Day 1.5. Pointer variable

Bản chất và cách sử dụng con trỏ:

Là 1 biến và lưu trữ địa chỉ của biến khác và có thể lưu trữ gì nữa ko? Giá trị của con trỏ là địa chỉ của biến khác.

Có thể lưu trữ cái khác ko? Có thể lưu trữ địa chỉ của 1 biến hoặc là HÀM (function)

Ví dụ cấp phát động thì malloc ra thì 1 đoạn mail tầm khoảng 100 byte chẳng hạn thì đoạn đấy ko là biến gì cả.

Chỉ cần nói là con trỏ là 1 kiểu biến lưu trữ địa chỉ, dừng lại ở địa chỉ được rồi, đừng kiểu 1 biến khác mà nó vốn dung để lưu trữ địa chỉ, thế còn địa chỉ ấy thì nó là **địa chỉ của 1 biến**, địa chỉ của **1 vùng nhớ**, địa chỉ cùa **1 function**, **thanh ghi** (con trỏ trỏ đến thanh ghi GPIO hay con trỏ trỏ đến thanh ghi dung để setup adc hay data của biến đầu ra) cho nên chỉ nên nói con trỏ là 1 kiểu **biến dùng để lưu trữ địa chỉ thế thôi**.

Con trỏ thì có bao nhiêu byte, kiểu dữ liệu int thì 4 byte, float thì 8 byte.

Còn kiểu biến con trỏ lưu trữ địa chỉ của 1 biến khác thì sẽ được hệ thống cấp phát bao nhiêu byte (tuỳ vào hệ thống).

Giả sử chúng ta biết là cho hệ thống của chúng ta chia tất cả các thành phần của chúng ta đều là địa chỉ và chia thành các ô nhớ nhỏ nhất thì được đánh địa chỉ là biến byte. Như vậy có nghĩa là trên mỗi 1 byte thì hệ thống quy định là nó sẽ **dùng bao nhiêu byte để chỉ định là 1 địa chỉ**

Ví dụ hệ thống 32 bit thì ta có ta có thể truy xuất được bao nhiêu địa chỉ: 0 đến 2 mũ 32 địa chỉ

0x00 đến 0xFFFFFFF có nghĩa là ta có 2 mũ 32 ô địa chỉ từ địa chỉ 0.

Lúc này con trỏ chỉ cần là 4 byte đủ để lưu trữ 32 giá trị đấy thôi, lưu trữ từ giá trị thấp đến giá trị cao và như vậy vùng nhớ của nó chỉ có thể truy xuất được là 4GB, đó là nếu như 1 hệ thống 32bit.

Hệ thống 64 bit thì có được nhiều hơn, nó sẽ sử dụng tất cả 64 bit hay gọi là 8 byte để định ra 1 địa chỉ. Ví dụ 1 byte sẽ dùng tất cả 64 bit để lưu trữ địa chỉ của 1 byte nào đó trên hệ thống và như vậy để con trỏ phụ thuộc vào hệ thống.

Con trỏ sử dụng bao nhiêu bit địa chỉ thì bộ nhớ của con trỏ sẽ có bằng ấy. (con trỏ chỉ trỏ được đến kiểu dữ liệu cùng kiểu con trỏ với nó)

Int *p; thì p này chỉ trỏ đến được các địa chỉ của xx kiểu int thôi.

Kiểu dữ liệu của con trỏ là kiểu gì ?

Con trỏ kiểu char hay con trỏ kiểu int thì chúng ta đang nói đến con trỏ kiểu gì (thứ rất là bản chất), là tính chất gì của con trỏ đó (vùng nhớ của con trỏ đó trỏ đến là bao nhiêu).

