CC41B : Sistemas Operativos
Control 2–Semestre Primavera'97
Prof.: Luis Mateu.

Pregunta 1

Se desea cambiar los semáforos de nSystem por semáforos que acepten peticiones prioritarias de tickets.

- `nSem nMakeSem(int count)` : Crea un semáforo con prioridades.
- `void nWaitSem(nSem sem, int pri)` : Solicita un ticket al semáforo `s`. La prioridad `pri` puede ser `LOW` o `HIGH`. Si el semáforo tiene tickets disponibles, la llamada extrae un ticket y retorna de inmediato. Si no hay tickets disponibles, la tarea se bloquea en espera de un ticket.
- `void nSignalSem(nSem sem)` : Deposita un ticket en el semáforo. Cuando hay múltiples solicitudes pendientes, se debe atender primero a las que tienen prioridad `HIGH`. La atención de solicitudes que tienen la misma prioridad es en orden FIFO.
- `void nDestroySem(nSem sem)` : Destruye un semáforo.

Defina la representación de estos semáforos e implemente `nWaitSem` y `nSignalSem`. Su implementación *no puede* basarse en otras primitivas de sincronización como mensajes, monitores, etc. Le serán de utilidad los siguientes recursos ya disponibles en nSystem :

- `Queue ready_queue` : La cola de tareas "ready".
- `void Resume()` : Retoma la primera tarea presente en la cola de tareas "ready".
- `void START_CRITICAL()` : Deshabilita las interrupciones.
- `void END_CRITICAL()` : Habilita las interrupciones.
- `void PutTask(Queue queue, nTask task)` : Agrega una tarea al final de una cola.
- `void PushTask(Queue queue, nTask task)` : Agrega una tarea al principio de una cola.
- `nTask GetTask(Queue queue)` : Extrae la primera tarea de una cola.
- `int EmptyQueue(Queue queue)` : Verdadero si la cola está vacía.

Pregunta 2

Se desea agregar a Unix los procedimientos `save_process` y `restore_process` en un computador con arquitectura segmentada. El procedimiento `save_process(filename)` almacena en el archivo `filename` una foto del proceso en el instante que invoca `save_process` (su código, datos y pila). El procedimiento `restore_process(filename)` descarta el proceso que lo invoca y retoma el proceso almacenado en `filename` en el punto en donde éste había invocado `save_process` (análogamente a lo que ocurre con `fork`). El proceso retomado conserva la identificación y los descriptores de archivo del proceso que invocó `restore_process` (como ocurre con `exec`).

a.- Haga un bosquejo de implementación de `save_process`. Explique si es posible implementarlo como un procedimiento de biblioteca o si debe ser una llamada al sistema.

b.- Haga un bosquejo de implementación de `restore_process`. Explique si es posible implementarlo como un procedimiento de biblioteca o si debe ser una llamada al sistema.

c.- Indique la razón fundamental por la cuál no se puede implementar para nSystem procedimientos similares : `save_task` y `restore_task`.

d.- ¿Cuál es el significado de una tarea de nSystem que invoca `save_process`? ¿Qué sucede si un proceso normal (sin nSystem) invoca `restore_process` de un archivo que fue grabado desde nSystem?