**Assignment – 1**

**2. (10 marks) Based on the information generated from the above program, produce a memory map table showing the layout of literals, initialised global variables, uninitialized global variables, formal parameters of each function, local variables, dynamically allocated variables, functions, environment and command line arguments in the memory when the program runs at label L in function f2.**

**The memory map table must show the addresses of each variable, literal, and function and their sizes. It should also show the start**

**and end addresses of the environment and the command line arguments and their sizes.**

**The memory map table must contain at least the following columns:**

**1. The start address of an entity such as a variable or a function**

**2. The length of the storage space of the entity in bytes**

**3. The name of the entity, such as global\_pointer1 or Hello, world!**

**4. The nature of the entity, such as *function*, or *uninitialized global variable***

**5. The memory section, e.g. environment, command line arguments, code (or text), global initialised data, global uninitialed data, stack, heap etc.**

**In addition, you must use seven different background colours to highlight the following seven memory sections as indicated below::**

**[Green] initialised global variables (including constants and literals)**

**[Red] uninitialised global variables**

**[Blue] stack (containing the local variables and returning addresses of function calls)**

**[Magenta] heap (containing the dynamically allocated memories)**

**[Yellow] code (functions)**

**[Cyan] process environment**

**[White] command line arguments**

**Please also note that in your memory table, *the memory addresses must be strictly sequential, from the highest address to the lowest address* to reflect how different components of a running program are layout in the virtual memory. Your memory tablewill not be accepted if the addresses are not lined up sequentially in the table. If you find that components from one section aresplit in more than one continuous area of memory, it is a sure indication that there is something wrong with your memory map andyou should find out what went wrong and fix it.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **entity name such as global\_pointer1 or Hello, world!** | **Start address** | **Length of Storage (byte)** | **Nature of entity (Memory type) *such as function*, or *uninitialized global variable*** | **Memory section e.g. environment, command line arguments, code (or text), global initialised data, global uninitialed data, stack, heap etc** |
| argc | 0x7fffc604dffc | 4 | Length of command line argument | Command line argument |
| argv | 0x7fffc604e0f8 | 8 | Command line arguments | Command line argument |
| &environ[0] | 0x7fffc604e138 |  | Process environment | Environment |
| &environ[sizeof  (environ-1)] | 0x7fffc604e170 |  | Process environment | Environment |
| &main | 0x7f1b790009b6 |  | Function | Initialised data |
| &f1 | 0x7f1b790008c9 |  | Function | Initialised data |
| &f2 | 0x7f1b7900081a |  | Function | Initialised data |
| global\_x | 0x7f1b79202010 |  |  | Global Initialised variable |
| global\_y | 0x7f1b79202050 |  |  |  |
| global\_array1 | 0x7f1b79202018 |  |  |  |
| global\_array2 | 0x7f1b79202058 |  |  |  |
| F1\_x | 0x7fffc904ec40 |  |  |  |
| f1\_y | 0x7fffc904ec44 |  |  |  |
| f1\_p1 | 0x7fffc904ec48 |  |  |  |
| f1\_p2 | 0x7fffc904ec50 |  |  |  |
| string literal “this is a string” | 0x7fffc904ec48 |  |  |  |
| &x | 0x7fffc904ebec |  |  |  |
| f2\_p | 0x7fffc904ec00 |  |  |  |
| f2\_x | 0x7fffc904ebfc |  |  |  |
| f2\_p[0] | 0x7fffc06974e0 |  |  |  |

**Output listing:**

root@vishnu-rana:/mnt/c/Users/Vishnu Rana/Documents/ICT374-OSSP/Assignment/q2# gcc memory.c -o memory

root@vishnu-rana:/mnt/c/Users/Vishnu Rana/Documents/ICT374-OSSP/Assignment/q2# ./memory sadvwer fbr fbr sfdw df

My OS bit size: 64

Address of argc 0x7fffc904ec7c

Address of all elements in argv

address from argv element 0 = 0x7fffc904ed78

address from argv element 1 = 0x7fffc904ed80

address from argv element 2 = 0x7fffc904ed88

address from argv element 3 = 0x7fffc904ed90

address from argv element 4 = 0x7fffc904ed98

address from argv element 5 = 0x7fffc904eda0

Address of the first command line arguments 0x7fffc904ed78

Address of the last command line arguments 0x7fffc904eda0

First address of environment variable 0x7fffc904edb0

Last address of environment variable 0x7fffc904ede8

Starting address of function main = 0x7f5d788009b6

Starting address of function f1 = 0x7f5d788008c9

Starting address of function f2 = 0x7f5d7880081a

address of global\_x = 0x7f5d78a02010

address of global\_y = 0x7f5d78a02050

address of global\_array1 = 0x7f5d78a02018

address of global\_array2 = 0x7f5d78a02058

address of global\_pointer1 = 0x7f5d78a02030

address of global\_pointer2 = 0x7f5d78a02048

address of global\_float = 0x7f5d78a02028

address of global\_double = 0x7f5d78a02068

Address of string literal hello = 0x7f5d78a02018

Address of string literal bye = 0x7f5d78a02030

location x1 = 0x7fffc904ec3c

location x2 = 0x7fffc904ec38

location x3 = 0x7fffc904ec34

location x4 = 0x7fffc904ec30

location x5 = 0x7fffc904ec28

location x6 = 0x7fffc904ec24

f1\_x address = 0x7fffc904ec40

f1\_y address 0x7fffc904ec44

f1\_p1 address = 0x7fffc904ec48

f1\_p2 address = 0x7fffc904ec50

Address of string literal (this is a string) 0x7fffc904ec48

x location = 0x7fffc904ebec

f2\_p location = 0x7fffc904ec00

f2\_x location = 0x7fffc904ebfc

f2\_p location = 0x7fffc06974e0

root@vishnu-rana:/mnt/c/Users/Vishnu Rana/Documents/ICT374-OSSP/Assignment/q2#