بسم الله الرحمن الرحيم

محاضرات فى نظم التشغيل (الجزء الأول) مقرر علوم + تقانة

نظم التشغيل OS) Operating Systems)

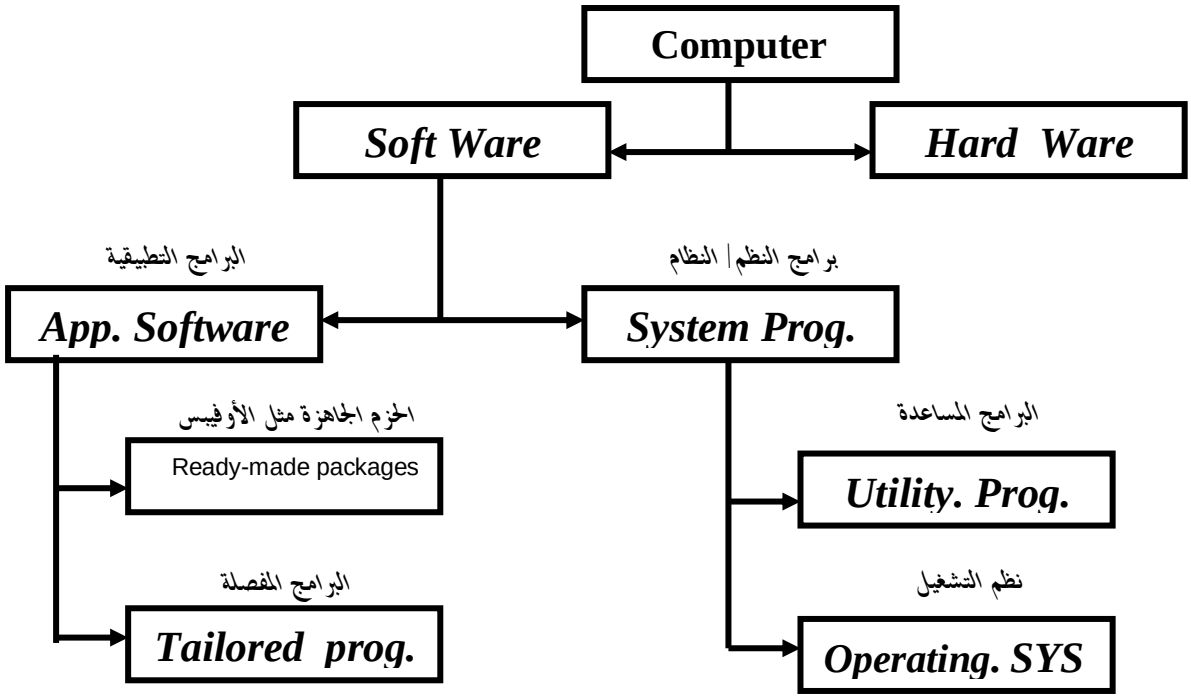
تعريف

- نظام التشغيل هو برنامج أو مجموعة برامج تقوم بإدارة معدات الحاسوب المختلفة .

أو : هو وسيط بين المستخدم وجهاز الحاسوب يمكن المستخدم من تنفيذ البرامج بكل سهولة و كفاءة .

- تم تصميم نظم التشغيل مع الجيل الثالث للحواسيب عام 1964م وذلك للإستفادة من الوقت الضائع من وحدة المعالجة المركزية عند إنتظار المشغل لتشغيل العملية التالية .

س| ما الهدف الأساسى من نظام التشغيل ؟ زيادة الإنتاجية أو رفع كفاءة تشغيل الحاسوب .



البرامج المكونة لنظام التشغيل:

(1) البرنامج المشرف (Supervisor) ويهتم بالإشراف على برامج نظام التشغيل .

(2) برامج التحكم (Controller) ويهتم بالتحكم فى سير العمليات أوهى التى يتعامل معها المشغل فى تنظيم وتشغيل العمليات وتنفيذالبرامج.

(3) برامج ترجمة اللغات (Compiler) ويهتم بعملية الترجمة إلى لغة الآلة (Machine Language) .

(4) برامج الصيانة (maintenance) ويهتم بصيانة نظم التشغيل .

وظائف أو أنشطة نظم التشغيل :

(1) إدارة معدات الحاسوب المختلفة .

(2) حماية البرامج والبيانات من الأخطاء المقصودة وغير المقصودة من قبل المستخدمين .

(3) إختيار البرنامج المناسب فى كل عملية جديدة كمثال مكتبة البرنامج المقيم (Resident program Library)

توضيح : فلكل عملية برنامج مناسب للتشغيل وهذه البرامج موجودة بهذه المكتبة كمثال جمع عددين .

(4) الإنتقال من عملية تم تنفيذها إلى أخرى فى إنتظار التنفيذ . وهو الهدف من نظام التشغيل (زيادة الإنتاجية) .

(5) التحكم فى إدارة وتوزيع الذاكرة .(Memory Management)

(6) التفاعل بين المستخدم والحاسوب (تنظيم إسلوب المواجهة) كمثال الشاشة التالية :

ويمكن تلخيص هذه الوظائف كالآتي :

(1) آلة ظاهرية (تخيلية | أفتراضية) Virtual Machine كمثال قراءة أو كتابة فلابد من وجود عدد من المعاملات (البارامترات) لتحديد الاشياء.

توضيح : من هذه المعاملات : أ- عنوان كتلة القرص المرن المراد قراءتها . ب- عدد القطاعات فى المسار ..... أخ

هذه التعقيدات لا يحتاج إليها المبرمج ويحتوى القرص على مجموعة من الملفات فيحتـــاج المبرمج إلى طريقة يفتح بها الملف لإمكانية القراءة منه أو الكتابة عليه وإعادته بعد الإنتهاء من عمليتى القراءة أو الكتابة للقرص الصلب فلا يراها المستخدم إذ يمكن القول أن البرامج التى تخفى حقيقة المكونات المادية عن المبرمج أو المستخدم وتقدم له شكلاً بسيطاً وجميلاً فى التعامل مع الملفات وإمكانية الكتابة او القراءة منها هى نظام التشغيل وبالتالى يعتبر آلة ظاهرية ونظام التشغيل يقدم خدمات تستطيع البرامج الحصـــول عليها بإستخدام تعليمات خاصة تسمى إستدعاءات أو نداءات النظام (System Calls) .

(2) إدارة المصادر أو الموارد Resource Manager . مثل الطابعة وتنقسم إلى :

- برامج (Software) : مثل الـ (process) أو (files) أو (Records) .
- أجهزة (Hardware) : مثل الـ (Memory) أو (Printers)

أمثلة لنظم تشغيل : ويندوز | يونكس | دوس | ماكنتوش وغيرها من النظم

طبقات نظام الحاسوب (Structures of Computers) موقع نظام التشغيل وسط الحاسوب لإزالة التعقيد

تسمى App. Prog →	Games (الألعاب)	Airline Reservation (مكاتب الحجز)	Banking System (نظام مصرفى)
تسمى SYS. Prog →	Compiler (المترجم)	Editors (المحرر)	Interpreter (المفسر)
	Operating System (نظام التشغيل)		
	Machine Language (لغة الآلة)		
تسمى Primitive S.W →	Micro Programming (البرامج | التعليمات) مثل Add		
	Physical Devices (الأجهزة الفيزيائية الدقيقة) مثل Chips و مزود الطاقة		

ورقة بحثية (1):

س| كيف كان يتم التشغيل قبل تصميم نظم التشغيل ؟

س| ناقش : لا وجود للحاسوب دون وجود نظام تشغيل ؟

س| تحدث عن تطور نظم التشغيل ؟ نظام التشغيل مر بتاريخ تطوير طويل يلخص في الآتي :

1| محتويات [أخف] Simple Batched Systems 2| أنظمة البرمجة المشتركة الخفيفة (Multiprogramming Batched Systems)

3| أنظمة المشاركة الزمنية (Time-Sharing Systems) 4| أنظمة الحواسيب الشخصية (Personal Computer Systems)

5| الأنظمة المتوازية (Parallel Systems)

6| الأنظمة الموزعة (Distributed Systems)

7| أنظمة الزمن الحقيقى (Real Time Systems)

س| لماذا تتعدد أنظمة التشغيل ؟ لوجود أنواع متعددة من المعالجات.

أنواع نظم التشغيل :

تقسم نظم التشغيل إلى أنواع حسب المهام (Tasks)والمستخدمين (Users) .

حسب المهام :

(1) مفرد المهام (*Single Tasking*) : وهو الذى يتعامل مع مهمة واحدة فى لوقت .

(2) متعدد المهام (*Multi Tasking*) : وهو الذى يتعامل مع أكثر من مهمة فى نفس الوقت .

حسب المستخدمين :

(1) مفرد المستخدم (*Single Users*) : وهو الذى يتعامل مع مستخدم واحد فى الوقت .(دوس)

(2) متعدد المستخدم (*Multi Users*) : وهو الذى يتعامل مع أكثر من مستخدم فى نفس الوقت .(ويندوز)

وبالتالى يمكن أن يكون لدينا أربعة أنواع من الأنظمة كالآتى :

(أ) نظام : مستخدم واحد ومتعدد المهام

(ب) نظام : مستخدم واحد ومفرد المهام

(ج) نظام : متعدد المستخدمين ومفرد المهام

(د) نظام : متعدد المستخدمين ومتعدد المهام

تشغيل نظام التشغيل :

بما انه برنامج أو مجموعة برامج فإنه يخزن فى ملف ويتم نقله إلى الذاكرة ويبقى فيها ليشرف على بقية البرامج الأخرى وعلى وحدات التخزين ويتم ذلك عند التشغيل فبمجرد الضغط على زر تشغيل الحاسوب يعمل برنامج صغير يوجد فى ذاكرة القراءة فقط(*Rom*) يسمى *IPL* حيث يقوم بفحص معدات الحاسوب ويتأكد من سلامتها ومن ثم يقوم بتحميل نظام التشغيل من القرص وبعدها يقوم نظام التشغيل بإستلام أوامر المستخدم وتنفيذها .

