الكاتب : JAAS
جمعان عبدالله البريكي
Jaas1001@hotmail.com

الموقع : Arabteam2000
الفريق العربي للبرمجة
www.arabteam2000.com

برمجة الـ BIOS وأنظمة التشغيل ١٦ بت و ٣٢ بت

السلام عليكم ورحمة الله ،،،

نظام تشغيل الكمبيوتر :
عبارة عن برنامج ضخم شغلتة الأساسية الربط بين المستخدم وبين أجزاء الكمبيوتر الأخرى (الهاردوير)

في هذا الكتاب سنحاول مناقشة هذا التعريف بشكل تفصيلي ، وسنقوم بتقسيمة إلى :

١ - ماهي مراحل تطور أنظمة التشغيل ومن أين أتت الفكرة (نظام BIOS)
(نظام البيوس ، ماهي طريقة عملة وهل تم نقل بقية أفكار الأنظمة منة)

٢ - كيف تكتب أنظمة التشغيل ١٦ بت وكيف تعمل (نظام دوس و يونكس ...)
(سنقوم بكتابة أمثلة تطبيقية على عملية إقلاع النظام وكيف تمت في بقية الأنظمة)

٣ - ماهو الجديد في برمجة الأنظمة ٣٢ بت (نظام ونـدوز ونظام لينكس ...)
(مفهوم الجداول و النمط الحقيقي و تعدد المهام والبوابات والإمتيازات)

قد تتطلب قرائة هذا الكتاب فهم بسيط للغة الإسمبلي بالإضافة لبعض قطع الجهاز الهاردوير

(وعلى بركة الله نبـــــــــدأ)

أولا : عملية الإقلاع

الآن عندك جهاز كمبيوتر بجانبك ... بشرط أن يكون مغلق . بمعنى أنة غير موصول بالكهرباء!

قم بتوصيل كل أجزاء الكمبيوتر +تأكد من توصيل أسلاك الكهرباء + ضع أصبعك على زر التشغيل

هل أنت مستعد لتشغيل الكمبيوتر ؟!

سنقوم الآن بكتابة كل عملية تحدث من أول ما تضغط على زر التشغيل ..إلى أن تظهر نافذة نظام التشغيل

والآن إضغط على زر أو مفتاح التشغيل وإقرأ كل كلمة في الموضوع بتفكير عميق ... البداية

يبدأ موزع الكهرباء أو البور سبلاي بإمداد اللوحة الأم بالطاقة .. مباشرة تصل الطاقة إلى المعالج

يشتغل المعالج أول ما يقوم بة هو تصفير لكل مسجلات المعالج مثل ax و bx و ds و..و..

بعد ذلك يقوم بشغلة مهمة لبدء التحكم .. يقوم المعالج بإعطاء المسجل cs وهو قسم الشفرة أو التنفيذ

القيمة F000(كل الأرقام بالرمز الست عشري أو الهكس) وبهذا يكون المعالج قد حدد قسم التنفيذ

وبعد ذلك يقوم بتحديد الإزاحة للقسم وأكيد بواسطة المسجل ip فيعطية القيمة FFF0

تلاحظ أن المعالج بدأ التنفيذ في النمط الحقيقي أي نمط ١٦ بت وقد إستخدم مسجلات هذا النمط

والسبب في ذلك هو التوافقية بين الأنظمة + إمكانية الوصول إلى أي عنوان في الذاكرة

والآن المعالج صفر أو هيأ كل المسجلات + حدد بداية التنفيذ في الذاكرة

بعد ذلك يقوم بنقل التنفيذ من العنوان الذي حددة المعالج إلى نظام الإدخال والإخراج الأساسي (البيوس)

ماهو البيوس : عبارة عن برنامج مخزن في ذاكرة على اللوحة الأم تسمى الذاكرةROM

وهي ذاكرة للقرائة فقط + أنها لاتفقد المعلومات المخزنة بها عند إغلاق الجهاز

والأمثلة على البرامج المخزنة في ذاكرة ROM كثيرة.. هل لاحظت شعارات الشركات في بداية التشغيل

مثل كرت الشاشة يظهر لك علامة تجارية مثلا SIS.. هذة البرامج كلها مخزنة في الذاكرة ROM

وللمعلومة : يمكن للهكر أو الكراكر(المختصين في البرمجة العكسية) الوصول للبرامج المخزنة في ROM وتغيير محتوها .. بل وكسر حمايتها ؟ أكيد تريد أمثلة .. تفضل

http://hackingthexbox.com/

http://www.xenatera.com/bunnie/proj/anatak/xboxmod.html

والآن كيف تعمل هذة البرامج ... إذا فهمت البيوس ستعرف القصة!

كيف يعمل البيوس ؟

بعد أن يجهز المعالج المسجلات يقوم بتسليم القيادة أو التحكم للنظام بيوس

يبدأ التنفيذ في البيوس عند السطر ٤٣٠ (في معالجات إنتل) يقوم البيوس بعدد من المهام الأساسية

أول مهمة للبيوس هي فحص قطع الجهاز :

لا تعتقد أن طريقة فحص الأجهزة أمر معقد ... بكل بساطة يقوم البيوس بإرسال إشارة لمنفذ الجهاز

عن طريق الأمر out وبعد ذلك يقوم بإستقبال أي إشارة من الجهاز عن طريق الأمر IN

إذا وصلت أي إشارة من الجهاز معنى ذلك أنة شغال ... وإذا لم تصل أي إشارة معناة خطأ في الجهاز

وهكذا مع كل أجهزة الكمبيوتر.

في نقطة مهمة في الفحص .. أول مايقوم بة البيوس فحص كرت الشاشة + منفذ الشاشة

إذا وجد أي خطأ فية (بمعنى أن الشاشة لا تعمل ولايمكن رؤية أي ملاحظة يكتبها البيوس على الشاشة)

يقوم البيوس في هذة الحال بإستخدام طريقة الأصوات أو الصافرة لعرض أي خطأ في عملية الفحص

أمثلة لأهل الصيانة

قمت بتشغيل الجهاز .. لم يعمل الجهاز .. لم يصدر البيوس أي صوت (على طول الخطأ في البور سبلاي)

لأنة كما ذكرنا. يقوم البيوس بإصدار صوت إذا وجد خطأ. وإذا لم يصدر أي صوت معنى ذلك أن التنفيذ

لم يصل إلى البيوس .. لاحظ معنى الأصوات في البيوس

صافرة واحدة قصيرة = لا توجد أخطاء في عملية الفحص

صافرتين قصيرة = خطأ في كرت الشاشة أو طريقة توصيل الشاشة

وهكذا مع كل الأخطاء

والنقطة الثانية ... إذا كانت الشاشة شغالة (يتم عرض الأخطاء بطريقة الأرقام)

أخطاء البيوس في كلى الحالتين تجدها هنا:

http://www.pchell.com/hardware/beepcodes.shtml

بعد أن يقوم البيوس بفحص الأجهزة يقوم بإنشاء جدولين

وهما جدول مقاطعات البيوس – و جدول معلومات البيوس

الأول : جدول مقاطعات BIOS :

عند العنوان ٠٠٠٠:٠٠٠٠ (العنوان صفر) يبدأ بكتابة عنواين المقاطعات .كل مقاطعة تأخذ لها ٤ بايت .. لتدل على عنوان بداية تنفيذ المقاطعة ، بهذا الشكل

المقاطعة صفر INT 0 عنوانها ٠٠٠٠:٠٠٠٠ (هذا العنوا يحتوي على عنوان التنفيذ)
المقاطعة واحد INT 1 عنوانها ٠٠٠٤:٠٠٠٠ ، المقاطعة ٢ عنوانها0000:0008

هذا الجدول الأول (جدول المقاطعات)

رقم المقاطعة (Hex)	مجال العنونة (Hex)	ملاحظات
INT 00 h	0000:0000h	مقاطعة لراية الفيض في القسمة
INT 01 h	0000:0004h	مقاطعة عمل المعالج خطوة ..خطوة
INT 02 h	0000:0008h	مقاطعة الأجهزة الخارجية لمايعرف بدبوس NMI
INT 03 h	0000:000Ch	مقاطعة لكتابة نقطة توقف للبرنامج
INT 04 h	0000:0010h	مقاطعة راية الفيض في مسجل الحالة
INT 05 h	0000:0014h	طباعة الشاشة
INT 06 h	0000:0018h	مقاطعة لإختبار تنفيذ تعليمة غير مصرح لها
INT 07 h	0000:001Ch	مقاطعة ضغط وتشفير التعليمات
INT 08 h	0000:0020h	خدمات المؤقت
INT 09 h	0000:0024h	خدمات الجهاز المشغل للوحة المفاتيح
INT 0A h	0000:0028h	مقاطعة تستخدم في التبديل بين المهام
INT 0B h	0000:002Ch	خدمة المنفذ التسلسلي com2
INT 0C h	0000:0030h	خدمة المنفذ com1
INT 0D h	0000:0034h	خدمة المنفذ المتوازي LPT 2
INT 0E h	0000:0038h	خدمات القرص المرن (الفلوبي)
INT 0F h	0000:003Ch	خدمة المنفذ المتوازي LPT1
INT 10 h	0000:0040h	خدمات شاشة العرض
INT 11 h	0000:0044h	مقاطعة تعيد الأجهزة المتصلة بالكمبيوتر
INT 12 h	0000:0048h	مقاطعة حجم الذاكرة
INT 13 h	0000:004Ch	خدمات القرص الصلب
INT 14 h	0000:0050h	خدمات منافذ لإتصالات
INT 15 h	0000:0054h	خدمات النظام
INT 16 h	0000:0058h	خدمات لوحة المفاتيح
INT 17 h	0000:005Ch	خدمات الطابعة
INT 18 h	0000:0060h	خدمات لتحميلROM BASIC
INT 19 h	0000:0064h	مقاطعة تحفيز قرص الإقلاع
INT 1A h	0000:0068h	خدمات الوقت الحقيقي للساعة
INT 1B h	0000:006Ch	تحديد مدة الإستجابة للوحة المفاتيح
INT 1C h	0000:0070h	مقاطة التوقيت مقدر باللحظة
INT 1D h	0000:0074h	مقاطعة جدول وبارمترات شاشة العرض
INT 1E h	0000:0078h	مقاطعة جدول وبارمترات القرص المرن
INT 1F h	0000:007Ch	مقاطعة خصائص الخط والمحارف

