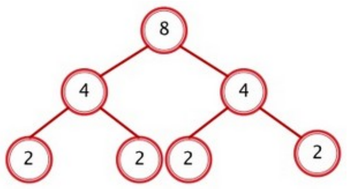
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
3ra práctica (tipo a)
(Segundo semestre de 2014)

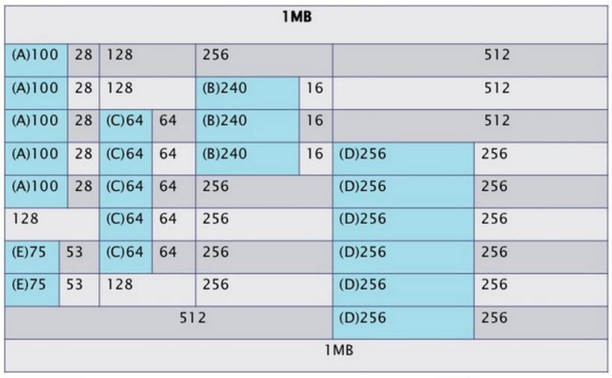
Horario 0781: prof. V. Khlebnikov

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 (12 puntos – 60 min.) (*Brodal, Demaine, Munro*) According to Donald Knuth, the buddy system was invented in 1963 by Harry Markowitz, who won the 1990 Nobel Memorial Prize in Economics. It was first described by Kenneth C. Knowlton (published 1965). There are number of buddy systems: binary, Fibonacci, weighted, tertiary.



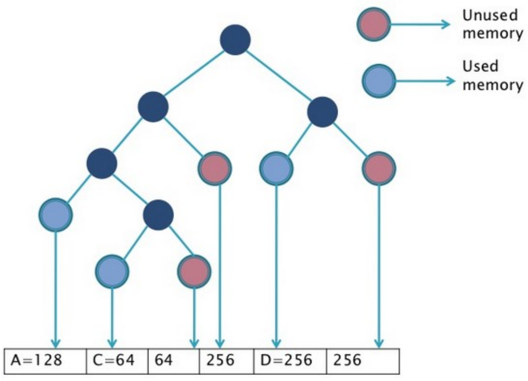
In binary buddy system the memory block of 2^m is into two equal parts of 2^{m-1}.
It satisfies the following recurrence relation: $L_i = L_{i-1} + L_{i-1}$.
It is crucial for performance purposes to know, given a block address, the size of the block and whether it is occupied. This is usually done by storing a *block header* in the first few bits of the block. More precisely, we use headers in which the first bit is the *occupied bit*, and the remaining bits specify the size of the block. Thus, for example, to determine whether the buddy of a block is free, we compute the buddy's address, look at the first bit at this address, and also check that the two sizes match. Because block sizes are always powers of two, we can just encode their logarithms in the block headers. This uses only lg lg n bits, where n is the number of (smallest) blocks that can be allocated. As a result, the smallest practical header of one byte long is sufficient to address up to $2^{128} \approx 3.4 \cdot 10^{38}$ blocks.



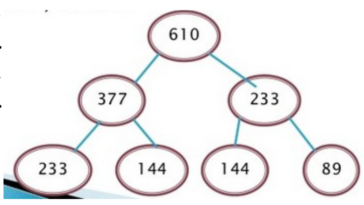
1MB							
(A)100	28	128		256		512	
(A)100	28	128		(B)240	16	512	
(A)100	28	(C)64	64	(B)240	16	512	
(A)100	28	(C)64	64	(B)240	16	(D)256	256
(A)100	28	(C)64	64	256		(D)256	256
128		(C)64	64	256		(D)256	256
(E)75	53	(C)64	64	256		(D)256	256
(E)75	53	128		256		(D)256	256
512						(D)256	256
1MB							

Let us consider 1-Mbyte of memory is allocated using binary buddy system, showing the binary tree and list form for the following: request 100K (A), request 240K (B), request 64K (C), request 256K (D), release B, release A, request 75K (E), release C, release E, release D.

a) (8 puntos – 40 min.) Indique las direcciones (en hexadecimal) de los bloques libres y los contenidos de sus *block headers* correspondientes al momento después de *release A*.



