# OPERATING SYSTEM

## *Lecture Notes*

*Dr. Professor, J.M. Khalifeh*

قسم المعلوماتية

الوحدة الثانية

**النسخة العربية**

ملاحظة هامة: النسخة الأساسية هي النسخة الإنكليزية

# Unit-3

# Process & Thread

## ملخص

سمحت أجهزة الكمبيوتر القديمة بتنفيذ برنامج واحد فقط في كل مرة. كان لهذا البرنامج سيطرة كاملة على النظام وكان بإمكانه الوصول إلى جميع موارد النظام. في المقابل، تسمح أنظمة الكمبيوتر المعاصرة بتحميل برامج متعددة في الذاكرة وتنفيذها بشكل متزامن. تطلب هذا التطور تحكمًا أقوى ومزيدًا من التجزئة للبرامج المختلفة؛ وقد أدت هذه الاحتياجات إلى نشوء فكرة العملية، وهي عبارة عن برنامج قيد التنفيذ. العملية هي وحدة العمل في نظام الحوسبة الحديث.

كلما كان نظام التشغيل أكثر تعقيدًا، كلما كان من المتوقع منه القيام بمهام أكثر نيابة عن مستخدميه. على الرغم من أن مهمة نظم التشغيل الرئيسية هي تنفيذ برامج المستخدم، إلا أنها تحتاج أيضًا إلى الاهتمام بمهام النظام المختلفة التي يتم إجراؤها بشكل أفضل في مجال المستخدم، بدلاً من داخل النواة. لذلك يتكون النظام من مجموعة من العمليات، بعضها يقوم بتنفيذ كود المستخدم، والبعض الآخر يقوم بتنفيذ كود نظام التشغيل. من المحتمل أن يتم تنفيذ جميع هذه العمليات بشكل متزامن، مع مضاعفة وحدة المعالجة المركزية (أو وحدات المعالجة المركزية) فيما بينها. في هذه الوحدة، سوف تقرأ عن ماهية العمليات وكيف يتم تمثيلها في نظام التشغيل وكيف تعمل.

نموذج العملية المقدم هنا يفترض أن العملية هي عبارة عن برنامج تنفيذي مع مسلك واحد للتحكم. ومع ذلك، فإن جميع أنظمة التشغيل الحديثة تقريبًا توفر ميزات تمكّن العملية من احتواء مسالك تحكم متعددة. أصبح تحديد فرص التوازي من خلال استخدام المسالك ذات أهمية متزايدة للأنظمة الحديثة متعددة النواة وتلك التي توفر وحدات معالجة مركزية متعددة. هنا، نقدم العديد من المفاهيم، بالإضافة إلى التحديات، المرتبطة بأنظمة الكمبيوتر ذات المعالجات متعددة النوى، ونستكشف كيف تدعم أنظمة تشغيل Windows و Linux مسالك التنفيذ على مستوى kernel.

## أهداف الوحدة

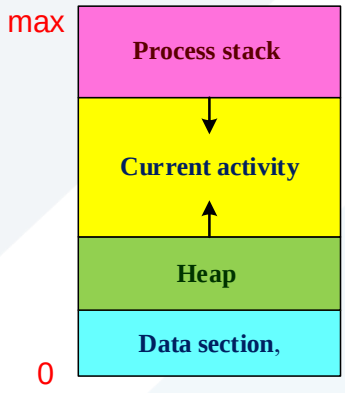
- تحديد المكونات المنفصلة للعملية وتوضيح كيفية تمثيلها وجدولتها في نظام التشغيل.
- وصف كيفية إنشاء العمليات وإنهاؤها في نظام التشغيل، بما في ذلك تطوير البرامج باستخدام استدعاءات النظام المناسبة التي تؤدي هذه العمليات.
- وصف ومقارنة الاتصال بين العمليات باستخدام الذاكرة المشتركة وتمرير الرسائل.
- التعرف على المكونات الأساسية للمسلك.
- وصف الفوائد الرئيسية والتحديات الكبيرة لتصميم عمليات متعددة المسالك.

## مفهوم العملية

العملية هي في الأساس البرنامج قيد التنفيذ. يجب أن يتقدم تنفيذ العملية بطريقة متسلسلة. يتم تعريف العملية على أنها كيان يمثل وحدة العمل الأساسية التي سيتم تنفيذها في النظام.

تكتب برامج الكمبيوتر الخاصة بنا في ملف نصي لوضعها بعبارات بسيطة، وعندما نقوم بتنفيذ هذا البرنامج، تصبح عملية تؤدي جميع المهام المذكورة في البرنامج.

عندما يتم تحميل برنامج في الذاكرة ويصبح عملية، يمكن تقسيمه إلى أربعة أقسام: مكدس، كومة، نص وبيانات. تُظهر الصورة التالية تخطيطًا مبسطًا لعملية داخل الذاكرة الرئيسية:




## وصف المكونات

### المكدس

يحتوي المكدس على البيانات المؤقتة مثل الوظيفة وعنوان المرسل والمتغيرات المحلية.

### كومة

يتم تخصيص هذه الذاكرة ديناميكيًا لعملية أثناء وقت تشغيلها.

### نص

يتضمن هذا النشاط الحالي الذي تمثله قيمة عداد البرامج ومحتويات سجلات المعالج.

### بيانات

يحتوي هذا القسم على المتغيرات العامة والثابتة.

إن برنامج الكمبيوتر هو عبارة عن مجموعة من التعليمات التي تؤدي مهمة محددة عند تنفيذها بواسطة الكمبيوتر. عندما نقارن برنامجًا بعملية، يمكننا أن نستنتج أن العملية كمثيل ديناميكي لبرنامج كمبيوتر.

## حالات العملية

عندما يتم تنفيذ العملية، فإنها تمر عبر حالات مختلفة. قد تختلف هذه المراحل في أنظمة التشغيل المختلفة، كما أن أسماء هذه الحالات ليست موحدة.

بشكل عام، يمكن أن تشتمل العملية على إحدى الحالات الخمس التالية في كل مرة.

### البدء

هذه هي الحالة الأولية عند بدء / إنشاء العملية لأول مرة.

### الجاهزية

تنتظر العملية أن يتم تخصيصها للمعالج. تنتظر العمليات الجاهزة تخصيص المعالج لها بواسطة نظام التشغيل حتى يمكن تشغيلها. قد تنتقل العملية إلى هذه الحالة بعد حالة البدء أو أثناء تشغيلها إذا تمت مقاطعتها بواسطة المجدول لتخصيص وحدة المعالجة المركزية لبعض العمليات الأخرى.

