv	5–8
P3	Tablas de páginas y superpages	xv6-riscv	8–10
P4	Compresión paralela (`pzip`)	C + pthreads	10–13
P5	Archivos grandes y enlaces simbólicos	xv6-riscv	13–16
Bono	`fork` con copy-on-write	xv6-riscv	11–16

POLÍTICAS

- **Trabajo en parejas:** P0 es individual. P1–P5 y el bono se pueden hacer en parejas.
- **Integridad académica:** se puede discutir ideas, pero el código entregado debe ser propio. Se usarán herramientas de detección de similitud. Copiar soluciones publicadas en internet cuenta como plagio.

Uso de herramientas e IA

Herramienta	¿Se permite en este curso?
Diagnósticos del compilador y sanitizers (`gcc -Wall`, `-fsanitize`, Valgrind, Helgrind, gdb)	Sí, en todo el trabajo del curso.
Documentación oficial (páginas man, OSTEP, libro de xv6, documentación del kernel) y búsquedas en la web	Sí, citando la fuente de las ideas o el código que no son propios.
Autocompletado del editor que no genera soluciones enteras	Sí.
Corrector ortográfico, traductores y herramientas de accesibilidad	Sí.
Formateadores y linters de código	Sí.
IA generativa para pedir explicaciones de conceptos, sin código generado	Sí, para estudiar; la respuesta no se copia en ninguna entrega.
IA generativa que escribe código	No, en ningún proyecto ni laboratorio. Entregar código producido con ella cuenta como plagio.

Quien use una herramienta permitida más allá de la documentación y el compilador lo declara en el `README.md` de la entrega: qué herramienta, para qué y qué cambió en su trabajo a partir de eso. En cualquier proyecto o laboratorio se puede pedir a cada estudiante que explique su código de forma oral: cada integrante responde por todo lo que entrega.

BIBLIOGRAFÍA

Texto principal

1. Arpaci-Dusseau, R. H. y Arpaci-Dusseau, A. C. (2018). *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books. Disponible gratuitamente en `https://pages.cs.wisc.edu/~remzi/OSTEP/`

Textos complementarios

2. Silberschatz, A., Galvin, P. B. y Gagne, G. (2018). *Operating System Concepts* (10.ª ed.). Wiley.
3. Tanenbaum, A. S. y Bos, H. (2022). *Modern Operating Systems* (5.ª ed.). Pearson.
4. Cox, R., Kaashoek, F. y Morris, R. (2025). *xv6: a simple, Unix-like teaching operating system* (RISC-V, rev. 5). MIT. `https://pdos.csail.mit.edu/6.1810/2025/xv6/book-riscv-rev5.pdf`
5. Kerrisk, M. (2010). *The Linux Programming Interface*. No Starch Press.

Recursos en línea

- Tareas y simuladores: `https://github.com/remzi-arpacidusseau/ostep-homework`
- Proyectos: `https://github.com/remzi-arpacidusseau/ostep-projects`
- Código del libro: `https://github.com/remzi-arpacidusseau/ostep-code`
- Kernel xv6-riscv: `https://github.com/mit-pdos/xv6-riscv`
- Laboratorios de MIT 6.1810: `https://pdos.csail.mit.edu/6.1810/2025/`
- Sitio del curso: diapositivas, apuntes, código y enunciados.