(2)

إدارة العمليات Process Management

مقدمة

إدارة العمليات Process Management هي إحدى الوظائف الأساسية لنظام التشغيل، حيث تتيح تنظيم وتنفيذ البرامج بكفاءة على الحاسوب.

العملية Process هي وحدة التنفيذ الأساسية في نظام التشغيل، وتتطلب إدارتها تنسيقاً دقيقاً بين المعالج، الذاكرة، وأجهزة الإدخال/الإخراج، في هذه المحاضرة، نستعرض مفهوم العملية، حالاتها، إدارتها، الإجراءات التي تُجرى عليها، والخيوط أو النياسب Threads كمفهوم متقدم لتحسين الأداء.

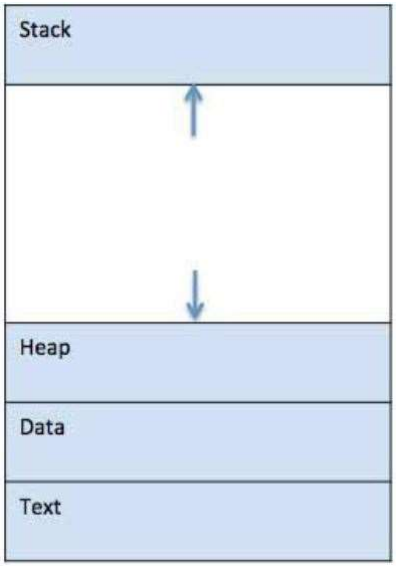
مفهوم العملية Process Concept

العملية هي البرنامج الذي يتم تنفيذه فعلياً على المعالج مع تخصيص الموارد اللازمة له، على خلاف البرنامج الخامل المخزن على القرص الصلب، فإن العملية لها طابع ديناميكي، وتتفاعل مع النظام أثناء التنفيذ.

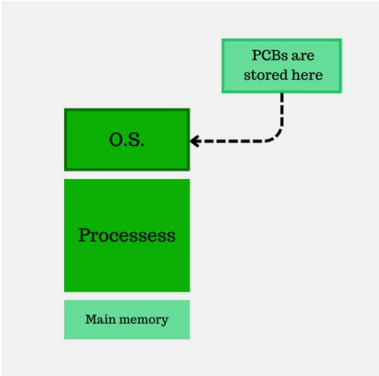
تتكون العملية من العناصر التالية:

1. **نص البرنامج Program Code** أو التعليمات البرمجية القابلة للتنفيذ، وتُسمى أحياناً "Text Segment"
2. **البيانات Data** هي المتغيرات المحلية أو العامة التي تستخدمها التعليمات البرمجية.
3. **الكومة Heap** منطقة الذاكرة المخصصة ديناميكياً للمتغيرات التي يتم تخصيصها أثناء التنفيذ.
4. **المكدس Stack** يحتوي على البيانات المؤقتة التي تستخدمها العملية (مثل معطيات الدوال).
5. **سجل العداد Program Counter** يشير إلى التعليمة التالية التي سيتم تنفيذها.
6. **سجلات المعالج Registers** تحتوي على البيانات المؤقتة التي يستخدمها المعالج أثناء تنفيذ العملية.

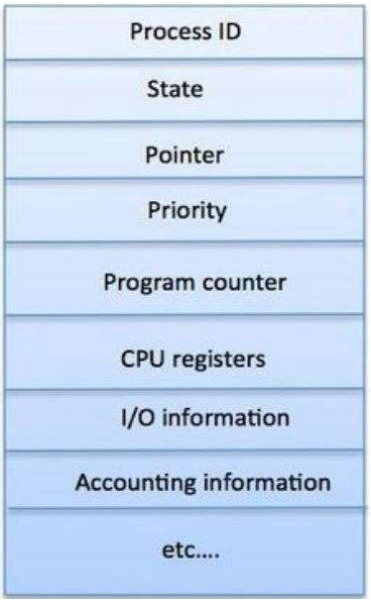
على سبيل المثال، عند تشغيل متصفح مثل Chrome ، يتم إنشاء عملية تحتوي على تعليمات المتصفح، بيانات المستخدم (مثل المواقع المفضلة) ، والموارد المخصصة (مثل الذاكرة) .



يبدأ المكدس والكومة من طرفي المساحة الحرة للعملية وينموان باتجاه بعضهما البعض، إذا التقيا، قد يحدث خطأ في تجاوز المكدس، أو قد يفشل طلب لتخصيص ذاكرة جديدة بسبب نقص الذاكرة المتاحة.



