السالب :

ازي بيتحسب او بدء ازاي؟؟

char x=1;

معانا متغير 1 بايت نقدر نمثل بيه 255 رقم في الموجب لكن لما نستغل اخر digit علي الشمال most significant bit نخليها تعبر ازا كان الرقم موجب ولا سالب

فمثلا تنفيذ ال1 وال -1 نفسه مع اختلاف الMS bit 0في الموجب و 1 في السالب

طيب جه يمثل الصفر لقي عنده +0 و -0 وده اسمه أي كلام

وكمان لما نجمع +1 وال -1 يطلع بصفر لكن بالطريقه دي طلعت ب-2 وده كمان غلط فالطريقه دي لتمثيل الموجب والسالب متنفعش

فطلعوا بطريقه تانيه اسمها

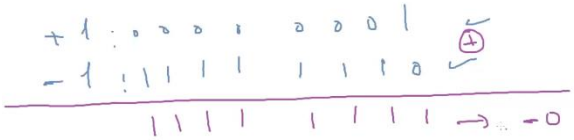
ال1’s compliment

وهي عباره عن عن تنفيذ عمليه ال Bitwise complement للرقم بالبينري

طيب دي شكلها حلو وهتشتغل

للأسف لا لان الصفر الوقتي ليه شكلين

طيب المشكله التانيه اتحلت بانه مثلا لو جمعنا زي



المثال الي فات ال +1 و -1 هيطلع لينا شكل من اشكال الصفر

والي هو ممكن يعتمد عليه في العمليات الحسابيه

طيب الناس علشان ا لطريقه دي فيها عيب قاتل ان الصفر ليه قيمتيمن

طلعوا بطريقه تانيه اسمها 2’s compliment

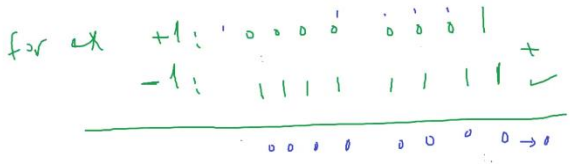
وهي عباره عن ال1+ 1’s compliment

طيب المشاكل اتحلت ؟

نجيب ال0 كده

طلعت نفس الناتج يبقي عندنا تمثيل واحد بس للصفر

طيب وعمليه الجمع

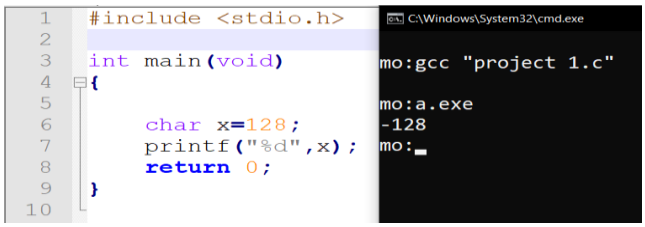


لما جعنا ال1 و ال-1 طلع الناتج صفر

احنا اهملنا الواحد الي علي الشمال لان احنا شغالين 8 بت

وهنا الرينج من -128 لحد 127+ والصفر ليه تمثيل واحد

فرضا جربا الكود ده الناتج هيطلع -128



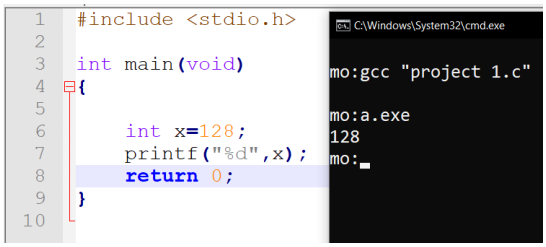
ليه؟

البيناري بتاع ال128 هو كالاتي

1000 0000 لكن احنا هنا في النظام ده ال MS bit بتبقي مسؤوله عن نعرف الرقم ده موجب ولا سالب

طبعا مفيش -0 في الcompliment 2’ لكن التمثيل ده يقابل الرقم -128 في ال8 بت لكن مفيش 128 موجب في ال8بت

