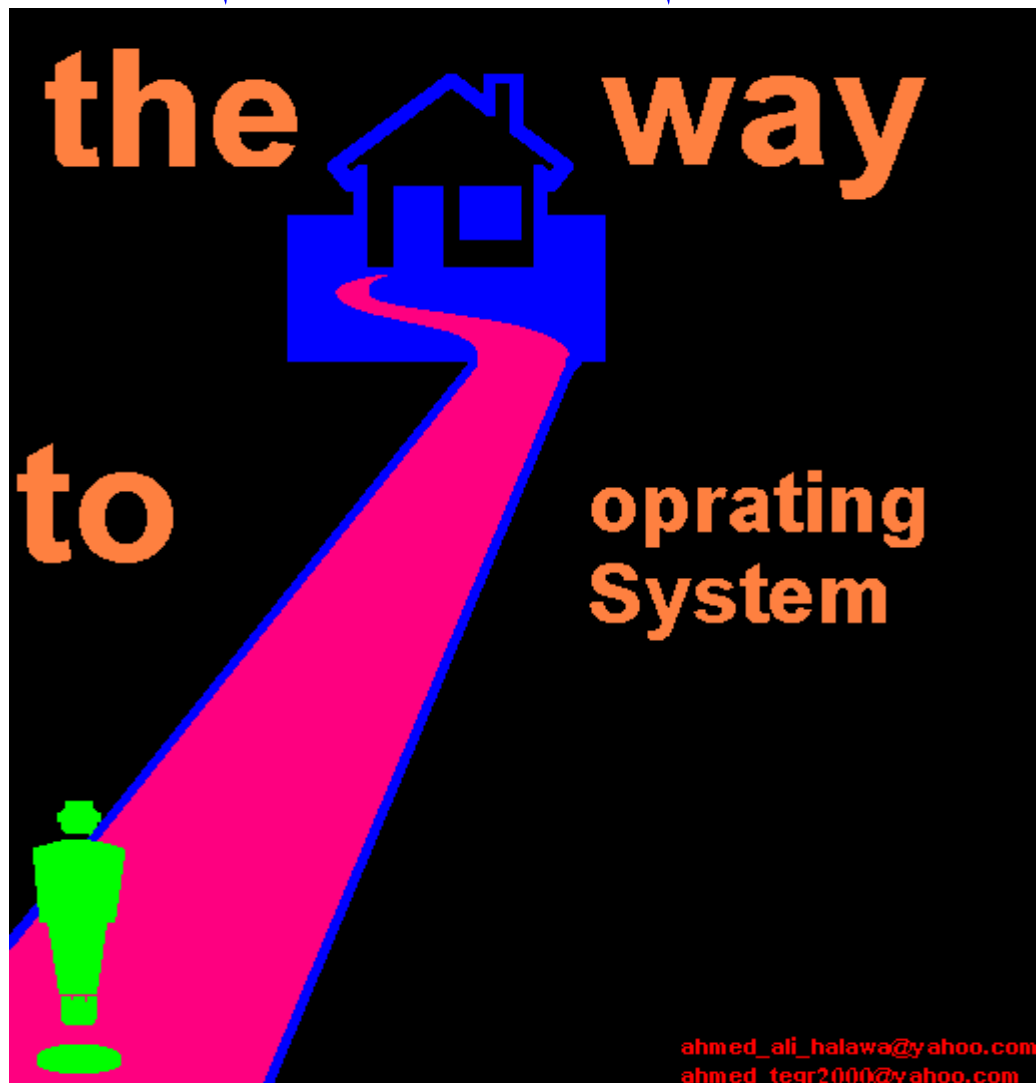


بسم الله الرحمن الرحيم



الدرس الثالث: "كتابه النواة بلغه السي/سي++"

الصفحة	العنوان
٢	١- نبذه عن البرمجة المتفرقة.....
٤	٢- ملف الربط Link.ld
٦	٣- ملف الترجمة Makefile
٧	٤- بنية الملف التنفيذي Elf
٧	٥- تحميل النواة ببرنامج Grub
٨	٦- ملف الإقلاع boot.asm
٩	٧- ملف التعريفات system.h
١٠	٨- ملف التعريفات system.cpp
١٢	٩- نبذه عن الشاشة النصية
١٥	١٠- ملف الشاشة النصية video.h
١٦	١١- ملف الشاشة النصية video.cpp
٢٠	١٢- ملف جدول المداخل Gdt.h (Gdt)
٢١	١٣- ملف جدول المداخل Gdt.cpp (Gdt)
٢٣	١٤- ملف التنفيذ الرئيسي main.cpp

◀ نبذه عن البرمجة المتفرقة ▶

معني البرمجة المتفرقة؟

هو إمكانية استخدام إمكانيات برنامج في برنامج آخر مثل مكتبات آل Dll في ويندوز وبالنسبة للغة السي/سي++ ملفات آل Header files ففيها يتم كتابه شكل الدالة الأولي وكود الدالة في ملف آخر

لماذا مستخدم البرمجة المتفرقة في نظم التشغيل؟

لان هناك ٦٠% من الدوال لا يمكن تنفيذها داخل ملف c++ / c بل لابد ترجمتها عن طريق مترجم الاسمبلي مثل ملف الإقلاع boot.Asm

كيف يتم استخدامها داخل لغة السي/سي++؟
عن طريق التعليمة التالية

C/C++ Code
extern "C" vertype vername ;

لابد وأنك لا تفهم الأمر حسننا

الأمر Extern معناه قم بوضع أسم المتغير Vername داخل ملف object file عن الترجمة لابد وانك تتذكر خطوات ترجمه كود للغة السي/سي++

C++/C → Object file → Exe file

ما الغرض من وضع الاسم داخل ملف Object file ؟ حتى يمكن للبرامج الأخرى استخدام هذا المتغير أو الدله كيف يمكن استخدامه ولم يتم ترجمه الملف لل Exe file ؟ حسنناً لا بد وانك قد قرأت في الفهرس Link.LD هذا الملف يحدد للمترجم كيفية ربط الملفات ببعضها حسننا ابسط الأمثلة
قم بفتح ملفين جديدين الأول اسمه 1.cpp والثاني 2.cpp
قم بكتابه الكود التالي في الملف الأول

C/C++ Code	2.Cpp	C /C++ Code	1.Cpp
extern "C" void _alloca(){}; extern "C" void __main(){} ; extern "C" int add(int a,int b) { return a += b; }		extern "C" int add(int a,int b) ; int main() { int x ; x= add(5,5); return 0 ; }	