Khi nói đến kiểu dữ liệu của con trỏ thì ta quan tâm đến **số ô nhớ mà con trỏ đó có thể access sau mỗi lần lấy ra được (có thể truy xuất)**

Ví dụ:

File main.c

```
#include <stdio.h>
Void main ()
{
        Int num = 0xffff0000;/* 4,294,901,760*/
        /*Khai báo thêm 2 con trỏ*/
        Int *p_int;
        Char *p_char;

        /*Hai con trỏ này cùng trỏ tới 1 vị trí*/
        P_int = &num;
        P_char = &num;

        /*In ra */
        Printf("Num address: %d\n", num);
        Printf("p_int address: %d\n", &p_int);
        Printf("p_char address: %d\n", &p_char);
        Printf("p_int = %d\n", p_int);
        Printf("p_char = %d\n", p_char);

        Printf("num = %d\n", num);
        Printf("p_int = %d\n", *p_int);
        Printf("p_char = %d\n", *p_char);
}
```

Kết quả in ra màn hình như sau:

```
Num address: 491162100
P_int address: 491162104
P_char address: 491162112
P_int = 491162100
P_char = 491162100
Num = -65536
*p_int =-65536
*p_char = 0
```

Biến num được cấp phát ô nhớ từ 491162100

P_int được cấp phát ô nhớ từ 491162104, vì là biến local nên được cấp phát nối tiếp nhau.

Đầu tiên num address được cấp phát ô nhớ vị trí 491162100

p_int được cấp phát vào 4 byte tiếp theo 491162104

Còn p_char (491162112) thì đang sử dụng untubu cho nên sử dụng 8 byte địa chỉ, riêng thằng p_int được cấp phát 8 byte cho nên 8 byte tiếp theo của p_char bắt đầu từ địa chỉ 491162112

Do P-int và p_char đều được cấp phát bằng &num cho nên giá trị mà nằm trong thanh ghi p_int (vùng nhớ mà p_int đang được cấp phát) là vùng nhớ của địa chỉ có biến num này.

Num = -65536

Còn num đang bằng giá trị này, vì sao lại là trừ bởi vì đang khai báo kiểu int cho nên nó có cả số âm, và ở đây ta đang thấy như thế này, giá trị của p_char và p_int đang truy xuất (hình như giá trị của p_char đang bị sai). Bởi vì p_int đang truy xuất đến là -65536 là đúng vì nó assign đủ 4 byte của giá trị num (0xffff0000), thằng p_char là con trỏ kiểu char nhưng khi in ra giá trị, ví dụ:

*p_int trả về giá trị nơi mà nó trỏ đến, chính là 65536

Vì sao lại như vậy

Num kiểu int, p_int kiểu int nên khi trả về thì nó trả đủ số byte mà nó được truy xuất theo kiểu int vì kiểu dữ liệu của con trỏ này là kiểu int.

suy ra nó lại đang trỏ đến num này nên nó lấy đủ cả vùng nhớ của ô num này ra và nó sẽ được gán giá trị -65536.

Còn *p_char cùng trỏ tới số num này, cùng truy xuất tới vùng nhớ này thế nhưng do p_char này kiểu char nên nó chỉ truy xuất 1 byte thôi

Giả sử địa chỉ 540 này tính từ byte đầu tiên (0xffff0000) đang có giá trị là byte thấp nhất, là số 0000 cho nên kết quả p_char trả về là 0.

Cho nên cần hiểu kiểu dữ liệu của con trỏ là gì, độ dài được cấp phát cho con trỏ là gì, độ dài của p_char và p_int khác nhau. (số byte cấp phát cho con trỏ p_int và p_char khác nhau)

Khi chúng ta khai báo con trỏ p_int hoặc p-char, có thể chúng ta chưa trỏ đến đâu cả thì bộ nhớ cấp phát cho con trỏ p_int và p_char khác nhau ko.

