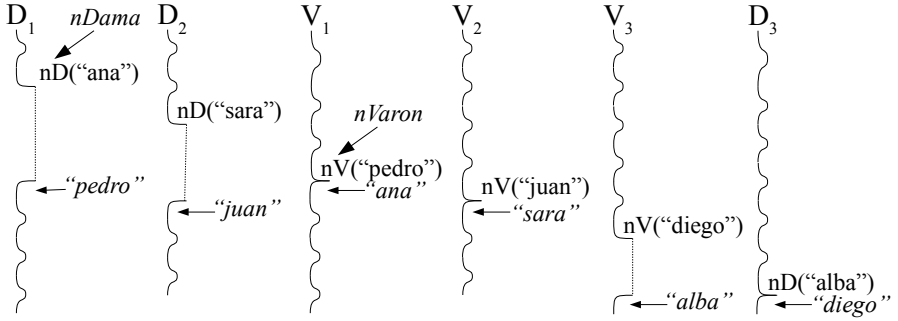
CC4302 Sistemas Operativos – Examen – Semestre Otoño 2017 – Prof.: Luis Mateu

Pregunta 1

I. (3 puntos) Las tareas de nSystem representan damas y varones que buscan una pareja para bailar en una discoteca invocando *nDama* y *nVaron*. La función *nDama* recibe como parámetro el nombre de la dama. Si hay varones esperando pareja, *nDama* entrega de inmediato el nombre del varón que llegó primero. Es decir que la asignación de parejas se hace por orden de llegada. Si no hay varones en espera, *nDama* espera por una invocación de *nVaron*. Análogamente *nVaron* entrega al instante el nombre de la primera dama que todavía busca su pareja, o espera una invocación de *nDama*. El siguiente es un diagrama de tareas que muestra un ejemplo de ejecución.



El diagrama muestra que la pareja de ana (D_1) es pedro (V_1) y que por lo tanto *nDama* en D_1 retorna "pedro" y *nVaron* en V_1 retorna "ana".

Implemente las funciones *nDama* y *nVaron* usando los procedimientos de bajo nivel de nSystem (*START_CRITICAL*, *Resume*, *PutTask*, etc.). Ud. no puede usar otros mecanismos de sincronización ya disponibles en nSystem como semáforos, monitores, mensajes, etc. Declare las variables globales que necesite y especifique los campos que agregará al descriptor de tarea. Los encabezados de las funciones pedidas son:

*char *nDama(char *nombre);*
*char *nVaron(char *nombre);*

II. (1 punto) Suponga que un proceso recorre 100 veces un arreglo de 4 MB secuencialmente en un procesador Intel x86 (con páginas de 4 KB). El arreglo se encuentra completamente residente en la memoria real. Estime el número de desaciertos en la TLB (*translation look-aside buffer*).

III. (1 punto) ¿Cuantos timers se necesitan en una máquina octa-core para implementar el núcleo de un sistema operativo? Explique por qué.

IV. (1 punto) Las variables globales son siempre una fuente de dataraces en los programas multi-thread. Explique cómo se resuelve el problema de los dataraces que podrían producir las variables globales declaradas en un núcleo clásico de Unix para una máquina mono-core. ¿Y en un núcleo moderno para una máquina multi-core?

Pregunta 2

a.- (3 puntos) Considere una máquina multi-core en la que no existe un núcleo de sistema operativo y por lo tanto no hay un scheduler de procesos. La siguiente es una implementación incompleta del mismo problema de la pregunta 1 parte I. En esta pregunta la asignación de parejas se hace en cualquier orden. Programe la función *varon* sin alterar el resto de la implementación. ¡Cuidado! No hay simetría con la función *dama*. No necesitará variables globales adicionales.

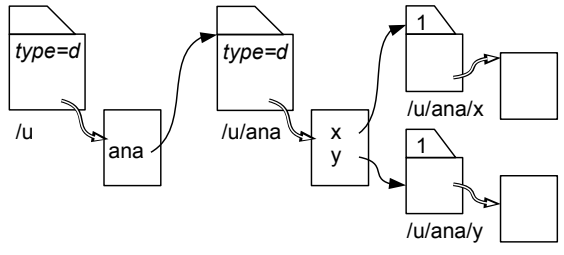
```
int qvarones= OPEN;
int qdamas= CLOSED;
int listo= CLOSED;
char *nom_varon;
char **ppareja_varon;

char *varon(char *nom) {
  …
}
```

```
char *dama(char *nom) {
  spinLock(&qdamas);
  char *pareja_dama= nom_varon;
  *ppareja_varon= nom;
  spinUnlock(&listo);
  return pareja_dama;
}
```

Restricción: La única forma de busy-waiting admitida es usando un spin-lock.

b.- (1 punto) 5 procesos se encuentra en estado de espera porque hicieron peticiones para leer bloques del disco en el siguiente orden: 900, 100, 800, 600, 700. El último bloque leído fue el 300 y el penúltimo el 200. Indique en qué orden se harán las lecturas de estos 5 procesos cuando la estrategia de scheduling de disco es: (i) *first come first served*, (ii) *shortest seek first*, (iii) método del ascensor (o *look*).

c.- (1 punto) La figura muesta a la izquierda varios archivos y directorios de la partición /u. El número que aparece en los inodos de los archivos *x* e *y* es el contador de links duros. A la derecha se muestran los comandos que ejecuta la usuaria *ana*:




```
$ cd /u/ana
$ cp x z
$ mkdir bin
$ ln y bin/w
$ ln -s bin exes
```

Rehaga la figura de la izquierda de acuerdo a los cambios realizados.

d.- (1 punto) Haga un diagrama de inodos, bloques de datos y bloques de indirección para un archivo de 100 KB que se encuentra en una partición con bloques de 512 bytes.