كل عملية لها معرف عملية Process ID يميزها، ويتم تخزين معلوماتها في Process Control Block PCB أو كتلة التحكم في العملية، وهي هيكل بيانات في نظام التشغيل تحتوي على:



- حالة العملية (جاهزة، قيد التشغيل، في الانتظار)
- المؤشر (يشير إلى العملية الأم)
- أولوية العملية (الأذونات والسماحيات)
- عداد البرنامج (يشير إلى عنوان الأمر التالي)
- سجلات المعالج (الحالات المخزنة بهدف تنفيذها عندما تصبح بحالة التشغيل.
- معلومات الإدخال/الإخراج (قائمة الأجهزة المخصصة للعملية)
- معلومات إدارة الذاكرة (حدود الذاكرة المخصصة)
- معلومات المحاسبة (مقدار استخدام المعالج).

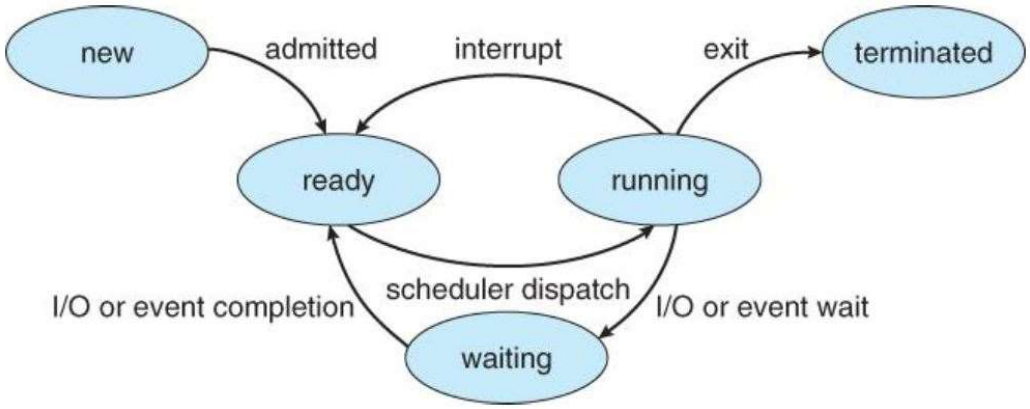
حالات العملية Process States

تمر العملية بعدة حالات أثناء دورة حياتها، وهذه الحالات تعكس وضعها في النظام، الحالات الرئيسية هي:

1. **جديدة New** العملية في مرحلة الإنشاء، حيث يتم تخصيص الموارد لها.
2. **جاهزة Ready** العملية جاهزة للتنفيذ وتنتظر تخصيص المعالج.
3. **قيد التنفيذ Running** العملية يتم تنفيذها فعلياً على المعالج.
4. **في انتظار Waiting/Blocked** العملية تنتظر حدثاً، مثلاً إكمال عملية إدخال/إخراج مثل قراءة ملف أو إشارة من عملية أخرى.
5. **منتهية Terminated** العملية اكتمل تنفيذها أو تم إيقافها، ويتم إزالتها من النظام.

الانتقال بين الحالات

- **من جديدة إلى جاهزة** عندما يتم إنشاء العملية وتخصيص الموارد.
- **من جاهزة إلى قيد التنفيذ** عندما يختار المجدول Scheduler العملية ويخصص لها المعالج.
- **من قيد التنفيذ إلى جاهزة** إذا تمت مقاطعة العملية مثل انتهاء الشريحة الزمنية في الجدولة .
- **من قيد التنفيذ إلى في انتظار** إذا احتاجت العملية إلى حدث خارجي.
- **من في انتظار إلى جاهزة** عند اكتمال الحدث المنتظر.
- **من قيد التنفيذ إلى منتهية** عند اكتمال العملية أو إيقافها.



