مُختارات من مدونتي

مدونة MaaSTaaR

منذ 2005 إلى 2009

تأليف : محمد قاسم حسين

جدول المحتويات

تجدون المقالات الا صليه هنا : http://www.maastaar.com

وقفه مع تقنية RSS

بتاريخ : 23/5/2005

ما هي RSS ؟

لنفرض إنّك إستيقظت باكراً و قبل الذهاب إلى العمل و مع قهوة الصباح أردت الاطلاع على مواقعك او منتدياتك المفضله، تخيل انك تريد الاطلاع على 10 منتديات كم ستستغرق من وقت ؟ ربع ساعه او نصف ساعه مثلاً ؟ تقنية RSS تساعدك على توفير وقتك، مثلاً وفرت لك هذه المنتديات امكانية قراءة آخر المواضيع من خلال تقنية RSS، كل ما عليك هو أنْ تُحمل أحد البرامج التي تترجم شيفرات RSS لكي تساعدك على القراءه، من خلال برنامج واحد و بضغطة زر واحده سوف تطلّع على آخر الموضوعات في منتدياتك المفضله من دون الحاجه لزيارة تلك المنتديات.

نظره على نظام التشغيل Agnix

بتاريخ : 22/6/2005

الليله الماضيه و أنا اتصفح ويكيبيديا الانجليزيه و بالتحديد في تصنيف انظمة التشغيل وجدت العديد من انظمة التشغيل، سواء أكانت Unix-Like او غيرها.

وجدت نظام تشغيل بإسم Agnix، توجهت إلى موقع النظام الرسمي و تعرفت على هذه المعلومات :
"Agnix نظام تشغيل مفتوح المصدر تحت رخصة GNU GPL، يهدف هذا النظام إلى ان يكون تعليمي (*) و يكون نظام تشغيل يتعامل تعاملاً جيداً مع الشبكات، كيرنل هذا النظام موسّع و مكتوب بلغة البرمجه C، نظام التشغيل Agnix سريع و صغير"

* المقصود بكلمة "تعليمي" هنا هو انّ الكيرنل يمكن الاستفاده منه لتعليم طرق كتابة نظم التشغيل.

لابد أنْ نلاحظ أنّ Agnix ليس نظام تشغيل و إنما Kernel، و على فكره المشروع حديث و بدأ في اغسطس 2004.

عموماً اترككم مع هذا المشروع الموقع الرسمي

نظام الملفات

بتاريخ : 13/7/2005

يعتبر نظام الملفات من أهم أجزاء نظام التشغيل لأنّه النظام الذي يقوم بتخزين محتوى الملفات و المعلومات الخا صه بها مثل ا سمها و حجمها و غيره، نظام وندوز الافتراضي هو FAT لا بد و انه قد مر عليكم، يقوم هذا النظام و الذي يحتوي على عدة بنيات للملفات بتخزين الملفات و ا سمائها و محتواها، و يقال أنّه هناك تكنلوجيا جديده تستخدم قواعد البيانات في نظام الملفات و ذلك في نظام WinFS في Windows Longhorn، و قد تم بالفعل تطبيق هذه الفكره في Linux مع نظام GNOME Storage و كذلك نظام BFS المحلق مع نظام التشغيل BeOS.

الآن يأتي حديثنا عن نظام ملفات Linux، في با دئ الامر كان نظام التشغيل Linux يستخدم نظام ملفات Minix حيث كان من السهوله نقل ملفات Minix إلى Linux و العكس صحيح، و في فترة إنتشار النظام شَعَرَ المستخدمون بأنّهم بحاجه إلى نظام ملفات أكثر تطوراً ، حينها تم كتابة نظام الملفات Ext و تلاه نظام Ext2 الذي قام بكتابته Rémy Card و الذي كان مبنياً على نظام Ext و بعدها ظهر نظام Ext3 و الذي يعتبر حالياً النظام الافتراضي للتوزيعات المشهوره مثل RedHat و Fedora و Debain و غيرها.

لا أ دري إ ذا كان نظام الملفات NTFS و الذي يعتبر ا سمه إختصاراً للجمله New Technology File System قد مر عليكم من قبل و هو نظام الملفات الذي يستخدم في Microsoft Windows NT و كذلك 2000 و XP (ليس النظام الافتراضي بالنسبه لاكس بي)، هناك مشروع لدعم NTFS في نظام Linux من هنا.

كذلك هناك مشاريع اعجبتني في مجال انظمة الملفات و اولها هو المشروع الذي ذكرته و هو دعم NTFS في Linux و الموجو د هنا، و كذلك مشروع OpenBFS و الذي يهدف إلى إنشاء نظام ملفات شبيه بنظام ملفات نظام التشغيل BeOS و هناك كتاب جيد يتكلم عن نظام ملفات BeOS من هنا (بصيغة PDF) .

تجدون ما يسركم من المعلومات هنا ا ستخدموا محرك البحث الموجو د على اليسار.

ما ذا بعد؟ انتهى حديثي، حان دوركم للبحث و التنقيب ;-)

يوم مع نظام Minix

بتاريخ : 25/8/2005

اولاً يجب ان اعرفكم على ضيفنا اليوم Minix، تحدثت عن هذا النظام في عدّة مقالات سابقه، Minix عباره عن نظام تشغيل كتبه Andrew S. Tanenbaum هذا النظام يتعبر احد انظمة Unix-Like و اسمه يتعبر اختصاراً (Mini Unix) اي يونكس الصغير .

البرفسور Andrew S. Tanenbaum هدفه من كتابة هذا النظام هو أنْ يبني نظام Unix-Like بسيط حتى يفهمه طلابه و يفي بالغرض كنظام Unix-Like بسيط .

من الجدير بالذكر أنّ Linus Torvalds (مبرمج نظام Linux) كان يستخدم نظام Minix عندما كتب لينكس.

عندما سُئل Andrew S. Tanenbaum عن رأيه في نظام Linux قال إنّها فرصة لشكر Linus Torvalds على كتابته لنظام Linux، لأنّ قبل نظام Linux كان نظام Minix (كنظام Unix-Like) هو السائد، و كانت تصل Andrew S. Tanenbaum الكثير من الرسائل التي تطلب اضافة ميزات جديده إلى النظام و كان Andrew S. Tanenbaum غالباً ما يرفضها لأنه يريد المحافظه على Minix كنظام بسيط حتى يفهمه طلابه خلال فصل دراسي واحد.

في هذا الشهر (17\8\2004) في مجموعة Usenet الخاصه بـ Minix كتب Andrew S. Tanenbaum عن نيته لإنشاء موقع جديد لـ Minix افضل من الحالي و ينوي اطلاقه في شهر اكتوبر، بالإضافه إلى تفكيره في انشاء واجهة مستخدم رسوميه (GUI) خاصه بظام Minix يمكننا تسميتها بمنفذ لـ X11 و Andrew S. Tanenbaum كعادته يفكر بالبساطه و الحجم الصغير.

انتهى قبل دقائق تحميل الشيفره المصدريه لنظام Minix لدي و سوف اطلع عليها و قد اكتب انطباعي عنها.

يوجد نقطه هامه جداً و هي كتاب Operating Systems: Design and Implementation الذي قام بتأليفه Andrew S. Tanenbaum, Albert S. Woodhull و الذي يتحدث عن مفاهيم نظم التشغيل بالإضافه إلى شرحه لنظام Minix.

و اخيراً موقع Minix الرسمي هنا

هذا كل ما لدي :-)

كلمات من ذهب لكيفن ميتنك

بتاريخ : 31/8/2005

اولاً لا بد ان تتعرفوا اولاً من هو كيفن ميتنك (Kevin Mitnick) هنا للقصه كامله .

كيفن ميتنك ألّف كتاباً اسمه "فن الخداع"، يتحدث به بالطبع عن تخصصه، في مقدمة هذا الكتاب كتب فقره أعجبتني بالفعل و يمكنني القول أنّ هذه الفقره "كلام من ذهب" :

"Some hackers destroy people's files or entire hard drives; they're called crackers or vandals. Some novice hackers don't bother learning the technology, but simply download hacker tools to break into computer systems; they're called script kiddies. More experienced hackers with programming skills develop hacker programs and post them to the Web and to bulletin board systems. And then there are individuals who have no interest in the technology, but use the computer merely as a tool to aid them in stealing money, goods, or services."

الترجمه :

"هناك هاكرز يدمرون ملفات الناس و كامل اقراصهم الصلبه , هؤلاء يطلق عليهم اسم "كراكرز" او "المخربون" , بعض الهاكرز المبتدئين لا يهتمون في تعلم التكنلوجيا، لكنهم ببساطه يقومون بتحميل البرامج الجاهزه لهذه الوظيفه، هؤلاء يطلق عليهم اسم "Script kiddies" (اطفال الكتابات) , بعض الهاكرز الذين يجيدون البرمجه و يستخدمونها في برمجة البرامج الجاهزه و نشرها على شبكة الانترنت حيث هؤلاء الاشخاص لا يهتمون بالتكنلوجيا , و لكن يستخدمون الحاسوب كـ اداة تعينهم على السرقه"

كلمات من ذهب بكل صراحه , بكل اسف هذه الظاهره انتشرت و رأي البعض ان سبب ذلك و سائل الاعلام، لا اقصد بكلمة "الظاهره" هو عمليات الاختراق، بل إطلاق كلمة "هاكرز" على غير مستحقيها و هم المخربون , أتضايق كثيراً عندما أرى أنّ أحد المنتديات قد أُخترقت أو سُرقت قواعد بيانات هذا المنتدى و يقولون ببساطه أن وراء هذا العمل "هاكرز" , كما و أن للاسف كل من يحمّل البرامج (والتي يطلق عليها برامج هاكرز) يظن أنه قد أصبح هاكرز، و ما أن يتمكن من استباحة (كما يسمونه اختراق) أحد الاجهزه حتى يظن أنّه أصبح أقوى هاكرز **!** (بالاصح كراكرز)

لذا أتمنى أنْ تكفّوا عن إطلاق إسم هاكرز على المخربين، و لكي تتعرفون من هو الهاكرز الحقيقي يمكنكم مراجعة مقالة كيف تصبح هاكرز (مترجمه إلى اللغه العربيه) .

محركات المُتصفحات

بتاريخ : 18/9/2005

اليوم لدي موضوع مختلف قليلاً، سأتكلم عن محركات المتصفحات او Layout engine (الترجمه الحرفيه محركات التخطيط)، حسناً ما هي Layout engine و ما ذا يعني هذا المصطلح ؟ في جميع المتصفحات على اختلاف انواعها، سواء كان المتصفح FireFox او IE او Opera، يحتوي هذا المتصفح على جزء ا سا سي يسمّى المحرّك (Layout engine)، هذا المحرك عباره عن برنامج يُكتب بأي لغة برمجه، و يقوم بأخذ لغات الويب كـ XHTML , CSS , HTML و غيرها و يقوم بعرضها، نستنتج من هذا التعريف أن المحرك احد اهم الاجزاء في المتصفح حيث يقوم هذا البرنامج بأخذ محتوى الصفحه و عرضها كما يجب.

هناك عدّة انواع من المحركات لنلقي نظره سريعه عليها :

Gecko :

هذا المحرك عباره عن محرك مفتوح المصدر وهو نفسه المستخدم في متصفح Mozilla FireFox، حيث قامت بإنشاءه شركة Netscape و تم تطويره بوا سطة منظمة Mozilla. المحرك كما ذكرت مفتوح المصدر و مكتوب بلغة السي++ و يدعم المعايير المفتوحه في الانترنت كـ , HTML , XHTML , XML , CSS JavaScript و غيرها كما أنّ هذا المحرّك غني بالـ API، لم اجد الشيفره المصدريه الخا صه بهذا المحرك حتى الآن و لكنني واثق انه شيفرته المصدريه ملحقه مع الشيفره المصدريه الخا صه بالـ FireFox.

Trident :

و هو المستخدم في متصفح مايكرو سوفت IE، اول ا صداره منه كانت في عام 1997 مع الا صداره الرابعه من IE , ليس لدي الكثير عن هذا المحرك للا سف.

KHTML :

هذا المحرك مطوّر مع مجموعة مشاريع KDE، بدأ هذا المحرك في عام 2000 حتى يُستخدم في مدير الملفات و المتصفح Konqueror، هذا المحرك مكتوب بلغة السي++ وهو تحت الرخصه LGPL كما انه يدعم HTML JavaScript , DOM , CSS , . في قام 2002 قامت شركة Apple بتطوير KHTML من اجل متصفحها Safari و قامت بنشره كذلك كمحرك مفتوح المصدر اسمته WebCore .

روابط :

Gecko في منظمة موزيلا

ويكي Gecko

صفحة KHTML

ميزات KHTML

WebCore في موقع ابل

Linux غير مبني على Minix !

بتاريخ : 25/ 10/2005

في الفتره الاخيره لاحظت هذا الاعتقا د الخاطئ من الكثير و هو أنّ نظام التشغيل Linux مبني على نظام التشغيل Minix !

بالطبع هذا الاعتقا د اعتقا د خاطئ و مع ذلك رأيت ذكره في اكثر من منتدى أنّ Linux لم يبدأ من الصفر و انه بدأ من Minix .

الصحيح هو أنّ Linus Torvalds (مبرمج لينكس) كان يستخدم Minix عندما كان يبرمج Linux و في اول ا صدار من Linux كان نظام الملفات الذي يعتمد عليه هو نفسه نظام Minix و لكن هذا لا يعني ان Linux مبني على Minix.

اتمنى أنْ تُصحح هذه المعلومه و التي يخطئ بها الكثير.

قبل التفكير في برمجة نظام تشغيل , ماذا يجب ان نفعل ؟

بتاريخ : 27/ 10/2005

منذ مده ليست بقصيره و نحن نلاحظ العديد من الجهود لبرمجة نظام تشغيل عربي متكامل، مروراً بمشروع الخوارزمي إلى الكثير من المواضيع التي كُتبت محاولة منها تجميع فريق عمل حتى يتم برمجة نظام الاحلام !

من وجهة نظري أنّه يجب علينا قبل التفكير ببرمجة نظام تشغيل يجب علينا معرفة الكثير من الامور التي تخص الحاسوب ، و ساحاول تلخيصها في هذه المقاله بإذن الله .

المعرفه في الكيان المادّي Hardware

حسناً ، لا بد من الفهم الجيد للقطع الاساسيه للحاسوب، مثلاً المعالج (CPU)، ذاكرة الوصول العشوائي (RAM) و غيرها من القطع الاساسيه، لأننا سوف نتعامل معها عند برمجة نظام التشغيل، مثلاً عند برمجة "ادارة الذاكره" في النظام لا بد من ان يكون لدينا فهم لذاكرة الوصول العشوائي.

إذاً لا بد من الفهم الجيد للـ hardware حتى نتمكن من التعامل معها .

المعرفه في لغات البرمجه

بالطبع حتى نتمكن من كتابة اي برنامج فإننا بحاجه إلى لغة برمجه نكتب بها هذا البرنامج، و على الارجح لغة C هي المفضله من اجل كتابة نظم التشغيل كذلك لا بد من لغة التجميع Assembly.

قد يعتقد البعض أنّ المعرفه بلغة الـ C يعني التعامل مع احد البيئات التي توفرها الشركات كـ Visual C او C++ Builder , او حتى استخدام المكتبات الرسوميه مثل GTK , بالطبع هذا الاعتقاد غير صحيح !

في الحقيقه اذا كنت محترفاً في العمل على بيئة Visual C فذلك لن ينفعني كثيراً ، لن تنفعني إلا معرفتي في اساسيات الـ C , لانني عندما ابرمج نظام تشغيل لن اتمكن من استخدام اي مكتبة جاهزه مقدمه من الـ Visual C مثلاً .

اما بالنسبه للغة التجميع فهي مهمه جداً ، ولا بد من اتقانها بشكل جيد حتى نتمكن من كتابة نظام التشغيل، مثلاً الـ Boot loader لا يمكننا كتابته إلا بإستخدام لغة التجميع، كذلك لغة التجميع تقدم لنا خدمات BIOS التي قد تساعدنا في كتابة بعض مكتباتنا الخاصه بـ نظام التشغيل.

دراسة مفاهيم نظم التشغيل

هذه الخطوه تعتبر خطوه هامه و قد تكون اطول مده تقضيها في اثناء مرحلة العمل ما قبل كتابة نظام التشغيل، مفاهيم نظم التشغيل مثل معرفة ما هي عملية الـ Boot (الاقلاع) و معرفة كيف تعمل و دراستها بشكل جيد و اخيراً قراءة بعض الـ Boot loader المفتوحة المصدر و البسيطه حتى تتعرف كيف يتم برمجة الـ Boot loader، و اتباع هذا الاسلوب مع باقي المفاهيم مثل ادارة الذاكره , انظمة الملفات و غيرها الكثير.

هذه بعض التلميحات الموجهه للمشاريع التي تنوي بناء نظم التشغيل، و اعتذر اذا كان هناك اي خطأ في الحقيقه خبرتي في هذه الامور ليست قويه و لكنها تسمح لي بكتابة مثل هذه المواضيع.

نظره سريعه على IceWM

بتاريخ : 31/ 10/2005

IceWM هو احد مدراء النوافذ الخا صه بانظمة Unix (مثل Linux)، و لكن يعتبر IceWM مدير نوافذ خفيف على الذاكره ولا يستهلك الكثير من الموارد، اي أنّه جيد للحوا سيب القديمه و التي ذاكرتها منخفضه و لا تتحمل الا صدارات الجديده من KDE او GNOME، ليس ذلك فحسب IceWM تحاكي واجهات الوندوز بشكل كبير و يوجد العديد من الانماط (theme) التي تجعل واجهة IceWM مشابهه تماماً لوندوز سواء كان وندوز 98 او XP او حتى 3.1 !

في الحقيقه اعجبني IceWM كثيراً نظراً لحجمه الصغير و عدم ا ستهلاكه للموارد و تتوفر له ثيمات جميله جداً ، ما جعلني اتعرف عليه هو تجربتي لأحد التوزيعات من لينكس و التي سوف اكتب عنها قريباً ان شاء الله.

