

Sistemas Operativos I

Examen final - 5 de Junio de 2002

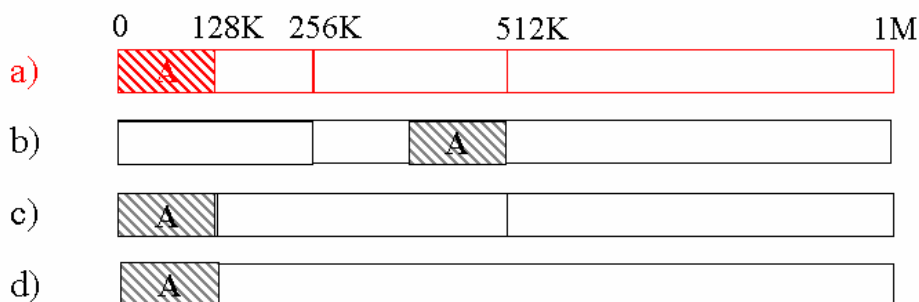
Modelo 444

TEORÍA (5 puntos)

Se proponen 12 preguntas tipo test. Cada pregunta tiene cuatro opciones, de las cuales sólo una es válida. Cada pregunta acertada valdrá 5/12 de punto. Cada fallo restará un tercio del valor asignado a la pregunta. Una pregunta no contestada valdrá 0 puntos.

1. ¿Cuál de las siguientes condiciones no se cumple en un interbloqueo?
 - a) Sólo un proceso puede usar un recurso cada vez.
 - b) Un proceso puede retener algunos recursos mientras espera que se le asignen otros que ha solicitado.
 - c) Los procesos se pueden ver forzados a abandonar recursos retenidos.
 - d) Un círculo vicioso de espera.
2. ¿Cuál de los siguientes elementos no interviene en el cálculo de parámetros de rendimiento del disco?
 - a) Retardo de giro
 - b) Tiempo de búsqueda
 - c) Tiempo de transferencia
 - d) Tiempo de planificación de acceso
3. Sea un sistema de asignación de memoria por particiones dinámicas que utiliza el método de los asociados. La memoria tiene un tamaño de 1Mbyte. Se produce la siguiente secuencia de cargas y descargas (finalización) de procesos:
T=0: Se carga un proceso A de tamaño 100 Kbytes
T=1: Se carga un proceso B de tamaño 260 Kbytes
T=2: Se carga un proceso C de tamaño 200 Kbytes
T=4: Se carga un proceso D de tamaño 50 Kbytes
T=5: Termina el proceso B
T=6: Termina el proceso C
T=7: Termina el proceso D

En este momento (después de que termine D), y suponiendo que a) en caso de que existan varios huecos de igual tamaño entre los que elegir, siempre se asigna el que tiene una dirección de comienzo menor y b) que después de que un proceso termine su ejecución siempre que se pueda se fusionan huecos (siguiendo la metodología de los asociados), di cómo será el mapa de memoria:



NOTA: Los espacios asignados a procesos están representados con sombreado, los huecos mediante rectángulos blancos.

4. Se tiene un array de dos dimensiones compuesto por enteros que ocupan dos bytes en memoria. El array está declarado del modo siguiente:

```
static int V[64][64];
```

donde V[0][0] está en la posición cero de la memoria virtual, y a continuación están V[0][1], V[0][2], etc. El sistema de paginación emplea páginas de 128 bytes, y sólo hay dos marcos disponibles para alojar las páginas de este proceso. Suponiendo que el algoritmo de remplazo de páginas es el óptimo, di cuántos fallos de página se producirán en cada uno de los dos siguientes fragmentos de código:

Opción 1:

```
for (i=0; i<64; i++)  
    for (j=0; j<64; j++)  
        V[i][j]=0;
```

Opción 2:

```
for (j=0; j<64; j++)  
    for (i=0; i<64; i++)  
        V[i][j]=0;
```

- | | |
|------------------------|-----------------------|
| a) <u>Opción 1: 64</u> | <u>Opción 2: 4096</u> |
| b) Opción 1: 2 | Opción 2: 2 |
| c) Opción 1: 32 | Opción 2: 2048 |
| d) Opción 1: 4096 | Opción 2: 64 |

