

Portada

Insertar

Eliminar

Ejemplo list entry Funciones Macros

Curso de programación en C moderno (II Edición)

Neira Ayuso, Pablo Falgueras García, Carlos

Tema 9

Listas encadenadas



Índice

Portada

Insertar

Eliminar

Ejemplo list_entry

Macros

- 1 ¿Qué es una lista encadenada?
 - 2 ¿Cuándo son útiles?
 - 3 Operaciones
 - Insertar
 - **■** Eliminar
- 4 Lista del Kernel
 - Ejemplo
 - ¿Cómo se obtiene el elemento a partir del list_head?
 - Funciones principales
 - Macros principales



¿Qué es una lista encadenada?

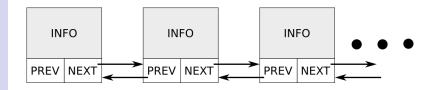
Portada

¿Qué es?

2400 00

Operacione Insertar

Eliminar



- Cada elemento guarda:
 - 1 Información
 - 2 Una referencia al siguiente elemento [y al anterior]
- Los elementos están dispersos en la memoria. Se reservan individualmente.



¿Cuándo son útiles?

Portada

Utilidad

Insertar Eliminar

Ejemplo

list entry Funciones Macros

- No sabemos cuántos elementos vamos a tener que guardar
- Vamos a recorrer los elementos de manera secuencial
- Necesitamos hacer inserciones y/o eliminaciones de elementos o sublistas



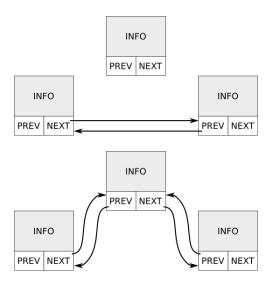
Insertar

Portada

¿Qué es?

Insertar Eliminar

Kernel





Eliminar

Portada

¿Qué es?

Z Que es

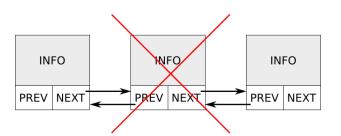
Operacione

Insertar

Eliminar

Elimina

Kernel







Lista del Kernel

Portada

¿Qué es

Operacione

Insertar Eliminar

Lista del Kernel

```
struct list_head *next
struct list_head *prev
```

```
struct list_head {
   struct list_head *next;
   struct list_head *prev;
};
```

```
int a
struct list_head *next
struct list_head *prev
    int b
```

```
struct element {
  int a;
  struct list_head list;
  int b;
};
```



Portada

```
struct list head *next
struct list head *preve
```

Insertar

Eliminar

```
int main()
2
3
    struct list head list;
    struct element *element:
4
5
6
    INIT LIST HEAD(&list);
7
```



Portada

¿Qué es

2 -- --

Operacione

Insertar

Eliminar

Lista de

```
struct list_head *next struct list_head *next struct list_head *prev struct list_head *prev 1
```

```
int main()
{
    struct list_head list;
    struct element *element;

    INIT_LIST_HEAD(&list);

    element = element_alloc(1, 1);
    list_add(&(element->list), &list);

    element = element_alloc(2, 2);
    list_add(&(element->list), &list);

    list_add(&(element->list), &list);

    list_or_each_entry(element, &list, list)
        printf("{%d, %d}\n", element->a, element->b);

    return 0;
}
```



Portada

¿Qué es

2 4 40 00

Operacione

Insertar Eliminar

Elimina: Lista de

Ejemplo list_entry Funciones

```
struct list_head *nextend tist_head *nextend tist_head *prev struct list_head *prev struct list_head *prev tist_head *prev tis
```

```
int main()
2
3
     struct list head list;
     struct element *element:
4
     INIT LIST HEAD(&list);
7
8
     element = element alloc(1, 1);
     list add(&(element->list), &list);
10
11
     element = element alloc(2, 2);
12
     list add(&(element->list), &list);
13
```



Portada

¿Qué es

Operacione

Insertar Eliminar

Lista del Kernel

```
> list_example
{2, 2}
{1, 1}
```

```
int main()
2
3
     struct list head list;
4
     struct element *element;
5
6
     INIT LIST HEAD(&list);
7
8
     element = element alloc(1, 1);
     list add(&(element->list), &list);
10
11
     element = element alloc(2, 2);
12
     list add(&(element->list), &list);
13
     list for each entry (element, &list, list)
14
       printf("{%d, %d} \n", element -> a, element -> b);
15
16
17
     return 0;
18
```



¿Cómo se obtiene el elemento a partir del list_head?

Portada

¿Qué es

Operacione Insertar

Eliminar Lista del

Ejemplo

list_entry Funciones Macros

```
ptr int a

struct list_head *next
struct list_head *prev

int b
```

```
/**

* list_entry - get the struct for this entry

* @ptr: the &struct list_head pointer.

* @type: the type of the struct this is embedded in.

* @member: the name of the list_struct within the struct.

*/

#define list_entry(ptr, type, member) \

((type *)((char *)(ptr)-(unsigned long)(&((type *)0)->member)))
```



Funciones principales

Portada

¿Qué es

Operaciones
Insertar
Eliminar

Eliminar Lista del Kernel

- list_add: Añade después de la cabeza (pila)
- list_add_tail: Añade antes de la cabeza (cola)
- list_move: Mueve un elemento de una lista a después de la cabeza de otra (pila)
- list_move_tail: Mueve un elemento de una lista a antes de la cabeza de otra (cola)
- list_del: Borra el nodo que recibe
- list_splice: Une dos listas
- list_empty: Comprueba si una lista está vacía



Macros principales

Portada

¿Qué es

Operacione Insertar

Insertar Eliminar

Kernel
Ejemplo
list_entry
Funciones
Macros

• INIT_LIST_HEAD: Inicializa la cabeza de una lista

- list_for_each: Recorre cada nodo de una lista
- list_for_each_safe: Recorre cada nodo de una lista y se puede borrar un elemento mientras se rocorre la lista.
- list_for_each_entry: Recorre cada elemento de una lista
- list_for_each_entry_safe: Recorre cada elemento de una lista y se puede borrar un elemento mientras se rocorre la lista.