dit
UPM
Universidad Politécnica de Madrid
E.T.S.I. Telecomunicación
Departamento de Ingeniería de Sistemas Telemáticos

EXAMEN FINAL ORDINARIO DE SISTEMAS OPERATIVOS
Convocatoria de Septiembre. 15–9–97
PRIMERA PARTE (SIN LIBROS)

Puntuación y longitud *máxima* —puede de hecho ser más corta— de la respuesta, en cada pregunta. **Una página es una cara**.

DURACIÓN: 1 hora

Pregunta 1

Al contrario de lo que ocurre en los sistemas centralizados, en un sistema distribuido puede suceder que un proceso realice una petición a un núcleo que está en un nodo diferente de aquél en el que se encuentra el proceso. ¿Qué relación existe entre este hecho y el mecanismo de *capability*? Explicar.

(*1/2 página; 1 punto*)

Cuando en un sistema centralizado un proceso hace una petición al núcleo, éste tiene todos los medios para investigar de qué proceso se trata, por cuenta de qué usuario actúa, con qué privilegios cuenta, etc. Y ello es así porque en ese momento el núcleo tiene el control absoluto de la máquina y, en especial, de las estructuras de datos que definen completamente el descriptor del proceso dado. En resumen, el núcleo puede realizar con facilidad un control de acceso exhaustivo.

Lo anterior no es el caso, sin embargo, cuando en un sistema distribuido un proceso realiza (a través de uno o varios servidores) una petición a un núcleo que se encuentra en un nodo diferente que él mismo. Ahora el núcleo no lo tiene tan fácil como antes para investigar los privilegios del proceso cliente, con vistas a realizar el control de acceso. En vez de tener que realizar una compleja operación de autenticación, sin embargo, el mecanismo de capability *permite realizar el mismo control de acceso de una manera eficiente y flexible. Eficiente porque los derechos de acceso van cifrados dentro de la misma* capability*, sin necesidad de tener que deducirlos por otros medios, y flexible porque los procesos pueden transferirse derechos de acceso entre sí sin más que intercambiarse las* capabilities *correspondientes.*

Pregunta 2

¿Qué son las hebras (*threads*)? ¿Cuál es su razón principal de ser?

(*1/2 página; 1 punto*)

Las hebras representan un segundo nivel de concurrencia, por encima de los procesos. Pueden entenderse como un segundo nivel de multiplexación, realizado ahora dentro de cada uno de los procesos, y no de la máquina completa. Ahora bien, mientras que en la concurrencia de procesos se multiplexa la UCP (en tiempo) y la memoria central (en espacio, un trozo o conjunto de trozos por cada proceso), en la concurrencia de hebras dentro de un mismo proceso solo se multiplexa la UCP, mientras que todas las hebras del mismo proceso comparten el mismo espacio de memoria central.

La razón principal de ser de las hebras es la eficiencia de su modelo de concurrencia, es decir, el hecho de que tanto la creación y destrucción de una hebra como, sobre todo, la multiplexación entre hebras pertenecientes al mismo proceso, son mucho más rápidas de realizar (un orden de magnitud en la escala decimal) que sus correpondientes con procesos. Ello es debido a que en la multiplexación de hebras no se requiere cambio de espacio de direcciones, al contrario de lo que ocurre en la multiplexación de procesos.

Pregunta 3

Explique los términos *resolución* y *resolución inversa*, en el contexto de los servicios de nombres. ¿Cuál se usa más? En el caso de DNS, ¿cuál es más costosa? Explíquese.

(*1/3 página; 1 punto*)

> Resolver *es obtener un nombre de bajo nivel (numérico) o un atributo a partir de un nombre de alto nivel (simbólico).* Resolución inversa *es obtener el nombre de alto nivel a partir del de bajo nivel. Se usa mucho más la resolución ordinaria, ya que personas y programas usan nombres de alto nivel y las máquinas nombres de bajo nivel. En el caso de DNS la resolución inversa es igual de costosa, ya que usa un mecanismo similar, considerando los cuatro octetos de una dirección IP, en orden inverso en el que se escriben, como subdominios de* in-addr.arpa.

Pregunta 4

Diga si el sistema de ficheros CODA es apropiado para contener grandes bases de datos con registros muy pequeños en los dos casos siguientes:

1. Las bases de datos se están actualizando continuamente.
2. Las bases de datos se actualizan raramente.

Explique y matice sus respuestas (por ejemplo, diga qué pueden significar los términos grande, raramente, continuamente). Sugiera un sistema de ficheros alternativo que no tenga los problemas observados.