### التنفيذ

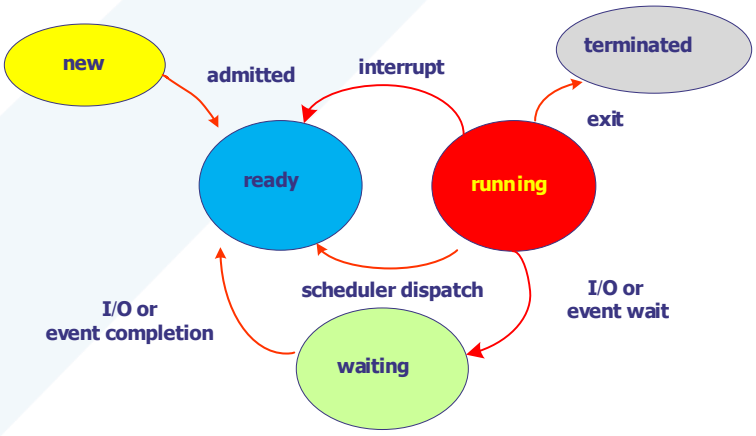
بمجرد تخصيص العملية بوقت المعالج بواسطة برنامج جدولة نظام التشغيل، يتم تعيين حالة العملية على التنفيذ ويقوم المعالج بتنفيذ تعليماته.

### الانتظار

تنتقل العملية إلى حالة الانتظار إذا احتاجت إلى انتظار مورد ما، مثل انتظار إدخال قيم من المستخدم أو انتظار توفر الملف.

### الإنهاء أو الخروج

بمجرد أن تنتهي العملية من تنفيذها، أو يتم إنهاؤها بواسطة نظام التشغيل، يتم نقلها إلى الحالة المنتهية حيث تنتظر إزالتها من الذاكرة الرئيسية.

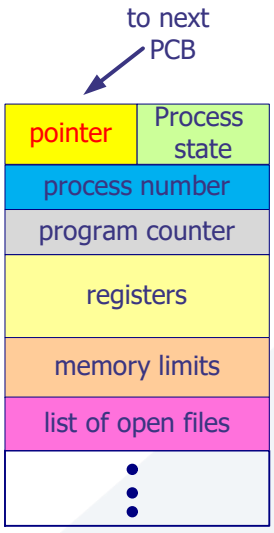



### كتلة التحكم في العمليات (PCB)

كتلة التحكم في العمليات هي بنية معطيات يحتفظ بها نظام التشغيل لكل عملية. يتم تحديد PCB بواسطة معرف العملية (PID) وهو عدد صحيح. يحتفظ PCB بجميع المعلومات اللازمة لتتبع العملية كما هو موضح أدناه في الجدول.

تعتمد بنية PCB بشكل كامل على نظام التشغيل وقد تحتوي على معلومات مختلفة في أنظمة تشغيل مختلفة.

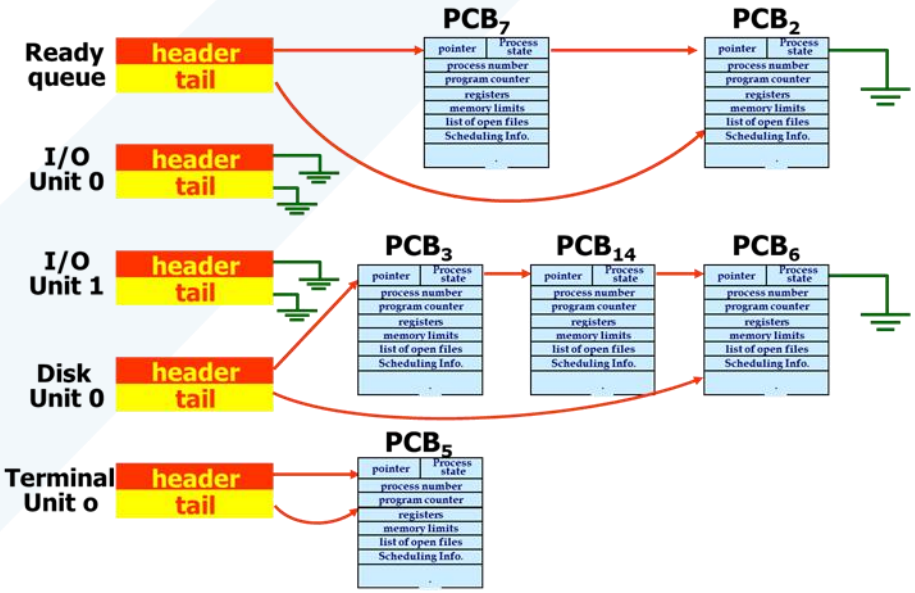
هنا رسم تخطيطي مبسط لكتلة التحكم في العمليات (PCB)




- **المؤشر:** هو مؤشر مطلوب حفظه عند تبديل العملية من حالة إلى أخرى للاحتفاظ بالوضع الحالي للعملية.
- **حالة العملية:** يخزن الحالة الخاصة بالعملية.
- **رقم العملية:** يتم تعيين كل عملية بمعرف فريد يُعرف باسم معرف العملية أو PID.
- **عداد البرنامج:** يخزن العداد الذي يحتوي على عنوان التعليمات التالية التي سيتم تنفيذها للعملية.
- **المسجلات:** هذه هي سجلات وحدة المعالجة المركزية التي تشمل: المجمع، والسجلات الأساسية، وسجلات الأغراض العامة.
- **حدود الذاكرة:** يحتوي هذا الحقل على معلومات حول نظام إدارة الذاكرة المستخدم بواسطة نظام التشغيل. قد يشمل ذلك جداول الصفحات وجداول الأجزاء وما إلى ذلك.
- **قائمة الملفات المفتوحة:** تتضمن هذه المعلومات قائمة الملفات المفتوحة لعملية ما.

يتم الاحتفاظ بـ PCB لعملية ما طوال عمرها، ويتم حذفها بمجرد انتهاء العملية.

## جدولة العمليات




جدولة العملية هي نشاط إدارة العملية الذي يتولى إزالة العملية الجارية من وحدة المعالجة المركزية واختيار عملية أخرى على أساس إستراتيجية معينة.

تعد جدولة العمليات جزءًا أساسيًا من أنظمة التشغيل التي تسمح بتحميل أكثر من عملية واحدة في الذاكرة لتنفيذها في وقت واحد، وتتشارك العمليات المحملة وحدة المعالجة المركزية باستخدام التجميع الزمني.

## أنواع الجدولة

هناك نوعان من الجدولة:

### جدولة غير استيلائية:

هنا لا يمكن أخذ المورد من العملية حتى تكمل العملية التنفيذ. يحدث تبديل الموارد عندما تنتهي العملية الجارية وتنتقل إلى حالة الانتظار.

### جدولة استيلائية:

هنا يخصص نظام التشغيل الموارد لعملية لفترة زمنية محددة. أثناء تخصيص الموارد، تنتقل العملية من حالة التشغيل إلى حالة الجاهزية أو من حالة الانتظار إلى حالة الجاهزية. يحدث هذا التبديل لأن وحدة المعالجة المركزية قد تعطي الأولوية للعمليات الأخرى وتستبدل العملية بأولوية أعلى بالعملية الجارية.

## قوائم انتظار جدولة العملية

يحتفظ نظام التشغيل بجميع كتل التحكم في العمليات (PCBs) في قوائم انتظار جدولة العمليات. حيث يحتفظ بقائمة انتظار منفصلة لكل حالة من حالات العملية ويتم PCB لجميع العمليات في نفس حالة التنفيذ في نفس قائمة الانتظار. عندما يتم تغيير حالة العملية، يتم إلغاء ربط PCB الخاص بها من قائمة الانتظار الحالية وينتقل إلى قائمة انتظار الحالة الجديدة.

