**PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ**
**FACULTAD DE CIENCIAS E INGENIERÍA**

# SISTEMAS OPERATIVOS

**3ra práctica (tipo a)**
**(Primer semestre de 2018)**

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 (5 puntos – 25 min.)** Considere el siguiente programa en C++:

```
$ cat -n 2018-1_pr3_v1.cpp  | expand
     1  #include <bits/stdc++.h>
     2  using namespace std;
     3
     4  void algorithm(int blockSize[], int m, int processSize[], int n)
     5  {
     6      int allocation[n];
     7      int k=0;
     8
     9      memset(allocation, -1, sizeof(allocation));
    10
    11      cout << "\nBlock No.\tBlock Size\n";
    12      for (int i=0; i<m; i++)
    13          cout << "   " << i+1 << "\t\t" << blockSize[i] << endl;
    14
    15      for (int j=0; j<n; j++) {
    16          while (k<m) {
    17              if (blockSize[k] >= processSize[j]) {
    18                  allocation[j] = k;
    19                  blockSize[k] -= processSize[j];
    20                  break;
    21              }
    22              k=(k+1)%m;
    23          }
    24      }
    25      cout << "\nProcess No.\tProcess Size\tBlock no.\t...\n";
    26      for (int i = 0; i < n; i++) {
    27          cout << "   " << i+1 << "\t\t" << processSize[i] << "\t\t";
    28          if (allocation[i] != -1) {
    29              cout << allocation[i] + 1;
    30              cout << "\t\t" << blockSize[allocation[i]];
    31          }
    32          else
    33              cout << "Not Allocated";
    34          cout << endl;
    35      }
    36
    37      cout << "\nBlock No.\tBlock ... Size\n";
    38      for (int i = 0; i < m; i++)
    39          cout << "   " << i+1 << "\t\t" << blockSize[i] << endl;
    40  }
    41
    42  // Driver code
    43  int main() {
    44      int blockSize[] = {160, 900, 280, 600};
    45      int processSize[] = {590, 50, 275, 460, 150, 140, 130};
    46      int m = sizeof(blockSize)/sizeof(blockSize[0]);
    47      int n = sizeof(processSize)/sizeof(processSize[0]);
    48
    49      algorithm(blockSize, m, processSize, n);
    50      return 0;
    51  }
```

```
$ g++ 2018-1_pr3_v1.cpp -o 2018-1_pr3_v1

$ ./2018-1_pr3_v1 | expand
```

```
Block No.       Block Size
   1            160
   2            900
   3            280
   4            600

Process No.     Process Size    Block no.       ...
   1            590             ...             ...
   2            50              ...             ...
   3            275             ...             ...
   4            460             ...             ...
   5            150             ...             ...
   6            140             ...             ...
   7            130             ...             ...

Block No.       Block ... Size
   1            ...
   2            ...
   3            ...
   4            ...
```

**a) (4 puntos – 20 min.)** Complete los resultados de ejecución del programa marcados con "...".

**b) (1 punto – 5 min.)** La ventaja del algoritmo *First-Fit* consiste en su simplicidad y rapidez por no revisar toda la lista. Y la desventaja está en colocar los programas al inicio de la memoria creando allá pequeños huecos, lo que provoca la perdida de su velocidad. Explique por qué exactamente este algoritmo trabajará más lento con el tiempo que está pasando. ¿En qué se pierde el tiempo?

**Pregunta 2 (5 puntos – 25 min.)** Consideremos un ejemplo que muestra cómo funciona el algoritmo *buddy system*. En nuestro caso, entre los nodos del mismo tamaño, asignaremos el nodo con la menor dirección. Complete el texto y la tabla con los valores que están marcados con "**...**". En su respuesta pueden omitir la columna 2 de la tabla.

Tenemos un solo bloque en la dirección 0x0 y de tamaño 16KB (0x**...**).
Solicitud A de 3686 bytes. Se le asigna el bloque que denotaremos con la tupla (dirección, tamaño) (ambos en hexadecimal): (0x0, 0x**...**). Tendremos los siguientes bloques libres (dirección,tamaño): **...** .
Ahora, en la forma de una tabla:

