Examen Febrero de 2002

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

- LEA ATENTAMENTE LAS SIGUIENTES INSTRUCCIONES.
- El examen es SIN material (no puede utilizarse ningún apunte, libro ni calculadora). Solo puede tener las hojas del examen, lápiz, goma de borrar y lapicera en su banco. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.
- Indique su nombre completo, número de cédula en cada hoja (las hojas sin nombre no se corregirán sin excepciones).
- Escriba el nombre en la hoja de letra, la cual deberá entregar conjuntamente con la solución.
- Indique la cantidad de hojas que entrega en la primer hoja.
- Escriba las hojas de un solo lado (solo se corregirá de un solo lado de la hoja sin excepciones).
- Empiece cada ejercicio 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.
- En las preguntas múltiple opción solo valdrá lo respondido en la hoja para el scanner.
- Solo se contestaran dudas de letra. No se aceptarán dudas los últimos 20 minutos del parcial.
- Solo se corregirán los ejercicios a aquellos estudiantes que obtengan un mínimo del 40% de las puntos correspondientes a las respuestas múltiple opción.
- Los puntajes de la parte múltiple opción son:
  - Respuesta correcta 3 puntos
  - Respuesta incorrecta 1 puntos
  - Sin contestar 0 punto
- El examen dura cuatro horas.
- **Marque en la casilla de control de la hoja del scanner el 1.**

Múltiple Opción

1. Los semáforos...
   a) eliminan el problema del interbloqueo
   b) son más potentes que las regiones críticas condicionales
   c) tienen dos operaciones públicas (wait y signal)
   d) todas las anteriores son ciertas
2. En un sistema multihilo:
   a) los hilos de un mismo proceso pesado comparten el mismo código
   b) los hilos de distintos procesos pesados comparten la misma pila
   c) los hilos de un mismo proceso pesado comparten el mismo contador de programa
   d) todas las anteriores son falsas
3. El modo dual de operación consigue que el sistema informático sea:
   a) más veloz
   b) más fácil de usar
   c) más seguro
   d) ninguna de las anteriores
4. Una condición no necesaria para la aparición del interbloqueo es
   a) la existencia de recursos con número finito de ejemplares
   b) la posibilidad de espera con retención de recursos
   c) la existencia de recursos que exigen exclusión mutua
   d) la aparición de un estado inseguro
5. En UNIX, ¿qué ocurre con los ficheros abiertos por un proceso cuando éste crea un proceso hijo?
   a) el hijo no tiene inicialmente ningún fichero abierto proveniente del padre.
   b) el usuario decide qué ficheros aparecen inicialmente abiertos en el hijo.
   c) sólo aparecen abiertos en el hijo los flujos estándares del padre (stdin, stdout, stderr).
   d) todos los ficheros del padre aparecen abiertos en el hijo.
6. Suponiendo que inicialmente los marcos disponibles están vacíos, la cadena de referencias a memoria (1,2,3,4,1,3,4,2,3,1,3,4):
   a) provoca al menos dos fallos de página.
   b) provoca como mucho tres fallos de página.
   c) provoca exactamente cuatro fallos de página, independientemente de la política de reemplazo.
   d) según la política de reemplazo, podría no provocar fallos de página.
7. Se analiza un sistema de paginación por demanda y se obtiene que, con cierta carga de trabajo, la CPU se emplea un 15% y el disco de intercambio (swap) está ocupado un 92% del tiempo. ¿Cuál de estas acciones aumentaría más la utilización de la CPU?
   a) ampliar la memoria principal.
   b) aumentar el grado de multiprogramación.
   c) cambiar el disco de intercambio por otro de más capacidad.
   d) cambiar la CPU por otra más rápida.