وعلشان نثبت ده نجربDatatype تانيه



ملحوظه: لو زودت ارقام اكتر من 255 هيديني ايرور overflow

طيب هو مدنيش ايرور ليه بما ان128 يعتبر overflow بردو

لان ال128 ال MS الرقم الزياده 0 والصفر معناه ان مفيش حاجه

طيب ليه 256 بتعمل ايرور لان فيه 1 والواحد معناها فيها داتا وبالتالي شايف ان فيه داتا زياده عملت overflow للمساحه بتاعه الData type

طيب انا مش عايز موجب وسالب وعايز موجب بس

باستخدام ال key word ← unsigned

وال signed ده علشان اجبره علي رينج معين

وبيطبقوا بس علي ال Char ,int مع ال Data type البتتعامل مع الdecimal

هنا الرينج بتاع السالب كله يروح للموجب ويبقي الرينج من صفر لحد255

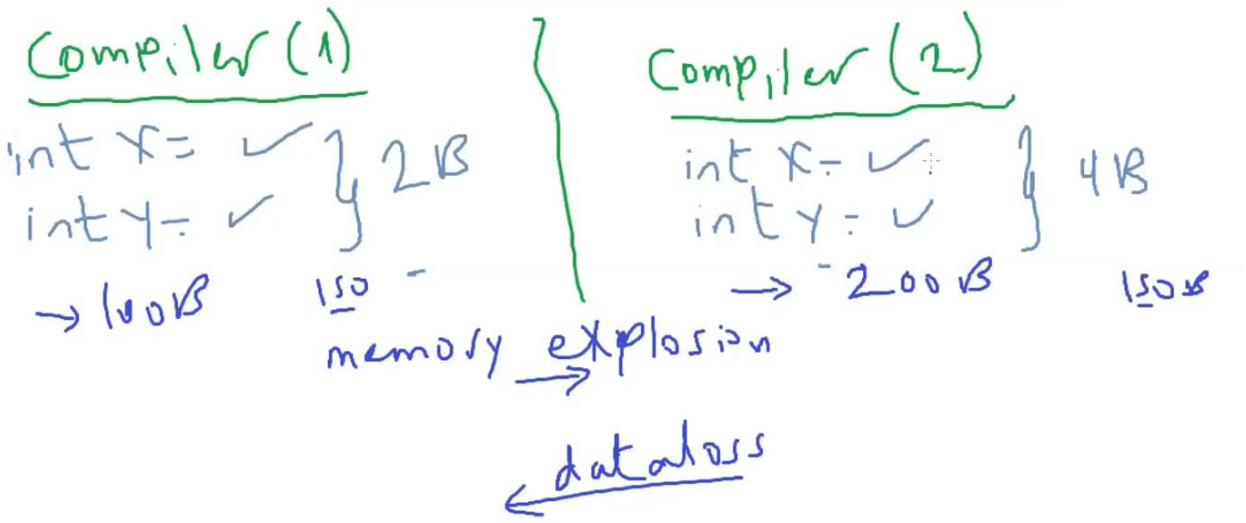
Size modeifer

يعني ايه ان احجام ال Datatype الي اتفقتنا عليها مش لازم تبقي ثابته بس مينفعش تقل عن الرقم ده

يعني ال Char ينفع يبقي 2 بايت عادي😂 ده علي حسب الcompiler

طيب حاجه زي دي تاثر عليا ازاي

ناخد مثال:



احنا في Compiler 1 لو عملنا البرنامج ممكن يشتغل عادي في مساحه معينه

لكن لو نقلنا نفس الكود علي Compiler 2 ممكن المساحه متكفيش

ولو كان الكود شغال علي compiler2 تمام ممكن اخسر داتا لو نقلته علي compiler 1 لان نفس الداتا متخزنه في مساحه كبيره ولو نقلتها لمساحه اصغر لازم اتخلص من الزياده وده هيضيع مني data انا ممكن بكون بستخدمها

احنا نسيبنا من الحوارات دي ونعمل ال Datatype بتاعنا

ال C بتقولنا نعمل كده عن طريق الtypedef

Syntax:

typedef oldtype newtype;