5. El procesador del ordenador Alpha proporciona paginación multinivel de tres o de cuatro niveles, y varios tamaños de página: 8K, 16K, 32K y 64K. Suponiendo que el tamaño máximo de las direcciones virtuales es de 64 bits, el tamaño de cada entrada de la tabla de páginas es de 64 bits y el tamaño máximo de cualquier tabla (o subtabla) de páginas de cualquier nivel es de 1 página, di cuántos bits útiles tendrán las direcciones virtuales del sistema en una configuración que utilice 4 niveles de tabla de páginas y un tamaño de página de 8K.

- a) 58
- b) 63
- c) 53
- d) 64

6. Sea un fichero de almacenado en un sistema de archivos que utiliza bloques de tamaño fijo iguales a 1K (1024 bytes). El fichero acaba de abrirse (el offset es el inicial, es decir 0) y por lo tanto no existe ningún bloque del disco en la memoria. Se quiere hacer una lectura en el byte 3500 del fichero. Di cuál de las siguientes afirmaciones es falsa.

- a) El número de accesos a disco (es decir, lecturas de bloques de disco a memoria) si el sistema de archivos utiliza indexación mediante un único bloque índice es **2** (supóngase que el bloque índice no está ya en la memoria)
- b) El número de accesos a disco necesarios si el sistema utiliza una tabla enlazada tipo FAT (que ya está en memoria en el momento de realizar esta lectura), es **1**
- c) El número de accesos a disco necesarios en un sistema indexado mediante i-nodos, donde los inodos contienen 5 apuntadores a bloques directos y un apuntador indirecto doble es **1**. (Supóngase que el inodo está ya copiado en la memoria en el momento de la lectura).
- d) El número de accesos a disco, si la asignación de bloques es contigua es igual a 4.

7. ¿Cuál de las siguientes afirmaciones es falsa con respecto a los sistemas de asignación de memoria para procesos?

- a) La asignación mediante particiones estáticas (fijas) puede producir fragmentación interna y externa
- b) La asignación mediante particiones dinámicas produce fragmentación externa pero no fragmentación interna.
- c) La asignación mediante paginación produce fragmentación interna cuyo valor medio por proceso es igual a la mitad del tamaño de una página, y no produce fragmentación externa.
- d) La segmentación pura puede causar fragmentación externa, pero no fragmentación interna.

8. En el algoritmo FCFS (primero en llegar, primero en servirse) en la planificación de procesos tiene como efecto:

- a) los procesos largos se ven penalizados.
- b) favorece a los procesos con carga de E/S.
- c) favorece a los procesos cortos y a los que no tienen carga de E/S.
- d) penaliza a los procesos cortos y a los que tienen carga de E/S.

9. La detección de interbloqueos tiene como característica propia que la diferencia de la predicción y la prevención:

- a) Facilita la gestión en línea.
- b) Puede retrasar el inicio de algún proceso.
- c) No hace falta procesamiento durante la ejecución.
- d) No es necesaria la apropiación.

10. Un proceso contiene 8 páginas virtuales en disco y se asignan con una ubicación fija de **tres** marcos de página en la memoria principal. Inicialmente los tres marcos asignados al proceso se encuentran vacíos. Se produce la siguiente serie de accesos a páginas:

1,0,2,2,1,7,6,7,0,1,2,0

Suponiendo que se utiliza un algoritmo de reemplazo de páginas LRU (página usada hace más tiempo), dí cuál es el número de fallos de página que se producirá en la ejecución de la secuencia de páginas anterior.

- a) 10
- b) 5
- c) 8
- d) 0

11. Sean dos procesos A y B. Los dos procesos comparten una zona de memoria compartida compuesta por las siguientes variables:

```
/* Compartido por los procesos */
int X=0;
semaforo Y=0;
semaforo Z=10;
semaforo M=1;
```

Proceso A	Proceso B
<pre>main() { int i=30; while(i) { down(Y); down(M); X--; --i; if(i==0) printf("%\d\t",X); up(Z); up(M); } }</pre>	<pre>main() { int j=15; while(j) { down(Z); down(M); X++; --j; if(j==0) printf("%\d\t",X); up(Y); up(M); } }</pre>

Suponiendo que los dos procesos se ejecutan de manera concurrente, y sin presuponer ningún orden particular en la ejecución de los procesos, di cuál será el valor impreso en la salida estandar del sistema.