حان وقت الروابط الآن.

الموقع الر سمي
ثيمات للـ IceWM

اعد المتعه للحو سبه , ا ستخدم لينكس ;-)

توزيعة Slackware

بتاريخ : 1/ 11/2005

مقدمه

Slackware هي احد توزيعات لينكس انشئها Patrick Volkerding، توزيعة Slacware اخذت طريقه مختلفه عن بقية التوزيعات المشهوره مثل ريدهات و ماندريفا او غيرها في انها تهدف لتكون توزيعة لينكس مشابهه لنظام Unix، سيا ستها أنّها تأتي فقط مع الا صدارات المستقره من البرامج، البعض يقول "اذا عرفت Slackware , هذا يعني انك عرفت لينكس ... و اذا عرفت RedHat , هذا يعني انك لم تعرف إلا Redhat"

تاريخ Slacware

اول نسخه تم ا صدارها من Slackware هي 1.00 في 16 يوليو من عام 1993 بوا سطة Patrick Volkerding و الذي يعتبر مؤ سس التوزيعه و مدير المشروع، كانت مبنيه على توزيعة Softlanding Linux System و مجهزه على قرص مرن، و كذلك كانت متوفره لأي شخص في حساب الـ FTP، توزيعة Slackware هي اقدم توزيعه باقيه إلى الآن .

في الا صدارات الاولى من التوزيعه كانت تأتي بثلاث حسابات لمستخدمي النظام بغرض عرض مثال للمستخدم، و لكن تم ازالة هذه الحسابات فيما بعد لأمور امنيه.

في عام 1999 تم زيا دة رقم الا صدار من 4 إلى 7 و السبب كما شرحه Patrick Volkerding هو لتسويق الجهو د و توضيح أنّ توزيعة Slackware توزيعه محدّثه .

في عام 2004 مرض Patrick Volkerding مرضاً شديداً مما جعل مستخدمي و مطوري توزيعة Slacware يقلقون بشأن مستقبل التوزيعه , بعدها تشافى و اكمل تطوير التوزيعه .

في عام 2005 تم ازالة بيئة سطح المكتب GNOME من التوزيعه .

و الجدير بالذكر انه هناك العديد من التوزيعات المبنيه على Slackware مثل SuSE , College Linux , miniSlack , Slax .

فلسفة تصميم Slacware

Keep it Simple, Stupid :

و تعني هذه الجمله باللغه العربيه "اجعلها بسيطه و غبيه !" , هذا المفهوم يشرح الكثير عن تصميم Slackware , حيث كلمة "بسيطه" في هذا السياق تؤدي إلى بساطه تصميم النظام , حيث يكون سهل للاستخدام , و هذا السبب وراء قلّة ادوات الاعداد التي تعتمد على واجهة المستخدم الرسوميه (GUI) , لأن واجهات المستخدم الرسوميه مركبّه لذا فأن مشاكلها تعتبر اكثر مقارنة بالواجهات البسيطه و التي تعتمد على سطر الاوامر , و نتيجه لهذا المبدأ توزيعة Slackware سريعه , مستقره و آمنه . النقّاد يقولون إنّ هذا المبدأ يجعلها توزيعه صعبه و مضيعه للوقت، مؤيدون هذه التوزيعه يقولون انها مرنه.

برامج البدايه :

Slackware تستخدم في اعدادها لبرامج البدايه طريقة BSD، بينما تستخدم معظم توزيعات لينكس طريقة نظام (System V) لإعداد برامج البدايه، مؤيدون طريقة BSD يقولون انها افضل لانه مع هذا النظام يصبح من السهل البحث او القراءه او التعديل على البرامج، و مؤيدون طريقة نظام V يقولون انها افضل لانها تجعل من البرامج اكثر قوه و مرونه.

مدير الحزم :

تتميز توزيعة Slackware بوجود مدير سهل للحزم مقارنة بالتوزيعات الاخرى، حزم Slackware دائماً تكون ذات الامتداد .tgz او .tar.gz حيث يتم فكها و تركيبها في اماكنها المناسبه بواسطة مدير الحزم .

هذه المقاله قمت بترجمتها و هذا هو مصدرها

الركائز الاساسيه لبناء برامج الـ PHP

بتاريخ : 25/2/2006

لدي موضوعاً هاماً جداً اليوم لمبرمجين لغة PHP .

حسناً ، لبناء مشروع ضخم في البي اتش بي سواء أكان هذا المشروع تجارياً ام مجانياً لا بد من مراعاة امور هامه جداً و الخصها كالتالي :

ترتيب الكود :

و هذا الامر هام ليس فقط في كتابة برامج البي اتش بي، بل في جميع اللغات لا بد من الاهتمام بترتيب الكود، الكود المرتب بالتأكيد سوف يسهّل من القراءه كثيراً كذلك يسهّل في تطويره .

ملف يشمل كل العمليات المتكرره :

اذا كان المشروع الذي تريد تطبيقه بالفعل يعتبر ضخم لا بد و إنّك سوف تحتاج لتكرير عمليات معينه في كل ملف برمجي، من الافضل تجميع كل هذه العمليات في ملف واحد و تضمين هذا الملف إلى كل ملفات البرنامج.

الامان ثم الامان ثم الامان :

مشروع ضخم او غير ضخم لا بد من الاهتمام بأمن البرنامج، مثلاً الاهتمام بالامور التالي :

- علامات الاقتباس السحريه : و هنا موضوع في قريتي الحبيبه عن هذه العلامات.
- استخدام الكوكيز بحذر : و هنا موضوع كذلك في قريتي الحبيبه.
- استخدام المصفوفات $_COOKIE$, $_POST$, GET$، بدلاً من استدعاء المتغيرات كمتغيرات عاديه مُعرفه من خلال المبرمج .

فصل الـ HTML عن الـ PHP

اي ا ستخدام نظام قوالب مهما كان نوعه، سواء يخزّن القوالب في الملفات او في قواعد البيانات، مثلما القوالب مفيده للمستخدم العا دي حيث تسهل عليه من عملية التحكم في شكل السكربت كذلك هي مفيده بالنسبه للمبرمج، بصرا حه نفسياً عندما اقرأ برنامج لا يفصل بين الـ HTML و الـ PHP لا اكمل قراءته او تكون قراءته بالنسبه لي غير ممتعه نهائياً , القوالب تسهل الكثير على المبرمجين.

عملية المبادلة Swapping

بتاريخ : 3/3/2006

مصطلح Swap لا بد و أنّه قد مر ولو مرور الكرام على من جرّب أو قام بتثبيت لينكس، الـ Swap تتطلب منك انشاء بارتشن خالي ليس له نظام ملفات و حجمه يساوي حجم الـ RAM مضروب بإثنين.

حسناً لما ذا هذا البارتشن و ما هي عملية المبا دله ا صلاً ؟ كما نعلم جميعاً أنّ ذاكرة الـ RAM تخزن المعلومات التي يقوم باخذها المعالج من اجل معالجتها، و بالطبع ا داء الحا سوب يعتمد بشكل رئيسي على سعة الـ RAM و سرعة المعالج بالتأكيد و لكن محور حديثنا الآن هو الذاكره.

بهدف تحسين الا داء اتبعت نظم التشغيل طريقه في ا دارة الذاكره تسمى هذه الطريقه بـ Virtual Memory بحيث يكون هناك ذاكره مساعده لذاكرة الـ RAM اكبر حجماً و هو القرص الصلب، أي البارتشن الخاص بعملية الـ Swap هو مساحه من القرص الصلب مخصصه لتخزين المعلومات التي تعتبر اقل ا ستخداماً في الـ RAM، و النتيجه بالطبع هي بقاء المعلومات الاكثر اهميه في الـ RAM مما يحسّن الا داء.

و صلات :

http://www.tldp.org/HOWTO/Swap-Space.html

http://computer.howstuffworks.com/question684.htm

قصة لغة السي و ماضيها مع يونكس

بتاريخ : 16/3/2006

لغة السي، و من لا يعرفها ؟ لغه السي هي أحد اللغات عاليه المستوى و المستخدمه بشكل كبير جداً في بناء التطبيقات، ولا بد أن اي شخص له اهتمام بتقنيات البرمجه قد سمع عن لغة السي او السي++ التي تعتبر خليفتها، لغة السي لها علاقه قويه مع نظام التشغيل Unix، حتى نتعرف على ماضي السي لا بد ان نتعرف اولاً على ماضي يونكس.

من كتاب نظره على نظم التشغيل المختلفه عن ماضي يونكس :
"بدأ نظام Unix في عام 1969 حين بدأ Ken Thompson, Dennis Ritchie و آخرون بالعمل في مختبرات بيل , قصة هذا النظام من اروع القصص التي مرّت علي.

نظام Unix صُمم لكي يكون نظام قابل للنقل من جهاز إلى آخر، متعد د المهام و متعد د المستخدمين و تميز بالعديد من المفاهيم.

في البدايه تم كتابة هذا النظام بلغة التجميع (Assembly) بعدها قام Dennis Ritchie بكتابة لغة برمجه جديده بإ سم سي (C) و اعا د كتابة نظام Unix بهذه اللغه الجديده في عام 1973 ، اعا دة كتابته بهذه اللغه جعله نظاماً قابلاً للنقل من جهاز إلى آخر حيث غيّر من تاريخ انظمة التشغيل ."

حسناً ، جميعنا نعرف لغة التجميع (Assembly) التي لا زالت تُستخدم حتى اليوم في وظائف معينه تتعلق بالعتا د و التي ظهرت في الجيل الثاني من الحوا سيب، في الحقيقه لغات التجميع كثيره ! سوف اوضح لكم كيف ذلك، لكل معالج لغة تجميع خا صه به , مثلاً معالجات Pentium لها لغة تجميعه الخا صه التي تختلف عن لغة تجميع المعالج PowerPC , حسناً هذا يعني أنني ا ذا كتبت أي برنامج بلغة التجميع و كانت لغة التجميع هذه هي الخا صه بـ معالجات Pentium لن اتمكن من تشغيل برنامجي هذا على الحوا سيب التي تعمل على معالج PowerPC و العكس صحيح، و كذلك بالنسبه للمعالجات الاخرى التي تستخدم معماريتها الخا صه، حسناً تخيلوا ان البرنامج الذي تحدثنا عنه هو نظام التشغيل Unix، في با دئ الامر تم كتابة يونكس كاملاً بإ ستخدام لغة التجميع

على الحا سوب PDP-11، هذا يعني انه لن نتمكن من ا ستخدام يونكس إلا على هذا الحا سوب و على العتا د الموجو ده فيه , و هذا يحد من نشر النظام على العامّه و يقيدهم بنوع معين من الحوا سيب من اجل تشغيل النظام .

الآن لنأتي للغة السي، عندما اقوم بكتابة برنامج بلغة السي على معالج Pentium سوف يكون نفس هذا البرنامج لمعالجات PowerPC و سوف يعمل برنامجي بدون اي مشاكل على المعالجات الاخرى قد احتاج إلى ترجمة البرنامج على كل معالج و لكن لن يتغير في الكو د شئ , هذا ما تم بالفعل في نظام يونكس، حيث قرروا كتابته بلغه عالية المستوى حتى يتمكنوا من تشغيل يونكس على مجموعه كبيره من الحوا سيب، في ذلك الوقت لم يكن هناك لغه ا سمها C بل كانت اللغه المستخدمه آنذاك لغه ا سمها B، هذه اللغه لم تكن بالقوه الكافيه التي تجعلها قا دره على كتابة نظام تشغيل لذا قرر Dennis Ritchie أنْ يطور هذه اللغه و اضاف عليها اضافات جديده حتى تصبح بالقوّه المطلوبه، بعد ذلك ظهرت لغة C و تم كتابة نظام يونكس كاملاً بها و أ صبح قابلاً للنقل إلى الاجهزه الاخرى.

هناك نقطه لابد من توضيحها، متابعون نظام التشغيل Linux سيلاحظون توفر العديد من النسخ من كيرنل لينكس الخا صه بالمعالجات الاخرى، اي لكل معالج إ صداره من النواه خا صه بها، قد يتسائل البعض لما ذا و لينكس مكتوب بـ لغة السي، في الحقيقه نعم لينكس مكتوب بلغة السي و لكن هناك اجزاء صغيره من كل نظام تشغيل لا بد من كتابتها بلغة التجميع بالإضافه إلى ذلك يجب ان لا ننسى انّ معمارية كل معالج تختلف عن معمارية المعالج الآخر.

عملية الاقلاع , ما هي ؟ وكيف تتم ؟

بتاريخ : 2/4/2006

عملية الاقلاع ببساطه هي العمليه التي تقوم بتحميل نظام التشغيل و تعطيه التحكم حتى تتمكن من ا ستخدامه، عندما تقوم بتشغيل الحا سوب يقوم نظام يسمّى BIOS مُخزن في ذاكرة القراءه فقط ROM بفحص كل و سائط التخزين القرص المرن ثم السي دي ثم الهار د سك، و عندما يجد برنامج مُحمّل الاقلاع (Boot loader) في احد و سائط التخزين يبدأ في عمله.

يقوم نظام BISO با خذ محمّل الاقلاع ووضعه في الذاكره و تحديداً في العنوان 07C00h و يقوم بتنفيذه، و قبلها يقوم بوضع قيمة في المسجل dl الموجو د في المعالج محتوى هذه القيمه هي اين وجد نظام BIOS محمّل الاقلاع ، في القرص المرن ؟ ام في القرص الصلب ؟ ام في القرص المدمج ؟ و لكل واحده منهم رقم خاص به.

محمّل الاقلاع هذا البرنامج وظيفته تحميل نظام التشغيل و يجب أنْ لا يزيد حجم هذا البرنامج المكتوب بلغة التجميع عن 512 بايت لانه هو حجم القطاع (sector) و يجب أنْ يكون محمّل الاقلاع دائماً في اول قطاع سواء في القرص الصلب او المرن او المدمج.

هذا بإختصار ما يتم عند عملية الاقلاع.

ملخصات لينكس - الجزء الاول

بتاريخ : 2/4/2006

اقوم حالياً بقراءة هذه الصفحات عن نظام التشغيل غنو\لينكس، بعد انتقالي إلى غنو\لينكس اريد أنْ اطوّر نفسي في هذا النظام و بدأت اليوم و في كل مره اقرأ جزء من الو صله المذكوره في الاعلى سوف اقوم بتلخيص ما قرأته و نشره حتى يستفيد الجميع

الحسابات في لينكس

* اولاً حساب الجذر (root) يمتلك كل الصلاحيات و لا يصلح ا ستخدامه من اجل الاغراض الروتينيه اليوميه و من الافضل إنشاء حساب جديد خاص بالاعمال الروتينيه .

* طريقة الإنشاء عن طريق الصَدفه كالتالي :

```
useradd username
```

و لإعطاء المستخدم كلمة المرور

```
passwd username
```

صدفة لينكس و اوامرها (Shell)

* الصدفه هي البرنامج الذي يقوم بالربط بين المستخدم و الحا سوب، و كل نظم التشغيل تحتوي على صدفه من اجل هذا الغرض، من ا شهر الصَدفات المستخدمه في لينكس هي BASH

* الامر cd :

و يستخدم هذا الامر من اجل الانتقال بين المجلدات

* الامر ls :

و يستخدم هذا الامر من اجل عرض قائمة الملفات و المجلدات و يعطيها الوان مميزه من اجل التفريق , يمكن ا ستخدام الخيار l من اجل عرض معلومات تفصيليه عن الملفات و المجلدات

```
ls -l
```

كما يمكن ا ستخدام المفتاح TAB للبحث عن ا سم ملف معين في المجلد , مثلاً

```
ls m [TAB] [TAB]
```

و سوف تظهر جميع الملفات التي تبدأ بالحرف m

التركيب الا سا سي لمجلدات لينكس

* المجلد bin :

و يعتبر هذا المجلد من المجلدات الضروريه في لينكس و الذي يحتوي على الاوامر و التي تُعتبر برامج ا صلاً .

* المجلد etc :

و هذا المجلد يحتوي على ملفات اعدا دات النظام و البرامج الاخرى

* المجلد usr :

و يحتوي هذا المجلد على البرامج التي يستخدمها جميع مستخدمي النظام

* المجلد dev :

في لينكس كل شئ عباره عن ملفات كذلك عتا د الحا سب عباره عن ملفات، مثل كرت الصوت , المو ديم , الشا شه , القرص الصلب , هذا المجلد يحتوي على كل ما يمكن ا ستخدامه من عتا د على الحا سوب

* المجلد boot :

هذا المجلد غالباً ما يحتوي على نواة لينكس المضغوطه .

* المجلد root :

يحتوي هذا المجلد على ملفات المستخدم الجذر

* المجلد sbin :

هذا المجلد شبيه بالمجلد bin حيث يحتوي على الاوامر و لكن هذه الاوامر لا يمكن إستخدامها إلا المستخدم الجذر

* المجلد tmp :

و يحتوي هذا المجلد على الملفات المؤقته و التي سوف تُحذف بعد مده

* المجلد var :

يحتوي هذا المجلد الملفات ذات الاحجام دائمة التغير، مثلاً قواعد البيانات

* المجلد lib :

و يحتوي هذا المجلد على المكتبات التي تستخدمها البرامج من اجل تنفيذ وظائفها

ملخصات لينكس - الجزء الثاني

بتاريخ : 5/4/2006

نكمل ما بدأناه من ملخصات , ابرز المصا در لهذا الجزء :

http://www.linuxforums.org/misc/understanding_/proc.html

http://www.linux.org/lessons/beginner/toc.html

ا ستخدام الانابيب

يمكن ا ستخدام الانبوب | في سطر الاوامر لكتابة اكثر من امر واحد دفعه واحده مثال :

```
command 1 | command 2 | command X
```

الامر history

يمكننا الاطلاع على آخر 400 امر قمنا بتنفيذه من خلال الامر

```
history
```

الملف motd

هذا الملف موجو د في المجلد etc و يحتوي على ر سالة الترحيب التي تظهر عند بدأ النظام، يمكن للمستخدم تغييرها بما ينا سبه

فهم /proc

تعريف :

هذا المجلد احد المجلدات الهامه في نظام لينكس حيث يحتوي معلومات غايه بالاهميه متعلقه بالمعالج، العتا د، كما يمكنك من خلال هذا المجلد معرفة الـ modules المُحمله على النظام و الوقت الذي قمت بتشغيل النظام به، و الذاكره المُستخدمه في نظامك، كما أنّ كُل عمليه في النظام لها ملف او مجلد في proc.