تلاحظ أننا توقفنا عند العنوان ... address=1F أين بقية المقاطعات

باقي المقاطعات سنكتبها في وقتها.. لأن نظام الدوس يتطفل ويبدأ في الكتابة عند العنوان الذي توقفنا عندة

المهم : ماالفائدة من هذا الجدول + ما الفائدة من المقاطعات ؟ !

أولا : الفائدة من الجدول هي نقل التنفيذ .. والمثال على ذلك في وندوز مكاتب الربط ذات الإمتداد dll

يمكن يكون الشرح غير مفهوم : والحل بالتطبيق العملي :

من قائمة start ثم run إكتب debug وإضغط أوكي (لتشغيل برنامج الديبغر)

ما رأيكم بمقاطعة حجم الذاكرة وهي المقاطعة ١٢ .. عنوانها في الذاكرة ٠٠٤٨:٠٠٠٠

في برنامج الديبغر أكتب الأمر 0000:0048 **D** وهو عنوان المقاطعة ، إحفظ أول ٤ بايتات

ولتكن مثلا 02 0E 09 AA : طبق عليها قانون العكس فتساوي 020E:09AA

هل تعرف مايمثلة العنوان ، هو بداية التنفيذ لشفرة المقاطعة ... جرب إذهب للعنوان

بواسطة أمر فك التجميع 020E:09AA **u** ولاحظ كيف تنفذ المقاطعة

الفائدة من المقاطعة:
أهم فائدة هي تبسيط الأمور وهو ا سلوب متبع إلى الآن في الأنظمة الحديثة مثل وندوز xp

ما رأيكم بتشريح مقاطعة إظهار حجم الذاكرة .. لأنها أسهل مقاطعة و أول مقاطعة ينفذها البيوس

كل مرة عند تشغيلك لجهازك ترى بأن النظام بيوس بعد فحص الأجهزة يعرض لك حجم الذاكرة

هل تريد أن تعرف كيف ينفذها .. عن طريق برمجة المنافذ + البرمجة المباشرة للذاكرة

بهذا الكود:

```
mov al,18h
out 70h,al
in al,71h
mov ah,al
mov al,17h
out 70h,al
in al,71h
```

بعد تنفيذ هذا الكود سيصبح مسجل = ax حجم الذاكرة

هل عرفت فائدة المقاطعة - فقط إستبدل كل الكود بتعليمة المقاطعة INT 12 وسيحمل المسجل ax بحجم الذاكرة

هذا هو مفهوم المقاطعة ، بعد ذلك يبدأ البيوس في بناء الجدول الثاني
جدول معلومات BIOS :

وهو جدول يخزن بة البيويس معلومات عن الأجهزة (الهاردوير) الموجودة في الجهاز
يبدأ نظام البيوس في كتابة معلوماتة عند العنوان ٠٠٤٠:٠٠٠٠

وهذة المعلومات هي:

العنوان	البيانات التي ستخزن
0040:0000h	عنوان المنفذ com1
0040:0002h	عنوان المنفذ com2
0040:0004h	عنوان المنفذ com3
0040:0006h	عنوان المنفذ com4
0040:0008h	عنوان المنفذ LPT1
0040:000Ah	عنوان المنفذ LPT2
0040:000Ch	عنوان المنفذ LPT3
0040:000Eh	عنوان المنفذ LPT4
0040:0010h	مجموعة الأجهزة وهو عبارة عن ١٦ بت تمثل الأجهزة المثبتة
0040:0012h	رايات المقاطعة والتحقق من الأخطاء
0040:0013h	حجم الذاكرة بالبايت
0040:0015h	معرف أو كود الخطأ
0040:0017h	رايات الحالة والتغير في لوحة المفاتيح
0040:001Ah	مؤشر إلى الحرف التالي في مخزن لوحة المفاتيح
0040:001Ch	مؤشر إلى الحرف الأخير في مخزن لوحة المفاتيح
0040:001Eh	محتوى مخزن لوحة المفاتيح
0040:003Eh	معرف القرص المرن.. A=0 - B=1 C=2
0040:003Fh	حالة محرك الأقراص المرن
0040:0042h	مسجل التحكم للقرص الصلب والمرن
0040:0049h	إعدادات نمط شاشة العرض
0040:004Ah	عدد الأعمدة في كل سطر
0040:004Ch	حجم الصفحة في شاشة العرض
0040:0050h	نقطة ظهور المؤشر في شاشة العرض
0040:0063h	يحتوي على عنوان منفذ الإدخال والإخراج لشاشة العرض
0040:0066h	لوحة الألوان لشاشة العرض
0040:0075h	عدد الأقراص الصلبة في الجهاز
0040:0076h	بايت التحكم للقرص الصلب
0040:0077h	عنوان منفذ الإدخال والإخراج للقرص الصلب
0040:008Dh	مسجل أخطاء القرص الصلب
0040:00A1h	معرف كرت الشبكة المحلية
0040:00F0h	منطقة معلومات الإتصالات بين التطبيقات

وبهذا يكون نظام البيوس قد جهز خدماتة وهي المقاطعات وبعد ذلك جهز معلوماتة

وفي النهاية يبحث عن نظام التشغيل لتحميلة إلى الذاكرة وتسليمة مهمة القيادة والتحكم

كيف ... تابـــع الموضوع

تحميل نظام التشغيل إلى الذاكرة وبداية عملة:

يبدأ البيوس في البحث عن مشغل النظام وهو ما يعرف بقطاع الإقلاع boot sector

قد يكون في قرص مرن أو قرص صلب أو حتى سيدي ، المهم أن البيوس يبدأ البحث

في كل الأقراص .. حسب الترتيب الذي حددتة في إعدادات البيوس

دائما يكون قطاع الإقلاع هو القطاع الأول في القرص ويجب أن يحمل علامة تدل على أنة قطاع إقلاع

العلامة هي أن يكون آخر بايتين في القطاع = 55 AA

كتابة برنامج إقلاع في القطاع الأول شغلة سهلة لأن الموضوع يتطلب فهم أمرين فقط

الأول: إستخدام مقاطعات البيوس

برنامج الإقلاع القياسي يستخدم مقاطعتين فقط وهي المقاطعة ١٣ لتحميل الملفات من القرص

والمقاطعة ١٩ لتحفيز القرص للإقلاع (وهذة المقاطعة قد لا تستخدم دائما)

والأمر الثاني : معرفة العناوين الذي سينتقل لها التنفيذ + معرفة مخزن القرائة من القرص

سنتابع قصة إقلاع الكمبيوتر وستتضح الأمور.

