Examen_Tema3_y_4.pdf

LosCocos

Sistemas Concurrentes y Distribuidos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Test – Sistemas Concurrentes –Grupo B **11-01-2019**

Apellidos: **Nombre:** **Grupo Prácticas:**

Cuestiones y respuestas alternativas (hasta 6.0 puntos)

1. Preguntas de respuesta alternativa (Hasta 3,0 puntos)

1.Seleccionar la única respuesta correcta respecto de la orden `MPI_Receive`:

a. El proceso que programe dicha orden siempre se bloquea independientemente del estado del búfer (vacío o con datos esperando)
b. Sólo si suponemos un mecanismo de comunicación síncrono sin búfer (“citas”) un proceso que programe dicha orden siempre se bloqueará
c. Si la ejecución de la orden anterior no bloquea al proceso, entonces no podemos asegurar que la operación de transmisión de los datos sea segura
d. ✓El proceso que programe dicha orden se bloqueará sólo si el búfer es encuentra vacío, e s decir, no hay mensajes pendientes de ser recibidos

2.Seleccionar la única respuesta correcta en el caso de que un proceso programe la operación insegura `MPI_Irecv(…)`:

a. La reducción del tiempo de recepción del mensaje en el receptor es independiente de que el sistema de ejecución cuente con hardware especializado o no
b. ✓Los datos ya transmitidos siempre se mantienen el búfer del sist hasta que el proceso receptor pueda descargarlos a su espacio de memoria y esto es independiente de que el proceso que ejecute la orden vuelva inmediatamente (sistema con hardware especializado)
c. Con esta operación no se iniciará la transmisión de datos entre el proceso emisor y el receptor inmediatamente
d. Con esta operación siempre se anulará el tiempo de espera en el proceso receptor

3. La elección y ejecución inmediata de 1 alternativa de la orden de espera selectiva (*select*) sólo se producirá si se cumple una de las condiciones siguientes (indicar cuál):

a. Sólo depende de que exista alguna orden *potencialmente ejecutable* en ese momento
b. Sólo depende de que alguna condición de las órdenes con guarda sea cierta
c. ✓Existe, al menos, una orden *potencialmente ejecutable* y además se nombra a un proceso del programa que ya ha iniciado su envío
d. Sólo de que exista algún proceso del programa que haya iniciado su envío

4.Indicar cuál de las siguientes afirmaciones puede considerarse correcta respecto de la orden de espera selectiva:

a. ✓Una vez que se ha seleccionado una orden *potencialmente ejecutable*, el proceso que la contiene se ejecuta secuencialmente hasta que termina el bloque componente (*do…end*)
b. Una orden de espera selectiva no termina hasta que no ejecute cada una de sus alternativas, lo cual ocurre no determinísticamente
c. Una orden de espera selectiva puede entremezclar la ecuencias de ejecución de las instrucciones de los bloques de sentencias componentes de cada una de sus alternativas
d. La orden de espera selectiva puede programar un array de guardas indexadas, pero cada una ha de incluir una operación de entrada (*receive(…)*)

5. Indicar cuál de las siguientes afirmaciones es correcta con respecto a la denominada *deriva acumulativa* que experimentan los programas de tiempo real:

a. La deriva acumulativa se produce en una tarea porque no es posible programar un retraso exacto hasta un instante en que produzca la siguiente activación, es decir, todos los retrasos que se programan indican un intervalo de suspensión relativo al instante en que se ejecutan

b. ✓Se puede eliminar la deriva de los retrasos relativos programados en cada tarea pero nunca se podrán eliminar completamente los retrasos causados por el sistema operativo
c. La orden `sleep_until(…)` permite eliminar completamente la deriva acumulativa
d. Siempre es posible programar una tarea periódica que se active transcurrido un periodo de tiempo exacto desde su última activación en todas sus ejecuciones

