1. I FAISO.

La estrategia que expone el enunciado corresponde a la evitación o predicción de interbloqueos. La prevención consiste en eLiminar alguna de las cuatro condiciones que son necesarias para que se de el interbloqueo entre dos o más procesos:

- Exclusión mutua.
- Retención y espera.
- No existencia de expropiación.
- Espera circular.

II FAISO

Para implementar la estrategia de la expropiación de recursos, el sistema operativo debe crear y mantener estructuras de datos que guarden información de los procesos a los que se les ha expropiado el recurso para poder continuar con la ejecución de dichos procesos cuando el planificador lo considere. Ello supone un aumento en la sobrecarga del sistema.

III VERDADERO

Crear o acabar con un hilo dentro de un

proceso requiere menos tiempo que crear o acabar con un proceso, asimismo, un cambio de proceso requiere de más tiempo que un cambio de hilo dentro del mismo proceso; además se ejecuta en modo usuario, reduciendo la sobrecarga del sistema.

IV FALSO

Un sistema operativo multiacceso permite el acceso a un sistema informático a través de dos o más terminales. No requiere necesariamente la existencia de multiprogramación. Ejemplo:

En un sistema de reserva de billetes un único programa atiende a centenares de terminales

2. Para que se produzca interbloqueo entre dos o más procesos es necesario que se cumplan cuatro condiciones:

- Exclusión mutua. Sólo un proceso puede tener una instancia de un recurso. El enunciado nos dice que el sistema permite la exclusión mutua.

- Retención y espera. Ningún proceso liberará el recurso mientras no se le asigne el recurso por el que permanece bloqueado. Igual que el punto anterior, el sistema lo permite.

- Espera circular. Se observa en el gráfico de asignación de recursos que dos procesos se encuentran esperando la liberación del recurso que posee el siguiente en la cadena y el último espera por el primero, con lo que se dá la condición de espera circular.

- No existencia de expropiación. En este sistema sí está permitido la ex-

propiación de recursos, ello significa que el sistema operativo puede prevenir el interbloqueo expropiando los recursos a los procesos susceptibles de bloqueo. Esto es sólo posible si el sistema puede salvar y guardar el estado del proceso antes de la expropiación del recurso para poder restaurarlo cuando vuelva a ser planificado. La creación de estructuras y mantenimiento para salvar el contexto de estos procesos expropiados aumenta la sobrecarga del sistema.

Concluyo que se puede prevenir el interbloqueo porque no cumple la última condición.

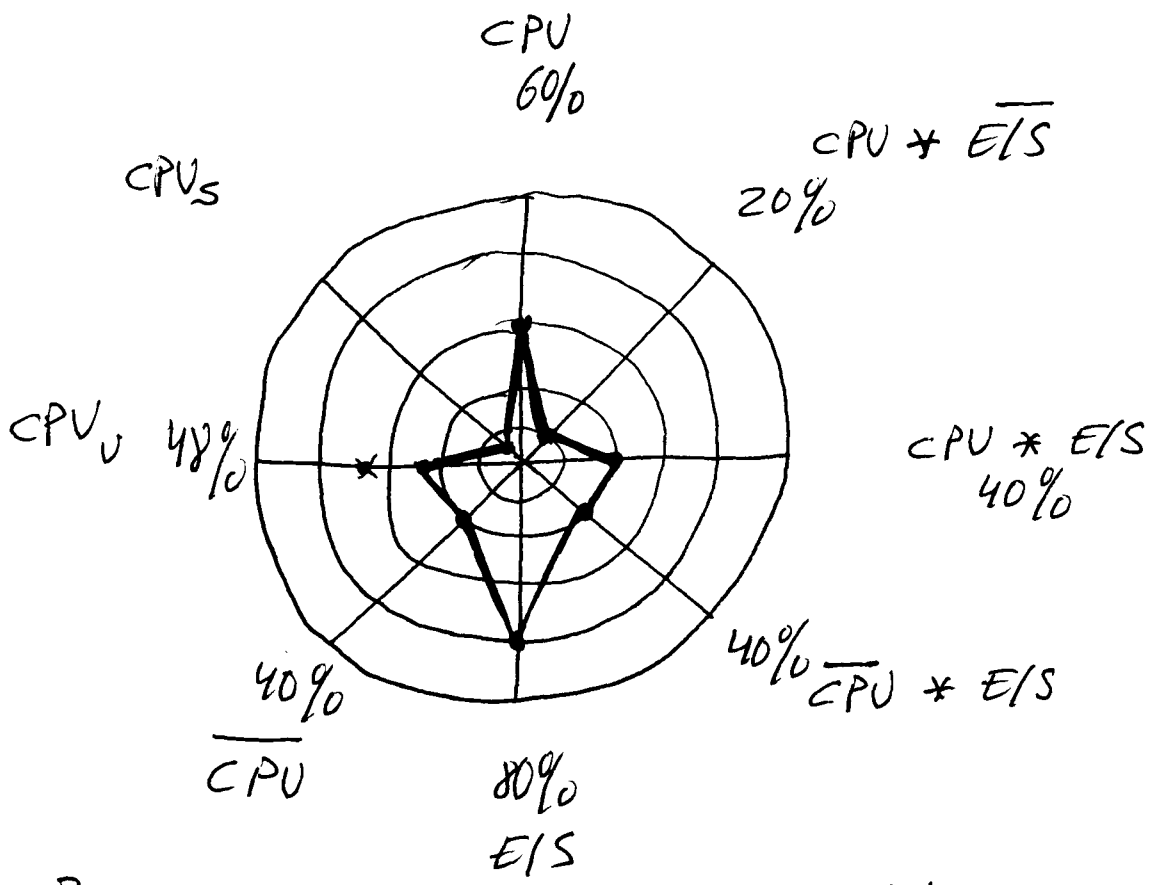
3. De la observación del gráfico inferimos los siguientes índices para la construcción del Diagrama de Kiviatt-Kent. En primer lugar los índices positivos:

