PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

3ra práctica (tipo a)
(Primer semestre de 2019)

Horario 0781: prof. V. Khlebnikov
Horario 0782: prof. A. Bello R.

Duración: 1 h. 50 min.
Nota: No se puede usar ningún material de consulta.
La presentación, la ortografía y la gramática influirán en la calificación.
Puntaje total: 20 puntos

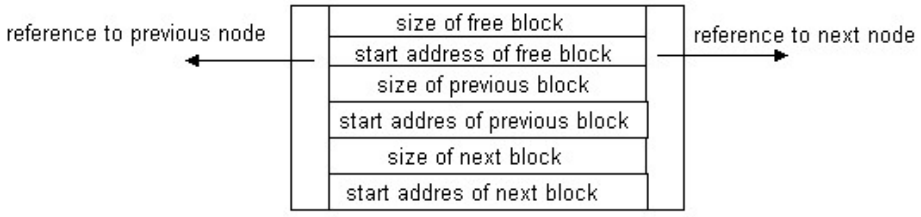
Pregunta 1 (3 puntos – 15 min.) Dados cinco huecos en memoria de 100 KB, 500 KB, 200 KB, 300 KB y 600 KB (en este orden desde el inicio de la memoria), ¿cómo asignarán la memoria los algoritmos *first-fit*, *next-fit*, *last-fit* (el algoritmo absolutamente simétrico al *first-fit*, y por eso, asigna **dentro del hueco** la memoria en sus direcciones **más altas** si la solicitud es menor que el tamaño del hueco), *best-fit*, *worst-fit* colocando los procesos de 212 KB, 417 KB, 112 KB y 426 KB (en este orden)? ¿Cuál de los algoritmos hace el uso más eficiente de memoria en este caso? ¿Qué sucedería con la posibilidad de condensación de todos los huecos en uno?

Pregunta 2 (4 puntos – 20 min.) Doubly-linked list implementation of free-list. In this implementation

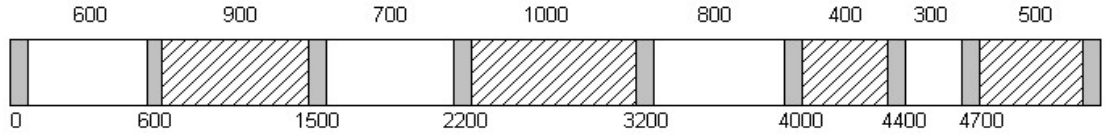
- List nodes are not sorted according to start addresses of free blocks.
- All memory blocks have boundary tags between them. The tag has information about the size and status (allocated/free).
- Each node in the doubly linked list represents a free block. It keeps size & start address of the free block and start addresses & sizes of the previous and next memory blocks. The adjacent blocks may be or may not be free.

The *release* operation does not combine adjacent free blocks. It simply prepends a node corresponding to a released block at the front of the free list. This operation is thus O(1).
Adjacent free blocks are combined by *acquire*(). The *acquire* operation traverses the free list in order to find a free area of a suitable size. As it does so it also combines adjacent free blocks.

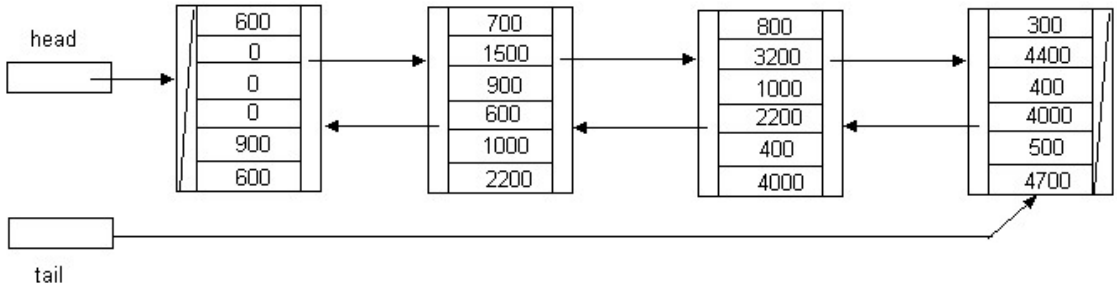
Node structure:



Initial state of memory (shaded = allocated, grayed = boundary tags):