عمل mount للمجلد proc :

الملجد proc يُعتبر نظام ملفات (filesystem) اي يمكن عمل mount و unmount له، العديد من توزيعات لينكس تقوم بعمل mount تلقائياً لهذا المجلد، لمعرفة اذا كان هناك mount لنظام الملفات proc نقوم بكتابة الامر

```
mount
```

و في حالة وجوده سوف نجد سطراً مشابهاً لهذا السطر :

```
proc on /proc type proc (rw)
```

أما في حالة عدم وجوده لا بد من عمل mount له كالتالي :

* نقوم بإنشاء مجلد اسمه proc على سبيل المثال في نظام الملفات \
* نضيف السطر التالي في الملف etc/fstab

```
proc        /proc    proc        defaults        0 0
```

* و اخيراً نقوم بتنفيذ الامر

```
mount proc
```

إظهار معلومات العمليات :

كما ذُكر في (التعريف) إنّ اي عمليه تتم في النظام (العمليات هي البرامج او اي شئ يتم تشغيله على النظام) يكون لها مجلداً خاصاً بها في proc و يحمل هذا المجلد رقم خاص بالعمليه يسمى PID (الكلمه PID اختصاراً لـ process identification number)، مثلاً يمكن أنْ تجد في proc مجلد بإسم 3247 و عندما تدخل إلى المجلد تجد ملفاتاً ذات اسماء مثل exe , fd/ , auxv , mem إلخ , اي انه هناك الكثير من المعلومات المتعلقه في العمليات تُخزن في proc، و الآن نُلقي نظره على الملفات الاساسيه الموجوده في مجلد كل عمليه يتم إنشائها في لينكس :

الملف cmdline :

يحوي هذا الملف على كيفيه تشغيل العمليه من خلال الـ command line، يمكن قرائتها من خلال محرر النصوص او من خلال استخدام الامر strings كالتالي :

```
strings cmdline
```

و مثالاً على الناتج :

```
wvdial
--config
/home/maastaar/.wvdial.conf
```

الملف cwd :

و يحتوي هذا الملف على المجلد الذي بدأ منه العمل، مثلاً عندما نقوم بفتح ملف معين في OpenOffice سوف يُظهر لنا cwd المجلد الذي تم فتح الملف منه.

الملف environ :

و يحتوي هذا الملف على البيئه التي تعمل فيها العمليه، مثلاً من اي صَدَفه تم تشغيلها إلخ ... , و من اجل الاطلاع على المعلومات نستخدم التالي :

```
srings environ | sort | less
```

الملف exe :

يحوي هذا الملف على الكود الثُنائي الفعلي للعمليه، و يمكن عمل نسخه اخرى من العمليه من خلال الامر التالي :

```
/proc/[PID]/exe
```

الملجد fd :

هذا المجلد عباره عن و صله رمزيه من اجل فتح الرموز المستخدمه بالبرنامج .

الملف loginuid :

في النواه كل عمليه تُمثل على شكل مجموعه من التراكيب، هذه التراكيب تحتوي في اجزاءها على العديد من المعلومات عن العمليه، احد هذه الاجزاء في التركيبه هي الصفه loginuid التي تحوي اي مستخدم بدأ بهذه العمليه .

الملف maps :

و هذا الملف يحوي خريطة العمليه في الذاكره و معلومات متعلقه بذلك , مثال على ما يحويه هذا الملف :

```
08048000-080a1000 r-xp 00000000 08:06 35310321   /opt/OpenOffice.org/progra
m/soffice.bin
080a1000-080a6000 rw-p 00058000 08:06 35310321   /opt/OpenOffice.org/progra
m/soffice.bin
080a6000-084b6000 rw-p 080a6000 00:00 0          [heap]
b0e2d000-b0e3d000 rwxp b0e2d000 00:00 0
b0e4d000-b0e81000 r--p 00000000 08:06 86555296   /opt/OpenOffice.org/help/e
n/picture.db
b0e81000-b0ec4000 rw-p b0e81000 00:00 0
b0ec4000-b0f34000 rwxp b0ec4000 00:00 0
b0f42000-b0f62000 rwxp b0f42000 00:00 0
b0f62000-b0f7d000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b0f7d000-b0f98000 r--s 00000000 00:07 21004324   /SYSV00000000 (deleted)
b0f98000-b0fb3000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b0fb3000-b0fce000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b0fce000-b0fe9000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b0fe9000-b1004000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b1004000-b101f000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b101f000-b103a000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b103a000-b1055000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b1055000-b1070000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b1070000-b108b000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
b108b000-b10a6000 r--s 00000000 00:07 20971555   /SYSV00000000 (deleted)
```

الحقل الاول من اليسار هو العنوان الذي تقع به العمليه .

الحقل الثاني هو صلاحيات هذه العمليه في الذاكره حيث يوجد read و write و exectue و shard و respectively .

الحقل الثالث يحتوي على الـ offset

الحقل الرابع يحتوي على المُحرك الذي تم وضع العمليه في الذاكره من خلاله

الحقل الخامس و يحتوي على رقم العُقده الخا صه بالملف الموجو د في الذاكره

الحقل السا دس و هو مكان وجو د الملف

الملف mem :

هذا الملف لا يمكن قراءته مبا شرة بل يجب ا ستخدام ا ستدعاءات النظام و هي open() و read() من اجل قراءته حيث يحتوي على صفحات ذاكرة المعالج .

الملف mounts :

و يحتوي عا دة على الاماكن الذي عمل النظام mount عليها

الملفات oom_* :

و تحتوي هذه الملفات سلوك العمليه عن وجو د حالة خروج عن الذاكره (Out of memory)

الملف seccomp :

ا ذا تم تغيير قيمة هذا الملف إلى 1 سوف يتم تمرير العمليه إلى Seccomp mode

الملف smaps :

هذا الملف يحتوي على الذاكره المُستهلكه من العمليه

الملف stat :

و يحتوي على حالة العمليه

الملف statm :

و يحتوي هذا الملف على معلومات حالة الذاكره

الملف status :

محتوى هذا الملف نفس محتوى stat و statm و لكن يعرضها بطريقه يمكن للبشر قراءتها

المجلد task :

يحتوي هذا المجلد نفس محتوى المجلد الاب له اي يحتوي ما يحتويه ملف العمليه مره اخرى

الملف wchan :

و يحتوي على وظائف النواة المستخدمه بوا سطة العمليه

مدير الاقلاع , نظره سريعه

بتاريخ : 6/4/2006

مرحباً بكم تدوينتي اليوم عن محمّلات الاقلاع .

مقدمه

عندما تقوم بتشغيل حا سوبك الشخصي يقوم نظام BIOS بالبحث عن برنامج يُسمى محمل الاقلاع (Boot loader) و احياناً يسمى بالانجليزيه Boot strap، محمّل الاقلاع عباره عن برنامج صغير يبلغ حجمه 512 بايتاً او حتى اقل، يقوم هذا البرنامج ببعض العمليات ثم يقوم بتحميل نواة نظام التشغيل و اعطائها التحكّم و بالطبع لكل نظام تشغيل محمّل اقلاع خاص به، في الحقيقه هناك برامج اقلاع خا صّه تسمّى بمدير الاقلاع (Boot Manager)، في هذه البرامج يقوم محمل الاقلاع الرئيسي الذي يبلغ حجمه 512 بايتاً بتحميل برنامج خاص بدلاً من نواة نظام التشغيل، وظيفة هذا البرنامج هو وضع قائمة بنظم التشغيل الموجوده في الحا سوب و الاقلاع إليها في حال طلب المستخدم ذلك، احياناً تسمّى مدراء الاقلاع بمحملات الاقلاع كذلك، و لكن افضّل ان نفصل بينهم حتى لا نخلط بالمعنى، لان مدير الاقلاع بحاجه ا صلاً لمحمّل اقلاع .

قد يتسائل البعض لما ذا لا يكون مدير الاقلاع هو نفسه محمّل الاقلاع، في الحقيقه الحجم المسموح لمحمّل الاقلاع هو 512 بايتاً فقط لا اكثر، و برنامج مدير الاقلاع غالباً ما يكون حجمه اكبر من 512 بايتاً ، كما انه يجب الملاحظه ان محمّل الاقلاع يجب أنْ يُكتب بلغة التجميع (الا سمبلي)، بينما مدير الاقلاع يمكننا كتابته بلغة السي و مدير الاقلاع يمكننا تسميته بنظام تشغيل مصغّر هدفه ا دارة نظم التشغيل الموجوده ا صلاً في الحا سوب.

و بالطبع هناك مدراء اقلاع حرّه و مفتوحة المصدر و اهمها :

GRUB

- مقدمه :

احد مشاريع (الذي اعشقه) GNU بالطبع يقع تحت رخصة GNU GPL، له شعبيه كبيره جداً و يستخدم غالباً في نظم التشغيل المشابهه لـ Unix مثل Linux، يسمح بتعد د نظم التشغيل و يحتوي على ميزات ممتازه من اهمها انه يمتلك سطر اوامر من اجل التحكم به، امكانية وضع كلمه سريه عليه، يمكن ان تضع خلف قائمة الاقلاع صوره

تختارها، كما انه يمكنك وضع نظام تشغيل افتراضي يقلع عليه بعد ما تحدده من الثواني

- ماضي GRUB :

بدأ هذا المشروع عام 1995 بواسطة Erich Boleyn حيث كان يحاول الاقلاع إلى نظام Hurd الخاص بـ GNU، وضع Erich و Brian Ford بوضع المواصفات الخاصه بمدير الاقلاع و بالفعل بدأ Erich بالتعديل على محمل الاقلاع الخاص بـ FreeBSD حيث كان محمّل اقلاع سهلاً للفهم و بسيطاً، وجد Erich أنّ كتابة محمّل اقلاع من الصفر امر بسيط، بعدها وضع محمل FreeBSD على جنب و بدأ في GRUB، اضاف بطل قصتنا Erich العديد من المميزات على GRUB و في عام 1999 تم اضافة GRUB كبرنامج رسمي في مشروع GNU و تم فتح المجال من اجل تطويره.

- و صلات :

/http://www.gnu.org/software/grub

http://en.wikipedia.org/wiki/GRUB

- صور :

http://yekubuntu.free.fr/warty/images/grub.png

http://tomos.webmasters.gr.jp/diary/image/grub_image.jpg

http://www.thecybersource.com/screenshots/grub.png

LILO

- مقدمه :

محمل الاقلاع LILO هو محمل اقلاع كُتب خصيصاً من اجل نظام التشغيل لينكس، اسمه هو اختصار للكلمه LInux LOader، قام بكتابته John Coffman و هناك العديد من توزيعات لينكس التي تستخدمه، يُقال ان GNU GRUB افضل منه حيث يحتوي على ميزاته و يتجنب عيوبه، LILO لا يقلع فقط إلى نظام لينكس بل يمكنه الاقلاع إلى العديد من نظم التشغيل مثل OS/2 , Windows , DOS و عائله BSD , لم اجد الكثير من المعلومات عن هذا المحمّل مع الاسف، و لكنه مشهور جداً في مجتمعات لينكس و يونكس.

- و صلات :

http://freshmeat.net/projects/lilo/

http://en.wikipedia.org/wiki/LInux_LOader

- صور :

http://en.wikipedia.org/wiki/Image:Lilo.png

GAG

- مقدمه :

رغم ان هذا المشروع ليس بشهره GRUB او LILO إلا انه يستحق أنْ نلقي عليه نظره، GAG هو اختصار لـ GRAPHICAL BOOT MANAGER و تعني مدير الاقلاع الرسومي، مدير الاقلاع هذا كنت استخدمه قبل استخدامي لـ GRUB و الحقيقه يعتبر مدير اقلاع جيّد حيث إنّه رسومي و يمكنك وضع صوره لاي نظام تضيفه، هذا المدير لا يتعرف على نظم التشغيل مباشره يجب عليك اضافتها إليه عند تركيبه، كما ان قائمته تتحمل 9 نظم تشغيل فقط، يحتوي على مجموعه من الميزات الجيده مثل التوقيت، و حماية الاقلاع بكلمه سريه (لن يتمكن احد من تشغيل حاسوبك بدون اذنك)، كما انه يمكن ترجمته إلى لغات اخرى (هل من مشمّر لترجمته للغه العربيه ؟) , بالطبع البرنامج مفتوح المصدر تحت رخصة GNU GPL.

- و صلات :

http://gag.sourceforge.net/

- صور :

http://gag.sourceforge.net/gagn3.gif

http://gag.sourceforge.net/gagn2.gif

توزيعة Ubuntu

بتاريخ : 11/5/2006

هل تعرفون التوزيعه المشهوره Ubuntu، تميزت هذه التوزيعه بدعمها للكثير من لغات العالم و من ضمنها اللغه العربيه، بالإضافه إلى صغر حجمها حيث تقع في سي دي واحد مع الادوات و البرامج الهامه و ما زاد من شهرتها هو امكانية طلبها مجاناً من موقعها الرسمي و تصلك بعد عدّة اسابيع، هذه التوزيعه مبنيه على التوزيعه الشهيره Debian و طريقة نطقها (اوبنتو) كما ان هذه الكلمه تعني انسانيه بالافريقيه.

تتميز هذه التوزيعه بان يتم طرح اصداره جديده منها كل 6 اشهر و يستمر دعم هذه الاصداره حتى 18 شهراً بعد صدورها سطح المكتب الافتراضي لها هو GNOME و بالطبع تأتي مع مدير الحزم APT الذي يخص التوزيعه الشهيره Debian.

بالنسبه لي فانا استخدم هذه التوزيعه و من وجهة نظري هي اكثر من رائعه دعم تلقائي للغه العربيه بالإضافه إلى تعرفها على العتاد، و حتى تعريف المودم كان موجوداً على شكل حزمة deb، يمكنني القول ان هذه التوزيعه هي التي ادخلتني فعلياً إلى عالم لينكس.

و الآن ما رأيكم بإلقاء نظره سريعه على التوزيعات المتفرعه من توزيعة Ubuntu ؟

اولاً : **التوزيعات الرسميه** و المقصود بهذه التوزيعات انها مدعومه من مطورين Ubuntu .

Kubuntu :

لان سطح المكتب المرفق مع توزيعة Ubuntu هو GNOME و هناك الكثير من المستخدمين الذين يفضلون سطح المكتب KDE فتم توفير هذه التوزيعه خصّيصاً لمحبي سطح المكتب KDE، هي بالاصل توزيعة Ubuntu و لكن سطح المكتب الافتراضي فيها هو KDE بدلاً من GNOME.

موقع التوزيعه : http://www.kubuntu.org/

Edubuntu :

توزيعه تعليميه بحته ممتازه جداً للاطفال، ما رأيكم أنْ تعلموا اطفالكم ا ستخدام لينكس مع بعض المتعه مع الالعاب و الا دوات التعليميه :-)، في الحقيقه اعجبتني فكرة هذه التوزيعه و قد امر في تجربه معها مع الابنه الصغيره المدللة و إن تم ذلك بالفعل بالطبع سوف اكتب عن التجربه.

من البرامج التعليميه التي تحتويها هذه التوزيعه GCompris و الذي يحتوي على اكثر من 70 نشاطاً تعليمياً من اجل الاطفال، منها اكتشاف الحا سب و علم الجبر و الجغرافيا و الالعاب و القراءه، كذلك تحتوي على برامج KDE التعليميه.

موقع التوزيعه : http://www.edubuntu.org/

Xubuntu :

سطح المكتب Xfce هو سطح مكتب خفيف ولا يستهلك الكثير من الذاكره و يعتبر منا سب للاجهزه القديمه مقارنة بالا صدارات الجديده من KDE و GNOME، و لو تلاحظون لتجدون ان اغلب التوزيعات الصغيره و الخفيفه على الحوا سيب القديمه تستخدم سطح المكتب هذا، هذه التوزيعه هي توزيعة Ubuntu و لكنها تجعل Xfce كسطح مكتب افتراضي.

موقع التوزيعه : http://www.xubuntu.com/

التوزيعه المنتظره Gnubuntu

لحد الآن لم يتم ا صدار هذه التوزيعه و لكنها مشروع مستقبلي، تهدف هذه التوزيعه إلى ارفاق البرامج التي تخص حركة المصا در المفتوحه (FSF) فقط , بالطبع تعرفون ما ذا اعني بحركة المصا در المفتوحه , نعم Free As In Freedom :-).

للمزيد عن هذا المشروع : https://wiki.ubuntu.com/Ubuntu-libre

ثانياً : **التوزيعات غير الرسميه** و هي توزيعات مبنيه على Ubuntu و لكن المسؤولين عنها ليسوا نفسهم مطوري Ubuntu .

nUbuntu :

هذه التوزيعه هدفها بناء توزيعه مُشتقّه من Ubuntu مع اضافة برامج تخص اختبار الامن و ازالة بعض البرامج التي لا حاجة لها مثل GNOME و OpenOffice، يمكننا القول ان هذه التوزيعه تخص الشبكات و الامن حيث الحرف n في اول اسمها يعني Network، هذه التوزيعه مقارنة بتوزيعة اوبنتو الاخرى صغيره الحجم حيث يبلغ حجمها حوالي 200 ميجا بايت، تأتي هذه التوزيعه مع Fluxbox المشهور بسرعته و خفته على الذاكره، بالإضافه إلى برامج الامن المعروفه مثل Ethereal و nmap و DSniff .

الموقع الرسمي للتوزيعه : http://www.nubuntu.org/

Ubuntu Lite :

اوبنتو من اجل الحواسيب القديمه هذا الوصف البسيط لها .