يحافظ نظام التشغيل على قوائم انتظار جدولة العمليات المهمة التالية:

### قائمة انتظار العمل:

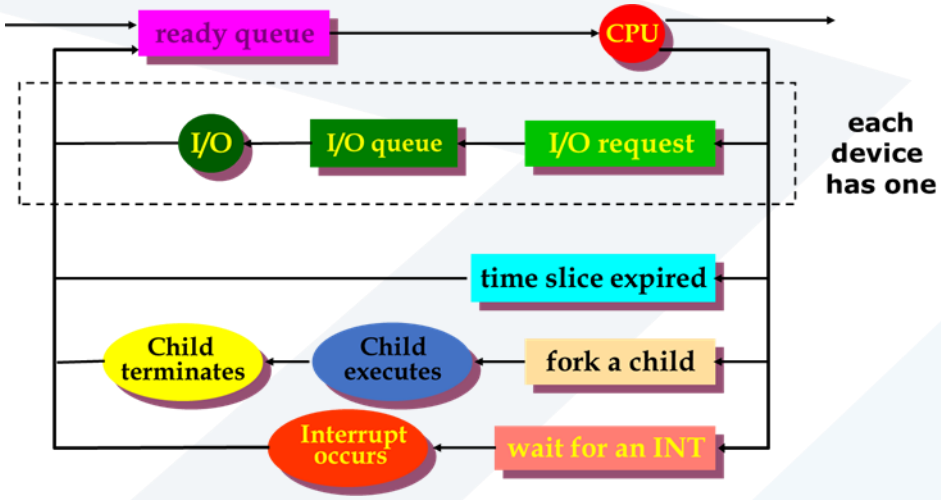
قائمة الانتظار هذه تحافظ على جميع العمليات في النظام.

### قائمة الانتظار الجاهزة:

تحتفظ قائمة الانتظار هذه بمجموعة من جميع العمليات الموجودة في الذاكرة الرئيسية، جاهزة وتنتظر التنفيذ. يتم وضع عملية جديدة دائمًا في قائمة الانتظار هذه.

### قوائم انتظار الأجهزة:

تشكل العمليات التي تم حظرها بسبب عدم توفر جهاز الإدخال / الإخراج قائمة الانتظار هذه.




يمكن لنظام التشغيل استخدام خوارزميات مختلفة لإدارة كل قائمة انتظار (FIFO، Round Robin، الأولوية، وما إلى ذلك). يحدد برنامج جدولة نظام التشغيل كيفية نقل العمليات بين قوائم الانتظار الجاهزة واقعة التشغيل والتي يمكن أن يكون لها إدخال واحد فقط لكل نواة معالج على النظام.

### نموذج العملية ثنائي الحالة

يشير نموذج العملية ثنائي الحالة إلى الحالات قيد التشغيل وغير قيد التنفيذ الموضحة أدناه:

### قيد التنفيذ

أي عندما يتم إنشاء عملية جديدة، وتدخل في حالة التنفيذ.

### خارج التنفيذ

يتم الاحتفاظ بالعمليات التي هي خارج التنفيذ في قائمة الانتظار، في انتظار دورها للتنفيذ. يتم تطبيق جدول الانتظار باستخدام القائمة المترابطة. عند مقاطعة عملية ما، يتم نقل هذه العملية الى قائمة الانتظار. إذا اكتملت العملية أو تمت مقاطعتها. في كلتا الحالتين، يقوم المرسل بعد ذلك بتحديد عملية أخرى من قائمة الانتظار ليتم تنفيذها.

### المجدولات

المجدول هم برنامج خاص من برامج النظام يتعامل مع جدولة العمليات بطرق مختلفة. مهمته الرئيسية هي تحديد الوظائف التي سيتم تقديمها في النظام وتحديد العملية التي سيتم تشغيلها. تكون المجدولات على ثلاثة أنواع:

جدولة طويلة المدى

جدولة قصيرة المدى

جدولة متوسطة المدى

## جدولة طويلة المدى

وتسمى أيضًا بجدولة الوظائف. يحدد المجدول طويل المدى البرامج التي يتم قبولها في النظام للمعالجة. يقوم بتحديد العمليات من قائمة الانتظار وتحميلها في الذاكرة للتنفيذ. يتم تحميل العملية في الذاكرة لجدولة وحدة المعالجة المركزية.

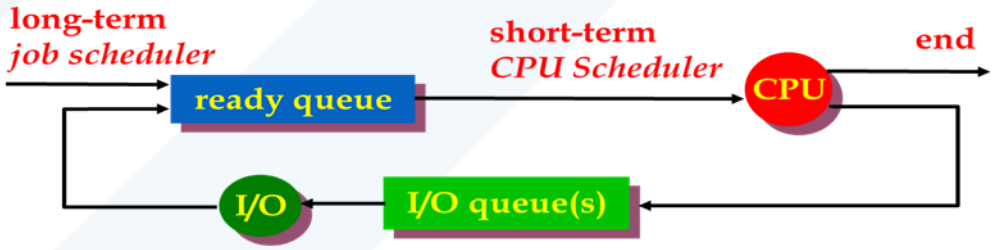
الهدف الأساسي لجدولة الوظائف هو توفير مزيج متوازن من الوظائف، مثل ربط الإدخال / الإخراج والمعالج. كما يتحكم في درجة البرمجة المتعددة. إذا كانت درجة البرمجة المتعددة مستقرة، فيجب أن يكون متوسط معدل إنشاء العملية مساويًا لمتوسط معدل المغادرة للعمليات التي تغادر النظام.

في بعض الأنظمة، قد لا يكون المجدول طويل الأجل متاحًا أو متاحا بشكل قليل. لا تحتوي أنظمة تشغيل مشاركة الوقت على برنامج جدولة طويل المدى. عندما تغير العملية الحالة من جديد إلى جاهز، يكون هناك استخدام جدولة طويلة المدى.

## جدولة قصيرة المدى

ويسمى أيضًا باسم جدولة وحدة المعالجة المركزية. هدفها الرئيسي هو زيادة أداء النظام وفقًا لمجموعة المعايير المختارة. إنه تغيير حالة الاستعداد إلى حالة تشغيل العملية. يحدد برنامج جدولة وحدة المعالجة المركزية عملية من بين العمليات الجاهزة للتنفيذ ويخصص وحدة المعالجة المركزية لإحدى هذه العمليات.

