Sistemas Operativos

Departamento de Computación – FCEyN – UBA

Segundo cuatrimestre de 2023

Nombre y apellido: _____

Nº orden: _____ L.U.: _____ Cant. hojas: _____

1	2	3	4	Nota

Primer parcial – 03/10 - Segundo cuatrimestre de 2023

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido, LU y número de orden.
- **Cada código o pseudocódigo debe estar bien explicado y justificado en castellano. ¡Obligatorio!**
- Toda suposición o decisión que tome deberá justificarla adecuadamente. Si la misma no es correcta o no se condice con el enunciado no será tomada como válida y será corregida acorde.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

Ejercicio 1.- Sincronización y concurrencia. **(25 puntos)**
Se ha inaugurado una nueva platea en River para aumentar la capacidad del concierto de *Taylor Swift*. Nuevos *tickets* saldrán a la venta y nos piden modelar la cola de espera del sistema web de ventas de la siguiente manera:

- Hasta 10 usuarios pueden entrar de manera concurrente a comprar entradas.
- Si cuando intenta entrar un cliente hay lugar (menos de 10 usuarios) puede pasar directo a la página de compras.
- Si está completa la capacidad de la cola, se debe esperar a que ésta se descongestione, es decir, debe llegar a 0 para poder dejar ingresar nuevos clientes.
- No se pide un orden explícito, pero sí considerar que no debe ocurrir que cuando se descongestione la fila, una persona que no haya estado en la fila pueda ingresar adelantándose a las personas que sí estaban esperando; si no, sería injusto.

Asumir que existe una función *comprar_ticket()*, y solo nos encargaremos de modelar el acceso a la compra y su correcta sincronización entre clientes, no la compra de entradas en sí.

Ayuda: puede utilizar la función min(a,b) que retorna el mínimo entre a y b.

Ejercicio 2.- Scheduling. **(25 puntos)**
Se está diseñando un sistema de videojuego de estrategia al que llamaremos SOCraft. En este sistema hay procesos interactivos dado que se usan el teclado y el *mouse* para realizar las acciones o comandos rápidos, y también procesos *real-time* que son las unidades oponentes o enemigas que incorpora el juego. El sistema se maneja usando constantemente varios procesos interactivos, para lograr la interacción del jugador, y con menos frecuencia aparecen los procesos *real-time*, que son todos los movimientos de los elementos que incluye el juego y deben tener respuestas en tiempo real. Estos procesos ejecutan a los personajes (oponentes) precargados del juego. Se espera que los procesos interactivos no queden muy rezagados con respecto a los procesos *real-time* y así se puedan ejecutar de forma tal que le den a cada jugador la sensación de interacción real.

Se pide diseñar una política de *scheduler* para este juego. Explique brevemente y justifique el diseño propuesto e indique cómo administraría cada tipo de proceso que llegue a la cola de *ready*.

Ejercicio 3.- Gestión de memoria: **(25 puntos)**
Se tiene la siguiente secuencia de solicitudes de página por parte de un proceso:

$2, 1, 7, 3, 0, 5, 2, 1, 2, 9, 5, 7, 3, 8, 5$

Considerando que:

- Se tienen 4 *frames* disponibles.
- En los *frames* ya se encuentran cargadas las siguientes páginas:

Frame 1	Frame 2	Frame 3	Frame 4
0	1	2	3

- Las páginas se solicitaron en el siguiente orden $0, 1, 2, 3$

Evaluar el comportamiento de los algoritmos de reemplazo de páginas *LRU* y *Second Chance*, indicando para cada solicitud de página el estado de la cola de reemplazos de páginas y el estado de la memoria (*frames*). Finalmente se pide que calcule la tasa (promedio) de *page faults* (fallos de página).

Ejercicio 4.- Comunicación Inter-procesos **(25 puntos)**
Desde el Departamento de Computación se inauguraron **m** laboratorios nuevos y necesitamos desarrollar un nuevo sistema para que **n** estudiantes sepan cuál número de laboratorio le corresponde. La asignación se hace de manera aleatoria. Se supone que esta aleatoriedad permitirá una distribución uniforme de los labos.

Para eso se pide desarrollar un sistema utilizando comunicación entre procesos mediante pipes y que cumpla con el siguiente criterio: usando programación en el lenguaje C, crear un código que ejecute un proceso (padre) que se encargará de pedirle a una cantidad **n** de procesos hijos (el número **n** será pasado como parámetro, y representa el número de estudiantes) que calculen qué laboratorio le corresponde a cada une. Una vez hecho eso, cada proceso hijo tendrá que comunicarle al padre el número de laboratorio que le corresponde. La forma en que cada proceso asigna un laboratorio es aleatoria, usando como semilla el id del proceso hijo. Para esto, el padre debe informarle a sus hijos cuántos laboratorios nuevos hay (el número **m**, pasado como parámetro en la entrada estándar del padre). Al finalizar, el padre debe imprimir los números de asignación de aula a cada proceso por salida estándar, mostrando el id del proceso hijo y el número del labo asignado.

a) Realizar el código en C que cumpla los requisitos de la comunicación. Justificar detalladamente la decisión que se toma en cuanto al cierre de pipes.
b) Se pide modificar el código anterior para realizar el siguiente comportamiento. Nos piden que, una vez asignados los laboratorios, el código del padre ya no las imprima por pantalla. En cambio se debe ejecutar un programa que nos proveen llamado "nuevaSalida". Este programa ejecutable toma como parámetros por entrada estándar el id de un proceso y el número de laboratorio asignado y lo imprime en pantalla (salida estándar), sin embargo, queremos que el resultado de este programa no sea escrito en el STDOUT sino en un archivo llamado "resultados.out" en donde guardaremos un log de la ejecución. Justifique la política de cierres de pipes.

Ayuda: Pueden usar las siguientes funciones auxiliares:

```
int open(char* archivo, O_WRONLY) // abre el archivo en modo de escritura, retornando el descriptor
     correspondiente.
int dameLabo(int process_id, m) // toma el process_id correspondiente a un proceso hijo de un
    alumno y el número de laboratorios nuevos "m" y devuelve aleatoriamente un número de labo
    asignado, entre 1 y m, usando como semilla el process_id enviado.
```