SISTEMAS OPERATIVOS
Solución PED2 (Enero 2014)

Solución Ejercicio 1

I) El grafo de asignación de recursos puede utilizarse para detectar la presencia de interbloqueos, para ello debe suponerse que ya se cumplen las condiciones de exclusión mutua y no existencia de expropiación, las cuales quedan fijadas por el sistema operativo. En conclusión la afirmación es **FALSA**.

II) La afirmación es **VERDADERA** ya que esta es una de las principales ventajas de la segmentación simple. Como cada parte del programa tiene asignado un segmento independiente, la protección y la compartición se realiza a nivel de segmento.

III) Esta afirmación es **VERDADERA** ya que efectivamente si se detecta la existencia de sobrepaginación, la forma de eliminarla es reduciendo el grado de multiprogramación del sistema. Para ello se deben intercambiar a memoria secundaria uno o varios procesos y repartir los marcos que ocupaban entre los procesos que estaban provocando la sobrepaginación. Esta operación deberá repetirse el número de veces que sea necesario hasta que se detenga la sobrepaginación.

IV) Esta afirmación es **FALSA** ya que es el subsistema de E/S el que se encarga de asignar buffers para el almacenamiento temporal de los datos que se leen o se escriben en las operaciones de E/S.

Solución Ejercicio 2

La técnica de paginación simple **no produce fragmentación externa**, solo produce *fragmentación interna* debida al posible espacio libre que puede existir en la última página del espacio de direcciones lógicas de un proceso.

1) Cálculo de la fragmentación interna del proceso A:

   Para calcular la fragmentación interna en primer lugar se va a calcular el número de páginas N_P en que se descompone el espacio de direcciones lógicas del proceso A:

$$N_P = \text{ceil}\left(\frac{C_X}{S_P}\right)$$

   Donde C_X es el tamaño del espacio de direcciones lógicas de proceso A y S_P es el tamaño de una página. Sustituyendo valores y operando se obtiene que:

$$N_P = \text{ceil}\left(\frac{31566}{2048}\right) = \text{ceil}\,(15{,}413) = 16\,\text{páginas}$$

   El espacio ocupado por estas 16 páginas ($i = 0, 1, 2, ..., 15$) es $16 \cdot 2048 = 32768$ bytes. Puesto que el espacio de direcciones lógicas del proceso A tiene un tamaño de 31566 bytes, el espacio libre existente en la última página asignada al proceso (página $i = 15$) es igual a:

$$32768 - 31566 = 1202 \text{ bytes}$$

   Por lo tanto, la fragmentación interna provocada por la carga del proceso A es de **1202 bytes**.

2) Cálculo de la fragmentación interna del proceso B:

$$N_P = \text{ceil}\left(\frac{18432}{2048}\right) = \text{ceil}\,(9) = 9\,\text{páginas}$$

   El espacio ocupado por estas 9 páginas ($i = 0, 1, 2, ..., 8$) es $9 \cdot 2048 = 18432$ bytes. Puesto que el espacio de direcciones lógicas del proceso A tiene un tamaño de 18432 bytes, el espacio libre existente en la última página asignada al proceso (página $i = 8$) es igual a:

$$18432 - 18432 = 0 \text{ bytes}$$

   Por lo tanto, la fragmentación interna provocada por la carga del proceso A es de **0 bytes**. Es decir, la carga de del proceso A no produce fragmentación interna.

Solución Ejercicio 3

Una dirección virtual se descompone en dos campos: número de página i y desplazamiento. Del tamaño de una página S_P expresado en palabras se puede determinar el tamaño d en bits del campo desplazamiento tanto de una dirección física como de una dirección virtual, para ello hay que resolver la siguiente desigualdad:

$$\min_d \left\{ S_P \leq 2^d \right\}$$

Del enunciado se sabe que $S_P = 2$ KiB $= 2^{11}$ bytes, dividiendo por la longitud de una palabra se obtiene el tamaño de una página expresado en palabras:

$$S_P = \frac{2^{11} \text{ bytes}}{1 \text{ (byte/palabra)}} = 2^{11} \text{ palabras}$$

Luego $d = 11$ bits.

i) En primer lugar se va a pasar la dirección virtual `0x1873` a binario:

```
0001 1000 0111 0011
```

Descomponiéndola en los campos número de página i y desplazamiento

```
00011 00001110011
```

se deduce que el número de página i expresado en decimal al que hace referencia esta dirección virtual es

$$i = 00011_2 = 3_{10}$$

A continuación se consulta la tabla de páginas y se comprueba el valor del bit de validez v. Como $v = 1$ significa que actualmente la página $i = 3$ está cargada en memoria principal, en concreto de acuerdo con la tabla, en el marco de página $j = 10$.

Pasando el número de marco a binario $j = 10_{10} = 01010_2$ se puede construir la dirección física en binario:

```
01010 00001110011
```

Que se puede expresar equivalentemente en la forma:

```
0101 0000 0111 0011
```

Finalmente pasando a hexadecimal se obtiene: **`0x5073`**

ii) En primer lugar se va a pasar la dirección virtual `0x2033` a binario:

```
0010 0000 0011 0011
```

Descomponiéndola en los campos número de página i y desplazamiento

```
00100 00000110011
```

se deduce que el número de página i expresado en decimal al que hace referencia esta dirección virtual es

$$i = 00100_2 = 4_{10}$$

A continuación se consulta la tabla de páginas y se comprueba el valor del bit de validez v. Como $v = 1$ significa que actualmente la página $i = 4$ está cargada en memoria principal, en concreto de acuerdo con la tabla, en el marco de página $j = 8$.

Pasando el número de marco a binario $j = 8_{10} = 01000_2$ se puede construir la dirección física en binario:

```
01000 00000110011
```

Que se puede expresar equivalentemente en la forma:

```
0100 0000 0011 0011
```

Finalmente pasando a hexadecimal se obtiene: **`0x4033`**

iii) En primer lugar se va a pasar la dirección virtual `0x1089` a binario:

```
0001 0000 1000 1001
```

Descomponiéndola en los campos número de página i y desplazamiento

```
00010 00010001001
```

se deduce que el número de página i expresado en decimal al que hace referencia esta dirección virtual es

$$i = 00010_2 = 2_{10}$$

A continuación se consulta la tabla de páginas y se comprueba el valor del bit de validez v. Como $v = 0$ significa que actualmente la página $i = 2$ no está cargada en memoria principal por lo que **se produce un fallo de página**.