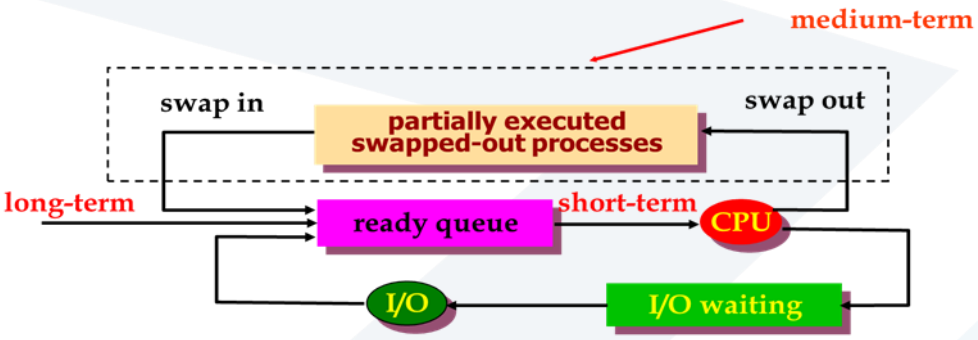
يقوم مجدول المدى القصير باتخاذ قرار بشأن العملية التي سيتم تنفيذها بعد ذلك. المجدول على المدى القصير أسرع من المجدول على المدى الطويل.




## جدولة متوسطة المدى

تشكل الجدولة متوسطة المدى جزءا من عملية المبادلة بين العمليات حيث تتم إزالة العمليات من الذاكرة . وتقلل من درجة البرمجة المتعددة. يعتبر المجدول على المدى المتوسط هو المسؤول عن التعامل مع العمليات الخارجية المبادلة.

قد يتم تعليق العملية الجارية إذا قدمت طلب إدخال / إخراج. لا يمكن للعمليات المعلقة إحراز أي تقدم في انجازها نحو الانتهاء. في هذه الحالة، لإزالة العملية من الذاكرة وإفساح المجال للعمليات الأخرى، يتم نقل العملية المعلقة إلى الحفظ. تسمى هذه العملية بالمبادلة، ويقال أن العملية يتم تبديلها.




## Comparison among Scheduler

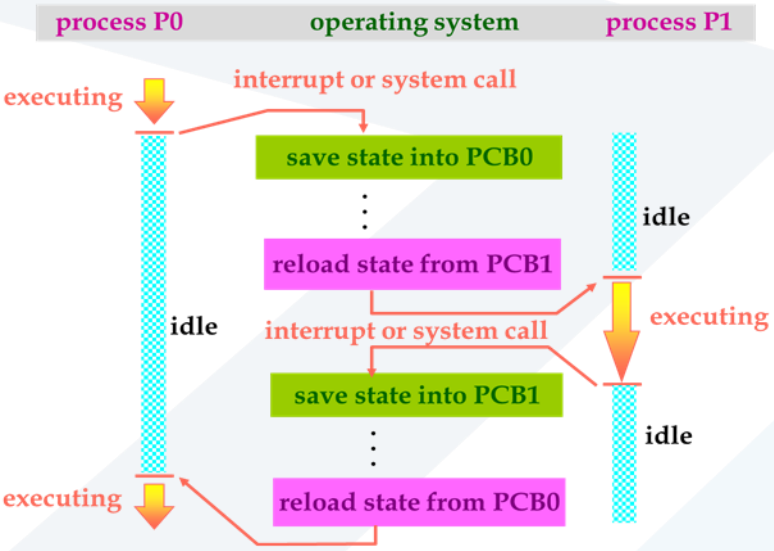
| جدولة طويلة الأجل | جدولة قصيرة المدى | جدولة متوسطة المدى |
|---|---|---|
| إنه برنامج جدولة للأعمال ككل | إنه جدولة وحدة المعالجة المركزية | هي جدولة عمليات التبديل. |
| السرعة أقل من جدولة المدى القصير | السرعة هي الأسرع بين الاثنين الآخرين | السرعة بين كل من جدولة المدى القصير والطويل. |
| يتحكم في درجة البرمجة المتعددة | يوفر تحكمًا أقل في درجة البرمجة المتعددة | يقلل من درجة البرمجة المتعددة. |
| يكاد يكون غائبًا أو ضئيلًا في نظام مشاركة الوقت | هو أيضا الحد الأدنى في نظام تقاسم الوقت | إنه جزء من أنظمة مشاركة الوقت. |
| يقوم بتحديد العمليات من التجمع وتحميلها في الذاكرة للتنفيذ | يختار تلك العمليات التي تكون جاهزة للتنفيذ | يمكنه إعادة تقديم العملية في الذاكرة ويمكن متابعة التنفيذ. |

## Context Switching تبديل السياق

تبديل السياق هو آلية لتخزين واستعادة حالة أو سياق وحدة المعالجة المركزية في كتلة التحكم في العملية بحيث يمكن استئناف تنفيذ العملية من نفس النقطة في وقت لاحق. باستخدام هذه التقنية، يتيح مبدل السياق لعمليات متعددة مشاركة وحدة معالجة مركزية واحدة. يعد تبديل السياق جزءًا أساسيًا من ميزات نظام التشغيل متعدد المهام.

عندما يقوم المجدول بتبديل وحدة المعالجة المركزية من تنفيذ عملية واحدة لتنفيذ عملية أخرى، يتم تخزين حالة عملية التشغيل الحالية في كتلة التحكم في العملية. بعد ذلك، يتم تحميل حالة العملية التي سيتم تشغيلها بعد ذلك من PCB الخاص بها واستخدامها لتخصيص الكمبيوتر، والسجلات، وما إلى ذلك. عند هذه النقطة، يمكن أن تبدأ العملية الثانية في التنفيذ.

تعد عملية تبديل السياق عملية حسابية هامة حيث يجب حفظ التسجيل وحالة الذاكرة واستعادتها. لتقليل وقت تبديل السياق، تستخدم بعض أنظمة الأجهزة مجموعتين أو أكثر من سجلات المعالج. عند تبديل العملية، يتم تخزين المعلومات التالية لاستخدامها لاحقًا.




عداد البرنامج

معلومات الجدولة

قيمة التسجيل الأساسية والحد

التسجيل المستخدم حاليا

الحالة المتغيرة

معلومات حالة I / O

وغيرها

## مراحل تشغيل العمليات

### 1. الانشاء:

هذه هي الخطوة الأولى لنشاط تنفيذ العملية. إنشاء العملية يعني إنشاء عملية جديدة للتنفيذ. قد يتم تنفيذ ذلك عن طريق النظام أو المستخدم أو العملية القديمة نفسها. هناك العديد من الأحداث التي تؤدي إلى إنشاء العملية. بعض هذه الأحداث هي التالية:

- عندما نبدأ تشغيل الكمبيوتر، يقوم النظام بإنشاء العديد من العمليات في الخلفية.
- قد يطلب المستخدم إنشاء عملية جديدة.
- يمكن للعملية إنشاء عملية جديدة بنفسها أثناء التنفيذ.

### 2. الجدولة / التوزيع:

الحدث أو النشاط الذي يتم فيه تغيير حالة العملية من جاهزة للتشغيل. هذا يعني أن نظام التشغيل يضع العملية من حالة الاستعداد في حالة التشغيل. يتم الإرسال بواسطة نظام التشغيل عندما تكون الموارد متاحة أو تكون للعملية أولوية أعلى من العملية الجارية. هناك العديد من الحالات الأخرى التي يتم فيها استباق العملية في حالة التشغيل ويتم إرسال العملية في حالة الاستعداد بواسطة نظام التشغيل.