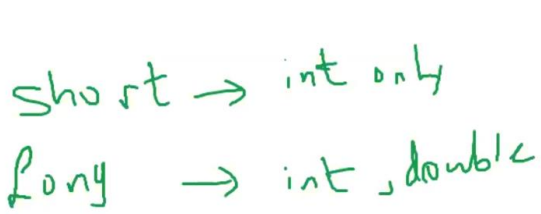
Ex:

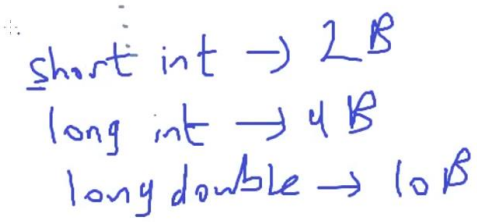
typedef unsigned char u8;

كل مره استخدم u8 في أي حته هيغيرها ويخليها unsigned char

لو رجعنا للمثال الي فات احنا لو نفذنا الكلام ده

هنغير لمساحه اصغر في Compiler2 تتناسب في المساحه مع compiler 1 في سطر واحد بس





دي ثابته في كل الcompiler

قواعد مهمه :

1. الترتيب مش مهم

يعني: unsigned short inr == short unsinged int

2. مش كل المودفير بيستخدموا مع كل ال datatype عندك ال Short بتيجي مع ال int بس وال long بيجي مع ال , int double
والunsigned , signed بيجوا مع ال int ,char بس
3. متستخدمش اتنين مودفير عكس بعض مع بعض
يعني short long int ده اسمه ايه كلام

هنعمل الِ standr type بتاعتنا : ده علي حسب انا عايز ايه واحتياجاتي

انا هعمل ال

u: unsigned , s:signed ,f:float

طيب احنا كده اتفقنا علي الأساسيات

طيب عايزين نتاكد من ال size لكل داتا طيب في الكومبيلر الي شغالين عليه

وهي الsizeof دي مش فانكش دي operator زيها زي ال shift left كده

```c
#include <stdio.h>

int main(void)
{


    printf("int is :%d\n",sizeof(int));
    printf("char is :%d\n",sizeof(char));
    printf("double is :%d\n",sizeof(double));
    printf("float is :%d\n",sizeof(float));
    printf("short int is :%d\n",sizeof(short int));
    printf("long int is :%d\n",sizeof(long int));
    printf("long long int is :%d\n",sizeof(long long int));
    printf("long double is :%d\n",sizeof(long double));
    return 0;
}

```

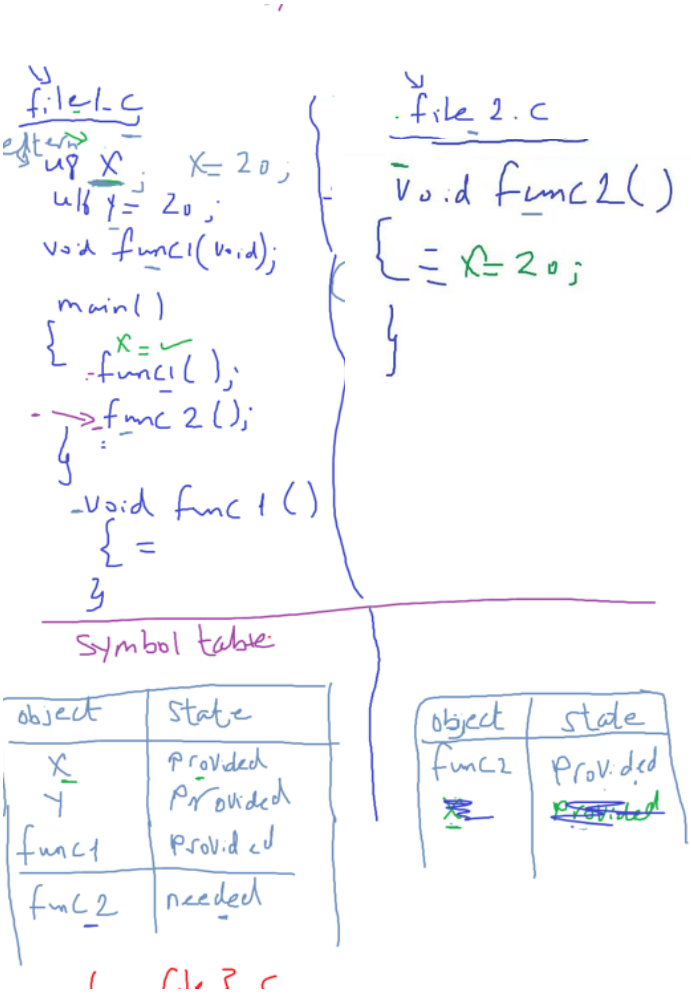
```
C:\Windows\System32\cmd.exe

mo:gcc "project 1.c"

mo:a.exe
int is :4
char is :1
double is :8
float is :4
short int is :2
long int is :4
long long int is :8
long double is :12

mo:
```

