## ثالثاً: نداءات النظام(System Calls )[4]

### تعريفها:

نداءات النظام هي ميكانيكية تستخدمها برامج التطبيقات للحصول على خدمة يقوم بها نظام التشغيل. أو هي الطريقة التي يستخدمها (عملية المستخدم) ليسأل نظام التشغيل لفعل شيء معين.

### متى تحدث ؟

تحدث نداءات النظام في وقت معالجة برامج التشغيل في الذاكرة حيث تحتاج إلى خدمات نظام التشغيل, مثل استخدام الأجهزة الملحقة بالنظام كبطاقة الشبكة أو بطاقة الصوت أو بطاقة الرسومات أو في الاتصالات بين البرامج التطبيقية.

عندما تُستخدم نداءات النظام, فإنه ببساطة يتم تحديد اسم الدالة المطلوبة ومناداتها, ولكن ما العمل عند الحاجة إلى معلومات إضافية؟ في هذه الحالة يتم إرسال هذه البيانات الإضافية عن طريق معاملات(parameters). وهناك عدة طرق تستخدم لإرسال المعاملات وهي:

1. إرسال المعاملات إلى النظام عن طريق وضعها في أحد المسجلات(register). هذه الطريقة سريعة ولكنها تُفضل فقط عندما يكون لدينا عدد قليل من المعاملات وذلك لأنه هناك عدد محدود من المسجلات داخل المعالج.

2. استخدام تركيب بيانات من نوع stack , بحيث يقوم البرنامج بدفع المعاملات داخلها ومن ثم يقوم نظام التشغيل باستخراجها. هذه الطريقة لا تحددنا في كمية البيانات المخزنة.

3. تخزين المعاملات في مكان محدد في الذاكرة(block) أو توضع في جدول في الذاكرة(table), ثم بعد ذلك يوضع عنوان المكان أو الجدول في مسجل(register) ويمرر هذا العنوان إلى نظام التشغيل. نستطيع القول أن هذه الطريقة تعتبر من أفضل الطرق الثلاث وذلك لأنها لا تحددنا في كمية المعلومات المخزنة , بالإضافة إلى أن النظام يستطيع الوصول إلى أي معلومة بسهولة على

---

[4] إعداد: منى البريه، خديجة أخرفي، قطر الندى عبدالرحمن السماعيل، نهى الطياش، نوف السفياني

عكس الطريقة السابقة, حيث إذا أراد النظام معلومة في أسفل الـــstack فسوف يضطر لإخراج جميع المعلومات التي تقع فوقها!

**يمكن تصنيف نداءات النظام إلى هذه الأنواع:**

1. **أعمال الملفات**: خلق / حذف/ فتح ملف ، قراءة / كتابة
2. **إدارة الأجهزة** : طلب / تحرير ، قراءة / كتابة
3. **صيانة المعلومات** : طلب /أخذ المعلومات ، معرفة الوقت و التاريخ وعملية الحصول على المعلومات
4. **التواصل** : خلق / حذف الروابط ، وإرسال / استقبال الرسائل
5. **التحكم في العمليات**.

## الأوامر البرمجية (API)application program interface :

لكل نظام تشغيل مجموعة من الأوامر البرمجية(API's), التي تقوم بمناداة نداءات النظام ( system call) في قلب النظام (kernel mode) ثم تنتقل إلى نظام التشغيل. فمثلاً عند عمل الأمر البرمجي –**open()**– فإنه يستدعي نداء النظام لهذا الأمر:

( open () >>>>>open system call) .

وليس من الضروري أن يعادِل الأمر البرمجي الواحد نداء واحد للنظام! فقد يتطلب المئات من نداءات النظام.

الصورة التالية توضح لنا المفهوم العام لعلاقة الأوامر البرمجية مع نداءات النظام:

API & System Calls

**من أكثر أنواع الأوامر البرمجية شيوعاً:**

1. Win32 API التي تستخدم في نظام الويندوز
2. POSIX API التي تستخدم في أنظمة UNIX و Linux و Mac OS X
3. JAVA API المستخدمة في الآلة الوهمية للغة الجافا.

