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. ,وبالتالي فإن 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 اجرائها داخل النواة. لذلك ينفذ النظام مهامه كمجموعة 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 نشرح 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 النوى 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 الحديثة مسالك التنفيذ على مستوى النواة.

أهداف الوحدة

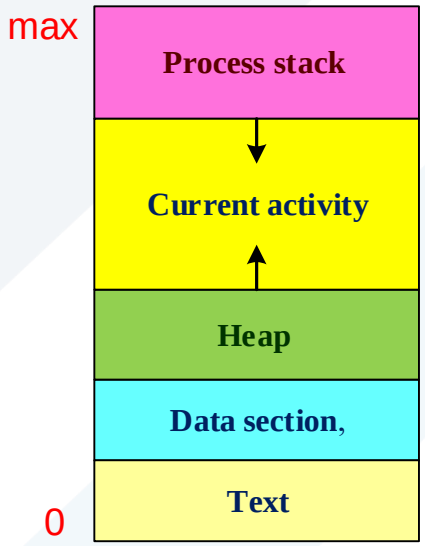
- تحديد المكونات المنفصلة للعملية وتوضيح كيفية تمثيلها وجدولتها في نظام التشغيل.
- وصف كيفية إنشاء العمليات وإنهاؤها في نظام التشغيل، بما في ذلك تطوير البرامج باستخدام استدعاءات النظام المناسبة التي تؤدي هذه العمليات.
- وصف ومقارنة الاتصال بين العمليات باستخدام الذاكرة المشتركة وتمرير الرسائل.
- التعرف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

مفهوم العملية Process Concept

العملية هي في الأساس البرنامج قيد التنفيذ. حيث يجب أن يتقدم تنفيذ العملية بطريقة متسلسلة. وبالتالي تكون العملية كيان يمثل وحدة العمل الأساسية التي سيتم تنفيذها في النظام.

تكتب برامج الكمبيوتر الخاصة بنا في ملف نصي بوضعها بعبارات مفهومة للمستخدمة وسهلة في الكتابة، 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، و heap، ونص وبيانات. يظهر الشكل 1 تخطيطًا مبسطًا لعملية داخل الذاكرة الرئيسية:



الشكل 1: العملية داخل الذاكرة

ويمكن وصف هذه المكونات كالتالي:

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

كومة Heap: يتم تخصيص هذه الذاكرة ديناميكيًا لعملية أثناء تشغيلها.

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

بيانات Data: 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 ننظر إلى العملية كمثيل ديناميكي لبرنامج كمبيوتر.

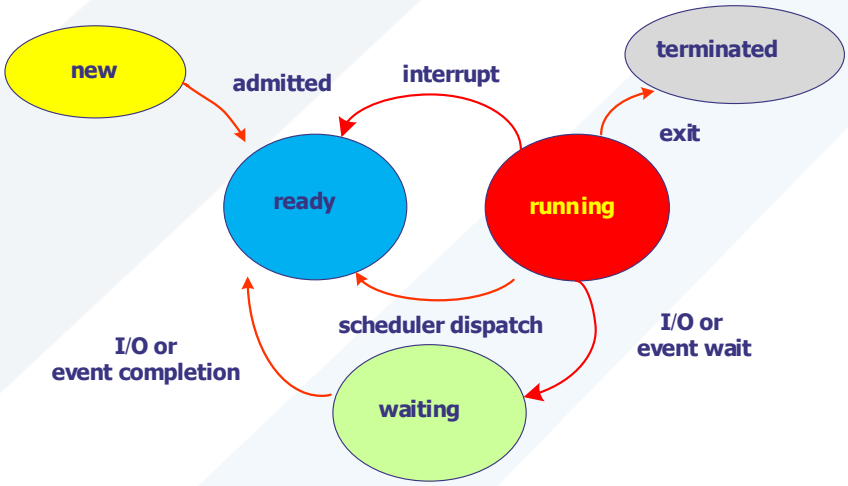
حالات العملية Process State

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 كما في الشكل 2.

- **عملية جديدة New:** هذه هي الحالة الأولية عند بدء / إنشاء العملية لأول مرة.