نجربها لانها ممكن تختلف ممكن نستخدمها عادي بدل المتغيرات

Building process:

هي ان الملف بيبقي file.c ← بيدخل علي ال toolchan بيتعملها حاجه اسمها preprocessor بيطلع منها ملف اسمه ← Intermideate file ← بيدخل علي الcompiler بيطلع منه الAssembly file ← بيدخل علي ال Assembler بتشيل أوامر الAssemble وتحاولها لصفر وواحد بيطلع ملف اسمه object file ← بيدخل علي حاجه اسمها ال linker وبيطلع الExe ال compilerكمان بيطلع حاجه اسمها ال symbol table

طيب ناخد سيناريو



فرضا عندنا ملفين في بروجكيت زي ما هو موضح

ال compiler لما بشتغل بيشتغل كل واحد لوحده

يعني مش بيبقي شايف file1 وهو شغال في file2

طيب انا معرف func2 في file2 ومستخدمها في file1

هيشوفوا بعض ازاي

هنا الكومبايل بيستخدم الsymbol table بيمسك كل ملف وبيعمله جدول بكتب فيه ايه الobject الي متعرف فيه وايه ال objedct الي مش متعرف فيه

بعد كده بتيجي فايده ال linker الي شايف البروجكيت كله بيعمل object verification

هي عباره عن خطوتين

1. يتاكد ان كل ال objects تبقي provided في نفس الملف او الملفات تانيه في نفس البروجكيت
   غير كده هيطلعلك linker error
   الي هي بتظهر undefined reference
2. يتاكد ان كل object معموله provided سواء في نفس الملف او ملف تاني في نفس البروجكيت انه يكون معموله provided مره واحده بس
   Multiple definition error

طيب احنا مثلا ال x واضحه global في الملفين هل ممكن استخدم ال x الي متعرفه في file1 استخدمها علطول في file2 زي ما معمول بالاخضر كده؟

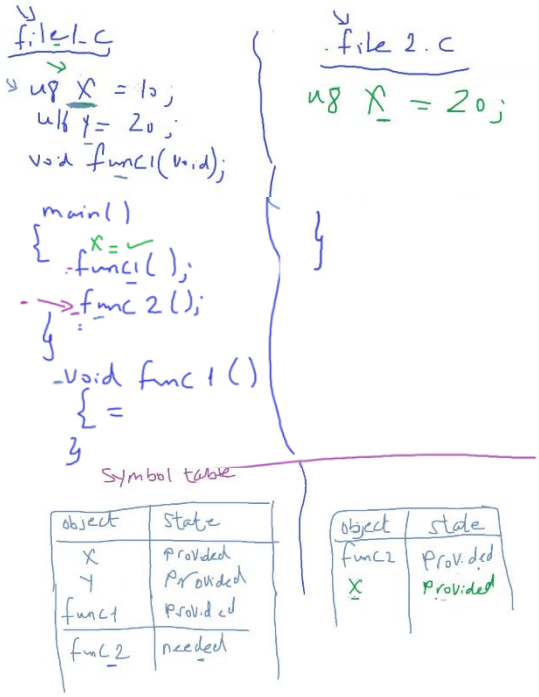
لا مش علطول كده في key word هنستخدمها الي هي Extern دي بنستخدمها : كاننا بنقوله في متغير تاني هو global في ملف تاني انا عايز استخدمه .