توقفنا عند البيوس وطريقة تحملية للنظام التشغيل ، وقلنا أنة سيحمل قطاع واحد فقط

بعد أن يجد قطاع الإقلاع سيقوم بتحميلة إلى العنوان address =7C00

ماذا تستفيد من معرفة عنوان التحميل .. أول شيء تقوم بة إعطاء المسجل cs هذة القيمة

لينتقل التنفيذ لها ، بعد ذلك مباشرة تقوم بإستخدام المقاطعة 13 لتحميل كل محتويات القرص

أو تحمل برنامج معين ، في أمور مهمة عند هذة النقطة ؟ ماذا يعرف البيوس عن القرص

هل مرت عليك أنظمة الملفات مثل FAT ,NFS ,ex2..3 وغيرها من أنظمة ملفات وندوز ولينكس

بإختصار برنامج البيوس لا يعرف شيء عن هذة الخرابيط ولا يعرف حتى أين تبدأ وأين تنتهي

ما يعرفة البيوس هو تركيب القرص الفيزيائي مثل الرأس و الإسطوانة و القطاع والمسار

سأعطيك مثال بارمترات مقاطعة قرائة محتوى القرص

```
AH=0x02 // خدمة قرائة القرص وتحميل البرامج إلى الذاكرة

AL=عدد القطاعات

CH=رقم المسار

CL=رقم القطاع

DH=رقم الرأس

DL=رقم القرص

ES:DX=مؤشر إلى المخزن
```

قد تعتقد أن تحديد ملف معين أمر معقد ، ولهذا السبب ظهرت أنظمة الملفات مثل FAT و..و

ولكن إذا قمت أنت بكتابة ملف معين في قطاعات القرص بكل بساطة تستطيع أن تحدد موقعة بالضبط

ملاحظة: في آخر بارمتر وهو ES:DX هذا البارمتر لا تكتب بة شيء لأنك عندما تنفذ المقاطعةint13

سيقوم البيوس بكتابة عنوان تحميل البرنامج إلى الموقع , ES:DX وبهذا تقوم بحفظ هذا العنوان

وتقوم بنقل التنفيذ من قطاع الإقلاع إلى العنوان الذي يمثل البرنامج أو النظام الذي حملتة و يبدأ التنفيذ

قد لا تصدق أن تحميل النظام بكل هذة البساطة ،، مارأيك بمثال

سنقوم بكتابة أمر بسيط في قطاع الإقلاع ، وهو مثال لإظهار عدد من الحروف ، وأكيد المثال سيكون على قرص مضغوط

وأول ما تشغل الكمبيوتر ، وبعد عملية فحص الأجهزة .. سيظهر برنامجنا بكل ثقة ؟!

سنقوم بكتابة البرنامج داخل قطاع الإقلاع بإستخدام برنامج debug في النظام وندوز

وإذا كنت لا تعرف الديبغر يمكنك مراجعة الرابط:

http://www.arabteam2000-forum.com/index.php?showtopic=44877

الآن قم بإدخال القرص المضغوط(الفلوبي) لا يهم إذا كان مفرمت أو لا لأن البيوس لا يعرف شيء من هذا

ولكن لوكان القرص مفرمت سيوفر علينا كتابة توقيع قطاع الإقلاع وهو AA 55 في آخر بايتين

بعد ذلك من قائمة start ثم run إكتب debug لتشغيل برنامج الديبغر وستلاحظ العلامة " - " ظهرت

عند العلامة إكتب الأمر **L** 100 0 0 1
الحرف L معناة تحميل قطاع من القرص إلى الذاكرة ١٠٠ تعني إلى العنوان ١٠٠ في نفس المقطع
الصفر الأولى = محرك الأقراص A و الصفر الثانية تعني رقم القطاع وهو ٠ أي قطاع الإقلاع
الواحد الأخير يعني عدد القطاعات التي تريد تحميلها إلى الذاكرة عند العنوان ١٠٠

بهذا الأمر نكون حملنا قطاع الإقلاع إلى الذاكرة ، والآن نريد أن نكتب بة برنامج صغير

أكتب عند علامة - التعليمة 100 **a** أي أننا نريد كتابة كود إسمبلي عند العنوان ١٠٠

بعد ذلك إكتب البرنامج:

```
- a 100

-0B3A:0100   mov ah,09
-0B3A:0102   mov al,42
-0B3A:0104   mov bh,0
-0B3A:0106   mov bl,14
-0B3A:0108   mov cx,10
```

```
-0B3A:010B    int 10
```

العناوين يكتبها الديبغر نفسة وقد تتغير .. لا يهم . أهم شيء هي التعليمات

البرنامج الذي كتبناة هو عبارة عن مقاطعة إظهار حرف على الشاشة , الرقم ٤٢ يمثل الحرفB

الرقم ١٤ يمثل لون الحرف ولون الخلفية الرقم ١٠ يمثل عدد تكرار الحرف على الشاشة

بعد ذلك سنقوم بإدخال برنامجنا إلى العنوان ١٠٠ وتخزينة في الذاكرة ، بإستخدام التعليمة e 100

```
-e 100
0B3A:0100  B4.   09.   B0.   42.   B7.   00.   B3.   14.
0B3A:0108  B9.   10.   00.   CD.   10.
```

كما شرحت سابقا يتم إدخال كل بايت بإستخدام مفتاح المسافة في لوحة المفاتيح

وبقي الآن كتابة هذا البرنامج إلى قطاع الإقلاع في القرص , بإستخدام الأمرW

بهذة الطريقة: W 100 0 0 1

```
-: W 100 0 0 1
```

ولآن تأكد من أن القرص المضغوط بداخل السواقة

تأكد من أن ترتيب الإقلاع في إعدادات البيوس يبدأ من القرص المرن

وأعد تشغيل الجهاز ..... ولاحظ (برنامجك على الشاشة)

ألف مبروك أصبحت مبرمج أنظمة معتمد!!؟

وبعد كل هذة المقدمة بقية كلمة واحدة وهي

لو كانت برمجة الأنظمة فقط إلى هنا .. لوجدت ١٠٠ نظام تشغيل عربي ؟!

بالفعل هذة الطريقة نفسها من أول ماظهر دوس و إلى الآن وندوز ٢٠٠٣ يستخدم نفس الطريقة

ولكن التعقيد يظهر بعد تحميل النظام.. أي بعد أن يسلمك البيوس مهمة التحكم في الجهاز

ماذا تفعل وكيف ؟!

سأحاول الإجابة على هذا السؤال بعد تقسيمة إلى قسمين:

١ - كيف تعمل الأنظمة تحت بيئة ١٦ بت (مثل النظام دوس و النظام يونكس ١٦ بت)

٢ - كيف تعمل الأنظمة تحت بيئة ٣٢ بت (مثل الوندوز و اللينكس)

وصلنا في الموضوع السابق إلى طريقة تحميل نظام التشغيل .. وسنبدأ من هذة النقطة.

قبل البدء : أنظمة التشغيل الموجودة حاليا تنقسم إلى قسمين هي أنظمة ١٦ بت وأنظمة ٣٢ بت

والإختلاف بين منصة ١٦ بت و ٣٢ بت إختلاف كبير جدا ولذلك سينقسم الموضوع إلى قسمين

١- بنية و عمل أنظمة ١٦ بت:

بإختصار أنظمة ١٦ بت هي الأنظمة القديمة وهي عبارة عن شاشة سوداء تستقبل الأوامر

ظهرت هذة الأنظمة للعالم بعد ظهور المعالج ٨٠٨٦ إختصارة (x86) من قبل شركة إنتل

ومن هذة الأنظمة الدوس و اليونكس و OS2 و بعض البرامج المستقلة وغيرها!

هذة الأنظمة تعمل في النمط الحقيقي (بمعنى أنة لاتوجد قواعد وشروط وإمتيازات للبرامج والنظام)

أهم نقطة في هذة الأنظمة : أنها لا تستطيع تشغيل أكثر من مهمة أو برنامج في نفس الوقت

معلومة : قد تلاحظ أنة في نظام الدوس مثلا أكثر من برنامج مقيم في الذاكرة وكلها شغالة

ومع ذلك لا يعتبر النظام متعدد المهام ، لأن كل البرامج المقيمة في الذاكرة لاتعمل في نفس الوقت

وإنما ينتقل التنفيذ من برنامج لآخر بالترتيب (وستتضح فكرة تعدد المهام في القسم الثاني من الموضوع)

المسجلات المتوفرة لهذة الأنظمة هي:

المسجلات العامة AX,BX,CX,DX,SI,DI,BP,SP :

مسجلات الأقسامES,SS,CS,DS :

وكل هذة المسجلات حجمها ١٦ بت (ولذلك سميت أنظمة ١٦ بت)

هذة النقاط تكفي للدخول في الموضوع ،،، سيتم إختيار النظام دوس من بين أنظمة ١٦ بت

لأنة متوفر عند الأغلبية لتطبيق أمثلة الدرس والنقطة الثانية أن كل الأنظمة متشابهة في طريقة عملها

ذكرنا في الدرس السابق أن آخر عملية يقوم بها البيوس هي تحميل قطاع الإقلاع ونقل التحكم للنظام

ماذا يكتب نظام الدوس في قطاع الإقلاع ؟

كيف يقوم نظام الدوس ببناء بيئة لنظام التشغيل ؟

يبدأ البيوس بتحميل قطاع الإقلاع إلى العنوان hex =0000:7C00 بمعنى أن هذا العنوان هو بداية برنامج الإقلاع

في النظام دوس يبدأ برنامج الإقلاع بتعليمة قفز إلى الإزاحة hex =3E والتعليمة هيJMP 3E

ولذلك فإن بداية التنفيذ الفعلي لبرنامج الإقلاع يبدأ عند العنوانhex =0000:7C3E

لماذا يتجاوز النظام دوس ٦٢ بايت تقريبا من بداية القرص ؟!