برنامج (IPL): initial program lood

– هو برنامج يقوم بتحميل نظام التشغيل من الـ *HD* إلى الذاكرة حيث يقوم بالوظائف التالية :

(1) يقوم بتنفيذ المهام المحددة . (2) البحث عن معدات الحاسوب والتأكد من سلامتها .

محاضرة رقم (2)

مفاهيم خاصة بنظم التشغيل : O. S . Concepts

– يعتبر مصممى نظام تشغيل يونكس أن نظام التشغيل يتكون من ثلاثة مفاهيم هى :

(1) النواة *Kernel* (2) القشرية *Shell* (3) نظام الملفات *Files System*

– بينما يعتبر مصممون آخرون غير مجموعة اليونكس المفاهيم التالية إضافة للمفاهيم السابقة :

* العملية أو المهمة *process / job*
* نداء النظام *System Call*
* المقاطعة *Interrupt*

النواة Kernel :

هى عبارة عن مجموعة الوظائف ذات المستوى الأدنى لنظام التشغيل والتى تحمل الى الذاكرة كلما قمت بتشغيل الجهاز وذلك مباشرة بعد أن تعمل بعض الوظائف الموجودة فى الـ (Bios) وهو نوع من أنواع برامج النظام .

User program
Sys call
Kernel
Operating system
hardware

القشرية shell

هي الواجهة المرئية ما بين المستخدم ونظام التشغيل وهي عبارة عن برنامج يعمل في الطبقة العليا منه ويسمح للمستخدمين بإصدار الأوامر إليه .

تعريف آخر : لنظام دوس القشرية هي البرنامج الموجود في ملف Command . com وبالتالي هي :

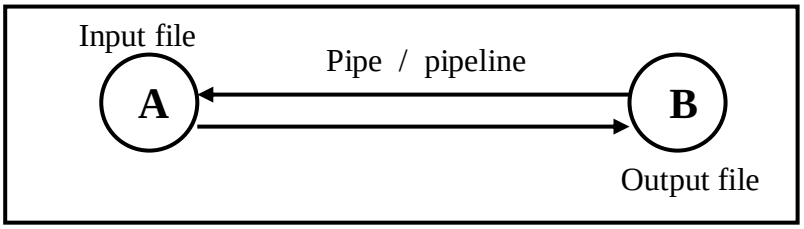
عبارة عن مفسر أوامر .shell is a command interpreter

- هنالك نظم تشغيل بها عدد من القشريات مثلاً نظام يونكس كما أنه يقوم بإظهار الـ Shell . وكذلك الكثير من أنظمة التشغيل .
- هنالك أشخاص يعتبرون إنها scripting language لغة برمجة غير متاحة في مستوى اللغات العليا (HLL) ومفرداتها لا توجد في هذه اللغات عدا بعض المفردات في لغة السي مثل END والبرامج التي تكتب بلغة الـ Shell تسمى الـ Shell scripts .وهي التي تمكن المبرمج من كتابة System Programming .
- عندما يحتاج المبرمج تصميم System Programming يستخدم أوامر الـ Shell .
- في نظام دوس هنالك أوامر تكتب في الـ Shell .
- عندما نكتب الأمر توجد process في نظام التشغيل تسمى أيضاً Shell تأخذ الأمر وتقوم بتنفيذه . وهنالك أوامر تكتب في الـ Shell يقوم بتحميلها إلى الـ **O .S** وهنالك بعض الأوامر لا يستطيع الـ Shell تنفيذها وتنفذ بواسطة برامج نظام **O .S** الأخرى .
- أعتبرت الـ Shell جزء من الـ Kernel في نظم التشغيل القديمة أما النظم الحديثة فصلتهما عن بعضهما تماماً .

المهمة / العملية Process

- هي برنامج تحت التنفيذ Program in execution & Program group of instruction
- البرنامج من الـ H D إلى الـ Memory لا يصبح Program ولكن Process .
- في كل عملية عندما ينفذ البرنامج يكون معه ما يسمى بالـ process address يوجد به عداد البرنامج (prog . counter) والمكدس (stack) ومؤشر المكدس (stack pointer) ..... إلخ كل هذا يسمى Process .
- يمكن أن يكون للعملية الواحدة مجموعة عمليات فرعية وبالتالي يكون لدينا شجرة عمليات (tree) وتسمى Process Hierarchical ويمكن أن تتصل مع بعضها وتسمى Inter process communication ويتم ذلك بوجود قناة إتصال تسمى الـ pipe أو الـ Pipe line وهو عبارة عن virtual file .

مثال : إذا كان لدينا عملية (A) وعملية (B) وعند طلب معلومة معينة مثلاً طلبت (B) من (A) معلومة عبر قناة الإتصال فإن (A) تعتبر بمثابة output file و (B) تعتبر input file وهكذا وتتم عملية الإتصال هذه عبر الـ virtual file ويمكن توضيح ذلك بالرسم التالي :



توضيح بصورة موسعة :

يصبح الـ pipeline إفتراضي virtual (pipe is a virtual file) بمعنى وقتي يكون عند الحوجة ثم يحدث له (terminate) فتكون هنالك حالتان كما موضح أعلاه وهما :

(1) process B wants info from process A
(2) process A wants info from process B

ففي الحالة الأولى فإن (A) تنظر للـ pipe على أنه output file بالنسبة لها وتطرح المعلومة التي تريدها (B) في الـ pipe و(B) تنظر للـ pipe على أنه input file وبالتالي ستقرأ هذا الملف وستجد المعلومة التي طلبتها من (B) . والحالة الثانية العكس .

نداء النظام system call :

هي مجموعة من الأوامر (واجهة ما بين التطبيقات والنظام) يتم فهمها في نظام التشغيل .

- هي أمر واحد وهو غير متاح على مستوى اللغات العليا إلا لغة السي حيث تسمح باستخدام بعض مفردات الـ sys call.
- نداءات النظام مصممة خصيصاً للمبرمجين فالمستخدمين لا علاقة لهم بها .

أنواع نداءات النظام : تضم خمسة مجموعات من الأوامر هي :

(1) أوامر العمليات process and job sys call (control) كمثال end & abort & load &execute &create & delete

(2) أوامر المعاملات | التعديل للملف file manipulation كمثال close & create & delete &open

(3) أوامر الأجهزة devices manipulation كمثال إضافة جهاز جديد أو هي نفس الـ file manipulation .

(4) أوامر الإتصال communication كمثال أوامر الضغط على أزرار لوحة المفاتيح . أو send & receive & delete links

(5) أوامر معلومات الصيانة information maintenance وهي خاصة بالنظام مثل time & cpu usage & memory

* ملحوظة : تحدث المقاطعة عند تنفيذ إي عملية أو تنفيذ أى من هذه الأوامر .

المقاطعة interrupt عبارة عن حدث يغير من تسلسل تنفيذ الأوامر في المعالج كمثال إدخال الفلاش فجأة في الحاسوب .

للمقاطعة عدة أسباب منها : إنتهاء الزمن المخصص للعملية .

أنواع المقاطعات:

(1) المقاطعة نتيجة إستدعاء المشرف : هى تقنية تساعد فى الحفاظ على أمن نظام التشغيل من المستخدمين حيث أن المستخدم يجب أن يطلب الخدمة من خلال إستدعاء المشرف وبالتالى فإن نظام التشغيل يرفض طلب المستخدم إذا كان المستخدم لا يملك الصلاحية المناسبة كمثال حذف| تغيير كلمة السر .

(2) مقاطعات الإدخال والإخراج I / O interrupts : تحدث عندما تكتمل عملية إدخال أو إخراج بيانات من وحدات الإدخال والإخراج (الأجهزة) أو حدث خطأ فى الإدخال والإخراج أو عند تجهيز جهاز معين .

(3) المقاطعات الخارجية External interrupts أو software interrupt وهي زمن معين لتنفيذ برنامج معين إذا لم ينتهي التنفيذ يتم إيقافه كمثال مقاطعة الطباعة . توضيح عندما تتم طباعة ملف معين وفجأة ينتهي الورق مثلاً . أو مثل إنتهاء الـ Quantum على الساعة وتسبب مقاطعة .

* ملحوظة : من الأسباب التي تجعل البرامج لا تنتهي حسب الزمن المعين دخولها في دوار لا نهائي وهو حدوث خطأ في برنامج معين .

(4) مقاطعة الإستئناف resume interrupts تقوم المقاطعة بإستئناف العمل عندما يصبح في الحالة ready . وهى تحدث عندما يضغط المشغل على زر الإستئناف للوحة التحكم أو عند وصول أمر إستئناف معالج بالإشارة إليه من معالج آخر فى نظام المعالجات المتعددة .

(5) مقاطعات تدقيق البرامج : تحدث لأحد الأسباب الآتية :

أ- مشاكل القسمة على الصفر . ب- محاولة تنفيذ شفرة لعملية غير صحيحة ج- محاولة الرجوع لموقع فى ذاكرة غير موجودة

(6) مقاطعات حدوث الخلل (تدقيق الآلة) machine accuracy interrupts : تحدث نتيجة لحدوث خلل فى جهاز الحاسوب .

