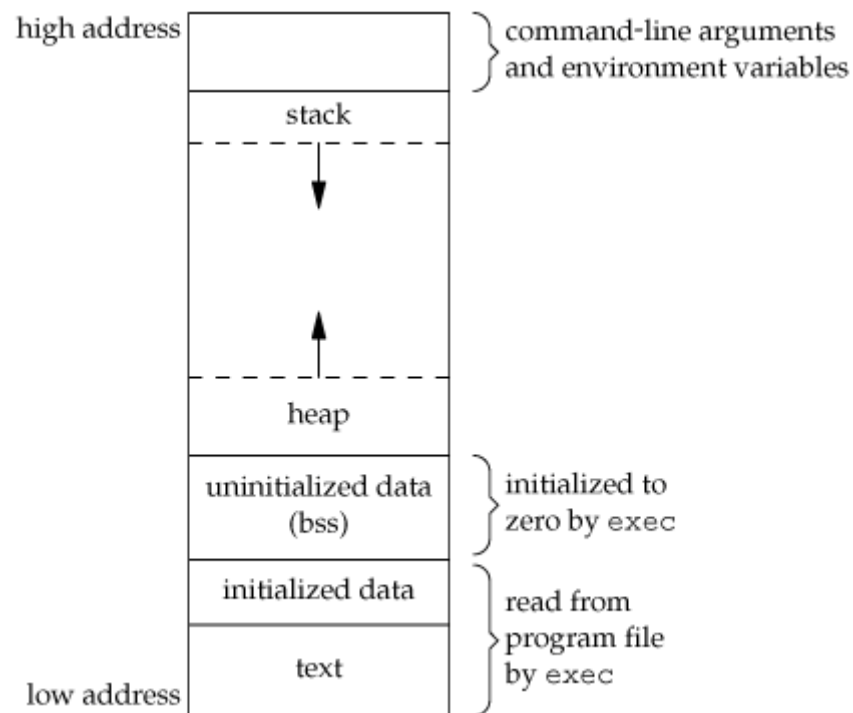


Nombre: Stefanie Muroya Lei
Grupo: CCOMP2-1.2

TAREA: Investigación de la distribución de la memoria en programas en C



1. Segmento de texto:

- Es también conocido como segmento de código.
- Es una parte del programa que contiene instrucciones en lenguaje máquina a ser ejecutadas.
- Está en el código objeto o en memoria.
- Generalmente es puesta debajo del stack y del heap para prevenir sobreescrituras a la hora que uno de estos haga un overflow.
- Generalmente el segmento de texto es compartible entre programas para que programas diversos lo puedan ejecutar (shells, compiladores C, editores de texto).
- Es un segmento de solo lectura para evitar errores y modificaciones inesperadas.

2. Segmento de datos inicializados:

- Llamado usualmente como segmento de datos.
- Es un segmento de direcciones virtuales.
 - Direcciones virtuales: es un conjunto de direcciones virtuales que el sistema operativo le da al programa para su uso durante su ejecución.
- Aquí se guardan las variables globales, estáticas y constantes inicializadas por el programador.
- No es un segmento de sólo lectura puesto que las variables varían durante la ejecución del programa.
- El segmento puede clasificarse en datos inicializados de solo lectura (variables constantes) y de lectura-escritura.

3. Segmento de datos no inicializados:

- También llamado segmento bss luego de que un operador (encargado de ejecutar el programa) de assembler lo nombre así por “block stated by symbol”.
- Este segmento es inicializado en 0 por el kernel antes de que se comience a ejecutar el programa.
- Contiene todas las variables globales y estáticas inicializadas en 0 o que no están inicializadas explícitamente en el programa.

