Aritmética de apuntadores

César Barraza Aguilar, A01176786
Ingeniería en Tecnologías Computacionales
Tecnológico de Monterrey
cbarrazaa1@gmail.com

Resumen

Los apuntadores son la base de la computación. La programación con abstracción más baja funciona con mayormente con apuntadores: apuntadores a bloque de un proceso, apuntador de stack, apuntador de código de ejecución, y muchos más. C es uno de los pocos lenguajes que le dan al programador poder total sobre apuntadores, lo cual permite acceso y manipulación de las direcciones de memoria relacionadas al programa. Esto permite un nivel de indirección de memoria que puede ayudar, por ejemplo, con arreglos o listas encadenadas. En este artículo, se discute el principal uso de apuntadores en C; operaciones aritméticas para manipular datos agrupados en bloques de memoria contiguos. Sin embargo, un gran poder conlleva una gran responsabilidad, y es muy fácil usar los apuntadores de una manera incorrecta.

Palabras clave

Apuntadores, aritmética de apuntadores, C, memoria, stack, heap, comportamiento indefinido

Introducción

Los apuntadores por sí mismos no nos sirven de mucho ya que solo representan un número o una dirección de memoria; más bien, su principal propósito es la facilidad que nos dan para desplazarnos por memoria utilizando simples operaciones aritméticas, y de esa manera acceder diferentes secciones de información que contienen datos que nos interesan.

Arreglos y strings

En C se tiene la costumbre de considerar los apuntadores, los arreglos y los strings como equivalentes; la verdad no es que sean lo

mismo, simplemente se representan de la misma manera mediante el uso de apuntadores. Los arreglos, y, por consiguiente, los strings (ya que un string es simplemente un arreglo de caracteres) son datos agrupados contiguamente en memoria. Es decir, si se conoce dónde está cualquier elemento del arreglo, es posible encontrar los demás, ya que están pegados los unos de otros. Es por eso por lo que, cuando tenemos lo siguiente:

```
int arr[] = {0, 1, 2};
```

`arr` no es más que un apuntador, que simplemente guarda la dirección de memoria del primer elemento del arreglo, es decir, el `0`. A partir de ahí, si le sumamos a ese apuntador, podemos llegar al `1`, al `2`, e incluso más allá de la memoria que tenemos asignada…

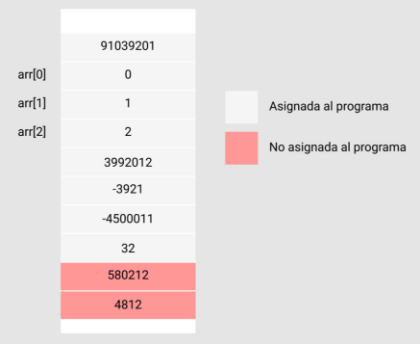
Inseguridad de memoria

Una de las mayores preocupaciones y razones de desprecio de C es que es un lenguaje muy inseguro; da demasiado poder al programador y permite facilita mucho la manipulación de memoria. Es verdad que los apuntadores son una herramienta muy útil, pero también es muy fácil darles un mal uso por la libertad que dan gracias a la aritmética de apuntadores.

Cuando empieza un programa, el sistema operativo lo divide en varios segmentos a los que se les asigna memoria en la RAM; aunque el sistema operativo impone esta regla estrictamente, los apuntadores dan facilidad de violarla y provocar que nuestro programa funcione inesperadamente.

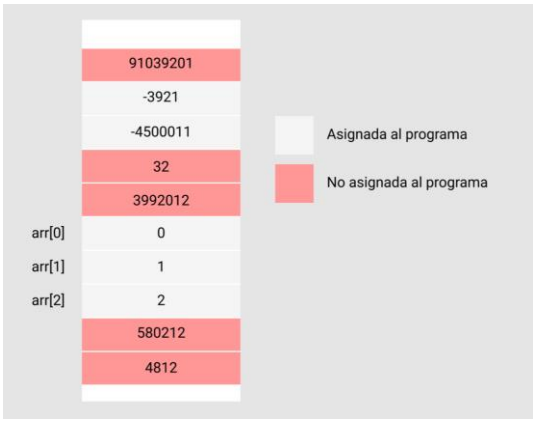
Comportamiento indefinido

El comportamiento indefinido puede ocurrir por varias razones; la violación de acceso de memoria es una de las principales. Salirse de un arreglo es el error más común, pues es muy fácil equivocarse con los límites de iteración. ¿Por qué se llama comportamiento indefinido? Porque pueden pasar diferentes cosas dependiendo del estado de nuestra computadora y de cómo está segmentada la memoria del programa. En la mayoría de los casos, no pasa nada; simplemente leeremos más allá de la memoria a la que se supone que tenemos acceso, y estaremos leyendo basura; el siguiente diagrama muestra la estructura de la memoria cuando no pasa nada.