قم بترجمتهما عن طريق الأمر التالي في برنامج Cygwin

Cygwin instruction
Gcc -c x:/folder/1.cpp -o x:/folder/1.cpp Gcc -c x:/folder/2.cpp -o x:/folder/2.cpp Ld x:/folder/1.cpp x:/folder/2.cpp -o x:/folder/out.exe

شرح الأوامر السابقة

Gcc -c x:/folder/1.cpp -o x:/folder/1.cpp

Gcc أسم المترجم المستخدم
-c تمنع المترجم من الوصول للمرحلة الأخيرة

c/c++ → object file

أي المرحلة الأولى من الترجمة

x:/folder/1.cpp العنوان بالكامل للملف الأول (ملف الكود) 1.cpp
x:/folder/1.cpp العنوان بالكامل للملف الأول (الملف المنتج) 1.cpp
الأمر الثاني كسابقه

Ld x:/folder/1.cpp x:/folder/2.cpp -o x:/folder/out.exe

LD برنامج الربط

x:/folder/1.cpp x:/folder/2.cpp اسم الملفات المراد ربطها
x:/folder/out.exe الملف النهائي Exe file وبهذا قد أتممنا مراحل الترجمة كاملة

بقي لنا المرحلة الأخيرة معرف ناتج الأوامر السابقة
عن طريق الأمر التالي

Cygwin instruction
Objdump -D x:/folder/out.exe

سوف يقوم هذا الأمر بإظهار كود الاسمبلي للملفين السابقين x:/folder/1.cpp x:/folder/2.cpp تتبع أوامر الاسمبلي سوف تجد أن الدالة _main وليس _main__ تنادي الدالة _add

ربما قد تكون فهمت الجزء الأول " البرمجة المتفرقة " بهذا المثال البسيط.

◀ ملف الربط Link.ld ▶

تلاحظ في المرحلة السابقة اننا قد قمنا بربط الملفين ببعضهما ولكننا لم نتحكم بخصائص الربط !!!
ما هي خصائص الربط؟

خصائص الربط الذي اقصدتها كالتالي

١ - من المعروف لدي مبرمجي سي/سي++ أن كل برنامج يعمل في عنوان في الذاكرة برنامج الربط إن لم نقوم بوضع العنوان الخاص بنا فإنه يقوم بوضع عنوان افتراضي يقوم الويندوز أو نظام التشغيل بتغييره حسب رغبته وقد تلاحظ أنني قلت عنوان افتراضي وليس حقيقي إذاً هذا العنوان ليس صالح لنا فنحن لم نجعل النواة تستخدم العناوين الافتراضية بعد إذاً فلا بد لنا من تحديد عنوان حقيقي يعمل فيه برنامجنا.

٢ - يقوم برنامج الربط بوضع أوامر تلك الأوامر تستخدم خدمات الويندوز ولكن في حالتنا نحن لا يمكننا استخدام خدمات ويندوز لأن البرنامج سيعمل في مكان حيث لا ويندوز ولا لينكس ولا
الذي كتبته في الأعلى جزء بسيط من الخصائص التي نريدها.

سؤال كيف اعدل تلك الخصائص؟

عن طريق ملف الربط Link.ld سوف اقوم بوضع ملف ربط جاهز يمكنك تعديله كما تشاء

Cygwin instruction

```
ENTRY(start)
SECTIONS
{
    . = 0x00100000;
    code = .;
    .text
    . :{
        *(.text*)
    }
    . = ALIGN(4096);
    data = .;
    .data
    . :{
        *(.data*)
    }
    bss = .;
    .bss
    : {
        *(COMMON)
        *(.bss*)
        . = ALIGN(4096);
        *(.stack)
    }
    end = .; _end = .;
/DISCARD/ :
{
    *(.note*)
    *(.indent)
    *(.comment)
    *(.stab)
    *(.stabstr)
}
}
```

شرح الأوامر
في البداية تجد الأمر ENTRY(start) يقوم هذا الأمر بتحديد نقطه البداية والافتراضي لبرنامج الربط هو الدالة main الخاصة بلغة السي/سي++ وقد قمنا بتحديد نقطه البداية بالدالة start

معناها تحديد خصائص الأقسام :	SECTIONS
متغير خاص يشير إلي العنوان الحالي في الذاكرة :	النقطة .
تعني جعل العنوان الحالي ب 0x00100000 :	. = 0x00100000
code متغير نستخدمه لمعرفة عنوان البداية الخاصة بالبرنامج(يستخدم عن طريق البرنامج وليس برنامج الربط)	code = .;
	.text
	. :{
جعل الرابط يقوم بضم الشفرة إلي البرنامج و المعلومات والمكدس بنفس لطريقة	*(.text*)
	}
	end = .;

_end = .; : _end متغيران مثل code ولكن يشيران إلى نهاية النواة

/DISCARD/ : جعل برنامج الربط يهمل بعض الأشياء مثل الملحوظات

بهذا يكون ملف الربط قد اكتمل

► ملف الترجمة Makefile ◀

اولاً اريد سؤال واحد وهو ماذا تفعل إذا كنت تريد ترجمه هذه الملفات وربطهم وإظهار كود الأسمبلي الخاص بهم ؟ 🤖

boot.cpp main.cpp system.cpp video.cpp gdt.cpp idt.cpp exptions.cpp\
keyboard.cpp timer.cpp isr.cpp isrs.cpp pic.cpp phmman.cpp phmem.cpp\
page.cpp
لا بد وانك ستتعب وتمل كل مره تقوم بتغيير شئ تعيد كل ما سبق .. لا تقلق هناك حل افضل وهو استعمال ملف Makefile فكل ما ستقوم به هو اضافه اسماء الملفات فقط وهناك برنامج سيتكفل بكل تلك التكرارات وهو برنامج Make ولإستخدامه انظر الأمر التالي

Cygwin instructions

Make -C x:/folder -f Makefile

X:/folder مسار ملف ال Makefile
اسم الملف Makefile
-C أمر معناه انتقل إلى المكان الحالي

سؤال آخر ما محتوى ملف ال Makefile

Cygwin instructions

CPPFLAGS = -c -fno-builtin -fno-rtti -fno-exceptions -WALL

.PHONY: clean

OBJS = boot.o

all: kernel.elf

kernel.elf: \$(OBJS) link.ld

```
ld -T link.ld $(OBJS) -o kernel.tmp
objcopy -O binary kernel.tmp a:/kernel.bin
objdump -M intel -D kernel.tmp > debug.txt
$(RM) kernel.tmp
$(RM) $(OBJS)
```

```
%.o: %.cpp
    g++ $(CPPFLAGS) $< -o@$
%.o: %.s
    nasm -f elf $< -o@$
%.o: %.asm
    nasm -f elf $< -o@$
```

