Examen Extraordinario de Sistemas Operativos Junio de 2014

Grado en Ingeniería Informática

Solución

- La fecha de publicación de las notas, así como de revisión se notificarán por Aula Global.
  * Duración del examen:
    -- 2:30 horas
  * No se pueden utilizar libros ni apuntes
  * Será necesario presentar el DNI o carnet universitario para realizar la entrega del examen

Ejercicio 1 (2,5 puntos). Autotest. 30 minutos.

Responda a las preguntas del autotest en el cuadro adjunto indicando la letra de la respuesta válida. Recuerde que por cada 3 fallos se quita una respuesta válida. No contestadas no penalizan.

NOMBRE: GRUPO:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
B	A	A	A	D	C	C	C	A	D	A	D	A	C	D	D	A	B	D	A

PREGUNTAS

1- La asignación de bloques de archivos mediante una lista enlazada
A) Evita la fragmentación del disco
B) Genera mucha fragmentación de disco
C) Es un método muy rápido
D) Consigue que la cantidad de espacio para los datos en un bloque sea potencia de dos

2 - ¿Cuál de las siguientes sentencias NO es correcta respecto a un semáforo?
A) Protege a un fichero de accesos de usuarios no autorizados por el propietario.
B) Proporciona a un proceso exclusión mutua en el acceso a variables compartidas con otros procesos
C) Permite a un proceso señaalar un evento a otro proceso.
D) Gestiona el uso de un conjunto N de recursos por parte de un conjunto de procesos.

3 - Considere la llamada al sistema execv(A/dir/bin/com@, argv); e indique cuál de las razones expuestas puede hacer que falle.
A) No tener permiso de ejecución en el directorio A/dir@.
B) No tener permiso de lectura en el directorio A/dir/bin@.
C) No estar situado en el directorio A/dir/bin@.
D) No tener el directorio A/dir/bin@ en el PATH.

4 - En sistemas sin memoria virtual pero con multiprogramación con particiones variables y con relación al swapping, ¿qué es falso?
A) Es necesario preasignar a todo proceso una zona de swap.
B) Un proceso ready (listo para ejecutar) puede estar en la zona de swap.

C) Un proceso activo o está totalmente en memorial principal o lo está en la zona de swap.
D) Hay en memoria principal aproximadamente la mitad de huecos que de procesos (regla del 50%).

5 - En una maquina UNIX con 16 Mb de memoria principal, 1 Gb de disco de usuarios y 32 Mb de área de paginación, sea el fichero f de 120 Mb. ¿Es posible ejecutar el siguiente comando?
cat f | grep palabra | wc
A) No, memoria principal insuficiente.
B) No, memoria virtual insuficiente.
C) No, tamaño del pipe insuficiente.
D) Si, sin problemas.

6 - ¿Cómo se hace en UNIX para que un proceso cree otro proceso que ejecute otro programa?
A) Con un CREATE
B) Con un EXEC
C) Con un FORK seguido de un EXEC
D) Con un EXEC seguido de un FORK

7 - Respecto a un sistema operativo sin memoria virtual que usa la técnica del intercambio (swapping), ¿cuál de las siguientes sentencias es correcta?
A) El uso del intercambio facilita que se puedan ejecutar procesos cuyo tamaño sea mayor que la cantidad de memoria física disponible.
B) El intercambio aumenta el grado de multiprogramación en el sistema.
C) En sistemas de tiempo compartido, el intercambio permite tener unos tiempos de respuesta similares para todos los usuarios, con independencia de su número.
D) El uso del intercambio aumenta la tasa de utilización del procesador.

8 - ¿Qué es falso acerca del swapping o intercambio?
A) Permite cambiar el grado de multiprogramación.
B) El área de swap se suele implementar en disco.
C) Se utiliza en conjunción con la paginación.
D) Se le llama también planificador a medio plazo.

