UNIVERSIDAD DE GUADALAJARA

Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en Computación

6to semestre

Tema: Simulación Procesamiento por Lotes con Multiprogramación

Materia:
Seminario de Sistemas Operativos

Sección:
D02

NRC:
103845

PRESENTA:
Saúl Alejandro Castañeda Pérez
Daniel Martínez Martínez

Docente:
VIOLETA DEL ROCIO BECERRA VELAZQUEZ

Fecha de entrega:
Domingo, 19 de febrero de 2023

Act de aprendizaje 4 - Simulación de procesamiento por lotes con multiprogramación

Objetivo

Desarrollar un programa que simule el procesamiento por lotes con multiprogramación, satisfaciendo los requerimientos planteados en Moodle.

Desarrollo

Los requerimientos planteados para la actividad son:

1. El programa preguntará el número de trabajos inicial, para luego conformar lotes con capacidad máxima de 4 (ver Requerimientos programa 1).
2. En esta simulación la información de cada trabajo se generará internamente, es decir ya no será necesario introducir datos desde teclado, la información a generar es:
    a. Número de cada proceso, validar que sea único. (Puede ser consecutivo)
    b. Tiempo Máximo Estimado, se generará de forma aleatoria, validando que sea mayor a 0 (rango entre 5 y 16).
    c. Operación (+, -, *, /, residuo) a realizar con sus operandos (validar operaciones), aleatoriamente.

- Se implementó el llenado automático de los campos para cada proceso.
- El ID se genera haciendo uso de la función *sample*, la cual retorna un elemento aleatorio no repetido de un rango o conjunto dados.
- La operación se genera aleatoriamente al generar ambos operandos con la función *random.randint*, mientras que el operador se selecciona de un conjunto con la función *random.choice*.

3. Los trabajos se ejecutarán conforme a su número asignado (ID).

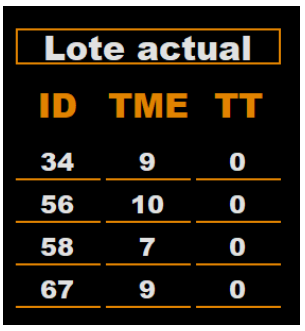
- Se aplica la función .sort para ordenar la lista de procesos. Como el ID es el primer atributo de cada registro, el ordenamiento se hace por predeterminado con respecto a ese dato.

4. Los procesos una vez en ejecución pueden ser interrumpidos por Entrada/Salida o bien terminados por Error (Observar la diferencia en la tabla).
5. La interrupción y la terminación de los trabajos en ejecución se generarán por medio de teclas.
6. Las teclas para utilizar son:

Tecla	¿Qué indica?	¿Qué hace?
I	Interrupción por entrada-salida	El proceso que está en uso del procesador (ejecución) debe salir de este e ir a la cola de los procesos del lote en ejecución (actual).
E	Error	El proceso que se esté ejecutando en ese momento terminara por error, es decir saldrá del procesador y se mostrara en terminados, para este caso como el proceso no termino normalmente se desplegara error en lugar de un resultado.
P	Pausa	Detiene la ejecución de su programa momentáneamente, la simulación se reanuda cuando se presione la tecla "C".
C	Continuar	Al presionar esta tecla se reanudará el programa pausado previamente con "P".

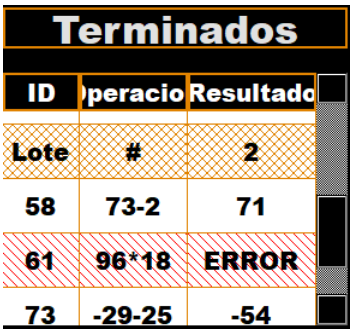
- Detección de teclas apretadas: Se utilizó la función detectora de eventos *event.text propia de la Interfaz Gráfica* PYQT5, la cual evalúa si la tecla pulsada coincide con alguna de las letras ofrecidas como opciones en la tabla de arriba (I, E, P, C, i, e, p, c).
- Determinar qué operación se va a aplicar en el proceso en ejecución: Para ejecutar alguna de las 4 acciones disparadas por las teclas señaladas, usamos un conjunto de banderas que cambian de estado según se les indique. Su comportamiento es el siguiente:
    - Si presionamos 'p' ó 'P' se establece en verdadero el valor de la bandera Bandera_P_C, indicando al programa que se debe poner en pausa el procesamiento de las tareas.
    - Si presionamos 'c' ó 'C' se pone en falso Bandera_P_C, indicando al programa que puede continuar procesando las tareas sin problema.
    - Si presionamos 'e' ó 'E' se pone en verdadero el valor de la bandera Bandera_Error, indicando que la tarea que se estaba procesando debe finalizar instantáneamente y marcarse con ERROR en la tabla de terminados. Cabe destacar que esta tecla sólo tiene efecto cuando Bandera_P_C se encuentra en falso, ya que eso indicaría que si es posible finalizar un proceso con estatus de error porque el programa si se encuentra procesando las tareas.
    - Si presionamos 'i' ó 'I' y la Bandera_P_C se encuentra en falso, se pone en verdadero el valor de BanderaInterrupción, indicando que la tarea en ejecución debe enviarse de regreso al final del lote actual.
        - Cuando una tarea es interrumpida, se pone en espera para ser procesada donde se quedó, respetando el tiempo de ejecución que había transcurrido.
        - La tarea interrumpida es enviada a la línea inicial donde apareció la tarea dentro del lote actual, además de que es enviada al final de la una lista denominada sublote, la cual almacena los 4 procesos que pertenecen a cada lote (algo que no se usaba en el programa anterior).

7. Deberá mostrarse en pantalla:
    a. Procesos en Espera correspondientes al Lote en ejecución:
        i. Número de Programa (en el programa anterior se listaba nombre del programador ahora es número de programa).
        ii. Tiempo Máximo Estimado.
        iii. Mostrar el tiempo transcurrido. En el caso de que el proceso no se haya ejecutado aún se desplegará 0.



b. Número de Lotes Pendientes: Especificar el número de lotes pendientes por ejecutar, si no hay se mostrará 0.
c. Proceso en Ejecución:
    i. Se mostrarán todos los datos correspondientes al proceso.
    ii. Tiempo que ya ha sido ejecutado.
    iii. Tiempo restante por ejecutar.

d. Trabajos Terminados:
    i. Número de Programa.
    ii. Operación
    iii. Resultado de la operación o bien ERROR cuando el proceso haya sido terminado con la tecla que genera error.



Conclusiones y notas

Esta práctica se nos facilitó bastante gracias a que reutilizamos gran parte del programa anterior, realmente solo fue necesario agregar unas cuantas mejoras para cumplir con los nuevos requerimientos. Las mejoras más notorias fueron:

- Generación aleatoria de datos.
- Ordenamiento de los datos con respecto al ID.
- La implementación de una función listener para las teclas.
- El uso de banderas globales para saber qué operación se puede aplicar a un proceso en ejecución.
- La implementación de retardos y de un estatus de error para los procesos (en caso de ser necesario).
- Implementación de una lista para los procesos que estén en el lote actual.