Examen Febrero de 2005

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida del examen.

Formato

- **Indique su nombre completo y número de cédula en cada hoja (No se corregirán las hojas sin nombre, sin excepciones) Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva. (No se corregirá la hoja que tenga el ejercicio compartido, sin excepciones)**
- **Si se entregan varias versiones de un problema solo se corregirá el primero de ellos.**

Dudas

- **Sólo se contestarán dudas de letra.**
- **No se aceptarán dudas en los últimos 30 minutos del examen.**

Material

- **El examen es SIN material (no puede utilizarse ningún apunte, libro ni calculadora). Sólo puede tenerse las hojas del examen, lápiz, goma y lapicera en su banco. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.**

Aprobación

- **Para aprobar el examen se debe tener un ejercicio entero bien hecho y medio más.**

Finalización

- **El examen dura 4 horas.**

Problema 1

Para las siguientes partes, conteste justificando brevemente, cada una de las preguntas.

1. Enumere y explique cuales son los tres requisitos que debe cumplir una solución al problema de la sección crítica.
2. Ilustre y explique brevemente como se implementa la correspondencia entre direcciones virtuales y físicas en un esquema de segmentación.
3. De tres ventajas de la segmentación frente a la paginación.
4. Indique la diferencia entre planificación a corto, mediano y largo plazo.
5. Describa brevemente los pasos para el manejo de un fallo de páginas (una vez que se determinó que este ocurrió) en un modelo de paginación por demanda.
6. Describa brevemente los algoritmos de reemplazo de página siguientes: FIFO, Óptimo y LRU.
7. Explique brevemente porqué se produce hiperpaginación (no la defina, ejemplarice).
8. Enumere y describa brevemente tres métodos de planificación de disco.
9. ¿Cuándo se dice que un despachador es apropiativo?
10. Utilizando el algoritmo del banquero, y dada la siguiente situación de procesos y recursos

	Asignación	Max	Disponible
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 4 5	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

determine si el sistema está en estado seguro, y en caso afirmativo, si al llegar un solicitud del proceso P1 de (0,4,2,0), ésta puede ser satisfecha de inmediato. Justifique su/s respuesta/s.

11. Defina brevemente: sistema distribuido y sistema de tiempo real.
12. Explique porqué debe protegerse el vector de interrupciones y las rutinas de procesamiento de las interrupciones. ¿Cómo se logra esta protección mediante hardware?
13. Mencione el principal problema que debe enfrentar un sistema operativo si ejecuta sobre un equipo multiprocesador, con una única memoria común.
14. Describa brevemente dos limitaciones de los hilos a nivel de usuario sin soporte del núcleo.
15. ¿Cuál es la principal diferencia que hay entre implementar un servicio de comunicación de mensajes entre procesos de un mismo computador, respecto a hacerlo entre hilos de un mismo proceso?
16. Mencione una ventaja y una desventaja de un sistema monolítico.

Problema 2

En una repostería se preparan postres con el siguiente equipo:

- **Veinte** expendedores de ingredientes. Cada uno entrega un ingrediente distinto (no hay repetidos), a solicitud de las cocineras. Invocan al procedimiento DAR_INGREDIENTE para hacer efectivo el envío.
- **Diez** cocineras dedicadas exclusivamente a cocinar. Necesitan **cinco** ingredientes para preparar cada postre. Utilizan la función INDICAR_INGREDIENTES que retorna la lista de ingredientes que se necesitan para el postre actual.

Cada cocinera trabaja de forma independiente del resto de las cocineras y hasta que no reciben todos los ingredientes no pueden cocinar. Las cocineras pueden pedir más de los cinco ingredientes que necesitan si les da lo mismo recibir uno u otro, pero solamente deben recibir cinco ingredientes de los que pidieron, es incorrecto recibir mas de cinco.

