Instituto Politécnico Superior "Gral. San Martín"
Especialidad Informática
6^{t0} Año - Instalacion Y Remplazo De Componentes Internos

Estructura de datos en assembly del MIPS R2000

Integrantes:
Agustín Fernández, Luciano Duarte, Ramiro Gatto, Santiago Méndez, Alaia Ibarbia

En este informe se explican las subrutinas y estrategias utilizadas en el código y se muestran algunas salidas del programa.

1. Variables en .data

En él .data están, aparte de los mensajes, están declaradas las siguientes variables:

free_slist:

Acá se guarda la dirección de una lista simplemente enlazada, donde se guardan, las direcciones de los nodos liberados con *sfree*. Los nodos se sacan de la lista usando *smalloc*. De esta manera no es necesario volver a pedir memoria al sistema operativo, sino que se usa la memoria ya pedida, pero que se ha dejado de usar.

cclist:

Acá es donde se almacena la dirección del primer nodo de la lista de categorías

wclist:

Acá es donde se almacena la dirección del nodo de la categoría seleccionada.

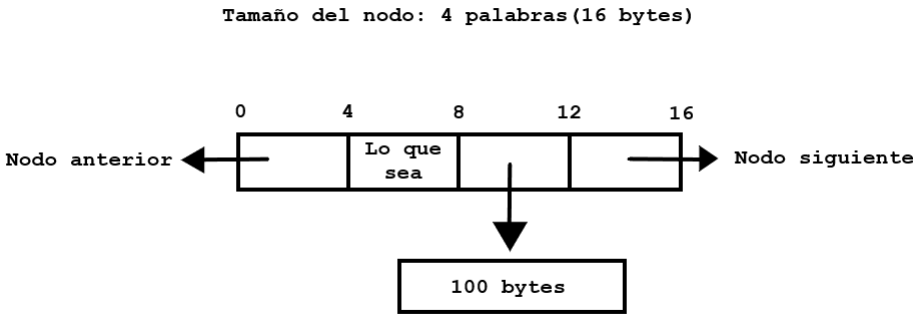
2. Subrutinas

Inicialmente el programa tiene una Rutina main, que realiza las siguientes tareas:

1. Muestra un menú con opciones, también muestra un mensaje con la categoría en curso (si la lista de categorías no está vacía).
2. Pide al usuario que ingrese un numero por teclado, si el número no está entre 1 y 9 muestra un mensaje de "Opción invalida" y vuelve a pedirlo.
3. Si la opción escogida es 9, finaliza el programa. Si no entonces selecciona la opción acorde al menú, llama a la subrutina que le corresponde y una vez terminada vuelve al paso 1.

El programa utiliza 5 subrutinas principales para realizar estas tareas, todas operan en base a una ***lista enlazada doblemente circular***:

Los nodos en la lista tienen esta forma:



El espacio de 100 bytes es usado para almacenar un string. Mientras que el espacio que dice "Lo que sea" puede almacenar desde un número, hasta la dirección otra lista enlazada.

2.1. Subrutinas principales

2.1.1. newnode

Toma por referencia el primer nodo de una lista a través de $\$a0$ y un parámetro extra por $\$a1$.
Lo primero que hace *newnode* es pedir un nodo de 16 bytes de tamaño usando la función *smalloc* proporcionada.

Luego llena el espacio que corresponde al string en el nodo con un string, esto lo hace a través de la subrutina *getstring*. Primero reserva 100 bytes de memoria para el string y luego llena su contenido con el string que hay en *buffer* usando la subrutina *strcpy*(que funciona igual que su homónima en C). Finalmente guarda la dirección de este nuevo espacio en la 3era palabra del nodo.

El parámetro extra que está en $\$a1$ se almacena en la 2da palabra del nodo y luego la subrutina continua.

Ahora hay que unir el nodo a la lista. Para eso hay 3 escenarios posibles:

El primer nodo de la lista es NULL:

La lista esta vacía, se toma al nodo recientemente creado como el primer nodo y se lo une a si mismo tanto por delante como por atrás.

El primer nodo de la lista, está unido a si mismo por delante y por detrás:

Hay un solo nodo de la lista, el nodo recientemente creado se une al primer nodo tanto por delante por atrás. Se hace exactamente lo mismo con el primer nodo de la lista, pero se lo une al nodo recientemente creado.

