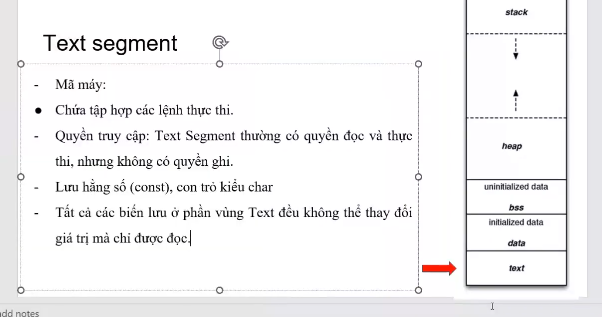
Phân vùng bộ nhớ

Mảng thì truy cập vào vị trị

Khi khai báo char \*str = “Hello Word ” thì ta không thể thay đổi được str[1]= “a” được nhé

Phần Text Segment là một phần của bộ nhớ trong mô hình bộ nhớ của một chương trình máy tính để sử dụng lưu trữ mã máy của chương trình

Text Segmen chứa mã máy được biên dịch từ mã nguồn của chương trình và được CPU thực thi để thực hiện các hành động quy định trong chương trình



Phân vùng Data là phân vùng chưa biến toàn cục chứa các biến static (toàn cục và cục bộ) khởi tạo với giá trị khác 0

Quyền truy cập là đọc và ghi, tức là có thể đọc và thay đổi giá trị cả biến

Char \*str= “Hello world” trình biên dịch quy định và điều đó

Truy cập đến địa chỉ của nó để có thể thay đổi được giá trị A screenshot of a computer

AI-generated content may be incorrect.

Đọc là và in ra màn hình được

Còn ghi tức là thay đổi được giá trị của nó

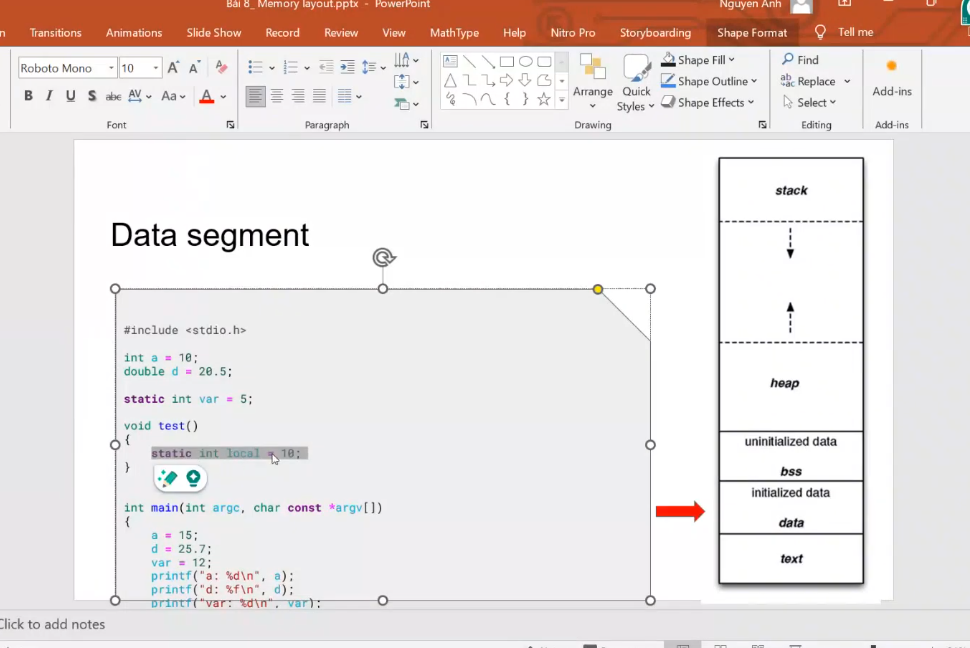
Tất cả các biến sẽ được thu hồi sau khi chương trình kết thúc

Khi mở Vscode thì các biến toàn cục và biến static thì các biến khời tạo

Khi tắt VSCODE thì các biến đó cũng mất đi

Khi khai báo các biến không nằm trong các hàm thì sẽ là biến toàn cục

Biến static được khai báo trong cái hàm là biến static cục bộ



Toàn cục

Static Toàn cục

Static cục bộ

A screen shot of a computer program

AI-generated content may be incorrect.

Chú ý ở đây không có const nhé

Khi có const thì không thể trực tiếp thay đổi giá trị của nó được

Const int c =10; // ở đây cũng được cấp địa chỉ tuy nhiên ta có thể tuy cập được biến con trỏ và thay đổi được giá trị của nó , con trỏ sẽ truy cập đến địa chỉ và thay đổi giá trị

A screenshot of a computer

AI-generated content may be incorrect.