متغير يحتوي علي تعليمات خاصة بالمترجم Gcc CPPFLAGS
متغير يحتوي علي أسماء الملفات المراد ترجمتها قم بإضافه إي ملف جديد قمت بكتابته OBJS
في هذه المنطقة ليتم ترجمتها بشرط ان تنتهي ب o. وليس ب .c
اسم ومسار ملف النواة الذي سيتم الترجمة إليها a:/kernel.bin
اسم الملف الذي يحتوي علي تعليمات الأسمبلي الخاصة بالبرنامج أو النواة debug.txt
اسم ومسار النواة علي القرص المرن أو القرص الوهمي المرن a:/kernel.bin
بقية الأوامر صعب شرحها أو بمعنى آخر لن تحتاجها ابداً

بهاذا نكون قد انهينا جزء آل Makefile

► بنيه الملف التنفيذي Elf ◀

سؤال بسيط قبل البدايه بشرح بنيه elf لماذا لم نستخدم exe. مثلاً ؟
لانه بنيه exe. الخاصه بويندوز تحتوي علي معلومات كثيره لا نحتاجها + لا يمكن تحديد الاقسام التي نريدها فيها

تتكون بنيه Elf من التالي

```
.text : قسم الكود او الشفرة
.data: قسم البيانات
.bss: قسم المكسد
.rodata: البيانات المقصوده هنا هي printf("xxxz"); بيانات للقرأه فقط مثل xxxz
.note
.comment: ملحوظات توضع بواسطه المترجم
.stab
.stabstr: تحتوي علي اسماء خاصه تستخدم عند إظهار كود الاسمبلي الخاص بالبرنامج
```

حاول مقارنة البنية بملف الربط

► تحميل النواة بواسطة Grub ◀

- Grub** برنامج تحميل متعدد المهام ويمكن تحميل اكثر من نظام تشغيل بواسطته يتم استخدامه بكثرة من قبل مبرمجي انظمه التشغيل ومن قبل نظام التشغيل Linux من مميزات Grub:
- ١- امكانيه لاتحميل من دعامات (Drivers and file systems) مختلفه.
 - ٢- دعم العديد من انظمه الملفات مثل Elf,Aout,Pe-i386,pei-i386,binary
 - ٣- دعم ملفات مضغوطة ب Gzip
 - ٤- دعم انظمه الملفات المختلفه fat,fat32,ext,...

- ولكي نقوم بتحميل نواتنا من قبل Grub لا بد من توفر بعض الشروط
- ١- يجب ان تكون النواه معده للعمل علي بيئه ٣٢ بت في النمط المحمي
 - ٢- علي النواه ان تحتوي علي مجموعه من المعلومات علي هيئه محدده وإلا لن يتم تحميل النواه
 - ٣- علي هذا الرأس (المقصود تلك المعلومات) ان يكون موجود في اول ٨ كيلوبايت

مكونات الرأس:

الرأس يحتوي علي معلومات إجباريه يجب ان تقوم بوضعها حتي يمكنك Grub من تحميل النواه وحقوق لإختباريه تضعها كما تشاء

Grub information			
البت رقم	إسم الحقل	وصف الحقل	إختباري
0	كلمه السر	كلمه سر خاصه ب Grub يتم وضعها لكي تخبر Grub ان النواه معده للعمل مع Grub	لا
4	البيانات	إخبار Grub بالمعلومات التي يجب توفيرها لنواتنا	لا
8	مجموع التحقق	عدد من ٣٢ بت إذا تم إضافته للحقليين السابقين يكون المجموع ٠	لا
12	عنوان الرأس	العنوان الذي يجب نقل هذا الرأس إليه في الذاكره	إذا كان البت ١ = ١٦
16	عنوان التحميل	عنوان فقره الشفره او الكود (راجع ملف الربط)	
20	نهايه النواه	عنوان نهايه فقره البيانات (راجع ملف الربط)	
24	عنوان نهايه Bss	أي عنوان نهايه مساحه المتغيرات	
29:44	اوضاع الفيديو	توفير اوضاع الفيديو المناسبه لنواتنا	

بالنسبه لحقل المعلومات كل ما يهمنا فيه هو

البت : الوصف

١ = ٠ :

١ = ١ : إمدادنا بالمعلومات الاساسيه مثل حجم الذاكره ورمز قرص الإقلاع إذا كان الهارد او القلوبي
دسك

بعد تحميل النواه يتم تحميل المسجل ebx بعنوان البيانات التي طلبناها

◀ ملف الإقلاع Boot.asm ▶

ملف الإقلاع هو الملف الذي سيقوم بتشغيل نواتنا (ليس الإقلاع الذي تقصده بل الملف الذي سيحمل من قبل Grub ربما تفهم إن اكملت)

محتوي ملف Boot.asm

Kernel code	
%define magic 0x1BADB002	نرجو عدم تغييرها حتي نتمكن من Grub كلمه السر الخاصه ب ; شغيل النواه
%define flags 0x00010003	المعلومات التي يجب توفيرها لنا ;
%define chkasm -(magic+flags)	مجموع التحقق ;
extern end,code,data,bss,_Hmain	بعض البيانات المطلوبه من ملف الربط ;
[global start]	object file إضافه عنوان البدايه إلي ملف ال ;
[bits 32]	Grub جعل ملف الإقلاع يعمل بحجم ٣٢ بت احد شروط ;
Ghead:	Grub الرأس الخاص ب ;
dd magic	كلمه السر (راجع الجدول السابق) ;
dd flags	البيانات (راجع الجدول السابق) ;
dd chkasm	مجموع التحقق (راجع الجدول السابق) ;
dd Ghead	عنوان الرأس (راجع الجدول السابق) ;
dd code	عنوان التحميل (راجع الجدول السابق) ;
dd bss	عنوان نهايه قسم البيانات او المتغيرات ;
dd end	عنوان نهايه النواه ;
dd start	عنوان التنفيذ الفعلي للنواه ;
start:	
mov esp,stack	يشير إلي مساحه المكس esp جعل ;
push ebx	Hmain إرسال عنوان البيانات التي طلبناها إلي الداله ;
*****;	
Jump \$	القفز إلي نفس المكان بغرض توقف النواه ;
*****;	
SECTION .stack	قسم المكس ;
times 0x4000 db 0	مساحته ١٦ كيلوبايت ;
stack:	

قم بحفظ الملفات السابقه (boot.asm + Makefile + link.ld) في مكان واحد قم بتشغيل قرص وهمي ثم
بفتح برنامج Cygwin
قم بكتابه الامر التالي