* ملحوظة : توجد مع كل مقاطعة ما يسمى (ISR) .

الـ Interrupt Service Routine (ISR) : هي شفرة تحدد الحدث المطلوب مع كل نوع مقاطعة .

- عند وصول المقاطعة لوحدة المعالجة المركزية تقوم المقاطعة بتحويل التحكم من الـ cpu إلى الـ ISR .
- أثناء تنفيذ الـ cpu للمقاطعة إذا حدثت مقاطعة أخرى تتوقف وتتعامل مع الأمر بطريقتين هما :

الأولى priority scheme الأولوية للأهم والثانية Enable / disable السماح وعدم السماح .

- تتمكن وحدة المعالجة المركزية من معرفة إنتهاء الـ ISR باستعمال حالة البرق Status flag

ورقة بحثية (2) أذكر خمسة من مقاطعات الأجهزة وخمسة مقاطعات للبرامج ؟

س| هل يمكن إعادة المقاطعة (حدوث أكثر من مقاطعة في نفس الوقت) ؟

يتم ذلك ب طريقتين هما : الأولى priority scheme الأولوية للأهم والثانية Enable / disable السماح وعدم السماح .

مقاييس الحماية protection measures

س| كيفية حماية برامج نظم التشغيل من المستخدم أو نظام التشغيل نفسه ؟

يتم ذلك بواسطة مقاييس الحماية وذلك لمنع أي تداخل بين برامج النظام والمستخدم . ويتم ذلك بأربعة طرق هي :

(1) Dual mode Operation

يجب أن يكون لكل CPU طريقة لمعرفة نوع العملية هل هي (O.S أم User) ويتم ذلك بواسطة الـ flag وهو عبارة عن bit إضافية للتمييز وتسمى الـ mode bit وهي خانة يتم إضافتها للبرنامج لمعرفة هل هو O.S أم User وإذا كانت الخانة واحد هو User وإذا كانت صفر هو O.S فمثلاً برامج نظام التشغيل تسمى الـ Monitor mode و المستخدم تسمى الـ User mode والبرامج والتعليمات الخاصة بالنظام تسمى الـ Privilege instruction كمثال الـ timer .

(2) **I / O protection**

عند كتابة برنامج وهو يحتوي على I / O دائماً يوجد في الـــ Monitor mode لذلك فإن كل أوامر الـــ I / O موجودة فيها وذلك لأن المستخدم لا يستطيع أن يحدد مكان حفظ البرنامج أو الشفرة تحديداً في الذاكرة .

(3) **Memory protection** :

هي الجهة المسئولة عن تحديد المواقع في الذاكرة والمسموح بتنفيذها فقط عن طريق نظام التشغيل مثلاً table of ISR في الذاكرة لا بد أن يكون نظام التشغيل على معرفة تامة به لمنع برامج المستخدمين من الدخول له في الذاكرة .

– تحدد مساحة الذاكرة من خلال الـــ Register من خلال الـــ limit & Base مساحة البرنامج داخل الذاكرة تساوى حاصل طرحهما

كمثال : limit = 20 & Base = 10 إذا مساحة البرنامج تساوى 10 .

(4) **CPU protection** :

يتم ذلك بجعل الـــ cpu تقوم بتنفيذ العمليات الطبيعية فقط (العمل المخصص لها) وليس الغير مرغوب فيها كالدوار اللانهائي (infinite loop) .

ولحل هذه المشكلة ظهر مفهوم الـــ Timer . (privilege instruction)

تم بحمد الله

محاضرة رقم (3)

هيكلة نظم التشغيل : O . S Structure

هنالك عدة جوانب يجب على المصمم مراعاتها قبل تصميم النظام تتمثل في :

(1) طبيعة نظام التشغيل : حسب نوعه multi أم Single أم غيره

(2) الهدف من النظام : هل يهدف للمستخدم أم للنظام .

(3) تصميم النظام: كمثال حسب طبيعة البيانات وحجمها وهذه الناحية مرتبطة بالـــ Modules الموجودة وكل العمليات المتعلقة بها من حماية وغيره.

أنواع بنائيات تصميم النظام :

(أ) Virtual machine (ب) Client - server system

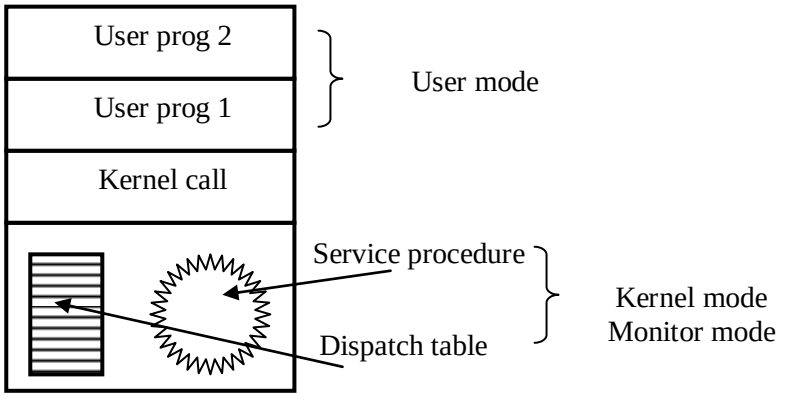
(ج) Monolithic system (د) Layered system

وما يهمنا الآن هو النوع (ج ، د) فقط .

الـــ Monolithic system :

تعريف : هو عبارة عن مجموعة إجراءات مرتبطة ببعضها لتنفيذ العمل لوجود بنية منظمة لهذا النظام . وفي هذا النوع من نظم التشغيل يكتب نظام التشغيل كمجموعة من الإجراءات (Procedures) . (sub system)

بمعنى أنه ليس لديه هيكل وتتم كتابة الـ O . S في شكل عدد من الإجراءات كل إجراء يتصل بإجراءات اخرى ويتم تجميل كل الإجراءات في ملف واحد لتكون برنامج software يعرف بنظام التشغيل .



لكل نداء نظام وسائط وكل وسائط النظام تخزين في الـــ Dispatch table كمثال ( ) delete .

* كل الـــ System call يشير لها الـــ Programm بواسطة Parameter معين يتم تخزينها في وسط معين يعرف بالـــ Dispatch table

وبالتالي الـــ O /S يقرأ الـــ System call من خلال الـــ Parameter . الموجودة في الـــ Dispatch table .

-User mode الـــCPU تنفذ الـ Instruction التي تخص الـــUser .

– Kernel mode الـCPU تنفذ الـــService المقدمة من الـــ S.O .

– في الـــUser mode تكون السيطرة للـــProgram في التمييز وبما أن الـــلا User program يطلب خدمة إذا الكنترول يتحول من الـ Program إلى الـ O. S وهذا يتم عبر الـــ Kernel call أو Supervisor call .

* الـ Dispatch table : هي منطقة تخزن فيها جميع وسائط النظام للنداء .

* الـ Kernel call : هي منطقة تقوم بنقل التحكم من منطقة الـ User Mode إلى الـــ Kernel call وتسمى أيضاً supervisor call

طريقة العمل : يقوم التحكم في البدء في التواجد في الـــ User Mode وعند كتابة نداء مثلاً الأمر () delete تكتب معه الوسائط وبعدها ينتقل التحكم إلى الـــ Dispatch table لمعرفة الوسيط المطلوب ومنه ينتقل البرنامج من الـــ User Mode إلى الـــ Monitor ويتم ذلك بواسطة الـــ Kernel call وبعدها ينتقل إلى الـــ Service وتبدأ الدورة من جديد هكذا .

ملحوظة : في بعض الأحيان لا تنفذ كل الخدمات عبر الـــ Service ماذا يفعل النظام عندها ؟

يقوم بإستدعاء البرامج المساعدة (Utility Program)

لمعلومات أكثر يرجى مراجعة المرجع بالمكتبة !!!!!!!!

الـــ Layered system :

هو نظام في شكل طبقات أو مستويات

Level 5	***Operators***
Level 4	***User program***
Level 3	***I / O management***
Level 2	***Operators process communication***
Level 1	***Memory management***
Level 0	***Process allocation & multi programming***

level 0 : يتعامل مع الـــ ***Process allocation & multi programming*** الموجودة في الذاكرة ومع الـ ***Cup*** في التحويل بين العمليات عند حدوث المقاطعات أو عند إنتهاء الـــ ***timer*** .

level 1 : مسئول من عملية تحويل البرامج من القرص الصلب إلى الذاكرة الرئيسية أو العكس .

level 2 : مسئول من عملية الإتصال بين اى process و الـــ ***operator console*** .

level 3 : مسئول عن إدارة أدوات الإدخال والإخراج .

level 4 : مسئول عن التحكم في برامج المستخدمين .

level 5 : مرتبط مع الـ System Operator حيث أن كل عمليات الـ System Operator موجودة في هذا المستوى .

* ملحوظة : الفرق بين الـ Monolithic & الـ Layered هو أن الأخير (نظام الطبقات) نظام واضح .