Ninguna de las situaciones anteriores se cumple:

Hay 2 o más nodos en la lista, se toman el primer y el ultimo nodo de la lista (por ejemplo, nodo a y nodo b). Luego se toma al nodo recientemente creado(c) y se hacen los siguientes enlazamientos.

$$c \leftrightarrows a$$
$$b \leftrightarrows c$$

De manera que la lista queda de esta forma:

$$(...) \leftrightarrows b \leftrightarrows c \leftrightarrows a \leftrightarrows (...)$$

Finalmente, la subrutina termina, el primer nodo de la lista queda actualizado en $\$a0$.

2.1.2. delnode

Esta subrutina se encarga de borrar un nodo de una lista. La dirección del nodo se pasa a través de $\$a0$

Si el nodo pasado es NULL la función no hace nada.

Si el nodo no es NULL, entonces se guarda en $\$t0$ y en $\$t1$ el nodo anterior y el nodo siguiente a $\$a0$

Luego se pasa el nodo $\$a0$ a la lista de nodos liberados usando la subrutina *sfree* proporcionada. Los valores de los registros $\$ra$, $\$a0$, $\$t0$ y $\$t1$ usando el stack y se restauran luego de llamar la subrutina.

Una vez liberado el nodo, hay que reenlazar la lista. Hay 2 situaciones posibles:

El nodo liberado está unido a sí mismo:

El nodo liberado era el único nodo de la lista, así que la lista queda vacía.

La situación anterior no se cumple:

Quedan 1 o más nodos en la lista. Así que la lista queda de esta forma:

$$(...) \leftrightarrows \$t0 \leftrightarrows \$t1 \leftrightarrows (...)$$

Donde $\$t0$ es el nodo que está a la izquierda del nodo liberado y $\$t1$ es el nodo que está a la derecha del nodo liberado.

Finalmente se actualiza $\$a0$ con NULL si la lista queda vacía o $\$t1$ sino. Luego la subrutina termina.

2.1.3. doinlist

Esta subrutina ejecuta otra subrutina pasada como argumento en cada nodo de una lista.

Toma como parámetros la dirección de un nodo de la lista en $\$a0$, la dirección de una subrutina $\$a1$ y un parámetro extra en $\$a2$.

Luego la subrutina entra en un bucle:

Si en algún momento la subrutina vuelve al nodo donde comenzó o si llega a NULL, el bucle termina.

Sino entonces realiza lo siguiente

- Ejecuta la subrutina pasada en $\$a1$ usando como parámetros el nodo actual y el parámetro extra en $\$a2$.
- Llama a la subrutina *next* pasándole el nodo actual. Esto hace que pase al nodo siguiente.

Luego la subrutina termina.

2.1.4. next

Esta subrutina toma un nodo por referencia y actualiza su valor con el nodo que le sigue. Si el nodo esta enlazado a sí mismo, entonces el resultado es el mismo nodo.

2.1.5. prev

Esta subrutina también toma un nodo por referencia y actualiza su valor con el nodo que le precede. i el nodo esta enlazado a sí mismo, entonces el resultado es el mismo nodo.

2.2. El resto de subrutinas

Gracias a las 5 subrutinas principales, es posible hacer todas las tareas del programa (en algunos casos es necesario una subrutina auxiliar). Ejemplo:

El menú del programa tiene las siguientes opciones:

1. Crear una nueva categoría
2. Pasar a la siguiente categoría
3. Pasar a la anterior categoría
4. Mostrar todas las categorías
5. Borrar la categoría seleccionada
6. Anexar un objeto a la categoría seleccionada
7. Borrar un objeto de la categoría seleccionada
8. Mostrar todos los objetos de la categoría seleccionada
9. Salir

Y las realiza de la siguiente forma:

2.2.1. Crear una nueva categoría

Esto se realiza usando la subrutina *newcategory* que lo que hace es:

1. Pide un string por teclado y lo almacena en *buffer*.
2. Llama a la subrutina *newnode* usando como parámetros a *cclist* y NULL (de esta forma se guarda una lista vacía).

2.2.2. Pasar a la siguiente categoría

Se llama a *next* usando como parámetro a *wclist*

2.2.3. Pasar a la anterior categoría

Se llama a *prev* usando como parámetro a *wclist*

2.2.4. Mostrar todas las categorías

Se llama a *doinlist* usando como parámetros a:

1. *cclist*
2. Una subrutina llamada *printnodestring* de la cual se hablará a continuación.
3. *wclist*

Lo que hace *printnodestring* es tomar 2 argumentos, los dos son direcciones de 2 nodos.

Si la dirección del nodo del primer argumento y del segundo argumento coinciden, entonces imprime un asterisco. De esta forma se puede marcar la categoría actual cuando se muestra.

