CC4302 Sistemas Operativos – Control 2 – Semestre Primavera 2019 – Prof.: Luis Mateu

Pregunta 1

I) (4,5 puntos) Un pub posee un único baño que debe ser compartido por damas y varones. El baño es amplio y admite un número ilimitado de personas. El problema consiste en evitar que las damas se encuentren con los varones dentro del baño. Las siguientes funciones coordinan los ingresos y salidas del baño:

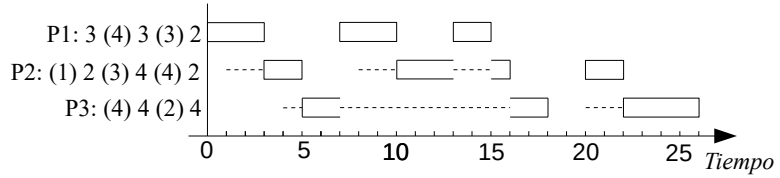
- *void nEntraDama( )*: Una dama solicita ingresar al baño.
- *void nSaleDama( )*: Una dama notifica que sale del baño.
- *void nEntraVaron( )*: Un varón solicita ingresar al baño.
- *void nSaleVaron( )*: Un varón notifica que sale del baño.

Se necesita que el baño sea ocupado alternadamente por damas, luego varones, luego damas, etc. de manera eficiente y sin provocar hambruna. Para lograrlo se pide que cuando un varón invoca *nEntraVaron*, debe esperar si y solo si hay damas dentro del baño o hay damas esperando ingresar. En caso contrario ingresa de inmediato. De igual forma, cuando una dama invoca *nEntraDama*, espera si y solo si hay varones dentro del baño o hay varones esperando ingresar. Si no, ingresa de inmediato. Cuando sale el último varón del baño, todas las damas en espera deben ingresar y cuando sale la última dama del baño, todos los varones en espera deben ingresar. Con estos requerimientos se logra que múltiples personas usen simultáneamente el baño siempre y cuando sean todas del mismo sexo. Además se evita la hambruna.

Programe las 4 funciones de más arriba (*nEntrarDama*, *nSaleDama*, etc.) 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 e indique cómo se inicializan. Use una cola (*Queue*) para hacer esperar a las damas y otra cola para los varones en espera.

Cuidado: las entradas no necesariamente ocurren por orden de llegada. Considere por ejemplo que el baño está siendo ocupado por damas y hay varones en espera. Entonces llega Ana, que debe esperar porque hay varones en espera. Luego llega Pedro que también debe esperar porque hay damas ocupando el baño. A pesar de que Pedro llega después que Ana, va a entrar antes que Ana, cuando todas las damas que ocupaban el baño salgan. Ana solo podrá entrar al baño cuando salga Pedro.

II) (1,5 puntos) El diagrama de abajo muestra las decisiones de scheduling para 3 procesos. A la izquierda se indica para cada proceso las duraciones de sus ráfagas de CPU y entre paréntesis las duraciones de sus estados de espera. En el diagrama la línea punteada indica que el estado del proceso es READY. Si el proceso está en estado de espera el espacio aparece en blanco.



¿De qué estrategia de scheduling se trata? Rehaga el diagrama completo considerando ahora la estrategia *first come first served*.

Pregunta 2

A) (4 puntos) Resuelva el mismo problema del pub de la pregunta 1 considerando ahora una máquina octacore en la que no existe un núcleo de sistema operativo y por lo tanto no hay un scheduler de procesos. Los nombres de las funciones que Ud. debe programar son: *entraDama*, *saleDama*, *entraVaron* y *saleVaron*. Para la sincronización Ud. dispone de spin-locks y la función *coreId()*. Para hacer esperar a las damas use un arreglo de 8 punteros a spin-locks y otro arreglo similar para los varones en espera.

Restricción: La única forma válida de hacer esperar a una persona es utilizando un spin-lock. Otras formas de *busy-waiting* no están permitidas. No hay *fifoqueues*.

B) (1 punto) ¿Cómo se evita en Unix que un proceso del usuario tenga acceso a la cola de procesos "ready", a los descriptores de proceso, al vector de interrupciones, etc.? ¿Pero si no tiene acceso, cómo logra la función *fork* crear un nuevo descriptor de proceso y agregarlo a la cola de procesos "ready"? Explique.

C) (1 punto) ¿Tiene sentido usar spin-locks en una máquina monocore? Explique. Responda entonces cómo implementaría la exclusión mutua en un núcleo moderno de Unix para una máquina monocore.