### 3. المنع:

عندما تستدعي عملية ما اجراء نظام الإدخال والإخراج لمنع العملية. وضع الكتلة هو في الأساس وضع تنتظر فيه العملية الإدخال والإخراج. ومن ثم بناءً على طلب العملية نفسها، يقوم نظام التشغيل بحظر العملية وإرسال عملية أخرى إلى المعالج. ومن ثم، في عملية منع العملية، يضع نظام التشغيل العملية في حالة "انتظار.

### 4. الاستيلاء:

عند حدوث مهلة، فهذا يعني أن العملية لم يتم إنهاؤها في الفترة الزمنية المخصصة وأن العملية التالية جاهزة للتنفيذ، فإن نظام التشغيل يستبق العملية. هذه العملية صالحة فقط عندما تدعم جدولة وحدة المعالجة المركزية الإجراءات الوقائية. يحدث هذا بشكل أساسي في جدولة الأولوية حيث يتم استباق العملية الجارية عند وصول العملية ذات الأولوية العالية. ومن ثم، في عملية استباق العملية، يضع نظام التشغيل العملية في حالة "جاهز".

### 5. الإنهاء:

إنهاء العملية هو نشاط إنهاء العملية. بمعنى آخر، إنهاء العملية هو استعادة موارد الكمبيوتر التي تتطلبها عملية التنفيذ. مثل الإنشاء. في الإنهاء قد يكون هناك العديد من الأحداث التي قد تؤدي إلى إنهاء العملية. البعض منهم:

- اكتمال تنفيذ العملية بشكل كامل وتشير لنظام التشغيل إلى أنه قد انتهى.
- نظام التشغيل نفسه ينهي العملية بسبب أخطاء الخدمة.
- قد تكون هناك مشكلة في الأجهزة التي تنهي العملية.
- يمكن إنهاء إحدى العمليات بعملية أخرى.

### التواصل بين العمليات

قد تكون العمليات التي يتم تنفيذها بشكل متزامن في نظام التشغيل إما عمليات مستقلة أو عمليات متعاونة. تكون العملية مستقلة إذا لم تشارك البيانات مع أي عمليات أخرى يتم تنفيذها في النظام. تتعاون العملية إذا كان من الممكن أن تؤثر أو تتأثر بالعمليات الأخرى التي يتم تنفيذها في النظام. من الواضح أن أي عملية تشارك البيانات مع العمليات الأخرى هي عملية تعاونية.

هناك عدة أسباب لتوفير بيئة تسمح بالتعاون في العمليات:

### مشاركة المعلومات.

نظرًا لأن العديد من التطبيقات قد تكون مهتمة بنفس جزء المعلومات (على سبيل المثال، النسخ واللصق)، يجب علينا توفير بيئة للسماح بالوصول المتزامن إلى هذه المعلومات.

### تسريع الحساب.

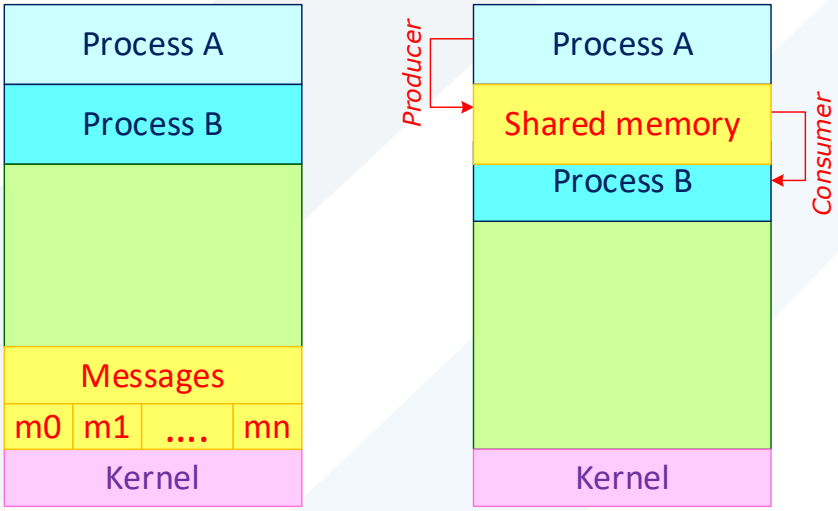
إذا أردنا تشغيل مهمة معينة بشكل أسرع، يجب علينا تقسيمها إلى مهام فرعية، سيتم تنفيذ كل منها بالتوازي مع المهام الأخرى. لاحظ أنه لا يمكن تحقيق مثل هذا التسريع إلا إذا كان الكمبيوتر يحتوي على نوى معالجة متعددة.

## نمطية.

قد نرغب في بناء النظام بطريقة معيارية، وتقسيم وظائف النظام إلى عمليات أو سلاسل عمليات منفصلة، كما ناقشنا في الفصل 2.

تتطلب العمليات التعاونية آلية اتصال بين العمليات (IPC) تسمح لها بتبادل البيانات – أي إرسال البيانات واستقبال البيانات من بعضها البعض.

## طرق الاتصال بين العمليات




تكون العملية مستقلة إذا لم تؤثر أو تتأثر بالعمليات الأخرى ولا تشارك البيانات مع العمليات الأخرى. عملية تعاون إذا كان من الممكن أن تتأثر أو تؤثر على العمليات الأخرى. يشارك البيانات مع العمليات الأخرى.

تحتاج عملية التعاون إلى آلية اتصال بين العمليات (IPC). هناك نوعان من نماذج IPC.

1. الذاكرة المشتركة: تم إنشاء منطقة مشتركة من الذاكرة لتبادل البيانات.
2. تمرير الرسائل: التواصل باستخدام تبادل الرسائل.

## نظام الذاكرة المشتركة

يحتاج نظام الذاكرة المشتركة لـ IPC إلى إنشاء منطقة ذاكرة مشتركة من خلال عمليات الاتصال. بعض خصائص منطقة الذاكرة المشتركة هي:

- تتواجد الذاكرة المشتركة في مساحة العنوان للعملية التي تنشئ منطقة الذاكرة المشتركة.
- يجب أن تضيف العملية أخرى مساحة عنوان مشتركة إلى مساحة العنوان الخاصة بهم للتواصل.
- يسمح نظام التشغيل بالوصول إلى مساحة العنوان من خلال عملية أخرى، ويجب أن تزيل العمليات هذا التقييد قبل إنشاء منطقة ذاكرة مشتركة.
- العمليات مسؤولة عن الموقع، والإنشاء، والوصول إلى سبب الذاكرة المشتركة. لا يتحكم نظام التشغيل في منطقة الذاكرة المشتركة.

### تمرير الرسالة

يمكن أن تتواصل العمليات المتعاونة بمساعدة تمرير الرسائل. هناك طرق مختلفة لحدوث الاتصال:

- بين نفس مسالك العملية.
- بين العمليات على نفس العقدة أو الكمبيوتر.
- بين عمليتين على عقد أو أجهزة كمبيوتر مختلفة.