8. Se tiene una cadena de referencias a memoria, que dan 10 fallos de página si se gestionan con una política FIFO. ¿Cuántos fallos de página se obtendrían con la política óptima?
   a) una cantidad mayor o igual.
   b) una cantidad menor o igual.
   c) una cantidad estrictamente mayor.
   d) una cantidad estrictamente menor.
9. La característica distintiva del enlace dinámico es que resuelve referencias a memoria en:
   a) tiempo de carga.
   b) tiempo de compilación.
   c) tiempo de ejecución.
   d) tiempo de respuesta.
10. Considere estas cadena de referencias a memoria: C1=(3,4,4,5,3,2,1,4,3,1,2) y C2=(3,5,5,4,3,2,1,5,3,1,2). ¿Cuál originará más fallos de página, suponiendo que inicialmente todos los marcos están libres?
    a) C1.
    b) C2.
    c) depende de la política de sustitución empleada.
    d) las dos dan los mismos fallos.
11. Para procesar simultáneamente operaciones de CPU con operaciones de E/S se requieren:
    a) Dispositivos rápidos de E/S.
    b) Un reloj del sistema.
    c) Interrupciones.
    d) Instrucciones hardware especiales de protección.
12. En el interbloqueo, la estrategia que puede dar lugar a una muy baja utilización de recursos es
    a) Estrategia liberal.
    b) Estrategia de prevención.
    c) Estrategia de detección y recuperación.
    d) Estrategia de mutua exclusión.
13. El algoritmo de la segunda oportunidad o NRU es una política de
    a) detección
    b) sustitución
    c) traducción
    d) ubicación
14. Un sistema de tiempo compartido es un caso particular de:
    a) Sistema multiprogramado
    b) Sistema monousuario
    c) Sistema de procesamiento por lotes
    d) Sistema monotarea

15. En UNIX, los privilegios de acceso a un archivo se guardan en
    a) El i-node
    b) La U-area
    c) El X-file
    d) El Z-buffer
    e) Ninguna de las anteriores
16. En el tratamiento del interbloqueo, el método de ordenación lineal de recursos actúa sobre la condición de:
    a) No apropiación
    b) Retención y espera
    c) Espera circular
    d) Exclusión mutua
17. La interface entre los procesos de usuario y el sistema operativo está definido por:
    a) Memoria compartida.
    b) Llamadas de sistema.
    c) El shell.
    d) Librerías de alto nivel.
18. ¿Cuál de las siguientes funciones la realiza el scheduler?
    a) Cambio a modo usuario.
    b) Cambio de contexto.
    c) Selección del siguiente programa de la lista de "listos para ejecutar".
    d) Poner el contador de programa en la posición adecuada del programa a ejecutar
19. El algoritmo de Dekker…
    a) Asegura la exclusión mutua de procesos sin soporte especial del hardware.
    b) Elimina la espera activa de dos procesos concurrentes.
    c) Asegura la exclusión mutua de procesos con soporte especial del hardware y/o zonas protegidas de memoria.
    d) Dos de las anteriores
    e) Ninguna de las anteriores

    ***) Ver Nota al final**
20. Un proceso A escribe una serie de datos (por ejemplo, un array de enteros…) en un área de memoria y n procesos deben imprimir con una determinada frecuencia (por ejemplo, cada un segundo) la última serie de datos escrita por el proceso A. ¿Cómo se puede asegurar exclusión mutua?
    a) No se necesita asegurar exclusión mutua ya que sólo un proceso es el que escribe.
    b) Se puede solucionar con el algoritmo de Dekker.
    c) Se puede solucionar con un semáforo.
    d) Se necesita por lo menos n semáforos.
    e) Dos de las anteriores