السبب لأن مقدمة القرص تحتوي على معلومات عن القرص يخزنها نظام التشغيل (عند عملية الفورمات)

وهذة المعلومات هي عدد القطاعات في كل تجمع وعدد التجمعات وعدد البايتات في كل قطاع ... وهكذا

وهي ما تعرف بمقدمة القرص , وتجد المعلومات عن كل بايت في هذة المقدمة هنا:

http://www.arabteam2000-forum.com/index.php?showtopic=44877

يبدأ التنفيذ الفعلي لبرنامج الإقلاع عند العنوان hex =0000:7C3E

بداية برنامج الإقلاع : الخطوة الأولى

```
0000:7C3E    CLI
0000:7C3F    XOR     AX,AX
0000:7C41    MOV     SS,AX
0000:7C43    MOV     SP,7C00
0000:7C46    PUSH    SS
0000:7C47    POP     ES
```

يبدأ برنامج الإقلاع بتصفير أعلام أو رايات المقاطعات عن طريق التعليمة الأولى

وهذة الطريقة لا تعتبر شرط لبداية الإقلاع لأن برامج الإقلاع لأنظمة IBM لا تستخدم هذة التعليمة

المهم بعد ذلك يصفر المسجل ax ومن خلالة يصفر قسم المكدس وهوss

ويعطا المسجل sp قيمة تمثل بداية البرنامج في الذاكرة ويتم دفع القيمة للمكدس

لماذا ؟ لأن المسجل sp مرتبط بالمقطع ss ويمثلان مكدس البرنامج وهو ss:sp ويساوي 7C00:0000

وبهذا يكون قد عرف مكدس البرنامج لتخزين المعلومات المؤقتة أو المتغيرات في البرنامج

الخطوة الثانية:

```
0000:7C48    MOV     BX,0078
0000:7C4B    SS:
0000:7C4C    LDS     SI,[BX]        // DS:SI = [0000:0078]
0000:7C4E    PUSH    DS
0000:7C4F    PUSH    SI
0000:7C50    PUSH    SS
```

```
0000:7C51    PUSH    BX
```

هذة الخطوة للتغيير في بعض خصائص القرص (بمعنى أن النظام دوس سيغير في خصائص القرص للبيوس) ؟؟

لاحظ التعليمة الأولى المسجل bx = 0078 بعد هذة التعليمة تلاحظ أن باقي التعليمات تنفذ داخل المقطع ss

وقيمة ss = 0000 والتعليمة lds تحمل مقطع البيانات بالقيمة صفر والمسجلsi = 0078

والعلاقة هي DS:SI أي أن المسجل والمقطع مرتبطين مع بعض مثل الخطوة الأولى (وهذة قوانين في لغة الإسمبلي)

المهم كل هذة الشغلة هي لجعل مقطع البيانات يمثل العنوان ٠٠٠٠:٠٠٧٨ بعد إرتباطة بهذة الطريقةDS:SI

ماذا يمثل العنوان ولماذا هذة الشغلة ؟! معرفة هذا العنوان سيحل اللغز الأول ؟ لاحظ

هل قرأت موضوع برمجة البيوس ؟ راجع الجدول الأول وهو لعناوين المقاطعات وخاصة المقاطعةINT 1E

ستجد عنوانها بهذة الطريقة INT 1E =0000:0078 وهي خاصة لجدول وبارمترات القرص ؟! أي خصائص القرص المرن

في هذا الجدول يخزن البيوس معلومات القرص الذي حمل منة قطاع الإقلاع ؟ فيقوم الدوس بقرائتها والتغيير حسب طلبة

كيف ؟ بهذة الطريقة:

```
0000:7C52     MOV       DI,7C3E
0000:7C55     MOV       CX,000B
0000:7C58    CLD
0000:7C59    REPZ
0000:7C5A    MOVSB
0000:7C5B    PUSH      ES
0000:7C5C   POP        DS
```

تلاحظ أن الدوس نقل بيانات القرص إلى نفس مقطع البرنامج وقد يكون أي عنوان في مقطع البرنامج

والملاحظة الثانية أن حجم البيانات تساوي B أي ١١ بايت

بعد ذلك يغير الدوس بعض الخصائص بهذة الطريقة

```
0000:7C5D   MOV    BYTE PTR [DI-02],0F
0000:7C61   MOV    CX,[7C18]
0000:7C65   MOV    [DI-07],CL
0000:7C68   MOV    [BX+02],AX
0000:7C6B   MOV    WORD PTR [BX],7C3E

0000:7C6F  STI
0000:7C70  INT    13
```

أول تعليمة يقوم نظام الدوس بتغيير خاصية تسمى مدة الإنتظار لرأس القرائة والكتابة في القرص

وهي إحدى خصائص القرص ويحدد المدة بالقيمة F (أعتقد أنها لتسريع القرائة)

التعليمة الثانية والثالثة : يقوم نظام الدوس بتغيير عدد القطاعات في كل مسار (لكي يحدد قطاعات ملف محدد)

ولو تلاحظ أنة قرأ القيمة الموجودة في مقدمة القرص والإزاحة هي ١٨ وتمثل عدد القطاعات في المسار

بعد ذلك يقوم بنقل الخصائص بعد تعديلها ، والتعليمة sti هي لإعادة تفعيل المقاطعات

وآخر شيء يقوم بتنفيذ مقاطعة إعادة تهئية القرص ،، ويصبح القرص جاهز بعد التغيير

والآن يبدأ النظام بيوس بتحميل الملفات وتشغيل النظام ،، كيف ؟

يقوم بقرائة مقدمة القرص ويستخرج كل المعلومات الموجودة . بهذة الطريقة

```
0000:7C84    MOV    AL,[7C10]
0000:7C87    MUL    WORD PTR [7C16]
0000:7C8B    ADD    AX,[7C1C]
0000:7C8F    ADC    DX,[7C1E]
0000:7C93    ADD    AX,[7C0E]
0000:7C97    ADC    DX,+00
0000:7C9A    MOV    [7C50],AX
0000:7C9D    MOV    [7C52],DX
0000:7CA1    MOV    [7C49],AX
0000:7CA4    MOV    [7C4B],DX
```

التعليمة الأولى يتعرف على عدد جداول نظام الملفات FAT وهي الإزاحة 10

والتعليمة الثانية عدد قطاعات الجدول فات الإزاحة ١٦ ،، ثم عدد القطاعات المخفية وهكذا

راجع مقدمة القرص لتعرف كل قيمة ماذا تمثل ، وكيفية الإستفادة منها

المهم أنة بعد الحسابات يحدد قطاعات ملف معين لبداية تشغيل النظام

وفي مثالنا فإن النظام دوس يبحث عن قطاعات الملف IO.sys يقوم بتحميلة إلى العنوان0070:0000 hex

ملاحظة : يتم قرائة أول ٣ قطاعات من الملف فقط وبعد ذلك تنقل للذاكرة

وأكيد بإستخدام الخدمة ٢ من المقاطعة INT 13 (بنفس الطريقة التي قمنا بتطبيقها في موضوع البيوس)

وآخر خطوة في برنامج الإقلاع هي تعلمية قفز إلى العنوان ٧٠٠(وهي أول٣ قطاعات من الملف) io.sys

هل لاحظت الدورة الغير منتهية ؟! وأهم شيء هل لاحظت تشابة الأفكار ؟!

المعالج يبدأ بقفزة إلى البيوس ثم يقوم البيوس بتحميل قطاع الإقلاع إلى العنوان0000:7C00 hex

ثم ينتهي البيوس بقفزة إلى قطاع الإقلاع

ثم يبدأ قطاع الإقلاع بقفزة ويبدأ التنفيذ ثم يحمل برنامج الإقلاع نواة النظام إلى العنوان0070:0000 hex

ثم ينتهي برنامج الإقلاع بقفزة إلى نواة النظام

وبعد ذلك تبدأ صناعة بيئة نظام التشغيل ؟!

في الحقيقة هذة المعلومات لقطاع الإقلاع متشابة في كل أنظمة التشغيل بما فيها ١٦ و ٣٢ بت

ولآن ماهي خطوات صناعة بيئة لنظام التشغيل لمنصة ١٦ بت ؟

قبل البدء في هذة النقطة ، لدي ملاحظة مهمة وهي:

لا تعتقد أن تحميل نواة النظام إلى الذاكرة هي عبارة عن تحميل ملف معين من القرص

والنقطة الثانية وهي الأهم : لا تعتقد أن محتويات القرص هي الملفات الموجودة بة فقط ؟!!

سؤحاول توضيح الفكرة ،، هل لاحظت برنامج الإقلاع الذي عرضنا بعض شفرتة وهو موجود في القطاع الأول

هل يوجد ملف في القرص يدل على هذا البرنامج ؟؟ أكيد لا رغم أنة موجود فعلا في القرص

وتستطيع أن تكتب برنامج في محتوى القرص لتحميل النظام .. دون وجود أي ملف مرئي في القرص

وستكون طريقة تحملية للذاكرة أسهل وأفضل لأن محتوياتة ستكون في قطاعات متتالية
ولكنة سيكون غامض للمستخدم

بإختصار أريد أن أصل لنقطة ، وهي أن النظام دوس يحمل برامج لصناعة بيئة للنظام من أماكن متفرقة

في القرص قد تكون من بعض الملفات المرئية أو من قطاعات مخفية لا توجد لها ملفات في القرص!

المهم : تبدأ نواة النظام في العمل .. خطوات بناء بيئة للنظام:

الخطوة الأولى : يتم بناء جدول مقاطعات الدوس وجدول معلومات الدوس ؟ أكيد مرت عليك هذة الجداول؟!