إدارة العمليات : Process Management

هي الإنتقال من نشاط إلى آخر حسب المقاطعة . أو هو برنامج state data I / O

تمهيد :

Women -------- **processor**
Recipe -------- **program**
Ingredients --------- **input data**
Activity ---------- **process**

إذا أصيبت المرأة أثناء إعدادها (الكيك) تقوم بعمل إسعافات أولية (first aid) وذلك لأن الإسعافات الأولية هي نشاط بمعنى عملية لها الأولوية في التنفيذ وبعد أن تنتهي تواصل عملها من نفس النقطة التي توقفت فيها وهكذا وهذا ما يعرف بالبرمجة المتعددة (المتوازية) Multi programming

حالات العملية : process states : لكل عملية إحدى خمسة حالات هي :

(1) الحالة New : وهذه الحالة التي تعني أن العملية قد تم إنشاؤها (Create)

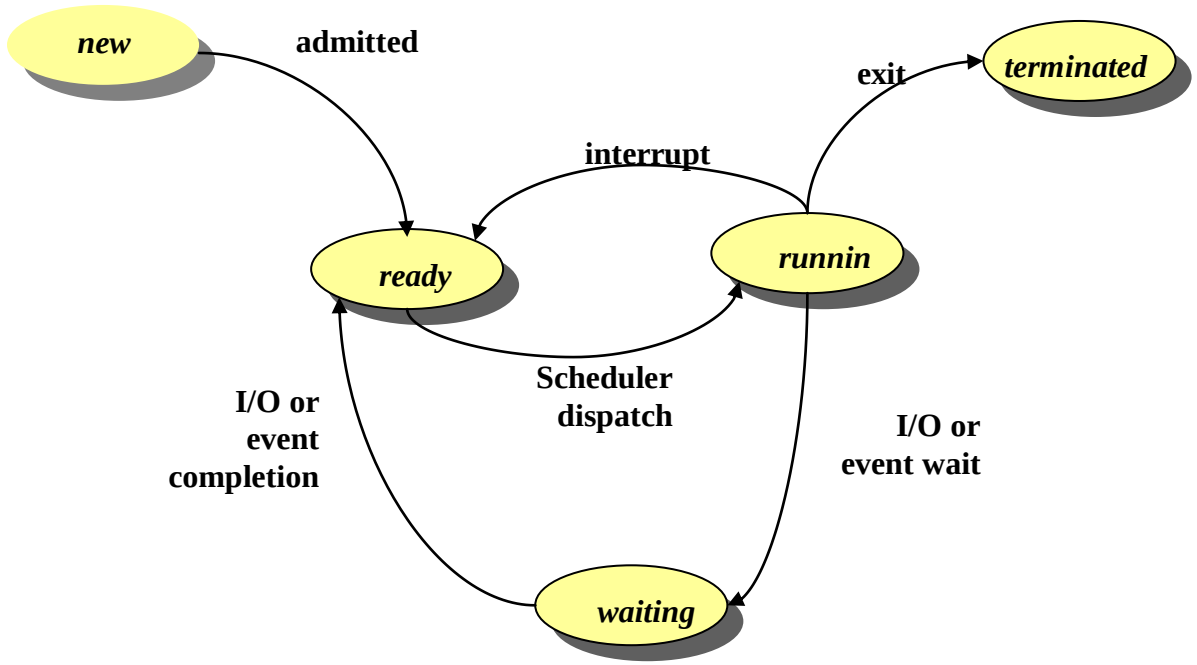
(2) الحالة Running : وتعني أن العملية في حالة تنفيذ وجودها في الـــ CPU

(3) الحالة Waiting / Blocked : تعني أن العملية تنتظر حدوث حدث معين مثل إكمال بيانات أو إستقبال إشارة معينة أو غيرها .

(4) الحالة Ready : تعنى أن العملية مستعدة للإستدعاء من قبل المعالج

(5) الحالة Terminated : تعنى أن العملية تم تنفيذها تماماً (فرغت من التنفيذ) .

* تكون هنالك عملية واحدة فى حالة تنفيذ فى أى معالج وحيد بينما هنالك العديد من العمليات فى حالة إستعداد وإنتظار .



من الـ Running إلى الـ Waiting تقوم بالنشاط العملية نفسها (process) أما فى البقية يقوم بالنشاط الـ **Scheduler dispatch**

محاضرة رقم (4)

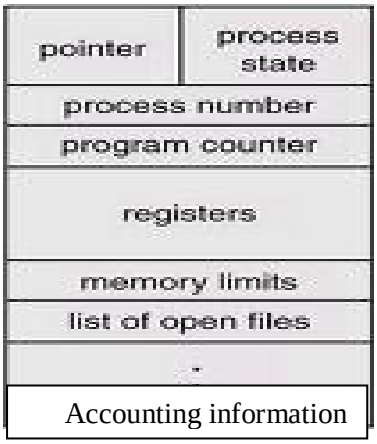
إدارة العمليات : Process Management

عبارة عن data structure module تمكن من تمييز كل process عن الأخرى (waiting , running) وغيرها ويستطيع تمييزها عن طريق أعداد صحيحة موجبة تعرف بـ process ID .

كتلة تحكم العمليات / المهمات Process Control Block : (PCB) (هى مخزن لكل المعلومات التى تخص العملية)

هى عبارة عن بنائيات أو هيكل بيانات (data structures) تخزن فيها كل المعلومات الخاصة بالعملية المعينة فى شكل سجلات .

- أى عملية موجودة بالنظام بها (PCB) أو (process table) أو (task control block) . فكل عملية تدخل النظام تمثل بـ (PCB) وتخزن فى الذاكرة فى شكل سجلات .
- تخزن الـ PCB على سجلات سريعة جداً أسرع كثير من سرعة عناصر الذاكرة الأساسية حتى يتمكن نظام التشغيل من متابعة حالات التنفيذ . وتحتوى على كل المعلومات والبيانات اللازمة لنظام التشغيل لإدارة المهمة .
- أى عملية تدخل النظام وعند تنفيذها داخل الـ (PCB) يتم تحديد أى نوع تحتاجه من المصادر هل هو (S.W & H.W) ويكون موجود فى ما يسمى الـ Resource S.W & H.W وتوضح العملية داخل نظام التشغيل بما يعرف بـ PCB



مخطط كتلة تحكم المهمات :

* الـ pointer : هو المؤشر للـ pcb التالية .
* الـ process state : حالة العملية هى ready & running & new .
* الـ process no : هو رقم العملية .
* الـ program counter : عداد برنامج التعليمات ويدل على عنوان التعليمة التالية التى ستنفذ للعملية .
* الـ Register : المسجلات تتنوع المسجلات فى الأرقام والأنواع حسب معمارية الحاسوب وتحتوى المسجلات على مخازن التجميع (accumulators) وفهارس المسجلات (register index) ومؤشرات المكدسات (stack pointer) و كل هذه المعلومات يجب حفظها عند حدوث أى مقاطعة

حتى تواصل العملية بصورة صحيحة . (الـ register تسجل فيه حالة العملية والزمن الذى توقفت فيه).

* معلومات عن جدولة وحدة المعالجة المركزية cpu scheduling information تحتوى على process priority ومؤشرات لصفوف الجدولة .

* الـ Memory limits : هى تحتوى على معلومات توضح حد الذاكرة (قيمة موقع العملية بالذاكرة) وقيمة الـ limit والـ base الخاصة بالمسجلات وعلى الـ page tables .

* الـ List of open file : هى الملفات التى يتم فتحها فى لحظة الـ I / O التى يتم تخصيصها للعملية .

* الـ Accounting information : ونعنى بها سعة الـ CPU للبرامج فى نفس اللحظة وكذلك عدد العمليات والزمن المستخدم .

جدولة العمليات : Process Scheduling

توضيح : إن مفهوم البرمجة المتعددة Multi programming وإشتراك الزمن Time sharing يعنى إدخال عدة برامج بالذاكرة ومعالجتها معاً فى نفس الوقت ونعنى بالـ Time مشاركة الوقت بأن تتحول الـ CPU من عملية لأخرى فى نفس الوقت .

والهدف من البرمجة المشتركة (المتعددة) هو الإستفادة من وحدة المعالجة المركزية للحد الأقصى وذلك بجعلها فى حالة تنفيذ دائم حتى تمكن المستخدم من التعامل مع أى برنامج أثناء تنفيذه .

– ملحوظة : لكل صف من صفوف الجدولة مؤشر يشير للعملية التالية وأى عملية لها PCB خاص بها .

صفوف الجدولة (جدولة الصفوف) : Scheduling Queues

الصفوف تخزن كل الـ linkedlist وأى عملية لا بد من وجودها فى صف معين . ولصفوف الجدولة أنواع منها :

(1) صف المهمة : **Job Queue** :

وهو صف توجد به كل عمليات النظام بمختلف حالاتها وهو عبارة عن هيكل بيانات .

(2) صف الإستعداد : **Ready Queue** :

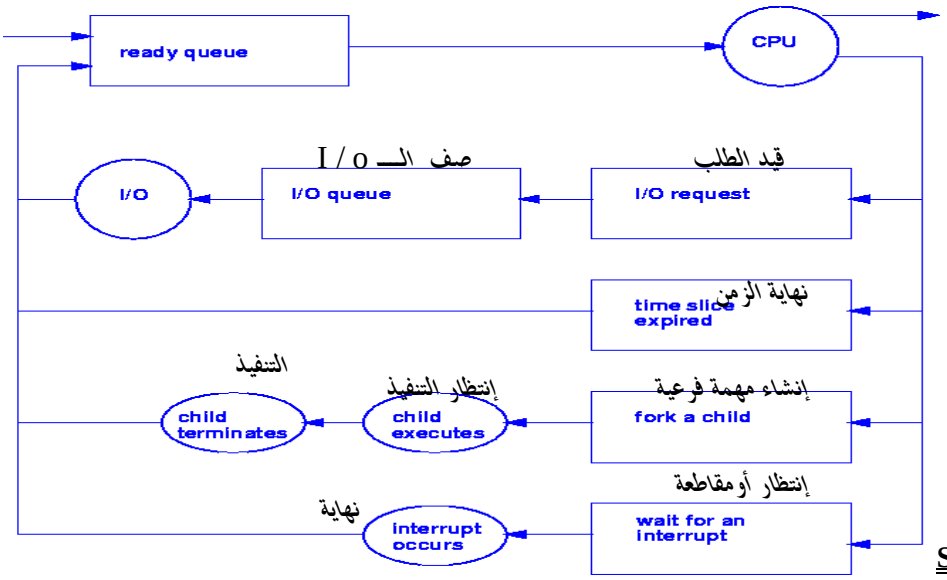
وهو صف توجد به العمليات الجاهزة للتنفيذ والتى تكون موجودة بالذاكرة الرئيسية فى حالة إنتظار أو إستعداد.