21. Si un proceso utiliza todo su quantum en la mayoría de los casos, podemos decir de él que:
    a) Nada.
    b) Es un proceso con mucha E/S.
    c) Es un proceso orientado a cálculo.
    d) Es una tarea del sistema.
    e) Es una tarea con alta prioridad
22. Cuando un proceso en modo usuario intenta ejecutar una instrucción privilegiada, ocurre:
    a) una interrupción
    b) una excepción
    c) una llamada al sistema
    d) una llamada al scheduler
23. De las siguientes operaciones, la que menos tiempo ha de consumir es:
    a) traducción de una dirección lógica a dirección física
    b) cambio de contexto
    c) detección de interbloqueo
    d) gestión de un fallo de página
24. Cuando un hilo se bloquea:
    a) Se bloquean también el resto de los hilos de la misma tarea si la implementación de los hilos se realizó a nivel de usuario.
    b) Se bloquean todos los hilos de ese usuario si los hilos se implementaron a nivel de usuario.
    c) Los demás hilos podrán seguir ejecutándose si los hilos se implementarón a nivel de usuario.
    d) Los demás hilos seguirán o no ejecutándose sin que importe la forma en que se llevó a cabo la implementación de los mismos.
25. De los servicios que se citan a continuación, ¿cuáles no deberían ser ofrecidos por un microkernel?
    a) Ocultamiento de interrupciones.
    b) Llamadas al sistema para suministrar acceso básico al sistema de archivos
    c) Herramientas para la comunicación entre procesos.
    d) Gestión básica de procesos.
26. La técnica de “swapping” permite:
    a) Aumentar el tamaño de los procesos que los usuarios pueden ejecutar.
    b) Aumentar la rapidez con que se ejecutan los procesos de usuario.
    c) Las respuestas a y b simultáneamente.
    d) Aumentar el nivel de multiprogramación del sistema.
27. De las diferentes técnicas de E/S, ¿cuál libera de más trabajo de E/S a la CPU?
    a) polling
    b) interrupciones
    c) DMA
    d) Las respuestas a y c.
28. Supongamos que en un sistema multiprogramado deseamos ejecutar 20 tareas intensivas en cálculo y todas de la misma duración. Si todas son lanzadas en t=0, los tiempos medios de retorno:

a) Son menores en sistemas con planificación FIFO

b) Son menores en sistemas round-robin.

c) Son iguales tanto en el caso a) como en el b)

d) Tanto en el caso a como en el b depende de las prioridades de las tareas

29. Para favorecer a los trabajos con largas ráfagas de CPU se debe implementar un algoritmo de planificación:

a) FCFS

b) SJF

c) RR

d) Basado en prioridades

30. Sea un sistema de memoria virtual paginada con páginas de 64kB. La memoria principal es de 1 GB. La Tabla de Páginas, que ocupa 512 KB, cuenta con descriptores de 16 bytes. ¿Cuál es el espacio de memoria virtual máximo disponible?

a) 1 GB.

b) 2 GB.

c) 4 GB.

d) Un valor distinto de los anteriores.

*) Para la pregunta nº 19 (del juego 1) la respuesta válida es la a) sin embargo el Tribunal consideró otorgar también 3 puntos a la opción d) y a no contestarla.

Juego 2

c b b a a c b c a c c a c d d a a b c d c b a a a d c a a b

Juego 3

c b a a a d c a a b c b b a a c b c a c c a c d d a a b c d

Juego 4

c b c a c c a c d d a a b c d c b a a a d c a a b c b b a a

Ejercicio 1

Se desea construir un `linker` que deberá leer un archivo que contiene uno o más módulos, cada uno formado de la siguiente manera:

Tipo de registro	Cantidad	Contenido
`header`	1	nombre de módulo
`extern`	0 a *n*	un símbolo y una dirección
`public`	0 a *m*	un símbolo y una dirección
`codigo`	1 a *l*	un largo y un código
`final`	1	un símbolo

Al final del archivo hay un registro con el siguiente formato:

`final_archivo`	1	un símbolo

Cada registro `extern` apunta a una palabra que deberá contener la dirección de un símbolo definido en otro módulo como `public`.

Cada registro `public` define la dirección de un símbolo, relativa al comienzo del módulo.

El registro de tipo `final_archivo` indica que la ejecución de este programa debe comenzar en la dirección de carga de este símbolo.

Se pide:

- Implementar dicho linker.

Ejercicio 2

La embotelladora Algarrobo Cola instaló una nueva planta y como lanzamiento quiere lanzar una oferta de un pack de 6 refrescos.

El departamento de marketing definió que el pack debe armarse en el siguiente orden: 2 botellas de bebida cola, 1 botella de pomelo, 1 botella de naranja y finalmente 2 botellas de pomelo.

Para esto la planta dispone de n_c, n_p y n_n cintas, cada una con dos brazos mecánicos que toman una botella de la cinta y la coloca en la empacadora en el orden indicado.