هل راجعت الموضوع السابق (برمجة البيوس) تقصدت شرح جدولين بالتفصيل وهما جدول مقاطعات البيوس وجدول معلومات البيوس ؟! فالنظام دوس سرق الفكرة من البيوس وبنفس التخطيط مئة بالمئة

والآن دعنا نراجع:

ذكرنا أن أول جدول هو جدول مقاطعات البيوس عند العنوان ٠٠٠٠:٠٠٠٠

بعد ذلك جدول معلومات البيوس عند العنوان ٠٠٤٠:٠٠٠٠

بعد ذلك يبدأ الدوس في كتابة جدول للمقاطعات عند العنوان الذي توقفنا عندة في الدرس السابق

ثم تتم كتابة جدول بيانات الدوس عند العنوان ٠٠٥٠:٠٠٠٠ حجم هذا الجدول ٢٠٠ بالعنونة =٠٠٢٠:٠٠٠٠

لوجمعت العنوان ٠٠٥٠:٠٠٠٠ و ٠٠٢٠:٠٠٠٠ فالنتيجة هي ٠٠٧٠:٠٠٠٠ ماذا يمثل العنوان

بداية برنامج نواة نظام التشغيل الذي إنتقل لة التنفيذ من قطاع الإقلاع ،، لاحظ الترتيب الدقيق جدا

ولا بايت فارغ من بداية الذاكرة إلى هنا ؟

إذ تعمد النظام من جعل مساحة فارغة تسبق عنوان نواة النظام ليتم كتابة الجداول بها,

والسؤال : ماهو البرنامج الذي كتب الجدولين للدوس وكيف ؟

أولا : الجدولين عبارة عن أرقام وبيانات لترتيب ونقل التنفيذ مثلا مقاطعة الدوس int 21

تتصل بجدول المقاطعات ، جدول المقاطعات ينقل التنفيذ إلى برنامج آخر في الذاكرة لتنفيذ المقاطعة

بمعنى أنة لإنشاء جداول تخطيط في الذاكرة لا نحتاج لأي تعليمة للعتاد أو المنافذ أو مقاطعة للبيوس

فقط نحتاج لتعليمات نقل المعلومات مثل mov ولذلك تجد البرنامج io.sys لا يحتوي على أي مقاطعة

خارجية ، فقط يحتوي على بعض تعليمات الفحص والمنافذ

والنتيجة : أن المسؤل عن بناء الجدولين هو البرنامج io.sys وقد تتدخل بعض قطاعات القرص المخفية

والآن أنشأ نظام الدوس الجدولين ولكن جدول المقاطعات ينقل كل مقاطعة إلى عنوان لتنفيذ المقاطعة

أين هذة العناوين وكيفية إنشائها ؟

يبدأ برنامج io.sys بتحميل جزء من الملف msdos.sys وهذا الملف لا يحتوي على أي تعليمة

فقط يحتفظ بإعدادات تخص الملف io.sys

وبعد ذلك يبدأ بتحميل باقي ملفات القرص بعد تجزئتها حسب العنوان التي ستتصل بة مقاطعة الدوس

كيف ؟

مثلا الملف command.com يقرأ منة ٥ قطاعات مثلا وينقلها لعنوان التنفيذ لمقاطعة محددة

وعندما تنفذ المقاطعة ينتقل التنفيذ لجدول المقاطعات بعد ذلك ينقل جدول المقاطعات التنفيذ لعنوان الخمس قطاعات

وهكذا مع بقية الملفات والقطاعات التي لا يوجد لها ملفات تعبر عنها في القرص

والآن مثال لتوضيح الفكرة ؟!

من قائمة start ثم run إكتب command لتشغيل برنامج الدوس

أكتب الأمر التالي mem /p : لإستعراض محتوى الذاكرة

ولاحظ الترتيب الذي سيظهر ، أول سطر جدول المقاطعات وعنواة ٠ ثم جدول بيانات البيوس وعنوانة40

ثم جدول بيانات الدوس وعنوانة ٥٠ وبعد ذلك لاحظ تقسيم البرامج مثلا io.sys ستجدة أكثر من قسم

وكذلك باقي الملفات مثل msdos و ,... command بإختصار هذة هي فكرة أنظمة ١٦ بت

والإختلاف بسيط بين الدوس واليونكس ١٦ بت وهو عبارة عن ترتيب للمقاطع التي تم ذكرها
فمثلا اليونكس يكتب معلوماتة بعد جدول معلومات البيوس بهذا الترتيب
-نواة النظام-
-ذاكرة مؤقتة لتحميل البرامج-
-وهنا يكتب جداول المعلومات-
-هنا يخصص قسم كبير يستخدم مكدس لكل البرامج-

وفي النهاية هذة الأساليب لبرمجة أنظمة ١٦ بت لم تعد موجودة أو نادرة الإستخدام

ولكن الهدف من ذكرها لتكون مدخل لأنظمة ٣٢ بت مثل وندوز ولينكس

لتكتشف الفرق الكبير ،،، والتحول من المقاطعات إلى الدوال API والبوابات والإمتيازات والمهام

وأفضل طريقة لكشف أسرار بناء الأنظمة هي برامج المتابعة والتصحيح وهي متوفرة

مثل برنامج debug في دوس والبرامج التجارية مثل السوفت آيس وبرنامج TD من بورلند وغيرها الكثير

وعند متابعة أي نظام يجب أن تعرف أن سياسة الأنظمة هي : الترتيب والغموض ؟

الترتيب : وهو عمل النظام وأفكارة دائما تأتي تحت ترتيب أو قانون محدد إما أن يكون مأخوذ من البيوس

كما شاهدنا أفكار نظام الدوس أو تكون فكرة جديدة بترتيب معين
والترتيب هو أساس في صناعة الأنظمة

الغموض : وهو أن النظام يحاول بكل الطرق أن يظهر مئة علامة تعجب فوق رأسك ؟ كيف قام بهذا العمل

لأن شركات الأنظمة تسرق من بعضها الأفكار لمحاولة تطوير أنظمتها ،، ودائما تجد بعض الطرق في الأنظمة

تلف وتدور للوصول لهدف يمكن الوصول لة بخطوة واحدة ببساطة ووضوح ؟ فلا تستغرب من هذة الدورة

أنظمة ٣٢ بت (وندوز && لينكس ..)

إنتشر المعالج (x86 أول معالج ١٦ بت) بشكل رهيب وحقق تحول كبير في صناعة الأنظمة ١٦ بت وظهرت لهذا المعالج أنظمة مثل دوس وينكس وغيرها ، وبالتأكيد حققت الشركات أرباح شبة خيالية مما جعل شركة إنتل تستمر في البحث لإنتاج معالج مطور يحدث نقلة أخرى وأرباح أكثر ؟؟

وفي سنة ١٩٨٨ ـ ١٩٨٩ تقريبا طرحت شركة إنتل معالج الأحلام وهو المعالج ٨٠٣٨٦ إختصارةx386 وهو المعالج ٣٢ بت الذي يملك كل الخدمات مثل الإمتيازات وتعدد المهام ومسجلات ٣٢ بت والأنماط الرسومية ولكن شركة إنتل لا تريد أن تخسر أنظمة المعالج ١٦ بت ، فجعلت المعالج يعمل في المنصتين ١٦ بت و٣٢ بت الآن عرفنا أن المعالج ٣٢ بت ظهر تقريبا في سنة ٨٩ ولكن شركات الأنظمة تحتاج وقت لفهم عمل

المعالج٣٢ بت وبناء أنظمة التشغيل له ,, بعد ٦ سنوات تقريباً ظهر النظام وندوز ٩٥ وهو نظام ٣٢ بت ؟!!

نظام الوندوز ٩٥ أول نظام يستخدم كل خدمات المعالج ٣٢ بت وهو بداية في عصر الأنظمة الرسومية

وبهذا هيمت شركة مايكروسفت على سوق البرمجيات والأنظمة وأصبح صانعها الأغنى في العالم!

ماهو الجديد في أنظمة ٣٢ بت ؟

ذكرنا في الدرسين السابقين أساسيات في برمجة أنظمة ١٦ بت ووجدنا أن النظام يستخدم مسجلات المعالج

في البرمجة وهي نفسها المسجلات التي تستخدمها البرامج ، وإكتشفنا أن أنظمة التشغيل تستخدم نفس أفكار نظام البيوس

بمعنى أن أنظمة ١٦ بت معظم أفكارها مستهلكة ... وطرق برمجتها واضحة ،،، ولكن في أنظمة ٣٢ بت الأمر يختلف!

المسجلات التي تستخدمها أنظمة التشغيل ٣٢ بت:

المسجلات العامة تغيرت لتصبح EAX,EBX,ECX,EDX,ESI,EDI,EBP,ESP :

مسجلات الأقسام ES,SS,CS,DS : وأضيف لهاGS,FS

هذة المسجلات كانت في أنظمة ١٦ بت وتم تطويرها لتتسع إلى ٣٢ بت أي ٤ بايت ولهذا سميت أنظمة ٣٢ بت

لو كان التطوير فقط لهذة المسجلات لما تطور النظام ،،، فمثلا الخدمات الجديدة مثل:

الذاكرة الوهمية وصفحات الذاكرة , النمط المحمي والبوابات ، تعدد المهام ، مراقبة أخطاء الهاردوير..