| | | Bloque asignado | Bloques libres |
|---|---|---|---|
| 1. | Solicitud A de 3686 bytes | (0x**...**,0x**...**) | (0x**...**,0x**...**), **...** |
| 2. | Solicitud B de 1536 bytes | (0x**...**,0x**...**) | **...** |
| 3. | Solicitud C de 1228 bytes | (0x**...**,0x**...**) | **...** |
| 4. | Solicitud D de 1946 bytes | (0x**...**,0x**...**) | **...** |
| 5. | Solicitud E de 2764 bytes | (0x**...**,0x**...**) | **...** |
| 6. | C termina | | **...** |
| 7. | B termina | | **...** |
| 8. | Solicitud F de 1536 bytes | (0x**...**,0x**...**) | **...** |
| 9. | Solicitud G de 1638 bytes | (0x**...**,0x**...**) | **...** |
| 10. | D termina | | **...** |
| 11. | A termina | | **...** |
| 12. | G termina | | **...** |
| 13. | Solicitud H de 6964 bytes | (0x**...**,0x**...**) | **...** |
| 14. | E termina | | **...** |
| 15. | Solicitud I de 850 bytes | (0x**...**,0x**...**) | **...** |
| 16. | Solicitud J de 610 bytes | (0x**...**,0x**...**) | **...** |
| 17. | Solicitud K de 1638 bytes | (0x**...**,0x**...**) | **...** |
| 18. | Solicitud L de 750 bytes | (0x**...**,0x**...**) | **...** |
| 19. | H termina | | **...** |
| 20. | Solicitud M de 3994 bytes | (0x**...**,0x**...**) | **...** |
| 21. | Solicitud N de 1740 bytes | (0x**...**,0x**...**) | **...** |

| | | | |
|---|---|---|---|
| 22. | L termina | | **...** |
| 23. | F termina | | **...** |
| 24. | K termina | | **...** |
| 25. | Solicitud P de 670 bytes | (0x**...**,0x**...**) | **...** |
| 26. | I termina | | **...** |
| 27. | N termina | | **...** |
| 28. | J termina | | **...** |

**Pregunta 3 (3 puntos – 15 min.)** Un sistema asigna espacios de direccionamiento de 32768 bytes, divididos en páginas de 2048 bytes. Un programa particular tiene 19450 bytes de texto, 2050 bytes de datos y requiere de 10200 bytes para la pila.

**a) (1,5 puntos – 7,5 minutos)** Explique por qué este programa no se puede ejecutar en el espacio de direccionamiento disponible.

**b) (1,5 puntos – 7,5 minutos)** En este sistema, ¿qué y cómo se puede modificar para lograr que el programa dado se ejecute? Justifique su respuesta.

**Pregunta 4 (2 puntos – 10 min.)** Suponga que un proceso emite una dirección lógica igual a 2368 y que se utiliza la técnica de paginación, con páginas de 2048 bytes.

**a) (1 punto – 5 min.)** Indique el número de página y el desplazamiento que corresponde a dicha dirección.

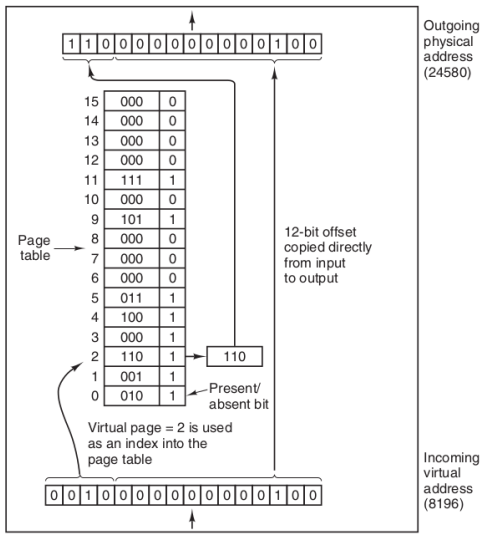
**b) (1 punto – 5 min.)** ¿Es posible que dicha dirección lógica se traduzca en la dirección física 8592? Justifique su respuesta.

**Pregunta 5 (3 puntos – 15 min.)** Dado un sistema de gestión de memoria basado en múltiples niveles de paginación, se trata de determinar el número de niveles necesarios para que la tabla de primer nivel pueda caber en una TLB de 32 entradas. Se supone que la dirección lógica tiene un formato de 36 bits, el tamaño de página es de 1024 bytes, el tamaño de la entrada de cualquiera de las tablas de páginas es de 8 bytes y que cada tabla de nivel superior al primero ocupa el tamaño de una página.

**Pregunta 6 (2 puntos – 10 min.)** Dada la siguiente tabla de páginas:

**a) (1 punto – 5 minutos)** Dibuje su correspondiente la tabla de páginas invertida.

**b) (1 punto – 5 min.)** Dibuje su tabla de páginas invertida, usando la función de *hash*: `f(x) = x % 3`.




La práctica ha sido preparada por AB(3-6) y VK(1,2)
en Linux Mint 18.3 Sylvia con LibreOffice Writer

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

Pando, 25 de mayo de 2018