Memoria de un proceso

martes, 23 de noviembre de 2021 07:18 a. m.

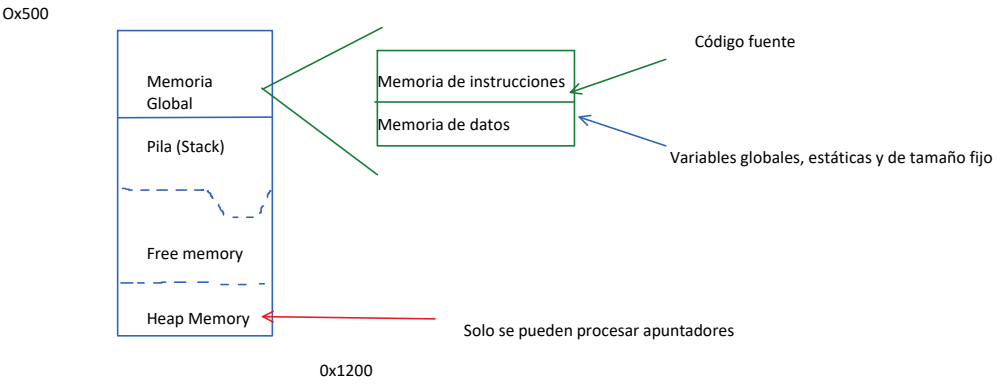
Cuando un programa se escribe, se asume que las variables se almacenan en memoria y esta disponibles para ser utilizadas. En principio, los detalles de cómo se almacenan y organizan los datos no son visibles para un programador. Sin embargo, como el lenguaje de programación C nos ofrece una gestión de memoria muy cercana a la RAM, para realizar un uso eficiente de los recursos es preciso conocer más de cerca cómo se organiza la memoria de un proceso.

Cuando un programa se ejecuta sobre un S.O. existe un proceso previo de carga para suministrar un bloque contiguo de memoria sobre el cual ejecutarse.

El programa que resulta de la compilación debe ejecutarse de tal forma que haga uso de este bloque. Para eso, el compilador incorpora al programa objeto el código de gestión.

Cada lenguaje de programación tiene diferente forma de gestionar ese bloque de memoria e incluso organiza de diferente forma a los datos y varía entre los modos de ejecución final.

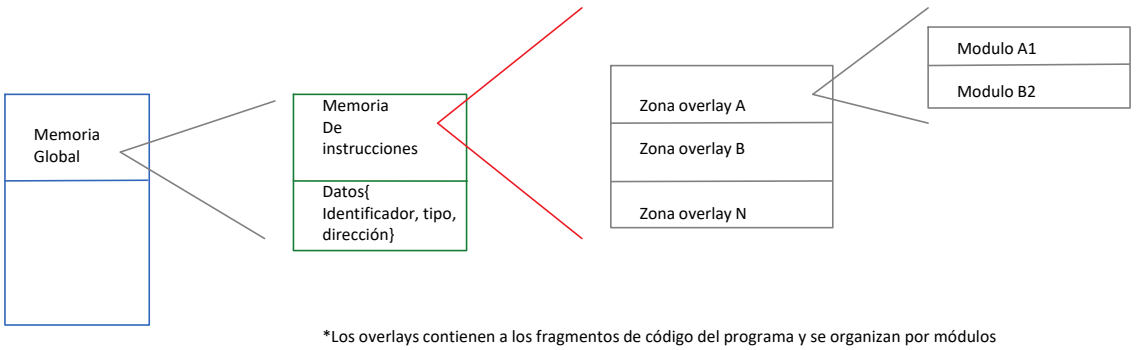
Un proceso (programa en ejecución) almacena sus datos en memoria en tres áreas generales.



MEMORIA GLOBAL

Es el área en la que están almacenadas las variables que se declaran globales o estáticas y las constantes, así como las cadenas y arreglos de tamaño fijo, es decir, en esta zona de memoria se almacenan todos aquellos datos que estaban presentes desde el comienzo del programa hasta que termina.

También en esta área se almacena el código del programa en ejecución, en la zona de memoria de instrucciones, su tamaño se fija en tiempo de compilación. Algunos compiladores e intérpretes fragmentan el código en zonas llamadas "overlays", estas zonas tienen organizado el código fuente y se cargan en tiempo de ejecución dependiendo el código/módulo a utilizar.