Cygwin information
Make -C x:/folder -f Makefile

X:/folder هي اسم ال PRATION او الملف الموجود فيه ملف ال Makefile
إذا لم يظهر لك برنامج ال Makefile قم بتشغيل برنامج الجهاز الوهمي الذي لديك سيفتح برنامج Grub قم
باختيار Halawa tutorial قم بالضغط عليها إذا ظهر لك عدده معلومات مثل نوع ملف النواه وعنوان
التحميل ثم اختفت ولم تظهر لك أي شئ بعدها فـ

مبروك لقد انجزت المهمه الأولى من الدرس وهي الإقلاع
بواسطة جروب

الان لنكمل ولنقم بتطبيق ما قلناه من قبل اول شئ وهو البرمجه المتفرقه لكي نقوم بقوم بتشغيل النواه بواسطه
السي/سي++ قم بالبحث في ملف boot.asm عن التالي

Extern end,code,data,bss,_Hmain

سوف تجد في نهايه السطر كلمه _Hmain ماهي تك الكلمه الذي نريدها في نواتنا ؟ تلك الكلمه هي تمثل العنوان الفعلي لنواتنا بواسطه السي/سي++ سوف تفهم ما اقصده إنشاء الله اكمل معنا.

ملحوظه قبل ان تكمل معنا قم بتعدي هذه الفقره في ملف ال boot.asm
jmp _Hmain ب \$

◀ ملف التعريفات system.h ▶

ملف التعريفات تاتي محتويات هذا الملف من اسمه فهو يحتوي علي معظم التعريفات التي تحتاجها نواتنا ومحتويات الملف كالتالي

```
System.h
#ifndef _H_system_
#define _H_system_

typedef unsigned int    uint           /** متغير بحجم ٣٢ بت */
typedef unsigned char   uchar         /** متغير بحجم ٨ بت */
typedef unsigned short  ushort        /** متغير بحجم ١٦ بت */

// داله لنسخ قيمه من مكان ما الي مكان اخر
void memcpy(uchar *dest , uchar *src,int count);
// داله لجعل مكان ما في الذاكره = قيمه معينه
void memset(uchar *dest, uchar val ,int count);

int  strlen(const char *txt); // حساب طول نص ما
void strcpy(char *f,char *s); // نسخ نص في متغير الي متغير اخر
//*****//

static inline uchar inb(ushort port) // في الاسمبلي in داله لإدخال قيم من منافذ معينه =
{
    uchar val;
    asm volatile("inb %1,%0" : "=a"(val) : "dN"(port));
    return val;
}

static inline void outb(ushort port,uchar val) // في الاسمبلي out داله لإخراج قيمه الي منفذ ما =
{
    asm volatile("outb %1,%0" :: "dN"(port), "a"(val));
}

//*****//
/* Grub البنيه الذي ستحمل قيم المعلومات التي سيعطيها لنا */
typedef struct multiboot
{
    ulong mflags;
    ulong memlow;
    ulong memhig;
    ulong bootde;
}multiboots;
```

في البدايه هناك شرط يختبر ما إذا تم تعريف هذا الملف من قبل (برنامج الربط هو من يختار هذا الشرط حتي لا يتم تكرير التعريف اكثر من مره ويتم الحصول علي قيم مختلفه لشئ واحد) الطر الموالم يقوم بالتعريف في حاله لم يتم التعريف .

الثلاث اسطر الموالميه هدفها تعريف ٣ انواع من المتغيرات بأحجم معينه وأظن ان بقيه الاسطر مفهومه

► ملف التعريفات System.cpp ◀

لابد وانك قرأت الفقره السابقه ولاحظت اننا نضع التعريفات الاوليه لبعض الدوال ولم نضع أي كود يقوم بعمل تلك الدوال لذا سغو نقوم بكتايه ملف اسمه system.cpp ما يحتويه هذا الملف

System.cpp

```
#include "system.h" // التي قمنا بصنعها system.h ضم المكتبه

void memcpy(uchar *dest , uchar *src,int count) // كود الداله الاولى
{
    int I;
    for(i=0;i<= count;i++)
    {
        dest[I] = src[i];
    }
}

void memset(uchar *dest, uchar val ,int count) // كود الداله الثانيه
{
    int I;
    for(i=0;I <= count;i++)
    {
        dest[i] = val;
    }
}

int strlen(const char *txt) // كود الداله الثالثه
{
    int i;
    for(i=0;*txt != 0;*txt++)
    {
        i++;
    }
    return i;
}

void strcpy(char *f,char *s) // كود الداله الرابعه
{
    for(int i=0;i <= strlen(f);i++)
    {
        s[i] = f[i];
    }
}
```

شرح كود الداله الأولى

الغرض من هذه الداله هو نسخ قيمه من عنوان معين إلي عنوان اخر بعدد التكرارات مثال:

نسخ قيمة العنوان 0x1000 إلى العنوان 0x2000 بعدد تكرارات ٤
الحل:

$$0x2000 = 0x1000$$

$$0x2001 = 0x1001$$

$$0x2002 = 0x1001$$

$$0x2003 = 0x1002$$

ربما تكون قد فهمت المقصود من هذا الدالة من المثال السابق
في البدايه نقوم بتعريف متغير اسمه I
في السطر الموالي نقوم بدوره بعدد التكرارات المعطاه لنا
هدف دوره هو جعل العنوان رقم I = القيمة الموجود في العنوان رقم I

شرح الدالة الثانية :

نفس الشرح السابق ولكن بدلاً من نسخ قيمة من عنواننا ما نقوم بجعلها تساوي قيمه.
مثال:

$$\text{جعل قيمة العنوان } 0x20 = 0x1000 \text{ بعدد تكرارات ٤}$$

$$0x1000 = 0x20$$

$$0x1001 = 0x20$$

$$0x1002 = 0x20$$

$$0x1003 = 0x30$$

شرح الدالة الثالثة :

الغرض من هذه الدالة هو حساب طول معين ومن المعروف لدي مبرمجي سي /سي++ ان أي ينتهي
بالرقم صفر إذا كنت قد فهمت هذه الجملة فحاول فهم الكود بنفسك بدون مساعدته الشرح اولاً.

في البدايه نقوم بتعريف المتغير I

نقوم بدوه حتي نصل لنهايه النص

كل دروه يقوم عداد الطول بزياده نفسه رقم واحد

مثال:

احسب طول النص التالي Hello world

في الدورة الاولى يختبر ما إذا كان الحرف الحالي = ٠ ام لا طبعا الحرف الحالي != ٠ فنقوم بزياده
المتغير I عن طريق I++ ثم يعود لإختبار الحرف التالي إذا كان = ٠ ام لا طبعا الحرف != ٠ فيقوم بزياده
المتغير I عن طريق I++ ويكر هذا كله حتي يصل إلي نهايه النص.