9 - Un usuario con uid=12 y gid=1 es el dueño de un fichero con modo de protección rwxr-x---. Otro usuario con uid=3 y gid=1 quiere ejecutar el fichero. ¿Puede hacerlo?
A) Siempre.
B) Nunca.
C) Sólo si se le pone el bit setuid.
D) Sólo si se le pone el bit getuid.

10 - El fichero pepe tiene los permisos rwxrwxrwx. ¿Qué mandato debería usarse para que el fichero sólo pueda ser leído y ejecutado por el propietario y los miembros de su grupo?
A) chmod 766 pepe
B) chmod +rx pepe
C) chgrp rx pepe
D) chmod 550 pepe

11 - La activación del sistema operativo:.
A) Por parte de un proceso del usuario se lleva a cabo a través de la instrucción "TRAP".
B) Cuando se ejecuta la interrupción software (TRAP), el hardware se para hasta que se active el planificador.

C) La activación del sistema operativo puede llevarse a cabo por llamadas al sistema.
D) Las interrupciones de dispositivos no activan el sistema operativo.

12 - Sea un programa concurrente con tres procesos iguales cuyo código consiste tan solo en incrementar en uno una variable V compartida entre ellos. ¿Cuál de las siguientes opciones es correcta respecto al posible valor resultante de V después de la ejecución concurrente de los tres procesos si V vale 0 inicialmente?
A) V tiene valor 1,5.
B) V tiene valor 4.
C) V tiene valor 0.
D) V tiene valor 2

13 - En el método de asignación contigua del espacio de disco:
A) Sólo es necesario la dirección del primer bloque y la longitud del archivo.
B) Unos pocos bytes del comienzo de los bloques se usan como puntero al siguiente bloque, el resto contiene datos.
C) Es necesario colocar los índices a los bloques de los archivos en una tabla de índices.
D) No existe tal método de asignación.

14 - El sistema operativo UNIX tiene una estructura:
A) De capas
B) De máquinas virtuales
C) Monolítica
D) Cliente-Servidor

15 - ¿Cuándo se produce hiperpaginación (thrashing)?
A) Cuando hay espera activa.
B) Cuando se envía un carácter a la impresora antes de que se realice un retorno de carro.
C) Cuando no hay espacio en la TLB.
D) Cuando los procesos no tienen en memoria principal su conjunto de trabajo.

16 - Para poder leer o escribir en un archivo, éste...
A) No tiene necesariamente que estar abierto
B) No puede estar abierto
C) Debe estar vacío
D) Debe estar abierto

17 - ¿Qué es falso con respecto a los semáforos?
A) El uso de una política FIFO de encolamiento en el semáforo evita el interbloqueo al usar semáforos.
B) Pueden usarse para sincronizar procesos ligeros.
C) Se pueden utiliza en sistemas multiprocesadores con memoria compartida.
D) Permiten resolver el problema de la sección crítica.

18 - ¿En qué situaciones se puede producir un cambio de contexto en un sistema con un planificador basado en el algoritmo SJF (Shortest Job First) además de cuando se bloquea o se termina el proceso?
A) Cuando se pone en estado de listo un proceso con mayor prioridad.
B) En ninguno más

C) Cuando se pone en estado de listo un proceso cuya próxima racha tiene una duración prevista menor que la del proceso que est· ejecutando.

D) Cuando se termina el cuanto del proceso.

19 - Un thread (hilo en ejecución) es:

A) Una entidad planificable con recursos independientes(registros, memoria, ficheros).

B) Una función dentro de un programa.

C) Un puntero a una función.

D) Una secuencia de código planificable de forma independiente dentro de un proceso.

20 - En un sistema con memoria virtual ¿cuál no es una función del sistema operativo?

A) Traducir las direcciones virtuales a físicas.

B) Asignar memoria física.

C) Gestionar el que los procesos compartan memoria.

D) Asignar memoria virtual.

Grado en Ingeniería Informática

Ejercicio 2 (2,5 puntos).