Se pueden utilizar tareas auxiliares, pero las cocineras solo pueden comunicarse con los expendedores. Algunos mensajes (send y/o receive) pueden demorar en llegar, por lo que se debe tener cuidado de no equivocarse con recepciones de mensajes tardíos (o sea de pedidos que ya fueron satisfechos)

Se pide:

Implementar el sistema descrito utilizando Send/Receive con nombrado explicito por lo menos para alguna de las dos primitivas. Definir el resto de la semántica.

Se cuenta con las siguientes funciones y procedimientos:

- DAR_INGREDIENTE (Nro.Cocinera): recibe como parámetro el número de cocinera que lo necesita. Este procedimiento es ejecutado por las expendedoras.
- INDICAR_INGREDIENTES: retorna un array con por lo menos 5 elementos, y máximo 10. Estos son números del 1 al 20, representando el ingrediente. Los elementos del array de retorno son distintos entre si.
- GET_PID: retorna el identificador del proceso que lo ejecuta
- COCINAR: invocado por la cocinera, se bloquea hasta que termine de cocinar. Este procedimiento debe ejecutarse una vez que terminaron de ejecutar 5 expendedoras su DAR_INGREDIENTE para esa cocinera.
- Pid_expendedores[1..20] of pids . array global que contiene los pids de las expendedoras
- Pid_cocineras[1..10] of pids array global que contiene los pids de las cocineras

Nota:

No olvidarse de los siguientes puntos:

- Las cocineras necesitan ni mas ni menos que 5 ingredientes para cocinar pero pueden pedir mas que 5.
- Las cocineras solo pueden comunicarse con las expendedoras.
- Los mensajes pueden demorar (las demoras de los mensajes no son fijas).
- Las expendedoras deben tener el mayor nivel de concurrencia posible.

Problema 3

Parte 1:

Se tiene un sistema con memoria virtual que utiliza la técnica de la segmentación paginada por demanda. En este sistema se ejecuta un proceso A con tres segmentos T (código), D (datos) y P (pila). Los segmentos de datos y de pila pueden crecer. Las longitudes de estos segmentos se especifican en la tabla del Proceso A. La memoria física consta de 4 marcos de 512 bytes cuyo contenido en un instante *t* es el de la tabla 2, donde la nomenclatura T*i* indica la página lógica *i* del segmento T (ejemplo: D0 es la página 0 del segmento D).

Segmento	Longitud
Texto	2500
Datos	1500
Pila	300

Tabla 1.- Segmentos del Proceso A

Marco	(Segmento,Página)
0	T2
1	D0
2	T1
3	P0

Tabla 2.- Describe que Página hay en cada marco de página en Memoria Principal

Si en ese instante *t* se produjese la referencia a memoria que se especifica en los siguientes casos (casos independientes entre sí), diga si se produciría fallo de página y, en caso afirmativo, el tipo de fallo y el tratamiento que recibiría por parte del sistema operativo. Justifique detalladamente su respuesta.

a) Acceso a la dirección 1000 del segmento de Datos (D, 1000) para escribir en una variable.

b) Acceso a la dirección 3510 del segmento de Texto (T, 3510) para ejecutar una instrucción.

c) Acceso a la dirección 1510 del segmento de Datos (D, 1510) para crear una variable dinámica.

d) Acceso a la dirección 950 del segmento de Texto (T, 950) para ejecutar una instrucción.

Parte 2:

Se ha decidido la compra un disco duro de 100 MB para tareas de alto rendimiento. El administrador de sistemas de la empresa ha decidido dar una serie de parámetros en la configuración del sistema de archivos tipo UNIX (tamaño del volumen, tamaño de bloque y número de i-nodos) con el objetivo de maximizar el rendimiento del sistema en lo que se refiere a número de accesos a disco. Las características finales que se aplicaran a la hora de realizar la creación del sistema de archivos son:

- Tamaño de bloque: 1024 bytes.
- Tamaño de la dirección de los bloques: 4 bytes.
- Número de i-nodos : 50
- Un bloque de carga (Boot) que ocupa 2 bloques.
- Un Superbloque que ocupa 8 Kbytes y se usa un Mapa de Bits para indicar que bloques están usados y cuales libres.
- Campos de un i-nodo:
    - Atributos del archivo:
        - 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).
    - 4 punteros directos.
    - 4 punteros indirectos simples.
    - 4 punteros indirectos dobles.
- Puede asumir que cada i-nodo ocupa un bloque en disco.