Luego de realizar esta comparación, imprime el string del primer nodo (independientemente de si el primer y segundo nodo son o no son iguales).

2.2.5. Borrar la categoría seleccionada

Se llama a *delnode* usando como parámetro a *wclist*.

Como se libera el nodo de la categoría seleccionada también es necesario liberar la lista de objetos que contiene.

Esto se realza usando *doinlist*, y se la llama usando como parámetros a la lista de objetos asignada a *wclist* y *delnode*. El parámetro extra no se usa.

2.2.6. Anexar un objeto a la categoría seleccionada

Se llama a *newnode* usando como parámetro a la lista de objetos de *wclist* y el ID del nuevo objeto.

Si se va a crear un objeto en una lista vacía, se pasa 1 como parámetro. Sino, se pasa el ID del ultimo objeto +1

2.2.7. Borrar un objeto de la categoría seleccionada

Acá hay 2 escenarios:

Se va a eliminar un objeto con un ID distinto de 1:

Se llama a *doinlist* usando como parámetros la lista de objetos en *wclist*, el ID del objeto a borrar como parámetro extra y una subrutina que toma un nodo y un parámetro extra y si el dato de la 2da palabra del nodo y el parámetro extra coinciden, entonces libera el nodo. El nodo es tomado por referencia así que automáticamente se actualiza su valor con el nodo que le sigue. De esta forma se elimina solo el nodo que queremos. Sin embargo, esto no funciona si se elimina el nodo con ID 1, como veremos a continuación.

Se va a eliminar el objeto de ID 1:

Se llama específicamente a *delnode* usando la lista de objetos de *wclist*, ya que usar a *doinlist* para hacerlo generaría un bucle infinito. Esto pasa porque *doinlist* termina una vez vuelve al inicio o llega a NULL. Si eliminas el inicio de la lista y aún quedan objetos, entonces nunca va a poder volver al inicio o a NULL.

2.2.8. Mostrar todos los objetos de la categoría seleccionada

El método es prácticamente el mismo que para mostrar todas las categorías. Solo que en vez de usar a *wclist*, se usa la lista de objetos asociada.

3. Ejemplos

3.1. Crear categorías y objetos

Vamos a usar el programa para hacer lo siguiente

1. Crear 2 categorías: Mamíferos y Reptiles
2. En la categoría de Mamíferos vamos a crear 3 objetos: "Perro", "Gato" y "Tigre dientes de sable".
3. En la categoría Reptiles vamos a crear 2 objetos: "Caimán" y "Tortuga".

Para mayor comodidad, se van a remplazar todos los mensajes del menú (salvo el primero) con "|**|".

```
1) Crear una nueva categoria
2) Pasar a la siguiente categoria
3) Pasar a la anterior categoria
4) Mostrar todas las categorias
5) Borrar la categoria seleccionada
6) Anexar un objeto a la categoria seleccionada
7) Borrar un objeto de la categoria seleccionada
8) Mostrar todos los objetos de la categoria seleccionada
9) Salir

Seleccione una opcion por su numero: 1
Ingrese una palabra: Mamiferos

Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 1
Ingrese una palabra: Reptiles

Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 4
*Mamiferos
Reptiles

Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 6
Ingrese una palabra: Perro

Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 6
Ingrese una palabra: Gatto

Categoria en curso: Mamiferos
```

```
|**|
Seleccione una opcion por su numero: 6
Ingrese una palabra: Tigre dientes de sable

Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 8

Perro
Gatto
Tigre dientes de sable

Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 2

Categoria en curso: Reptiles
|**|
Seleccione una opcion por su numero: 6
Ingrese una palabra: Tortuga

Categoria en curso: Reptiles
|**|
Seleccione una opcion por su numero: 6
Ingrese una palabra: Caiman

Categoria en curso: Reptiles
|**|
Seleccione una opcion por su numero: 8
Tortuga
Caiman
```

3.2. Borrar categorías y objetos

Ahora vamos a probar de borrar lo que creamos

1. En la categoría Mamíferos, vamos a borrar el 1er y 2do objeto.
2. En la categoría Reptiles, vamos a borrar directamente la categoría
3. Luego vamos a terminar el programa.

```
Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 7
Ingrese un numero: 1

Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 7
Ingrese un numero: 1

Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 8
Tigre dientes de sable

Categoria en curso: Mamiferos
```

```
|**|
Seleccione una opcion por su numero: 2

Categoria en curso: Reptiles
|**|
Seleccione una opcion por su numero: 5

Categoria en curso: Mamiferos
|**|
Seleccione una opcion por su numero: 4
*Mamiferos
```