



UNIVERSIDAD
DE MÁLAGA

Dpto. Lenguajes y
Ciencias de la Computación

Programación de Sistemas y Concurrencia Control 19/4/2012

APELLIDOS _____ NOMBRE _____

DNI _____ ORDENADOR _____ GRUPO/TITULACIÓN _____

Ejercicio de C

Se desea simular el almacenamiento en memoria de un planificador de tareas basado en prioridades. El planificador gestiona la lista de tareas del sistema que están en estado ejecutable, es decir, que pueden ejecutarse en cuanto les llegue el turno. Cada tarea se representa con un identificador (`char *id`) que es una cadena de caracteres, y un número (`int pri`) que representa la prioridad de la tarea. El planificador ordena las tareas en función de su prioridad teniendo en cuenta que cuanto más grande sea `pri` la tarea tiene más prioridad. Además, en el caso de haya tareas de igual prioridad, la ordenación depende del momento en que llegan a la estructura. Como se puede ver en la figura, la tarea **t2** está después de la tarea **t1** y esto se deberá a que **t2** habrá pasado al estado de “lista para ejecutar” después que **t1**:

planif



Implementar las siguientes operaciones:

```
void crear(T_Planificador *planif);
```

Inicializa el planificador creando un planificador vacío.

```
void insertar_tarea(T_Planificador *planif, int pri, char *id);
```

Inserta una nueva tarea **id** de prioridad **pri** en el planificador **planif**. La lista está ordenada por prioridad y en el caso de que exista una tarea con la misma prioridad se almacenará por orden de llegada. El identificador de tarea es único.

```
void mostrar (T_Planificador planificador);
```

Muestra el estado del planificador.

```
void eliminar_tarea(T_Planificador *planif, char *id, unsigned *ok);
```

Dado un planificador, elimina una tarea **id** que está preparada para ejecución. En el caso de que no exista dicha tarea, se devolverá 0 en el parámetro `ok`. OK valdrá 1 en el caso de que se haya realizado el borrado.

```
void planificar(T_Planificador *planif);
```

Extrae de la estructura la tarea que le corresponde ejecutarse.

```
void destruir(T_Planificador *planif);
```

Destruye toda la estructura eliminando y liberando la memoria de todos los nodos.

Nota.- Se recomienda utilizar las funciones `strcpy` y `strcmp` de manejo de cadena de caracteres:

```
char *strcpy(char *s1, const char *s2);  
Copia la cadena apuntada por s2 (incluyendo el carácter nulo) a la cadena apuntada por s1  
  
int strcmp(const char *s1, const char *s2);  
Compara la cadena apuntada por s1 con la cadena apuntada por s2. Y  
Mayor que cero si s1>s2 (orden lexicográfico)  
Cero si s1==s2 (orden lexicográfico)  
Menos que cero si s1<s2 (orden lexicográfico)
```

Ejercicio de Java

Diseña un programa java que implemente un algoritmo de búsqueda de un número **int valor** en un array de enteros **int[] vector** de forma recursiva y concurrente. El programa debe construir un árbol binario de hebras, de profundidad variable, que depende del número de componentes de **vector**. El sistema sólo necesita una clase **Nodo** que se comporta como se describe a continuación.

Inicialmente, el método **main()** crea un primer nodo (el nodo raíz del árbol) al que se le pasa un array **vector** de números aleatorios. Cada uno de los nodos que se vayan creando en el sistema se comporta de la misma forma. Si el vector que se le pasa al nodo en el constructor tiene 0 o 1 elementos, es trivial comprobar si el **valor** está en el array. En otro caso, la hebra crea dinámicamente dos nuevas hebras, y le pasa a cada una de ellas una de las dos mitades del vector. Una vez que cada hebra hija ha buscado el **valor** en su mitad, la hebra padre comprueba si **valor** estaba en alguna de las dos mitades, y devuelve el resultado correspondiente. Como ayuda para la implementación, se sugiere la siguiente estructura para el constructor de la clase **Nodo** y el método **main**:

```
public Nodo(int[] vector,int inic,int fin,int valor,boolean[] res){  
    // busca "valor" en vector[inic..fin-1] y deja el resultado en res[0]  
    ....  
}  
  
public static void main(String[] args){  
    Random r = new Random();  
    int[] vector = new int[r.nextInt(20)+1];  
    for (int i = 0; i<vector.length; i++){  
        vector[i] = r.nextInt(20);  
    }  
    int valor = r.nextInt(20);  
    System.out.println("Buscamos "+valor+" en "+Arrays.toString(vector));  
    boolean[] res = new boolean[1];  
    Nodo nodo = new Nodo(vector,0,vector.length,valor,res);  
    ....  
}
```