موقع التوزيعه : http://www.ubuntulite.org

Ebuntu :

توزيعه مبنيه على Ubuntu و لكنها تستخدم مدير النوافذ Enlightenment . التوزيعه لا زالت في مراحل البيتا و هي صغيره الحجم حيث يبلغ حجمها 34 ميجا بايت . تم ازالة OpenOffice في هذه التوزيعه نظراً لحجمه الكبير .

صفحة التوزيعه : https://wiki.ubuntu.com/Ebuntu

اعد المتعه للحوسبه استخدم لينكس :-)

نظام التشغيل BSD لنتعرف على قصّته

بتاريخ : 31/5/2006

إذا كُنت احد متابعين نظم التشغيل مفتوحة المصدر لا بد و أنّه قد مرّ عليك نظام تشغيل يُسمى FreeBSD او OpenBSD او PC-BSD او حتى DesktopBSD، الكثير من الاشخاص لديهم اعتقاداً خاطئاً و هو أنّ نظام التشغيل FreeBSD هو احد توزيعات لينكس، و لكن الحقيقه هي انّ FreeBSD ليس Linux بل هو نظام تشغيل مُختلف مبني على نظام تشغيل يُسمى BSD، ما رأيكم الآن في التعرف على ما هو نظام التشغيل BSD ؟

في الحقيقه نظام التشغيل BSD ليس شبيه بيونكس فحسب بل هو نظام تشغيل مبني على يونكس، كما نعلم ان في الايام الاولى لنظام يونكس كانت تتوفر معه الشيفره المصدريه للجامعات و كان التعديل على الشيفره المصدريه مسموحاً ، و صل نظام يونكس إلى جامعه بريكلي حيث تم تركيبه على حواسيبهم في عام 1974 , الكثير أحبوا النظام و كان ناتج ذلك إنّ احد الطلّاب في الجامعه و هو Bill Joy قام بإطلاق اضافه تُسمى 1BSD في عام 1977 ، كانت 1BSD (و التي هي اختصاراً لـ First Berkeley Software Distribution) عباره عن اضافه لنظام التشغيل يونكس الاصدار السادس و كانت هذه الاضافه عباره عن مكونات الاساسيه لنظم التشغيل و هي كومبايلر للغة الباسكال و محرر نصوص اطلق عليه اسم Ex.

في عام 1978 قام Bill Joy بإطلاق حزمه جديده بإسم 2BSD حيث تعتبر تحديث لـ 1BSD ، تم اضافة برنامجين جديدين في هذه الحزمه و هما البرنامجان المعروفان الآن في عالم يونكس محرر النصوص vi و قشرة السي (C Shell) , في عام 1983 تم اصدار النسخه 2.9BSD و التي تعتبر نظام تشغيل متكامل مبني على الاصدار السابع من يونكس.

بعدها توالت الاصدارات إلى ان وصلت إلى 4.4BSD و إليكم قصّته : في عام 1994 ، كان النظام يتكون من 18 الف ملف و لم يتم ازالة إلا 3 منهم و تم التحرير في 70 ملف من اجل وضع ملاحظه حقوق الملكيه، في منتصف هذا

العام تم طرح الا صدار 4.4BSD و كان عباره عن ا صدارتين، الا صداره الاولى تُسمى 4.4BSD-Lite حيث كانت تُوزع مجاناً و لم تكن تحتوي على اي من شيفرات AT&T، و الا صدار الاخرى و هي 4.4BSD-Encumbered و كانت موجهه فقط لمن لديهم ترخيص لدى AT&T و سبب هذه الخطوه بالطبع هي تحويل نظام يونكس إلى نظام تجاري للبيع من خلال شركة AT&T و BSD في الا صل مبني على يونكس، و في حالة طرحهم للنسخه المبنيه على يونكس الحقيقي سوف يواجهون مشاكل قانونيه مع شركة AT&T.

و الآن جميع النُظم التي ذكرتها في اول التدوينه مبنيه ا صلاً على هذا النظام و هو BSD هل عرفتم الآن الفرق بين نُظم BSD و Linux ؟

فرق العمل .. لماذا لا ننجح في تكوينها ؟

بتاريخ : 4/7/2006

(نظام تشغيل عربي، لغة برمجه عربيه، كومبايلر عربي، بيئه تطوير عربيه) مؤخراً كثرت هذه العبارات في المنتديات المتخصصه، يمكنني القول انه في معدل كل سنه تقريباً يظهر موضوعاً جديداً يحاول صاحبه إنشاء "نظام تشغيل عربي" و قبل محاولته لإنشاء هذا النظام يحاول تكوين **فريق عمل** من اجل هذا النظام، و يستمر الحوار ما بين الواقعيون و الذين تمر عليهم هذه المواضيع كثيراً و ما بين المتحمسون الذين يريدون في اقل من سنه برمجة نظام تشغيل من الكيرنل إلى الواجهه الرسوميه **!** , بعد فتره بسيطه من النقاش (و المشاجره احياناً) يذهب الموضوع، و يظهر مره اخرى موضوع مشابه له بعد فتره، و هكذا هي الحاله، مع كثرة هذه المواضيع إلا انه لحد الآن لا يوجد اي موضوع نجح في تكوين فريق عمل ولو من 3 اشخاص و قاموا بعمل حقيقي ملموس **!**

ليس مشروع نظام التشغيل فقط، بل اغلب المشاريع التي نحاول أنْ نعمل منذ البدايه فيها جماعة تفشل، هل تسائلنا يوماً عن المشكله ؟ و اقترحنا حل لهذه المشكله ؟ حسناً برأيي ان هناك عدة اسباب لهذه المشكله و هي :

* عدم التخطيط الجيد
* الحماس الزائد و الاحلام الزائده بدون وجود عمل حقيقي
* عدم وجود خبره كافيه (احياناً)
* عدم التنظيم
* بالإضافه إلى كثرة الكلام و قلّت العمل **!**

و النقطه الثالثه اخصّها للمشاريع الضخمه مثل نظم التشغيل و لغات البرمجه و ما شابه، في الحقيقه ارى انه من الصعب جداً تكوين فريق عمل بهذه الطريقه (و هي كتابة موضوع في المنتديات منتظراً المتطوعين) و قبل البدأ في المشروع، و هناك اسلوبان اضمن نجاحهم في تكوين فريق عمل لمشروع معين و هما :

عمل فردي ثم جماعي :

و أقصد بها أنْ يقوم مبرمج واحد فقط ببرمجة البرنامج و طرحه، و بعد فتره من إصدار البرنامج و الاستمرار به يتم فتح

الابواب لمن يريد المشاركه ! في هذه الحاله يمكن للناس ان يروا شيئاً ملموساً يشجعهم على المشاركه، ولا يسمعون هتافات الحماس فقط بدون اي عمل، شخصياً مررت بهذا النوع من فرق العمل في MySmartBB .

فريق العمل من يبرمج البرنامج :

و اقصد بها أنْ يبدأ البرنامج بواسطة فريق عمل و ليس مبرمجاً واحداً فقط، و هذا الذي يأسنا منه و لم نتمكن من تحقيقه ! و لكن برأيي أنّه يمكن تحقيقه في حاله واحده فقط، و هي أنْ يكون المشاركون في فريق العمل هذا يعرفون بعضهم حق المعرفه حتى يتمكنوا من التواصل بسهوله، و لا يكونوا اشخاصاً لا يعرفون بعضهم و يبدأون بتكوين الفريق و قبل البدايه ينتهي المشروع !

الآن لو اتينا إلى مشروع "نظام التشغيل" الذي يأتينا سنوياً شخص يحلم به و يحاول إنشاء فريق عمل و لا يفلح في ذلك، بفرض إنّه إتبع الاسلوب الاول و هو أنْ يقوم بكتابة نظام تشغيل في منتهى البساطه و يبدأ بطرح اصدارات متعدده منه، و يفتح الابواب لمن يريد مساعدته بأي طريقه، في هذه الحاله سوف يرون الناس شيئاً ملموساً و موجوداً فعلاً و هناك "مشروع" و ليس مجرد "هتافات حماسيه"، و افضل مثال ناجح على هذا النوع هو نظام التشغيل Linux، بدأه Linus Torvalds بمفرده، و الآن يطوره عدد كبير من المبرمجين حول العالم

حسناً أما بالنسبه للطريقه الثانيه و هي أنْ يبدأ البرنامج بوجود فريق عمل و ليس بوجود مبرمج واحد، برأيي أنّ هذا الاسلوب لن ينجح إلا في حاله واحده و هي أنْ يعرف اعضاء الفريق بعضهم اشد معرفه – كما ذكرت مسبقاً -، لانه في هذه الحاله سوف يكون التواصل بينهم سهلاً، بالإضافه إلى أنّهم يعرفون ظروف بعضهم، و اخيراً سوف يكونون متفاهمين لأكبر درجه، و مثالاً على ذلك برنامج منتديات الـ vBulletin الذي بدأ بشخصين على ما اتذكر.

و الآن هل هناك من سوف يطبّق ذلك بالفعل ؟ ام ان هذه المقاله مجرد "حبر في ورق" :-)

مره اخرى : إنشاء نظام تشغيل غير مستحيل !

بتاريخ : 16/8/2006

الواضح اننا لن ننتهي من هذا الموضوع :-)

بالنسبه لي فأنا متابع و بشدّه موضوع "نظام التشغيل العربي" و النقا شات التي تدور حوله آخر موضوع قرأته بهذا الخصوص تفضل به احد الاخوان و ذكر نقطه ان "انشاء نظام تشغيل من الصفر امر مستحيل" .

حسناً ، الآن لنتعرف على قصدنا في كلمة "إنشاء نظام تشغيل" هل نقصد اننا نريد انشاء نظام تشغيل متكامل مع واجهته الر سوميه و برامجه، ام اننا نريد بناء نواة نظام تشغيل ؟

طبعاً الامر ليس مستحيلاً في كلتي الحالتين .

حسناً سوف اتكلم عن النقطة الثانيه و هي **بناء نواة نظام تشغيل،** بالطبع هي ليست مستحيله و ممكنه و لكنها في الحقيقه سوف تحتاج إلى بذل جهد و قد يكون هذا الجهد كبيراً ، سوف نحتاج إلى خبره برمجيه في كل من السي و الا سمبلي، و سوف نحتاج إلى فهم جيّد للمفاهيم (العمليات , خيوط التنفيذ , ا دارة الذاكره , نظام الملفات) هذه تقريباً هي المفاهيم الا سا سيه في نظم التشغيل.

موضوع بناء النواة قد يكون ابسط من موضوع بناء نظام تشغيل كامل , طبعاً لا اقصد انه ابسط من نا حية البرمجه و لكنني اقصد انه يتطلب وقت اقل لاننا في حالة بناء نظام تشغيل متكامل سوف نقوم ببناء نواة، وواجهه ر سوميه، البرامج الا سا سيه .

هممممممم , ما رأيكم بأخذ شاهد بسيط على كلامي بخصوص بناء النواة و ان الموضوع غير مستحيل؟

تفضلوا :

/http://www.djm.co.za/spoon
/http://clicker.sourceforge.net
/http://www.geocities.com/taj_os
/http://agnix.sourceforge.net
/http://virtuose.tuxfamily.org
http://manrix.sourceforge.net/index.htm

ما رأيكم لو قلت لكم ان اغلبها برمجها شخص واحد او على الاقل بدأها شخص واحد.

الآن لنتكلم عن النقطه الاولى و هي **بناء نظام تشغيل متكامل** و اقصد به النواة بالإضافه إلى البرامج المهمه (و غالباً تكون الواجهه الرسوميه احد هذه البرامج خصوصاً مع الانظمه الحديثه) , هذا الامر كذلك غير مستحيل و لكنه يتطلب بعض الوقت الاضافي بعد برمجة النواه

تفضلوا هذه الامثله البسيطه :
http://www.skyos.org
http://www.reactos.org

هذا فقط في حالة اردنا ان يكون نظامنا مختلف عن البقيه ما رأيكم لو جعلنا نواتنا متوافقه مع معايير POSIX ؟ اقصد جعلناها Unix-Like سوف يعمل مترجم GCC على نواتنا و بالتالي GNU BASH و بالتالي الكثير من برامج Unix و من ضمنها GNOME او KDE و اخيراً حصلنا على نواة تخصنا و ادوات من اجل تكامل النظام من مجتمع المصادر المفتوحه.

الذين يذكرون دائماً قصة لينوس تورفالدز و نجاحه مع لينكس لا بد أنْ يدركوا انه قام ببناء النواة، و كانت نواته شبيهه بيونكس، و بالتالي عملت البرامج التي تخص يونكس عليها، و الناتج الآن هو لينكس الذي ترونه.

و الآن لنأخذ عينه من المشاريع التي تريد بناء "نظام تشغيل متكامل" و ليس بناء نواة فقط.

سوف اتكلم اولاً عن Haiku في هذا المشروع يريدون بناء "نظام تشغيل" شبيه بـ BeOS، سهل للاستخدام،

قوي للاداره، حر و مجاني. له واجهته الخا صه , غالباً له برامجه الخا صه التي تعمل عليه، النقطه الاهم و التي اريد ذكرها انه نواته لم يقم ببنائها فريق عمل المشروع، بل قاموا بأخذ نواة جاهزه تسمى NewOS (للمزيد حول هذه النواة ا ستخدموا Google)

و الآن لنأخذ عينه اخرى

نظام تشغيل آخر ا سمه Syllable، يريد هذا المشروع إنشاء "نظام تشغيل" هدفه أنْ يكون نظاماً سهلاً للاستخدام، مجاني و مفتوح المصدر، صالح للاستخدام المنزلي و المكاتب الصغيره، كذلك هذا النظام له واجهته الر سوميه التي تخصه و له برامجه التي تخصه، و ا ذا تحدثنا عن نواته فلم يكتبها ا صحاب المشروع بل بنوا نظامهم على نظام كان موجو داً مسبقاً ا سمه AtheOS !

تعالوا معي قليلاً و دعونا ننظر إلى مشروع Agnix، هذا المشروع يريد بناء "نواة نظام تشغيل" يريدها نواة مفتوحة المصدر، صغيره و سريعه، ما رأيكم أنْ نقوم بأخذ هذه النواة و تشغيل GCC عليها ثم تشغيل BASH ثم تشغيل X11 و اخيراً تشغيل GTK و تشغيل GNOME ؟ حسناً حسناً هل تريدونه نظاماً مختلفاً عن نظم يونكس الاخرى، لا بأس سوف نحتاج فقط إلى مكتبات للغة السي و تشغيل بعض لغات البرمجه الاخرى على هذا النظام و نصبح مستعدين للعمل ! نقوم ببرمجة مكتبتنا الخا صه لإنشاء الواجهات الر سوميه مثل الـ GTK، ثم نقوم ببناء واجهتنا الر سوميه التي تخصنا و في كلتا الحالتين حصلنا على نظام تشغيل متكامل من صنع يدنا.

ا ذا ار دتم المزيد من الامثله فلدي المزيد !

حسناً الآن اريد شخص واحد فقط ير د على سؤالين بعد قرائته لهذه المقاله و هذه الامثله

* هل من المستحيل أنْ ابني نواة نظام تشغيل ؟
* هل من المستحيل أنْ ابني نظام تشغيل متكامل ؟

حد دوا هدفكم فقط و ستعرفوا طريقكم هل تريدون نظام تشغيل متكامل ام نواة نظام تشغيل ؟

لو تلاحظوا ان جميع الامثله التي تحدثت عنها و جميع النظم كانت نظم **مفتوحة المصدر**، حتى لا يأتي احد و يقول ان النظم التي ذكرتها تخص شركات و لدى الشركات مئات المبرمجين الذين يعملون لصالحها بل هذه المشاريع جميعها مفتوحة المصدر انشئت بجهود افراد، و بعضها يعمل عليها شخص واحد فقط مثل Agnix.

و الآن اترككم مع و صلات النظم التي ذكرتها بالامثله :

http://agnix.sourceforge.net

/http://haiku-os.org

http://syllable.org

اتمنى ان يصل هذا الموضوع للمهتمين به .

الفرق بين اللغات المُترجمه و اللغات المُفسره

بتاريخ : 27/ 10/2006

ما رأيكم أنْ ندخل اليوم في انواع لغات البرمجه؟ اذا كنت قد برمجت من قبل لا بد و أنّه مرّ عليك هذا الاختلاف بين مجموعه من اللغات، بعض لغات البرمجه مثل السي و البا سكال عندما تكتب برامجك بها في النهايه سوف تقوم بترجمتها (Compile) و سوف يكون الناتج النهائي ملف تنفيذي تقوم بتوزيعه من اجل الا ستخدام، و هناك لغات اخرى مثل الـ PHP و الـ Python عندما تقوم بكتابة برامجك بها و تريد توزيع برنامجك يجب عليك توزيع الشيفره البرمجيه للبرنامج.

هل تسائلت ما هو الفرق ؟

السي و البا سكال و غيرها من اللغات التي يكون ناتجها ملف تنفيذي تُسمى باللغات المُترجمه (Compiled Language) ، تعمل هذه اللغات بالآليه التاليه :

* يقوم المبرمج بكتابة الشيفره المصدريه .
* يقوم بتمريرها إلى **المترجم** .
* يقوم المترجم (او الكومبايلر) بعملية الترجمه و صنع ملفات Object ثم تبدأ عملية الربط بوا سطة Linker .
* الناتج يكون **ملفاً تنفيذياً** .

اما الـ PHP و الـ Python و ما شابهها من لغات تُسمى باللغات التفسيريه (Interpreted Language) و تعمل هذه اللغات بالآليه التاليه :

* يقوم المبرمج بكتابة الشيفره المصدريه .
* يقوم بتمريرها إلى **المفسّر** .
* يبدأ المفسر بـ **تنفيذ** الملف سطراً تلو الآخر .