- **الجاهزية Ready:** 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 Running:** بمجرد تخصيص العملية بوقت المعالج بواسطة برنامج جدولة المعالج، يتم تعيين حالة العملية على التنفيذ ويقوم المعالج بتنفيذ تعليماته.
- **الانتظار Waiting:** تنتقل العملية إلى حالة الانتظار إذا احتاجت إلى انتظار مورد ما، مثل انتظار إدخال قيم من المستخدم أو انتظار توفر الملف.
- **الإنهاء terminated:** بمجرد أن تنتهي العملية من تنفيذها، أو يتم إنهاؤها بواسطة نظام التشغيل، يتم نقلها إلى الحالة المنتهية حيث تنتظر إزالتها من الذاكرة الرئيسية.



الشكل 2: حالات العملية

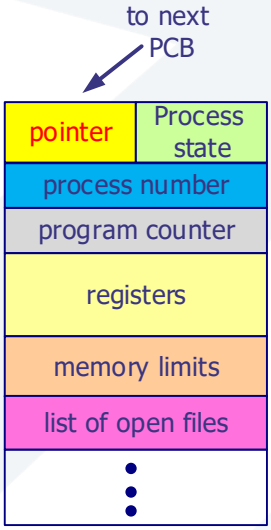
كتلة التحكم في العمليات (PCB)

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. يبين الشكل 3 توضيحا مبسطا 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.

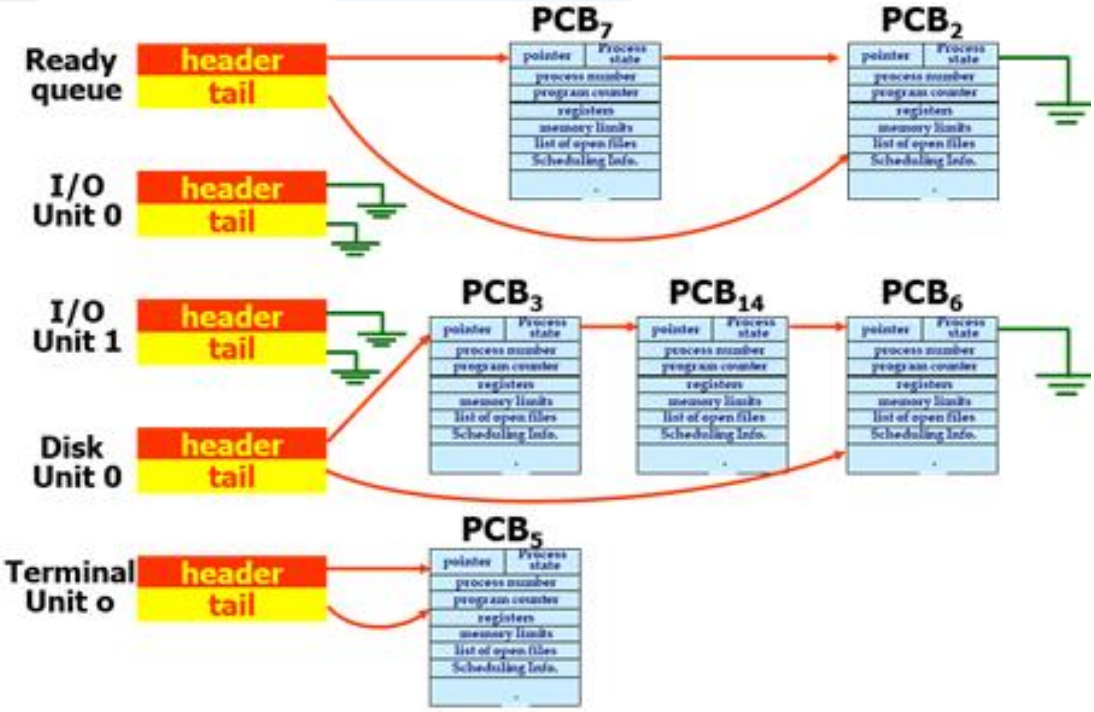


الشكل 3: مكونات الـ PCB

- **قائمة الملفات المفتوحة:** تتضمن هذه المعلومات قائمة الملفات المفتوحة لعملية ما.

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

جدولة العمليات



الشكل 4: جدولة العمليات

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

تعد جدولة العمليات جزءًا أساسيًا من أنظمة التشغيل التي تسمح بتحميل أكثر من عملية واحدة في الذاكرة لتنفيذها في وقت واحد، كما في الشكل 4. وتتشارك العمليات المحملة وحدة المعالجة المركزية باستخدام التجميع الزمني.