**إن من الجدير بالذكر أنه يُفَضَّل استخدام الأوامر البرمجية بدلاً من نداءات النظام, وذلك للأسباب التالية:**

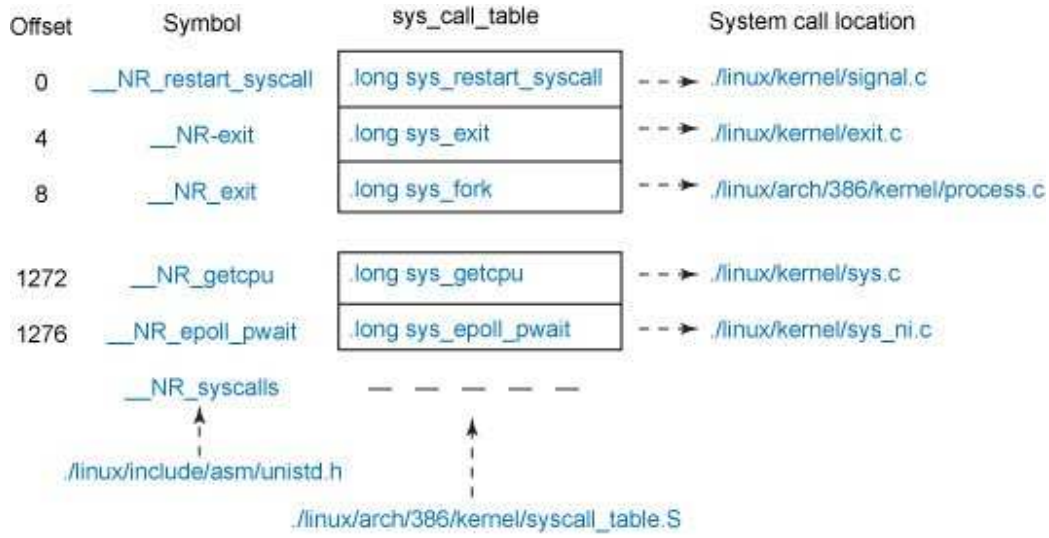
1. قابلية النقل(Portability) في الأوامر البرمجية.

فمثلا: عند كتابة برنامج بلغة الجافا فإننا نستطيع تشغيله في أي نظام تشغيل مثل Windows وLinuxوMac بدون أي تغيير في أوامر نداء النظام.

2. أوامر استدعاء النظام أكثر تفصيلاً وتعقيداً ويصعب التعامل معها وتختلف من نظام لنظام, بينما الأوامر البرمجية أسهل وأقل تعقيداً.

**الآن نعود إلى كيفية تنفيذ نداءات النظام داخل نظام التشغيل (system calls implementation of):**

يتم ربط الأمر البرمجي (API) برقم (index) وهذا الرقم يُربط بأمر نداء النظام , وتلك الأرقام تكون مدونة في جدول يسمى جدول النظام(system table) , ولكل نداء للنظام رقم (index) .
الشكل التالي يوضح جدول النظام حيث نلاحظ فيه كل أمر نظام مرتبط برقم(index):




مثلاً عند عمل الأمر البرمجي –**open()**– , سوف يتم الانتقال من أسلوب المستخدم إلى أسلوب لب النظام, وبالتالي لابد من استخدام نداء النظام ,ويتم ذلك بعمل مناداة للرقم المرتبط بالنداء الذي يتم إيجاده من خلال جدول النظام (system table) كما هو موضح في الشكل التالي:

**المصادر:**


- Operating system Concepts (Seventh Edition): Abraham Silberschatz,Peter Baer Galvin,Greg Gagne
- http://en.wikipedia.org/wiki/System_call
- http://data.uta.edu/~ramesh/cse3320
- http://tiger.la.asu.edu/Quick_Ref/Linux_Syscall_quickref.pdf
- http://www.slideshare.net/guestd1b5cb/adding-a-system-call

## رابعاً: برامج النظام[5](System Programs)

هي مجموعة برامج توفر بيئة تخاطبية بين نظام التشغيل والبرامج المطوَّرة من قبل المستخدمين ومطوري البرامج, وأكثر المستخدمين يتعاملون مع نظام التشغيل عن طريق برامج النظام وليس عن طريق الاتصال المباشر بنظم التشغيل.