ال syntax :

```
extern u8 x;  //file2
```

طيب انا لو سبتها منغير ال extern كان هيعملي ايرور بس علشان هو needed في file2 بس متعملوش provide

هتقولي ازاي ما هو متعرف في الfile1 هقولك ماشي دي كانت بتظهر من ال linker لكن هنا الي عمل ال error مش ال linker الي عمل ال error هو ال compiler لانه شافك بتستخدم متغير منغير ما تعرفه لكن لو قولت extern كانك بتقوله اصبر وال linker هو الي هيحلها



طيب مشكله تانيه

انا عرفت الاتنين x u8 وجلوبل

كده الكومبيلر عمل اتنين x حالتهم provided

والمفروض حسب كلامنا ان ال linker يعترض نحل مشكله زي دي ازاي؟؟

فيه key word بتخلي الx مينزلش في ال sympol table أصلا ويبقي ال X مقصوره بس علي الملف الي هي فيه ← Static

لان وظيفه ال Sympol table انه يسلم ويستلم ال objects الي ممكن تتنقل من الفايلات لبعض ومعني اني مخلتش فيه x تبقي needed او provided ده مش هيعمل مشكله مع ال linker وال x تترجم من ال compiler بس وتشتغل عادي في الملف بتاعها بس

وال Static كده بتخلي ال Scope بتاع ال gloabl varible يبقي من ال programe scope يتنقل يبقي file scope

ونتذكر ان Static لما اتستخدمناها مع الfunction حولنا بيها ال life من function life لprogram life يعني بتكمل بعد ما ال function بتنتهي او بتموت.

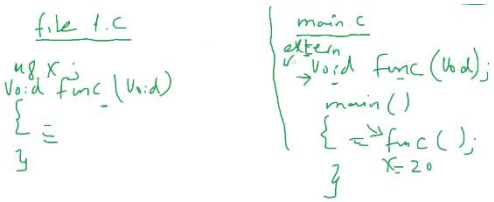
لو حاولت استخدم extern مع static هيعملي error لانه مش هينزل في الSympol table لانه ميعرف المتغير ده ايه

و مينفغش اعرف متغير واعلمه extern واعمله assigne باي Value في نفس السطر ممكن اديله value السطر الي بعده

طيب لزمه ايه الgloabl variable وهو مش عارف يتحرك بين الفايلات : علشان يبقي متشاف في كل ال functions الموجوده في الملف بس