(3) صف الأجهزة : **Device Queue** :

وهو صف توجد به كل عمليات I / O device الخاصة بالنظام .

توضيح: أثناء تنفيذ العملية فى الـ cpu عند توقفها فجأة يكون هذا نتيجة مقاطعة أو حدوث طلب (I / O Request) فإذا كان المصدر غير متوفر أى تم طلبه من عملية أخرى فإن العملية يجب أن تنتظر حتى يتم تحرير ذلك المورد من العملية الاخرى والعمليات التى تكون فى حالة انتظار I / O device معين تعرف بـــ device Request

الرسم التالى يوضح صفوف الجدولة



يعنى صفوف

يعنى الحالات

يعنى المصادر

المجدولات (الجدولة) : Schedulers

يوجد نوعان منها فى كل نظم التشغيل هما الأول Long Term Scheduler والثانى Short Term Scheduler أو CPU Scheduler وفى بعض نظم التشغيل يوجد نوع ثالث يسمى Medium Term Scheduler وبه ما يسمى الـ Swapping ونعنى به حركة العمليات من وإلى الذاكرة .

* فى الـ batch system لا يتم تنفيذ كل العمليات حالاً والتى لم يتم تنفيذها يتم تجميعها وتوضع فى معدة تخزين مثلاً القرص حتى تنفذ لاحقاً وذلك بواسطة الـ long ثم بعد ذلك بواسطة الـ short .

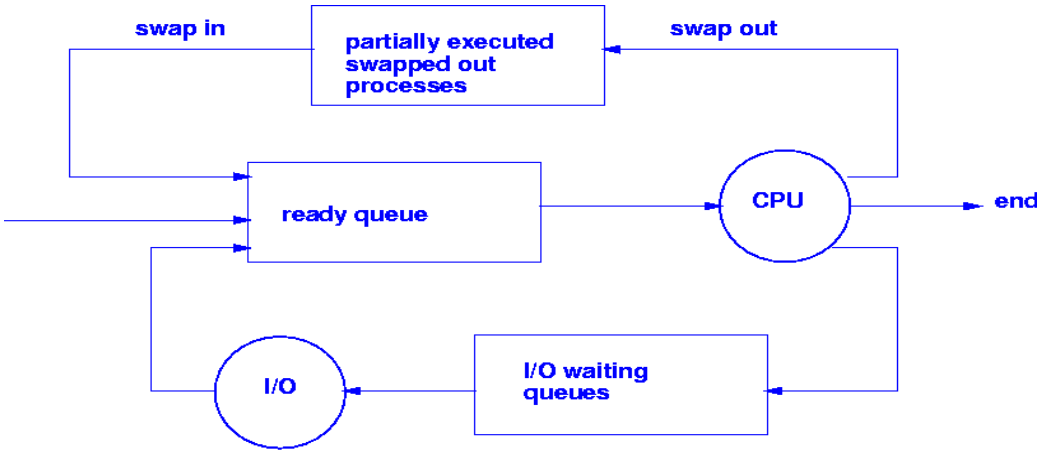
ملحوظة : أى عملية تدخل النظام للتنفيذ موجودة فى صف المهمات (job queue) .ولكنها فعلياً موجودة بالذاكرة .

*الـ Long Term Scheduler : هى الآلية التى تقوم باختيار نقل العمليات التى يجب تنفيذها من الذاكرة إلى الـ cpu .

*الـ Short Term Scheduler : هى الآلية التى تقوم باختيار العمليات التى يتم تنفيذها من حالة الـ Ready إلى الـ Running .

*الـ Medium Term Scheduler : تقوم بالـ Swapping وتعنى تحريك العملية من الذاكرة وإعادتها مرة أخرى وإعادتها وهكذا وذلك للإستفادة القصوى من الذاكرة .

الرسم التالى يوضح الـ Medium Term (Time-sharing) Scheduler



*الـ context switch : هو عملية تحويل وحدة المعالجة المركزية من عملية لأخرى بعد حفظ العملية القديمة وتحميلها لأجل العملية الجديدة .

(1) سرعة الذاكرة

(2) عدد المسجلات التى يجب نسخها

(3) وجود تعليمات خاصة مثل تعليمة تخزين المسجلات ويتراوح زمن العملية context switch بين 1 إلى 1000 ميكرو ثانية .

يوجد نوعان من المجدولات للـ process هما :

(1) CPU poundprofide وسميت بهذا الإسم لأنها تستخدم وحدة المعالجة المركزية (تعمل بطريقة حسابية) .

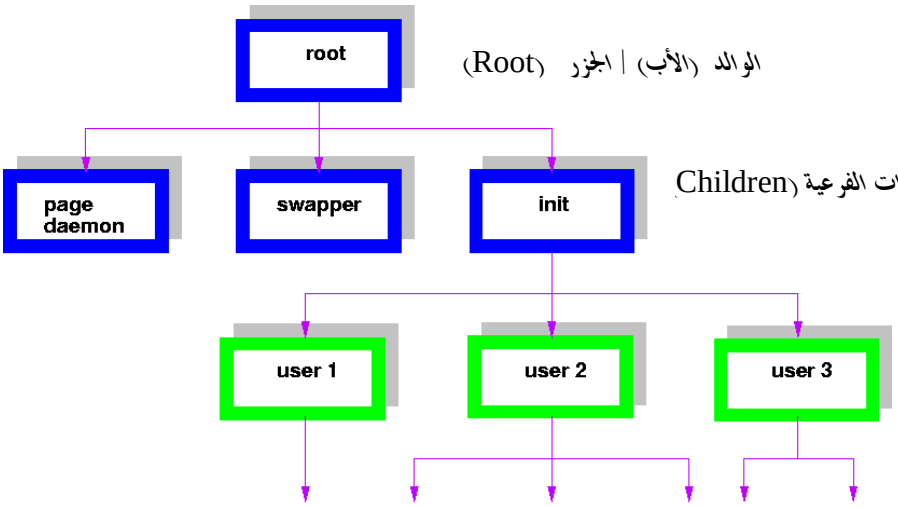
(2) I / O boundprode سميت بذلك لأنها تستخدم I / O فى أغلب الأحيان .

العمليات على المهمات / العمليات: Operators On Process

تنفيذ العمليات يتم بصورة متزامنة ويتم أنشاؤها وتدميرها بصورة ديناميكية .

(1) إنشاء العملية : Process Creation

عند إنشاء أى عملية نستخدم Sys Call تسمى Create وكما هو معلوم أن أى عملية يمكن أن تقوم بمجموعة عمليات أخرى تسمى العملية الرئيسية بالأب Parent process والعمليات المولدة منها تسمى الأبناء Children process وبذلك يكون لدينا شجرة من العمليات كمثال عملية الطباعة عملية رئيسية أنشأت عمليات أخرى كمثال إدخال الورق للطابعة



الوالد (الأب) | الجزر (Root)

الأبناء | العمليات الفرعية (Children)

* الموارد :

لنفترض أن العملية الأب تسمى (A) والعمليات المولدة (C , D , E) وأن العملية (C) ولتكن فى الرسم (init) وهكذا ولكل عملية إنشاء عمليات فرعية (مولدة) تحتاج لمجموعة موارد أخرى قد تكون S.W & H.W وهنالك ثلاثة إحتمالات لهذه الموارد هى :

(1) أن العملية (C) لا تتشارك مع (A) فى الموارد .

(2) أن العملية (C) تتشارك مع (A) فى بعض الموارد .

(3) أن العملية (C) تتشارك مع (A) فى كل الموارد .

* التنفيذ :

*يتم التنفيذ بصورة متزامنة ولكن هنالك إحتمال آخر أن تنفذ العملية الـ (Sub) وهى الفرعية أولاً ثم تتم مواصلة العمل .

* تنفيذ العملية المولدة بجزئية من موارد العملية الأب يمنع أى عملية أخرى من إنشاء العديد من العمليات الأخرى منها وذلك بالطبع يؤدى إلى تحميل زائد للنظام .

* التخزين :

من الممكن أن يتشارك الأب والأبناء نفس المساحة فى الذاكرة أو أن يكون للأب مساحة خاصة به فى الذاكرةأما فى مساحة العنوان يكون هنالك إحتمالان هما :

(1) تكون العملية المولدة عبارة عن نسخة من العملية الأب . (2) تكون العملية المولدة لديها برنامج خاص محمل بها .

* فى نظام يونكس يتم إنشاء العملية الجديدة بإستخدام نداء نظام يسمى **Fork system call** .

ملحوظة : إذا إستمرت عملية توليد العمليات سوف يحدث **Over load** لذا يتم إيقاف ذلك بشرط محدد مرتبط بالموارد .