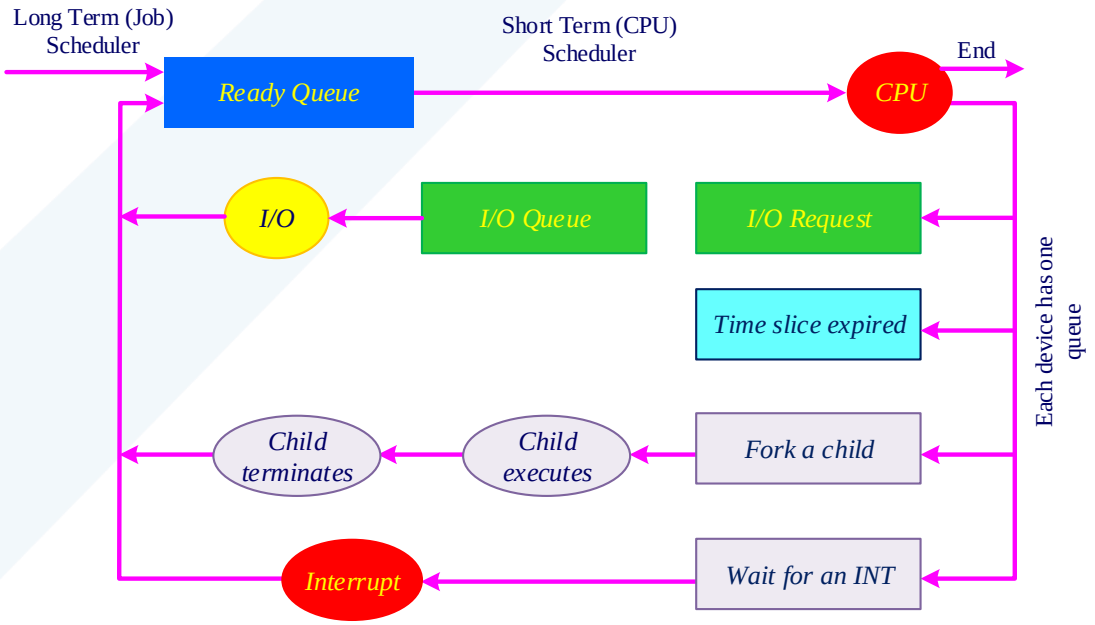
أنواع الجدولة

هناك نوعان من الجدولة:

- **جدولة غير استيلائية Non-Preemptive:** حيث لا يمكن أخذ المورد من العملية حتى يكمل المعالج تنفيذ العملية. ويحدث تبديل الموارد عندما تنتهي العملية الجارية أو تنتقل إلى حالة الانتظار.
- **جدولة استيلائية Preemptive:** 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 الجارية بعملية ذات أولوية أعلى.

طوابير انتظار العملية Process Waiting Queues

يحتفظ نظام التشغيل بجميع كتل التحكم في العمليات (PCBs) في قوائم انتظار جدولة العمليات. حيث يحتفظ بطابور انتظار منفصل لكل حالة من حالات العملية ويكون PCB لجميع العمليات في نفس حالة التنفيذ في نفس طابور الانتظار. عندما يتم تغيير حالة العملية، يتم إلغاء ربط PCB الخاص بها من طابور الانتظار الحالي وينتقل إلى طابور انتظار الحالة الجديدة ويبين الشكل 5 بيانا لطوابير الانتظار.



الشكل 5: طوابير انتظار العملية

يحافظ نظام التشغيل على طوابير انتظار جدولة العمليات المهمة التالية:

طابور العمل Job Queue: طابور الانتظار هذه تحافظ على جميع العمليات في النظام.

طابور انتظار العمليات الجاهزة Ready Queue: تحتفظ قائمة الانتظار هذه بمجموعة من جميع العمليات الموجودة في الذاكرة الرئيسية، جاهزة وتنتظر التنفيذ. يتم وضع عملية جديدة دائمًا في قائمة الانتظار هذه.

طابور الأجهزة Device Queue: تتشكل من العمليات التي تم حظرها بسبب عدم توفر جهاز الإدخال / الإخراج.