أهمية إدارة العمليات

- **تحسين الأداء** من خلال تقليل وقت الانتظار وزيادة استخدام المعالج.
- **العدالة** ضمان تخصيص عادل للموارد بين العمليات.
- **الاستقرار** منع الأخطاء مثل التوقف Deadlock أو Race Condition

الإجراءات التي تتم على العمليات Operations on Processes

ينفذ نظام التشغيل مجموعة من الإجراءات لإدارة دورة حياة العملية وتشمل:

1. **إنشاء العملية Process Creation**
   - تحدث عند تشغيل برنامج أو استدعاء عملية فرعية، فيتم تخصيص كتلة التحكم PCB والموارد (ذاكرة، معالج).
   - مثال ذلك استدعاء fork في Linux لإنشاء عملية فرعية، أو CreateProcess في Windows
2. **إنهاء العملية Process Termination**
   - تحدث عند اكتمال العملية أو بسبب خطأ ما، مثلاً القسمة على صفر، فيتم تحرير الموارد وإزالة كتلة التحكم PCB .
   - مثال ذلك استدعاء exit في Linux أو إنهاء عملية عبر Task Manager في Windows .
3. **تعليق العملية Process Suspension**
   - يتم نقل العملية إلى حالة الانتظار مؤقتاً، مثل انتظار المدخلات من المستخدم.
   - يمكن استئنافها لاحقاً.
4. **تبديل السياق Context Switching**
   - عندما يتم إيقاف عملية قيد التنفيذ وتشغيل عملية أخرى يتم تنفيذ الإجراءات التالية:

- يحفظ النظام حالة العملية الحالية في كتلة التحكم PCB
- يستعيد حالة العملية الجديدة.
- يتطلب تبديل السياق وقتاً إضافياً، لذا يسعى النظام التقليل من عدد التبديلات.

5. **التواصل بين العمليات IPC InterProcess Communication**

- يتيح هذا الإجراء التنسيق أو تبادل البيانات بين للعمليات، ويتم تنفيذه بإحدى الآليات التالية:
  - **الأنابيب Pipes** لنقل البيانات باتجاه واحد.
  - **طوابير الرسائل Message Queues** لتبادل الرسائل.
  - **الذاكرة المشتركة Shared Memory** للوصول إلى منطقة ذاكرة مشتركة.
  - **الإشارات Signals** لإشعار العمليات بحدث معين.

6. **تغيير الأولوية Priority Adjustment**

- يمكن للنظام تعديل أولوية العملية بناءً على أهميتها أو متطلبات النظام.

7. **جدولة العمليات Scheduling** تحديد أي عملية ستحصل على المعالج ومتى، ويُستخدم لذلك خوارزميات مثل
   - **First Come, First Served** ويكون التنفيذ حسب ترتيب الوصول.
   - **Shortest Job Next** ويتم تنفيذ العملية ذات الوقت الأقصر أولاً.
   - **Round Robin** تخصيص شريحة زمنية لكل عملية بشكل دوري.
   - **Priority Scheduling** إعطاء الأولوية للعمليات ذات الأهمية العالية.

8. **إدارة التزامن Synchronization** لضمان عدم حدوث تعارض عندما تتشارك عدة عمليات بنفس الموارد مثل استخدام القفل Locks

9. **إدارة الانقطاعات Interrupts** التعامل مع الانقطاعات الناتجة عن أجهزة أو عمليات أخرى لضمان استجابة النظام.

إدارة العمليات في Windows

الويندوز هو نظام تشغيل يعتمد على نواة هجينة (Hybrid Kernel)، ويوفر واجهات رسومية وأدوات برمجية لإدارة العمليات، وتشمل إدارة العمليات في Windows إنشاء العمليات، جدولتها، إنهائها، التواصل فيما بينها، وإدارة الخيوط، ويوفر النظام أدوات مثل Task Manager وواجهات برمجة التطبيقات APIs

