Departamento de Ciencias e Ingeniería de la Computación

Ingeniería en Computación
Ingeniería en Sistemas de Información

SISTEMAS OPERATIVOS

Segundo Cuatrimestre de 2018

Primer Proyecto

1. Proyecto: Experimentación de Procesos y Threads con los Sistemas Operativos

1.1. Procesos y threads

1. VALIDADOR DE SUDOKU. El juego de Sudoku utiliza una grilla de 9x9 en la cuál cada fila, columna y subgrillas de 3x3 contine todos los dígitos del 1 al 9. La figura 1 presenta un ejemplo de una jugada de Sudoku válida. Este problema consiste en diseñar un aplicación que determine si la jugada de Sudoku es válida o no.

   Se pide:

   a) Explique las decisiones de diseño que utilizó para modelar la resolución del problema con el objetivo de maximizar la concurrencia.

   b) La jugada de Sudoku debe cargarse desde un archivo denominado "sudoku.txt"que contenga 9 líneas dónde cada una contiene 9 números separados por comas.

   c) Implemente el Validador de Sudoku de acuerdo a las decisiones tomadas en el primer punto:

      1) Utilizando procesos.
      2) Utilizando hilos.

2. MINI SHELL. Construir un shell que acepte un conjunto limitado de comandos de Unix. Tiene que considerar como mínimo 6 comandos. Explique las opciones de diseño que consideró al momento de implementarlo. No puede invocar los comandos mediante la función system y para la implementación de los mismos debe utilizar llamadas al sistema (system call) ó funciones de librerías.

   Los mínimos comandos que debe tener son los siguientes

   a) Crear un directorio.

   b) Eliminar un directorio.

   c) Crear un archivo.

   d) Listar el contenido de un directorio.

   e) Mostrar el conenido de un archivo.

   f) Mostrar una ayuda con los comandos posibles.

6	2	4	5	3	9	1	8	7
5	1	9	7	2	8	6	3	4
8	3	7	6	1	4	2	9	5
1	4	3	8	6	5	7	2	9
9	5	8	2	4	7	3	6	1
7	6	2	3	9	1	4	5	8
3	7	1	9	5	6	8	4	2
4	9	6	1	8	2	5	7	3
2	8	5	4	7	3	9	1	6

Figura 1: Jugada SUDOKU

1.2. Sincronización

Nota: Recuerden NO utilizar variables globales para la sincronización.

1. Impresoras compartidas

   Suponga tener n usuarios que comparten 2 impresoras. Antes de utilizar la impresora, el U[i] invoca requerir(impresora). Esta operación espera hasta que una de las impresoras esté disponible, retornando la identidad de la impresora libre. Después de utilizar la impresora, U[i] invocan a liberar(impresora).

   a) Resuelva este problema considerando que cada usuario es un thread y la sincronización se realiza utilizando semáforos.

   b) Resuelve este problema considerando que cada usuario tiene una prioridad única. Se otorga la impresora al usuario con mayor prioridad que está esperando.

2. Asistente

   En un departamento de computación de una universidad existe un asistente de docencia (AD) que ayuda a los alumnos con sus tareas de programación durante las horas de consulta en su oficina. La misma es bastante pequeña y tiene espacio sólo para un escritorio y una computadora. Existen tres sillas en el pasillo fuera de la oficina donde los estudiantes pueden sentarse y esperar si el que el AD está ayudando a otro estudiante. Cuando no hay estudiantes que necesiten ayuda, el AD se sienta en el escritorio y duerme una siesta. Si un estudiante llega a la ofician durante el horario de consulta y encuentra al AD durmiendo, el estudiante debe despertar al AD para pedirle ayuda. Si un estudiante llega y encuentra al AD ayudando a otro estudiante, el primero se sienta en una de las sillas del pasillo y espera. Si no hay sillas disponibles, el estudiante se va para probar suerte más tarde.

   a) Resuelva este problema considerando que tanto el AD como cada uno de los estudiantes es un thread y la sincronización se realiza utilizando semáforos.

2. Problemas

1. Dado siguiente artículo:
2. "Best Practices for Operating System Deployment"

   a) Realice un comentario general del artículo (al menos 500 palabras).

   b) ¿Qué aspectos del artículo se relacionan con la carrera que están estudiando?¿Cómo podría afectarle en su futura profesión?

3. Se tiene un sistema operativo en el que se ejecutan los siguientes procesos:

   - Proceso P1, en el instante t=0, con dos threads de usuario sobre un thread de kernel. En este proceso cada uno de los dos threads de usuario computan durante 4 unidades de tiempo sin hacer entrada/salida y luego terminan.
   - Proceso P2, en el instante t=3, con dos threads de kernel. En este proceso, cada thread computa durante 2 unidades de tiempo, efectúa entrada/salida durante 4 unidades de tiempo, computa durante 1 unidad de tiempo y luego termina.
   - Proceso P3, en el instante t=4, sin threads de usuario, con un thread de kernel. Este proceso computa durante 1 unidad de tiempo, hace entrada/salida durante 1 unidad de tiempo, computa durante 1 unidad de tiempo y luego termina.

   Se pide:

   a) Suponiendo que la planificación del sistema es round-robin con un quantum de 2 unidades de tiempo, dibuje un diagrama (procesos/tiempo) donde se muestre en cada unidad de tiempo en qué estado está cada uno de los procesos.

   b) Responder a la misma pregunta anterior pero suponiendo que la planificación del sistema sigue el algoritmo de Sortest Remaining Time First (SJF apropiativo).

   c) Para cada uno de los incisos anteriores, calcule el tiempo de retorno de cada proceso y el tiempo de espera.

   d) Realice los incisos anteriores considerando que tiene 2 procesadores.

Indicaciones

- Los experimentos deben realizarse en lenguaje C.
- Las pruebas deben realizarse sobre el sistema operativo Linux (Red Hat/ Fedora/ Centos/ Suse/ Arch) para PC y Raspberry Pi. Puede traer una extensión en alguna otra distribución o arquitectura.
- Se debe entregar los fuentes realizados para cada una de las experiencias debidamente identificados (impresos y en disco) y un informe con los resultados obtenidos y las preguntas realizadas. Además se debe facilitar el compilado y ejecución de cada una de las experiencias por medio de un script o make y describiendo la forma de ejecución.