Основные управляющие символы:

‘\n’ — перевод строки;

‘\t’ — горизонтальная табуляция;

‘\v’ — вертикальная табуляция;

‘\b’ — возврат на символ;

‘\r’ — возврат на начало строки;

‘\a’ — звуковой сигнал.

%d — целое число типа int со знаком в десятичной системе счисления;

%u — целое число типа unsigned int;

%x — целое число типа int со знаком в шестнадцатеричной системе счисления;

%o — целое число типа int со знаком в восьмеричной системе счисления;

%hd — целое число типа short со знаком в десятичной системе счисления;

%hu — целое число типа unsigned short;

%hx — целое число типа short со знаком в шестнадцатеричной системе счисления;

%ld — целое число типа long int со знаком в десятичной системе счисления;

%lu — целое число типа unsigned long int;

%lx — целое число типа long int со знаком в шестнадцатеричной системе счисления;

%f — вещественный формат (числа с плавающей точкой типа float);

%lf — вещественный формат двойной точности (числа с плавающей точкой типа double);

%e — вещественный формат в экспоненциальной форме (числа с плавающей точкой типа float в экспоненциальной форме);

%c — символьный формат;

%s — строковый формат.

cp -r - копировать папку

cp - копировать

mv -переместить

rm- удалить

08.11.2021

**malloc***(кол-во байт)-*выделяет память из кучи

**calloc***(кол-во элементов, сколько байт занимает один элемент)* -выделяет память из кучи, которая гарантированно заполнится "0"

**realloc***(указатель на ранее выделенную область памяти,новый размер выделяемой области памяти)*-изменяет величину выделенной памяти

//если свободной памяти недостаточно для выделения в куче блока, то возвращается нулевой указатель//

//возвращают указатель на начало выделенной области памяти //

примеры:

1. int \*d = malloc(10\*sizeof(int));
2. char \*s =calloc(100,sizeof(char));
3. int \*temp=realloc(a, sizeof ( int ));

if (temp)

a= temp;

//увеличиваем размер массива a, только в том случае, если realloc не возвращает NULL//

strcat??

Опции, Управляющие Видом Вывода

**file.c** Исходный код на C, который нуждается в препроцессировании.

**file.i** Исходный код на С, который не нуждается в препроцессировании.

**file.h** C заголовочный файл (не для компиляции или линковки).

**file.s** Ассемблерный код.

**file.S** Ассемблерный код, который нуждается в препроцессировании.

**другие** Объектный файл, который нужно отдать прямо на линковку. Так

поступают с любым именем файла с нераспознанным суффиксом.

GCC

**-c**

Компилировать или ассемблировать исходные файлы, но не линковать. Стадия ликовки просто не выполняется. Конечный вывод происходит в форме объектного файла для каждого исходного файла.

По умолчанию, имя объектного файла делается из имени исходного файла заменой суффикса '.c', '.i', '.s', и.т.д. на '.o'.

Нераспознанные входные файлы, не требующие компиляции или ассемблирования, игнорируются.

**-S**

Остановиться после собственно компиляции; не ассемблировать. Вывод производится в форме файла с ассемблерным кодом для каждого не ассемблерного входного файла.

По умолчанию, имя файла с ассемблерным кодом делается из имени исходного файла заменой суффикса '.c', '.i', и.т.д. на '.s'.

Входные файлы, которые не требуют компиляции игнорируются.

**-E**

Остановиться после стадии препроцессирования; не запускать собственно компилятор. Вывод делается в форме препроцессированного исходного кода, который посылается на стандартный вывод.

Входные файлы, которые не требуют препроцессирования игнорируются.

**-o файл**

Поместить вывод в файл 'файл'. Эта опция применяется вне зависимости от вида порождаемого файла, есть ли это выполнимый файл, объектный файл, ассемблерный файл или препроцессированный C код.

Поскольку указывается только один выходной файл, нет смысла использовать '-o' при компиляции более чем одного входного файла, если вы не порождаете на выходе выполнимый файл.

Если '-o' не указано, по умолчанию выполнимый файл помещается в 'a.out', объектный файл для 'исходный.суффикс' - в 'исходный.o', его ассемблерный код в 'исходный.s' и все препроцессированные C файлы - в стандартный вывод.

**-v**

Печатать (в стандартный вывод ошибок) команды выполняемые для запуска стадий компиляции. Также печатать номер версии управляющей программы компилятора, препроцессора и самого компилятора.

**-pipe**

Использовать каналы вместо временных файлов для коммуникации между различными стадиями компиляции. Это может не работать на некоторых системах, где ассемблер не может читать из канала, но ассемблер GNU не имеет проблем.

15.11.2021

если хотим создать многомерный массив ,то количество вложенности-это количество " \* "

примеры:

* arr\*\*-двумерный массив
* arr\*\*\*-трехмерный массив

пример кода:

#include<stdio.h>

#include<string.h>

#include<stlib.h>

int main(){

//memory alocation

int \*\*arr = malloc(N\*sizeof(int));

for (int i=0;i<N;i++)

arr[i]=malloc(M\*sizeof(int));

//processing

for (int i=0;i<N;i++)

for (int j=0;i<M;j++)

arr[i][j]=i\*+j;

for (int i=0;i<N;i++){

for (int j=0;i<M;j++)

printf("%d\n",arr[i][j]);

printf("\n");

}

//memory free (обязательно!!!)

for (int i=0;i<N;i++)

free(arr[i];

free(arr);

return 0;

}

* работа с динамическим массивом символьных строк

в char\*\*(матрица) храним char\*(строки) храним char (символы)

* ОТЛАДКА (разобрать самостоятельно)

gcc -g main.c

**ПОБИТОВЫЕ /ПОРАЗРЯДНЫЕ/BITWIZE**

* & - побитовое И
* | - побитовое ИЛИ
* ^ - побитовое исключающее ИЛИ (сумма по модулю 2(одно из двух, НО не оба))
* << - побитовый сдвиг влево
* >> - побитовый сдвиг вправо
* ~ - дополнение (унарная операция)

пример:

* а(операция)b=с

результат "с"- это применение (операции) побитово

* a= 1111111

a<<2

a=1111100

* a=1111111

a>>2

a=0011111

пример кода c &:

#include<stdio.h>

#include<stdlib.h>

#define N 5

#define M 50

int main(){

unsigned char a, b, c;

a=10;

b=7;

c=a&b;

printf("%d\n",c);

return 0;

}

**КР**

* 5 задач
* время 1 пара
* неограниченное число попыток , кроме сборки файлов
* форматный ввод-вывод

(вывести число с плавающей точкой с 5-4 разрядами после запятой)

* сборка программ

((gcc main.c -o qwe ) (заголовочные файлы компилировать не нужно:) ))

* Обработка массива данных
* Многомерные массивы

(int arr[3][7][4][8] на каком расстоянии в байтах находится элемент arr[1][4][2][28])

* Поразрядные операции