شرح الدالة الرابعة :

هي تقريباً ك الدالة الأولى

ملحوظه في برمجته انظمه التشغيل نحن بحاجة إلي صناعه كل شئ بأنفسنا من قرأه وكتابه.

قم بإضافه system.o إلي ملف Makefile (إذا لم تكن تعرف اين تضيفه فراجع جزء ال Makefile)

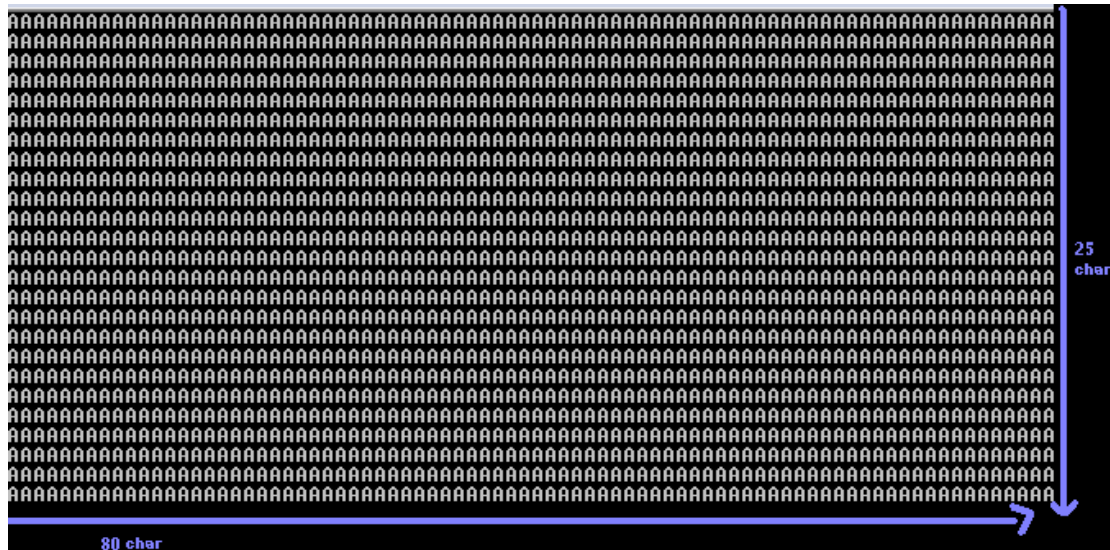
► نبذه عن الشاشة النصيه ◀

الشاشة النصيه احد اسهل اوضاع الفيديو بالنسبه للتعامل معها ومن حيث الامكانيات تعتبر اسوء اوضاع
الفيديو المعروفه حيث لا يمكنك الرسم فيها ولاكن تكتفي بإظهار الحروف .
مكونات الشاشة النصيه :

١- تتكون من طول ٢٥ حرف

٢- تتكون من عرض ٨٠ حرف

أي ان الشاشة النصيه تسع ٢٥ حرف بطول الشاشة و ٨٠ حرف بعرض الشاشة انظر الصوره ربما تكون
الفكره اوضح.



سؤال كيف اقوم بعرض حرف علي الشاشة وله لون معين؟
عن طريق تحميل عنوان الشاشة النصيه والذي يساوي 0xb8000 برقم الحرف في جدول
ascii ويتبعه لون الحرف والوان الحروف كالتالي
0: اسود 1: ازرق 2: اخضر 4: احمر 6: بني 9: ازرق فاتح 10: اخضر فاتح 12: احمر فاتح
14: بني فاتح 15: ابيض

وقد تفهم الامر اكثر في الأمثلة التالية:
نريد عرض الحرف A علي الشاشة ولن الخلفيه اسود ولون الحرف ابيض .

الحل

سوف اختار الاسميلي لانها اسهل
اولاً اقوم بتحميل احد المسجلات بعنوان الشاشة النصيه

Mov ebx,0xb8000

تحميل احد المسجلات برقم حرف A في جدول اسكي (انظر المرفق ASCII code)

Mov al,'A'

تحميل عنوان الشاشة النصيه برقم الحرف

Mov byte[ebx],al

أي ان العنوان 0xb8000 = 0x41
والآن دور اللون حسننا لون الحرف(لون الحرف والخلفيه في وقت واحد)
يتكونان من طول ٦ بت ال ٣ بتات الاولى = لون الحرف وال ٣ بتات الموجودين
في النهايه = لون الخلفيه والمطلوب في المثال لون الخلفيه اسود والحرف ابيض
إذا بتات الحروف =

بتات لون الحرف	بتات الخلفيه
1111	0000

لون الحرف = 0x0F

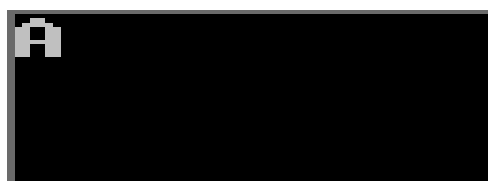
الان يجب كتابه لون الحرف في العنوان الموالي لعنوان الحرف أي العنوان 0xb8001

Inc ebx ; ebx= 0xb8001

Mov al,0x0f

Mov byte[ebx] , al

وسوف تكون النتيجة في النهايه



السؤال التالي : كيف اقوم بتحميل حرف إلي السطر الثالث البعد ٤ (أي بعد الحرف عن بدايه السطر)
الحل:

حسننا لكي نقوم بتحميل حرف إلي مكان معين في الذاكرة نستخدم احد اشهر المعادلات في معرفه موقع الشاشة وهل (رقم السطر $\times ٨٠$) + البعد $\times ٢$ ونقوم بتطبيقه علي مثالنا
رقم السطر = ٣ البعد = ٤

$((٨٠ \times ٣) + ٤) \times ٢ = ٢٤٨$ نقوم بإضافته علي عنوان الشاشة النصيه فيصبح العنوان 0xb8248
نقوم بالخطوات السابقه وعندها نحصل علي مرادنا

سؤال لماذا الرقم ٢ و الرقم ٨٠ في المعادله؟

حسنناً الرقم ٨٠ يمثل عرض الشاشة

الرقم ٢ لاننا لا نملك علي عناوين للحرف فقط بل نملك عناوين للحروف والالوان

هناك اشياء اخري لم اضعها عن الشاشة النصيه واهمها المؤشر .