- a) 0 -15
b) Un número entre 1 y 10
c) Dos números cualesquiera entre 1 y 10.
d) 0 10
12. En el sistema de archivos nativo de UNIX, los i-nodos contienen 10 apuntadores directos a bloques, 1 apuntador indirecto sencillo, 1 apuntador indirecto doble y 1 apuntador indirecto triple. Dí cual es el tamaño máximo posible de un fichero si el tamaño de un bloque en disco es de 1024 bytes y cada apuntador a bloques físicos de disco ocupa 32 bits.
- a) 10240 bytes
b) 68989001728 bytes
c) 13312 bytes
d) 17247250432 bytes

Sistemas Operativos I

Examen final - 5 de Junio de 2002

Problemas (5 puntos)

A continuación se proponen tres problemas y una pregunta adicional. Debes responder cada pregunta en una hoja distinta. La duración del examen será de tres horas.

Lee **BIEN** los enunciados, y suerte!

Problema 1 (1.5 pts)

Escribe una función **crea_hijos**(int n) en C. El proceso llamante va a crear un **máximo de n** procesos hijo. Cada vez que cree un hijo, comprobará el identificador de proceso (PID) **del proceso creado**. Si el PID del proceso creado es un número impar, dejará de crear hijos y esperará la terminación de sus hijos, tal como se describe más adelante. Si el identificador del hijo creado es un número par, el proceso continuará creando hijos hasta que 1) haya creado **n hijos**, ó 2) uno de los hijos creados tenga un PID impar. Cada proceso hijo creado volverá a hacer lo mismo, es decir, creará hijos hasta que uno de los hijos creados tenga un PID impar o llegue a generar **n hijos**. Esto se repite hasta crear la **séptima (7)** generación de procesos (considera el proceso llamante como la primera generación). Los procesos de séptima generación terminan sin generar hijos.

Cada proceso que ha creado procesos hijo debe ponerse en espera hasta que todos sus hijos terminen. Antes de terminar, la función deberá devolver el número total de procesos creados durante la ejecución de la misma. Para eso, cada proceso antes de salir debe devolver a su progenitor el número (acumulado) de procesos de generaciones inferiores que provienen de él. Esto lo harán a través del parámetro de retorno del **exit()**.

NOTA 1: Aunque este dato no debes utilizarlo, ya que la solución debe hacerse en función del parámetro **n**, se entiende que en un caso real **n** deberá ser siempre un número menor o igual a 3 para que los valores acumulados de número de hijos puedan enviarse a través del valor de retorno del **exit**.

NOTA 2: No se considerarán soluciones que empleen llamadas recursivas a **crea_hijos** o a cualquier otra función equivalente.

NOTA 3: No se considerarán soluciones que empleen llamadas recursivas a **crea_hijos** o a cualquier otra función equivalente.

/* ESTA ES UNA POSIBLE SOLUCIÓN AL PROBLEMA, QUE CUMPLE TODAS LAS ESPECIFICACIONES DEL ENUNCIADO. EVIDENTEMENTE, NO ES LA ÚNICA POSIBLE */

```
crea_hijos(int n) {

register int i;
int fid;                /* valor de retorno del fork() */
int hijos_creados=0;    /* lleva cuenta del número de hijos creados */
int generacion=0;       /* contador de generaciones */
int hijos_retorno=0;    /* número de hijos que se deben esperar */
int status;             /* valor de retorno utilizado en el wait() */
int error=0;            /* condición de error en la terminación de los procesos de menor */
                        /* generación */

while(++generacion < 10) {
    hijos_creados=0;
    for(i=0;i<n;++i) {
        if((fid = fork()) == -1) printf("Error al crear proceso\n");
        else if(fid ==0) break;
        else {
            ++hijos_creados;
            if((fid %2)!=0) break; /* El identificador de proceso hijo es un número par */
        }
    }
    if(fid !=0) break;
} /* fin del while que controla el número de generaciones de procesos */

if(generacion==10) exit(0); /* Los procesos de generación 10 salen */

hijos_retorno=hijos_creados;
while(hijos_creados) { /* bucle de espera a la terminación de procesos hijo */
    wait(&status);
    if(WIFEXITED(status)) {
        /* Si hay error en la salida activo un código de error que se transmitirá al padre */
        if(WEXITSTATUS(status)== -1) error=1;
        /* Sumo a hijos_retorno el número de descendientes del proceso que termina */
        else hijos_retorno += WEXITSTATUS(status);
    } else error=1;
    --hijos_creados;
}
if(error) hijos_retorno = -1;
if(generacion!=1) exit(hijos_retorno);
else return(hijos_retorno);
}
```

