Лабораторная работа №8

Итак, начнём. В данной работе я продемонстрирую и расскажу, почему может возникнуть та или иная ошибка на различных этапах компиляции программы.

1. Рассмотрим первых этап - препроцессинг. Здесь происходит преобразование программы для дальнейшего компилирования (работа с препроцессорными директивами). Например, препроцессор добавляет хэдеры в код (**#include**), убирает комментирования, заменяет макросы (**#define**) их значениями, выбирает нужные куски кода в соответствии с условиями **#if**, **#ifdef** и **#ifndef**. Таким образом, ошибка в этом процессе может возникнуть, если указать в программе несуществующую директиву препроцессора или файл:

test.c:2:2: error: invalid preprocessing directive #inlude

2 | #inlude <abc>

| ^~~~~~

test.c:3:10: fatal error: abcdef: No such file or directory

3 | #include <abcdef>

| ^~~~~~~~

compilation terminated.

Узнать, успешно ли преобразуется наша программа на этапе препроцессинга, мы можем следующим образом:

root@DESKTOP-5HM2HTK:~# gcc -E test.c

…

# 3 "test.c"

Ошибок нет – значит всё хорошо (пока что). Теперь преобразуем наш файл:

root@DESKTOP-5HM2HTK:~# gcc -E test.c -o test\_E.c

test\_E.c – результат преобразования файла на этапе препроцессора.

1. Второй этап - работа компилятора. Тут происходит преобразование полученного на прошлом шаге кода без директив в ассемблерный код. Ошибка на данном этапе может произойти, если программа была неправильно написана:

Код test.c:

#include <stdio.h>

int main() {

int i=10

printf("%d", i);

}

root@DESKTOP-5HM2HTK:~# gcc -E test.c -o test\_E.c

root@DESKTOP-5HM2HTK:~# gcc -S test\_E.c

test.c: In function ‘main’:

test.c:5:2: error: expected ‘,’ or ‘;’ before ‘printf’

5 | printf("%d", i);

| ^~~~~~

В данной программе в конце одной из строк тела программы не было знака “;” .

Код test.c:

#include <stdio.h>

int main() {

int i=10;

a=0;

printf("%d", i);

}

root@DESKTOP-5HM2HTK:~# gcc -E test.c -o test\_E.c

root@DESKTOP-5HM2HTK:~# gcc -S test\_E.c

test.c: In function ‘main’:

test.c:5:2: error: ‘a’ undeclared (first use in this function)

5 | a=0;

| ^

В данной программе переменная “a” не была задана.

Правильное преобразование на данном этапе выглядит так:

root@DESKTOP-5HM2HTK:~# gcc -S test\_E.c

root@DESKTOP-5HM2HTK:~# ls

test.c test\_E.c test\_E.s

1. Затем идёт ассеблирование. Ассемблер преобразовывает ассемблерный код в машинный код, сохраняя его в промежуточный файл, хранящий кусок машинного кода. Здесь может произойти ошибка, если, например, указать несуществующую команду на ассемблере:

Код:

#include <stdio.h>

#include <stdbool.h>

int main(){

int i=1;

\_\_asm\_\_("OSHIBKA");

printf("%d", i);

}

root@DESKTOP-5HM2HTK:~# gcc -E test.c -o test\_E.c

root@DESKTOP-5HM2HTK:~# gcc -S test\_E.c

root@DESKTOP-5HM2HTK:~# gcc -C test\_E.s

test.c: Assembler messages:

test.c:5: Error: no such instruction: `oshibka'

Преобразование на данном этапе выполниться успешно, если в изначальном коде ассеблерных вставок не будет вовсе либо они будут соотвествовать языку ассеблера:

Код:

#include <stdio.h>

#include <stdbool.h>

int main(){

int i=1;

\_\_asm\_\_("NOP");

printf("%d", i);

}

Результат преобразвания:

root@DESKTOP-5HM2HTK:~# gcc -E test.c -o test\_E.c

root@DESKTOP-5HM2HTK:~# gcc -S test\_E.c

root@DESKTOP-5HM2HTK:~# gcc -C test\_E.s -o test\_E.o

root@DESKTOP-5HM2HTK:~# ls

test.c test\_E.c test\_E.o test\_E.s

Файл test\_E.o – преобразованный на этапе ассемблирования файл (текст на ассемблере заменён бинарным кодом).