Podemos ver cómo tenemos un arreglo declarado de 3 elementos, sin embargo, nuestro programa sigue teniendo asignada memoria más allá de ese arreglo. Debido a esto, no pasará nada si nos salimos del arreglo…simplemente leeremos memoria que tenemos asignada, pero a la que se supone que no tenemos referencia.

Otra posibilidad es que el programa tenga un crash con un error que se conoce como "segmentation fault" o error de segmentación. Este es un error que levanta el sistema operativo cuando se da cuenta que alguno de los programas que están corriendo hace una violación de acceso de memoria, y la acción más apropiada para proteger la memoria es terminar la ejecución del programa en cuanto antes (crash). El siguiente diagrama ilustra la estructura de memoria cuando ocurre un segfault.



Este estado de memoria suele ocurrir, en la mayoría de los casos, cuando reservamos memoria dinámicamente (en el heap), ya que el heap crece en memoria conforme se necesite, por lo que es muy probable que se presente el caso diagramado previamente.

Sumas y restas con apuntadores

El uso más común que se le da a un apuntador es utilizarlo como índice dentro de un arreglo, sumándole la posición del elemento que deseamos. Sin embargo, estas operaciones aritméticas funcionan un poco diferente. Básicamente, aunque un apuntador sea solo un número (y por lo cual, todos los apuntadores de cualquier tipo ocupan la misma memoria), se comporta diferente dependiendo de su tipo de dato. El tipo de dato de un apuntador denota cómo interpretarlo, pero más importante aun, permite que la aritmética de apuntadores sea más segura y fácil de utilizar. Se utiliza la

siguiente fórmula para aplicar una suma o una resta a un apuntador:

```
ptr + i = ptr + (sizeof(ptr_tipoDato))
ptr - i = ptr - (sizeof(ptr_tipoDato))
```

Donde **`sizeof`** devuelve el tamaño en bytes de un tipo de dato o una variable. De esta manera, si tenemos un arreglo de números de 4 bits, los saltos cuando sumas, por ejemplo, 1, realmente son saltos de 4. Esto para facilitar el recorrido de bloques de memoria homogéneos, es decir, que guardan información del mismo tipo.

Obviamente esta herramienta se puede utilizar de manera incorrecta. Dado que los apuntadores son sólo números, en C son "inter-casteables" entre sí, es decir, puedes convertir un apuntador tipo entero a un apuntador de tipo estructura, y el compilador lo permite. En estos casos, hay que tener cuidado con cómo se usa la aritmética de apuntadores ya que siempre se utiliza la fórmula previamente descrita para realizar las operaciones.

Conclusiones

Los apuntadores son una herramienta muy controversial en el mundo de la programación. Mucha gente los adora por la libertad y facilidad que dan para manipular memoria, mientras que muchos otros lo desprecian por ser tan inseguro y no ser "beginner-friendly". Definitivamente es una herramienta que debes de dominar si te quieres enfocar a programar en lenguajes como C, C++, Go, etc., que tienen soporte para apuntadores. Te guste utilizar los apuntadores o no, es importante conocer las ventajas y riesgos, ya sea para utilizarlos a tu favor, o que tengas cuidado cuando no te quede de otra más que usarlos.

Referencias

https://nullprogram.com/blog/2018/07/20/

https://www.eskimo.com/~scs/cclass/notes/sx10b.html

https://www.ibm.com/support/knowledgecenter/SSGH2K_12.1.0/com.ibm.xlc121.aix.doc/language_ref/ptarith.html

https://courses.washington.edu/css342/zander/css332/pointerarith.html

http://pages.cs.wisc.edu/~fischer/cs536.s08/lectures/Lecture31.4up.pdf