يمكن لنظام التشغيل استخدام خوارزميات مختلفة لإدارة كل قائمة انتظار (FIFO، Round Robin، الأولوية، وما إلى ذلك) وسنعود إلى هذه الخوارزميات لاحقا بشيء من التفصيل. يحدد برنامج جدولة نظام التشغيل كيفية نقل العمليات بين طوابير وفقا لبنية المنظومة الحاسوبية والخوارزميات التي تعمل وفقها.

نموذج العملية ثنائي الحالة Process Dual Mode Operation

يشير نموذج العملية ثنائي الحالة إلى الحالات قيد التشغيل وخارج 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 طوابير الانتظار، في انتظار دورها للتنفيذ. يتم تطبيق جدول الانتظار باستخدام القائمة المترابطة. عند مقاطعة عملية ما، يتم نقل هذه العملية الى طابور الانتظار. وبشكل عام إذا اكتملت العملية أو تمت مقاطعتها. في كلتا الحالتين، يقوم الموزع الخاص بالجدولة بعد ذلك بتحديد عملية أخرى من طابور 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 Long Term Scheduler

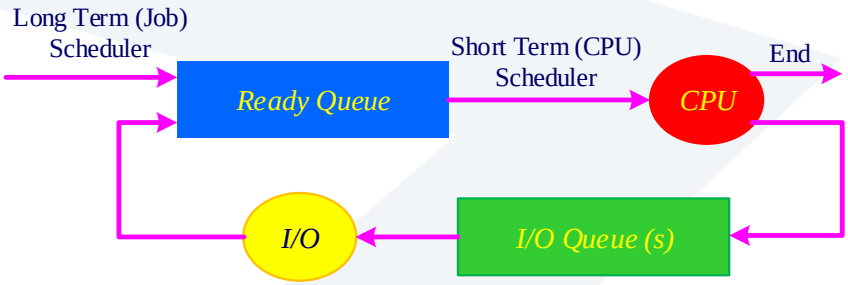
جدولة قصيرة المدى Short Term Scheduler

جدولة متوسطة المدى Med Term Scheduler

جدولة طويلة المدى

وتسمى أيضًا بجدولة الوظائف Job Scheduler. يحدد المجدول طويل المدى، الشكل 6 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.



الشكل 6: المجدول طويل الأمد

في بعض الأنظمة، قد لا يكون المجدول طويل المدى متاحًا أو متاحا بشكل جزئي. حيث 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.

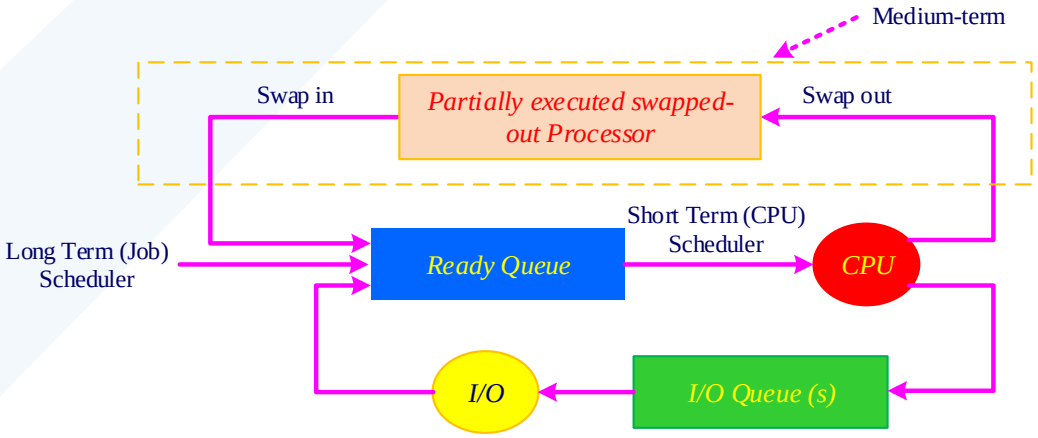
جدولة قصيرة المدى

ويسمى أيضًا باسم جدولة وحدة المعالجة المركزية. هدفها الرئيسي هو زيادة أداء النظام وفقًا لمجموعة المعايير المختارة. إنه تغيير حالة الجاهزية 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، الشكل 6، 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، الشكل 7 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 والمبادلة.