أثناء التنفيذ : يكون هنالك إحتمالان هما :

الأول : تواصل العملية الأب التنفيذ بصورة متزامنة مع العملية المولدة . الثانى : تنتظر العملية الأب حتى يتم تدمير بعض أو كل العمليات المولدة .

(2) تدمير العملية : Process Termination :

*يتم تدمير العملية بعد أن تنتهى من تنفيذ كل التعليمات الخاصة بها عن طريق **Sys call** لإنهاء العمليات يسمى **Exit** (إنهاء) ويتم تحرير كل الموارد الخاصة بها عن طريق نظام التشغيل

* فى أحيان أخرى نستخدم نداء نظام يسمى **Abort** (قطع) ولكن القطع لا تستخدم إلا عن طريق العملية الأب **Parent** وذلك لمنع المستخدمين من تدمير وظائف بعضهم البعض .

أسباب تدمير العمليات :

(1) إنتهاء المهمة للإستفادة من مساحة الذاكرة (تنفيذ العملية) .

(2) إنتهاء الوقت المخصص للمورد .

(3) تدمير العملية الأب بواسطة Abort .

* العديد من نظم التشغيل تقوم بتدمير العمليات المولدة بصورة مباشرة بعد تدمير العملية الأب

محاضرة رقم (5)

العمليات المتعاونة Co operating process : نظام التشغيل يقوم بتقسيم الـ process إلى نوعين هما :

(1)عمليات مستقلة Independent process (2)عمليات متعاونة Co operating process

العمليات المستقلة هي عملية لا تؤثر ولا تتأثر بالعمليات الأخرى أي إنها لا تتشارك معها في أي نوع من البيانات .

العمليات المتعاونة هي عملية تؤثر وتتأثر بالعمليات الأخرى لأنها تتشارك معها في البيانات أو المصادر .

ما الفائدة من المتعاونة ؟

(1) المشاركة في المعلومة : أحياناً يكون هنالك العديد من المستخدمين بحاجة إلى المعلومة نفسها (ملف مثلاً)

(2) السرعة في إنجاز المهام Speed up : إذا كنا نريد القيام بمهمة بسرعة أكبر فإنه يمكن تجزئتها إلى عدة مهام أصغر وسيتم تنفيذ هذه المهام بصورة متوازية مع بعضها البعض وهذا يحدث إذا كان هنالك عناصر معالجة متعددة .

(3) التوافقية Convenience : أحياناً يحتاج المستخدم إلى أكثر من مهمة فى وقت واحد مثلاً قد يحتاج أن يطبع أو يترجم أو يحرر فى وقت واحد .

يقسم البرنامج إلى مجموعة عمليات | مهام وذلك لوجود مجموعة معالجات متعددة بالجهاز

* لتوضيح مفهوم التعاون بين العمليات مثلاً مشكلة المنتج والمستهلك .

مشكلة المنتج والمستهلك Producer & consumer Problem

هنالك عملية منتجة تنتج معلومات يتم إستهلاكها بواسطة المستهلك ويتم تخزينها في **Buffer** بسعة معينة وعندما يمتلئ توقف عملية الإنتاج حتى يتم الإستهلاك وإذا كان الـ **Buffer** فارغ لا يتم إستهلاك .

إذاً :

يتم الإنتاج أولاً ثم الإستهلاك بعد ذلك .مثلاً برنامج الطباعة ينتج حروف تستهلك بواسطة الطابعة والمترجم ينتج شفرات تجميعية يتم إستهلاكها بواسطة الحاسوب بالـ**Assembler** وهكذا .

وهنالك مشكلتان لهذه العملية في الـ**Buffer** هما :

(1) قد يكون الـ**Buffer** غير محدود **Unbound Buffer** لذلك يكون في حالة إنتاج دائم .

(2) قديكون الـ **Buffer** محدود **Bound Buffer** وفي هذه الحالة تكون العملية قد توقفت لمحدودية الـ**Buffer** .

ملحوظة في الحالة الثانية إذا لم يتم انتاج فإن المستهلك تكون في وضع الـ **waiting** وكذلك إذا لم يتم إستهلاك فالمنتج ينتظر .

* المخزن يتم دعمه عن طريق نظام التشغيل او عن طريق تشفير واضح من قبل مبرمجي البرامج التطبيقية .

* تقوم العملية المنتجة بإنتاج مفردة واحدة في أثناء إستهلاك العملية المستهلكة لمفردة أخرى كما يجب أن تعمل العملية المنتجة والمستهلكة بصورة متزامنة **synchronized** حتى لا تحاول العملية المستهلكة إستهلاك مفردة لم يتم إنتاجها بعد .

ورقة بحثية رقم (3)

ما هي الخيوط Threads ؟

جدولة وحدة المعالجة المركزية :CPU Scheduling

* تعريف :

هي إختيار أحد العمليات من صف الـ ready Queue إلى مرحلة التنفيذ وذلك بواسطة أحدى خوارزميات الجدولة .

الهدف من ذلك : جعل الـ CPU مشغولة دوماً .

توضيح : تعتبر جدولة وحدة المعالجة المركزية من أهم الأشياء بالنسبة لنظم التشغيل متعددة البرمجة حيث أنه عن طريق تحويل وحدة المعالجة المركزية من عملية لأخرى تزيد إنتاجية الحاسوب والهدف الأساسي من البرمجة المتعددة هو الإستفادة القصوى من وحدة المعالجة المركزية وذلك بجعلها في حالة تنفيذ دائم للعمليات ،،،،،،،،،، في نظم الحاسوب البسيطة عندما تكون العملية في حالة تنفيذ تكون وحدة المعالجة في وضع مستقر ولا تنفذ أي عمل آخر طيلة فترة الإنتظار .

خوارزميات الجدولة :Algorithms **Scheduling**

مفاهيم أساسية : قبل التعرف على خوارزميات الجدولة لا بد أن نتعرف على هذه المفاهيم الأساسية وهي :

* جدولة عدم الإيقاف **Non – preemptive scheduling** وهي جدولة عدم إيقاف المهمة .

* جدولة الإيقاف (الجدولة الشفعية) **preemptive scheduling** وهي جدولة الإيقاف لمهمة.

* البرمجة المتعددة : **Multi Programming** : يعني إدخال عدة برامج بالذاكرة ومعالجتها معاً في نفس الوقت

* الـ **Job / batch** : برامج يتم تنفيذها من غير المستخدم مثال برامج النظام .

* الـ **User** : برامج تحتاج في تنفيذها للمستخدم مثال البرامج التطبيقية .

* الـ **CPU burst cycle** : CPU : الفترة التي تقضيها العملية داخل وحدة المعالجة وهو عادة أقل بكثير من زمن الـ **I / O** .

* الـ **I / O burst cycle** : I / O : الفترة التي تقضيها العملية في وحدات الـ **I / O** .

* مفهوم المنجز **Dispatcher** هي آلية تعطي وحدة المعالجة المركزية التحكم في العمليات المختارة بواسطة الـ Short Term Scheduler

– يجب أن يكون المنجز سريعاً جداً عند التحول من عملية لأخرى .

– يعرف الزمن الذي يستغرقه المنجز لإيقاف عملية وبدء أخرى بـ **Dispatch latency** .

* تعريف :

جدولة الإيقاف هي: مقاطعة تحدث للعملية أثناء تنفيذها تتسبب في وقفها وتحريرها من وحدة المعالجة ولذا تنزع الـ **CPU** منها وتعطى لعملية أخرى .

ملحوظة: لا يعني وجود العملية في أول الصف تنفيذها أولاً وربما نعامل هذا العنصر على إنه **LIFO** أو نظام **Linkedlist**

* الجدولة الناجحة لوحدة المعالجة المركزية تعتمد على :

(1) تنفيذ عملية تحتوى دورة لتنفيذ **cpu** و **I/ O wait** (2) عملية تتناوب بين الحالتين

أما تعريف جدولة عدم الإيقاف الـ **Non** عكس ما سبق ذكره تماماً .

ويقابل جدولة الإيقاف و عدمها يوجد لدينا ما يسمى بالـ **preemptive process & Non preemptive process**

* عندما تكون وحدة المعالجة المركزية فى وضع مستقر (**idle**) يقوم نظام التشغيل بإختيار إحدى العمليات من الـ **ready queue** للتنفيذ ويخصص لها وحدة المعالجة والـ **ready queue** ليس **fifo** بالضرورة ويمكن أن يكون **fifo** أو **priority** أو **unordered linked list tree**

س/ متى يتم إتخاذ قرار جدولة وحدة المعالجة المركزية في إحدى أربع حالات هي :

(**1**) في حالة تحويل العملية من **Running** إلى **Waiting** (تحتاج إلى **I / O**)

(**2**) في حالة تحويل العملية من **Running** إلى **Ready** (في حالة مقاطعة)

(**3**) في حالة تحويل العملية من **Waiting** إلى **Ready** (توفر لها **I / O**)

(**4**) في حالة تدمير العملية **Terminate** (إنتهاء المهمة)

* يلاحظ أن الجدولة في الحالة الأولى والرابعة هي **Non preemptive** أما الحالة الثانية والثالثة هي **preemptive** . تستخدم جدولة عدم الإيقاف (**Non preemptive**) فى بيئة ويندوز حيث أنها الطريقة الوحيدة التى يتم إستخدامها على برامج عتاد معينة وذلك لأنها لا تتطلب عتاد معين مثل الـ **timer** .