Responder a las siguientes preguntas:

a) ¿Que tamaño máximo podrá tener un archivo almacenado en este disco? Justifique adecuadamente.
b) Si el objetivo es minimizar el acceso a bloques de disco ¿Qué parámetro se deberían incrementar/decrementar por el administrador para conseguir dicho propósito? ¿Qué contrapartida tendría la solución elegida para minimizar el acceso a bloques de disco?

Explique detalladamente.

==========

Soluciones

==========

Problema 1.

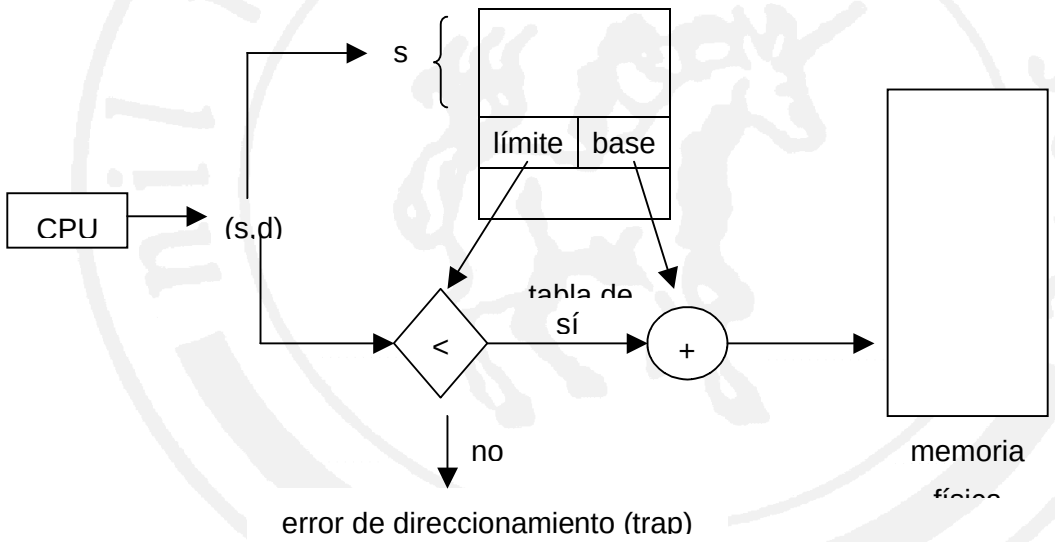
1.

Exclusión mutua: A lo máximo un proceso puede estar ejecutando en su sección crítica.

Progreso: La decisión de cuál será el próximo proceso a entrar en su sección crítica debe ser tomada solo por aquellos que no estén ejecutando en su sección restante, y esta decisión no puede postergarse indefinidamente.

Espera limitada: Desde el momento en que un proceso particular intentó entrar a su sección crítica, debe haber un límite en la cantidad de veces que otros procesos logran entrar antes que él.

2.



La dirección lógica está dada por un número de segmento (s) y un desplazamiento (d) dentro del mismo. Con s se indiza la tabla de segmentos, que tiene la base y el límite del segmento en memoria física. Si d < 0 ó d ≥ límite, es un intento de acceso erroneo o no válido, si no, se suma el desplazamiento a la base y se obtiene la dirección real.

3.

Perspectiva de memoria del usuario: Cada segmento representa una porción de un programa definida semánticamente, haciendo que todos los registros del mismo se utilicen de la misma forma. El usuario no ve la memoria como un arreglo lineal de palabras (el enfoque de la paginación), sino que identifica estructuras de datos como tablas, pilas, módulos, etc. Cada uno se puede indentificar individualmente y tiene un tamaño variable.