(*1/3 página; 1 punto*)

> *CODA copia enteros los ficheros del servidor al cliente en lectura y los copia de nuevo al servidor si se han modificado. Eso significa al menos que la base de datos debe caber en el disco del cliente, para ser utilizable. Si la base de datos se actualiza raramente, los clientes sólo tendrán que transferírsela completa una vez entre actualizaciones, por lo que puede ser perfectamente utilizable. Si las actualizaciones son frecuentes y vienen de otros clientes, una gran parte de las lecturas van a requerir una transferencia completa, que además retrasará las operaciones en todos los demás clientes, ya que debe ponerse un cerrojo durante el acceso. Por supuesto que la base de datos en ningún caso debería usarse en modo desconectado, ya que los cerrojos no pueden ponerse. Los problemas citados debidos a los ficheros grandes podrían soslayarse si la base de datos se divide en muchos ficheros pequeños e independientes lógicamente. Con NFS, por ejemplo, no habría esos problemas, ya que sólo se transfieren pequeñas porciones de los ficheros a los clientes.*

Pregunta 5

Diga en qué casos la replicación mejora el rendimiento y en qué casos lo empeora. ¿Qué relación hay entre coherencia y rendimiento?

(*1/3 página; 1 punto*)

> *Depende de la aplicación y de los requisitos de coherencia: cuando más coherencia se requiere, el rendimiento es menor. Generalmente la replicación mejora el rendimiento de las lecturas, repartiéndose la carga y acudiéndose siempre a la copia más cercana o accesible. No así las escrituras, generalmente menos frecuentes, ya que en ellas se suele desear una actualización consistente de todas las copias. No obstante, si no importa que las lecturas obtengan información obsoleta, las actualizaciones perezosas pueden llegar a ser también eficientes.*

dit
UPM
Universidad Politécnica de Madrid
E.T.S.I. Telecomunicación
Departamento de Ingeniería de Sistemas Telemáticos

EXAMEN FINAL ORDINARIO DE SISTEMAS OPERATIVOS
Convocatoria de Septiembre. 15–9–97
SEGUNDA PARTE (CON LIBROS)

Puntuación y longitud *máxima* —puede de hecho ser más corta— de la respuesta, en cada pregunta. **Una página es una cara**.

DURACIÓN: 1 hora y cuarto

Pregunta 1

Considérese un sistema de transmisión de mensajes punto a punto (*unicast*) en el que ningún proceso envía un nuevo mensaje hasta que ha recibido la confirmación de recepción del (inmediatamente) anterior que haya enviado. Además, la confirmación de todo mensaje se realiza automáticamente apenas se recibe.

Se pide demostrar que esta regla es suficiente para garantizar entrega causal (ésta es, por ejemplo, la forma como funcionan las RPCs).

(1/2 página; 1 punto)

La entrega será no causal si sucede que, bien:

1. *(No FIFO) Existen mensajes $m^1_{i,j}$ y $m^2_{i,j}$ del proceso p_i al proceso p_j tales que el segundo se envía por p_1 después que el primero y, sin embargo, se recibe antes por p_2; pero esto es imposible con la regla de confirmación dada, ya que con ella el envío del segundo mensaje es posterior a la (confirmación de la) recepción del primero, y en cualquier caso la recepción de un mensaje es siempre posterior a su envío.*

   O bien,

2. *Existen tres mensajes, $m_{i,k}$, $m_{i,j}$ y $m_{j,k}$, enviados entre los procesos indicados por los subíndices y en el orden dado y, sin embargo, el mensaje $m_{i,k}$ se recibe en el proceso p_k después que el mensaje $m_{j,k}$; pero de nuevo esto es imposible con la regla de confirmación dada, ya que $m_{i,j}$ sólo ha podido ser enviado después de la (confirmación de la) recepción de $m_{i,k}$, y sólo aún más tarde se habrá enviado $m_{j,k}$, que, de nuevo, solo podrá haber sido recibido después de haber sido enviado.*

Q.E.D.

Pregunta 2

Dos sucesos e_1 y e_2 tienen horas vectoriales de $[2, 1, 5]$ y $[6, 3, 5]$, respectivamente. No se sabe a que procesos pertenecen esos dos sucesos.

