Creación del array de estructuras hijo

C Prácticas

En esta entrada vamos a ver cómo podemos crear el array de estructuras hijo. Principalmente tenemos dos opciones, utilizando memoria reservada estáticamente (en tiempo de compilación) y utilizando memoria reservada dinámicamente (en tiempo de ejecución)

Si tenemos la estructura definida en un .h como este

include.h

```
typedef struct {
    int pid ; // Pid del hijo
    int num ; // Numero de orden del hijo
    int num_aciertos; // Numero de aciertos.
 int pipehijo[2]; // Descriptores el pipe del hijo
    long premio;
} HIJO ;
```

podemos hacer uso de esta estructura para crear un array de este tipo de estructuras

En el siguiente código vemos cómo hacerlo de dos formas.

```
#include <stdio.h>
#include <stdlib.h>
#include "include.h"

#define COUNT(x)((int)(sizeof(x)/sizeof(HIJO)))

HIJO *dynamicArrayHijos;


int main(int argc, char *argv[])
{
 int numHijos = atoi(argv[1]);
 int i;

 HIJO staticArrayHijos[numHijos];

 dynamicArrayHijos=(HIJO *)malloc(numHijos*sizeof(HIJO));
 if (dynamicArrayHijos==NULL) {
 printf("Error allocating memory!\n");
 return 1;
 }

 printf("Número de hijos: %d\n",numHijos);
 printf("Size of struct HIJO: %d\n",sizeof(HIJO));

 printf("\n----- STATIC MEMORY ARRAY ------\n");
 printf("Size of staticArrayHijos            : %d\n",sizeof(staticArrayHijos));
 printf("Num of elmenents in staticArrayHijos: %d\n",COUNT(staticArrayHijos));

 printf("\n----- DYNAMIC MEMORY ARRAY ------\n");
 printf("Size of dynamicArrayHijos            : %d\n",sizeof(dynamicArrayHijos));
 printf("Num of elmenents in dynamicArrayHijos: %d\n",COUNT(dynamicArrayHijos));
 printf("La macro COUNT sólo funciona con memória reservada estáticamente.\n");
 printf("sizeof(dynamicArrahHijos)=\"num_bytes_puntero\"\n");

 printf("\n----- DYNAMIC MEMORY ARRAY ------\n");
 printf("Size of dynamicArrayHijos            : %d\n",sizeof(HIJO)*numHijos);
 printf("Num of elmenents in dynamicArrayHijos: %d\n",numHijos);

 for (i=0; i<numHijos; i++){
 staticArrayHijos[i].num=i;
 dynamicArrayHijos[i].num=i;
 }
```

```
    printf("\n----- STATIC MEMORY ARRAY ------\n");
    for (i=0; i<numHijos; i++){
    printf("Hijo %d\n",staticArrayHijos[i].num);
    }

    printf("\n----- DYNAMIC MEMORY ARRAY ------\n");
    for (i=0; i<numHijos; i++){
    printf("Hijo %d\n",dynamicArrayHijos[i].num);
    }

    free(dynamicArrayHijos); //liberamos memoria dinámica

    return(0);
}
```

Vemos que si definimos estáticamente el array, tenemos que declararlo como un variable local al main puesto que necesitamos el número de hijos que nos pasan como parámetro. A pesar de ser una variable local al main será vista por todas las funciones llamadas desde main, con lo que podríamos pensar que es global.

Como vemos el array estático es entendido por la macro, y sizeof del array estático da el tamaño total de bytes asignados en tiempo de compilación. Lo que la macro hace es dividir este tamaño por el tamaño que ocupa cada una de las estructuras para determinar el número de elementos en el array.

Sin embargo, cuando utilizamos memoria dinámicamente reservada con calloc/malloc, no tenemos forma de saber cuántos elementos del array hay, puesto que lo que le estamos pidiendo a malloc/calloc es que reserve un número de bytes, sin más. Nos devuelve un puntero al primer byte reservado y es responsabilidad nuestra el saber que se trata de un array de determinado número de elementos.

Como vemos en el código, la primera vez que intentamos acceder al tamaño del array fallamos puesto que estamos accediendo al tamaño de un puntero y la macro no reconoce la memoria dinámica.

Por eso, en la segunda vez tenemos que mostrar la información con las variables que creamos para reservar la memoria.

Los dos últimos bucles iteran por los elementos del array que han sido inicializados previamente. ¿por qué si el array de memoria dinámica se ha reservado como un conjunto de bytes, es capaz de devolvernos correctamente los datos del número de orden?

La respuesta es que hemos tenido que hacer un casting a puntero a estructura hijo del puntero que nos devolvió malloc.

← Entrada anterior

Entrada siguiente →

2 comentarios en “Creación del array de estructuras hijo”

9 DICIEMBRE, 2016 A LAS 14:51

Hola Miguel, hay dos valores para DYNAMIC MEMORY ARRAY

printf(“\n—— DYNAMIC MEMORY ARRAY ——\n”);
printf(“Size of dynamicArrayHijos : %d\n”,sizeof(dynamicArrayHijos));

printf(“\n—— DYNAMIC MEMORY ARRAY ——\n”);
printf(“Size of dynamicArrayHijos : %d\n”,sizeof(HIJO)*numHijos);

Durante la ejecución dan valores diferentes

—— DYNAMIC MEMORY ARRAY ——
Size of dynamicArrayHijos : 8

—— DYNAMIC MEMORY ARRAY ——
Size of dynamicArrayHijos : 64

¿Es posible que la segunda sea estática?

Gracias.

Responder

MIGUEL ONOFRE MARTÍNEZ RACH

9 DICIEMBRE, 2016 A LAS 19:37

Hola,

No, el array dynamicArrayHijos se ha reservado dinámicamente.
El hecho de que haya dos salidas para dynamicArrayHijos es para hacer ver el error, y es que en la primera no podemos utilizar sizeof() para determinar el tamaño del array puesto que lo que tenemos es un puntero y no un array estáticamente definido, por eso sizeof en la primera vez muestra 8 que es el tamaño en bytes del propio puntero.
Por tanto, en la segunda vez, que corrige el error de la primera, no tengo más remedio que utilizar la multiplicación del tamaño de la estructura por el número de elementos. Estas variables son las que se utilizaron para pedir la memoria necesaria con malloc. Los dos primeros printf del ejemplo muestran estos datos.

Responder

Deja un comentario

Conectado como alu28. Edita tu perfil. ¿Salir? Los campos obligatorios están marcados con *

Escribe aquí...

Publicar comentario »

Copyright © 2025 Sistemas Operativos
Escuela Politécnica Superior de Elche
Universidad Miguel Hernández
Miguel Onofre Martínez Rach