6. Respecto del esquema de planificación dinámica de tareas EDF, indicar cuál de las siguientes afirmaciones es la correcta

a. La aplicación de este método es incompatible con aplicar el RMS
b. ✓Un conjunto de tareas cuya utilización del procesador es del 100% y que resulta ser planificable con el algoritmo EDF podría ser también planificable con RMS
c. EDF es el único esquema de planificación dinámico
d. Siempre podremos planificar con EDF cualquier conjunto de tareas cuya utilización del procesador no sea mayor del 100%, independiente de valor de su plazo de tiempo límite

Cuestión 1: Explicar cómo se consigue (qué función de consulta añadirías) para programar un intercambio de mensajes que evite el interbloqueo asociado al paso de mensajes síncrono; utilizar el siguiente código para la explicación, explica también por qué que hay que esperar a que terminen en el orden adecuado las operaciones MPI_Irecv(…) y MPI_Isend(…): .

```
//Declarar variables

MPI_Status                          //declarar variables
MPI_Request                         //declarar variables



MPI_Comm_rank(MPI_COMM_WORLD, &id_propio);
If (id_propio%2==0) id_vecino= id_propio+1 % num_procesos;
  else id_vecino=id_propio-1 % num_procesos;

//Completar
MPI_Irecv(
//Completar
MPI_Isend(
//Falta código : utilizar órdenes MPI_Wait(…) para resolverlo
```

Cuestion 2: Explicar el significado de la denominada condición suficiente de planificabilidad de un conjunto de N tareas periódicas con prioridades asignadas estáticamente según RMS; para explicarlo utilizar el siguiente conjunto de tareas:

	Ci	Ti
$\tau 1$	3	5
$\tau 2$	1	8
$\tau 3$	2	10

Dado el límite de utilización del procesador para 3 tareas U0(3)= 78%. Construir, si fuera necesario, el diagrama de Gantt. Calcular los tiempos de respuesta de cada tarea.

Ejercicios (hasta 4.0 puntos)

1. **"N" procesos envían de forma segura 1 único mensaje, que contiene un entero con el identificador del proceso emisor, al proceso receptor. El receptor ha de imprimir el número del proceso que inició el envío; a continuación, terminará dicho proceso emisor y el resto de emisores se quedarán bloqueados. Resolver, si se puede, la comunicación anterior utilizando en el receptor: (a) orden `MPI_Receive(…)`, (b) `MPI_Ireceive(…)`, (c) orden "select". En los casos en que no se pueda explicar por qué.**

2. **Considerar un programa en que cada uno de los procesos envían al resto de procesos y que cada ítem de datos a producir y transmitir es un bloque de bytes con muchos valores. Suponemos las funciones: `ConsumirBloque(b)`: que sirve para consumir los datos recibidos en lugar de imprimirlos y `ProducirBloque(…)` que produce y escribe una secuencia de bytes en el bloque[i].**

   En cada ordenador existe la suficiente memoria para contener al menos "N" bloques. Para evitar un colapso de memoria del sistema de paso de mensajes, que se produciría si se dan (N-1)^2 mensajes en tránsito, se pueden utilizar *operaciones de envío inseguras* del tipo `MPI_Isend(…)` y se ha de utilizar una operación "`select`" para conseguir que el orden en que se consumen los bloques coincida con el orden en que se inician los envíos.

7. Respecto de las operaciones de paso de mensajes no-bloqueantes:
   (a) Siempre (incluso son soporte hardware) es necesario programar operaciones de comprobación que indiquen si es seguro acceder a los datos en transmisión antes de que la ejecución de la operación `receive()` devuelva el control al proceso receptor
   (b) Si el proceso receptor está preparado para recibir los datos en transmisión, la ejecución de la operación `receive()` *vuelve* inmediatamente siempre
   (c) El proceso emisor siempre supone que la ejecución de la operación `send()` accederá a datos en un estado inseguro
   (d) ✓ Existe un caso en el cual la ejecución de la operación `receive()` no detiene al proceso receptor aunque no se hayan terminado de transmitir los datos que han de recibirse