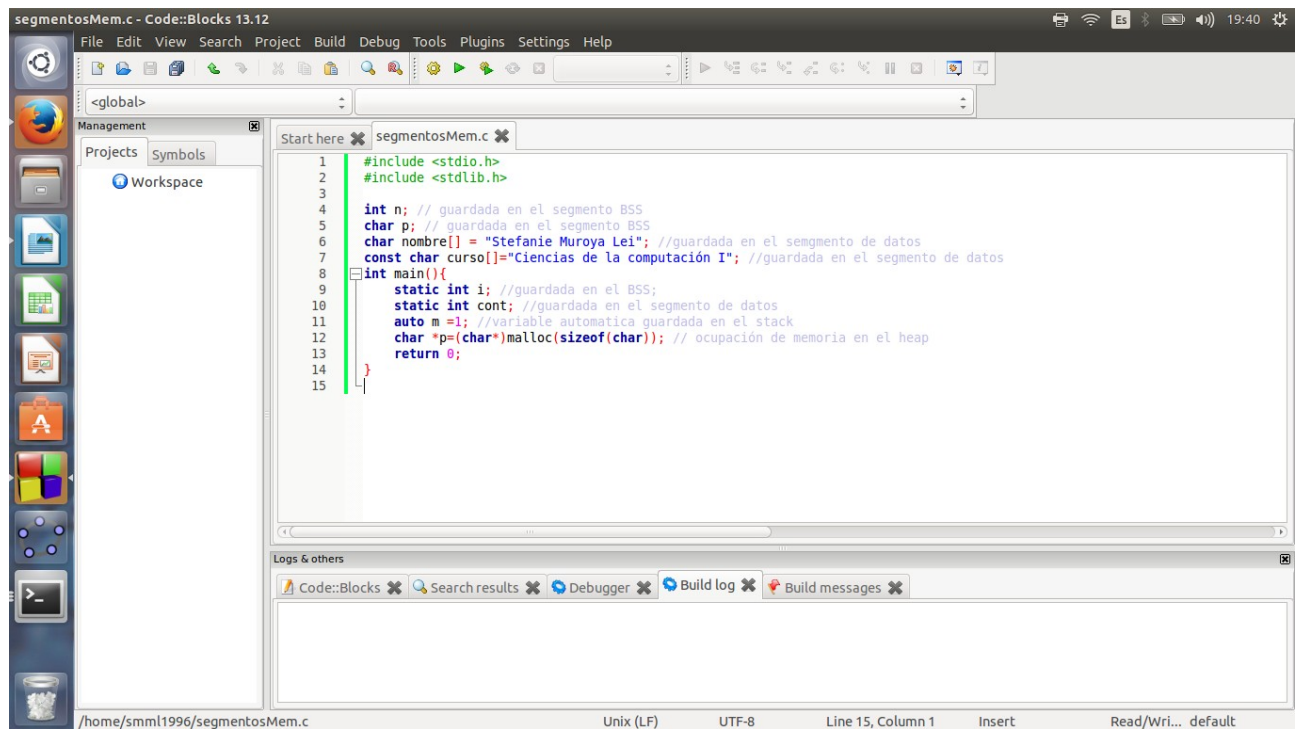
4. Segmento Heap:

- Aquí es donde sucede la ocupación dinámica de memoria.
- Funciones malloc(), calloc() retornan punteros void* a ser casteados a direcciones en esta parte de la memoria.
- Crece hacia arriba a medida que se va usando (las direcciones van aumentando).
- Comienza en el fin del segmento BSS.
- Es un area compartida entre las librerías y modulos cargados dinámicamente durante la ejecución del programa.

5. Segmento Stack:

- Crece en dirección opuesta al heap. Los punteros de ambos segmentos se encuentran es cuando la memoria se acaba.
 - Con las tecnicas de memoria virtual y grandes espacios de memoria, en teoría los punteros pueden ser puestos en cualquier lugar.
- Es de estructura LIFO(último ingresado, primero en salir).
- El puntero stack mantiene marcado el tope del stack. Se ajusta cada vez que una nueva variable es declarada.
- Un conjunto de variables válidas para una llamada de función están en un “stack frame”.
 - Un “stack frame” consiste al menos de un retorno a una dirección de memoria donde guardar el resultado.
- Algunas direcciones de registros son guardadas aquí.
- Las variables de las funciones son guardadas aquí. Se crea un nuevo frame cada vez que se llama a la función.