Data segment thì có quyền thay đổi nên ta có thể thay đổi giá trị của a và d ở phần main dưới

Ví dụ như int a=10; có địa chỉ là 0x01 thì ở dưới a =15 thì truy cập đến và thay đổi giá trị của nó thôi hen

Biến static int var =5;

Có địa chỉ 0x05 thì truy cập địa chỉ 0x05 để thay

Khai báo biến một const trong hàm main (khai báo local)

Const int c=10; // cx có một địa chỉ

Tuy nhiên mình không thể trực tiếp thay đổi để c=5 mình có thể khai báo biến con trỏ tuy cập vào địa chỉ để thay đổi giá trị

Còn tuy nhiên không thể dùng trong khai báo global

#include<stdio.h>

int main ()

{

    const int a = 5;

    int \*p = (int \*)&a;

    \*p = 7;

    printf("%d", a);

}

Kết quả :

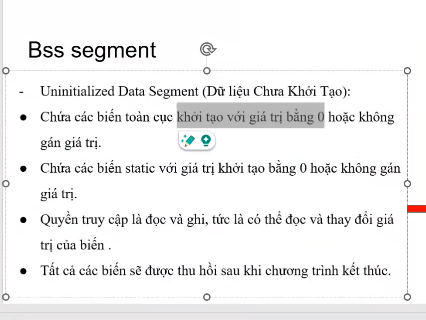
7

(int\*)&a → Ép kiểu địa chỉ của a thành **con trỏ kiểu int\***.

(void\*)&a -> Ép kiểu thành con trỏ void (void\*), nghĩa là con trỏ không có kiểu dữ liệu cụ th ể.

BSS segment

Chứa các biến toàn cục khởi tạo với giá trị bằng 0 hoặc không gán giá trị



Với các biến khai báo bằng 0 và khai báo không có giá trị hay là các biến static thì sẽ được cho vào phân vùng BSS segment.

Struct là kiểu dữ liệu nên nó không năm ở phân vùng nào

Nếu bình thường ở phân vùng BSS vì khai báo bằng 0 nên nếu thay đổi giá trị thì nó vẫn ở BSS không chuyển sang các vùng nhớ khác (căn cứ vì lần đầu khởi tạo )

Ví dụ :

Static Point\_data p1 ={5,7}

Biến static với giá trị khác không thì có thể ở vùng nhớ Data còn nếu khai báo ở static point\_data p1= {0,0}

A screen shot of a computer program

AI-generated content may be incorrect.

Khởi tạo giá trị khác không (0) mặc dù biến toàn cục thì nó {5,7} ở Data

Stack

Chứa các biến cục bộ, tham số truyền vào

Quyền truy cập, đọc và ghi -> nó có thay đổi giá trị của biến trong suốt thời gian chương trình chạy.

Sau khi ra khỏi hàm thì sẽ bị thu hồi trong vùng nhớ

Biến cục bộ vào phân vùng stack

A computer screen with text

AI-generated content may be incorrect.

Các thông số truyền vào như int a, int b, or thêm nữa int x cx làm phân vùng stack

Int c cx ở Stack luôn

A screen shot of a computer

AI-generated content may be incorrect.

Sẽ cho a có giá trị bằng 3 ở địa chỉ 0x01 và cho b có giá trị bằng 5 ở địa chỉ 0x02

Hoạt động theo LIFO last in first out

Khi int a, int b sẽ được cấp địa chỉ 0x01 và 0x02

Biên int c sẽ được lưu vào

Các biến con trỏ được lưu ở phân vùng nhớ nào ?

Tùy thuộc vào biến con trỏ nó trỏ tới thì các biến con trỏ sẽ nằm trong phân vùng nhớ của nó

#include <stdio.h>

#include <stdlib.h>

int globalVar = 10; // Biến toàn cục (Data Segment)

static int staticVar = 20; // Biến static (Data Segment)

void pointerTest() {

int localVar = 30; // Biến cục bộ (Stack)

int\* ptrStack = &localVar; // Con trỏ trên Stack, trỏ đến biến trên Stack

int\* ptrHeap = (int\*)malloc(sizeof(int)); // Con trỏ trên Stack, trỏ đến vùng Heap

\*ptrHeap = 40; // Gán giá trị cho vùng nhớ trên Heap

static int\* ptrStatic; // Con trỏ static trên Data Segment

ptrStatic = &staticVar; // Trỏ đến biến static trên Data Segment

int\* ptrGlobal = &globalVar; // Con trỏ trên Stack, trỏ đến biến toàn cục (Data Segment)

printf("Địa chỉ của biến cục bộ localVar: %p (Stack)\n", (void\*)&localVar);

printf("Địa chỉ của con trỏ ptrStack: %p (Stack)\n", (void\*)&ptrStack);

printf("Địa chỉ của giá trị mà ptrStack trỏ đến: %p\n", (void\*)ptrStack);

printf("Địa chỉ của con trỏ ptrHeap: %p (Stack)\n", (void\*)&ptrHeap);

printf("Địa chỉ của giá trị mà ptrHeap trỏ đến: %p (Heap)\n", (void\*)ptrHeap);

printf("Địa chỉ của con trỏ ptrStatic: %p (Data Segment)\n", (void\*)&ptrStatic);

printf("Địa chỉ của giá trị mà ptrStatic trỏ đến: %p\n", (void\*)ptrStatic);

printf("Địa chỉ của biến toàn cục globalVar: %p (Data Segment)\n", (void\*)&globalVar);

printf("Địa chỉ của con trỏ ptrGlobal: %p (Stack)\n", (void\*)&ptrGlobal);

printf("Địa chỉ của giá trị mà ptrGlobal trỏ đến: %p\n", (void\*)ptrGlobal);

free(ptrHeap); // Giải phóng bộ nhớ Heap để tránh rò rỉ

}

int main() {

pointerTest();

return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.