الشكل 7: مجدول متوسط الأمد

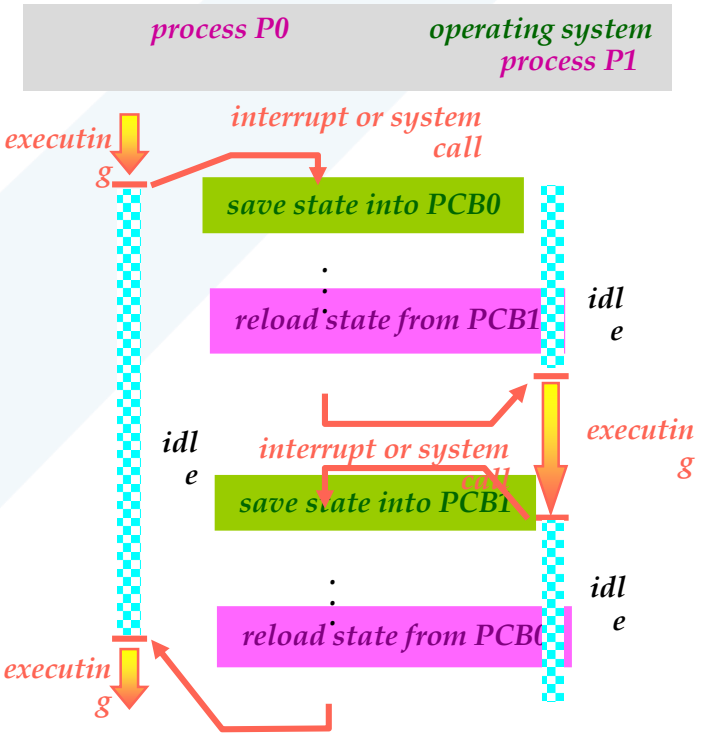
قد يتم تعليق العملية الجارية إذا قدمت طلب إدخال / إخراج. لا يمكن للعمليات المعلقة إحراز أي تقدم في انجازها نحو الانتهاء. في هذه الحالة، لإزالة العملية من الذاكرة وإفساح المجال للعمليات الأخرى، يتم نقل العملية المعلقة إلى الحفظ. تسمى هذه العملية بالمبادلة، ويقال أن العملية يتم تبديلها.

Comparison among Scheduler

يبين الجدول أدناه مقارنة بين المجدولات المذكورة أعلاه

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



الشكل 8: مثال لتبديل السياق عمليتين

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. ويعد 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. ويبين الشكل 8، توضيحا للمخطط الزمني لتبديل السياق بين عمليتين.

مراحل تشغيل العمليات

1. **الانشاء Creation:** 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 Scheduling/Dispatching:** الحدث أو النشاط الذي يتم فيه تغيير حالة العملية إلى 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.
3. **المنع Blocking:** عندما تستدعي عملية ما استدعاء نظام إدخال وإخراج تحظر العملية، ويوضع نظام التشغيل في وضع الحظر. وضع الحظر هو ببساطة وضع تنتظر فيه العملية الإدخال والإخراج. وبناءً على طلب العملية نفسها، يحظر نظام التشغيل العملية ويرسل عملية أخرى إلى المعالج. وبالتالي، في عمليات حظر العملية، يضع نظام التشغيل العملية في حالة "انتظار".
4. **الاستيلاء "الاستباق" Preemption:** عند انتهاء المهلة، يعني ذلك أن العملية لم تُنهَ ضمن الفترة الزمنية المخصصة وأن العملية التالية جاهزة للتنفيذ، فيستبقها نظام التشغيل. لا تسري هذه العملية إلا عندما تدعم جدولة وحدة المعالجة المركزية الاستباق. يحدث هذا في جدولة الأولوية، حيث تُستبق العملية الجارية عند وصول عملية ذات أولوية عالية. وبالتالي، في عملية استباق العملية، يضع نظام التشغيل العملية في حالة الجاهزية.

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

- 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 *Inter-process communication*

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.

