الاسبوع السادس عشــر

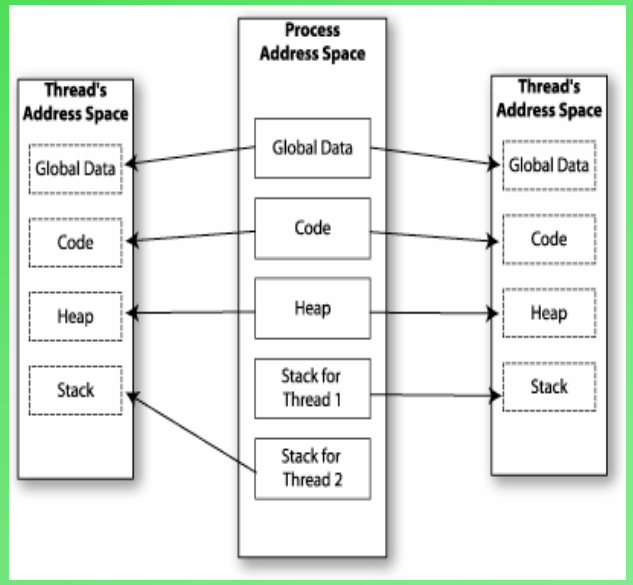
الخيوط Threads

الخيوط Threads

في أنظمة التشغيل التقليدية ، كل عملية لها فضاء عناوين يحوي نص البرنامج وبياناته بالإضافة إلى الموارد الأخرى منها الملفات المفتوحة والعمليات الأبناء ، وأنّ وضع هذه الموارد مع بعضها على شكل عملية يجعل إدارتها أسهل.

المفهوم الآخر للعملية هو خيط التنفيذ ، الذي يسمى اختصاراً بالخيط (Thread) . ويعتبر الخيط الوحدة الأساسية لوحدة المعالجة المركزية. الخيوط هي مجموعة من المهام التي ينقسم إليها البرنامج وذلك ليقوم بأكثر من مهمة بشكل متزامن حقيقي (وذلك عند وجود أكثر من معالج) أو تزامن كاذب (عند وجود معالج واحد) حيث يتم التبديل بين المهام بسرعة كبيرة تعطينا انطباع بالتزامن.

إن وجود عدة خيوط تنفيذ تعمل على التوازي ضمن عملية واحدة يشبه وجود عدة عمليات تعمل على التوازي ضمن حاسوب واحد. في الحالة الأولى، تتشارك الخيوط على فضاء العناوين والملفات المفتوحة والموارد الأخرى. أما في الحالة الثانية ، فتتشارك العمليات على الذاكرة الفيزيائية والأقراص والطابعات وغيرها من الموارد. وبما أن الخيوط تملك الكثير من خصائص العمليات، فإنها تدعى أحيانا بالعمليات ذات الثقل الخفيف (Lightweight processes) كما يستخدم مصطلح الخيوط المتعددة (Multithreading) للدلالة على حالة وجود خيوط متعددة في نفس العملية. نلاحظ في الشكل (29) عملية بأكثر من خيط ونلاحظ اشتراك الخيوط في فضاء العنوان (address space) .



الشكل (29) عملية مع الخيوط عدد/2

ولكل خيط:

1- رقم (ID) للتعريف به.

2- عداد (Program Counter PC) .

3- سجل (Register) .

4- مكدس (Stack) : منطقة عمل نضع فيها المتغيرات المحلية (local variable) .

5- حالة الخيط (Thread state) .

العلاقة بين الخيط (thread) والعملية (process)

1- الخيوط هي أجزاء من العملية.

2- لكل عملية فضاء عنوان (address space) مختلف وبيئة وقت التشغيل (runtime environment) ورقم تعريف (process ID) أما الخيوط طالما أنها مشتقة من العملية لها عنوان واحد , ويشترك الخيط مع الخيوط الأخرى بالموارد التابعة له.

3- لا يوجد خيط من غير عملية لكن العكس ممكن.

4- في حالة خزن السياقات (context switch) مع العملية يظهر (over head). أما بحالة الخيط لا يظهر إلا إذا استدعينا نظام التشغيل وغالباً لا نستدعيه .