1. إنشاء العمليات

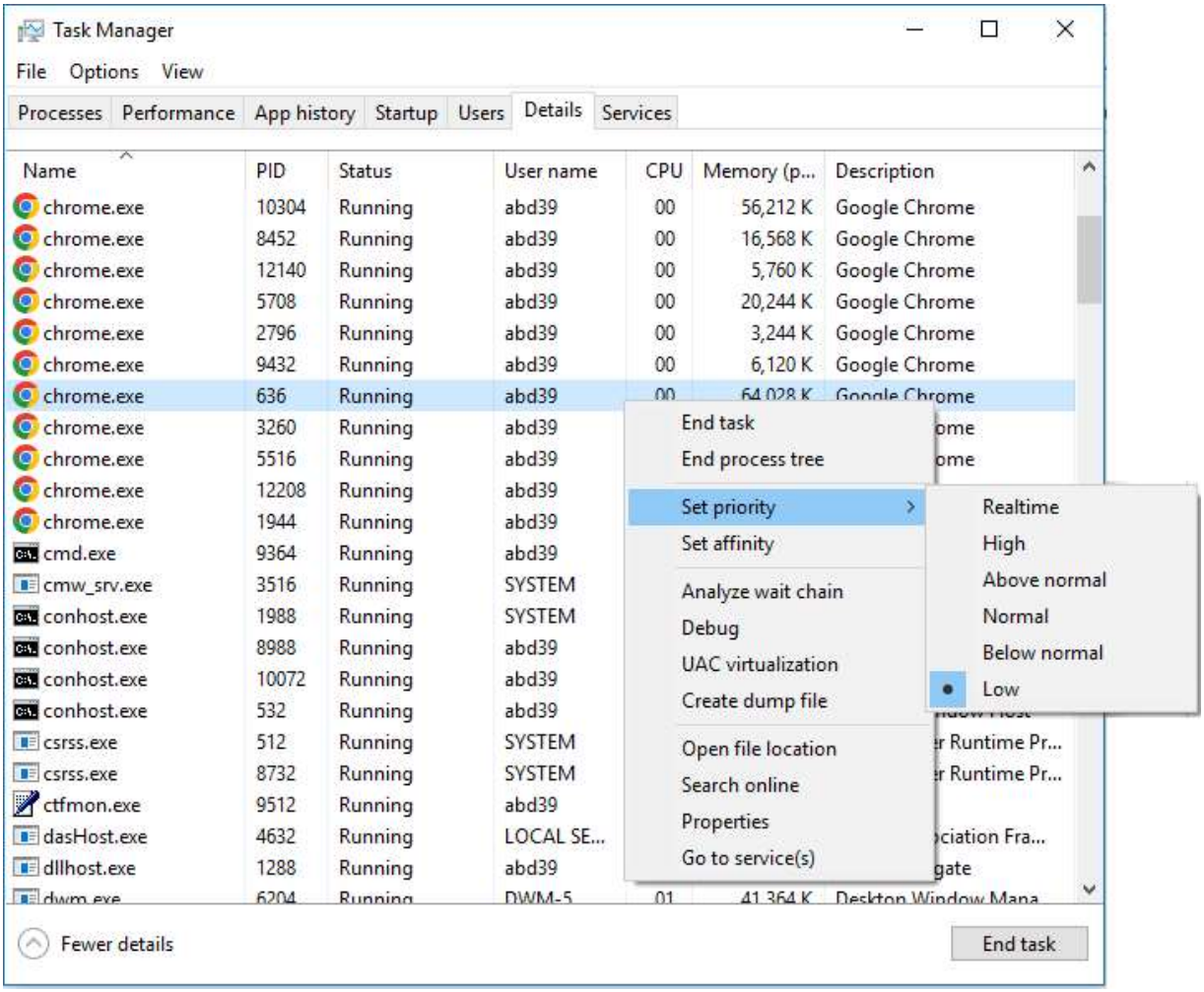
يتم إنشاء عملية في Windows عند تشغيل برنامج أو عند استدعاء عملية جديدة بواسطة عملية موجودة، يمكن إنشاء عملية باستخدام دالة CreateProcess المتوفرة في Win32 API.

2. جدولة العمليات

يستخدم Windows خوارزمية جدولة تعتمد على الأولوية (Priority-Based Scheduling) مع دعم الجدولة الدورية (Round Robin) للعمليات ذات الأولوية المتساوية، يمكن للمستخدم تعديل أولوية العمليات من خلال Task Manager

- لفتح إدارة العمليات اضغط (Ctrl + Shift + Esc).
- انتقل إلى علامة التبويب Details.
- انقر بزر الفأرة الأيمن على عملية (مثل chrome.exe)، واختر Set Priority (High أو Low).

مثال: إذا كان متصفح Chrome يستهلك موارد كبيرة، يمكن تقليل أولويته إلى "Below Normal" لإعطاء الأولوية لتطبيق آخر.



3. إنهاء العمليات

يمكن إنهاء العملية يدويًا من خلال Task Manager أو برمجيًا باستخدام دالة TerminateProcess.

مثال عبر Task Manager:

- افتح Task Manager.
- حدد عملية (مثل notepad.exe) من علامة التبويب Processes.
- انقر على End Task لإنهاء العملية.

أدوات إدارة العمليات في Windows

Task Manager: لعرض العمليات، الخيوط، استهلاك الموارد، وإنهاء العمليات.

Resource Monitor: لمراقبة استخدام (المعالج، الذاكرة، والقرص) لكل عملية.