En un sistema de tiempo real, en el que los procesos se ejecutan en función de su prioridad, van llegando al sistema los procesos que se especifican a continuación:

PROCESO	T. DE LLEGADA	T. DE EJECUCIÓN	PRIORIDAD
P1	0	0,5	2
P2	0,2	0,3	3
P3	0,3	0,4	1
P4	0,5	1	3
P5	0,7	0,6	2

Nota: Cuando las prioridades entre procesos sean iguales, se utiliza el algoritmo FCFS. A menor valor, mayor prioridad

Se pide: realizar el cronograma de ejecución de los procesos, así como calcular los tiempos de retorno, espera y retorno normalizado de cada proceso, para los casos de planificación NO apropriativa y apropiativa. (indicar las soluciones en las tablas que se adjuntan)

SOLUCION

a) Las planificación es no apropiativa.

0 0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 1,0 1,1 1,2 1,3 1,4 1,5 1,6 1,7 1,8 ----- 2,8

P1

P2

P3

P4

P5

PROCESO	T. RETORNO	T. ESPERA	T. RETORNO NORMALIZADO
P1	0,5	0	1
P2	1,6	1,3	1,6/0,3
P3	0,6	0,2	0,6/0,4
P4	2,3	1,3	2,3
P5	0,8	0,2	0.8/0,6

b) La planificación es apropiativa.

0 0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9 1,0 1,1 1,2 1,3 1,4 1,5 1,6 1,7 1, 8 ----- 2,8

P1

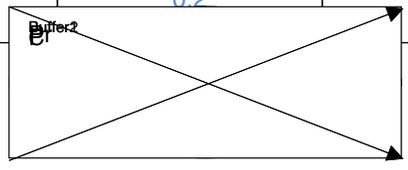
P2

P3

P4

P5

PROCESO	T. RETORNO	T. ESPERA	T. RETORNO NORMALIZADO
P1	0,9	0,4	0,9/0,5
P2	1,6	1,3	1,6/0,3
P3	0,4	0	0,4/0,4
P4	2,3	1,3	2,3/1
P5	0,8	0,2	0,8/0,6



c) Describa brevemente que problema puede surgir al utilizar un algoritmo de planificación basado en asignación por prioridades fijas y que mecanismo se podría utilizar para evitar ese problema.

La asignación de prioridades fijas puede ocasionar que la continua llegada de procesos con alta prioridad ocasione que determinados procesos con baja prioridad no se terminen de ejecutar nunca, fenómeno conocido como inanición.

La forma de solucionar este problema consiste en utilizar mecanismos de envejecimiento los cuales elevan la prioridad de los procesos a medida que aumentan sus tiempos de espera.

Ejercicio 3 (2,5 puntos).

Se quiere construir con threads un "pipeline" compuesto por tres fases: productor, procesador y consumidor. Para ello, se recurre al uso de dos buffers. Se pide:

a) Implementar la sincronización entre los tres threads mediante variables de condición. Declare las variables y coloque en el lugar adecuado las instrucciones y las llamadas a funciones que crea necesarias añadir al código de partida que se proporciona
b) Indique en qué parte del código se corre más riesgo de provocar una condición de DeadLock y por qué cree que la solución propuesta lo evita.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
```

```
#define MAX_BUFFER 250

int buffer1[MAX_BUFFER];
int buffer2[MAX_BUFFER];

void *productor(void *arg) {
	int * pdatos = (int*) arg;  // Puntero al parámetro
	 int datos_a_producir = *pdatos; // Número de datos a producir
	int pos = 0;

	for (int i = 0; i < datos_a_producir; i++) {

		buffer1[pos] = i;

	}
}

void *procesador(void *arg) {
	int pos1=0, pos2=0, dato_recibido, dato_procesado;
	int datos_a_procesar = *(int *)arg;

	for (int i = 0; i < datos_a_procesar; i++) {

		dato_procesado = funcion_procesamiento(dato_recibido);