Protección: Con segmentación es posible definir el nivel de acceso a cada segmento. Agregando bits en las entradas de la tabla de segmentos, por hardware se pueden evitar acceso ilegales, p. ej. escribir en un segmento de sólo lectura.

Compartimiento: Cada proceso tiene una tabla de segmentos asociada a su PCB. El despachador la copia a la tabla de segmentos única del hardware cuando pasa un proceso a ejecución, y la respalda desde allí cuando pierde el procesador. Como un segmento se corresponde a un objeto de programa particular, se puede compartir este objeto haciendo que las respectivas entradas en las tablas de segmentos apunten a la misma localidad física.

4.

Planificador a largo plazo: Selecciona un proceso pendiente de ejecución (en un spooler, en almacenamiento secundario) para que pase a estado listo y empiece a competir con los demás procesos por CPU y memoria.

Planificador a corto plazo: Selecciona un proceso de la cola de proceso listos para asignarle CPU.

Planificador a mediano plazo: Sirve para reducir una excesiva contienda por CPU, para eliminar temporalmente procesos de memoria hacia almacenamiento secundario. Cuando hay más recursos disponibles, los retorna a donde quedó.

Nota: Es incorrecto definir cada planificador por la frecuencia con que ejecuta. La frecuencia es una característica, pero no una definición.

5.

Como ya se determinó que ocurrió el fallo de página, sabemos que no es un acceso inválido, ni que la página está en memoria.

a. Encontrar un marco libre. Si no se encuentra, utilizar un algoritmo de reemplazo para seleccionar un marco, y moverlo a disco. (Esto incluye dejar el proceso que provocó el fallo en estado bloqueado, mientras espera en la cola del dispositivo primero, y luego la transferencia).

b. Planificar una operación para leer de disco la página hacia el marco elegido. Misma aclaración que en (a). Durante ambas transferencias ejecutan otros procesos que estaban en la cola de listos

c. Cuando finalice la transferencia (por una interrupción de fin de E/S), el sistema modificará la tabla de páginas para indicar que ahora se encuentra en memoria, y cambiará el estado del proceso que provocó el fallo a listo.

d. Cuando la CPU se asigne al proceso, se reiniciará en la instrucción que provocó el fallo.

6.

FIFO (First In First Out): Para reemplazar una página, se elige la que estuvo en memoria más tiempo. Es fácil de programar, pero no siempre es bueno pues no toma en cuenta el uso que se le está dando a la página seleccionada. Podría haber una página que se cargó hace mucho y tiene un uso frecuente, y otra cargada hace menos tiempo pero que se usó sólo esa vez. FIFO elegirá la primera y no la segunda.

Óptimo: Tiene la menor tasa de fallos de página pues reemplaza la página que no se utilizará durante el mayor período de tiempo. Es díficil de implementar sin un conocimiento futuro de la serie de referencias.

LRU (Least Recently Used): Remplaza la página que no se ha utlizado en más tiempo, suponiendo el pasado reciente como una aproximación de las próximas referencias. Requiere ayuda de hardware.

7.

Se produce cuando un proceso tiene más páginas activas que marcos asignados, lo que hace que pase más tiempo paginando que ejecutando.

Nota: No es válido definirlo como "exceso de paginación". Se pide la causa, no el efecto que produce.

8.

Se listan todas las posibles (sólo se piden 3):

FCFS (First Come First Served): Atender las solicitudes en orden de llegada. Es fácil de programar pero no ofrece el mejor tiempo de servicio, pues se invierte mucho tiempo en cambios de dirección de la cabeza de lectura.

SSTF (Shortest Seek Time First): Atiende la solicitud de menor tiempo de posicionamiento, respecto a la posición actual. Debe estudiarse como evitar posposición indefinida.

SCAN: La cabeza de lectura comienza en un extremo y se desplaza al otro, sirviendo las solicitudes al llegar a cada pista, hasta alcanzar el extremo opuesto, y así sucesivamente.

C-SCAN: Como SCAN, pero cuando llega al extremo opuesto del disco retorna de inmediato al inicio, donde hay más solicitudes pendientes, sin atender solicitudes intermedias.