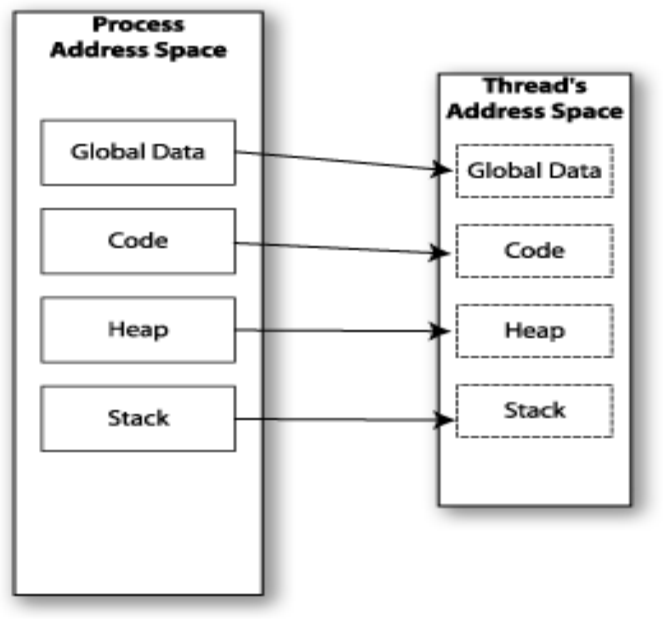
5- الخيوط هي أجزاء متزامنة التنفيذ داخل عملية ما.

6- العملية هي كلمة أعم ولفترة أطول. الخيط يقتصر على مفهوم "خيط التنفيذ".

7- العملية مرتبطة غالباً مع مستوى نظام التشغيل (مثل في متعدد المعالجة) ، بينما الخيط مرتبط بمستوى اللغة المنطقي المجرد.

8- الخيوط ليست مستقلة مثل العملية.

ويحصل اشتراك تام في كل شيء إذا كانت عملية واحدة فقط وخيط واحد فقط وكما في الشكل (30) أدناه .



الشكل (30) عملية وخيط واحد

مثال :

تشغيل برنامج (word) يسبب تفعيل العملية الخاصة به و لكن عندما يتم فتح مهمة أخرى ضمن (word) مثل نافذة البحث عن كلمة أو نافذة التنسيقات فإنه يتم تشكيل خيوط تنفيذ أخرى ضمن العملية (process) حيث أن كل (thread) يعبر عن مهمة مستقلة , وعند فتح نسخة أخرى من البرنامج فإنه لا يتم تشكيل عملية جديدة وإنما يتم خلق (thread) جديد ضمن العملية. إن مفهوم تعدد الخيوط (multithread) يعني تجزئة المهمة إلى مهمتين بحيث يكون هناك استقلالية بين المهمتين و بالتالي تشكيل خيطي تنفيذ مستقلين بحيث يمكن تنفيذهما في نفس الوقت على وحدتي معالجة فينخفض زمن التنفيذ إلى النصف.

عندما يكون هناك وحدة معالج واحدة (cpu) فإنه لا يمكن تنفيذ خيطين بنفس الوقت . لا يستطيع المعالج التسلسلي التعامل إلا مع خيط (thread) واحد بنفس الوقت . و لذلك فإن المعالج يقوم بخدمة كل خيط بشكل متتالي , أي أنه يقوم بتنزيل خيط و العمل عليه إلى حد معين ثم يقوم باستلام خيط آخر و العمل عليه إلى حد معين ثم يعود للخيط السابق ويتابع العمل من نقطة التوقف و هكذا.

لماذا نستخدم الخيوط ؟ (Why Use Threads?)

إن السبب الرئيسي لوجود خيوط في العديد من التطبيقات، هو وجود عدة فعاليات تحصل في نفس الوقت. وربما تتوقف بعض هذه الخيوط من وقت إلى آخر . أن تجزئة مثل هذه التطبيقات إلى خيوط تسلسلية متعددة تعمل على التوازي ظاهرياً، يجعل نموذج البرمجة بسيطاً جداً.

ما ينطبق على العمليات من مقاطعات وحالات العملية وخزن السياقات ينطبق على الخيوط ، يمكننا التفكير بالعمليات المتوازية ، لذا يضاف عنصر جديد هو إمكانية تشارك الخيوط المتوازية على فضاء عناوين واحد. سبب آخر لوجود الخيوط أنه بما أنها لا ترتبط بأي موارد ، فإن إنشاؤها وحذفها أسهل من العمليات. يكون إنشاء خيط أسرع بمئة مرة من إنشاء عملية . وبهذا يمكن أن نجمل بعض الأسباب لاستخدام الخيوط:

1- تقاسم الموارد : حيث أن خيوط البرنامج الواحد تتشارك في أجزاء عدة كالذاكرة.

2- توفير الوقت : حيث يتم عمل عدد من الخيوط في نفس الوقت.

