|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Char 1b | Int 2b | Float 4b | Double 8b |  |  | Указатель | 0х00А0 |  |  |

p = &c указывает на с (р= 0х00А00)

& выдаете адрес ячейки на которую ссылается 0ч0010. ( к переменным и элементам массив)

\*выдает сам объект (значение) на который данный указатель указывает (оператор косвенного доступа)

Если указатель ни на что не указывает , приравнивать его к 0

имя массива – это указатель к первому элементу этого массива

4 байта для памяти указаеля

строка char\* указатель на символ. (char в зависимости от системы от 4 до 8 байтов

~~int\* px, py, p.~~ ~~\*\*объявили 1 указатель и две переменные~~

int\* pa, \*pb, \*pc

int x=1, y = 2, z[10]

int \*ip; \*\*ip указатель на int, \* - часть типа данных

ip = &x; \*\*ip указывает на х

y = \*ip; \*\*привоение у значение ip, у=1, \* для доступа к данных в месте

\*ip = 0; \*\*присвоение \*ip значение 0, х=0

ip = &z[0] \*\*. \*ip указывает на z[0]

int \*ip \*\*объявление указателя ip

doble \*dp, atof (char\*); \*\*аргументом функции есть указатель на char

указателю разрешено только на объекты определенного типа (кроме void, может на объекты любого типа с \*)

если х – целого типа, то \*ip  в любом месте, применения х

\*ip = \*ip + 10 \*\*на что указывает ip + 10 и присваивает у

y = \*ip +1

\*ip +=1

++ \*ip

(\*ip)++ \*\*если не будет скобок – увеличится значение самого указателя, а не то на что он указывает

\*\* \*ip это переменная, в коде может быть без \*уц4ф

Указатели - тип переменных, который позволяет вам хранить адрес другой переменной и редактировать его удаленно.. При запуске процесс имеет право использовать всю память, в 32 бит: 4 ГБ, а в 64 бит: 2 в степени 64, посчитайте сколько.

И так, это вириальная память. Это резервная память, эта память отображается либо в оперативной памяти, либо на жестком диске; Сделайте быстрый поиск, чтобы узнать больше о памяти, в том числе о подкачке памяти "swap". Эта память управляется MMU (блоком управления памятью), что позволяет нам иметь память в определенном месте (физически) на компьютере. В свою очередь, у процесса есть собственная область памяти, которой он управляет автономно. Память разделена на две части: память нижнего и ­­­высокого уровня. Память высокого уровня называется стек "stack". "stack" используется каждый раз, когда вы вызываете функцию, каждый раз, когда вы делаете что-либо и т.д. Например, при выделении переменной типа "int" вы выделяете 4 / 8 байтов (в зависимости от архитектуры) памяти в стеке, то есть в памяти высокого уровня. Таким образом, вы все памяти, на уровне 2^64, медленно опускаете вниз. Чем больше элементов вы добавите, тем больше элементов вы опустите в стек. Память нижнего уровня называется кучи "heap". Мы проанализируем ее в другой день. Это область, где вы можете выделить память, но, как я уже сказал, мы увидим это в другой день. В следующем видео мы узнаем как работают указатели и зачем они вообще и т.д.

int main(void)

{

int a;

char c;

int \*ptr; \*\* объявил указатель на то указатель указывать на то, что находится слева int

a = 3;

ptr = &a; \*\*указатель задержит адрес a

~~ptr = &b;~~ \*\*Ошибка поместить адрес переменной типа «char» в указателе на «int».

\*\* несовместимый тип указателя в присваивании

ft\_putaddr(ptr); \*\*отобразить адрес а

return(0);

}

void ft\_putchar(char ptr);

void ft\_putnbr(int ptr);

void ft\_putaddr(void \*ptr);

int main(void)

{

int a;

int \*ptr;

a = 3;

ptr = &a; \*\*указатель указывает на адрес а

ft\_putnbr(\*ptr); \*\*позволяет отображать числа, что нужно в качестве параметра

~~ptr =&c~~ \*\*несовместимый тип указателя на int в присваивании к переменной типа char

\*ptr = 12; \*\*присвоили значение указателю. тоже самое что а=42;

ft\_putaddr(a); \*\*адрес переменной а

ft\_putchar(‘\n’); \*\*’’ значение символов “” значение строк

return(0);

}

void ft\_putchar(char ptr);

void ft\_putnbr(int ptr);

void ft\_putaddr(void \*ptr);

int main(void)

{

Int a;

Int \*ptr;

Int \*\*ptr2;

a=3;

ptr =&a;

ptr2 =&ptr; \*\*указатель на указатель целое число

ft\_putnbr(\*\*ptr2);

\*\*ptr2=42;

ft\_putchar(‘\n’);

ft\_putchar(a);

return (0);

}

int main(void)

{

int a;

int \*ptr;

int \*\*ptr2;

int \*\*\*ptr3;

int \*\*\*\*ptr4;

int \*\*\*\*\*ptr5;

int \*\*\*\*\*\*ptr6;

int \*\*\*\*\*\*\*ptr7;

a=3;

ptr = &a;

ptr2 = &ptr;

ptr3 = &ptr2;

ptr4 = &ptr3;

ptr5 = &ptr4;

ptr6 = &ptr5;

ptr7 = &ptr6;

ft\_putnbr(\*\*\*\*\*\*\*ptr7);

return (0);

}

Арифметика указателей

void ft\_putchar(char ptr);

void ft\_putnbr(int ptr);

void ft\_putaddr(void \*ptr);

int main(void)

{

int a;

int b;

int \*ptr;

a = 3; \*\*а на 4 байта далье b

b = 42; \*\*b  находится в стеке и его адрес приходит первым

ptr = &b;

ft\_putaddr(ptr +1); \*\*ptr является указателем на целое число int , ptr+1 (добавляю 4 байта)

ft\_putchar(‘ ‘);

ft\_putnbr(\*(ptr +1)); \*\*отобразить значенин адреса ptr+1 (сдвинули адрес b на адрес а на 4 байта

~~ft\_putnbr(\*(ptr +а));~~  \*\*не ошибка, а статическая переменная

ft\_putchar(‘\n’); \*\*адрес может с компиляцией поменяться

~~ft\_putaddr(&b);~~

~~ft\_putchar(‘ ‘);~~

~~ft\_putnbr(b);~~

return(0);

}