LOOK: Como SCAN, pero el algoritmo mira si hay más solicitudes pendientes en la dirección en que actualmente se está moviendo, y si no hay, cambia de dirección. C-LOOK es el equivalente para C-SCAN. En general son la forma en que se implanta SCAN y C-SCAN, por razones de eficiencia.

9.

Cuando asegura que en todo momento se esté ejecutando el proceso de mayor prioridad, según el esquema de prioridad, explícito o implícito, que se maneje. Para esto se hace necesario poder quitar el procesador a un proceso que actualmente esté en ejecución.

10.

Primero establecemos las matrices Necesidad (Max – Asignación), Trabajo = Disponible, y Fin = F para todo proceso

Necesidad

	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	1	1
P3	0	0	2	0
P4	0	6	4	2

Trabajo

A	B	C	D
1	5	2	0

Fin

F	F	F	F	F

Busco secuencia segura (hay más de una, aquí se muestra una)

P0 [(0,0,0,0) < (1,5,2,0)]
Trabajo queda en (0,0,1,2) + (1,5,2,0) = (1,5,3,2)
Fin en (T,F,F,F,F)

P2 [(1,0,1,1) < (1,5,3,2)]
Trabajo queda en (1,3,4,5) + (1,5,3,2) = (2,8,7,7)
Fin en (T,F,T,F,F)

P1 [(0,7,5,0) < (2,8,7,7)]
Trabajo queda en (1,0,0,0) + (2,8,7,7) = (2,15,12,7)
Fin en (T,T,T,F,F)

P3 [(0,0,2,0) < (2,15,12,7)]
Trabajo queda en (0,6,3,2) + (2,15,12,7) = (2,21,15,9)
Fin en (T,T,T,T,F)

P4 [(0,6,4,2) < (2,21,15,9)]
Trabajo queda en (0,0,1,4) + (2,21,15,9) = (2,21,16,15)
Fin en (T,T,T,T,T)

Existe al menos una secuencia segura (<P0,P2,P1,P3,P4>), por lo que el sistema está en estado seguro.

Con la solicitud del proceso P1 de (0,4,2,0), el estado queda en:

Asignación					Necesidad			
	A	B	C	D		A	B	c	D
P0	0	0	1	2	P0	0	0	0	0
P1	1	4	2	0	P1	0	3	3	0
P2	1	3	4	5	P2	1	0	1	1
P3	0	6	3	2	P3	0	0	2	0
P4	0	0	1	4	P4	0	6	4	2

Disponible				Fin				
1	1	0	0	F	F	F	F	F

Trabajo			
1	1	0	0

Busco secuencia segura

P0 [(0,0,0,0) < (1,1,0,0)]
Trabajo queda en (0,0,1,2) + (1,1,0,0) = (1,1,1,2)
Fin en (T,F,F,F,F)

P2 [(1,0,1,1) < (1,1,1,2)]
Trabajo queda en (1,3,4,5) + (1,1,1,2) = (2,4,5,7)
Fin en (T,F,T,F,F)

P1 [(0,3,3,0) < (2,4,5,7)]
Trabajo queda en (1,4,2,0) + (2,4,5,7) = (3,8,7,7)
Fin en (T,T,T,F,F)

P3 [(0,0,2,0) < (3,8,7,7)]
Trabajo queda en (0,6,3,2) + (3,8,7,7) = (3,14,10,9)
Fin en (T,T,T,T,F)

P4 [(0,6,4,2) < (3,14,10,9)]
Trabajo queda en (0,0,1,4) + (3,14,10,9) = (3,14,11,13)
Fin en (T,T,T,T,T)

Con esta asignación, existe al menos una secuencia (<P0,P2,P1,P3,P4>), por lo que es posible atender esa solicitud.

11.

Sistema distribuido: Sistema operativo fuertemente acoplado en software y debilmente acoplado en hardware, donde los procesadores no comparten memoria física ni reloj. El objetivo es compartir recursos, compartir cargas, sostener la comunicación entre procesos y usuarios, y aumentar la confiabilidad del conjunto.