Para esto, cada brazo debe llamar cuando corresponda a los siguientes procedimientos:

- `tomar_botella_cinta`: toma la botella de la cinta para ponerla en la empacadora. No puede activarse para ambos brazos de una cinta al mismo tiempo. Debe tomarse la siguiente botella tan pronto haya dejado en la empacadora la botella anterior.
- `colocar_botella_empacadora`: deja la botella en la empacadora. Deberá colocarse respetando el orden definido anteriormente. Contemple que solo se puede colocar una botella en la empacadora por vez.

Se pide implementar utilizando semáforos las tareas:

- `BrazoCola(nro_cinta: in integer);`
- `BrazoPomelo(nro_cinta: in integer);`
- `BrazoNaranja(nro_cinta: in integer);`

Se deberá también definir las estructuras globales utilizadas y su inicialización.

Observaciones:

- Cada instancia de una tarea, representa a un brazo mecánico.
- Siempre hay suficiente cantidad de botellas de cada tipo para que sea posible formar un pack.
- No se puede implementar tareas auxiliares.
- El programa principal debe incluir el siguiente código:

```
cobegin
 BrazoCola(1);
 BrazoCola(1);
 BrazoCola(2);
 BrazoCola(2);
      ...
 BrazoCola(nc);
 BrazoCola(nc);
 BrazoPomelo(1);
 BrazoPomelo(1);
      ...
 BrazoPomelo(np);
 BrazoPomelo(np);
 BrazoNaranja(1);
 BrazoNaranja(1);
      ...
 BrazoNaranja(nn);
 BrazoNaranja(nn);
coend
```

Examen Febrero de 2002

Ejercicio 1

Se desea construir un `linker` que deberá leer un archivo que contiene uno o más módulos, cada uno formado de la siguiente manera:

Tipo de registro	Cantidad	Contenido
`header`	1	nombre de módulo
`extern`	0 a *n*	un símbolo y una dirección
`public`	0 a *m*	un símbolo y una dirección
`codigo`	1 a *l*	un largo y un código
`final`	1	un símbolo

Al final del archivo hay un registro con el siguiente formato:

`final_archivo`	1	un símbolo

Cada registro `extern` apunta a una palabra que deberá contener la dirección de un símbolo definido en otro módulo como `public`.

Cada registro `public` define la dirección de un símbolo, relativa al comienzo del módulo.

El registro de tipo `final_archivo` indica que la ejecución de este programa debe comenzar en la dirección de carga de este símbolo.

Se pide:

- Implementar dicho linker.

==========

Ejemplo de modulos :

```
          PUBLIC                  EXTERN
         ----------              ----------
          AQUI,y                  AQUI,x


         Modulo 1                Modulo 2
          -----                   ------
          -----                   ------
          -----                   ------
 y-> AQUI: -----                  JMP AQUI    <-x
          -----                   ------
```

En tabla Public está la dirección del símbolo, en la extern dice en que lugar hay un hueco que hay que sustituir por la dirección del símbolo (o sea en la dirección x hay que poner y).

Suponemos que la salida del linker va a un cargador dinámico, con direccionamiento por hardware

Entonces el linker será:

Recorre todos los registros Extern y forma una tabla con ellos.

Recorre todos los registros Public de los módulos y forma una tabla con ellos.

Forma un archivo de código con todos los registros de código.

Recorre la tabla Extern formada, busca en la tabla Public la dirección que corresponde a cada símbolo y sustituye en el archivo de código.

```
procedure LINKER (entrada: arch_linker, var salida: arch_load);
  type simb : record
                simbolo : char;
                direccion : tipo_dir;
                sig : simbolo;
              end;
  var extern,public : ^simb;
      dir_act : tipo_dir;
begin
  dir_act := 0;
  para cada modulo de entrada, mod_ent
    para cada registro Extern de mod_ent, reg_ext
      Agregar(extern, reg_ext.simbolo,
              reg_ext.direccion + dir_inicial)
      {agrega a la lista extern un registro con los valores
       indicados}
    para cada registro Public de mod_ent, reg_pub
      Agregar(Public,reg_pub.simbolo,
              reg_pub.direccion + dir_inicial)
      {agrega a la lista public un registro con los valores
       indicados}
    para cada registro Codigo de mod_end, reg_cod
      Agregar(Salida,reg_cod.codigo)
      {agrega al archivo de salida el codigo}
      dir_act := dir_act + reg_cod.largo
      {acumulo para proxima dir_act del siguiente modulo}
  fin para;
  para cada elemento de la lista Extern, nodo_ext
    buscar en lista Public el símbolo
    si no existe => error
    sino=>ir a la dirección apuntada
         reemplazar por la dirección de lista Public encontrada
end; {fin LINKER}
```