	}
}

void *consumidor(void *arg) {
	int datos_a_consumir = *(int *)arg;
	int pos = 0;

	for (int i = 0; i < datos_a_consumir; i++) {

		printf("< Consumiendo: %d\n",buffer2[pos]);

	}
}

int main(int argc, char *argv[]) {

	pthread_t th1, th2, th3;  int num_items = 1000;

	pthread_create(&th1, NULL, productor, (void *) &num_items);
	pthread_create(&th2, NULL, procesador, (void *) &num_items);
	pthread_create(&th3, NULL, consumidor, (void *) &num_items);

	pthread_join(th1, NULL); pthread_join(th2, NULL);
	pthread_join(th3, NULL);

	printf("Fin del thread principal\n");
	return 0; }
```

Solución Ejercicio

Apartado a)

Una posible solución puede ser la que se proporciona a continuación.

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <pthread.h>

#define MAX_BUFFER 250

int funcion_procesamiento(int);

int buffer1[MAX_BUFFER]; // El primer buffer se usa para pasar datos del
productor al procesador
int buffer2[MAX_BUFFER]; // El segundo buffer se usa para pasar los datos del
procesador al consumidor


int cont1 = 0, cont2 = 0;  // Variables compartidas. Contadores de items
actualmente en cada buffer.

pthread_cond_t no_lleno1;
pthread_cond_t no_vacio1;
pthread_mutex_t mutex1;

pthread_cond_t no_lleno2;
pthread_cond_t no_vacio2;
pthread_mutex_t mutex2;

void *productor(void *arg) {
	int datos_a_producir = *(int *) arg;
	int pos = 0;

	for (int i = 0; i < datos_a_producir; i++) {

		pthread_mutex_lock(&mutex1);

		while (cont1 == MAX_BUFFER)
			pthread_cond_wait(&no_lleno1, &mutex1);

		buffer1[pos] = i;
		printf("> Produciendo: %d\n",i); fflush(stdout);

		pos = (pos + 1) % MAX_BUFFER;
		cont1++;

		pthread_cond_signal(&no_vacio1);
		pthread_mutex_unlock(&mutex1);
	}

	printf("Productor: No más items.Saliendo...\n");
}

void *procesador(void *arg) {
	int dato_recibido, dato_procesado, pos1=0, pos2=0;
	int datos_a_procesar = *(int *)arg;

	for (int i = 0; i < datos_a_procesar; i++) {

		pthread_mutex_lock(&mutex1);

		while (cont1 == 0)
			pthread_cond_wait(&no_vacio1, &mutex1);
		dato_recibido = buffer1[pos1];
```

```
		pos1 = (pos1 + 1) % MAX_BUFFER;
		cont1--;

		pthread_cond_signal(&no_lleno1);
		pthread_mutex_unlock(&mutex1);

		dato_procesado = funcion_procesamiento(dato_recibido);


		pthread_mutex_lock(&mutex2);

		while (cont2 == MAX_BUFFER)
			pthread_cond_wait(&no_lleno2, &mutex2);
		buffer2[pos2] = dato_procesado;
		pos2 = (pos2 + 1) % MAX_BUFFER;
		cont2++;

		pthread_cond_signal(&no_vacio2);
		pthread_mutex_unlock(&mutex2);
	}

	printf("Procesador: No más items.Saliendo...\n");
}

void *consumidor(void *arg) {

	int datos_a_consumir = *(int *)arg;
	int pos = 0;

	for (int i = 0; i < datos_a_consumir; i++) {
		pthread_mutex_lock(&mutex2);

		while (cont2 == 0)
			pthread_cond_wait(&no_vacio2, &mutex2);

		printf("< Consumiendo: %d\n",buffer2[pos]);   fflush(stdout);

		pos = (pos + 1) % MAX_BUFFER;
		cont2--;

		pthread_cond_signal(&no_lleno2);
		pthread_mutex_unlock(&mutex2);

	}
	printf("Consumidor: No más items.Saliendo...\n");
}

int main(int argc, char *argv[]) {

	pthread_t th1, th2, th3;

	if (argc != 2) {
		printf("Introduzca solo el numero de items a generar\n");
		exit(-1);
	}

	int num_items = atoi(argv[1]);
```

```
pthread_create(&th1, NULL, productor, (void *) &num_items);
pthread_create(&th2, NULL, procesador, (void *) &num_items);
pthread_create(&th3, NULL, consumidor, (void *) &num_items);

pthread_join(th1, NULL);
pthread_join(th2, NULL);
pthread_join(th3, NULL);

printf("Fin del thread principal\n");
return 0;}
```