- La utilización de la CPU por el sistema ha sido del 60% (CPU 60%).
- La utilización conjunta de la CPU y los dispositivos de E/S es del 40% (CPU * E/S 40%)
- El uso de los dispositivos E/S es de 80% (E/S 80%)
- Y la utilización de la CPU en modo usuario del 48% (CPU, 48%).

Ahora los índices negativos:

- EL uso de la CPU sin solapamiento con los dispositivos E/S es del 20% (CPU * $\overline{E/S}$).
- Asimismo el uso de los dispositivos de E/S sin el solapamiento de la CPU es del 40% ($\overline{CPU}$ * E/S 40%).
- EL tiempo que el sistema no

ha utilizado la CPU es del 40% ($\overline{CPU}$ 40%). Y la utilización de la CPU en modo supervisor es del 12% (CPU_S 12%).



Para mejorar el rendimiento del sintema convendría ~~no~~optimizar el uso simultáneo de la CPU y los dispositivos de E/S; asimismo el uso de la CPU en modo usuario. También mejoraría el rendimiento si aumentáramos la utilización de la CPU, dado que parece que el sistema es más orientado a los dispositivos de E/S.

4. Puesto que se trata de la sincronización de dos procesos concurrentes utilizaré dos semáforos binarios, uno por cada proceso, e inicarlo a cero.

Supongamos que se planifica el procesoA, ejecuta tarea1() y llega al punto de encuentro: ejecuta signal_sem(s2) para avisar al procesoB del evento y desbloquearlo (ahora s2=1). Si el planificador continua con el procesoA lo bloquea al ejecutar wait_sem(s1) y espera hasta que el procesoB llegue al punto de encuentro y lo desbloquee ejecutando signal_sem(s1). No podemos predecir qué proceso terminará primero y ello es debido a varios factores: La imposibilidad de predecir la velocidad de ejecución de cada proceso, el criterio del planificador escogido o de la gestión por parte del sistema operativo para las interrupciones.

```
/* Pseudocódigo programa coordinación */
/* Definición de semáforos binarios */
semáforos_binarios S1,S2;
void procesoA() ;          /* Proceso A */
{
    tarea1();
    signal_sem(S2);        /* Avisa que ha llegado
                              al punto de encuentro */
    wait_sem (S1);         /* Espera que el procesoB
                              llegue al punto de encuen-
                              tro */
    tarea2();
}
void procesoB();           /* Proceso B */
{
    tarea3();
    signal_sem(S1);        /* Avisa al Proceso A que
                              ha llegado al punto de
                              encuentro */
    wait_sem(S2);          /* Espera que el proceso A
                              llegue al punto de
                              encuentro */
    tarea4();
}
```

```
main ()
{
  init-sem (s1, 0);      /* Inicialización de
  init-sem (s2, 0);         los procesos */

  ejecución_concurrente (procesoA, procesoB);
}
```