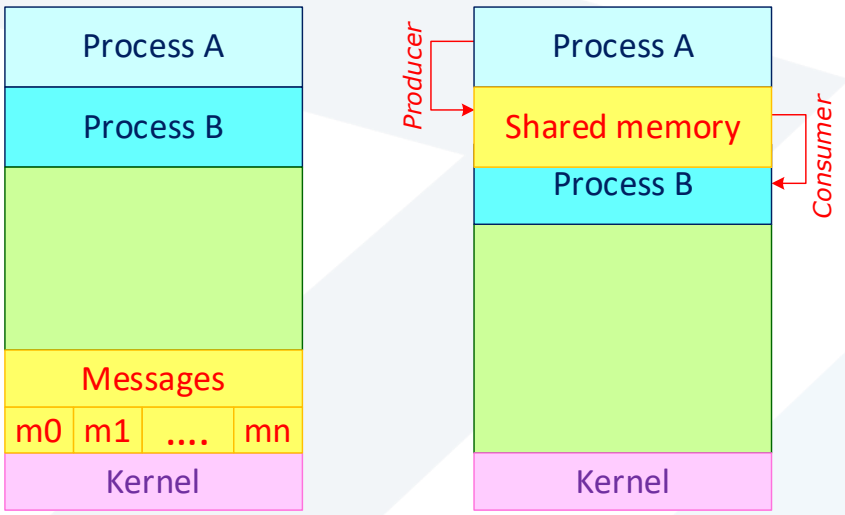
هناك عدة أسباب لتوفير بيئة تسمح بالتعاون في العمليات:

- **مشاركة المعلومات Information sharing**: 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 Computation speedup**: 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.
- **النمطية modularity**: قد نرغب في بناء النظام بطريقة معيارية، وتقسيم وظائف النظام إلى عمليات أو سلاسل عمليات منفصلة.
- **الملائمة Convenience**: حتى المستخدم الفردي قد يعمل على عدة مهام في الوقت نفسه. على سبيل المثال، قد يقوم المستخدم بالتحرير والطباعة والتجميع بالتوازي وهذا يتطلب تواصلا سلساً بين العمليات.

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 independent إذا لم تؤثر أو تتأثر بالعمليات الأخرى ولا تشارك البيانات مع العمليات الأخرى. وتكون العملية تعاونية Cooperative إذا كان من الممكن أن تتأثر أو تؤثر على العمليات الأخرى و يشارك البيانات مع العمليات الأخرى.



الشكل 9: الاتصال بين العمليات

تحتاج العمليات المتعاونة إلى آلية اتصال بينها (IPC). هناك نوعان من نماذج IPC كما في الشكل 9.

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

نظام الذاكرة المشتركة shared memory

يوضح الشكل 9 بنية أساسية للتواصل بين العمليات عبر أسلوب الذاكرة المشتركة وأسلوب تمرير الرسائل. يتطلب التواصل بين العمليات باستخدام الذاكرة المشتركة مشاركة بعض المتغيرات، ويعتمد ذلك كلياً على كيفية تطبيق المبرمج لذلك. يمكن تصور إحدى طرق التواصل باستخدام الذاكرة المشتركة على النحو التالي: لنفترض أن العملية 1 والعملية 2 تعملان في وقت واحد، وتتشاركان بعض الموارد أو تستخدمان بعض المعلومات من عملية أخرى. تُولّد العملية 1 معلومات حول عمليات حسابية أو موارد معينة قيد الاستخدام، وتحتفظ بها كسجل في الذاكرة المشتركة. عندما تحتاج العملية 2 إلى استخدام المعلومات المشتركة، فإنها ستتحقق من السجل المخزن في الذاكرة المشتركة، وتأخذ المعلومات التي تولدها العملية 1 في الاعتبار، وتتصرف بناءً عليها. يمكن للعمليات استخدام الذاكرة المشتركة لاستخراج المعلومات كسجل من عملية أخرى، وكذلك لتوصيل أي معلومات محددة إلى عمليات أخرى.

تبادل الرسالة Message Passing:

يُعد تبادل الرسائل بين العمليات طريقةً تتواصل من خلالها العمليات عبر إرسال واستقبال الرسائل لتبادل البيانات. في هذه الطريقة، ترسل إحدى العمليات رسالة، وتستقبلها العملية الأخرى، مما يسمح لهما بمشاركة المعلومات. يمكن تحقيق تمرير الرسائل من خلال طرق مختلفة مثل المقابس Sockets، أو باستخدام طوابير الرسائل Message Queuing، أو الأنابيب Pipes.

توفر المقابس نقطة نهاية للاتصال، مما يسمح للعمليات بإرسال واستقبال الرسائل عبر الشبكة. في هذه الطريقة، تفتح إحدى العمليات (الخادم) Server مقبسًا وتستمع إلى الاتصالات الواردة، بينما تتصل العملية الأخرى (العميل) Client

بالخادم وترسل البيانات. يمكن للمقابس استخدام بروتوكولات اتصال مختلفة، مثل TCP بروتوكول التحكم في الإرسال لاتصالات موثوقة ذات اتصال مسبق وبروتوكول UDP كبروتوكول غير مسبق الاتصال.