مثال: نظام دردشة عبر الإنترنت

يتضمن نظام تمرير الرسائل عمليتين بدائيتين على الأقل:

- إرسال رسالة
- تلقي رسالة

يمكن أن تكون الرسائل ذات حجم ثابت أو متغير الحجم. إذا كان النظام يستخدم رسائل ذات حجم ثابت، فسيكون التنفيذ على مستوى النظام أمرًا سهلاً. إذا تم اختيار رسائل ذات حجم متغير، فسيكون تنفيذ مستوى النظام صعبًا، ولكن مهمة مستوى البرمجة سهلة.

إذا أرادت عملية P و Q التواصل، فيجب أن يكون هناك رابط اتصال بينهما. هناك عدة طرق لتنفيذ الارتباط المادي (الذاكرة المشتركة أو الأجهزة أو الشبكة)، لكننا مهتمون بالتنفيذ المنطقي للرابط. فيما يلي طرق مختلفة للتواصل باستخدام تمرير الرسائل:

1. الاتصال المباشر أو غير المباشر
2. الاتصال المتزامن أو غير المتزامن
3. تخزين مؤقت تلقائي أو صريح

## تعدد المسالك Threads (غير مطلوب في الامتحان الثاني سيتم شرحه لاحقاً)

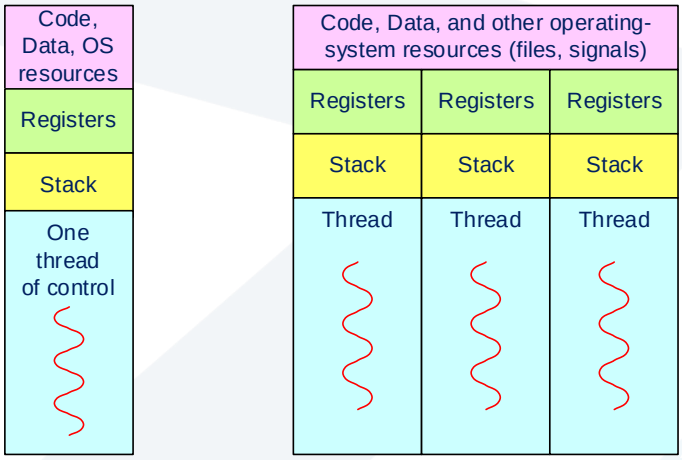
### ما هو المسلك؟

المسلك هو تدفق للتنفيذ من خلال كود العملية، مع عداد البرنامج الخاص به الذي يتتبع التعليمات التي يجب تنفيذها بعد ذلك، وسجلات النظام التي تحتوي على متغيرات العمل الحالية، والمكدس الذي يحتوي على محفوظات التنفيذ.

يشارك المسلك مع نظرائه بعض المعلومات مثل مقطع التعليمات البرمجية وقطاع البيانات والملفات المفتوحة. عندما يغير المسلك الذاكرة المرتبطة بتنفيذ مقطع برمجي ما، فإن كل المسالك الأخرى ترى ذلك.

يسمى المسلك أيضًا عملية خفيفة الوزن. توفر المسالك طريقة لتحسين أداء التطبيق من خلال التوازي. تمثل المسالك طريقة برمجية لتحسين أداء نظام التشغيل عن طريق تقليل المسلك العلوي وهو ما يعادل العملية الكلاسيكية.

كل مسلك ينتمي إلى عملية واحدة بالضبط ولا يمكن أن يوجد مسلك خارج العملية. يمثل كل مسلك تدفق منفصل للتحكم. تم استخدام المسالك بنجاح في تنفيذ خوادم الشبكة وخادم الويب. كما أنها توفر أساسًا مناسبًا للتنفيذ المتوازي للتطبيقات على معالجات الذاكرة المشتركة المتعددة. يوضح الشكل التالي عمل عملية أحادية المسالك ومتعددة المسالك.




## الفرق بين العملية والمسلك

| العملية | المسلك |
|---|---|
| العملية ثقيلة التنفيذ أو كثيفة الموارد. | المسلك مرن التنفيذ، ويستهلك موارد أقل من العملية. |
| يحتاج تبديل العمليات إلى التفاعل مع نظام التشغيل. | لا يحتاج تبديل المسلك إلى التفاعل مع نظام التشغيل. |
| تنفذ كل عملية في بيئات المعالجة المتعددة، نفس الكود ولكن لها ذاكرة وموارد ملفات خاصة بها. | يمكن لجميع سلاسل الرسائل مشاركة نفس مجموعة الملفات المفتوحة والعمليات التابعة. |
| إذا تم حظر إحدى العمليات، فلا يمكن تنفيذ أي عملية أخرى حتى يتم إلغاء حظر العملية الأولى. | أثناء حظر أحد المسالك وانتظاره، يمكن تشغيل مسلك آخر في نفس المهمة. |
| تستخدم العمليات المتعددة دون استخدام المسالك المزيد من الموارد. | تستخدم المسالك المترابطة موارد أقل. |
| في الأنظمة متعددة العمليات، تعمل كل عملية بشكل مستقل عن الآخرين. | يمكن لأحد المسالك قراءة أو كتابة أو تغيير بيانات مسلك آخر. |

## مزايا المسالك

- المسالك تقلل من وقت تبديل السياق.
- استخدام المسالك يوفر التزامن في العملية.
- التواصل الفعال.
- من الأكثر اقتصادا إنشاء سلاسل الرسائل وتبديل سياقها.
- تسمح المسالك باستخدام البنى متعددة المعالجات على نطاق وكفاءة أكبر.

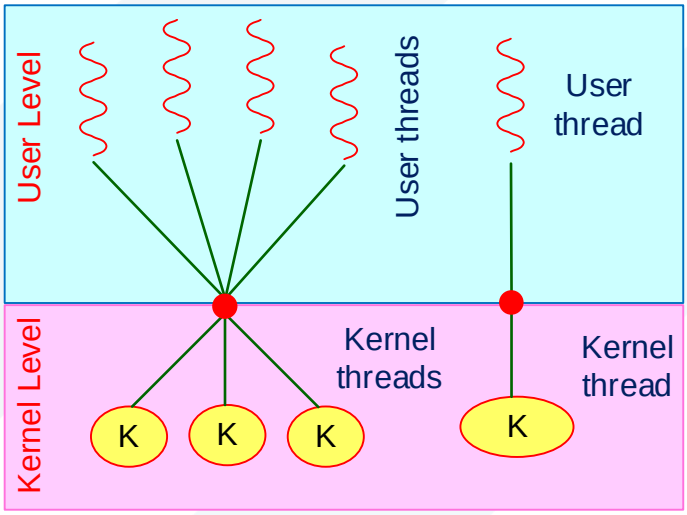
## أنواع المسالك

يتم تنفيذ المسالك باتباع طريقتين:

- المسالك على مستوى المستخدم: المسالك التي يديرها المستخدم.
- مسالك مستوى النواة: مسالك إدارة نظام التشغيل تعمل على kernel، أحد نواة نظام التشغيل.