1. ¿Qué relación causal existe entre e_1 y e_2?. Explicar.
2. ¿Se puede afirmar sin que exista ningún género de duda que los dos sucesos son *pairwise consistent*? (que pueden formar parte conjuntamente de la frontera de un corte coherente). Explicar.

Suponiendo ahora que las horas vectoriales de los dos sucesos son, respectivamente, $[2, 1, 5]$ y $[6, 3, 4]$, contestar de nuevo a las dos preguntas anteriores.

(1 página; 2 puntos)

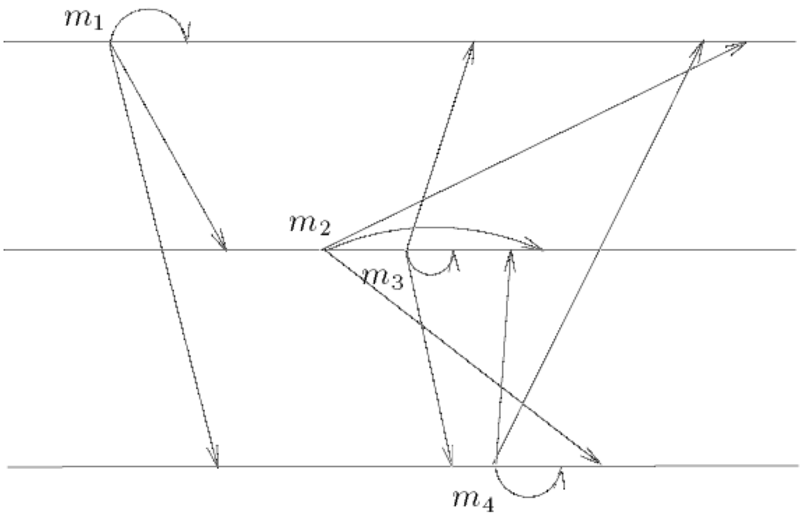
1. (a) *e_1 precede causalmente a e_2 ya que todos los componentes de su hora matricial son menores o iguales que los correspondientes de e_2, sin que las horas sean idénticas.*
   (b) *No se puede afirmar, no, pues si, por ejemplo, e_1 pertenece al proceso de orden uno, e_2 "sabría" en cualquier caso más del proceso uno que el mismo e_1.*
2. (a) *En este caso e_1 y e_2 son sucesos concurrentes, ya que ninguna de las dos horas matriciales preceda a la otra.*

(b) *Tampoco se puede afirmar, no, pues si, por ejemplo,* e_1 *sigue perteneciendo al proceso de orden uno,* e_2 *seguiría "sabiendo" en cualquier caso más del proceso uno que el mismo* e_1.

Continúa en la página siguiente ...

Pregunta 3

Considere la ejecución siguiente que muestra el multienvío de cuatro mensajes entre tres procesos y donde se marcan los instantes de transmisión y entrega:



1. ¿Satisface la condición de orden total en la entrega? Explicar.
2. ¿Satisface la condición de orden FIFO en la entrega? Explicar.
3. ¿Satisface la condición de causalidad en la entrega? Explicar.

(*1/2 página; 1 punto*)

Satisface el orden total, ya que los mensajes se entregan en todos los procesos en el mismo orden. Sin embargo, la entrega no es FIFO, ya que la entrega de m_3 adelanta a la de m_2 en todos los procesos, habiendo salido antes del mismo proceso. Finalmente, tampoco es causal, simplemente por no ser FIFO.

Pregunta 4

Un servidor NTP **B** recibe del servidor **A** un mensaje a sus 16:34:23.480 con una marca de tiempo de 16:34:13.430, y responde. **A** recibe la respuesta a sus 16:34:15.725 con marca de salida de **B** a las 16:34:25.007. Estime la diferencia entre los relojes de ambas máquinas y acote el error de esa estimación.

(*1/2 página; 1 punto*)

Usando la terminología del libro de Colouris, tenemos:

$a = T_{i-2} - T_{i-3} = 16:34:23.480 - 16:34:13.430 = 10,050$ *segundos*
$b = T_{i-1} - T_i = 16:34:25.007 - 16:34:15.725 = 9,282$ *segundos*

*lo que da una diferencia estimada (**B** está adelantado) de*

$o = \frac{a+b}{2} = 9,666$ *segundos*

y el error máximo de esta estimación es la mitad del retardo de ida y vuelta, es decir:

$\epsilon = \frac{|a-b|}{2} = 0,384$ *segundos*