Práctica 3

Procesos – Hilos – Mutex

Datos útiles

Para realizar esta práctica puede utilizar la máquina virtual provista por la cátedra (se recomienda hacer un snapshot de la VM antes de empezar a usarla, para poder volver atrás en caso de que algo deje de funcionar).
El link de descarga se encuentra en el moodle junto a las credenciales de acceso. Además, la VM se encuentra instalada en las PCs de la Sala A del Lab. Bacarla donde se realizan las prácticas de la materia.

1) Hilos
    a) Explique la diferencia entre hilos y procesos.
    b) Enumere las principales ventajas del uso de hilos.
    c) ¿Cuáles son los estados que puede tomar un hilo? Explique.
    d) ¿Por qué es necesario que cada hilo tenga su propia pila? ¿Qué otros elementos forman parte del contexto particular de un hilo?
2)
    a) Explique el funcionamiento de la función pthread_create().En particular cuáles son argumentos opcionales y cuales son obligatorios.
    b) Idem para pthread_join() y pthread_yield().
    c) Explique cómo se accede a los resultados que devuelve un hilo mediante el uso de pthread_exit().
    d) Compile y ejecute el programa de ejemplo pthreads2.c. Analice su funcionamiento, en particular que información se pasa a través de los argumentos de las funciones pthread_join() y pthread_exit().
3)
    a) Explore el funcionamiento y utilidad de las funciones pthread_self() , pthread_detach(), pthread _equal(), pthread_attr_init() y de los tipos de variables pthread_attr_t y pthread_t.

b) Compare el funcionamiento de los dos programas pruebafork.c y pruebathreads.c mediante el comando time. ¿Puede llegar a alguna conclusión?
c) Analice los ejemplos pruebath2.c y pruebath3.c

4) Escriba un programa que permita sumar los valores de un vector de N elementos utilizando hilos o utilizando procesos.
    a) El valor de N se pasa en la línea de comandos y el programa debe comenzar, almacenando en el vector los valores de 1 a N. una vez realizado esto se queda esperando recibir una señal
    b) Si recibe SIG_USR1, crea un hijo, el cual realiza la suma y la imprime en pantalla.
    c) Si recibe la señal SIG_USR2, crea un hilo que realiza la suma y la devuelve al hilo principal, este hilo principal luego imprime el resultado.

5) Mutex
    a) Describa el proceso a seguir para crear y utilizar mutex en varios hilos.
    b) Describa en particular las funciones utilizadas para crear (create), adquirir (lock), soltar (unlock) y destruir (destroy) un mutex.
    c) Explique en que consiste el problema conocido como deadlock
    d) Describa la manera en que funciona la variante trylock
    e) Analice los ejemplos mutex1.c, mutex2.c y mutex3.c

6) **(Ejercicio entregable)**

Escriba un programa que utilice un arreglo de 20 `int`, compartido entre dos hilos A y B de manera tal que A escriba los primeros 10, y B los lea solamente después que se completó la escritura. De la misma manera B escribirá los 10 últimos enteros del arreglo y A los leerá solamente después que B complete la escritura. Este proceso se repetirá hasta que el hilo A reciba la señal `SIGTERM`, y cuando la reciba debe a su vez hacer terminar el hilo B, y liberar todos los recursos compartidos utilizados.

Los datos serán números entre 0 y 26, (representando letras entre la **a** y la **z**), que serán cifrados previo a la escritura mediante una función de cifrado de la forma $f(x) =$

$(ax + b)\ mod\ n$. En donde (a,b) serán claves públicas que estarán disponibles en memoria compartida. Para el descifrado de los datos se usará la función inversa $f^{-1}(x) = (kx + c) mod\ n$, en donde (k,c) son claves privadas conocidas solo por el proceso lector. Como función de cifrado se usará $f(x) = (4x + 5)\ mod\ 27$ y para el descifrado$f^{-1}(x) = (7x + 19)\ mod\ 27$

Ambos hilos realizarán primero la escritura de los datos en la zona correspondiente y luego la lectura. Ninguno de los dos puede escribir en su zona hasta que el otro no haya leído los datos, excepto la primera vez. La escritura en ambos casos se realizará a razón de un valor por segundo, mientras que la lectura se realizará tan rápido como se pueda y se imprimirá en pantalla junto con la identificación del hilo que está leyendo.

El programa debe ser único, es decir el mismo programa puede actuar como hilo A o como hilo B, dependiendo de un parámetro que se pase en la línea de comando. (por ejemplo “programa a” o “programa b”). Realice su implementación basada en variables compartidas y mutex.

Pautas de entrega

La resolución debe hacerse en base a las pautas explicadas en clase, es decir:

- Seleccione una plataforma de control de versiones (ej: GitLab, GitHub, Bitbucket, …)
- Dentro de la misma, debe crear un grupo:
  - Nombre: **`SOyR 2024 Grupo X`** (**`X`** representa el número de grupo).
  - Descripción: apellido, nombre y número de alumno de cada integrante del grupo.
  - Debe agregar al docente asignado con el rol de `Owner`.
- Dentro del grupo, debe crear un repositorio:
  - Nombre: **`Entregable Práctica X`** (`X` representa el número de práctica)
  - El repositorio debe contener lo siguiente:
    - Archivo de programa **`eX.Y`** (**`X`** representa el número de ejercicio. **`Y`** representa la extensión del archivo, ejemplo: **`c`**, **`sh`**) correctamente estructurado y con los comentarios pertinentes.
    - Archivo **`README.md`**. Usando sintaxis **`markdown`** incorporar lo siguiente:

    - Copia del enunciado
    - Interpretación del problema. (qué debo hacer, qué datos tengo y cómo se organizan, cuál es la interfaz con el usuario)
    - Propuesta de solución (Cómo resuelvo lo pedido) describo los modelos utilizados: algoritmos, matemáticos o conceptuales. Aproximar la solución con un diagrama o pseudocódigo, modularizar.
    - Instrucciones de uso
- Directorio `imagenes` ó `figuras` (opcional: en caso de que se requiera adjuntar imágenes en el `README.md`) con las figuras correspondientes.
- `.gitignore` (opcional: en caso de que haya trabajado con más archivos en local)