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 أوليتين 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 Multithreading

ما هو المسلك؟What is the Thread

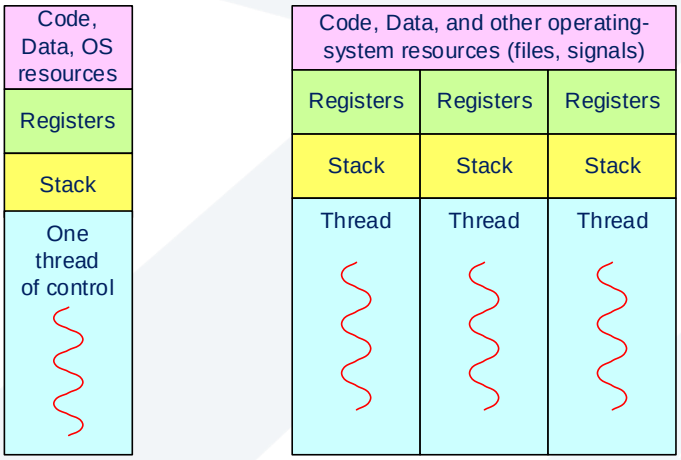
المسلك هو مسار لتنفيذ جزء من كود العملية، مع عداد البرنامج الخاص به بحيث يتبع التعليمات التي يجب تنفيذها بعد ذلك، وله مسجلات النظام التي تحتوي على متغيرات العمل الحالية، والمكدس الذي يحتوي على المحفوظات قيد التنفيذ.

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

يسمى المسلك أيضًا عملية خفيفة الوزن .Lightweight Process توفر المسالك طريقة لتحسين أداء التطبيق من خلال توازي المسارات. ويكون بذلك طريقة برمجية لتحسين أداء نظام التشغيل عن طريق تقليل التكلفة وتحسين معدل المعالجة.

كل مسلك ينتمي إلى عملية واحدة بالضبط، ولا يمكن أن يوجد مسلك خارج العملية. يمثل كل مسلك مسار منفصل للتحكم. يتم استخدام المسالك بنجاح في تنفيذ خوادم الشبكة وخادم الويب. كما أنها توفر أساسًا مناسبًا للمعالجة المتوازية

للتطبيقات على معالجات الذاكرة المشتركة المتعددة. يوضح الشكل التالي مسارات تنفيذ عملية أحادية المسلك ومتعددة المسالك.



الشكل 10: مسارات تنفيذ عملية أحادية المسلك ومتعددة المسالك

الفرق بين العملية والمسلك

يختصر الجدول التالي الفروق 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.

- تؤمن المسالك التواصل الفعال بين أجزاء العملية الواحدة.
- يصبح باستخدام المسالك من الأجدى اقتصاديا إنشاء سلاسل الرسائل وتبديل سياقها.
- تسمح المسالك باستخدام البنى متعددة المعالجات على نطاق وكفاءة أكبر.

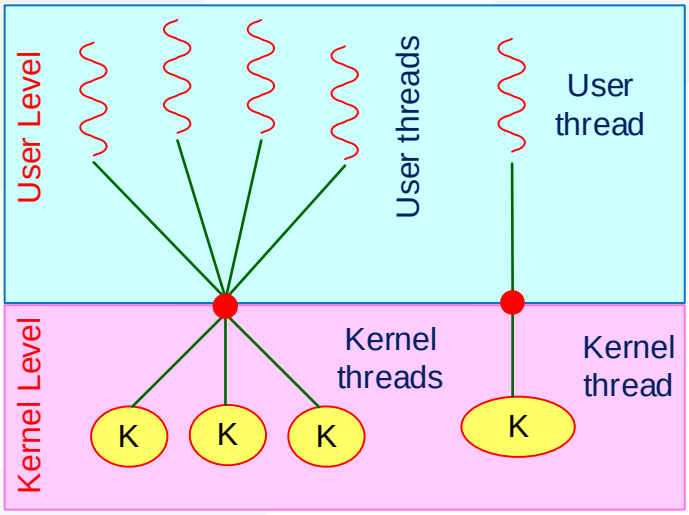
أنواع المسالك

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 User level threads

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



الشكل 11: المسالك على مستوى المستخدم والمسالك على مستوى النواة

ولتطبيق المسالك على مستوى المستخدم ميزات نذكر منها:

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

ويعاني أيضا من بعض السلبيات مثل:

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

المسالك على مستوى النواة Kernel level thread

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

تحتفظ النواة بمعلومات السياق للعملية ككل وللمسالك الفردية داخل العملية. يتم إجراء الجدولة بواسطة النواة على أساس المسلك. تقوم الـ النواة بإنشاء سلاسل الرسائل وجدولتها وإدارتها في مساحة النواة. تكون سلاسل النواة بشكل عام أبطأ في الإنشاء والإدارة من سلاسل رسائل المستخدم.

ولهذا الأسلوب أيضا ميزاته، ومنها:

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

بينما يعاني من سلبيات منها:

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

تعدد المسالك Multithreading

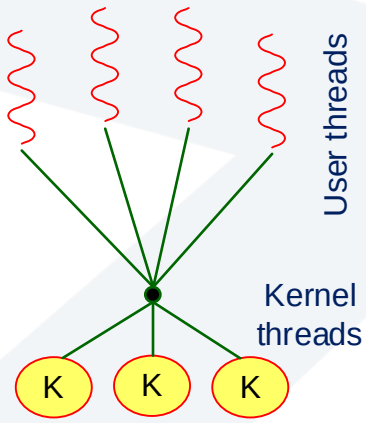
توفر بعض أنظمة التشغيل مسلكا مشتركا على مستوى المستخدم وإمكانية المسلك على مستوى النواة. يمكن تشغيل العديد من المسالك داخل نفس التطبيق بالتوازي على معالجات متعددة ولا يلزم أن يؤدي استدعاء نظام الحظر إلى حظر العملية بأكملها. نماذج تعدد المسالك ثلاثة أنواع

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

نموذج متعدد إلى لمتعدد Many to Many

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

يُظهر الشكل 12 نموذج مسالك متعدد إلى متعدد حيث يتم توزيع 4 مسالك على مستوى المستخدم باستخدام 3 مسالك على مستوى النواة. في هذا النموذج، يمكن للمطورين إنشاء أكبر عدد ممكن من مسالك المستخدم حسب الضرورة ويمكن أن تعمل مسالك النواة 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.

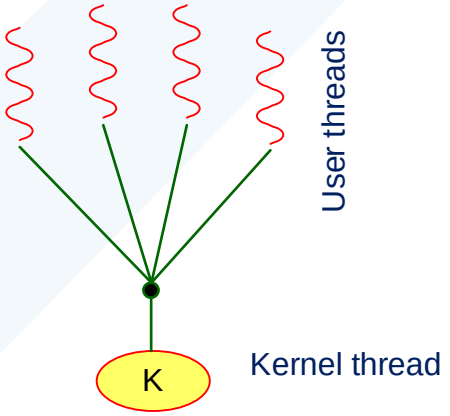


الشكل 12: نموذج مسالك متعدد إلى متعدد

نموذج كثير لواحد Many to one

يخصص نموذج متعدد إلى واحد، الشكل 13 العديد من المسالك على مستوى المستخدم لمسلك واحد على مستوى النواة. 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 النواة 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 النواة طريقة واحد لواحد.

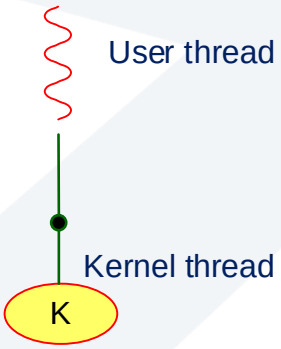


الشكل 13: نموذج مسالك متعدد إلى واحد

نموذج واحد لواحد One to one

يربط نموذج واحد لواحد، الشكل 14، على المسلك الواحد على مستوى المستخدم بمسلك واحد على مستوى النواة. 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 النواة المقابل. يستخدم OS / 2 و windows NT و windows 2000 نموذج علاقة واحد لواحد.



الشكل 14: نموذج المسالك واح إلى واحد

الفرق بين المسلك على مستوى المستخدم وعلى مستوى النواة

يبين الجدول التالي جدولا للفروقات بين المسالك على مستوى المستخدم والمسالك على مستوى النواة

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.