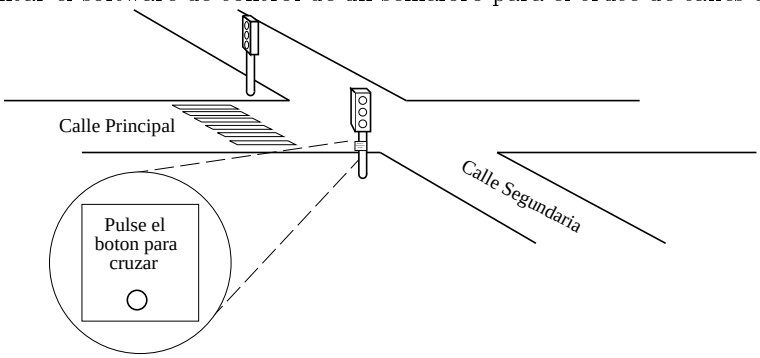
CC41B : Sistemas Operativos
Examen–Semestre Primavera'97
Prof.: Luis Mateu.

Pregunta 1

Se desea implementar el software de control de un semáforo para el cruce de calles de la siguiente figura:



La luz verde de la calle principal dura 60 segundos (si no hay peatones que deseen cruzar) mientras que la luz verde de la calle segundaria dura sólo 30 segundos ya que tiene menor circulación. La luz amarilla de ambas calles dura 3 segundos. Por lo tanto la luz roja de la calle principal dura 33 segundos y la de la calle segundaria 63 segundos.

La calle principal tiene además un cruce peatonal. Los peatones disponen de un botón que pueden presionar para acelerar la luz verde que les permite cruzar. Cuando se ha presionado este botón, la luz verde de la calle principal se reduce a 30 segundos. Si el boton se presiona pasado los 30 segundos de luz verde de la calle principal, ésta se coloca en amarillo de inmediato.

Ya se han implementado los procedimientos que controlan las luces de ambas calles. Al invocar `luzPrimVerde()` la luz de la calle principal se coloca en verde. Este procedimiento retorna de inmediato, dejando la luz en verde hasta que se invoque `luzPrimAmarilla()` que la coloca en amarillo. Al invocar `luzPrimRoja()` la luz queda en rojo hasta que se invoque nuevamente `luzPrimVerde()`. Análogamente los procedimientos `luzSegVerde()`, `luzSegAmarilla()` y `luzSegRoja()` sirven para controlar el color de la luz de la calle segundaria.

Además se ha implementado el procedimiento `esperarBoton()` que bloquea la tarea que lo invoca hasta que un peatón presione el botón.

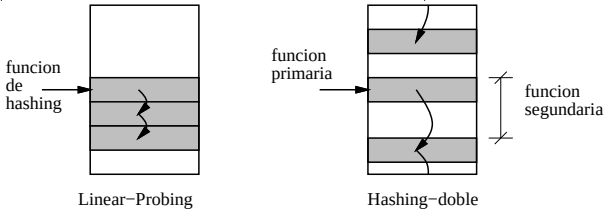
Implemente un procedimiento que controle las luces del cruce usando tareas de nSystem. Ud. puede lanzar tareas auxiliares si lo estima necesario.

Pregunta 2

- **Parte a.-**

  Se le ha pedido a Ud. optimizar una aplicación que pasa la mayor parte del tiempo haciendo búsquedas en un diccionario. Para acelerar las búsquedas Ud. ha decidido utilizar una tabla de hashing cerrada.

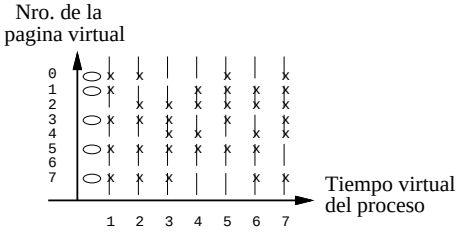
  Su dilema consiste en decidir si usar linear-probing para las colisiones o más bien hashing-doble. En linear-probing, en caso de colisión se busca secuencialmente hacia abajo a partir de la fila indicada por la función de hashing. En hashing-doble, se calcula una segunda función de hashing que se usa como incremento de la fila (módulo el tamaño primo de la tabla).



Se estima que la tabla está un 75 % llena. Los análisis matemáticos indican que con linear-probing habrá que visitar en promedio 2.5 filas por cada búsqueda, mientras que con hashing doble se nesitará visitar 1.8 filas por cada búsqueda.

Haga una estimación del tiempo promedio de búsqueda (en milisegundos) en cada caso considerando que la tabla es grande, se trata de un sistema que implementa demand-paging y hay penuria de memoria. Elija uno de los esquemas justificando su decisión. Haga supuestos razonables.

- **Parte b.-**



La figura anterior muestra qué páginas de un proceso pertenecen al working-set para cada instante en que se calcula el working-set. Una página marcada con una x en el instante t significa que esa página pertenecía al working-set que se calculó en t. Además se indica parcialmente con una círculo el intervalo de tiempo en que algunas páginas pudieron producir un page-fault.

Complete esta figura con todos los intervalos en que una página puede producir un page-fault e indique el número máximo de page-faults que podría contabilizar este proceso.

Pregunta 3

- **Parte a.-**

¿Cuánto espacio se requiere para almacenar una FAT16 de una partición de 128 MB con bloques de 2 KB c/u?

Explique por qué no es posible tener una partición de 512 MB con bloques de 2 KB. Cuál es el mínimo tamaño del bloque en este caso.

- **Parte b.-**

Considere un archivo de 40 MB (en una partición de 128 MB) con todos sus bloques contiguos en el disco. Se ejecuta un programa en C que realiza las siguientes operaciones.

1. Abre el archivo (con `fopen`).
2. Posiciona el próximo byte a leer como aquel que se ubica a 39MB del comienzo del archivo (usando `fseek`).
3. Lee un byte (con `fread`).

Haga una estimación del número de bloques que es necesario leer del disco para ejecutar estas operaciones. No olvide que la FAT se almacena en el disco.

- **Parte c.-**

Repita su estimación considerando que el archivo se encuentra completamente disperso en el disco (la distancia entre un bloque y otro se puede considerar aleatoria). Haga supuestos razonables.

- **Parte d.-**

Suponga ahora que el programa corre en un sistema Unix. Repita su estimación considerando que el archivo está formado por bloques de 1 KB completamente dispersos en el disco.

Concluya comparando la eficiencia de DOS y de Unix para implementar `fseek`.