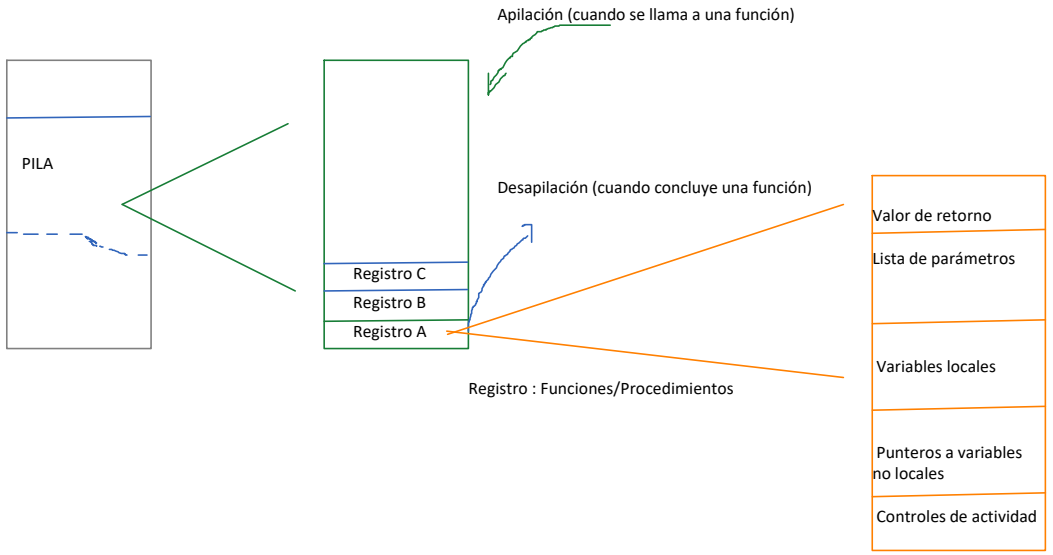
*Los overlays contienen a los fragmentos de código del programa y se organizan por módulos

En la sección de memoria de datos se guardan las variables globales o estáticas a excepción de:

- Variables declaradas dentro de funciones o procedimientos
- Estructuras de datos dinámicas como las lista, colas, pilas, árboles o arreglos.

La PILA

Es un área en la que las variables aparecen y desaparecen en un momento puntual de ejecución del programa. Se utiliza principalmente para almacenar variables locales dentro de las funciones. Estas variables tienen un ámbito reducido, sólo están disponibles mientras se ejecutando la función en donde han sido definidas. En la Pila se encuentran todas estas variables locales y por lo tanto, en esta Zona se están constantemente insertando y borrando variables. En caso de que el tamaño inicial de la Pila no fuese suficiente, entonces se hace uso de la "Free Memory" a través de su zona de desbordamiento.




```
int suma(int op1, int op2){
        int aux;
        aux=op1+op2;
        printf("Resultado=%d", aux);
        return aux;
}
```

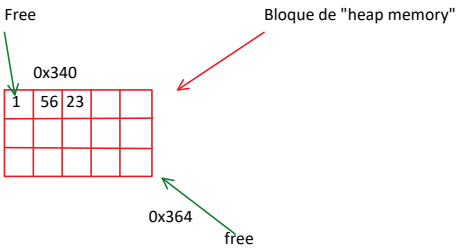
Heap Memory

Esta zona, conocida también como montículo o del "montón" contiene memoria disponible para que se reserve y libere en cualquier momento durante la ejecución de un programa. No está dedicada a variable locales de las funciones, como la PILA, sino que esta memoria denominada "dinámica" está dedicada a estructuras de datos/arreglos cuyo tamaño puede variar a lo largo de la ejecución de programa.

Nota: Se debe notar que la única zona con tamaño fijo de memoria es la "Memoria Global" y es porque se sabe su tamaño cuando se compila o se interpreta.

Tanto la PILA como la Memoria del montón albergan datos cuyo tamaño no se puede saber hasta que el programa se encuentra en ejecución.

La manipulación de la "Heap memory" en C se hace con un mecanismo muy simple, pero a la vez muy propenso a errores. Para gestionar esta zona sólo necesitan dos operaciones: petición de memoria (alojamiento) y desalojo (liberación de memoria). El ciclo es sencillo, cuando se precisa almacenar un nuevo dato, se solicita tanta memoria en bytes como sea necesaria y una vez que ese dato ya no se utilice, entonces la memora se devuelve para ser reutilizada. Este esquema se conoce como "gestión explicita de memoria" pues requiere ejecutar una operación explicita para pedir memoria y otra explicita para liberarla.



- Alojamiento: Se reserva un bloque contiguo de memoria para poder almacenar una variable de cierto tamaño.
- Liberación: Se indica que ya no es necesario conservar un espacio y por lo tanto, la memoria que se estaba utilizando debe quedar libre, para ser reutilizada en caso de que se requiera por otra variable.

*En la liberación de memoria NUNCA se borra la información, solo se marca como disponible.