اذاً ببساطه الفرق بين اللغات المترجمه و اللغات المفسره ، أنّ اللغات المترجمه تقوم بترجمة الشيفره المصدريه إلى

اللغه الثنائيه ، بينما تقوم اللغات المفسره بالتنفيذ مبا شره بدون الترجمه .

قد تتسائل كيف يمكنني معرفة ا ذا كانت اللغه تفسيريه ام مُترجمه ، بكل بساطه انظر إلى الناتج ، هل ستقوم اللغه في النهايه بإنشاء ملف تنفيذي يمكنك توزيعه – في هذه الحاله سوف تكون اللغه مترجمه (مثل السي ، البا سكال ، الجافا) ، اما ا ذا لم يكن الناتج ملفاً تفسيرياً و كان توزيع برنامجك يتطلب إرفاق الشيفره البرمجيه كما انه يتطلب وجو د برنامج معين (وهو المفسر) في جهاز المستخدم فهذه اللغه تفسيريه . مثلاً البايثون عندما تريد توزيع برنامجك يجب ان توزعه كشيفره مصدريه و على من يريد الا ستخدام تثبيت مفسّر البايثون على حا سوبه و بالتالي يعمل البرنامج.

وقفه مع الجافا ، هل هي تفسيريه ام مترجمه؟

بتاريخ : 27/ 10/2006

في مقالة الفرق بين اللغات المفسره و المترجمه ذكرت ان لغة الجافا من اللغات المترجمه ، قد يتبا در في ذهن البعض عندما يسمع هذه الكلمات سؤال و هو "ا ذاً لما يضطر المستخدم الذي يريد تشغيل برامج جافا بتثبيت آلة جافا الافتراضيه (JVM) على حا سوبه حتى تعمل برامج جافا؟"

حسناً في الحقيقه يمكنني القول ان لغة جافا خلطت بين اللغات التفسيريه و اللغات المترجمه ، كيف ذلك ؟ تعمل لغة الجافا كالتالي :

* يكتب المبرمج الشيفره المصدريه .
* يمررها إلى **مُترجم** جافا . كما نعلم ان مترجمات لغات البرمجه مثل السي و البا سكال تترجم الشيفره المصدريه إلى اللغه الثنائيه و **لكن** مترجم الجافا يقوم بترجمة الشيفره المصدريه إلى لغه خا صه تُسمى Byte Code .
* ينتج ملف يكون مكتوب بـ لغة البايت كو د .

ملف البايت كو د يحتوي على شيفرات لا يمكن ان يفهمها إلا جهاز جافا الافتراضي JVM ، عندما اقوم بتوزيع البرنامج (وهو الملف الناتج من عملية الترجمه) لا بد من وجو د JVM في جهاز المستخدم حتى يتمكن من تشغيل البرنامج بحيث يقوم JVM بتنفيذ شيفرة البايت كو د الموجو ده ضمن البرنامج .

و الآن نستنتج ان لغة الجافا في الحقيقه لغه مترجمه تقوم بترجمة الشيفره المصدريه إلى شيفرات تسمى بـ البايت كو د ، و بالتالي تُمرر هذه البايت كو د على برنامج يفهمها و ينفذها يُسمى هذا البرنامج آلة جافا الظاهريه JVM .

وقفه مع النواة الاحاديه و النواة الصغريه

بتاريخ : 21/12/2006

حديثي اليوم عن نُظم التشغيل و تصميمها ، اولاً لا بد ان نعرّف مصطلح نواة او Kernel .

النواة او الكيرنل هو قلب نظام التشغيل ، يقوم الكيرنل بجميع العمليات التي تخص ا دارة العتا د من ا داره للذاكره و ا داره للعمليات (التي بدورها تتعلق بالمعالج) و ا دارة الملفات (التي بدورها تتعلق بالقرص الصلب و الاقراص المرنه و غيرها من و سائط تخزين) كما ان النواة لها علاقه بالتشبيك (لكنني في الحقيقه لا اعلم الكثير عن موضوع التشبيك و علاقته بالنواه) .

تعمل النواة في فضاء (Space) يسمى بفضاء النواة (Kernel-Space) ا ذا عمل اي برنامج في هذا الفضاء سوف تكون له القدره الكامله على ا ستخدام العتا د بدون اي حدو د و سوف يتمكن من ا ستخدام كامل الطاقه العتا ديه في المعالج و دائماً النواة هي التي تعمل في هذا الفضاء بما انها تقوم بإ دارة العتا د و هي بحاجه إلى كل خصائص العتا د .

هناك فضاء آخر يسمى بفضاء المستخدم (User-Space) في هذا الفضاء تعمل جميع البرامج الاخرى (جميع البرامج من متصفح انترنت و محرر نصوص و غيره) و بالطبع لديها بعض المحدو ديات اي انها لن تتمكن من الو صول إلى جميع خصائص العتا د .

في عام 1992 اي بعد وضع لينكس على الانترنت بسنه تقريباً ، قام البرفسور اندرو تانينباوم (مبرمج نظام مينكس ـ Minix ـ) بطرح انتقا د لـ لينكس و بعدها دارت بينه و بين لينوس تورفالدز (الاب الروحي لـ لينكس ـ Linux ـ) مناقشه طويله سُميت في ما بعد بـ (مناقشة تانينباوم و تورفالدز) كان ا ساس هذه المناقشه يدور حول النواة الاحا ديه (Monolithic kernel) و النواة الصغريه او المصغره (Microkernel) ، كان اعترا ض البرفسور تانينباوم على اتخا ذ نواة لينكس التصميم الاحا دي بدلاً من التصميم الصغري .

حسناً ما رأيكم ان نتعرف على قصة المونولثك كيرنل و المايكرو كيرنل ؟

النواة الاحا ديه - Monolithic :

من اقدم التصاميم و كان نظام التشغيل يونكس يستخدمها يصفها البرفسور تانينباوم في كتابه (تصميم و تنفيذ نظم التشغيل الحديثه) بـ فوضى كبيره ! .

في هذا التصميم تعمل النواة كاملة بما فيها من خدمات (ا دارة العمليات ، ا دارة الذاكره ، نظام الملفات) و كل ما يخصها سواء احتاج إلى الو صول إلى الميزات منخفضة المستوى ام لا .

من سلبيات هذه الطريقه انها سوف تُنتج نواة كبيره الحجم و لكن في نفس الوقت من ايجابياتها انها ا سرع من النواة الصغريه .

من نُظم التشغيل التي تستخدم هذا الا سلوب هي Linux , BSD , Unix و هناك الكثير و لكن هذا ما اتذكر

النواة الصغريه - Microkernel

هي نوع احدث من تصاميم الانويه ، تدور فكرة النواة الصغريه كالتالي :
النواة الا سا سيه تقوم بجميع الوظائف منخفضة المستوى في ا دارة الذاكره و ا دارة المعالج و غيره و تعمل هذه النواة بالطبع في فضاء النواة .
هناك برامج عا ديه تُسمى بـ الخوا دم (Servers) هذه البرامج تعمل كـ عمليات منفصله في فضاء المستخدم و تقوم بتقديم خدمات نظام التشغيل مستعينة بالنواة الا سا سيه في حال احتياجها إلى القيام بعمليات منخفضة المستوى (و التي تتطلب من البرنامج العمل في فضاء النواة) اما العمليات مرتفعة المستوى فتقوم بها بدون الا ستعانه بالنواة .

مثلاً لدينا خا دم (Server) و هو مدير العمليات ، يعمل كبرنامج عا دي فوق النواة الصغريه ، يقوم بتقديم جميع الخدمات التي يقدمها اي مدير عمليات و في حال حاجته إلى القيام بعملية منخفضة المستوى يقوم بالاتصال بالنواة حتى تقدّم له ما يريد ، لنقدم مثالاً على ذلك . عند انشاء عملية جديده يقوم مدير العمليات بوضع هذه العمليه في جدول العمليات و بالطبع سوف يحتاج إلى بعض المعلومات عن مكان العمليه في الذاكره بالإضافه إلى وضع العمليه في الذاكره و ما شابه من العمليات منخفضه المستوى ، يقوم مدير العمليات مبا شره بالاتصال بنواتنا الصغريه و طلب ما يحتاجه منها من عمليات منخفضه لن يتمكن من تنفيذها بمفر ده .

تتميز هذه الطريقه إنها تعطينا نواة صغيرة الحجم كما انها سوف تكون سهله التطوير (Debug) طالما ان الخوادم تعمل في فضاء المستخدم و يمكننا عمل اعادة تشغيل لها من دون اعادة تشغيل الجهاز بالكامل و من السلبيات انها تكون ابطأ من النواة الاحاديه .

و من الامثله على النواة الصغريه Mach , QNX , Minix و لا اتذكر البقيه :-)

حسناً نتيجة لتركيب النواة الصغريه بهذه الطريقه جعلت البعض يأخذ نواة صغريه جاهزه و يبني فوقها خوادم تخصّه و هذا ما حدث مع GNU/HURD حيث اخذ نواة Mach الصغريه و بدأ ببناء الخوادم فوقه (طبعاً هذا ملخص القول هناك تفصيل تقني آخر و تصميم محترف لـ GNU/HURD قد اذكره في تدوينه منفصله)

هناك بالطبع تصاميم اخرى للانويه و لكنها ليست بشهرة ما ذُكر في الاعلى و في الحقيقه اغلبها عباره عن تعديلات على فكرة النواة الصغريه او تكون هجينه ما بين الاثنين و قد اذكرها في موضع آخر

للاستزاده راجعوا -Microkernel , Monolithic Kernel , The Tanenbaum Torvalds Debate في جووجل

GNU/HURD قصته و علاقته بالنواة الصغريه

بتاريخ : 21/12/2006

في عام 1985 بدأ ريتشارد ستالمن و مجموعه في بناء نظام تشغيل حُر يكون بديلاً لنظام يونكس الذي تحوّل إلى نظام تجاري ، بالفعل بدأوا في بناء الادوات الاساسيه من مترجم (GCC) و مفسر اوامر (BASH) و غيره ، عندما وجدوا ان جميع البرامج الاساسيه جاهزه كان عليهم كتابة قلب نظام التشغيل (النواة ـ Kernel) و بالفعل بدأوا بـ GNU/HURD و اختاروا له تصميماً احترافياً جعله يتأخر لسنوات كثيره و لا زال حتى يومنا هذا (2006) في مراحله التجريبيه الاولى (بدأوا به في عام 1990) و قد كان هذا سبباً لاختيار نواة لينكس كبيئه لعمل ادوات GNU بواسطة ريتشارد ستالمن .

حسناً ما هو هذا التصميم الاحترافي الذي جعل GNU/HURD يتأخر كل هذه الفتره ؟ قبل كل شئ ارجو منك مراجعة موضوع (وقفه مع النواة الاحاديه و النواة الصغريه) في حال عدم معرفتك لاساسيات النواة الصغريه .

لا يمكننا القول ان GNU/HURD نواة و لكنها عباره عن خوادم تعمل فوق نواة صغريه (Microkernel) تسمى بـ Mach في الحقيقه كان اختيار النواة الصغريه في بادئ الامر متخبطاً حيث اختاروا نواة غير Mach و لكن وجدوها ليست جيده مما دعاهم للانتقال إلى Mach و يتم العمل الآن على جعل GNU/HURD قابل للعمل فوق نواة صغريه احدث تسمى L4 .

إذاً GNU/HURD ليس نواة و لكنها خوادم مبنيه فوق نواة صغريه ، كما ذكرنا سابقاً ان الخوادم تعمل فوق نواة صغريه و تجري اتصالات في ما بينها لإعطاء و اخذ المعلومات ، و لكن تصميم GNU/HURD اكثر احترافيه حيث تعمل فوق النواة شياطين (Daemons) بدلاً من خوادم (Servers) ـ و معنى الشياطين هنا عباره عن عمليات تعمل في الخلفيه لا يشعر بها المستخدم ـ و تعمل فوق هذه الشياطين الخوادم ، و هذا يعني ان GNU/HURD متعدد الخوادم Multi Server بعكس الانويه الصغيره الاخرى احادية الخادم Single Server .

هذا التصميم الاحترافي جعل GNU/HURD يتأخر كثيراً

راجع :

* http://www.gnu.org/software/hurd/hurd.html
* http://en.wikipedia.org/wiki/GNU/HURD
* /http://www.serdal.com/archives/2006/07/05/gnu-linux

خيوط التنفيذ ... نظره سريعه

بتاريخ : 1/ 1/2007

مرحباً جميعاً كل عام و انتم بخير و عيدكم مبارك ان شاء الله .

ما هي خيوط التنفيذ

في الحقيقه هو ليس مفهوماً برمجياً فحسب بل يقع كذلك ضمن مفاهيم نظم التشغيل ، اي برنامج يعمل على الحا سوب يُسمى "عمليه" و عندما يكون نظام التشغيل متعدد العمليات (كما هو الحال مع نظم التشغيل الحديثه الآن) فإنّه يقوم بتوزيع وقت المعالج على هذه العمليات ، بحيث يعطي لكل عمليه وقتها من المعالج .

خيوط التنفيذ (بالانجليزيه Threads) هي عباره عن تقسيم لعمليه واحده إلى مجموعه من الخيوط التي تعمل ضمن نطاقها و لكل خيط (Thread) وظيفه خا صه يقوم بها ، تتعاون خيوط التنفيذ مع بعضها البعض حتى تنجز اعمال العمليه الا سا سيه التي قمنا بتقسيمها إلى مجموعه من الخيوط ، اما العمليات فيمكننا و صفها بأنها في حالة حرب حتى تحصل كل واحده منها على اكبر وقت ممكن من المعالج .

اقرب مثال على ا ستخدام الخيوط هو برامج معالجات النصوص ، نجد ان معالج النصوص يستقبل من المستخدم الدخل و في نفس الوقت يقوم بتخزين نسخه احتياطيه من الملف و كذلك يقوم بالبحث بدون ان يشعر المستخدم و يكمل عمله كما و ان شئ لم يكن و بالطبع كل وظيفه تتم عباره عن خيط منفصل ، مثلاً تخزين النسخه الاحتياطيه خيط منفصل عن البحث و هكذا .

لا للجداول!

بتاريخ : 23/6/2007

مقدمه :

قبل فتره طويله قمت بوضع الشكل الحالي للمدونه، و قمت بأخذ احد القوالب الحرّه من اجل تركيبها على المدونه بعدما اقترح علي سر دال ذلك و ظهرت المدونه كما ترونها الآن باللون الاخضر و بشكل يشبه المدونات الاخرى ، عندما قمت بتركيب القالب على برنامج المدونه كنت لا اعلم الكثير عن XHTML و المعايير القيا سيه، ا ستغربت ان القالب يقوم بتقسيم الصفحه بدون ا ستخدام الو سم table بل يكتفي بـ div، و اعتقد انني عبّرت عن ذلك في التدوينه التي كتبت فيها عن تحديث المدونه تنا سيت الموضوع قليلاً و لم اقرر ان اقوم بأي تطبيق عملي، قبل ايام قليله عدت لرؤية القوالب الجاهزه و عا د شعوري بأنني ضعيف في CSS مره اخرى، لذلك قررت ان ا سترجع ما در سته مسبقاً حول هذه اللغه (ان صحّت تسميتها لغه او تقنيه)، و عا د موضوع و سم table، بحثت قليلاً لأجد صفحات بعنوان **لا للجداول**.

الموضوع الا سا سي :

حسناً ، الغرض الا سا سي من وجو د الجداول في HTML هو **عرض البيانات**، و لكن لو نلاحظ قليلاً سوف نرى اننا نستخدمها لأكثر من عرض البيانات**!** نستخدمها في تصميم تركيبة الصفحه كامله، نقسّم الصفحه بإ ستخدام الجداول، و نقوم بتقسيم هذه الاقسام من خلال الحقول بالطبع مع ا ستخدام border="0" حتى يختفي الجدول كاملاً .

هناك العديد من الا سباب التي قد تجعلنا نعيد النظر بهذه الطريقه**!** كما ذكرت مسبقاً الجداول لعرض البيانات فقط، و ليست طريقه لتصميم الصفحات بها، قد لا يهمك ذلك و لكن هناك المزيد من الا سباب التي تستحق ان نلتفت لها، مثلاً الجداول ابطأ في العرض، كما انها تزيد من الحجم، و تجعل من عملية الو صول إلى الموقع عن طريق الاجهزه الكفيه مثلاً امراً صعب، ا ذاً يجب علينا ان نترك الجداول و نستخدمها في تخصصها فقط**!** و هي عرض البيانات لا اكثر، مثلاً لدينا بيانات يجب علينا وضعها في جدول مثل ا سماء الموظفين و معلومات عنهم، يجب ان نستخدم الجداول، و لكن ا ذا كنّا نريد تقسيم الصفحه و وضع صوره برأس الموقع و تحتها قائمه للو صلات فالجداول ليست الحل الافضل**!**

بعد كل هذا ما هو الحل يا ترى؟ ما هو البديل للجداول، ببساطه هي CSS من خلال CSS يمكننا تقسيم صفحاتنا بالطريقه المناسبه و حسب الخواص التي نحتاجها، و لن نفقد اي شئ كنا نتمتع به ايام استخدامنا للجداول من اجل تقسيم صفحاتنا! بالعكس سوف تكون هناك ميزات و فوائد اكثر.

كيف ابدأ؟

يبدو انك اعدت التفكير :-) حسناً الموضوع ليس بالصعوبه المتوقعه، اخذ مني يوماً واحداً فقط (لدي معرفه مسبقه بـ CSS)، سوف الخص لك الخطوات في نقاط :

1. اذا كنت قد درست CSS مسبقاً قم بمراجعه سريعه عليها هنا (من وجهة نظري ان اول ثلاثة دروس كافيه).
2. اذهب هنا و قم بإختيار احد التصاميم البسيطه و حمّلها.
3. استعن بـ المرجع حتى تتمكن من معرفة الخواص المتوفره لديك.
4. قم باللعب قليلاً في ملف CSS الموجود مع التصميم الذي قمت بتحميله و قم بفتح صفحة HTML و اعرف وظائف CSS التي قام بتعريفها المصمم.
5. عُد إلى الموقع المذكور في الخطوه الثانيه و قم بتحميل تصميماً آخراً بسيطاً و قم بمحاولة تقليده.
6. راجع فهي صفحه مفيده

اتمنى ان تكون الخطوات مساعده بالنسبه لي سوف احاول تغيير جميع صفحاتي إلى هذا النمط، لمعرفة المزيد عن سيئات استخدام الجداول لغير الغرض التي صُممت لها راجع جووجل: كلمة البحث No Table CSS

المعايير ضروريه جداً!

