Nº Orden	Apellido y nombre	L.U.	Cantidad de hojas

Sistemas Operativos
Recuperatorio del primer parcial – 28 de Junio de 2012

Aclaraciones

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Cada ejercicio se califica con Bien, Regular o Mal. La división de los ejercicios en incisos es meramente orientativa. Los ejercicios se califican globalmente. El parcial se debe aprobar con 2 ejercicios bien y a lo sumo 1 mal.
- El parcial es a libro abierto, con lo cual se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.
- Por favor entregar esta hoja junto al examen.
- **Importante:** Justifique sus respuestas.

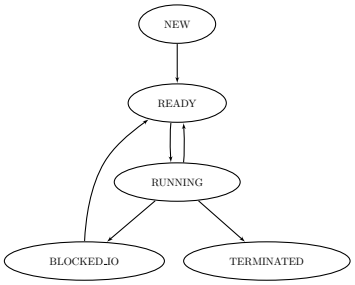
Ejercicio 1) Escriba el código de un programa que, al ejecutarse, cree N procesos hijos, numerados del 0 al N-1. El proceso hijo 0 recibe una cadena de texto por parte del padre. Este debe concatenar a la cadena recibida su número de hijo y pasar la nueva cadena al hijo 1. Esto se repite para todos los procesos, es decir, el hijo i recibe la cadena, le concatena su número y se la pasa al hijo i+1. El hijo N-1 debe pasarle la cadena al proceso padre quien la escribirá por pantalla. Los hijos finalizan luego de que el padre les mande un mensaje con el texto `Terminar!`. El padre finaliza, luego de que todos los hijos terminen. El output del programa debe tener la siguiente forma:

```
"padre; 0; 1; 2; 3; 4; 5; 6; ... ; N-1"
```

Notas:

- Los procesos se mandan mensajes entre sí *exclusivamente* a través de pipes.
- Puede asumir que el tamaño de la cadena es siempre menor a 1024 caracteres.

Ejercicio 2) Se tiene un sistema operativo con el siguiente diagrama de estados de un proceso:



Se quiere implementar soporte para semáforos (contadores):

a) Modifique el diagrama de estados de procesos como crea necesario de modo que el sistema soporte la implementación de semáforos.
b) Explique qué información debe contener la representación interna (a nivel del `KERNEL`) de un semáforo. Qué información es necesario agregar/modificar de los procesos (a nivel de `KERNEL`) y cómo representaría a los semáforos a nivel de usuario. Explique además qué debe suceder con los semáforos que tiene un proceso cuando el mismo ejecuta `fork()`.
c) Describa la interfaz (aridad) de *system calls* que debe proveer el sistema.

d) Escriba el pseudocódigo de la *syscall* que implementa el *signal* sobre un semáforo. Para ello puede asumir que existe una función dentro del `KERNEL` que le permite obtener los procesos (su representación a nivel de `KERNEL`) que se encuentran en un determinado estado. Sea coherente con lo descripto en los puntos anteriores.

Ejercicio 3) Después de implementar los semáforos del ejercicio anterior, los desarrolladores del mismo sistema implementaron también *mutexes* de modo que:

- Un *mutex* puede estar libre o tomado por un proceso y,
- un *mutex* tomado sólo puede ser liberado por el mismo proceso que posee el *lock* sobre el mismo.

Ahora los desarrolladores quieren implementar un mecanismo que permita detectar cuándo el sistema va a entrar en *deadlock* por causa de los *mutexes* y abortar los procesos involucrados:

a) Explique cómo se podría implementar dicho mecanismo indicando qué condición de Coffman es monitoreada por el mismo.
b) Explique por qué dicho mecanismo no serviría para detectar *deadlock* causado por la utilización de semáforos contadores.

Ejercicio 4) En una guardia de un sanatorio los pacientes deben registrarse en Admisión para poder ser atendidos. Los mismos, al llegar, forman una fila y los recepcionistas los van llamando en orden. Cuando un recepcionista atiende a un paciente le solicita su documento y el carnet de la obra social mediante la función `dame_credenciales()` y el peciente se las otorga mediante la función `presentar_credenciales()`. Luego, el recepcionista ingresa los datos en el sistema (`ingresar_datos()`) y le indica al paciente que se dirija a la sala de espera, donde será llamado por algún médico.

Cuando hay pacientes esperando para ser atendidos, los médicos consultan el sistema para saber a qué paciente le corresponde ser atendido primero (aquel cuyos datos hayan sido ingresados primero por los recepcionistas). Cuando es llamado, el paciente ingresa al consultorio del médico. El médico lo revisa (`revisar_paciente()`). Si lo considera necesario, el médico puede mandar al paciente a realizarse algún estudio. En ese caso el paciente se va del consultorio, se realiza el estudio (`realizarse_estudio()`) y se va a su casa.

La cantidad de recepcionistas es `N`. La cantidad de médicos trabajando y de pacientes no está acotada.

Resuelva la sincronización de los procesos `PACIENTE(), MEDICO() y RECEPCIONISTA(i)` mediante el uso de semáforos contadores y *mutexes*. La aridad de las funciones relevantes es la que sigue:

```
tipo_dolencia_t revisar_paciente();

enum tipo_dolencia_t { REQUIRE_ESTUDIO = 1; NO_REQUIERE_ESTUDIO = 0; }
```