ونفس الفكره في الextern

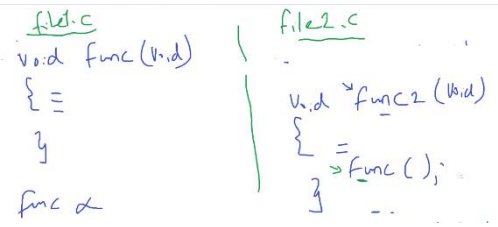
بس مبكتبش extern 😁



هي تنفع بس انا لو عملت ال prototype في الأول تغني اني اكتب extern

بما ان الي بيحصل لل Variable بيحصل لل function

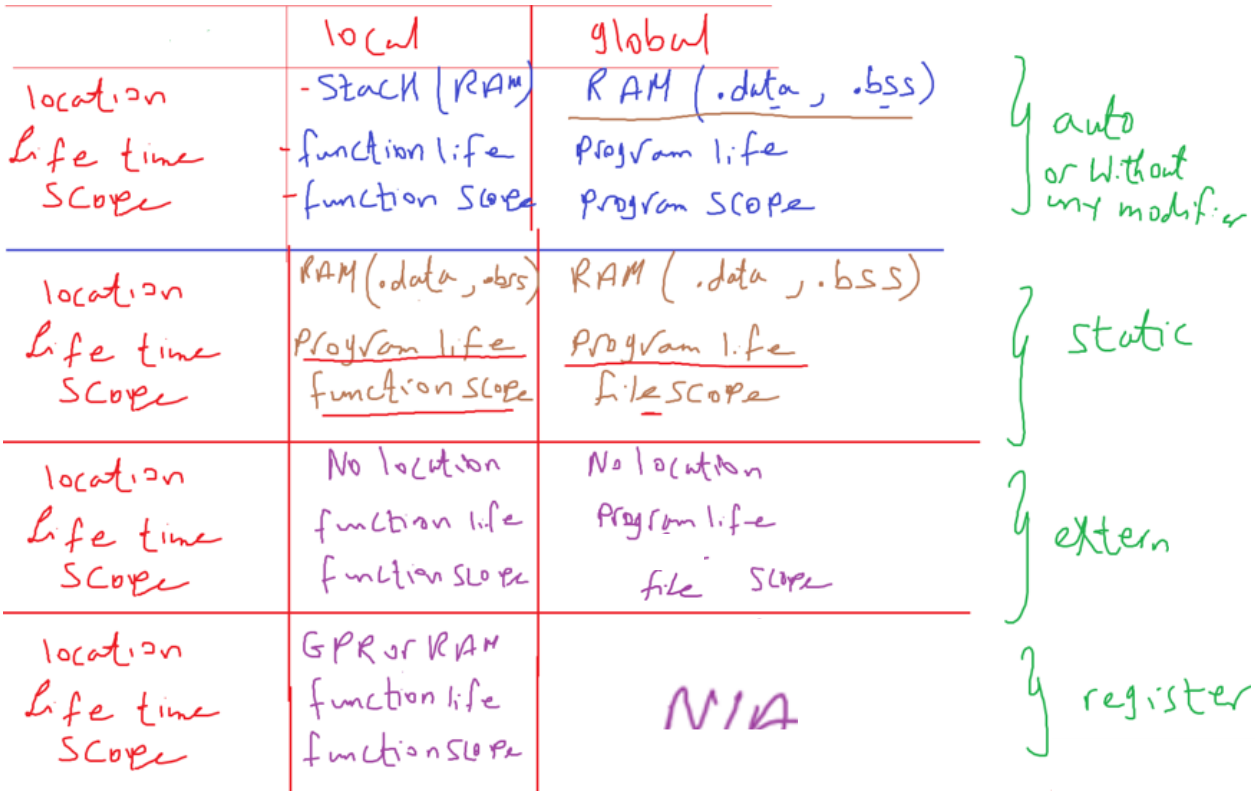
يعني لو عملت Static ل function مش هينفع استخدمها في ملف تاني لنفس الأسباب



ساعات ال compiler لو معرفتش الprototype في الأول و استخدمت الفانكشن بيبعت wrninng للlinker وبيقوله شوف لو فيه حاجه كده لها نوع function ونوعها حسب النوع المتعرف و النبوت بتاعه نفس الانبوت الي انا مستخدمه لو لقاه هو الي هيعرف ال decleration لوحده ويبعتلك Wrninng

وفيه compiler تانيه تديك error علطول

الفرق بين ال gloable وال local variables



الRegister هي عباره عن جزء من الميموي سريع قريب من البروسيسور

فيه جزء منهم اسمه الGPRS دول registers ممكن اليوزر يهملهم Access وبيستخدم الcpu في العمليات الحسابيه

الsyntax

Register u8 x;

معانها اني بقترح علي البرسيسور ممكن تخزن القيمه دي في الريجيسترات

البرسيسور مش هيرد عليا ممكن يوافق وممكن لا

هيقبل او يرفض بناء علي شويه شروط ال life time لو local ممكن يقبله و ال Size لو صغير ممكن يتقبل لو بستخدمه كتير ممكن يتقبل

Const modifer

Promise not change variable value

Syntax : ex:

const u16 x=10;

x=20; //compilre error

const with pointers:

```
int *const ptr = &x;
//constant pointer to variable data
int const *ptr =&x;
//variable pointer to constant data
int const *const ptr=&x;
//constant pointer to constant pointer
```

في حاجه في ال C اسمها Constant variable hacking

```
Constant int x =20;
X= 30; //compiler error
```

لو استخدمت البوينتر ينفع اغير الداتا

```
Int *ptr=&x;
*ptr=30;
```

بس بشروط تبقي local لكن لو كانت global مش هتقدر تغيرها بس مش هيطلع ايروور

وده بيرجع لل toolchane ممكن متشتغلش معاك