Sistema de tiempo real: Sistema que tiene restricciones temporales bien definidas, a diferencia de los sistemas por lotes o de tiempo compartido. La ejecución para considerarse correcta debe hacerse dentro de un margen de tiempo definido. Generalemente se utilizar ROM en vez de almacenamiento secundario. Suelen utilizarse para aplicaciones dedicadas.

12.

Porque un programa de usuario (malintencionado) podría modificarlos para que le retornen el control a él, y así pasar a ejecutar en modo monitor.

Una forma posible de hacerlo es con registros base y límite que indiquen el intervalo de direcciones legales de cada proceso. El hardware de CPU verifica que cada dirección generada por el proceso esté en el intervalo, y si no, se generará una trap, y el sistema abortará el proceso. Este mecanismo también protege los programas entre sí.

13.

El principal desafío es proteger las secciones críticas del sistema operativo. Esto no era un problema cuando había un solo procesador y a lo sumo un proceso del sistema ejecutando en un momento dado.

14. (Se listan todas las posibles, sólo se piden 2)

Un fallo de página de un hilo hace que se bloquee el proceso con todos los otros hilos.

Idem con E/S.

Idem con llamadas al sistema.

Aún si el sistema es multiprocesador y tiene CPU's ociosas, todos los hilos competirán por un solo procesador, el del proceso que los contiene.

En caso de ser múltiples hilos de un proceso que ejerce como máquina virtual de ellos (por ejemplo planificándolos), una vez que está ejecutando un hilo, no existe forma expropiarle el control para dárselo al proceso/máquina virtual. La única forma es que los propios hilos lo cedan (por ejemplo llamando algún servicio de la máquina virtual, en vez de hacer llamadas directas al sistema).

15.

El primero requiere un mecanismo gestionado por el sistema, como colas de mensajes, memoria compartida, etc, con el costo de las llamadas correspondientes, y los cambios de contexto. Con hilos es posible hacerlo sin intervención del sistema, direccionando sobre el mismo espacio virtual.

16.

La ventaja es la eficiencia, la desventaja es que la falta de modularización lo hace más dificil de mantener y portar por parte de los desarrolladores del sistema.

```
================================================

Problema 2.


/*
  Send no bloqueante con nombrado explícito
  Receive bloqueante desde cualquier proceso
*/

/* arreglo con los identificadores de los expendedores */
Var pidExpendedores: array [1..20] of integer;

/* arreglo con los identificadores de las cocineras */
Var pidCocineras: array [1..10] of integer;

/* pid del administrador */
Var pidAdmin: integer;

type mensage =    record
                        tipo: string;
                        dato1, dato2, dato3: integer;
                  end

type pedido =     record
                        numeroPedido: integer
                        expendedores: Lista(integer)
                        satisfecho: boolean;
                  end


Procedure cocinera
begin
      var ingredientes: Lista(integer);
          i, numPedido, pid: integer;

      numPedido := 0;
      pid := GET_PID();
      while true do
      begin
            ingredientes = INDICAR_INGRDIENTES();
            for i:=1 to ingredientes.largo do
                  SEND(pidExpendedores[ingredientes[i]], ("COCINERA", pid,
                                                            numPedido, 0));
            for i:=1 to 5 do
                  RECEIVE(NULL);
            numPedido = numPedido + 1;
            COCINAR();
      end
end


Procedure administrador()
begin
      var msg: mensaje;
          pedidos: Array [1..10] of Lista(pedido);
          expendedores: Lista(integer);
          completo: pedido;
          pidExpendedor, pidCocinera, numPedido, nroCocinera: integer;
```

```
      while true do
      begin
            RECEIVE(msg);
            pidExpendedor = msg.dato1;
            pidCocinera = msg.dato2;
            numPedido = msg.dato3;
            nroCocinera = buscar(pidCocinera, pidCocineras);

            if not pedidoSatisfecho (pedidos[nroCocinera], numPedido) then
                  if existePedido (pedidos[nroCocinera], numPedido) then
                        agregarAPedido(pedidos[nroCocinera], numPedido,
                                                            pidExpendedor);
                  else
                        nuevoPedido(pedidos[nroCocinera], numPedido,
                                                            pidExpendedor);

            for i:= 1 to 10 do
            begin
                  /* busca en la lista un pedido no satisfecho con 5
                     expendedoras, si encuentra retorna true y el pedido en la
                     variable completo */
                  while obtenerPedidoCompleto (pedidos[i], completo) do
                  begin
                        expendedores = completo.expendedores;
                        for i:=1 to expendedores.largo do
                              SEND (expendedores[i], ("ADMIN", pidCocineras[i],
                                                                  0, 0);
                        completo.satisfecho = true;
                  end
            end
      end
end

Procedure expendedor()
begin
      var msg: mensaje;
          pid, pidCocinera, numPedido: integer;

      pid = GET_PID();
      while true do
      begin
            RECEIVE(msg);
            pidCocinera = msg.dato1;
            numPedido = msg.dato2;
            if msg.tipo = "COCINERA" then
                  SEND(pidAdmin, ("EXPENDEDOR", pid, pidCocinera, numPedido));
            else if msg.tipo = "ADMIN" then
            begin
                  DAR_INGREDIENTE(pidCocinera);
                  SEND(pidCocinera, NULL);
            end
      end
end
```