قد يظن البعض من الحديث السابق ان المؤشر يتحرك تلقائياً من مكانه إلي المكان الذي نقوم بتحميل الحرف
إليه ولكن إجابتي لهم هو ان المؤشر لا يتحرك من تلقاء نفسه بل يجب تحريكه يدوياً

سؤال كيف اخبر المعالج او الشاشة او محرك الفيديو ان يقوم بقل المؤشر إلي العنوان x ؟

مؤشر الفيديو يحتوي علي مسجلين الاول خاص بالتحكم والاوامر المرسله له والثاني خاص بالبيانات

المرسله له رمز مسجل التحكم = 0x3d4 رمز مسجل البيانات = 0x3d5

لكي نقوم بتحريك المؤشر يجب إرسال امرين في مسجل التحكم ويخلف كل امر جزء من موقع الشاشة
الامر غرض الأمر الجزء المتبوع

0xE إرسال الجزء الأعلى من موقع المؤشر ال ١٦ بت الموجودين بعد اول ١٦ بت يتم إرسالهم

إلي مسجل البيانات

0xF إرسال الجزء السفلي من موقع المؤشر اول ١٦ بت بفرض ان موقع المؤشر موجود في

المسجل Eax ف ال ١٦ بت المقصودين هنا هم

الموجودين في المسجل AX ويتم تحميلهم إلي مسجل
البيانات

مثال:

نريد نقل المؤشر إلي السطر ٣ والبعد ٤

الحل:

اولاً نستخدم معادله قريبه من السابقه وهي

(رقم السطر $\times ٨٠$) + البعد
تلاحظ انه لا يوجد الرقم ٢ في المعادله لانه لا نملك إلا علي عنوان
للموقع فقط ولا يوجد عناوين اخري

$((٨٠ \times ٣) + ٤) = ٢٤٤ = ١١١١٠١٠٠$ بالبايت

Ebx نقوم بتحميل احد المسجلات بالرقم السابق وليكن

Mov Ebx,244

نقوم بتحميل المسجل Edx برمز مسجل التحكم والمسجل ax بالأمر

Mov Edx,0x3d4

Mov ax,0xE

Out Edx,ax

الآن قد اخبرنا مسجل الفيديو اننا نريد إرسال الجزء الاعلي من الموقع والان

نقوم بتحميل المسجل Edx برمز مسجل البيانات

Mov Edx,0x3d5

نقوم بإرسال ال ١٦ بت الاعلي من موقع الشاشة

Shr Ebx,16

Out Edx , Ebx

Mov Ebx , 244

الان قمنا بإرسال الجزء الاعلي من موقع الشاشة والآن بقي لنا ارسال اول ١٦ بت
حاول مطابقة الكود التالي مع الشرح في الاعلي

Mov Edx , 0x3d4

Mov ax, 0xF

Out Edx,ax

Mov Edx , 0x3d5

Out Edx , ax

إن التطرف إلي الامور الاخرى التي لم اضعها مثل ما هي الاجهوه المسؤاله عن تحريك المؤشر او المسؤاله
عن إظهار الحرف ولونه يتطلب كتاب كامل لوحده.

▶ ملف التعريفات video.h ◀

ملف التعريفات الخاص بدعم الفيديو علي الشاشة يحتوي علي التعريفات الاولى لدعم الشاشة النصيه وبعدها
سوف يمكنك عرض النصوص علي الشاشة.

Video.h

```
#include "system.h"
#ifndef _H_video_
#define _H_video_

Class video
{
public:
    void putc(char c);
    void clearscreen();
    void print(const char *txt);
    void setcolor(uchar color);
    void setscr(int x, int y);

    int getx();
    int gety();
};

#endif
```

في البدايه نقوم بضم ملف التعريفات system.h والذي يحتوي علي بعض اختصارات التعريفات مثل
Unsigned int = uint

في السطر التالي شرط لمنع تكرار تعريف البيانات الموجوده داخل ملف video.h
في السطر التالي في حاله عدم تعريف الملف video.h يتم تعريفه
في السطر التالي يتم تعريف فنه باسم video تضم الدوال
: غرضها طباعه حرف علي الشاشة **Putc**
: مسح الشاشة بأكملها **Clearscreen**
: طباعه نص علي الشاشة **Print**
: تغيير لون النص **Setcolor**
: تغيير موقع الكتابه وجعل المؤشر يذهب اليه **Setscr**
: معرف البعد الحالي **Getx**
: معرفه السطر الحالي **Gety**

ملحوظه:

في حاله إستخدام الداله **Setcolor** فإن اللون يبقي كما غيرته ولا يعود إلي حالته السابقه
في حاله إستخدام الداله **Setscr** فإن موقع الكتابه يبقي كما غيرته ولا يعود إلي حالته السابقه

في الجزء القادم سوف يتم كتابه الكود الذي سيقوم بدعم الشاشة النصية في حاله عدم وجود أي فهم أو أي معلومه لديك من الجزء السابق (نبذه عن الشاشة النصية) فنرجوا ألا تكمل إلي بعد فهمك لو لجزء واحد فقط لان الجزء القادم كله يعتمد علي فهمك البسيط للجزء السابق.

▶ ملف الشاشة النصيه video.cpp ◀

حسنناً ناتي لاهم جزء في هذا الدرس وهو دعم الشاشة النصيه سوف اقوم بعرض كود كل داله علي حدي حتي تستطيع التركيز معي في الداله المراد شرحها هناك عده اسئله في البدايه
ماذا سنفعل إذا تجاوز البعد ٨٠ أي عرض الشاشة؟
ماذا سنفعل إذا تجاوز السطر ٢٥ أي طول الشاشة؟
هل نستطيع دعم رمز التحكم التي كنا نستخدمها في الداله printf مثل \n \t \r

كل هذه الاسئله سوف نجيب عنها في الفقره القادمه

```
video.cpp [ putc(char c) ]  
  
Int x=0,y=0;  
Uchar color = 0x1F;  
Uchar *tmem = (uchar *)0xb8000;  
  
Void video::putc(char c)  
{  
    if(c == '\n')  
    {  
        x=0; y++;  
    }else if(c == '\b')  
    {  
        if(x > 0)x--;  
    }elseif(c == '\t')  
    {  
        x += 4;  
    }elseif(c == '\r')  
    {  
        x = 0;  
    }else if(c >= ' ' )  
    {  
        int where= ((y*80) + x ) *2;  
        tmem [where] = c;  
        tmem [where +1]=color;  
        x++;  
    }  
    if(x > 80)  
    {  
        x= 0; y++;  
    }  
    if( y> 24)  
    {  
        scroll();  
    }  
    mov_scr();  
}
```

في البدايه نقوم بتعريف متغيرين وهما x و y والمراد منهما حفظ الموقع الحالي للشاشه وقد تم جعلهما يساويان صفر أي السطر الاول والبعد صفر أي اقصي اعلي يسار الشاشه(⬅).