The corresponding free list:



a) (1 punto – 5 min.) Presente la figura del estado de memoria después de realizar la operación *release*(400,4000).

b) (1 punto – 5 min.) ¿Cuál será la lista resultante de bloques libres? Presente solamente los datos modificados.

c) (1 punto – 5 min.) ¿Cuál serán la figura del estado de memoria y la lista de bloques libres después de la **1ra fase** (*traversing* & *merging*) de la operación *acquire*(600)? Presente solamente los cambios correspondientes. **Nota**: tome en consideración que el sistema operativo trabaja con la lista enlazada y no con la figura del estado de memoria. La última es solamente referencial e ilustrativa para una persona humana.

d) (1 punto – 5 min.) ¿Cuál serán la figura del estado de memoria y la lista de bloques libres después de la **2da fase** (*allocating*) de la operación *acquire*(600) si se usa la política de asignación *first-fit*? Presente solamente los cambios correspondientes.

Pregunta 3 (3 puntos – 15 min.) In this problem you are to compare the storage needed to keep track of free memory using a bitmap versus using a linked list of allocated and free memory segments. The 256-MB memory is allocated in units of *n* bytes. For the linked list, assume that memory consists of an alternating sequence of segments and holes, each 64 KB. Also assume that each node in the linked list needs a 32-bit memory address, a 16-bit length, and a 16-bit next-node field. How many ***bytes*** of storage is required for each method (Presente el cálculo.)? ¿En qué caso los tamaños del mapa de bits y de la lista enlazada serán iguales, y en qué caso la lista enlazada ocupará menos espacio que el mapa de bits?

Pregunta 4 (5 puntos – 25 min.) Un proceso tiene como máximo 64 páginas de memoria virtual. Además conocemos la tabla de paginas en tres momentos. En cada momento se hace referencia a tres direcciones físicas (d. f.) de forma correspondiente, es decir, la primera d. f. corresponde con la primera tabla, la segunda con la siguiente tabla, etc. A continuación las direcciones y las tablas:

Dirección Física : 1EF1h

Página	Marco	P/A	R	M
0	3h	1	0	1
1		0		
2		0		
3	5h	1	0	0
4		0		
5	Ah	1	0	0
6		0		
7	Fh	1	1	0

Dirección Física : 1554h

Página	Marco	P/A	R	M
0	3h	1	0	1
1		0		
2	Ah	1	1	1
3	5h	1	0	0
4		0		
5		0		
6		0		
7	Fh	1	1	0

Dirección Física : 0A33h

Página	Marco	P/A	R	M
0	3h	1	0	1
1		0		
2	Ah	1	1	1
3		0		
4	5h	1	1	0
5		0		
6		0		
7	Fh	1	1	0

Se le solicita

a) (3 puntos – 15 min.) La dirección lógica de la dirección física: 0A33h

b) (2 puntos – 10 min.) El espacio de direcciones virtuales del proceso (en bytes) y el tamaño máximo que puede tener la memoria física.

Pregunta 5 (5 puntos – 25 min.) Los 5 segmentos de un proceso son los siguientes (numerados de 0..4)

Segmento 0
Página 0
Página 1
Página 2

Segmento 1
Página 0
Página 1
Página 2

Segmento 2
Página 0
Página 1
Página 2
Página 3
Página 4

Segmento 3
Página 0
Página 1
Página 2
Página 3

Segmento 4
Página 0
Página 1
Página 2
Página 3
Página 4
Página 5

Inicialmente ninguna página está en memoria. Se generan las direcciones lógicas de la tabla y se conocen sus correspondientes direcciones físicas.

Direcciones Lógicas	Direcciones Físicas
00043h	5243h
80AFCh	70FCh
606B4h	58B4h
60464h	7064h
80693h	5893h

Determine:

a) (2 puntos – 10 min.) El número de marcos asignados al proceso y cuáles son estos.
b) (1 punto – 5 min.) El tamaño del marco de página.
c) (1 punto – 5 min.) El número máximo de páginas que puede tener un segmentos.
d) (1 punto – 5 min.) El tamaño máximo que puede tener la memoria real.

La práctica ha sido preparada por AB(4) y VK(1-3)
en Linux Mint 19.1 Tessa con LibreOffice Writer

Profesores del curso: (0781) V. Khlebnikov
(0782) A. Bello R.

Pando, 31 de mayo de 2019