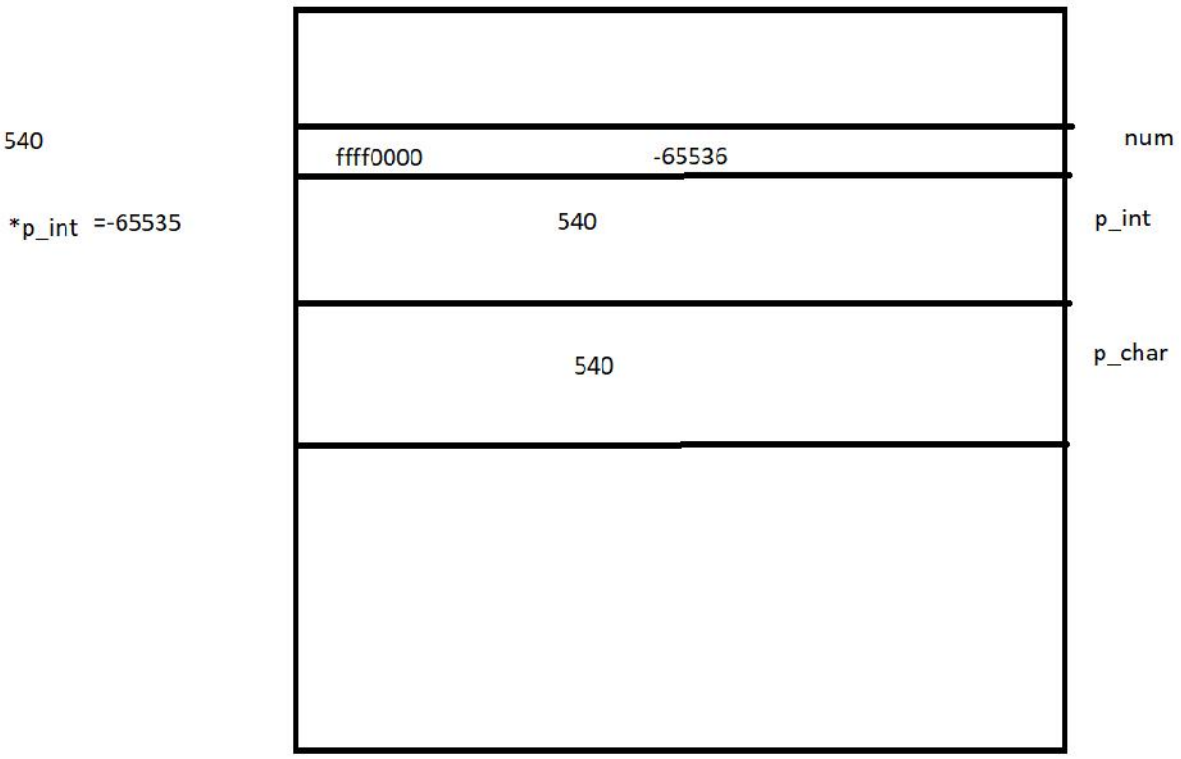
Khi đọc đến dòng

Int *p_int;

Int *p_char;

Có khác nhau ko, ko vì nó cùng là kiểu con trỏ, còn tuỳ vào kiểu lưu trữ giá trị địa chỉ, kể cả là 1 byte, 4 byte hay 100 byte đi nữa thì địa chỉ nó được cấp phát chỉ là trong 1 giá trị giới hạn phụ thuộc vào hệ thống.

Ví dụ ở đây hệ thống đang cấp phát 8byte.



2 ô của p_int và p_char đang vẽ to bằng nhau có nghĩa là trong này mình có 8 byte thế có nghĩa là thằng num này có 4 byte.

Khi chúng ta truy xuất thì giá trị mà trả về khi mà chúng ta sử dụng lệnh *p_int hoặc *p_char thì giá trị của bộ nhớ trả về nó sẽ khác nhau. 1 thằng trả về giá trị bộ nhớ 4 byte, 1 thằng trả về giá trị bộ nhớ 1 byte.

Cách sử dụng con trỏ, thứ nhất là sử dụng con trỏ để truy xuất bộ nhớ hoặc cấp phát bộ nhớ cache của, thứ 2 là sử dụng con trỏ để lưu giữa các file code, các hàm với nhau mà ko cần sử dụng đến biến toàn cục và trong các cấu trúc dữ liệu giải thuật thì sử dụng con trỏ.

Cần hiểu được biến con trỏ là gì và nó được thể hiện trên bộ nhớ như thế nào.