في السطر التالي نقوم بتعريف متغير اسمه color الهدف منه حفظ اللون المستخدم في الكتابه وقد تم جعله = 0x1F أي خلفيه زرقاء ولون كتابه ابيض

السطر الثالث قمنا بتعريف مؤشر يشير إلي عنوان الشاشة النصيه وهو 0xb8000 ويستطيع حمل بيانات بحجم ٨ بت = ١ بايت .

الداله pute تأخذ عامل واحد وهو c من نوع char وهو الذي سيحمل رقم الحرف في جدول آسكي وسوف نقوم بعمل بعض الشروط علي هذا المتغير

١- إذا كان \n :

يعود إلي البعد = ٠ ثم ينزل سطرأ واحداً

٢- إذا كان \b :

نقوم بإختبار إذا كان البعد الحالي < من صفر ام لا إذا كان نقوم بعملية تنقيص للبعد بمقدار ١

٣- إذا كان \t :

نقوم بزياده مؤشر البعد بمقدار ٤ وهو يساوي تقريباً الزر tab في الويندوز عن الضغط عليه فإنه يتقدم بمقدار حوال من ٤ إلي ٦ خطوات إلي الامام

٤- إذا كان \r :

نقوم بالرجوع إلي بدايه السطر أي جعل البعد = صفر

٥- إذا كان < " " او مسافه:

فإنه يقوم بتحميل الحرف إلي الموقع الحالي في الذاكره والموافق $(y*80) + x$ تذكر المعادله السابقه حيث y هي السطر و x هي البعد ثم نقوم بزياده مؤشر البعد حتي لا نقوم بالكتابه في نفس الموقع.

سؤال لماذا تم إختيار المسافه بالذات في الشرط ؟

إذا نظرت إلي الرموز السابقه (b \t \n) سوف تجد انه تقع جميعاً في جدول

اسكي قبل الرقم 0x20 الذي بدوره = المسافه والحرف جميعها تقع بعد جدول اسكي.

بعض الاختبارات علي متغيرات موقع الشاشة (x,y) :

إذا كان $x > 80$:

او بمعنى اخر تجاوزت x حدود الشاشة يقوم بالنزول سطر واحد والعوده إلي بدايه الشاشة.

إذا كان $y > 25$:

او بمعنى اخر تجاوزنا عدد الاسطر المسموح بها نقوم بعملية رفع الاسطر او scroll

في نهايه الداله نقوم بمناده داله تحريك المؤشر إلي الموقع الحالي في الشاشة.

هدف هذه الداله في النهايه هو تحميل حرف ما إلي مكان ما في الذكره وبكتابتها كودها نكون قد انتهينا من ٦٠ % من كود ملف video.cpp

الداله الثانيه وهي المذكوره في الكود السابق وهي scroll:

Video.cpp [scroll()]

```
void scroll()
{
    int i;
    for(i=0; i < (80 * 24) * 2; i++)
    {
        tmem[i] = tmem[i+160];
        i++;
    }
    for(i=0; i < 160; i++)
    {
```

```
videomem[i + (80 * 24) * 2] = ' ';  
i++;  
}  
y=24;x=0;  
}
```

الهدف من هذه الدالة هو نقل الاحرف من السطر رقم x إلى السطر رقم (x - 1) أو السطر الذي يسبقه حتي يصل إلى السطر الاخير ثم يقوم بنقل الحروف الموجودة فيه إلى السطر الذي يسبق
.....
في البدايه نقوم بتعريف متغير والمستخدم في هذا الكود كعداد

في البدايه نقوم بدوره هدف هذه الدوره هي نقل الاحرف من سطر إلى السطر الذي يسبقه

طول الدوره = $2 * (24 * 80)$ أي اننا سنقوم بدوره بعدد الاسطر * طول السطر * 2
نقوم بجعل الحرف الموجود في الموقع رقم I = الحرف الموجود في الموقع رقم (I + 160) لماذا ١٦٠ تذكر
اننا نحتوي علي عناوين خاصه بالالوان والحرف أي عناوين زوجيه
مثال إذا كان الحرف A موجود في الموقع ١ والحرف S موجود في الموقع ٨١ فلكي نقوم بالتبديل بين
الحرف أي تبديل الحرف A ب Z يجب أولاً الحصول علي موقع الحرف بالنسبه ل z وملاحظه اننا نحتوي علي
عناوين خاصه بالالوان وأن طول السطر (أي طول الاحرف = ٨٠) إذا بعد Z عن A = ١٦٠
٨٠ للالوان و ٨٠ للحروف

في الدوره التاليه نقوم بمسح السطر رقم ٢٤ لماذا ٢٤ وليس ٢٥ لانه هناك شاشات تجد فيه ان السطر الاخير
دائما مخفي.

=====

الداله **mov_scr** او داله نقل المؤشر هي تطبيق فعلي لشرح الشاشه النصيه
الكود:

```
Video.cpp [mov_scr()]  
void mov_scr()  
{  
    int index = (y*80) + x;  
    outb(0x3d4,0xE);  
    outb(0x3d5,(index >> 8) & 0xFF);  
    outb(0x3d4,0xF);  
    outb(0x3d5,index & 0xFF);  
}
```

حاول تطبيق هذا الكود مع شرح الداله النصيه
نستخدم 0xFF للحصول علي بتات معينه بدون أي إضافات.

=====

```
Video.cpp [Clearscreen]  
void video::clearscreen  
{  
    for(int i=0;i <= (80 * 25)*2;i++)  
    {  
        videomem[i] = ' ';  
        i++;  
        videomem[i] = 0x1F;  
    }  
}
```

داله مسح الشاشه : لابد من دراستك للدوال السابقه انك ستفهم ما المراد منها
كل ما نقوم به هي استبدال جميع الحروف بالمسافه

في النهايه لم يتبقى لنا إلّا أهم داله وهي داله طباعه النصوص

```
Video.cpp [print(const char *txt) ]
Void video:: print(const char *txt)
{
    int i;
    for(i=0;i <= strlen(txt);i++)
    {
        video::putc(txt[i]);
    }
}
```

كل ما في الداله هي عمل دوره عدد دوراتها = طول النص ويتم طباعها حرف حرف.

ربما قد تلاحظ انني لم اضع كود كل الدوال فقد تركتها لك لتقوم بها بنفسك 😊.

قم بإضافه video.o إلي ملف ال Makefile

////////////////////////////////////

► ملف جدول المداخل (Gdt) gdt.h ◀

في الدرس السابق عن النمط المحمي كنا نقوم بكتابه الكود تلقائياً وكنا الآن نحتاج إلي كتابه الكود بواسطه السي/سي++ أي ان حجم المعلومات غير معروف مثلاً البت G حجمه ١ بت ولكن في السي/سي++ نحن لا نعرف الحجم بالضبط لذا إذا كنت ستجرب النواه علي جهاز حقيقي فسوف يعيد الجهاز التشغيل كل مره تفتح فيها النواه.