Problema 2 (1.5 ptos)

Dibuja la estructura de un manejador de memoria virtual que combina segmentación y paginación. Las características del sistema son las siguientes:

- Tamaño de direcciones virtuales: 32 bits
- Número de segmentos: 512
- Tamaño de página: 4 K
- Tamaño de las entradas de la tabla de páginas: 4 bytes
- Tamaño de la memoria física: 1 Gbyte

Deberás dibujar un esquema comentado que contenga las siguientes partes:

- formato de la dirección virtual (DV)
- tabla de segmentos (TS): formato de la entrada de la tabla de segmentos (ETS), número de entradas de la tabla de segmentos
- tabla de páginas (TP): formato de la entrada de la tabla de páginas (ETP), número de entradas de la misma de páginas (elige un algoritmo de reemplazo de páginas e incluye en la entrada la información que necesites para implementarlo).
- Sobre el esquema anterior debes indicar como se realizaría la traducción de las direcciones virtuales.

SOLUCIÓN:

(COMO EN EL CASO ANTERIOR, ESTA SOLUCIÓN NO ES LA ÚNICA BUENA, AUNQUE CUMPLE LAS ESPECIFICACIONES DEL ENUNCIADO)

Para el cálculo de las partes que componen la DV:

- Número de segmentos: $512=2^9$, por lo tanto la dirección tendrá reservados 9 bits para el número de segmento
- Tamaño de página $4K=2^{12}$, por lo tanto la dirección tendrá reservados 12 bits para el offset dentro de cada página
- El número de bits reservados para el número de página sera el total menos los reservados para el número de segmento y el número de página, es decir ($32-9-12 = 11$)

Para el cálculo del formato de la ETS:

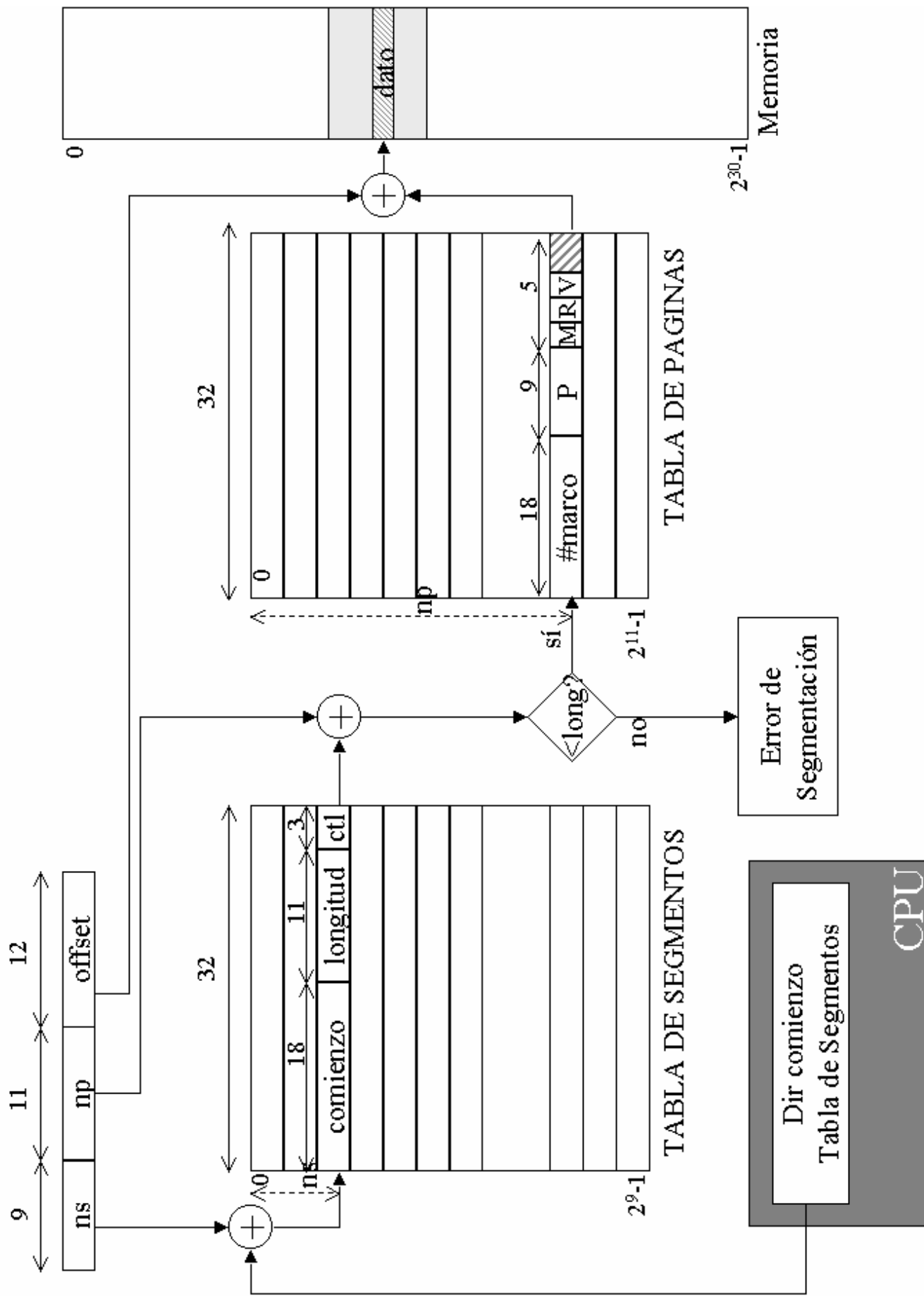
- Dirección de comienzo de la TP de cada segmento. Si la entrada contiene la dirección física de comienzo, la ETS deberá de tener reservados al menos 30 bits para almacenar esa dirección (debido a que la memoria física es de $1Gb=2^{30}$ bytes). Si lo que se almacena es el número de marco físico en el que comienza la tabla de páginas, entonces el número de bits deberá de poder hacer referencia a cualquier marco físico. El número de marcos físicos es igual al tamaño de la memoria física dividido por el tamaño de cada página (o marco). Como $2^{30}/2^{12} = 2^{18}$, harán falta al menos 18 bits para la dirección de comienzo.
- La longitud de un segmento, como máximo será del número máximo de páginas que pueden estar direccionadas 2^{11} , por lo que harán falta 11 bits para almacenar el tamaño del segmento.
- Otros bits de control. Si optamos por indicar el comienzo de la TP mediante un número de marco (18 bits), y necesitamos 11 bits para la longitud del segmento, nos quedan 3 bits para completar una ETS de 32 bits de longitud. Si hubiesemos elegido identificar el comienzo de la TP mediante una dirección física (30 bits), habríamos optado por una ETS de 64 bits, con lo que habrían quedado $64-32-11=21$ bits para los bits de control y un relleno no utilizado.

El número de ETS's será igual al número máximo de segmentos posibles, que es 512. Por lo tanto la TS tendrá 512 ETS's.

Para el cálculo del formato de la ETP:

- Marco físico asociado. Deberá de contener el número del marco físico que se corresponde con la ETP. Por lo tanto, como se ha calculado anteriormente, harán falta 18 bits para el número de marco.
- Bits de control. Además del bit de Validez (V) y los 9 bits de permiso (P) harán falta tantos como sean necesarios para implementar el algoritmo de remplazo de páginas. Si por ejemplo, se utiliza un NRU (página no recientemente usada), bastará con tener un bit de referencia (R) y un bit de modificación de página (M). Como nos dicen que las entradas tienen una longitud de 4 bytes (32 bits), tendremos un relleno de $32-18-9-1(V)-1(R)-1(M)=2$ bits.