مصطلحات مهمة :

* إستخدام المعالج **CPU utilization** ونعني به إبقاء المعالج مشغولاً بقدر الإمكان بنسبة معينة مثلاً **24**٪ والزمن الفعلى له (**40 – 100** ٪).

* زمن الإنتظار **Waiting time** هو متوسط الفترة الزمنية لبقاء العملية في حالة الـ **Waiting** حتى تتوفر لها الـ**CPU** .

خوارزمية الجدولة لا تؤثر فى الزمن المستغرق بالنسبة للعملية فى تنفيذ أدنى عمل **I / O** ولكنها تؤثر فى الزمن الذى تستغرقه فى صف الـ **ready** أى زمن الإنتظار .

* الطاقة الإنتاجية **Through Put** نعني به عدد العمليات المنفذة في وحدة زمنية معينة . (الطاقة الإنتاجية محددة أومعدل العمليات المنجزة)

مثال: إذا كان لدينا عمليات موضحة من **A** إلى **D** حسب الزمن المخصص لكل عملية فإن الطاقة الإنتاجية هي مجموع العمليات

العملية **Process**	الزمن المخصص **Burst time**
A	**5**
B	**2**
C	**5**
D	**3**

العمليات في الـــ **15** وحدة زمنية عددها أربعة

الإنتاجية = مجموع زمن الإنتظار ÷ عدد العمليات = **15÷4 = 3.75** وحدة زمنية

* الزمن الدوري **Turnaround time**: هى الفترة الزمنية التى تستغرقها العملية حتى نهاية التنفيذ .

تعريف آخر : هى مجموع الفترات الزمنية التى تمر بها العملية كمثال: فترة الإنتظار حتى تصل الـ **CPU** وفتـــرة الـــ **Running** وفترة الـــ**Ready Queue** وفترة الـــ **I / O Request** أو : هو يعنى زمن الإنتظار + زمن التشغيل .

وبالرجوع للمثال السابق فإن الزمن الدورى هو :

A= 0 + 5 = 5 **B = 5 + 2 = 7** **C= 7 + 5 = 12** **D = 12+ 3 = 15**

وبالرجوع للمثال السابق فإن زمن الإنتظار هو : **A = 0** **B = 5** **C= 7** **D = 12**

* زمن الإستجابة : **Response time** : هو الفترة الزمنية من بداية الطلب حتى حدوث أول إستجابة للطلب وهنا يجب الإشارة إلى أنه ليس الزمن الذى يستغرق لإخراج تلك الإستجابة وبالتالى يجب زيادة (إستخدام المعالج والطاقة الإنتاجية) وتقليل (الزمن الدورى وزمن الإنتظار وزمن الإستجابة)

* الأولويات **priorities**: كل عملية تحمل عنوان به إشارة تحدد أفضلية المهمة وأولويتها فى التنفيذ وتقوم خوارزمية الجدولة بإختبار العمليات ذات الأولوية العالية .

* الزمن المخصص **Burst time**: هو الزمن الذى يستغرقه المعالج للإنتهاء من المهمة ويسمى أيضاً (**service time**)

خوارزميات الجدولة :

هنالك ست خوارزميات للجدولة هى :

(**1**) الأول فى الوصول الأول فى المعالجة (**FCFS**) **First Come First Serve Scheduling**

(**2**) أقصر المهمات الأول فى المعالجة (**SJFS**) **Shortest Job First Scheduling**

(**3**) خوارزمية الأولوية **Priority Scheduling**

(**4**) التخصيص الدورى **Round Robin Scheduling**

(5) الصفوف ذات المستويات المتعددة **Multi Level Queues Scheduling**

(6) صفوف التغذية الراجعة ذات المستويات المتعددة **Multi Level Feedback Queues Scheduling**

الأهداف :

(1) العدل **Fairness** : وهو توفير الـ **CPU** لكل العمليات بغض النظر عن إتمام المهمة أو عدمه (إعطاء فرصة لأى عملية) .

(2) التوازن **Balance** : ونعنى به توازن النظام عموماً (إبقاء كل أجزاء النظام مشغولة تؤدى عمل ما)

(3) إنفاذ سياسة نظام التشغيل **Policy enforcement** : ونعنى به تنفيذ سياسة معينة للتعامل مع نظام التشغيل .

(1) الأول فى الوصول الأول فى المعالجة (FCFS) _ First Come First Serve Scheduling

العملية التى تطلب الـ **cpu** أولاً يتم تخصيصها لها أولاً .

مثال : إذا كان لدينا العمليات التالية والتى وصلت على الترتيب الموضح **p1** أولاً وهكذا

process	Burst time
P1	24
P2	3
P3	3

يمكن تمثيلها بمخطط جانت (**Gant Chart**) كما يلى :

P1	P2	P3
0	24	27 30

زمن الإنتظار بالنسبة لـ **p1** هو **0** و بالنسبة لـ **p2** هو **24** وبالنسبة لـ **p3** هو **27**

ومتوسط زمن الإنتظار (**average waiting time**) = (**0 + 24 + 27**) ÷ **3** = **17 mlsc** ملى ثانية .

– مثال آخر : إذا وصلت العمليات على الترتيب **p2** ثم **p3** ثم **p1** يكون زمن الإنتظار أقل = **3 mlsc** = **3** ÷ (**3 + 6**)

P2	P3	P1
0	3	6 30

* هذه الطريقة جيدة فى حالات المهمات الطويلة ولكنها غير عادلة بانسبة للمهمات الصغيرة كما فى المثال الأول.

* تعتبر هذه الخوارزمية من خوارزميات إيقاف الجدولة (**Non preemptive**) فعندما يتم تخصيص الـ **cpu** لعملية معينة فإنها تحتفظ بالـ **cpu** حتى تقوم بتحريرها وذلك بتدميرها أو بطلبها لـ **I / O** .

* إنتظار العمليات الصغيرة فى الـ **ready queue** حتى يتم الإنتهاء من الكبيرة وحتى تحرر وحدة المعالجة

* غير مناسبة لأنظمة الـ **time sharing system** حيث أن المهم هو أن يتشارك كل مستخدم والـ **cpu** مع المستخدمين الآخرين وبالتالى ستكون هنالك مشكلة إذا إحتفظت العملية بوحدة المعالجة لفترة طويله .

(2) أقصر المهمات الأول فى المعالجة (SJFS) _ Shortest Job First serve Scheduling

تعتمد على طول الزمن المخصص (**cpu burst**) حيث نختار العملية التى يكون لها أقصر طول زمن مخصص وإذا كانت هنالك عمليتان لهما نفس طول الزمن فإننا نستخدم فى هذه الحالة خوارزمية الأول فى الوصول (**FCFS**) لحل هذا التعقيد .

مثال : مثال : إذا كان لدينا العمليات التالية والتى وصلت على الترتيب الموضح **p1** أولاً وهكذا

process	Burst time
P1	6
P2	8
P3	7
P4	3

يمكن تمثيلها بمخطط جانت (**Gant Chart**) كما يلى :

P4	P1	P3	P2
0	3	9 16	24

نحسب متوسط زمن الانتظار = (**0+9+16+3**) ÷ **4** = **7** ملى ثانية .

* باستخدام خوارزمية الأول فى الوصول **fcfs** سيكون (**10.25**) ملى ثانية وبالتالى متوسط زمن الإنتظار فى خوارزمية أقصر المهمات **sjfs** أقل لأنها تختار العملية التى يكون لها زمن إنتظار اقل .

* هذه الخوارزمية قد تكون (**preemptive**) وقد تكون (**non – preemptive**) ويتم الإختيار عندما تصل عملية جديدة للـ **ready queue** أثناء تنفيذ العملية السابقة والتى يكون لها زمن تخصيص أقل من تلك التى تكون فى حالة تنفيذ وفى هذه الحالـــــة فإن **preemptive - SJFS** ستستولى على العملية التى تكون فى حالة تنفيذ فى حين أن الـ **non – preemptive - SJFS** ستسمح للعملية التى تكون فى حالة تنفيذ بإكمال الزمن المخصص لها ،،،،،،،،،،،،، أحياناً تعرف خوارزمية **preemptive - SJFS** بـــ **SRTF Scheduling**

مثال : **SJFS** بالطريقتين : أولاً **non–preemptive** :

process	Arrive time	Burst time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

يمكن تمثيلها بمخطط جانت (**Gant Chart**) كما يلى :

P1	P3	P2	P4
0	7	8	12	16

ثانياً **preemptive** :

P1	P2	P3	P2	P4	P1
0	2	4	5	7	11	16

* هذه الطريقة تقلل متوسط زمن الإنتظار ولكن تظهر مشكلة تسمى المجاعة (**Starvation**) وهى تعنى أن هنالك مهمة قد لا يتم تنفيذها إطلاقاً .

* عملية إعطاء وحدة المعالجة المركزية لأصغر مهمة تعرف بعملية الإشباع (**Saturation**) وهى إشباع وحدة المعالجة المركزية .

(3) خوارزمية الأولوية Priority Scheduling:

تسمى أيضاً جدولة أعلى معدل إستجابة (**HRRN**) **High Response Ration Next** .

* هذه الخوارمية تعتمد على الأسبقية أو الأقـــدمية حيث تقوم بتخصيص وحدة المعالجة المـــركزية للعملية التى يكون لها أقدمية أو أسبقية بين العمليات الأخرى وفى حالة إشتراك أكثر من عملية فى الأسبقية فإنه تتم جدولة العملية عن طريق خوارزمية الأول فى الوصول .