بتاريخ : 20/7/2007

يا إلهي، لا زال البعض يناقش من جدوى تطبيق المعايير القيا سيه التي تخص لغة HTML ، موضوع المعايير يدور في ذهني منذ ايام و لكنني اليوم رأيت احد المواضيع التي تناقش المعايير القيا سيه التي وضعتها W3C من اجل لغة HTML لذلك قررت كتابة هذه التدوينه.

الحقيقه، ان وجو د المعايير شئ هام و ضروري في كل مكان و ليس في الحا سوب فقط، هناك العديد من الامثله التي يمكن ذكرها بخصوص المعايير في عالم الحا سوب و سوف ا ذكر بعضها حتى يقتنع غير المقتنعون بأهمية المعايير.

مثالنا الاول هو لغة SQL الكثير منّا يعرف هذه اللغه، من خلال هذه اللغه يمكن للمبرمج ان يتعامل مع قواعد البيانات، مثلاً يضيف البيانات، يحدّثها او يستعلم عنها ... إلخ، حسناً في الحقيقه لا يوجد لـ لغة SQL جهه ر سميه تقوم بتصميمها حتى اوضّح هذه الفكره، هناك برامج تُسمى محركات قواعد البيانات و مثال على هذه البرامج هو محرّك MySQL، هل تسائلنا ذات يوم ما هي علاقة لغة SQL في MySQL؟ هل MySQL هي SQL نفسها؟

في الحقيقه الجواب لا، محرّك MySQL عباره عن "برنامج" وظيفة هذا البرنامج تخزين البيانات بطريقه معينه و إحضارها عند طلبها و تحديثها عند طلب ذلك، حتى يؤ دي MySQL هذه الوظائف يستخدم SQL من اجل ذلك كواجهه للمستخدم، حيث يستخدم المُستخدم لغة SQL للتعامل مع MySQL، هذا يعني ان لغة SQL موجو ده في محرّك MySQL نفسه، مثلاً عندما نر سل احد اوامر SQL إلى محرك MySQL فإن محرك MySQL **هو الذي يعالج هذه الاوامر**، لدينا مثالاً آخراً من محركات قواعد البيانات و هو محرّك PostgreSQL، هذا المحرك كذلك عباره عن "برنامج" وظيفته تخزين البيانات بطريقه معينه و احضارها و تحديثها عن الطلب، اي ان وظيفته مشابهه لمحرك MySQL، كذلك محرّك PostgreSQL يستخدم لغة SQL من اجل التعامل معه. النقطه التي اريد الو صول إليها ان لغة SQL ليست موجو ده على شكل ملفات برمجيه يمكن لمن يريد اخذها و وضعها في محرك قواعد البيانات الذي يبرمجه، بل هي موجو ده على شكل "معايير"، مثلاً فريق محرك MySQL عندما بدأوا ببرمجته و قرروا انهم سوف يستخدمون SQL كواجهه لا ستخدام محركهم، لم يذهبوا إلى موقع SQL الر سمي و يحمّلوا بعض

الملفات البرمجيه و يرفقوها في مشروعهم حتى يدعموا SQL فيه، **بل قاموا هم** بكتابة هذه الملفات البرمجيه التي تدعم SQL في محركهم، كذلك بالنسبه لمشروع PostgreSQL.

ما ذا يعني كل هذا! تخيلوا معي الآن انني قررت برمجت محرك قواعد بيانات خاص بي بدلاً من MySQL الذي ا ستخدمه منذ سنين، و كذلك قررت ان تكون لغة SQL هي واجهة الا ستخدام لانني اعرفها بشكل جيّد، بدأت في كتابة المحرك و انتهيت منه كاملاً و ا سميته "محرك قواعد البيانات سين"، بعدها شعرت ان لغة SQL بها بعض القصور لذلك قررت **اضافة 10 ميزات جديده لها** حتّى تسهل علي الامور، و بدأت انشر هذا المحرك حتى يتمكن من ا ستخدامه الجميع، و اتى مبرمج آخر و قام بكتابة محرك قواعد بيانات آخر و دعم SQL بمحركه الذي ا سماه "محرك قواعد بيانات صا د" و اضاف 25 ميزه جديده لـ SQL في محركه و بدأ بنشره، تخيلوا كم مبرمج سوف يبرمج محرك قواعد بيانات ثم يرفق لغة SQL فيه و **يضيف إليها ميزات جديده**، هذا يعني ان الشخص ا ذا ا ستخدم "محرك قواعد البيانات سين" لفتره ثم قرر ان ينتقل إلى "محرك قواعد البيانات صا د" بعدها لسبب او لآخر سوف يضطر لتعلم ا شياء جديده في SQL لم يكن لها وجو داً في المحرك سين، و هكذا مع كل محرك قواعد بيانات سوف يكون هناك تباين فهناك محرك اضاف الكثير من الميزات الجديده و محرك آخر لم يضيف اي ميزه و اكتفى بالا سا سيات و هكذا، في هذه الحاله سوف يكون **لكل برنامج محرك قواعد بيانات لغة SQL خا صه** و لن تكون لدينا لغة SQL واحد بل سوف تكون لدينا **لغات** SQL لا يوجد بينها عامل مشترك سوى الا سم، في هذه الحاله **تأتي المعايير لتحل هذه المشكله الضخمه** من خلال تحديد معايير معينه للغة SQL سوف تتوحد لغة SQL في جميع محركات قواعد البيانات، و بالتالي الذي يتعلم لغة SQL سوف يتعلمها لمره واحده و يستخدمها في جميع محركات قواعد البيانات بدون ان يشعر بزيا ده او نقصان.

حسناً الآن بعدما اكتشفت انه هناك معايير قيا سيه للغة SQL، عندما اقرر ان ابرمج محرك قواعد البيانات الخاص بي سوف ا ذهب لدرا سة هذه المعايير ثم اطبقها في محركي حتى يتوافق مع معايير SQL القيا سيه و بالتالي لن يجد المستخدم فرقاً في لغة SQL الموجو ده في MySQL و SQL الموجو ده في محركي.

مثال آخر، اللغه المعروفه C، هناك برنامج يعرفه المبرمجون بشكل جيّد و هو "الكومبايلر" Compiler، وظيفة هذا البرنامج هي تحويل الشيفرات المصدريه المكتوبه بأي لغة برمجه إلى شيفرات بـ لغة الآله حتى يتمكن الحا سوب من فهم البرنامج و تنفيذه، لو ألقينا نظره سريعه جداً لوجدنا ان هناك العديد من كومبايلرات لغة السي، منها كومبايلر GCC الذي برمجه و يُشرف عليه مجتمع البرامج الحره و Turbo C الخاص بشركة بورلاند و الكثير غيرها، ما يحدث مع السي هنا هو بالضبط ما يحدث مع SQL، في با دئ الامر قام دينيس ريتشي بتصميم لغة " سي" و قام بكتابة كومبايلر خاص بها و ا ستخدمه، و بعدما ا شتهرت اللغه وا صبح الكومبايلر الا صلي الذي قام دينيس ريتشي ببرمجته لا

فائده منه لسبب او لآخر، قرر المبرمجون كتابة كومبايلرات للغة السي، لذلك كان لا بد من وجود معايير يتبعها هؤلاء المبرمجون.

حسناً لنفرض ان كل شخص قام ببرمجت كومبايلر لـ السي قام بعمل بعض التغييرات على التركيب النحوي لـ اللغه بدلاً من الالتزام بالتركيب النحوي للغه الا صليه، سوف يصبح لكل كومبايلر مميزاته الخا صه و بالتالي عندما اكتب برنامج لترجمته من خلال الكومبايلر "س" لن اتمكن من ترجمته على الكومبايلر "ص"، ببساطه **وجود معيار يعني التوحيد**، هناك معيار للغة السي و بالتالي ا صبحت لغه واحده لجميع الكومبايلرات (ابحث عن ANSI C).

الكلام السابق يدلنا على حقيقه هامه جداً و هي ان الكثير من لغات البرمجه ما هي إلا **معايير** و خصو صاً لغات البرمجه القديمه و اللغات التي يتم ا ستخدامها ضمن البرامج، اما بالنسبه للغات القديمه لان ا صحابها صمموها اولاً و بعدما ا شتهرت و بدأ الناس بتصميم كومبايلرات لها كانت هناك حاجه لوضع معايير يتبعها كل من يريد تصميم كومبايلر للغه، كما ذكرت اللغات القديمه هي غالباً التي ينطبق عليها هذا الشئ (لغات قديمه مثل السي،فورتران،بيسك،با سكال و لكنني لست متأكداً من وجود معايير معتمده بالنسبه للغات الثلاث الاخيره) اما اللغات الجديده مثل جافا،بايثون،روبي،بي اتش بي فهي ليست **معايير** انما هي **لغات** موجوده و الا شخاص الذين صمموها لا زالوا يقفون وراءها و يطورها، لنفرض بعد 50 سنه من الآن ذهبت لغة الجافا الا صليه و لم يعد لها وجود و بدأ الناس بإيجا د الكومبايلرات البديله هنا سوف تأتي الحاجه إلى معيار محد د حتى لا يكون هناك اختلاف ما بين الكومبايلرات، نعو د لنقطتنا اما بالنسبه للغات التي يتم ا ستخدامها ضمن البرامج هي لغات تستخدمها البرامج من اجل ان تُعطيها فائده معينه مثل SQL، البرامج التي تترجم هذه اللغه ليست كومبايلرات و انما برامج تستخدمها من اجل عملية ا دارة البيانات، كذلك بالنسبه لـ لغات مثل HTML و التي تستخدم ضمن برنامج المتصفح من اجل عرض الصفحات، هذه اللغات لابد من توافرها على شكل معايير، لان هناك الكثير من المتصفحات التي تتعامل مع الـ HTML مثلاً، و وجود معيار لـ HTML سوف يوحّد جميع هذه المتصفحات و لا يسمح لمتصفح معين ان يضيف ا شياءاً جديده لـ HTML لا يعرفها المتصفح الآخر، و هذه نقطه قمه بالاهميه.

لا زال بعض من مطورين صفحات الويب معترضون على معايير HTML التي عرفوها مؤخراً، من قرأ هذه التدوينه سوف يُجزم على اهمية المعايير و اهمية إتباعها، تخيلوا معي لو لم يكن هناك معاييراً للغة HTML، و كان لدينا هذا الكم من المتصفحات الذي نعرفه الآن مثل Firefox, Opera, Safari و غيرها، لو لم يكن هناك معاييراً تتبعها هذه المتصفحات من اجل معالجة لغة HTML لقام مثلاً مطورون Safari بإضافة و سم HTML جديد لم يكن معروفاً مسبقاً، و ا ستخدمه ملايين مطورين صفحات الويب و لكن! هذا الو سم الذي اُضيف في Safari لن يُعالج

إلا به و لن يراه إلا مستخدمي Safari، اما مستخدمي Opera و Firefox مثلاً سوف تظهر لهم اخطاء في الصفحه و لن تظهر بشكل صحيح، و بالتالي سوف يضطرون من اجل حل هذه المشكله اضافة هذا الوسم و هكذا مع التطور يقوم كل متصفح بإضافة اوسمته الخاصه و يصمم المطورون صفحاتهم على اساس متصفح معين، تخيلوا مدى ضخامة المشكله عندما اُريد تصفح موقع معين لا بد ان اتصفحه عن طريق متصفح معين ولا يمكنني تصفحه عن طريق المتصفح الذي استخدمه! في الحقيقه هذه المشكله ليست موجوده الآن و لكن بعض المتصفحات لا زالت لا تُطبق المعايير القياسيه بشكل صحيح، اقرب مثال إلى ذهني الآن هو alt التابع للوسم img، يجب وضع نص في داخله و الهدف من ذلك اذا لم يجد المتصفح الصوره يطبع الكلام الموجود في داخل alt، و لكن متصفح IE (الاصدار السادس لا ادري عن الاصدار السابع) لا يفعل ذلك بل يقوم بإظهار مربع صغير يحتوي على الكلام الموجود في alt بعد وضع الفأره لفتره على الصوره، و عندما حدثت ثورة Mozilla و Firefox و كثر استخدامهما و لانهما يطبقان المعايير القياسيه، استغرب الناس و ظنوا ان الخطأ من Firefox لانه لا يُظهر ذلك المربع الصغير، و انما هناك خاصيه اخرى غير alt من اجل هذا الغرض، هذا بالنسبه لمبرمجين المتصفحات، اما بالنسبه لمطورين الويب فأعتقد انهم لا بد لهم من تطبيق المعايير القياسيه، لانهم من خلالها سوف يستخدمون الوسوم الصحيحه و يتأكدون من انها سوف تعمل على **جميع** المتصفحات، كما ان المعايير القياسيه التي تخص الـ HTML تساعدهم على تنظيم شيفرتهم

لقد اطلت في الحديث :-(و لكن لدي مثال اخر بالنسبه للمعايير، ماضي يونكس معروف و كتبت عنه شخصياً في اكثر من موضع، بعدما تعددت نسخ يونكس كان الجميع يضيف ما يشاء بدون وجود معيار محدد، و لكن بعد مده تم وضع المعيار المعروف الآن بـ POSIX، و الآن كل من يطبق هذا المعيار في نواته (Kernel) سوف توافق نواته نظم يونكس الاخرى، كذلك بالنسبه للينكس بعدما كثرت توزيعاته ظهر المعيار LSB من اجل توحيد هذه التوزيعات، حتى على مستوى برامج اسطح المكاتب في نظم يونكس (GNOME و KDE و ما شابهها) ظهرت معايير حتى تتوافق اسطح المكاتب هذه مع بعضها البعض.

المعايير هامه جداً، من رأيي لا بد ان يكون لكل شئ في عالم الحاسوب معايير تخصه محررات النصوص، برامج المراسله، برامج المحادثه، برامج ادارة المواقع، برامج المنتديات كل شئ! حتى البرامج التي تساعدنا على التحكم بالصوت في حاسوبنا يجب ان يكون لها معايير! (لا بد و انني بدأت بالمبالغه قليلاً 😁).

و اخيراً انتهت التدوينه 😁

نظره على عملية السُبات (Hibernate)

بتاريخ : 12/11/2007

عملية السُبات هي افضل ترجمه وجدتها لـ Hibernate، هذه الميزه لابد و انها مرّت على اغلب مستخدمين نظم التشغيل الحديثه، ا صحاب الحوا سيب المحموله (Laptop) خصو صاً يستخدمونها كثيراً من اجل توفير الطاقه، و لكن هل تسائلنا ما الذي يتم بالضبط في هذه العمليه؟

ببساطه عندما تقوم بعمل Hibernate يُغلق الحا سوب، و عندما تقوم بتشغيله مره اخرى يعود كل شئ كما كان قبل الاغلاق، جميع البرامج التي كانت تعمل و الملفات المفتوحه لا زالت موجوده.

كما نعلم، ان جميع البرامج التي تعمل حالياً على الحا سوب و الملفات التي نعمل بها من خلال هذه البرامج مُخزنه على الذاكره الرئيسيه (RAM)، و ذاكرة الـ RAM كما هو معروف **متطايره**، المقصود من متطايره انها **تفقد محتوياتها بمجرد قطع التيار الكهربائي عنها**، عملية الـ Hibernate تقوم بأخذ محتوى ذاكرة RAM و تخزينها في **القرص الصلب** ثم تُغلق الحا سوب، و عندما يتم تشغيل الحا سوب من جديد يتذكر نظام التشغيل ان هناك عملية Hibernate قد حصلت، و بالتالي يقوم بقراءة الملف الذي كتبه على القرص الصلب و وضع محتواه في ذاكرة RAM مره اخرى، و بالتالي جميع البرامج التي كانت تعمل قبل عملية Hibernate سوف تعود مره اخرى كما كانت، فكره رائعه اليس كذلك؟

كما عودتنا نظم التشغيل الحرّه (و خصو صاً لينكس) بالمرونه العاليه، حيث يوجد اكثر من برنامج (ان صحّت التسميه، هي في الحقيقه رُقع (Patch)) يقوم بنفس العمليه و يمكن لمدير النظام اختيار المنا سب منها، هناك swsusp و TuxOnIce و uswsusp (الذي يعمل في فضاء المستخدم)، قد اكتب عنها بشكل منفصل ا ذا وجدت ما يستحق الكتابه.

مفهوم لا بد من تصحيحه، برامج ادارة المحتوى

بتاريخ : 22/11/2007

برامج ادارة المحتوى (Content management system او اختصاراً CMS)، يستخدم مطورون المواقع العربي كلمة "مجله" غالباً لوصف برامج ادارة المحتوى.

الفكره من برنامج ادارة المحتوى انها تسهّل من عملية إنشاء الموقع، مثال على ذلك يرغب شخص بإنشاء موقع، و يجب أنْ يحتوي موقعه هذا على منتديات، دليل مواقع، قسم للمقالات و آخر للاخبار، و بالتالي سوف يضطر هذا الشخص إلى تركيب سكربت منتديات و سكربت دليل مواقع و سكربت للمقالات و سكربت للاخبار، و بالتالي كل سكربت سوف يكون منفصلاً عن الآخر سواء في قواعد البيانات، او بالتصميم الخارجي او حتى بالاداره عن طريق لوحة التحكم.

برامج ادارة المحتوى تحل مثل هذه المشكله، بحيث يكون البرنامج الاساسي عباره عن "نواة" و السكربتات المطلوب تركيبها على الموقع تكون "برامج اضافيه" متوافقه مع "النواة" يمكن تركيبها عليها و حذفها منها بسهوله، و تقوم النواة بعملية الربط و الدمج، مثلاً جميع البرامج تكون ضمن قاعدة بيانات واحده، كما يمكن توحيد التصميم، كذلك جميع البرامج يمكن ادارتها عن طريق لوحة تحكم واحده، و كذلك يتم توحيد الاعضاء المُسجلين.