مثال: عندما نكتب برنامج (word) ، في وقت تشغيله يصبح عملية (process) وعندما نكتب فيه تكون هناك عمليات تعمل بنفس وقت عملية الكتابة ،

مثل تنفيذ spell check و save every two minutes هنا عمل شيئين في عملية واحدة.

3- العملية (process) مع الخيوط تجعل الخادم (serves) يقوم بعمله بفاعلية اكبر.

4- عند إيقافه أو إنهائه يأخذ وقت اقل من العمليات.

5- لا حاجة للاستعانة بنواة نظام التشغيل (kernel) أو إخبارها بأي عملية تتم عن طريق الخيط لأنه لا تعلم بوجودها.

مستويات الخيوط Level of Threads :

تشترك الخيوط في نفس العملية بالبرمجة (Code) والبيانات (Data) والموارد (Resources) .
وتستفرد بالسجل (Register) والمكدس (Stack) .

يوجد نوعين مختلفين للعمليات (Processes) وهما:

1- عمليات مفردة الخيوط (Single Thread).
2- عمليات متعددة الخيوط (.(Multi threads

هناك نوعان من الخيوط :

1- خيط على مستوى المستخدم (User level Threads) :

- بنية المعطيات للخيط معرفة ضمن منطقة الذاكرة الخاصة بعملية المستخدم.
- إنشاء وإدارة الخيط من مهام العملية نفسها بدون استخدام استدعاءات نظام، وبالتالي يتم ذلك بسرعة أكبر.
- تتقاسم الخيوط فيما بينها الشريحة الزمنية المخصصة للعملية.
- إذا حصل توقف إحدى الخيوط بسبب استدعاء نظام فإن العملية بالكامل تتوقف (تتوقف بالتالي جميع الخيوط).

2- خيط على مستوى النواة (Kernel level Threads) :

- بنية المعطيات للخيط معرفة ضمن منطقة الذاكرة الخاصة بنواة نظام التشغيل.
- إنشاء وإدارة الخيط من مهام نظام التشغيل (استدعاءات نظام).
- يحصل كل خيط على شريحة زمنية خاصة به.
- إذا حصل توقف إحدى الخيوط بسبب استدعاء نظام ما، فإن ذلك لا يؤثر على بقية الخيوط.

المعالجة المتعددة المنتظمة وغير المنتظمة Symmetric and Asymmetric Multiprocessing

- **المعالجة المتعددة المنتظمة** (Symmetric Multiprocessing) **:**

كل معالج له جدولة خاصة فيه ، بحيث أن كل معالج له قراراته وليس له علاقة بأي معالج آخر، حيث توضع نسخة واحدة من نظام التشغيل في ذاكرة مشتركة يمكن الوصول إليها من قبل باقي المعالجات.

المشاكل التي تواجه هذا النوع :

1- عندما يطلب معالجان أو أكثر بنفس الوقت التعامل مع نظام التشغيل الموجود في الذاكرة وهي من أهم المشاكل التي تواجه هذا النوع .

2- تحقيق التزامن بين المعالجات.

3- كيفية منع الاختناق (Deadlocks).

4- تنظيم المشاركة الزمنية (time slice).

- **المعالجة المتعددة غير المنتظمة** (Asymmetric Multiprocessing)**:**

معالج واحد فقط يملك كل قرارات الجدولة وعمليات الإدخال والإخراج ونشاطات النظام الأخرى ويسمى بالخادم (server) ، وبقية المعالجات الأخرى تنفذ المطلوب منها، هي تعتبر بسيطة فمعالج واحد فقط هو الذي يستطيع الوصول إلى تراكيب البيانات، حيث توجد نسخة واحدة فقط من نظام التشغيل على معالج (master) الذي يقوم بتنفيذ تعليمات نظام التشغيل، بينما تكون جميع المعالجات الأخرى هي معالجات تابعة (slaves) تنفذ فقط المهام الموكلة إليها من قبل المعالج (master). أما المشاكل التي تواجه هذا النوع:

1- تكمن في الحمل الزائد على المعالج (master).

2- كثرة الطلبات والاستدعاءات حيث تظهر مشكلة ما يسمى بعنق الزجاجة.

أسئلة اختبارية :

س1: ماذا تعني العملية ذات الثقل الخفيف Lightweight process ؟

س2: ما هي الفروقات بين العملية والخيط ؟

س3: ما هي الاسباب لاستخدام الخيط ؟

س4: ما هي انواع الخيوط بالاعتماد على المستويات ؟

س5: ما هي المعالجة المتعددة غير المنتظمة للخيوط ؟