PowerShell:

Get-Process عرض قائمة العمليات
Stop-Process إنهاء عملية

Command Prompt:

tasklist عرض العمليات
taskkill /IM notepad.exe /F إنهاء عملية

إدارة العمليات في Linux

Linux هو نظام تشغيل مفتوح المصدر يعتمد على نواة أحادية (Monolithic Kernel)، وتشمل إدارة العمليات في Linux على إنشاء العمليات، جدولتها، إنهائها، التواصل فيما بينها، ويوفر النظام سطر أوامر CLI، يمكن الوصول إليها باستخدام (CTRL+ALT+T) وواجهات برمجية لإدارة العمليات، تتميز إدارة العمليات في Linux بالمرونة والتحكم الدقيق.

1. إنشاء العمليات

يتم إنشاء العمليات في Linux، باستخدام استدعاءات النظام مثل fork وexec.

2. جدولة العمليات

يستخدم Linux خوارزمية جدولة متقدمة مثل Completely Fair Scheduler (CFS)، التي توزع وقت المعالج بناءً على الأولوية واستهلاك الموارد، يمكن تعديل الأولوية باستخدام أوامر مثل nice وrenice.

nice –n 10 firefox تشغيل Firefox بأولوية منخفضة

3. إنهاء العمليات

يمكن إنهاء العمليات باستخدام أوامر مثل kill أو pkill.

kill −9 1234 PID 1234 إنهاء العملية ذات رقم التعريف

الإشارة 9− تُنهي العملية فورًا دون السماح لها بحفظ البيانات

الخيوط أو النياسب Threads

الخيط أو النيسب Thread هو وحدة تنفيذ صغرى داخل العملية، ويُعرف أحياناً بـ العملية الخفيفة Lightweight Process ، تتشارك الخيوط داخل نفس العملية في نفس الموارد مثل الذاكرة والملفات المفتوحة، لكن لكل خيط (مكدس، عداد برنامج، وسجلات المعالج الخاصة به).

أنواع الخيوط

1. **خيوط مستوى المستخدم UserLevel Threads**
   - تُدار بواسطة مكتبات في التطبيق مثل POSIX Threads
   - سريعة، لأنها لا تتطلب تدخل النواة، لكن إذا توقف خيط، قد تتوقف العملية بأكملها.
2. **خيوط مستوى النواة KernelLevel Threads**
   - تُدار بواسطة نظام التشغيل.
   - أبطأ، بسبب تدخل النواة، لكنها توفر استقراراً أكبر لأن توقف خيط لا يؤثر على بقية الخيوط.

فوائد الخيوط

- **الأداء**: الخيوط أسرع في الإنشاء والتبديل مقارنة بالعمليات.
- **التزامن** : تتيح تنفيذ مهام متعددة داخل نفس العملية، مثل تحميل صفحة ويب وتشغيل فيديو في نفس المتصفح.
- **استغلال الموارد** : تقلل من استهلاك الذاكرة بسبب مشاركة الموارد.

- **التوسع:** تدعم الأنظمة متعددة المعالجات، حيث يمكن تشغيل خيوط على معالجات مختلفة.

تحديات الخيوط

- **التزامن** : يحتاج لإدارة الوصول إلى الموارد المشتركة باستخدام أدوات مثل القفل Mutex أو السيمافور Semaphore .
- **التعقيد:** تصميم التطبيقات متعددة الخيوط يتطلب خبرة لتجنب مشكلات مثل التوقف Deadlock

أمثلة عملية

- **متصفح الإنترنت** يستخدم خيوطاً متعددة لـ (تحميل الصفحات، عرض الصور، ومعالجة إدخال المستخدم).
- **خوادم الويب** مثل Apache، يتعامل كل خيط مع طلب مستخدم مختلف.
- **تطبيقات معالجة الصور** تُنفذ الخيوط بالتوازي على مهام مثل تطبيق الفلاتر.

إن فهم العملية وحالاتها، واسلوب إدارتها، والإجراءات التي تنفذ عليها، يسمح بإدراك التعقيدات التي تتطلبها الأنظمة الحديثة.

وفهم تقنيات إدارة العمليات مثل (الجدولة، التواصل بين العمليات، واستخدام الخيوط) ، يساعد على فهم التطبيقات المتقدمة متعددة المهام.

ننصح بتجربة أدوات مثل Linux Process Viewer لاختبار هذه المفاهيم عملياً.