Carrera INGENIERIA EN INFORMATICA		
Asignatura SISTEMAS OPERATIVOS		
Trayecto Infraestructura		
Año académico 2025	**Segundo Cuatrimestre**	
Responsable / Jefe de catedra Lic. Fabio E. Rivalta		
Carga horaria semanal 4	**Carga horaria total** 64	**Créditos**
Modalidad: presencial / virtual on-line		
Correlativas anteriores Arquitectura de Computadoras	**Correlativas posteriores** Virtualización de Hardware Seguridad Aplicada y Forensia	
Conocimientos necesarios		

Equipo docente		
Nombre	**Cargo**	**Titulo**
Fabio E. Rivalta	Profesor Adjunto	Lic. En Sistemas de Computación
Leonardo Catalano	Profesor Adjunto	Ing. En Informática
Alexis Villamayor	Profesor Adjunto	Ing. En Informática
Ramiro de Lizarralde	Jefe de trabajos prácticos	Ing. En Informática
Federico Loiacono	Ayudante de 1°	Ing. En Informática
Fernando Piubel	Ayudante de 1°	Ing. En Informática
Alejandro Rodriguez	Ayudante de 1°	Ing. En Informática

Trabajo Práctico (GRUPAL):

Pautas de desarrollo y entrega:

- Debe estar desarrollado en C o C++
- Se puede utilizar IA como herramienta para apoyo y/o generación del código
- Entregar el código fuente y un makefile para poder compilarlo
- No se toman por válidos binarios precompilados
- Entregar lote de prueba utilizado, junto con los archivos necesarios para poder probarlo, en caso de que sea necesario.
- Herramientas utilizadas para validar el monitoreo de los procesos.
- Todo lo relacionado a la entrega debe estar en un archivo comprimido **ZIP** y ser entregado por la plataforma **MIeL**.

Ejercicios:

Ejercicio 1: Generador de Datos de Prueba con Procesos y Memoria Compartida

Enunciado:
Se debe desarrollar un sistema de generación de datos mediante procesos en paralelo.
Un **proceso coordinador** administra la asignación de identificadores y la escritura de registros en un archivo **CSV**, mientras que **N procesos generadores** producen datos de prueba.

Requisitos:

- El programa debe permitir especificar por parámetro cantidad de **procesos generadores** y **cantidad total de registros** a generar.
- Cada generador solicita al proceso coordinador los próximos 10 IDs válidos.
- Con cada ID, genera un registro con **datos aleatorios** (por ejemplo: valores tomados de listas predefinidas o números generados en un rango).
- Cada registro es enviado **de a uno por vez** al coordinador mediante **memoria compartida (SHM)**.
- El coordinador guarda el registro recibido en el archivo **CSV**.
- El archivo CSV debe:
    - Tener en la **primera fila** el nombre de las columnas.
    - Incluir obligatoriamente el **ID como primer campo**.
    - El resto de los campos pueden ser definidos libremente.
- No es necesario que los IDs queden ordenados en el CSV; se registran en el orden en que son recibidos.

Monitoreo en Linux:

- Uso de ipcs o /dev/shm, ps, htop, vmstat para evidenciar creación de procesos, memoria compartida y concurrencia.
- Limpieza de recursos IPC al finalizar.

Criterios de corrección:

- Se verificará con un script AWK:
    - Que los IDs sean correlativos y no presenten saltos en su numeración.
    - Que no existan IDs duplicados.
- Se deben validar correctamente los parámetros y mostrar ayuda en caso que no se informen o sean incorrectos.
- Ante la finalización prematura de un proceso, el resto de los procesos deben responder de forma controlada, pudiendo continuar su ejecución o bien finalizar dependiendo qué proceso finalizó.
- No dejar abiertos recursos en el sistema (semáforos, memorias compartidas, archivos temporales, etc.) una vez que finalicen todos los procesos.

Ejercicio 2: Cliente-Servidor de Micro Base de Datos con Transacciones

Enunciado:
Se debe implementar un sistema **cliente-servidor** que utilice **sockets** para permitir consultas y modificaciones remotas sobre el archivo CSV generado en el Ejercicio 1.

Requisitos:

- El servidor debe aceptar hasta **N clientes concurrentes** y mantener hasta **M clientes en espera** (ambos por parámetro).
- El servidor utiliza el archivo CSV como **base de datos**.
- Los clientes deben ejecutarse manualmente y ser **interactivos**, manteniéndose conectados hasta que el usuario decida salir.
- El cliente puede enviar:
    - **Consultas** (ej. búsquedas, filtros, etc.).
    - **Modificaciones de datos** (alta, baja, modificación).

  - El diseño del protocolo de comunicación y de las operaciones (consultas y DML) queda a decisión del grupo.
- **Transacciones:**
  - El cliente puede iniciar una transacción con BEGIN TRANSACTION.
  - Mientras tiene la transacción abierta, obtiene un **bloqueo (lock) exclusivo** sobre el archivo.
  - Puede realizar varias modificaciones de registros.
  - Debe confirmar con COMMIT TRANSACTION.
  - Durante una transacción:
    - Ningún otro cliente puede realizar consultas ni modificaciones.
    - Si otro cliente lo intenta, debe recibir un error indicando que existe una transacción activa y que debe reintentar luego.
- El servidor permanece a la espera de nuevos clientes luego que todos los clientes se desconectaron.
- Debe manejar correctamente cierres inesperados (caídas de clientes o del servidor).
- La dirección IP, o nombre de host, y el puerto de escucha se debe poder informar por parámetro o archivo de configuración.

Monitoreo en Linux:

- Uso de netstat, lsof, ps, htop para documentar la creación de procesos, threads y la gestión de sockets.
- Evidenciar concurrencia de clientes y bloqueos por transacciones.
- Limpieza de recursos y archivos temporales al finalizar.

Criterios de corrección:

- Se deben validar correctamente los parámetros y mostrar ayuda en caso de que no se informen o sean incorrectos.
- Los procesos pueden conectarse por la red TCP/IP
- Ante la finalización prematura de un proceso, el resto de los procesos deben responder de forma controlada, pudiendo continuar su ejecución o bien finalizar dependiendo qué proceso finalizó.
- No dejar abiertos recursos en el sistema (semáforos, memorias compartidas, archivos temporales, etc.) una vez que finalicen todos los procesos.