### أنواع برامج النظام:

تقسم برامج النظام إلى عدة أقسام وهي:

- **إدارة الملفات:** وهي المسئولة عن خلق, حذف, إعادة تسمية, نسخ وغيرها من العمليات على الملفات والأدلة.
- **معلومات حالة النظام:** هي برامج تسأل النظام عن الوقت, التاريخ, حجم الذاكرة, عدد المستخدمين.
- **تعديل الملفات:** وهي عبارة عن مجموعة من محررات النصوص لعمل تغيرات في محتويات الملفات.
- **دعم ملفات البرمجة:** وهي المسئولة عن التجميع في برامج لغات البرمجة.
- **تنفيذ وتحميل البرامج:** وهي المسئولة عن تنفيذ البرامج بعد تحميلها.
- **الاتصالات:** وهي المسئولة عن التواصل بين العمليات أو بين المستخدمين أو بين أجهزة أخرى مختلفة.

### المصدر:


- Operating system Concepts (Seventh Edition): Abraham Silberschatz,Peter Baer Galvin,Greg Gagne

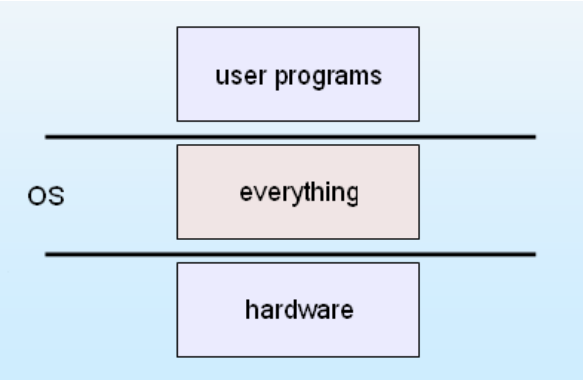

[5]إعداد: أمل الصبيح

## خامساً: تركيب نظم التشغيل(Operating Systems Structure)[6]

هناك عدة طرق لبناء وتركيب نظم التشغيل و هي:

### 1. التركيب البسيط (Monolithic) :

بحيث يكون نظام التشغيل في مستوى واحد أو في مستويين.




**الميزة الرئيسية** : تكلفة التفاعلات الداخلية في النظام تكون منخفضة لأنها جميعا تقع على نفس المستوى.

**العيوب:**

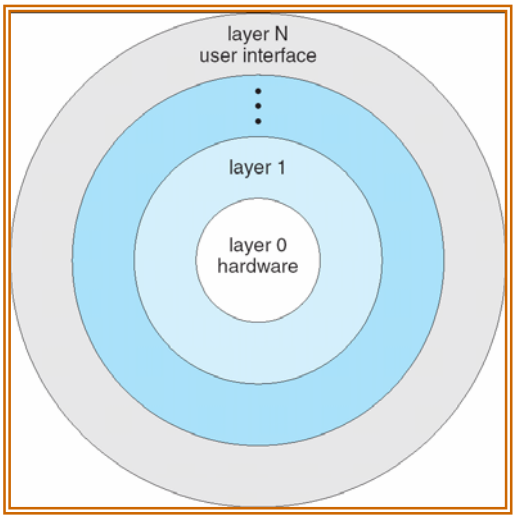
- صعوبة الفهم .
- صعوبة التعديل .
- صعوبة الصيانة .
- غير موثوق فيه .