مثال على هذه البرامج، برنامج جمله (Joomla!)، و هو برنامج ادارة محتوى حُر، "جمله" بحد ذاتها عباره عن "نواة" و تقوم بالربط بين "الوحدات" (Modules) و هي برامج مكتوبه بطريقه متوافقه مع "النواة" (جمله)، و قد تكون هذه الوحدات اي نوع من البرامج، منتديات، سجل زوار، ادارة مقالات، ادارة اخبار.

من الناحيه البرمجيه (**وهذا ما يهمني**) برامج ادارة المحتوى **ليست** مجموعه من البرامج المتصله مع بعضها البعض و التي لا يمكن فصلها عن نواة البرنامج نفسه و التي تكون مرتبطه معه بصفه دائمه، بل يجب ان تكون هذه البرامج عباره عن "نواة" توفر طريقه سهله لـ "وحدات" ان تعمل فوقها، و لا مانع من وجود "وحدات" رسميه تأتي بشكل رسمي مع هذه النواة و لكن بشرط معاملة هذه "الوحدات" على انها وحدات و ليست جزء من "النواة"، اقصد ان لا يتم خلط

ملفات هذه "الوحدات" مع "النواة" و جعلها ا سا سيه بحيث لا يمكن التخلص منها!

هل و صلت الفكره الآن؟

اتمنى من الذين كتبوا او يفكرون بكتابة برنامج ا دارة محتوى بالا سلوب غير الصحيح محاولة اعا دة النظر بالفكره و كتابتها بشكل صحيح، لان الطريقه الصحيحه هي اكثر مرونه و سوف تُشعرني حتماً بأن البرنامج اكثر احترافيه :-)

نظره على المعالجات

بتاريخ : 10/3/2008

المعالج (او المعالج الصغري Microprocessor و يُسمى احيانا CPU اختصاراً لـ Central Processing Unit) من اهم الاجزاء العتا ديه في الحا سوب، لذا سوف اتناول المعالجات في هذه التدوينه مع شئ من التفصيل

عندما يتم تشغيل برنامج معين، فإن نظام التشغيل يقوم بنسخ شيفرة البرنامج (والتي تكون وقتها عباره عن شيفره ثنائيه Binary او بتعبير ابسط عباره عن مجموعه من الا صفار و الواحدات) إلى الذاكره الرئيسيه (او كما نعرفها جميعاً بـ RAM)، الذاكره الرئيسيه تقوم بالتخزين فحسب اما الذي يقوم بتنفيذ البرنامج هو **المعالج**، في المعالج مجموعه من الدارات الالكترونيه الصغيره و التي تساعده على تنفيذ التعليمات، و غالباً ما تكون هذه التعليمات عباره عن عمليات رياضيه او منطقيه، يدور المعالج دائماً عندما يعمل و في كل دوره يقوم المعالج بإحضار التعليمات من الذاكره الرئيسيه (RAM) و ينفذها، لا بد من التنويه ان كذلك التعليمات عباره عن رمز يتكون من الا صفار و الواحدات و يحفظ المعالج هذا الرمز على انه لتعليمه معينه، مثلاً لتعليمة الجمع هناك رمز معين لنفرض انه 100110 ، عندما يجد المعالج هذا الرمز في شيفرة احد البرامج فإنه يقوم بعملية الجمع.

لان عملية إحضار التعليمات من الذاكره الرئيسيه امر قد يستغرق بعض الوقت، فإن مصممين المعالجات حاولوا تقليل هذا الوقت عن طريق إضافة الذاكره المخبئيه (Cache) و ا ستخدموا مجموعه من الا ساليب الاخرى مثل Pipeline.

بالنسبه للمسجلات (Registers) فإنها عباره عن ذاكره اقرب إلى المعالج من الذاكره الرئيسيه، مصنوعه من نفس ما دة المعالجة وتكون في داخله حتى يكون من السهل على المعالج الو صول إليها بسرعه، هناك نوعان ا سا سيان من المسجلات، نوع عام و يستخدم للاغراض العامّه كتخزين المعطيات مثلاً و نوع خاص و تكون المسجلات في هذا النوع لها وظائفها الخا صه مثل المسجل Programe Counter الذي يُخزن به مكان الذاكره للتعليمه التي سوف يتم تنفيذها بعد التعليمه الحاليه، احياناً يتم تحديد المعالجات حسب المسجلات، مثلاً هناك معالجات تسمى

بمعالجات 32 بت و هذا يعني ان حجم المسجل في هذا المعالج هو 32 بت مرفوع الأُس لاثنين و كذلك بالنسبه لمعالجات 64 بت و التي تعني انه حجم المسجل.
كما نلاحظ في تعاملتنا اليوميه اننا نستخدم مصطلح جيجاهيرتز عندما نتحدث عن المعالجات، الجيجاهيرتز = 1000000000 هيرتز (الف مليون هيرتز)، و الهيرتز الواحد يعني دوره واحده بالثانيه، لنأخذ مثالاً سريعاً، اذا كانت سرعة معالجك تساوي 1 جيجاهيرتز هذا يعني انه يدور الف مليون دوره في الثانيه الواحده، للتبسيط اذا كنّا نريد استخدام وحده اقل من الجيجاهيرتز للتعبير عن سرعة المعالج فإننا نستخدم الميجاهيرتز و ليس الهيرتز مباشره.

كما ذكرنا سابقاً ان لكل تعليمه في المعالج رمز، تستخدم البرامج هذا الرمز من اجل إخبار المعالج بأنه يُريد من المعالج تنفيذ وظيفه معينه، يُمكننا القول (مثال فقط للتوضيح) ان المُعالج يُخزّن التعليمه (الرمز) و الوظيفه المقابله لهذه التعليمه في جدول بداخله، يضم هذا الجدول كُل التعليمات التي يوفرها المعالج (يُسمى هذا الجدول او القائمه بـ Instruction Set Architecture او ISA اختصاراً)، هناك نوعان من المعالجات الشائعه، المعالجات التي تحتوي على عدد كبير من التعليمات يُطلق عليها CISC (اختصاراً لـ Complex instruction set computer) بما معناه ان الجدول الذي يحتوي على التعليمات (او التعليمات التي يوفرها هذا المعالج) مقعد نظراً لعدد التعليمات الكبيره التي يحتويها هذا الجدول، المعالجات متعددة الاغراض غالباً ما تُصنّف على انها CISC، لانه و بالطبع الاجهزه متعدده الاغراض قد تُستخدم في الكثير من الوظائف ولا يوجد وظيفه محدده، و بالتالي لا بد ان يخدم المعالج جميع هذه المتطلبات، هناك نوع آخر من المعالجات و هي بعكس معالجات CISC حيث تحتوي على عدد اقل من التعليمات و يُطلق عليها اسم RISC، و غالباً ما تكون هذه المعالجات ذات اغراض معينه، مثلاً مُوجهه للاجهزه الصغيره مثل الهواتف النقاله او حتى اجهزة الالعاب مثل بلاي ستيشن، بالطبع هناك انواع اخرى غير RISC و CISC و لكن الاشهر هي التي ذكرتها.

في اللحظه الواحده لا يمكن للمعالج تنفيذ اكثر من برنامج، هذه هي الحقيقه نظم التشغيل الحديثه تتيح لمستخدميها تشغيل اكثر من برنامج في نفس الوقت و العمل عليها، و لكن في اللحظه الواحده لا يُنفذ المعالج إلا برنامج واحد فقط، الجدير بالذكر ان المعالجات الحديثه توفر الكثير من الخدمات لمبرمجين النُظم و تسهل عليهم الكثير حتى يتمكنوا من كتابة نظم تشغيل تدعم تعدد العمليات او ادارة الذاكره بإستخدام اسلوب الذاكره الظاهريه.

المعالجات بشكل عام ما عدا البسيطه منها تحتوي على نمطين، النمط الاول هو نمط النواة Kernel-mode في هذا النمط يمكن استخدام جميع التعليمات الموجوده في المعالج حيث لا حدود، اما النمط الثاني هو نمط المستخدم User-mode و في هذا النمط لا يمكن استخدام كل شئ في المعالج بل هناك حدود، و كما هو معروف غالباً

ان نظام التشغيل يعمل في نمط النواة حتى يتمكن من التحكم الكامل بالعتا د، بينما تعمل برامج المستخدم في نمط المستخدم حيث لا يمكنها ا ستخدام كل القوه العتا ديه للمعالج، ذكرت هذا الموضوع مسبقاً بشئ من الا سهاب في التدوينات السابقه و التي تتحدث عن Microkernel و Monolithic kernel.
قد يتسائل البعض كيف يمكن للمعالج وهو آله من القيام بالعمليات الرياضيه او المنطقيه، جواباً على هذا السؤال اقرءوا عن البوابات المنطقيه (Logic Gates)

ما ذا بعد؟ لا اتذكر قد يكون لدي المزيد و لكن لا اتذكره الآن خصو صاً انني ا شعر بالنعاس :-)، قد اتذكر في ما بعد و اكتب تدوينه منفصله

كتب تطوير الذات كيف نتعامل معها؟

بتاريخ : 8/9/2008

قبل كل شئ، مبارك عليكم الشهر

لابد و ان الكثير منّا قام بشراء مجموعه من كتب تطوير الذات، سوف اخبركم بقصتي مع هذا النوع من الكتب و قد تستفيدون منها، اولاً لابد ان نتعرف على كتب تطوير الذات ماهي؟ لست متأكداً من صحّة معلوماتي و لكنني اعتقد انها إلى حد كبير صحيحه، فكتب تطوير الذات هي تلك الكتب التي تحاول أنْ تقدّم لك افكاراً في مجال معيّن حتى تتطور فيه، مثلاً كتب الاداره بأنواعها، الكتب التي تتحدث عن تحقيق الاهداف، بالاضافه إلى ذلك الكتب التي تتحدث عن مواضيع معينه تخص تطوير الشخصيه مثل التفكير الايجابي او البرمجه اللغويه العصبيه (NLP)، حسب اعتقادي ان هذه هي الكتب التي تُصنّف على انها كتب تطوير الذات (ارجو من مَن يخالفني بالرأي اخباري بذلك).

لندخل في صُلب الموضوع، خلال الفتره السابقه قمت بشراء العديد من كتب تطوير الذات، و كلما بدأت بقراءة احدها اشعر بإنه غير مفيد و ممل و بالتالي اقرر إغلاقه لقراءته فيما بعد و انتقل إلى كتاب آخر و على هذا المنوال، تسائلت اذا كانت المشكله منّي ام من الكتب نفسها؟ اكتشفت اخيراً ان الخطأ هو خطأي كنت اقرأ كتاب تطوير الذات قبل الذهاب للنوم و غالباً ما كنت اقرأه و انا على السرير اتجهز للنوم و كأنني اقرأ احد كتب التاريخ! عندما اكتشفت ان الخطأ منّي قررت ان اتخذ مجموعه من الخطوات و اذا نجحت معّي اكتب عندها في المدونه.

اولاً : كتب تطوير الذات بحاجه إلى تركيز شديد، غالباً ما يقوم الكتاب بطرح مجموعه من التعريفات التي يجب استيعابها بشكل جيّد ثم يطرح افكاراً مبنيه على هذه التعريفات، او على الاقل يقوم بطرح فكره من اجل تنفيذها، و بالتالي لابد من التركيز الشديد اثناء القراءه.
ثانياً : لان كتاب تطوير الذات كما اسلفنا يطرح مجموعه من الافكار التي يقترح عليك تنفيذها بالتالي نحن نحتاج إلى تلخيص، بالطبع الكتاب متوسط الحجم سوف يقترح عليك الكثير و الكثير من الافكار، و بالتأكيد لن تتمكن من تذكر جميع هذه الافكار بعد قراءتك لها لمره او حتى لمرتين، وبالتالي يكون التلخيص فكره ممتازه، كل ما تحتاج إليه هو ورقه و قلم امامك او حاسوب ان كنت تفضّله، و كل تعريف او فكره تستخلصها من الفقره تقوم بتلخيصه بإسلوبك، بهذه

الحاله سوف تلخّص جميع افكار الكتاب و يمكنك الرجوع إليها فيما بعد بشكل سريع دون الحاجه إلى اعادة قراءة الكتاب، كما يمكنك عند التلخيص تجاهل بعض الافكار غير المهمه بالنسبه لك. (من الجميل مشاركة الآخرين بملخصاتك عن طريق مدونتك مثلاً)

ثالثاً : اعتقد انه من الجيّد قراءة الكتاب مرتين على الاقل، بحيث اذا لم تفهمه بشكل جيد في المره الاول تفهمه جيداً في المره الثانيه.

بالنسبه لي، اتبعت هذه الطريقه و فعلاً تمكنت من قراءة احد كتب تطوير الذات بشكل كامل بدون الشعور بالضجر او الشعور انه غير مفيد، على العكس كنت غالباً ما اشعر بالمتعه و يزيد حبّي للكتاب و إعجابي بالمؤلف، اسم الكتاب هو "غيّر تفكيرك غيّر حياتك" للمؤلف "بريان تراسي"، انتهيت من قراءته لأول مره و انا اعيد قراءته حالياً للمره الثانيه بمزيد من التدقيق على الافكار و تلخيصها، بالتأكيد سوف اطرح الملخصات ان شاء الله عندما انتهي منها

نظره على عمليات الـ Bitwise

بتاريخ : 6/ 12/2008

توفر اغلب لغات البرمجه عمليات الـ Bitwise، كنت دائماً اتسائل عن فائدة هذه العمليات؟ حتى نتعرف على فائدة هذه العمليات لا بد من دراسة اكثر من موضوع، سوف استخدم لغة سي في هذه التدوينه من اجل التوضيحات.

جميعنا نعلم ان كل شئ في الحاسب يُخزن بالصيغه الثنائيه (اصفار و واحدات)، بما في ذلك المتغيرات التي نُعرّفها في برامجنا، و نعلم ان الرقم الثنائي الواحد يُمثّل بما يسمى بـ Bit، و اذا جمعنا 8 بتات مع بعضها سوف نحصل على بايت واحد، يمكن للبايت الواحد أنْ يمثّل قيمه مفيده، بعكس البت الواحد الذي لن يمثّل إلا قيمه واحده و هي إما الصفر او الواحد، لنفرض اننا الآن نريد تمثيل الرقم العشري 6 بشكل ثنائي، سوف يكون تمثيله ضمن بايت واحد كالتالي :

```
0000 0110
```

في لغة البرمجه سي، و تحت المعالجات من نوع 32 بت يكون حجم الرقم الصحيح int عباره عن 4 بايت، و حجم الحرف char عباره عن بايت واحد، الذي اريد توضيحه من هذا الشرح ان عمليات الـ Bitwise تتعامل على مستوى البت، مثلاً لو استخدمنا مع العدد 6 الذي مثلناه بشكل ثنائي في الاعلى احد عمليات الـ Bitwise فسوف تتعامل هذه العمليه مع البت، سنأخذ امثله لتوضيح الفكره اكثر، و لكن بعد التعرف على عمليات الـ Bitwise.

عمليات الـ Bitwise اربع هي : AND و OR و XOR و NOT، الثلاث عمليات الاولى تقارن بت ثنائي مع بت ثنائي آخر و كما نعلم، البت الواحد إما ان تكون قيمته 1 او تكون 0، لذا سوف تكون لدينا نتائج محدوده، اما العمليه الاخيره فإنها تتعامل مع بت واحد فقط مما يقلل النتائج، لا بد من ملاحظة شئ هام جداً و هو ان القيمه 1 تعني True (بمعنى صحيح) و القيمه 0 تعني False (بمعنى غير صحيح).

لنبدأ بتفصيل عملية AND، لنفترض السطر التالي :

```
X AND Y
```

تقول العمليه AND : اذا كان البت الاول **و** البت الثاني قيمتهما 1 ، فأرجع القيمه 1 ، و إلا أرجع القيمه 0.
ولا يخفى ان قيمة X إما ان تكون 1 او 0 و قيمة Y إما ان تكون 1 او 0 لاننا نتعامل مع النظام الثنائي.

في هذه الحاله يمكننا حصر القيم التي تُعيدها العمليه AND كالتالي :

```
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1
```

نلاحظ ان AND دائماً تُعيد 0 إلا اذا كان البت الاول و البت الثاني قيمتهما واحد.

ننتقل الآن إلى العمليه الثانيه وهي OR، لنفترض السطر التالي :

```
X OR Y
```

تقول العمليه OR : اذا كان البت الاول **او** البت الثاني قيمة احدهما يساوي 1 ، فأرجع القيمه 1 ، و إلا ارجع القيمه 0.

في هذه الحاله يمكننا حصر القيم التي تُعيدها العمليه OR كالتالي :

```
0 OR 0 = 0
0 OR 1 = 1
1 OR 0 = 1
1 OR 1 = 1
```

بعكس العمليه AND، نلاحظ ان OR دائماً تُعيد 1 إلا اذا كانا البتّين قيمتهما 0.

نعرَج على العمليه الثالثه وهي XOR، بفرض السطر التالي :

```
Z XOR Y
```

تقول العمليه XOR : اذا كان البت الاول **او** البت الثاني قيمة احدهما يساوي 1 ، فأرجع القيمه 1 **إلا اذا كان البتّان متشابهين فأرجع 0**.

كما تلاحظ ان الفرق بين XOR و OR في التشابه بين البتّين، فإذا تشابها سوف نُعيد 0، و اذا لم يتشابها سوف نعيد 1 ، و بالتالي يمكننا حصر القيم التي تعيدها XOR كالتالي :

```
0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0
```

العمليه الاخيره هي NOT و تستقبل بتاً واحداً و تقوم بعكس هذا البت، فإذا قلنا NOT 1 فالقيمه المُعاده سوف تكون 0 و العكس صحيح.