PREGUNTAR QUE PASA CON EL REGISTRO FINAL : PARA CADA MODULO NO ME IMPORTA AHORA,QUIERO EL PRIMERO SOLAMENTE.

==========

Ejercicio 2

La embotelladora Algarrobo Cola instaló una nueva planta y como lanzamiento quiere lanzar una oferta de un pack de 6 refrescos.

El departamento de marketing definió que el pack debe armarse en el siguiente orden: 2 botellas de bebida cola, 1 botella de pomelo, 1 botella de naranja y finalmente 2 botellas de pomelo.

Para esto la planta dispone de n_c, n_p y n_n cintas, cada una con dos brazos mecánicos que toman una botella de la cinta y la coloca en la empacadora en el orden indicado.

Para esto, cada brazo debe llamar cuando corresponda a los siguientes procedimientos:

- `tomar_botella_cinta`: toma la botella de la cinta para ponerla en la empacadora. No puede activarse para ambos brazos de una cinta al mismo tiempo. Debe tomarse la siguiente botella tan pronto haya dejado en la empacadora la botella anterior.
- `colocar_botella_empacadora`: deja la botella en la empacadora. Deberá colocarse respetando el orden definido anteriormente. Contemple que solo se puede colocar una botella en la empacadora por vez.

Se pide implementar utilizando semáforos las tareas:

- `BrazoCola(nro_cinta: in integer);`
- `BrazoPomelo(nro_cinta: in integer);`
- `BrazoNaranja(nro_cinta: in integer);`

Se deberá también definir las estructuras globales utilizadas y su inicialización.

Observaciones:

- Cada instancia de una tarea, representa a un brazo mecánico.
- Siempre hay suficiente cantidad de botellas de cada tipo para que sea posible formar un pack.
- No se puede implementar tareas auxiliares.
- El programa principal debe incluir el siguiente código:

```
cobegin
 BrazoCola(1);
 BrazoCola(1);
 BrazoCola(2);
 BrazoCola(2);
      ...
 BrazoCola(nc);
 BrazoCola(nc);
 BrazoPomelo(1);
 BrazoPomelo(1);
      ...
 BrazoPomelo(np);
 BrazoPomelo(np);
 BrazoNaranja(1);
 BrazoNaranja(1);
      ...
 BrazoNaranja(nn);
 BrazoNaranja(nn);
coend
```

==========

Solución #1

```
procedure BrazoCola(nro_cinta: in integer)
begin
  loop
     P(mutex_cinta_COLA[nro_cinta]);
     tomar_botella_cinta();
     V(mutex_cinta_COLA[nro_cinta]);

     P(COLA);
     P(mutex_COLA);
     count_COLA := count_COLA + 1;
     colocar_botella_empacadora();
     if (count_COLA = 2) then
       V(POMELO);
     end if
     V(mutex_COLA);
   end loop
end { BrazoCola }

procedure BrazoPomelo(nro_cinta: in integer)
begin
  loop
     P(mutex_cinta_POMELO[nro_cinta]);
     tomar_botella_cinta();
     V(mutex_cinta_POMELO[nro_cinta]);

     P(POMELO);
     P(mutex_POMELO);
     count_POMELO := count_POMELO + 1;
     colocar_botella_empacadora();
     if (count_POMELO = 1) then
       V(NARANJA);
     else if (count_POMELO = 3) then
       count_COLA := 0;
                    -- No necesito P(mutex_COLA) porque los
                    -- BrazoCola están bloqueados por
                    -- P(COLA), y BrazoPomelo está protegido
                    -- por P(POMELO).
       count_POMELO := 0;
       V(COLA);
       V(COLA);
     end if
     V(mutex_POMELO);
   end loop
end { BrazoPomelo }
```

```
procedure BrazoNaranja(nro_cinta: in integer)
begin
  loop
     P(mutex_cinta_NARANJA[nro_cinta]);
     tomar_botella_cinta();
     V(mutex_cinta_NARANJA[nro_cinta]);

     P(NARANJA);
     colocar_botella_empacadora();
     V(POMELO);
     V(POMELO);
   end loop
end { BrazoNaranja }
```