كل هذة الخدمات تحتاج لمسجلات جديدة ليستخدمها نظام التشغيل ،، وسنبدأ بشرح مسجلات ٣٢ بت الجديدة

مسجلات نظام التشغيل (الجداول) وهي:

IDTR -IDT :جدول المقاطعات (ملاحظة : المسجل idtr يمثل الجدول ، والمسجل idt يمثل دليل للجدول)
GDTR-GDT : مسجلات جدول الواصفات الشامل
LDTR -LDT : مسجلات جدول الواصفات المحلي
TSS -TR : مسجلات تعدد المهام

مسجلات التحكم للمعالج والذاكرة وهي :

CR0 و CR1 و CR 2 و CR3 وCR4

مسجلات الإختبار والتنقيح للبرامج والهاردوير:

TR0,TR1,TR2,TR3,TR4,TR5,TR6,TR7

تعليمات بعض المسجلات السابقة:

LGDT, LIDT, LLDT, LMSW, LTR,
SGDT, SIDT, SLDT, SMSW, STR,
RDMSR, WRMSR, RDTSC

وبقية المسجلات تستخدم التعليمة MOV وسيتم عرض الأمثلة في نهاية الموضوع

نبدأ بشرح الجداول,,,

جدول الواصفات الشامل والمحلي:

ماهي الجداول ؟ في أنظمة ٣٢ بت
تم حجز مساحات فارغة في الذاكرة يستخدمها النظام كجداول

مثل جدول الواصفات الشامل والمحلي و جداول الذاكرة وجدول المقاطعات ، وتستطيع إستخراج

أي قيمة في أي جدول من خلال معلومتين هما : الأساس وهو يمثل عنوان الجدول

والحد وهو يمثل دليل للجدول ، وسنقوم بعرض هذة الجداول والمعلومات التي تحتويها

جداول الواصفات من إسمها تعني : وصف عناوين المقاطع ووصف لخصائص المقاطع وأنواعها ،، ماهو المقطع ؟!

نعرف بأن نظام التشغيل عبارة عن برامج (عمليات) البرامج عبارة عن مقاطع ، مثل مقطع الكود ومقطع البيانات

ومقطع الكود يمثلة المسجل CS والبيانات المسجل , DS ماذا لو فتحت أي برنامج بواسطة الديبغر وأردت أن تعرف

عنوان مقطع الكود مثلاً ، أنا جربت بواسطة برنامج olly وظهر لي في النافذة اليمنى أن مقطع الكود يساوي CS =21 ؟؟

ومقطع البيانات مثلا , DS =1B هل من المعقول أن هذا العنوان في الذاكرة يمثل عنوان كود البرنامج ؟! أكيد لا

جرب شغل أي برنامج آخر بواسطة الديبغر ,, ستجد أن عناوين المقاطع نفسها في البرنامج السابق لم تتغير ؟

هل كل البرامج لها عنوان واحد في الذاكرة ؟ بإختصار هل العناوين التي نراها حقيقية أم أن النظام يخدعنا ؟!!

الجواب هو جدول الواصفات الشامل ،،، ماهو جدول الواصفات الشامل ؟

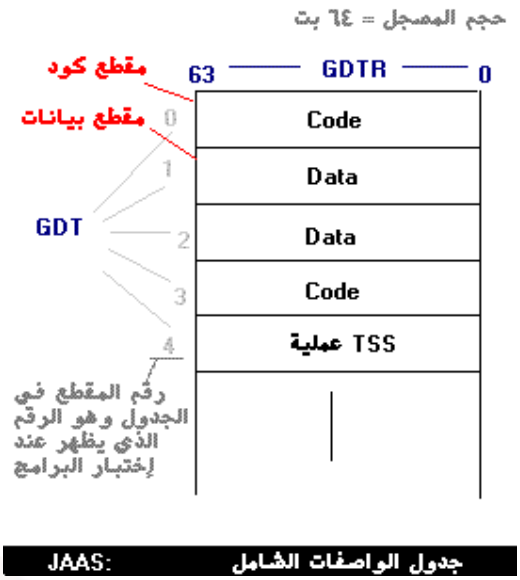
يتكون الجدول من مسجلين الأول GDTR ويمثل الجدول(الأساس) ، والمسجل GDT يمثل الدليل (الحد)

مثلا لو كان GDT =10 يعني السطر ١٠ في الجدول GDTR

والمعلومة الثانية أن المسجل GDTR حجمة ٦٤ بت (يحتوي على العنوان الذاكري للقسم وخصائصة)

وبهذا سنفهم الفكرة ، نظام التشغيل يعطي أي عملية في الذاكرة (مثلا مقطع كود) يعطية رقم مثلا ٢٠

هذا الرقم يمثل رقم السطر في جدول الواصفات الشامل ،، بهذا الشكل



جدول الواصفات الشامل: JAAS

الملاحظة الأولى : الجداول في الذاكرة تبدأ الترتيب من الأسفل إلى الأعلى، مثل مقطع المكدس في البرامج

الملاحظة٢ : جدول الواصفات لا يحتوي على مقاطع للبيانات والكود فقط ، وإنما يحتوي على أكثر من نوع

للعمليات مثل TSS مسجل حالة لعملية معينة ! و قد يحتوي LDT وهو يمثل إزاحة في جدول آخر

وقد يحتوي على إتصال لبوابة أو مقاطعة لبوابة أو مصيدة لبوابة ، المهم أن القاعدة هي

جدول الواصفات الشامل هو المسؤل عن نقل التحكم ؟إما إلى جدول آخر مثل جدول المقاطعات أو الجدول المحلي

أو إلى جدول عناوين الذاكرة أو إلى عملية في الذاكرة ، وستتضح الفكرة بعد قليل!

ولآن لو أخذنا أي قيمة من الجدول مثل القيمة ١ قسم بيانات , حجمها سيكون ٦٤ بت ، ماذا تمثل هذة ٦٤ بت

يقوم نظام التشغيل بكتابة كل معلومات العملية أو القسم في هذة الخانة ، مثل العنوان الذاكري الذي سيتجة لة التنفيذ

نوع القسم مثل كود أو بيانات أو معرف لجدول أو بوابة ، ويكتب إمتيازات القسم مثلا قسم للنظام أو للمستخدم

ومعلومات أخرى كثيرة ،، يستفيد منها النظام في تنظيم القيم وتحديد مستويات التنفيذ ، دعنا نشرح كل بت بالتفصيل!

التقسيم العام لقيم جدول الواصفات الشامل , GDTR حجمة ٦٤ بت:

البتات من ٠ إلى ١٥ + البتات من ٤٨ إلى ٥١ : يمثل الحد (للإنتقال لجدول آخر)

البتات من ١٦ إلى ٣٩ + البتات من ٥٦ إلى ٦٣ : يمثل الأساس (وهو عنوان الجدول)

وهذة القيم تمثل عنوان الإنتقال للتنفيذ إما لجدول واصفات آخر أو إلى جدول عناوين وصفحات الذاكرة

وبعد ذلك يتم الإنتقال الفعلي للكود التنفيذي ، نرجع لتقسيم ٦٤ بت ، بقية البتات تمثل خصائص المقطع:

البت من ٤٠ إلى ٤٣ : وعددها ٤ بتات والقيمة التي تحتويها هذة ٤ البتات تمثل خصائص القسم,

تنقسم خصائص القسم إلى نوعين ، الأول خصائص القسم لعملية النظام ، وخصائص القسم لعمليات المستخدم

والذي يحدد نوع العملية هو البت ٤٤ الذي يلي هذة ٤ البتات ، إذا كان بت نوع العملية ٠ فإن العملية

تكون للنظام ، وبذلك فإن خصائص أقسام عمليات النظام المكونة من ٤ بت تكون بالشكل التالي

القيمة ٠ : محجوزة للمستقبل (لايوجد)
القيمة ١ : نوع القسم عملية TSS من نوع ١٦ بت
القيمة ٢ : نوع القسم دليل لجدول LDT وهو يمثل دليل لجدول الواصفات المحلي
القيمة ٣ : عملية TSS من نوع ١٦ بت وهي في حالة التنفيذ بمعنى عملية مشغولة
القيمة ٤ : نوع العملية إتصال لبوابة من نوع ١٦ بت
القيمة ٥ : نوع العملية بوابة للتنفيذ
القيمة ٦ : مقاطعة لبوابة من نوع ١٦ بت
القيمة ٧ : مصيدة لبوابة من نوع ١٦ بت
القيمة ٨ : قيمة محجوزة للمستقبل
القيمة ٩ : عملية TSS من نوع ٣٢ بت
القيمة ١٠ : محجوز للمستقبل
القيمة ١١ : عملية TSS من نوع 32 بت ، مشغولة
القيمة ١٢ : إتصال لبوابة ٣٢ بت
القيمة ١٣ : محجوز
القيمة ١٤ : مقاطعة لبوابة ٣٢ بت
القيمة ١٥ : مصيدة لبوابة ٣٢ بت