In Exercise 2.5.31 of his book, Knuth proposed the use of Fibonacci numbers as block sizes instead of power of two, resulting in the *Fibonacci buddy system*. This idea was detailed by Hirshberg, and was optimized by Hinds and Cranson and Thomas. Block sizes satisfy the following recurrence relation: $L_i = L_{i-1} + L_{i-2}$. The first 21 Fibonacci numbers are: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, …



b) (4 puntos – 20 min.) Muestre los resultados de uso del *Fibonacci buddy system* para las siguientes solicitudes: request 60 bytes (A), request 135 bytes (B), request 70 bytes (C), release A, request 60 bytes (D), release B, release D, release C.

Pregunta 2 **(8 puntos – 40 min.)** Considere el siguiente programa:

```
$ cat -n affiche_adresses.c | expand
...
     6  #include <stdio.h>
     7  #include <stdlib.h>
     8  #include <unistd.h>
     9
    10  int     i;                      /* Variable not initialized (segment BSS) */
    11  int     j = 2;                  /* Variable initialized (segment DATA) */
    12  extern int      _end;
    13  extern int      _etext;         /* End of code segment */
    14  extern int      _edata;         /* End of data segment */
    15  extern int      __bss_start;    /* Start of BSS segment */
    16  extern char     **environ;      /* Pointer to environment */
    17  int     *x;
    18
    19  int main (int argc, char *argv[])
    20  {
    21          int k;
    22
    23          printf ("The page size for this system is %ld (%p) bytes\n",
    24                  sysconf(_SC_PAGESIZE), sysconf(_SC_PAGESIZE));
    25          printf ("Address of the function main      = %p\n", main);
    26          printf ("Address of the function printf    = %p\n", printf);
    27          printf ("Address of the symbol _etext      = %p\n", &_etext);
    28          printf ("Address of the variable j         = %p\n", &j);
    29          printf ("Address of the symbol _edata      = %p\n", &_edata);
    30          printf ("Address of the symbol __bss_start = %p\n", &__bss_start);
    31          printf ("Address of the variable i         = %p\n", &i);
    32          printf ("Size of the int variable          = %lu\n", sizeof(int));
    33          printf ("Address of the symbol _end        = %p\n", &_end);
    34          printf ("Address of the variable k         = %p\n", &k);
    35          printf ("Address of the arguments counter  = %p\n", &argc);
    36          printf ("Value   of the arguments counter  = %d\n", argc);
    37          printf ("Address of the first argument     = %p\n", &argv[0]);
    38          printf ("Address of the first env.variable = %p\n", environ[0]);
    39          x = (int *) malloc(sizeof(int));
    40          printf ("Address of the variable in x      = %p\n", x);
    41          free(x);
    42          exit(0);
    43  }
```

```
$ ./affiche_adresses
The page size for this system is 4096 (0x1000) bytes
Address of the function main      = 0x4006dd
Address of the function printf    = 0x400590
Address of the symbol _etext      = 0x40090d
Address of the variable j         = 0x601060
Address of the symbol _edata      = 0x601064
Address of the symbol __bss_start = 0x601064
Address of the variable i         = 0x601098
Size of the int variable          = 4
Address of the symbol _end        = 0x6010a0
Address of the variable k         = 0x7fffbeca125c
Address of the arguments counter  = 0x7fffbeca124c
Value   of the arguments counter  = 1
Address of the first argument     = 0x7fffbeca1358
Address of the first env.variable = 0x7fffbeca33e8
Address of the variable in x      = 0x1917010
```

a) (2 puntos – 10 min.) Suponiendo que este programa tiene acceso a las direcciones desde 0x0, ¿cuál sería el tamaño su espacio de direcciones?

b) (2 puntos – 10 min.) ¿Qué páginas ocupa este programa y sus datos?

c) (2 puntos – 10 min.) Si la tabla de páginas sería de un nivel con cada entrada suya de 4 bytes, ¿cuál sería su tamaño?

d) (2 puntos – 10 min.) Si se organiza la tabla de páginas de multinivel con las entradas de tablas de 4 bytes y con la condición que cada tabla ocupe 1 página, ¿cuántos niveles se necesita y cuál sería la estructura de la dirección virtual?

La práctica ha sido preparada por VK
en Linux Mint 17 con LibreOffice Writer.

Profesor del curso: (0781) V. Khlebnikov

Pando, 4 de noviembre de 2014