El esquema de traducción de direcciones es por lo tanto el siguiente:



Problema 3 (2 ptos)

En una empresa hay un director, unos jefes de sección y un grupo de técnicos bajo la responsabilidad de cada jefe. Debido a que las secciones no tienen trabajo especializado, la forma de repartir tareas en esta empresa es la siguiente: el director tiene un tablón en la puerta en el que va escribiendo las tareas a realizar. Los jefes de sección visitan regularmente el tablón, escojen una tarea y la borran. A renglón seguido estos jefes utilizan el mismo sistema con los técnicos de tal forma que cuando llegan a su despacho escriben en el tablón la tarea que han borrado del tablón del director. Los técnicos de cada sección cada vez que acaban con una tarea necesitan encontrar otra a realizar. Para ello visitan el tablón del jefe, eligen una tarea y la borran del tablón.

Además, esta empresa tiene un sistema antiestrés que hace que cada jefe o director sepa el máximo número de tareas que puede poner en el tablón sin que sus subordinados sufran estrés. El director sabe que puede haber M tareas para los J jefes y cada jefe sabe que puede tener en su tablón N_i tareas (donde i indica el número identificativo del jefe correspondiente) para sus T_i técnicos.

Crear los procesos *director*, *jefe* y *tecnico* para que se lleven a cabo las tareas eficientemente.

El problema debe plantearse como un doble productor consumidor consecutivo

Push y pop se suponen implementadas con variables internas locales que identifican a los elementos a introducir o sacar. Al estar siempre dentro de secciones críticas no hay problemas de sincronización para esas variables.

```
semaforo mutexDirector = 1, mutexJefe[J]={1,1,1,1, ... , 1};
semaforo vaciosDirector = M, vaciosJefe[J]={N1, N2, N3 .... NJ};
semaforo llenosDirector = 0, llenosJefe[J]={0,0,0,...,0};
```

```
char *tablónDirector[M]; /* M elementos de tamaño variable */
char *tablónJefe[N1], *tablónJefe[N2], ..., *tablónJefe[NJ]; /* tablonés para cada jefe */
```

```
Director()
{
    while (cierto)
    {
        /* PRODUCTOR */
        pensar_nueva_tarea(); /*sección no crítica */
        down (vaciosDirector);
        down (mutexDirector); /* sección crítica */
        push(&tablónDirector);
        up(mutexDirector); /* fin sección crítica */
        up(llenosDirector);
    }
}
```

```
Jefe(i) /* El índice i identifica al jefe */
{
    while (cierto)
    {
        /* CONSUMIDOR */
        DecideIrPorTarea (); /*sección no crítica */

        down (llenosDirector);
        down (mutexDirector); /* sección crítica */
```

```

    pop(&tablonDirector);
    up (mutexDirector); /* fin sección crítica */
    up (vaciosDirector);

    /* PRODUCTOR */
    down (vaciosJefe[i]);
    down (mutexJefe[i]); /* sección crítica */
    push (&tablonJefe[i]);
    up (mutexJefe[i]); /* fin sección crítica */
    up (llenosJefe[i]);

    DescansaUnRatoDeLaArduaTarea(); /*sección no crítica */
}

Tecnico(i) /* La i identifica al jefe del técnico y por tanto a su tablón */
{
    while (cierto)
    {
        /* CONSUMIDOR */
        down (llenosJefe[i]);
        down (mutexJefe[i]); /* sección crítica */
        pop(&tablonJefe[i]);
        up (mutexJefe[i]); /* fin sección crítica */
        up (vaciosJefe[i]);

        RealizaTarea();
    }
}

```

Pregunta 4 (Opcional, vale 0.5 ptos sobre la nota total del examen)

Escribe un breve resumen (máximo 30 líneas) de uno de los tres artículos recomendados para el trabajo opcional de la asignatura. Indica los aspectos más importantes del mismo desde el punto de vista de la Historia de los Sistemas Operativos.