### 2. تركيب الطبقات (Layered) :

أي أن نظام التشغيل مقسم لطبقات (مستويات) بحيث يكون كل جزء من النظام في طبقة مستقلة ، و بحيث أن الطبقة 0 (layer 0) مخصصة للعتاد (Hardware) ، والطبقة **ن** (layer N) مخصصة لواجهة المستخدم, كما هو موضح في الشكل التالي:

---

[6] إعداد: سمية الخنيزان




**الميزة الرئيسية** : وجود الطبقات أدّى إلى تسهيل عملية الصيانة .

**العيوب**: المشكلة تكمن في عملية ترتيب الطبقات ، فلا توجد لدينا طريقة واضحة للترتيب .

### 3. تركيب النواة الصغيرة ( Microkernel) :

تكون نواة النظام في هذا التركيب صغيرة جداً ، ولا يوضع بداخلها سوى الوظائف الأساسية . أما الوظائف الأخرى فتوضع في مساحة المستخدم ، ويكون الاتصال بين مساحة المستخدم والـــنواة عن طريق الـــرسائل العابرة (message passing).

**الميزات** :

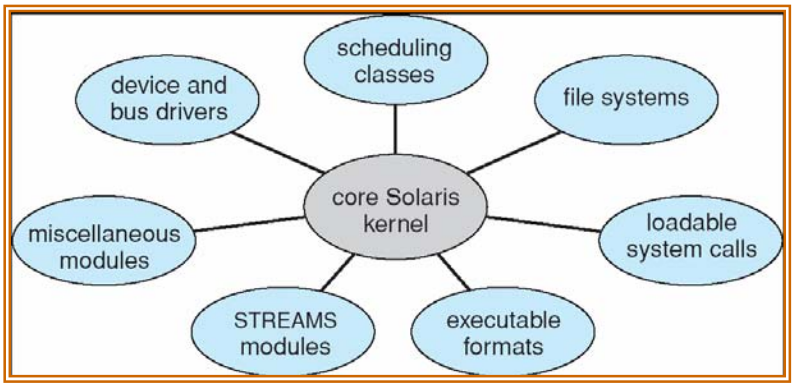
- من السهل توسيع (تمديد) النظام.
- النظام أكثر ثقة و أكثر أمناً.

**العيوب** :

الاتصال بين مساحة المستخدم ونواة النظام عملية مكلفة.

### 4. تركيب الوحدات (Modules-based) :

معظم أنظمة التشغيل الحديثة مبنية بهذه الطريقة ، حيث تكون النواة الأساسية في المركز وبقية الوظائف تتفرع منها ، وهي مشابهة للطبقات ولكن أكثر مرونة وأكثر كفاءة.




المصدر :


- Operating system Concepts (Seventh Edition): Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

## سادساً: الآلات الافتراضية أوالتخيلية(Virtual Machines[7])

### ما هي الآلة الافتراضية؟

الآلة الافتراضية هي عبارة عن برنامج يسمح بتشغيل أكثر من نظام تشغيل تخيلي افتراضي على جهاز شخصي واحد. بحيث يمكن تثبيت أكثر من نظام تشغيل على نفس الجهاز والتنقل بين هذه الأنظمة دون المساس بالنظام الحالي ودون خسارة كبيرة في الأداء. و تعمل الآلة الافتراضية كتطبيق على نظام التشغيل المضيف.

### خصائصها:

- تمكن من استخدام أنظمة تشغيل متعددة على جهاز واحد والتنقل بينها دون الحاجة لإعادة تشغيله .
- تمكن من تركيب أنظمة تشغيل متعددة دون تقسيم جديد للقرص الصلب.
- توفر الاتصال بين أنظمة تشغيل متعددة على جهاز شخصي واحد.

### فوائـــدها:

- إمكانية تجربة أنظمة تشغيل متعددة بأقل التكاليف.
- إمكانية إجراء تعديلات وتجارب على النظام التخيلي بحرية وذلك لأن الموارد التي يستخدمها معزولة تماماً عن موارد النظام الأساسي.
- يساعد على توفير بيئة برمجية جيدة , مما يتيح لمُطوّري أنظمة التشغيل القيام بالتجارب والبحوث على الآلة الوهمية بدلاً من القيام بها على النظام الأساسي وبالتالي لا يؤثر على أداء هذا النظام.

### مثال على الآلات الافتراضية:

آلة جافا الوهمية (JAVA Virtual Machine), التي تمكن ملفات الجافا من العمل على جميع أو معظم أنظمة التشغيل.

---

[7]إعداد: حليمة حكمي, نوف السفياني, نورة الخالدي