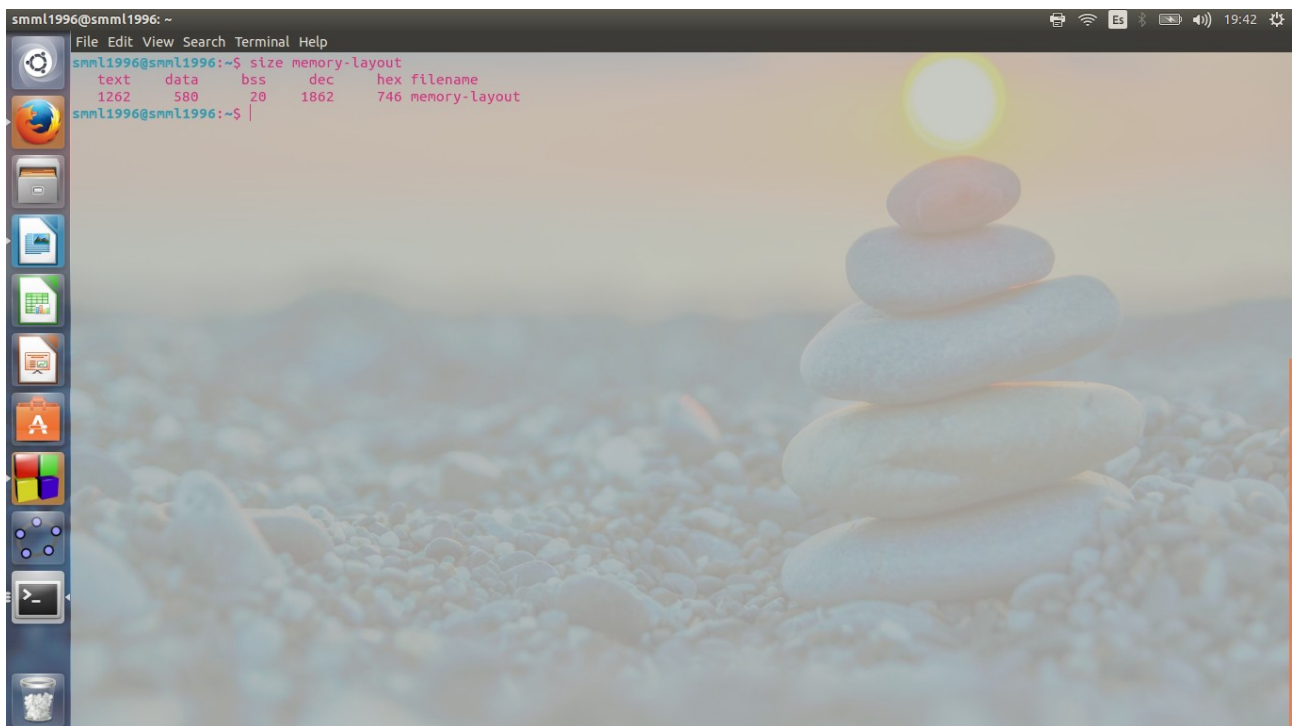
EJEMPLO:



The screenshot shows the Code::Blocks IDE with the file 'segmentosMem.c' open. The code is as follows:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int n; // guardada en el segmento BSS
5 char p; // guardada en el segmento BSS
6 char nombre[] = "Stefanie Muroya Lei"; //guardada en el segmento de datos
7 const char curso[]="Ciencias de la computación I"; //guardada en el segmento de datos
8
9 int main(){
10     static int i; //guardada en el BSS;
11     static int cont; //guardada en el segmento de datos
12     auto m =1; //variable automatica guardada en el stack
13     char *p=(char*)malloc(sizeof(char)); // ocupación de memoria en el heap
14     return 0;
15 }
```

The interface includes a menu bar, a toolbar, a sidebar with 'Management' and 'Workspace' tabs, and a bottom status bar showing the file path, encoding (UTF-8), and line/column information (Line 15, Column 1).



The screenshot shows a terminal window with the following output:

```
smml1996@smml1996: ~
smml1996@smml1996:~$ size memory-layout
text  data  bss   dec   hex filename
1262   580    20   1862   746 memory-layout
smml1996@smml1996:~$
```

The terminal background features a serene image of a stack of smooth, light-colored stones on a beach, with a bright sun or moon low on the horizon, creating a soft, hazy glow.

*Esta última imagen nos muestra el espacio ocupado en bytes para el segmento de texto, de datos y el bss. Luego nos muestra también el espacio total ocupado en sistema decimal y hexadecimal.