1. Четвёртый этап – линковка. **Линкер** связывает все объектные файлы и статические библиотеки в единый исполняемый файл, который мы и сможем запустить в дальнейшем. Ошибка на данном этапе может возникнуть, например, когда переменная объявлена дважды:

root@DESKTOP-5HM2HTK:~# gcc -E test.c -o test\_E.c

…

int i(int p){

return p\*10;

}

root@DESKTOP-5HM2HTK:~# gcc -S test.c test1.c

root@DESKTOP-5HM2HTK:~# gcc -c test.s test1.s

root@DESKTOP-5HM2HTK:~# gcc test.o test1.o

root@DESKTOP-5HM2HTK:~# ls

test.c test.o test.s test1.c test1.h test1.o test1.s

root@DESKTOP-5HM2HTK:~# gcc test.o test1.o

/usr/bin/ld: /tmp/ccBg3dqJ.o: in function `i':

test1.c:(.text+0x0): multiple definition of `i'; /tmp/cc0rBasI.o:test.c:(.text+0x0): first defined here

collect2: error: ld returned 1 exit status

Содержание файлов:

test.c:

#include <stdio.h>

#include "test1.h"

int i(int p){

return p\*10;

}

int main(){

int p=i(10);

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

}

test1.c:

#include "test1.h"

int i(int p){

return p\*10;

}

test1.h:

int i(int p);

Успешная линковка выглядит так:

root@DESKTOP-5HM2HTK:~# gcc main.c square.c abcsignmod.c minmax.c

root@DESKTOP-5HM2HTK:~# ls

a.out abcsignmod.c abcsignmod.h main.c minmax.c minmax.h square.c square.h

Содержания файлов:

main.c:

#include <stdio.h>

#include <stdbool.h>

#include "minmax.h"

#include "square.h"

#include "abcsignmod.h"

int main(){

int i=1;

int j=-30;

int l=1;

int a,b,c;

for(int k=1;k<51;k++){

if(k>2){

a=i;

b=j;

c=l;

i=max(min(a+b-c-(k-1),a-b+c-(k-1)),min((k-1)+a-b-c,(k-1)-a-b+c));

j=b+mod(c\*sign(b),20)+mod((k-1)\*sign(a),10);

l=abc(a-b+c-(k-1))\*sign(a)\*sign(b);

}

if(is\_in\_area(i,j)==1){

printf("%d %d %d %d\n",i,j,l,k);

return 0;

}

if(is\_in\_area(i,j)==0 && k==50){

printf("%d %d %d %d %s\n",i,j,l,k,"The End");

}

}

}

square.c:

#include <stdbool.h>

#include "square.h"

bool is\_in\_area(int i,int j){

if(((i-10)\*(i-10)+(j-10)\*(j-10))>=25 && ((i-10)\*(i-10)+(j-10)\*(j-10)<=100)){

return 1;

} else {

return 0;

}

}

square.h:

#include <stdbool.h>

bool is\_in\_area(int i, int j);

abcsignmod.c:

#include "abcsignmod.h"

int abc(int a){

if(a<0){

return -a;

} else {

return a;

}

}

int sign(int a) {

if(a>0){

return 1;

} else if(a<0){

return -1;

} else {

return 0;

}

}

int mod(int a, int b){

if(a>=0 || a%b==0){

return a%b;

} else {

return b\*((abc(a)/b)+1)-abc(a);

}

}

abcsignmod.h:

int abc(int a);

int sign(int a);

int mod(int a, int b);

minmax.c:

#include "minmax.h"

int min(int i, int j){

if(i<=j){

return i;

} else {

return j;

}

}

int max(int i, int j){

if(i>=j){

return i;

} else {

return j;

}

}

minmax.h:

int min(int i, int j);

int max(int i, int j);

a.out – результат компиляции файлов.

1. Заключительным этапом является запуск файла с результатом компиляции. Ошибка на данном этапе может возникнуть, если мы, например, поделим число на 0:

root@DESKTOP-5HM2HTK:~# gcc test.c

test.c: In function ‘main’:

test.c:5:5: warning: division by zero [-Wdiv-by-zero]

5 | i=i/0;

| ^

root@DESKTOP-5HM2HTK:~# ./a.out

Floating point exception (core dumped)

Правильный вывод выглядит так:

root@DESKTOP-5HM2HTK:~# gcc main.c square.c abcsignmod.c minmax.c

root@DESKTOP-5HM2HTK:~# ./a.out

8 4 -8 4