أما إذا كان نوع العملية (عملية مستخدم) بمعنى أن البت ٤٤ يساوي ١

تكون خصائص القسم لل ٤ بت بهذا الشكل

القيمة ٠ : قسم بيانات للقرائة فقط
القيمة ١ : قسم بيانات للقرائة + إمكانية الوصول
القيمة ٢ : قسم بيانات للقرائة والكتابة
القيمة ٣ : قسم بيانات للقرائة والكتابة + إمكانية الوصول
القيمة ٤ : قسم بيانات للقرائة فقط + قسم موسع
القيمة ٥ : قسم بيانات للقرائة فقط + قسم موسع + إمكانية الوصول
القيمة ٦ : قسم بيانات للقرائة والكتابة + قسم موسع
القيمة ٧ : قسم بيانات للقرائة والكتابة + قسم موسع + إمكانية الوصول
القيمة ٨ : قسم شفرة قابل للتنفيذ فقط
القيمة ٩ : قسم شفرة تنفيذي + إمكانية الوصول
القيمة ١٠ : قسم شفرة تنفيذي + للقرائة
القيمة ١١ : قسم شفرة تنفيذي + للقرائة + إمكانية الوصول
القيمة ١٢ : قسم شفرة تنفيذي متوافق
القيمة ١٣ : قسم شفرة تنفيذي+ متوافق + إمكانية الوصول
القيمة ١٤ : قسم شفرة تنفيذي + للقرائة + متوافق
القيمة ١٥ : قسم شفرة تنفيذي + للقرائة + متوافق + إمكانية الوصول

هذة هي نفسها خصائص أقسام البرامج في نظام وندوز ، وهي نفسها خصائص أقسام جداول الواصفات
-
بعد ذلك يأتي البت ٤٤ في جدول الواصفات الشامل ، نكمل باقي ٦٤ بت

البت ٤٤ : هذا البت يحدد إذا كانت العملية للنظام أو للمستخدم ، وهو البت المسؤل عن كيفية التقسيم

لخصائص الأقسام كما ذكرناة سابقاً ، إذا كان ٠ عملية نظام وإذا كان 1 عملية مستخدم

البت ٤٥ و ٤٦ : يمثل رقم الإمتياز أو الحلقة التي ستعمل بها هذة العملية ، والحلقات من ٠ إلى ٣

البت ٤٧ : يمثل هذا البت وجود القسم الحالي في الذاكرة أو عدم وجودة ؟!
يستفيد النظام من هذا البت في أمور كثيرة منها مثلا معرفة الأقسام النشطة والغير نشطة لنقل محتواها من القرص إلى الذاكرة والعكس ، كما أن البت يستخدم للتبديل بين الأقسام

البت ٥٢ : هذا البت إحتياطي ؟! إلى أن يتم التبديل بين الأقسام فيصبح البت مكمل لقيم الحد بمعنى أن البت لا يستخدم في هذا التقسيم وسيتم إستخدامة في تقاسيم أخرى ,

البت ٥٣ : هذا البت غير مستخدم دائما ٠ محجوز للمستقبل

البت ٥٤ : إذا كان قيمة البت ٠ فإن المقاطع تكون ١٦ بت وإذا كان ١ تصبح المقاطع ٣٢ بت

البت ٥٥ : هذا البت يتسخدم في طريقة التعامل مع قيمة الحد للعملية ، إذا كان ٠ فإن المعالج يحسب قيمة الحد كماهي ، وإذا كانت قيمة البت ١ يستخدم المعالج قانون جديد ، وهو حساب كل قيمة للحد ٤ كيلوبايت

-

وبهذا نكون أكملنا كل ٦٤ بت لعملية نظام داخل جدول الواصفات الشامل ,

بقية معلومة وهي : يوجد تقسيم آخر لجدول الواصفات الشامل , إذا كانت العملية تخص البوابات

فإن بعض الخصائص في ٦٤ بت تختلف - والإختلاف بسيط بهذا الشكل,,

تقسيم جدول الواصفات لعملية نوعها (إتصال لبوابة)

عملية الإتصال لبوابة تستخدم تقسيم آخر غير التقسيم العام ولكن مشابة بشكل كبير للتقسيم العام

فمثلا البتات من ٥٢ إلى ٥٥ : لا تستخدمها البوابة ، وتضاف بتات جديدة وهي

البت من ٣٢ إلى ٣٦ : وتستخدمها البوابة لتمرير مجموع البارمترات التي ستمرر عبر البوابة

أما بقية البتات : الخصائص ومستوى الإمتيازات و.و. ، تبقى كما هي

نستنتج : أن أي عملية تخص البوابات مثل مقاطعة لبوابة أو مصيدة لبوابة فإنها لا تستخدم

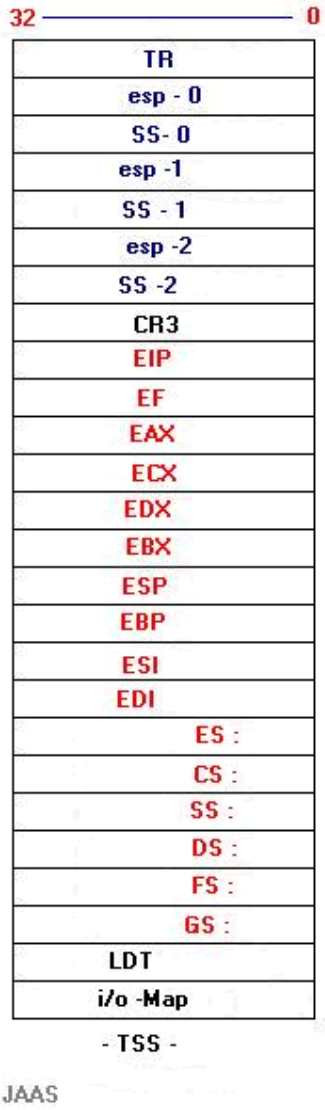
البتات من ٥٢ إلى ٥٥ وبقية الخصائص تبقى كما هي

-

التبديل بين المهام وتقسيم المسجل TSS تقسيم:

من أعقد العمليات في نظام التشغيل ٣٢ بت هي عملية التبديل بين المهام ، ولكن المعالج يتكلف بأغلب الشغلة ، تستطيع أن تقول أن كل عملية في الذاكرة عبارة جدول معلومات يمثلة المسجلTSS

بهذة الصورة:



- TSS -

JAAS

تلاحظ القيم التي باللون الأزرق : المسجل TR يمثل رقم العملية السابقة التي إنتقل منها التنفيذ للعملية الحالية

مثلا كنت تعمل على برنامج ومن خلالة شغلت برنامج آخر ، يتم تسجيل رقم البرنامج الأول في هذا المسجل

ليس بالتحديد رقم البرنامج الذي كنت علية ؟!! ولكن لتقريب الفكرة ، يمكن تكون عملية نظام سبقت التنفيذ

وبهذا يصبح ترابط بين العمليات في الذاكرة ، مثال : يبدأ النظام بأول عملية (أكيد مافي شيء قبلها تكون هذة القيمة ٠)

بعد ذلك تشتغل عملية أخرى ترتبط مع العملية التي قبلها بهذا المسجل وهكذا مع ثالث عملية ، وتصبح كل العمليات مترابطة

ليقوم المعالج بسرعة كبيرة بالتبديل بين هذة المهام من الأحدث إلى الأقدم ، وبهذا تعتقد أن لكل عملية معالج خاص ؟!!

وتلاحظ مسجلات المكدس ss0 و .. ss1 ودليل المكدس esp1 و esp2 (هذة المسجلات آخر ٣ عمليات)

سبقت هذة العملية ، والسؤال لماذا إختار المعالج مسجل المكدس لثلاث عمليات ، لماذا لم يختر مسجل الكود مثلاً

لأن أي عملية إتصال تخرج التنفيذ من البرنامج إلى كود خارج البرنامج ، فإنة يتم تسجيل معلوماتها في المكدس

مثل عنوان الإتصال و عنوان العودة للبرنامج ، سأعطيك مثال على هذة النقطة ؟!! تخيل

أنك تعمل على الكمبيوتر وفجأة ظهرت رسالة خطأ في النظام ثم يغلق النظام ؟ كيف تعرف مصدر الخطأ

عن طريق عرض العمليات المقيمة في الذاكرة , وإلقاء نظرة على مكدس البرنامج لتجد آخر عملية خروج من البرنامج

وستجد عنوان العودة ، إذهب إلية ولاحظ الكود الذي تسبب في الخطأ ، هذة النقطة سنتفيد منها كثير في البرمجة العكسية

المهم : وما غرض المعالج من معرفة المكدس لآخر ثلاث عمليات ؟! لكي يعرف مكان التنفيذ قبل الإنتقال للعملية

إذا كان التنفيذ خارج البرنامج ، مافائدت الإنتقال للعملية بشكل مباشر ؟! ولذلك فإنة يتم التبديل بين المهام

بسرعة أكبر ، بعد ذلك يأتي المسجل CR3 هذا المسجل خاص للصفحات الذاكرية للعملية

وسيتم مناقشتة في القسم الثاني من الموضوع (مسجلات التحكم للمعالج والنظام)