Heap :

Khi khai bảo một cái mảng như này

A screenshot of a computer

AI-generated content may be incorrect.

Sẽ cấp phát các địa chỉ liền kề vì ở đây là uint8\_t là cách nhau một byte

Còn ở nếu khai báo với uint16\_t arr[] ={1,5,8,10,9}

Thì sẽ cấp phát địa chỉ cách nhau 2 byte

A white rectangular object with black text

AI-generated content may be incorrect.

A screenshot of a math equation

AI-generated content may be incorrect.

Còn uint32\_t thì sẽ cách nhau 4 byte

Char name [12] đây là khai báo tĩnh -> sử dụng cấp phát động

int \*ptr =NULL;

ptr = (int\*)malloc (4); // tạo ra bộ nhớ 4 byte 0x01 0x02 0x03 0x04 trả về con trỏ kiểu void

// ptr = (int\*)malloc(sizeof(int)\*4) ; // 4\*4 cấp một vùng nhớ 16 byte có địa chỉ liền kế nhau 0x01 0x02 …0x0e

A computer screen with text and numbers

AI-generated content may be incorrect.

Muốn sử dụng uint16\_t thì phải báo #include <stdint.h>

For (int i=0; i>4; i++)

Ptr[i]= 2\*I;

For (int i=0; i<8; i++)

{

Prrintf(“địa chỉ :%p, gia trị :%d\n”,(uint8\_t)ptr+i,\*(uint8\_t)ptr+i)

}

Ptr =realloc ()

Ptr = (uint16\_t)realloc ( ptr,sizepf(uint16\_t)\*6);

// sử dụng Chat GPT để xem cái realloc

A screen shot of a computer

AI-generated content may be incorrect.

Truyền 2 tham số đó là địa chỉ

Khai báo con trỏ trỏ tới vị trí đầu tiên của con trỏ

Nên sử dụng cấp phát động để tối ưu bộ nhớ

A screenshot of a computer

AI-generated content may be incorrect.

Chạy xong một hàm sẽ thu hồi vùng nhớ

Thử không thư hồi vùng nhớ cho nên khi gọi test1 test 2 lấy địa chỉ cấp phát vùng nhớ

Stlib sẽ cho malloc, calloc, realloc, free

Khai báo con trỏ để

Int \*ptr= NULL;

Ptr =(int\*) malloc (sizeof(int)\*4); // sẽ cấp vùng với cho 16 phần từ

0x01 0x02 0x03 ….. 0x

Return 0;

Có cách nào để thu hồi vùng nhớ ? không dùng hàm free

Thu hồi vùng nhớ

Nếu không sài hàm free thì có cách nào thì thu hồi vùng nhớ ?

Realloc cấp phát bằng 0 để thu hồi vùng nhớ

Bài tập trang 22 và trang 23 để xem sự khác nhau của vùng nhớ heap và vùng nhớ stack

Sự khác nhau malloc calloc realloc

Khai báo biến const ở local thì được khai báo ở phân vùng stack

Câu hỏi : Tại sao khai báo ở biến hằng số ở local thì có dùng con trỏ để thay đổi giá trị nhưng mà khi khai báo ở global thì không thể thay đổi được giá trị

Những biến ở cái hàm nó làm ở stack chứ nó không nằm trong phân vùng text nên nó có thể thay đổi được

Nhưng cái này đặc biệt là khi khai báo static trong hàm

Hằng số const ở phạm vi toàn cục, trình biên dịch sẽ lưu nó phân vùng bộ nhớ Read Only Data đây là vùng chỉ đọc Khi cố gắng thay đổi giá trị của global\_const thông qua con trỏ, hệ điều hành sẽ phát hiện và **gây lỗi Segmentation Fault** do vi phạm quyền truy cập bộ nhớ. rodata giúp bảo vệ dữ liệu hằng số khỏi bị thay đổi ngoài ý muốn, đảm bảo tính an toàn trong chương trình.

Malloc cấp phát nhớ động

File c và file exe được lưu ở bộ nhớ SSD hoặc FLASH (ROM) thì những chương trình này sẽ được copy vào bộ nhớ RAM để thực thi

Khi khai báo static ở toàn cục thì phân vùng text mà phân vúng stack lại sửa được