==========

Problema 3.

Parte 1:

Referencia y tipo de fallo	Acciones
(D,1000) → Fallo de Página	La página D1 debe cargarse en memoria reemplazando a una de las cuatro presentes en memoria.
(T,3510) → Dirección fuera de rango	La dirección no está dentro del tamaño del segmento. El sistema provoca una excepción
(D,1510) → Fallo de Página	Aunque la dirección está fuera del rango del segmento D, el sistema permite que crezca dicho segmento. La página D2 será asignada al proceso (puede que necesite ser llevada desde disco si todavía no ha sido referencia y pertenece al área de datos sin valores iniciales), reemplazado una de las que hay en memoria.
(T,950) → No hay fallo	Es una dirección de la página T1, ya presente en memoria. Por lo tanto, se atenderá la solicitud de dicha dirección.

Parte 2:

a) El tamaño máximo de un archivo depende de las referencias del inodo y del N° 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 (4 bytes), así:

REFERENCIAS	N° BLOQUES
4 bloques directos	4
4 referencias indirecta simple:	4*1024 / 4 = 1024
4 referencias indirecta doble:	4*(1024 /4)*(1024 / 4) = 262144

Por tanto el N° de bloques es: 4 + 1024 + 262144 = 263172 bloques;

Así, el tamaño máximo (en bytes) de un archivo en este sistema de archivos es:

263172 * 1024 = 269488128 bytes =>263172 KB (aproximadamente 263 Mbytes)

Dado que el disco es de 100 MB, un archivo de tamaño máximo no cabría ; por lo tanto el tamaño de un archivo 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 archivo = (Tamaño del disco / Tamaño del bloque) – 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 = 8 bloques

Tamaño Mapa de Bits = (100 * 2^20) / 2^10 = 102400 bits => 100 bloques

Tamaño Nodos-I = 50 bloques

Tamaño máximo del archivo = (100* 2^20 / 2^10) - (2+8+100+50) = 102400 – 160 = 102240 bloques

Tamaño máximo en bytes = 102240 * 1024 bytes = 104693760 bytes = 102240 kbytes = 99,84 MB

b) En general, el parámetro clave a la hora de conseguir menor número de accesos a disco es el tamaño de bloque. La transferencia de un archivo requiere buscar cada bloque que lo forma en disco, esperar el tiempo de latencia y hacer la transferencia de datos.

Como contrapartida al tamaño grande de bloque se tiene que la cantidad de fragmentación interna es elevada y eso provoca que el porcentaje de de disco aprovechado para datos decrezca.