Apartado b)

La parte del código donde sería más probable generar una condición similar al DeadLock sería dentro de la función procesador. En el caso de que se intente hacer lock sobre el segundo mutex sin haber liberado el primero, cualquier problema en el thread consumidor que evite liberar el mutex2 llevaría a un bloqueo de todo el pipeline. (Una situación genuina de DeadLock se podría dar si el consumidor también tuviera que sincronizarse con el productor y no se cuidara el manejo de los mutex)

La solución proporcionada evita este problema cuidando que solo intentamos bloquear mutex2 una vez que hemos liberado mutex1.

Ejercicio 4 (2,5 Puntos).

Disponemos de un disco duro de 50 MB, cuyos parámetros de configuración del sistema de ficheros tipo UNIX son los siguientes:

- Tamaño de bloque: 512 bytes.
- Tamaño de la dirección de los bloques: 2 bytes.
- Número de i-nodos : 100
- Bloque de carga (Boot) que ocupa 4 bloques.
- Superbloque ocupa 8 Kbytes.
- Se usa un Mapa de Bits para indicar que bloques están usados y cuales libres.
- Campos de un i-nodo:
    - Atributos del fichero:
        - Id. del Propietario y del grupo.
        - Permisos de lectura para el dueño, grupo y resto del mundo.
        - Permisos de escritura para el dueño, grupo y resto del mundo.
        - Permisos de ejecución para el dueño, grupo y resto del mundo.
    - Contador de enlaces (1 byte).
    - 5 punteros directos.
    - 2 punteros indirectos simples.
    - 2 punteros indirectos dobles.

El sistema de ficheros puede utilizar enlaces duros y simbólicos

Responder a las siguientes preguntas:

a) ¿Qué tamaño máximo podrá tener un fichero almacenado en este disco? Razona la respuesta.

b) Número máximo de enlaces duros (además del nombre inicial) que se pueden realizar de un fichero en este disco.
c) Número máximo de enlaces simbólicos que se pueden realizar de un fichero.
d) El sistema de ficheros tiene la siguiente estructura:
/x/y/z

Se realizan sobre este sistema las siguientes órdenes:

ln /x/y/z /x/y/t

ln -s /x/y/t /x/y/r

rm /x/y/t

indicar que cambios se producen en el sistema de ficheros al realizar cada una de estas acciones.

e) Una vez realizadas estas operaciones, indicar el número de accesos a disco cuando se realiza la instrucción cat /x/y/r, y el resultado de la misma. Suponer que en disco solo se encuentra cargado el i-nodo /

SOLUCION:

a) El tamaño máximo de un fichero depende de las referencias del inodo y del Nº de direcciones de bloque que caben en un bloque de datos (referencia indirecta) que es igual al tamaño del bloque (1024 bytes) entre el tamaño de la dirección de bloque (2 bytes), así:

REFERENCIAS	Nº BLOQUES
5 bloques directos	5
2 referencias indirectas simple:	2*512/2 = 512
2 referencias indirectas dobles:	2*(512/2)*(512/2)= 131072

Por tanto el Nº de bloques es:

5 + 512 + 131072 = 131589 bloques

Así, el tamaño máximo (en bytes) de un fichero en este sistema de ficheros es:

131589 *512 B = 67373568 B (aproximadamente 64 MB)

Dado que el disco es de 50 MB, un fichero de tamaño máximo no cabría ; por lo tanto el tamaño de un fichero viene limitado por la capacidad del disco. Para este caso, el tamaño máximo del disco deberá de calcularse de la siguiente forma:

Tamaño máximo de un fichero = Número de bloques del disco Numero de bloques ocupados por el resto de estructuras.