* تختلف الأنظمة فى مفهوم الأولوية والتى تنحصر فى المدى ما بين (**0 - 7**) أو (**0 - 40**) بعض الأنظمة تستخدم الأرقام الصغـــــيرة للدلالة على الأولوية وبعضها يفعل العكس .

* يمكن أن تكون هذه الخوارزمية إيقاف أو عدم إيقاف . مثال : بالطريقتين :

process	Arrive time	Priority	Burst time
P1	0	3	4
P2	1	4	3
P3	2	6	3
P4	3	5	5

أولاً بعدم الإيقاف **non–preemptive** :

P1	P2	P3	P4	P2	P1
0	1	2	5	10	12	15

ثانياً بالإيقاف **preemptive** :

P1	P3	P4	P2	P1
0	2	5	10	13	15

* المشكلة الأساسية لخوارزمية الأولوية تتمثل فيما يعرف بـــ **Indefinite blotching** أو الـــ **Starvation** ويعنى ان العملية أو المهمة تكون فى حالة إستعداد للتنفيذ ولكن أولويتها صغيرة وبالتالى فى كل مرة يتم تنفيذ العمليات ذات الأولويات الأكبر وعموماً قد يحدث أحد أمرين هما : الأول : أن يتم تنفيذها فى آخر الأمر . أو الثانى : أن يتوقف نظام الحاسوب (**crash**) ويفقد العمليات غير المكتملة ذات الأولوية الأقل وقد حدث هذا عام **1973**م فى شركة **IBM** . ،،،،،،، والحل الأمثل لهذه المشكلة ما يعرف بتقنية الـ **ging** .

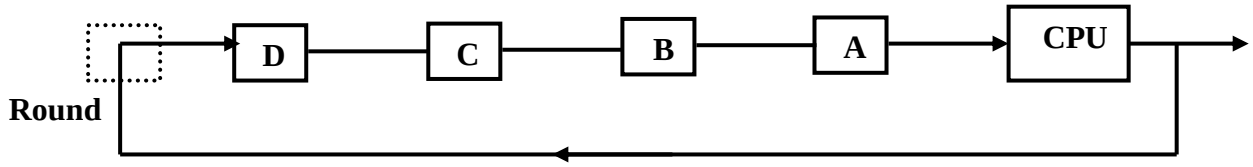
* تقنية الـــــ ging : هى تقنية تعمل على زيادة الأولوية بالنسبة لتلك العمليات التى إنتظرت فى النظام لفترة طويلة بعد كل فترة زمنية معينة .

(4) التخصيص الدورى Round Robin Scheduling

* تم تصميمها خصيصاً لأنظمة إشتراك الزمن (**time sharing systems**) وهى تشبه خوارزمية الأول فى الوصول ولكن بها وحدة زمنية صغيرة تسمى الـ **time quantum** أو الـ **time slice** .

* تتم معالجة الـ**ready queues** كـ **Circler queues** حيث يقوم الـ **cpu scheduling** بالذهاب إلى الـ**ready queues** ويقوم بتخصيص وحدة المعالجة المركزية لكل العمليات لفترة زمنية **quantum** معينة .

* تتم معاملة الـ**ready queues** كـ **Fifo queues** حيث تتم إضافة العمليات الجديدة فى مؤخرة الصف ويقوم بوضع مؤقت يعمل على المقاطعة بعد وحدة زمنية واحدة (**1 time quantum**) وقد يحدث أحد أمرين هما : الأول : إما أن يكون الزمن المخصص الخاص بالعملية أقل من الـ **quantum** وفى هذه الحالة فإن العملية نفسها تقوم بتحرير وحدة المعالجة المركزية ويقوم المجدول بإختيار العملية التالية من الصف والأمر الثانى هو : أن يكون زمن التخصيص أكبر من الـ **quantum** فتحدث مقاطعة وتحدث عملية **Context Switch** وسوف توضع العملية فى مؤخرة الصف وبالتالى يقوم المجدول بإختيار العملية التالية من الصف .



process	Burst time
P1	24
P2	3
P3	3

time quantum = 4 milliseconds & Average waiting time = 17/ 3 = 5 .6 mlsc

P1	P2	P3	P1	P1	P1	P1	P1

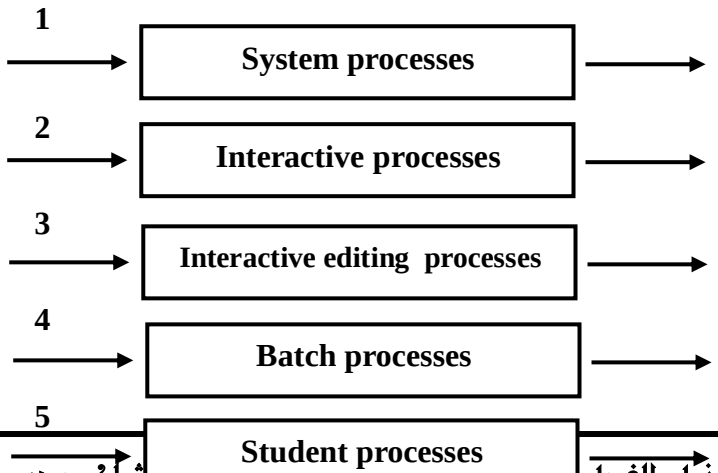
0 4 7 10 14 18 22 26 30

* يجب أن تكون الفترة الممنوحة لكل مهمة مناسبة لأنه إذا كانت كبيرة لكل المهمات فإن خوارزمية الأولوية تصبح خوارزمية الأول فى الوصول فى لحظة من اللحظات . وإذا كانت الفترة قصيرة تكثر عمليات التبديل بين المهمات وهذا يؤثر على كفاءة النظام (إذا كانت صغيرة فإن منهجية **RR**) تعرف بـ **Processor sharing** .

(5) الصفوف ذات المستويات المتعددة Multi Level Queues Scheduling يتم تقسيم العمليات إلى :

(1) foreground (interactive) processes **(2) background (batch) processes**

يختلف كل من هذين القسمين فى إحتياجاته من ناحية الجدولة مثلاً تقوم خوارزمية الصفوف ذات المستويات المتعددة بتقسيم الصف (**ready queue**) إلى العديد من الصفوف الأخرى والتى تحتوى على العديد من العمليات ويكون لكل صف خوارزمية جدولة خاصة به حسب نوع العمليات وحسب أولويتها ... أخ فمثلاً **background queue** قد تتم جدولتها بـ **RR** بينما تتم جدولة **foreground queue** بخوارزمية **FCFS** مثلاً بالإضافة أنه لابد من وجود جدولة بين الصفوف وتنفذ عادة كـ **fixed priority preemptive scheduling**



* كل صف تكون له أولوية أعلى من أولوية الصفوف التى تقع بالأسفل مثلاً لا تستطيع أى عملية موجودة فى الصف *Batch processes* أن تبدأ فى التنفيذ إذا لم يتم تنفيذ كل العمليات التى توجد فى الصفوف (1 ، 2 ، 3) وأصبحت فارغة تماماً .

*إذا دخلت *Interactive processes* إلى الـ Ready queue أثناء تنفيذ الـ *Batch processes* فى هذه الحالة ستتوقف الـ *Batch processes* عن التنفيذ .

*يوجد **time slice** بين الصفوف حيث أن كل صف يكون له وقت محدد فى وحدة المعالجة .

(6) صفوف التغذية الراجعة ذات المستويات المتعددة Multi Level Feedback Queues Scheduling

* فى الخوارزمية السابقة نلاحظ أن الخوارزمية إذا كانت **background** أو **foreground** فإنها لا تستطيع التنقل بين الصفوف بينما نجد أن هذه الخوارزمية تسمح بذلك .

* تتلخص الفكرة فى تقسيم العمليات إعتماداً على خصائص الـ **cpu burst** لكل عملية فإذا كانت ذات زمن تخصيص كبير جداً فإنه يتم تحريكها إلى صف أقل أولوية وهذه الطريقة تجعل الـ **I / O bound** والـ **Interactive** موضوعة فى صف أكبر أولوية والعملية التى تكون فى حالة إنتظار طويلة فى صف أقل أولوية يتم نقلها إلى صف أكبر أولوية مما يمنع حدوث المجاعة (**Starvation**) .

توضيح : إنتقال العمليات من صف إلى آخر حسب الـ **quantum** وإذا لم تكتمل العملية تنتقل إلى مؤخرة الصف الذى يله ،

– لا تنفذ الصفوف السفلى بخوارزمية الأول فى الوصول إلا إذا كانت الصفوف العليا (التى قبلها) فاغة تماماً

اللهم صلى على النبى و آله ما إهتزت الأثلات من نفس الصبأ
اللهم صلى على النبى و آله ما كوكب فى الجو قابل كوكبا
اللهم صلى على النبى و آله ما قال ذو كرم لضيفٍ مرحبا

****

إذا ركلك الناس من الخلف فأعلم أنك فى المقدمة
وأزد على علمك أن ناج القبصر لا يقبه من الصداع
فالفشل فى التخطيط يقود إلى التخطيط للفشل

صدى –ديسمبر **2012** – أسألكم خاص الدعاء لى ولوالداى

لا تتجاهل المحاضرات أثناء العام الدراسى ///// وتتسارع بالإجتهاد لحظة قدوم الإمتحان ////// فهما أمران لايستقيمان أبداً///// تمعن بيسر وإقرأ بهدوء البال وراحة النفس وإستمتع بفكر دواخلك ولا تنحصر فى بوتقة اللجوء إلى الأسباب أو ما شابه ذلك //