لذا سوف استخدم شئ صحيح لا اعرف ماذا يسميه المبرمجين العرب ولكني احب ان اسميه إعادة التحجيم مثال

Unsigned char x:1;

هنا المتغير x بحجم ١ بت

Unsigned char x:2;

هنا المتغير x بحجم ٢ بت

والتالي هو محتوى ملف system.h

Gdt.h

```
#include "system.h"
#ifndef _H_gdt_
#define _H_gdt_

struct gdt_entry
{
    ushort limit;
    ushort base;
    uchar base_1;
    uchar type:4;
    uchar syst:1;
    uchar dpl:2;
    uchar pres:1;
    uchar limit_2:4;
    uchar bit64:1;
    uchar bit64_1:1;
    uchar d_b:1;
    uchar gran:1;
    uchar base_2;
```

```
__attribute__((packed));
struct gdt_pointer
{
    ushort limit;
    uint address;
}__attribute__((packed));

class gdt
{
public:
    void setgate(int num,bool iscode,int dpl,int pres);
    void install();
}
#endif
```

الثلاث الاسطر الاولى طبعاً مفهومة

struct gdt_entry

نقوم بتعريف بنيه تحتوي هذه البنية علي معلومات حول مداخل الذاكرة تذكر الدرس السابق

struct gdt_pointer

نقوم بتعريف البنية التي سيتم تحميلها إلي الذاكرة

__attribute__((packed))

هذه التعليمة يتم إضافتها لمنع المترجم من عمل أي تعديلات علي البنية

نقوم بتعريف فئة اسمها gdt يوجد فيها دالتان

الاولي تقوم بملى المداخل بالمعلومات والثانية تقوم بتحميل الجدول إلي الذاكرة

◀ ملف جدول المداخل gdt.cpp ▶

كود الملف

```
Gdt.cpp
#include "inc/system.h"
#include "inc/gdt.h"

gdt_entry gdt[5];
gdt_pointer gdt_p;
void gdttable::setgate(int num,int dpl,bool code,int pres)
{ /** entry limite **/
    gdt[num].limit = 0xFFFF;
    gdt[num].limit_2= 0xF;
    /** entry base **/
    gdt[num].base = 0;
    gdt[num].base_1 = 0;
    gdt[num].base_2 = 0;
    /** bit 40:48 **/
    gdt[num].pres = pres;
    gdt[num].dpl = dpl&3;
    gdt[num].syst = 1;
    gdt[num].type = code?0xb:0x3;
    /** bit 52:56 **/
    gdt[num].gran = 1;
    gdt[num].d_b = 1;
    gdt[num].bit64 = 1;
```

```
        gdt[num].bit641=0;
    }
    void gdttable::install()
    {
        /** Null entry **/
        memset((uchar*)&gdt,0,sizeof(gdt_entry));
        /** Code entry **/
        gdttable::setgate(1,0,true,1);
        /** Data entry **/
        gdttable::setgate(2,0,false,1);
        /** Install it **/
        gdt.limit = (sizeof(gdt_entry)*5) -1;
        gdt.address=(uint)&gdt;
        asm volatile ("lgdt %0"::"m"(gdt));
        asm volatile ("ljmp $8,$1f");
        asm volatile ("1:");
        asm volatile ("mov $16,%ax");
        asm volatile ("mov %ax,%es");
        asm volatile ("mov %ax,%ds");
        asm volatile ("mov %ax,%gs");
        asm volatile ("mov %ax,%fs");
        asm volatile ("mov %ax,%ss");
    }
```

في البدايه نقوم بضم المكتبتين اللذان قمنا بعملهما وهما `system.h` + `gdt.h` السطر الموالي نقوم بحجز مساحه لمداخل الجدول بحجم ٥ (لا نستطيع ان اذكر ابسط من ذلك) أي ان عدد الجداول المتوفره لنا هي ٥ مداخل

الداله الاول ← تقوم بملئ بيانات الجدول حاول مطابقتها مع الدرس السابق
الداله الثانيه ← تقوم بصنع المداخل الخاصه بالذاكره المدخل الاول كله صفر تبعاً لتعليمات كتاب `Intle` المدخل الثاني خاص بالكود والثالث خاص بالبيانات

```
gdt.limit = (sizeof(gdt_entry)*5) -1;
gdt.address=(uint)&gdt;
```

في هذين السطرين نقوم بمعرفه طول الجدول وعنوانه حتي يتم تحميلهم إلي الذاكره في السطور المواليه نقوم بعده تعليمات بالأسمبلي لتحميل معلومات الجدول إلي المعالج وقد استخدمت التعليمة `volatile` لمنع المعالج من عمل أي تعديل علي البرنامج قم بإضافه `gdt.o` إلي ملف ال `Makefile` .

لم يتبقي لنا سوي تجريب النواه وتجزير ملف التنفيذ الرئيسي

▶ ملف التنفيذ الرئيسي main.cpp ◀

محتوي ملف التنفيذ الرئيسي

Main.cpp

```
#include "system.h"
#include "video.h"
#include "gdt.h"

extern "C" void Hmain(multiboot *mb)
{
    video v; // تعريف الفنتين video - gdt
    gdt g;

    v.clearscreen();
    v.print("installing gdt");
    g.install();
    v.printf("\t\t [Done]");
}
```

أظن ان كل شئ مفهوه ما عدا السطر

extern "C" void Hmain(multiboot *mb)

اتذكر ونحن نقوم بكتابه ملف boot.asm حين قلت لك قم باستبدال \$ jmp _Hmain بالأمر jmp _Hmain
اتذكر البرمجه المتفرقه هناك علاقته وثيق بينها حاول اكتشافها .

لم يتبقى لنا سوي تجريب النواه قم بإضافه main.o
اولاً قم بفتح القرص الوهمي المرفق عن طريق برنامج vfd

ثم قم بفتح برنامج Cygwin عن طريق

Start → programs → Cygwin → Cygwin Bash Shell.

وقم بكتابه الامر التالي

Make -C x:/folder -f Makefile

حيث:

X:/folder هي مسار ورمز القرص

Makefile اسم ملف ال Makefile الذي قمت بحفظه
قم بتشغيل جهازك الوهمي واجعله يقلع من القرص المرن الوهمي.

مبروك لقد انجزت شوطاً طويلاً في البرمجه ب c/c++

في حاله حدوث أي اخطاء او استفسار نرجوا مراسلتي علي بريدي الالكتروني

Ahmed_ali_halawa@yahoo.com

Ahmed_tegr2000@yahoo.com

او العضو Halwa في منتدى الفريق العربي للبرمجه

www.arabteam2000-forum.com