وبعد ذلك المسجلات التي باللون الأحمر : وهي بالطبع المسجلات العامة ومسجلات الأقسام التي تخص العملية

الحالية ، يحتفظ بها المعالج عندما يبدل بين العملية وعندما يرجع إليها يعيد القيم للمسجلات

المسجل LDT وهو دليل لهذة العملية في جدول الواصفات المحلي ، وتستخدم لمعالجة الإستثنائات في العملية

وهذة القيمة لاتستخدم في النظام ، دائما صفر

بقي آخر قيمة : تخطيط الذاكرة لعناوين الإدخال والإخراج في البرنامج , وهذا الموضوع سيكون لة شرح

أكثر عندما ندخل في طرق البرمجة و المقاطعات للهاردوير + مقاطعات البرامج

والآن سنقوم بمراجعة سريعة للجداول لتوضيح طرق الترابط بينها ، وأماكن تواجدها

علمنا أن أي عملية في نظام التشغيل يمكن أن تكون:

قسم بيانات - قسم كود - عملية أو معرف للمسجل - TSS دليل لجدول آخر مثلLDT

أو إتصال لبوابة - مقاطعة لبوابة - عملية لبوابة - مصيدة لبوابة

وأي عملية من هذة العمليات يمثلها ٦٤ بت(جداول الواصفات) هذة ال ٦٤ بت تمثل عنوان الإنتقال + الخصائص

والسؤال أين المكان المناسب لكل عملية في أي جدول ، الجداول هي

جدول الواصفات الشامل : GDTR العمليات التي يمكن أن يحتويها ٥ وهي:

إما أن تكون قسم كود - أو قسم بيانات - أو معرف - TSS أو دليل لجدول - LDT أو إتصال لبوابة

جدول المقاطعات , IDTR يمكن أن يحتوي على ٣ وهي : عملية لبوابة - أو مقاطعة لبوابة - أو مصيدة لبوابة

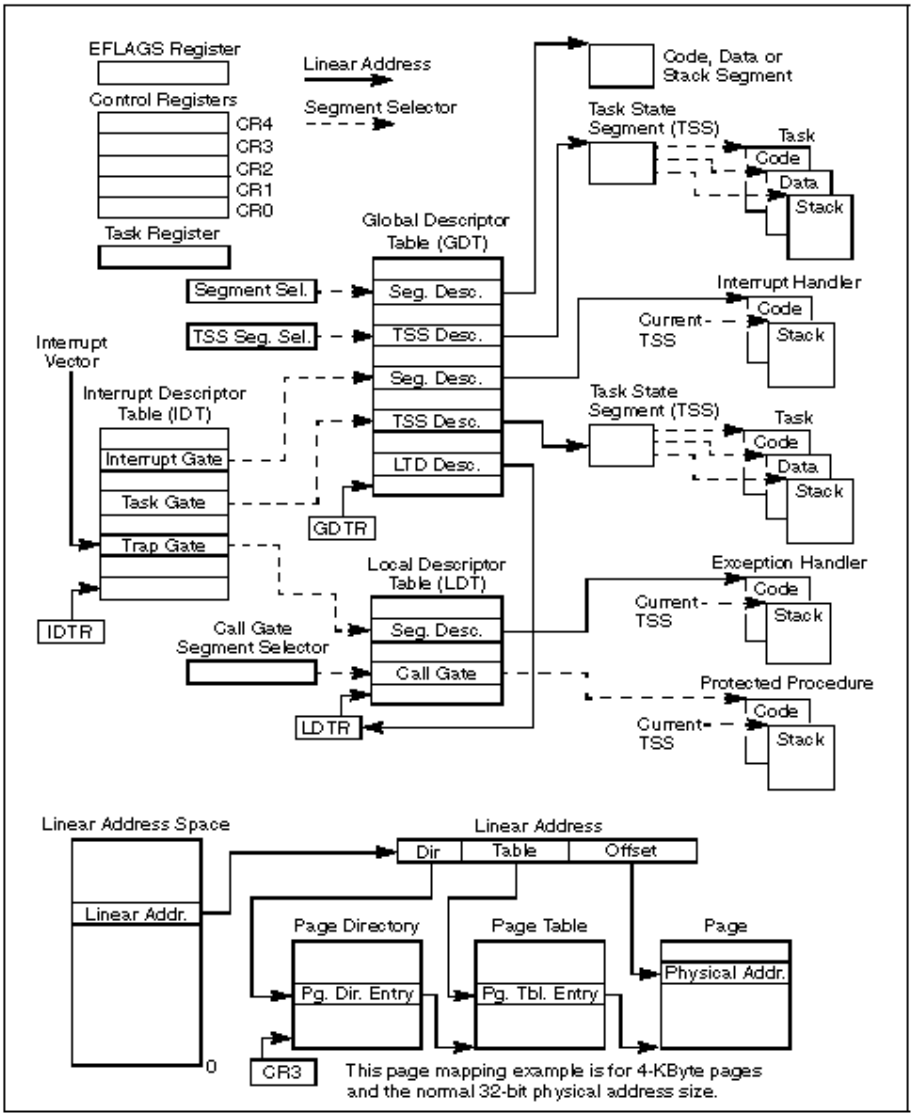
ماذا عن جدول الواصفات المحلي , LDTR هذا الجدول مشابة لجدول الواصفات الشامل GDTR

والفرق أن الأول محلي والثاني عام ، وتستطيع أن تستخدم الجدول العام بدل المحلي

ويستخدم المحلي عادة لمعالجة الإستثناءات والأخطاء وبعض الإجرائات ، وبعد قليل سنرى

في بعض الأمثلة على نظام وندوز أنة لا يستخدم جدول الواصفات المحلي ؟!!

المهم في النهاية سيصبح التخطيط لهذة الجداول بهذا الشكل:



وبعد الدراسة النظرية بقي التطبيق العملي على الثلاث جداول ،، لتثبيت المعلومات

أعتقد أنة لايوجد برنامج يستطيع الوصول إلى هذة المناطق في النظام عدى البرنامج الإحترافي - السوفت آيس-

نبدأ : شغل السوفت آيس عن طريق المفتاح Ctrl+D لتظهر لك نافذة الأوامر,

الأمر الأول GDT وهو لعرض جدول الواصفات الشامل ومحتواة في نظام التشغيل الذي تستخدمة

لاحظ العمليات وكيف ترتيبها ، ستلاحظ نفس المعلومات التي ذكرناها وهي

الحد - الأساس - حلقة البرنامج أو إمتيازاتة - وخصائصة

ولآن سنحاول الوصول لكود برنامج معرف في جدول الواصفات الشامل ، نبدأ

شغل اي برنامج ديبغر تستطيع من خلالة عرض قيم مسجلات الأقسام للبرنامج ، أنا سأستخدمolly

بعد تشغيل olly تجول في جهازك وإختر أي ملف تنفيذي ، وبعد فك التجميع لاحظ قيم المسجلات

وأهم شيء مسجل قسم الكود , CS وليكن مثلا يساوي , CS = 1B والآن شغل السوفت آيس

ثم أكتب في سطر الأوامر GDT وإضغط إنتر ، بعد ذلك إبحث عن الإزاحة , 1B x لاحظ المعلومات التي

بجانبة ، أولاً: القسم هو قسم كود ٣٢ بت بهذا الشكل code32 مستوى الإمتياز أو الحلقة يساوي ٣

أي أنة برنامج في مستوى المستخدم ، أما عن العنونة ، سنفهمها في القسم الثاني من الدرس لأنها تتطلب فهم

لمجال العنونة وترقيم الصفحات والجداول,,,

والآن جدول الواصفات المحلي ، أنا ذكرت أن الوندوز لايستخدمة لأني أستند إلى هذة المعلومات

شغل السوفت آيس وفي سطر الأوامر أكتب , LDT لتظهر لك رسالة NO LDT

والجزء الأخير البوابات والمقاطعات ، جدول المقاطعات

من خلال الدرس قد تستغرب ماهي البوابة ؟ !سأدعك تجاوب بنفسك

شغل السوفت آيس وأكتب الأمر idt وسترى البوابة ! وهي عبارة عن إجرائات معرفة في الذاكرة

ويعرض لك السوفت آيس إسم التعريف وملف النظام الذي يحتويها ,,,

وفي مثالنا على وندوز إكس بي ، هذة بعض ملفات النظام التي تحتوي على البوابات:

HAL.dll , ntoskrnl.exe ,dbgmsg.sys,..

بقي مسجل الحالة , TSS بالتأكيد لعرض معلوماتة تستخدم الأمر TSS

وبهذا نكون قد أخذنا مقدمة في برمجة الأنظمة٣٢ بت ، وبقيت بعض المسجلات للتحكم و بعض الجداول

الخاصة بالذاكرة وصفحات الذاكرة ، وبقي أهم جزء وهي الأساليب الجديدة في برمجة أنظمة ٣٢ بت

بإذن الله تكون هذة المواضيع الجزء الثاني لبرمجة أنظمة ٣٢ بت .

وبالتوفيق,,,,