بالإضافه إلى هذه العمليات الاربع، تعتبر عملية التحريك (Shift) من ضمن عمليات الـ Bitwise، تقوم عمليات التحريك بتحريك التمثيل الثنائي إلى اليمين او إلى اليسار بعدد من البتات، لنفرض القيمه التي مثلناها مسبقاً و هي :

```
0000 0110
```

اذا اردنا **تحريك** هذه القيمه إلى اليسار بطول بت واحد سوف تصبح القيمه كالتالي :

```
0000 1100
```

يساوي هذا التشميل القيمه 12 بالنظام العشري، و اذا اردنا تحريك هذه القيمه إلى اليسار بطول 2 بت، سوف تنتج لدينا القيمه التالي :

```
0011 0000
```

يعمل التحريك إلى اليمين بنفس الطريقه، و لكن لابد اخذ موضوع Two's Complement بعين الاعتبار (قد نتطرق لهذا الموضوع في تدوينه اخرى).

حسناً اخذنا الآن نظره كافيه على عمليات الـ Bitwise، لنأخذ الآن بعض الامثله، نعود إلى تمثيل العدد 6 ضمن بايت واحد :

```
0000 0110
```

لنطبّق معه العمليه AND :

```
0000 0110 AND 1
```

لاحظ اننا قلنا فيما سبق ان عمليات الـ Bitwise تتعامل على مستوى البت، هذا يعني ان السطر السابق سوف يقوم بعمل AND بين البت 1 الموجود على يمين AND و البت 0 الموجود في اقصى اليمين بالنسبه للعدد 6 المُمثل بشكل ثنائي، الناتج بالطبع لن يتغير، لان 1 و 0 = 0، حسناً ما ذا لو كنّا نريد عمل AND مع آخر بتّين؟ سوف يكون السطر كالتالي :

```
0000 0110 AND 11
```

ما ذا لو كنّا نريد التعامل مع اول بتّين من اليسار؟ سوف تكون العمليه كالتالي :

```
0000 0110 AND 1100 0000
```

و هذا ما سينطبق على بقية العمليات.

فهمنا الآن بفضل الله عمليات الـ Bitwise بشكل جيّد، السؤال القادم، ما هي الفائده من وراء هذه العمليات؟ في

الحقيقه هناك مجموعه من خصائص الاعداد الثنائيه (و تمثيلها) التي يمكننا من خلال عمليات الـ Bitwise الاستفاده منها.

الخاصيه الاولى هي : ان البت الاول من اليمين يحدد اذا كان الرقم المُمثل زوجي ام فردي، بحيث اذا كانت قيمة هذا البت 1 سوف يكون العدد فردياً اما اذا كانت قيمة هذا البت 0 سوف يكون العدد زوجياً، قبل معرفتك لهذه الخاصيه في الارقام الثنائيه لو كنت قد طلبت منك كتابة برنامج صغير بلغة السي يقوم بإكتشاف اذا كان العدد زوجياً ام فردياً سوف تكتبه كالتالي :

```
#include <stdio.h>

int main()
{
    int x;

    scanf("%d",&x);

    if ((x % 2) != 0)
    {
        printf("Odd\n");
    }
    else
    {
        printf("Even\n");
    }

    return 0;
}
```

في هذه الحاله، استخدمنا الخاصيه الرياضيه في ان اذا كان العدد يقبل القسمه على 2 بدون باقي فإنه عدد زوجي، و لكن اذا كان هناك باقي فإنه عدد فردي، ما رأيكم لو استخدمنا خاصية الارقام الثنائيه التي تحدثنا عنها منذ قليل؟ وهي ان البت الاول من اليمين اذا كانت قيمته 1 هذا يعني ان الرقم المُمثل رقم فردي و اذا كانت قيمته 0 هذا يعني ان الرقم المُمثل رقم زوجي، كيف يمكننا يا ترى استخدام عمليات الـ Bitwise لإستغلال هذه الخاصيه؟ تذكر اننا قلنا ان العمليه AND تعمل كالتالي "اذا كان البت الاول **و** البت الثاني قيمتهما 1، فأرجع القيمه 1، و إلا أرجع القيمه 0."، و بالتالي عندما نستخدم العمليه AND مع الرقم و نستخدم الرقم 1 كبت ثاني سوف تُعيد عملية AND القيمه 1 اذا كان الرقم فردياً لان 1 **و** 1 يساوي 1 و تُعيد القيمه 0 اذا كان الرقم زوجي 0 **و** 1 يساوي 0، لنرى المثال بلغة سي :

```
#include <stdio.h>

int main()
```

```
{
    int x;

    scanf("%d",&x);

    if ((x & 1) == 1)
    {
        printf("Odd\n");
    }
    else
    {
        printf("Even\n");
    }

    return 0;
}
```

و لاحظ ان & في الجمله الشرطيه هنا تعني AND الخا صه بالـ Bitwise و ليس لها علاقه بعنوان الذاكره او ما شابه، هذه احد الفوائد و بالاضافه إلى ذلك هناك العديد من الخصائص في التمثيل الثنائي التي يمكن ا ستغلالها عن طريق عمليات الـ Bitwise مثل Two's Complement، و عمليات الضرب التي يمكن انجازها بإ ستخدام التحريك، يمكنك البحث في جووجل للمزيد.

من الا شياء المثيره التي لا بد من ذكرها قبل إنهاء التدوينه هي انه يمكننا إنجاز عملية التبديل (swap) بين مُتغيران عن طريق ا ستخدام XOR و هذه الخوارزميه تسمى بـ (XOR Swap Algorithm)، إذا ار دنا انجاز عملية تبديل عا ديه بلغة السي سوف تقوم بالتالي :

```
#include <stdio.h>

int main()
{
    int x = 10,y = 20;
    int temp;

    temp = x;
    x = y;
    y = temp;

    return 0;
}
```

لاحظ اننا في هذه الحاله احتجنا إلى متغير جديد temp من اجل عملية التبديل، و لكن مع خوارزمية التبديل بواسطة XOR يمكننا التبديل بدون الحاجه إلى متغير ثالث، لنرى المثال المنقول من ويكيبيديا :

```
void xorSwap (int *x, int *y)
 {
     if (x != y) {
         *x ^= *y;
         *y ^= *x;
         *x ^= *y;
     }
 }
```

يمكنكم التبحر اكثر عن طريق البحث، كان هدفي من هذه التدوينه تعريف هذه العمليات و تبيين جزء من فائدتها، و ارجو من الله ان اكون وُفقت في ذلك.

لغة البرمجه Kbasic

بتاريخ : 4/2/2009

مرحباً بكم من جديد

لغة برمجه جديده و كما يقترح اسمها فإنها احدى لغات BASIC، تستخدم هذه اللغه مكتبة Qt بشكل اساسي و طبعاً كما توقعت تعتبر لغة KBasic لغة رسوميه، هذا يعني انه بإمكانك تصميم واجهة برنامجك بسرعه كبيره و كتابة شيفرتك بدون الحاجه للقلق بخصوص الواجهه الرسوميه، لغه حرّه و تدعم البرمجه الكائنيه Object-Oriented متوافقه مع Visual Basic 6 و QBasic كما انها تأتي بدعم لـ VB.NET من ناحية التركيب النحوي و من ناحية الاصناف و الكائنات التي تُقدمها.

عندما ألقيت نظره سريعه على هذه اللغه فكرت يا ترى ماهو الفرق بينها و بين Gambas التي لها نفس المميزات تقريباً ؟ ما يجعل لغة KBasic مثيره بالفعل انها لغه متعددة المنصات، فهي لا تعمل على غنو\لينكس فقط مثل Gambas و لكنها تعمل على ماك و وندوز، وهذا يعني انه حتى البرامج التي تكتبها تعمل على الانظمه الثلاث و اعتقد انها اللغه الوحيده التي تتميز بأنها حرّه و رسوميه و تعمل على الانظمه الثلاث.

اتى اسمها اساساً من QtBasic و كما اخبرتكم ان Qt هي المكتبه (او اطار العمل بالاحرى) التي تعتمد عليها هذه اللغه و لان هذا الاسم ممنوع قرروا استخدام الاسم QBasic و لكن هذا الاسم اشهر من نار على علم و بالتالي قرروا استخدام الاسم Kbasic.

النسخه الكامله الوحيده هي نسخة غنو\لينكس، اما نسخة ماك و نسخة وندوز فإنها نسخ تجريبيه، لا زالت اللغه في مراحل بيتا و بنظره خاطفه عليها وجدت انها تستحق الاهتمام.

يمكنكم معرفة المزيد حول هذه المكتبه عن طريق مراجعة الموقع الرسمي : http://www.kbasic.com/

لغة البرمجه FreeBASIC

بتاريخ : 15/3/2009

لابد و ان الكثير منّا مرّت عليه لغة BASIC، بدأت هذه اللغه في عام 1964 وهي اختصار للجمله Beginner's All-purpose Symbolic Instruction Code، و كان الهدف منها تسهيل عملية البرمجه على الطلاب غير المتخصصين، ا صبحت هذه اللغه مشهورةً خلال 1970 و 1980 ولا يخفى علينا أنّ الكثيرَ من اللغاتِ الحديثةِ اعتمدت على BASIC، منها Microsoft Visual Basic و Gabmas و KBasic و غيرها الكثير، سنتحدث اليوم عن احد اللغات التي تعتمد على لغة BASIC.

لغة FreeBASIC عباره عن لغة برمجه حرّه مفتوحة المصدر تقع تحت رخصة GNU GPL و تعمل على غنو\ لينكس، وندوز و دوس، مُترجم (Compiler) لغة FreeBASIC مكتوب بـ FreeBASIC نفسها و يبلغ عدد سطوره البرمجيه 120 الف سطر برمجي، تحاول هذه اللغه ان تكون متوافقه مع لغة BASIC و تحاول ان تكون متوافقه بالتحديد مع QuickBasic كما انها تدعم مجموعه من المفاهيم الجديد من ابرزها البرمجه الكائنيه (Object-Oriented) و تقدّم دعماً للمؤشرات (Pointers)، تعمل على FreeBASIC عدّة مكتبات مثل مكتبة سي القيا سيه، جي تي كي، SDL، Allegro و Windows API، بالإضافه إلى ذلك يمكنك ا ستخدام لغة Assembly في الشيفره المصدريه بالإعتما د على طريقة Intel في كتابة لغة التجميع (Intel Syntax).

الشئ المثير الذي تقدمه هذه اللغه هو امكانية ا ستخدام المكتبات المكتوبه بلغة سي بقليل من الجهد (الذي لا يذكر ا ذا ما قارناها بلغات اخرى)، لان FreeBASIC تدعم المؤشرات بالإضافه إلى وجو د دوال يمكننا من خلالها التعامل مع الذاكره الديناميكيه وبالتالي يمكننا ا ستخدامها في كتابة بنى المعطيات (Data-Structure) مثل اللائحه المرتبطه (Linked-List) و الا شجار الثنائيه (Binary Tree) و غيرها، يمكنك مراجعة احد الدروس في الموقع الر سمي لرؤية تطبيق للائحه المتربطه بإ ستخدام FreeBASIC.

و اخيراً اترككم مع الموقع الر سمي : http://www.freebasic.net

ما هي بنى المعطيات (Data Structure) ؟

بتاريخ : 22/3/2009

سنتحدث اليوم عن مفهوم هام للمبرمجين وهو Data Structure او كما نترجمه باللغه العربيه "بنى المعطيات". ما هي بنى المعطيات؟ سنجاوب على هذا السؤال في هذه التدوينه

لا يخفى عليكم أنّ البرامجَ بشتى اشكالها تتعامل مع البيانات، بحيث تستقبل بيانات من المستخدم و تعطي بيانات للمستخدم، مثلاً برامج تصفح الويب مثل فايرفوكس يأخذ فايرفوكس عنوان الموقع المُراد زيارته من المستخدم و يقوم ببعض العمليات مستخدماً عنوان الموقع للوصول إليه ثم ينتظر بيانات الصفحه من الخادم الخاص بالموقع المطلوب و عندما يستقبل هذه البيانات يُظهرها للمستخدم، مثال آخر البرامج التي تتعامل مع قواعد البيانات تأخذ المُدخلات من المستخدم و تُخزنها ثم يطلبها المستخدم فيما بعد لعرضها او لتنفيذ عمليات عليها و معالجتها.

نستنتج من ذلك أنّ للبيانات اهميه كبيره في البرامج، ونحن كمبرمجين لابد و اننا نعلم بذلك خصوصاً انه لا يمكننا كتابة برنامج مفيد بدون استخدام المتغيرات.

لأن البيانات في قمّة الاهميه بالنسبه للبرامج لابد من ظهور طرق مختلفه لتخزين هذه البيانات، اذا كنت قد تعاملت مع اي لغة برمجه سترى اننا نتعامل مع البيانات عن طريق المتغيرات، و لكن المتغيرات البسيطه وحدها لا تكفي مثلاً لغة سي تقدّم لنا المصفوفات (Array) لماذا؟ لاننا قد نحتاج لتخزين مجموعه من البيانات ذات نوع واحد يبلغ عددها 100 مثلاً، و بالتالي من غير المعقول استخدام المتغيرات العاديه بالاضافه إلى صعوبات اخرى مثل عدم التمكن من التعامل مع المتغيرات العاديه بشكل ديناميكي لذلك استخدمنا المصفوفات، تُقدم لغة سي كذلك ما يسمى بالبُنى (Structure) وهي مجموعه من المتغيرات المختلفه في النوع تجتمع في سجل واحد.

البُنى و المصفوفات عباره عن بنى معطيات بسيطه تقدمها لغة سي (ومجموعة لغات اخرى و بعضها تستخدم مسميات مختلفه) تُستخدم هذه البُنى في تخزين البيانات مهما كان نوع برنامجنا، لنفرض اننا نريد كتابة برنامج يخزّن قائمة

الموظفين و يتعامل مع هذه القائمه، يمكننا استخدام المصفوفات بالاسلوب المتوازي او البُنى مع المصفوفات لتحقيق ذلك فكلا الاسلوبين يُخزّن البيانات و يسترجعها، و لكن ايهما افضل لبرنامجنا هذا؟ فنحن لا نريد كتابته فحسب و لكن نريد منه أنْ يعملَ بأفضل طريقة ممكنه، مثلاً اذا ادخل المستخدم اكثر من مليون موظف فلا نريد أنْ تتأثرَ السرعه كثيراً فمن غير المعقول مثلاً ان يأخذ البرنامج ساعه في البحث عن موظف معين، كذلك بالنسبه للحجم الذي ستستهلكه البيانات من الذاكره.

كلنا نعلم أنّ حجمَ المصفوفات ثابتٌ يحدده المبرمجُ اثناء كتابة الشيفره المصدريه، لنفرض ان المبرمج حدد حجم المصفوفه على انه 200 مثلاً، فماذا سنفعل في حال اراد المستخدم ادخال اكثر من 200 سجل؟ لهذه الاسباب ظهرت العديد من بنى المعطيات، لكل منها ايجابيات و سلبيات و كل واحده منها تركز على نقاط معينه مثل السرعه او سعة التخزين، نذكر منها مثلاً اللائحه المرتبطه (Linked-List) وهي احد بنى المعطيات والتي دائماً ما تُقارن مع المصفوفات فهي تتفوق على الثانيه بأن حجمها ديناميكي فلا يجب على المبرمج تحديد اقصى حد للسجلات في الشيفره المصدريه و بالتالي يمكن للمستخدم تخزين ما يشاء من البيانات (لا ننسى حدود الذاكره طبعاً) و لكن عيبها بالنسبه للمصفوفات انها ابطأ بالإضافه إلى انها تتطلب حجماً اكبر من الحجم الذي تطلبه المصفوفات، و لهذه الاسباب ظهرت تعديلات على Linked-List محاولة حل هذه المشاكل مثل Doubly Linked-List و XOR Linked-List و Unrolled Linked-List، و بالطبع هناك الكثير من بنى المعطيات الاخرى غير اللائحه المرتبطه، مثل Tree و Hash Table و عدد لا يُحصى.

لنعود إلى برنامج الموظفين، عندما يكون لدينا علم في بنى المعطيات و انواعها و ايجابيات و سلبيات كل واحد منها يمكننا تحديد ما هي بنية المعطيات المناسبه لمتطلبات برنامجنا، فإذا خُيرنا بين المصفوفات و اللائحه المرتبطه في هذا البرنامج ماذا سنختار يا ترى؟ يعود ذلك على متطلبات البرنامج، هل سيخزن الكثير من البيانات؟ من اهم سعة التخزين ام السرعه؟ و مجموعه مشابهه من هذه الاسئله يمكننا من خلالها تحديد ما هي بنية المعطيات التي تناسب برنامجنا.

قبل إنهاء التدوينه لابد من التحدث عن نقطه ، يمكننا تقسيم بنى المعطيات إلى قسمين، القسم الاول ثابت الحجم Fixed-size لابد من تحديد حجمه من خلال الشيفره المصدريه و اقرب مثال المصفوفات و غالباً ما تُقدم لغات البرمجه هذا النوع كميزات في اللغه و يُستخدم المُكدّس Stack لتخزين هذا القسم، القسم الثاني متغير الحجم او لنسميه القسم الديناميكي Dynamic وهو غير ثابت الحجم و الذي يتم حجز قطع من الذاكره من اجله خلال وقت تشغيل البرنامج run-time و اقرب مثال اللائحه المرتبطه ويُستخدم الـ Heap لتخزين هذا القسم.

اذاً نستنتج من هذه التدوينه ان بنى المعطيات عباره عن طريقه لتخزين البيانات في الحاسوب (في الذاكره الرئيسيه RAM غالباً)، هناك العديد من بنى المعطيات لان حاجتنا تختلف من تطبيق إلى آخر فهناك بنى معطيات تُركّز على توفير مساحة التخزين و اخرى تُركّز على سرعة البحث و اخرى تُركز على سرعة الحصول على المعلومه و هكذا، و نستخدم غالباً ما يناسب تطبيقنا من بنى المعطيات.