## المسالك على مستوى المستخدم

تحتوي مكتبة المسالك على كود لإنشاء وإنهاء المسالك، ولتمرير الرسائل والبيانات بين المسالك، ولجدولة تنفيذ المسالك ولحفظ واستعادة سياقات المسالك. يبدأ التطبيق بمسلك واحد.




## مزايا

- لا يتطلب المسلك امتيازات وضع Kernel.
- يمكن تشغيل المسلك على مستوى المستخدم على أي نظام تشغيل.
- يمكن أن تكون الجدولة تطبيقًا محددًا في مسلك مستوى المستخدم.
- المسالك على مستوى المستخدم سريعة في الإنشاء والإدارة.

## سلبيات

- في نظام تشغيل نموذجي، يتم حظر معظم استدعاءات النظام.
- لا يمكن للتطبيق متعدد المسالك الاستفادة من المعالجة المتعددة.

## المسالك على مستوى النواة

في هذه الحالة، تتم إدارة المسلك بواسطة Kernel. لا يوجد رمز إدارة المسلك في منطقة المسلك. يتم دعم مسالك Kernel مباشرة بواسطة نظام التشغيل. يمكن برمجة أي تطبيق ليكون متعدد المسالك. يتم دعم جميع سلاسل الرسائل داخل التطبيق في عملية واحدة.

يحتفظ Kernel بمعلومات السياق للعملية ككل وللمسالك الفردية داخل العملية. يتم إجراء الجدولة بواسطة Kernel على أساس المسلك. ينفذ Kernel إنشاء سلاسل الرسائل وجدولتها وإدارتها في مساحة Kernel. تكون سلاسل Kernel بشكل عام أبطأ في الإنشاء والإدارة من سلاسل رسائل المستخدم.

### مزايا

- يمكن لـ Kernel جدولة مسالك متعددة في نفس الوقت من نفس العملية على عمليات متعددة.
- إذا تم حظر مسلك واحد في عملية ما، يمكن لـ Kernel جدولة مسلك آخر لنفس العملية.
- يمكن أن تكون إجراءات Kernel نفسها مسالك متعددة.

### سلبيات

- سلاسل Kernel بشكل عام أبطأ في الإنشاء والإدارة من سلاسل رسائل المستخدم.
- يتطلب نقل التحكم من مسلكإلى آخر خلال نفس العملية تبديل الوضع إلى Kernel.

### تعدد المسالك

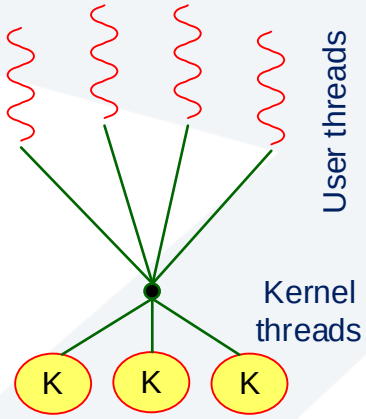
توفر بعض أنظمة التشغيل مسلكا مشتركا على مستوى المستخدم وإمكانية المسلك على مستوى Kernel. يمكن تشغيل العديد من المسالك داخل نفس التطبيق بالتوازي على معالجات متعددة ولا يلزم أن يؤدي استدعاء نظام الحظر إلى حظر العملية بأكملها. نماذج تعدد المسالك ثلاثة أنواع

- نموذج متعدد لمتعدد.
- نموذج متعدد لواحد.
- نموذج واحد لواحد.

### نموذج متعدد إلى لمتعدد

يقوم نموذج متعدد إلى متعدد بتوزيع أي عدد من مسالك المستخدم على عدد مساوٍ أو أصغر من مسالك الـ kernel.

يُظهر الرسم التخطيطي التالي نموذج ترابط متعدد إلى متعدد حيث يتم توزيع 4 مسالك على مستوى المستخدم باستخدام 3 مسالك على مستوى kernel. في هذا النموذج، يمكن للمطورين إنشاء أكبر عدد ممكن من مساالك المستخدم حسب الضرورة ويمكن أن تعمل مسالك Kernel المقابلة بالتوازي على جهاز متعدد المعالجات. يوفر هذا النموذج أفضل دقة في التزامن وعندما ينفذ المسلك حظر استدعاء نظام ، يمكن للنواة جدولة مسلك آخر للتنفيذ.

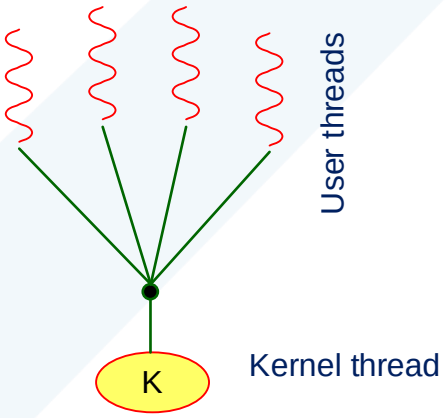



### نموذج كثير لواحد

يخصص نموذج متعدد إلى واحد العديد من المسالك على مستوى المستخدم لمسلك واحد على مستوى Kernel.

تتم إدارة المسلك في مساحة المستخدم بواسطة مكتبة المسالك. عندما يقوم المسلك بإجراء حظر استدعاء نظام، سيتم حظر العملية بأكملها. يمكن لمسلك واحد فقط الوصول إلى Kernel في كل مرة، لذلك لا يمكن تشغيل العديد من مسالك العمليات بالتوازي على المعالجات المتعددة.

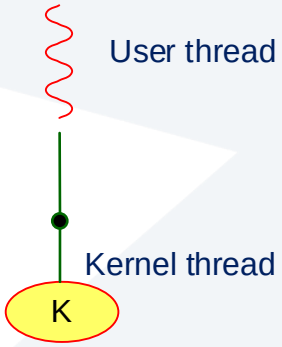
إذا تم تنفيذ مكتبات المسالك على مستوى المستخدم في نظام التشغيل بطريقة لا يدعمها النظام، عندئذٍ تستخدم مسالك Kernel طريقة واحد لواحد.




### نموذج واحد لواحد

يربط نموذج واحد لواحد على المسلك الواحد على مستوى المستخدم بمسلك واحد على مستوى kernel. يوفر هذا النموذج تزامنًا أفضل من نموذج متعدد إلى واحد. كما يسمح أيضًا بتشغيل مسلك آخر عندما يقوم المسلك بإجراء حظر استدعاء النظام. وهو يدعم مسالك متعددة ليتم تنفيذها بالتوازي على المعالجات الدقيقة.

عيب هذا النموذج هو أن إنشاء مسلك مستخدم يتطلب مسلك Kernel المقابل. يستخدم OS / 2 و windows NT و 2000 windows نموذج علاقة واحد لواحد.