Variables globales:

```
var
  count_COLA, count_POMELO : integer;
  mutex_COLA, mutex_POMELO : semaphore;
  COLA, NARANJA, POMELO : semaphore;
  mutex_cinta_COLA    : array[1 .. n_c] of semaphore;
  mutex_cinta_NARANJA : array[1 .. n_n] of semaphore;
  mutex_cinta_POMELO  : array[1 .. n_p] of semaphore;
```

Inicialización de variables globales:

```
  count_COLA := 0;
  count_POMELO := 0;

  init(mutex_COLA, 1);
  init(mutex_POMELO, 1);
  init(COLA, 2);
  init(NARANJA, 0);
  init(POMELO, 0);

  for idx=1 to n_c do
    init(mutex_cinta_COLA[idx], 1);
  end for;

  for idx=1 to n_n do
    init(mutex_cinta_NARANJA[idx], 1);
  end for;

  for idx=1 to n_p do
    init(mutex_cinta_POMELO[idx], 1);
  end for;
```

Solución #2

```
procedure BrazoCola(nro_cinta: in integer)
begin
  loop
     P(mutex_cinta_COLA[nro_cinta]);
     tomar_botella_cinta();
     V(mutex_cinta_COLA[nro_cinta]);

     P(COLA);
     count_COLA := count_COLA + 1;
     colocar_botella_empacadora();
     if (count_COLA = 1) then
       V(COLA);
     else { if (count_COLA = 2) then }
       count_COLA := 0;
       V(POMELO);
     end if
   end loop
end { BrazoCola }

procedure BrazoPomelo(nro_cinta: in integer)
begin
  loop
     P(mutex_cinta_POMELO[nro_cinta]);
     tomar_botella_cinta();
     V(mutex_cinta_POMELO[nro_cinta]);

     P(POMELO);
     count_POMELO := count_POMELO + 1;
     colocar_botella_empacadora();
     if (count_POMELO = 1) then
       V(NARANJA);
     else if (count_POMELO = 2) then
       V(POMELO);
     else { if (count_POMELO = 3) then }
       count_POMELO := 0;
       V(COLA);
     end if
   end loop
end { BrazoPomelo }

procedure BrazoNaranja(nro_cinta: in integer)
begin
  loop
     P(mutex_cinta_NARANJA[nro_cinta]);
     tomar_botella_cinta();
     V(mutex_cinta_NARANJA[nro_cinta]);

     P(NARANJA);
     colocar_botella_empacadora();
     V(POMELO);
   end loop
end { BrazoNaranja }
```

Variables globales:

```
var
  count_COLA, count_POMELO : integer;
  COLA, NARANJA, POMELO : semaphore;
  mutex_cinta_COLA    : array[1 .. nc] of semaphore;
  mutex_cinta_NARANJA : array[1 .. nn] of semaphore;
  mutex_cinta_POMELO  : array[1 .. np] of semaphore;
```

Inicialización de variables globales:

```
  count_COLA := 0;
  count_POMELO := 0;

  init(COLA, 1);
  init(NARANJA, 0);
  init(POMELO, 0);

  for idx=1 to nc do
    init(mutex_cinta_COLA[idx], 1);
  end for;

  for idx=1 to nn do
    init(mutex_cinta_NARANJA[idx], 1);
  end for;

  for idx=1 to np do
    init(mutex_cinta_POMELO[idx], 1);
  end for;
```

==========