Las estructuras típicas que nos define el enunciado son:

Boot	SuperBloque	Mapas de Bits	Nodos-I	Datos y Directorios

Por lo tanto:

Tamaño Boot = 2 bloques

Tamaño Superbloque = 4 bloques

Tamaño Mapa de Bits = 1 bit/bloque * 131589 bloques / 512 * 8 Tamaño del bloque en bits = 32.16 => aproximadamente 33 bloques * 2 Mapas de bit => 66 bloques

Tamaño Nodos-I = 100 bloques

Tamaño máximo del fichero = 50 MB/512~(2 + 4 + 66 + 100) = 102400~172 = 102228 bloques => aproximadamente 49 MB

b) El número de enlace duros solo depende del tamaño del contador de enlaces que indica el número de enlaces duros realizados. El contador de enlaces de un inodo de este sistema de ficheros dispone de 1 byte con lo cual puede haber desde 0 hasta 255 enlaces duros. Dado que el nombre de fichero inicial también es un enlace duro, el Nº de posibles enlaces duros, además del nombre de fichero inicial, será de 254 enlaces.

c) El número de enlaces simbólicos solo depende del número de ficheros que se puedan crear. Dado que el disco solo dispone de 100 inodos, el número máximo de enlaces simbólicos sería de 100, pero como mínimo en el disco existen dos inodos ocupados (el del directorio raíz y el del fichero "original"). Por lo tanto el número máximo de enlaces simbólicos a un fichero puede ser 98.

d) Los cambios que se producirían serían los siguientes:

ln /x/y/z /x/y/t

se crearía una nueva entrada en el directorio y con nombre t, cuyo i-nodo sería el mismo que el del fichero z

se sumaría 1 al contador de enlaces del fichero z

ln -s /x/y/t /x/y/r

se crearía una nueva entrada en el directorio y con nombre r, para la que se reservaría un nuevo i-nodo cuyo contador de enlaces sería 1, dicho i-nodo apuntaría a un nuevo bloque de datos cuyo contenido sería /x/y/t.

rm /x/y/t

se liberaría el bloque de datos del fichero t, se comprueba el contador de enlaces del i-nodo de t, y vemos que es 1, con lo que se liberaría también el i-nodo del fichero t, desaparecerá la entrada del directorio /x/y correspondiente al fichero t.

e) Para realizar la instrucción cat /x/y/r, accedemos a memoria al i-nodo /,
   - accedemos a disco al bloque de datos del directorio /, obtenemos el número de i-nodo de x -> 1 acceso
   - accedemos a disco al i-nodo de x, comprobamos cual es su bloque de datos y accedemos a él, leemos cual es el i-nodo del directorio y -> 2 accesos
   - accedemos a disco al i-nodo de y, comprobamos cual es su bloque de datos y accedemos a él, leemos cual es el i-nodo del fichero r -> 2 accesos
   - accedemos a disco al i-nodo de r, comprobamos cual es su bloque de datos y accedemos a él, leemos la siguiente información /x/y/t -> 2 accesos
   - tenemos que resolver la ruta /x/y/t, luego hay que acceder al directorio /, al bloque de datos de este directorio, accedemos al i-nodo del directorio y, y a su bloque de datos, toda esta información estaba en disco, tratamos de acceder a la entrada correspondiente a t, pero no está en este directorio luego se producirá un error.
   - Total 7 accesos a disco