### الفرق بين ال**مسلك** على مستوى المستخدم وعلى مستوى النواة

| المسالك على مستوى المستخدم | المسالك على مستوى النواة |
|---|---|
| أسرع في الإنشاء والإدارة. | أبطأ في الإنشاء والإدارة. |
| يتم التنفيذ عن طريق مكتبة المسالك على مستوى المستخدم. | يدعم نظام التشغيل إنشاء مسالك Kernel. |
| المسلك على مستوى المستخدم عام ويمكن تشغيله على أي نظام تشغيل. | المسلك على مستوى Kernel خاص بنظام التشغيل. |
| لا يمكن للتطبيقات متعددة المسالك الاستفادة من المعالجة المتعددة. | يمكن أن تكون إجراءات Kernel نفسها ذات مسالك متعددة. |

## أمثلة على أنظمة التشغيل (هذه الفقرة للاطلاع)

### مسالك Windows

في Windows، قد تحتوي كل عملية على مسلك واحد أو أكثر. يستخدم النموذج واحد لواحد

تشمل المكونات العامة للمسلك ما يلي:

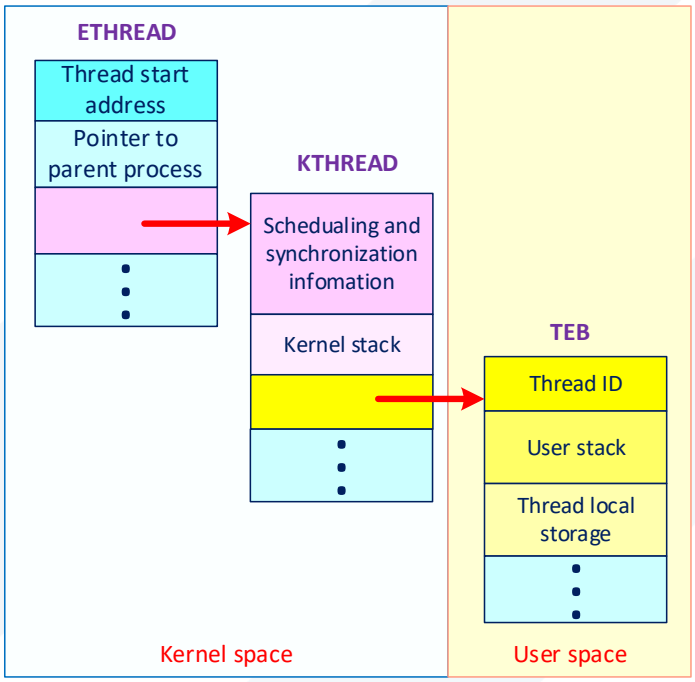
- معرف المسلك
- مجموعة سجل تمثل حالة المعالج
- عداد برنامج
- مكدس مستخدم، يتم استخدامه عند تشغيل المسلك في وضع المستخدم، ومكدس النواة، يتم استخدامه عند تشغيل المسلك في وضع kernel
- منطقة تخزين خاصة تستخدمها العديد من مكتبات وقت التشغيل ومكتبات الارتباط الديناميكي (DLL)
- تُعرف مجموعة التسجيل والمكدسات ومنطقة التخزين الخاصة بسياق المسلك.

تشمل بنى المعطيات الأساسية للمسلك ما يلي:

- ETHREAD – كتلة مسلك تنفيذية
- KTHREAD – كتلة مسلك النواة

- TEB – كتلة بيئة المسلك

تتضمن المكونات الرئيسية لـ ETHREAD مؤشرًا للعملية التي ينتمي إليها المسلك وعنوان الروتين الذي يبدأ فيه المسلك في التحكم. يحتوي ETHREAD أيضًا على مؤشر إلى KTHREAD المقابل.




يتضمن KTHREAD معلومات الجدولة والمزامنة للمسلك. بالإضافة إلى ذلك، يتضمن KTHREAD مكدس kernel (يستخدم عند تشغيل المسلك في وضع kernel) ومؤشر إلى TEB.

إن ETHREAD و KTHREAD موجودان بالكامل في مساحة النواة ؛ هذا يعني أن النواة فقط هي التي يمكنها الوصول إليها. TEB عبارة عن بنية بيانات لمساحة المستخدم يتم الوصول إليها عند تشغيل المسلك في وضع المستخدم. من بين الحقول الأخرى، يحتوي TEB على معرف المسلك، ومكدس وضع المستخدم، وصفيف للتخزين المحلي للمسلك.

### مسالك لينكس

يوفر Linux استدعاء نظام fork() مع الوظيفة التقليدية لإنشاء نسخة العملية. كما يوفر القدرة على إنشاء المسالك باستخدام استدعاء النظام clone() ومع ذلك، لا يميز Linux بين العمليات والمسالك. يستخدم مصطلح المهمة، بدلاً من العملية أو المسلك، عند الإشارة إلى تدفق التحكم داخل البرنامج.

عندما يتم استدعاء clone() ، يتم تمرير مجموعة من العلامات التي تحدد مقدار المشاركة التي يجب أن تحدث بين المهام الرئيسية والمهام الفرعية. يتم سرد بعض هذه العلامات في الجدول التالي. على سبيل المثال، افترض أن clone()قد تم تمرير العلامات CLONE FS و CLONE VM و CLONE SIGHAND و CLONE FILES. ستقوم المهام الرئيسية والفرعية بعد ذلك بمشاركة نفس معلومات نظام الملفات (مثل دليل العمل الحالي)، ونفس مساحة الذاكرة، ونفس معالجات الإشارة، ونفس مجموعة الملفات المفتوحة. استخدام clone() بهذه الطريقة يعادل إنشاء سلسلة رسائل كما هو موضح في هذا الفصل، نظرًا لأن المهمة الرئيسية تشترك في معظم مواردها مع المهمة الفرعية الخاصة

بها. ومع ذلك، إذا لم يتم تعيين أي من هذه العلامات عند استدعاء clone()، فلن تحدث مشاركة، مما ينتج عنه وظيفة مشابهة لتلك التي يوفرها استدعاء النظام fork().

## *Linux Threads*

| Flag | meaning |
|---|---|
| CLONE_FS | يتم مشاركة معلومات نظام الملفات. |
| CLONE_VM | نفس مساحة الذاكرة مشتركة. |
| CLONE_SIGHAND | معالجات الإشارة مشتركة. |
| CLONE_FILES | مجموعة الملفات المفتوحة مشتركة. |

المستوى المتغير للمشاركة ممكن بسبب الطريقة التي يتم بها تمثيل المهمة في Linux kernel. يوجد هيكل بيانات kernel فريد (على وجه التحديد، هيكل Task_struct) لكل مهمة في النظام. تحتوي بنية البيانات هذه، بدلاً من تخزين البيانات للمهمة، على مؤشرات لهياكل البيانات الأخرى حيث يتم تخزين هذه البيانات – على سبيل المثال، هياكل البيانات التي تمثل قائمة الملفات المفتوحة ومعلومات معالجة الإشارات والذاكرة الافتراضية. عندما يتم استدعاء fork() يتم إنشاء مهمة جديدة، مع نسخة من جميع هياكل البيانات المرتبطة بالعملية الرئيسية. يتم أيضًا إنشاء مهمة جديدة عند إجراء استدعاء نظام clone(). ومع ذلك، بدلاً من نسخ جميع هياكل البيانات، تشير المهمة الجديدة إلى هياكل البيانات الخاصة بالمهمة الأصلية، اعتمادًا على مجموعة العلامات التي تم تمريرها لـ clone().