Lecture Four
المحاضرة الرابعة
(Threads)

المسار التنفيذيThread

- A thread is a flow of execution through the process code.
- **هي مجموعة متسلسلة من التعليمات التي ينقسم إليها برنامج ليقوم بأكثر من مهمة Task بشكل متزامن.**
- Each thread belongs to exactly one process and no thread can exist outside a process. كل مسار تنفيذي تابع لعملية ولا يمكن ان يكون بدون عملية
- A thread is sometimes called **a light weight process. يطلق على المسا رالتنفيذي عملية خفيفة الوزن**

Processes and Threads

- ***Multithreading مسارات متعددة*** - The ability of an OS to support multiple, concurrent paths of execution within a single process.
- قدرة نظام التشغيل على دعم مسارات تنفيذ متعددة ومتزامنة ضمن عملية واحدة.
- The unit of dispatching is referred to as a ***thread*** or ***lightweight process.*** يشار إلى وحدة الإرسال باسم عملية خيط أو خفيفة الوزن.

- البرنامج الذى يقوم بأداء أكثر من مهمة اى يحتوى على مسارات تنفيذ يسمى Multi Thraeded Proccess.
- يوجد مثال مبسط لفهم كيفية تنفيذ العمليات حسب انواع ال threads
- لو تحدثنا عن احد المستندات في برنامج ال Microsoft word:عند تشغيل هذا البرنامج يعنى انه process ثم لو حاولنا القيام بعملية طباعة المستند print، والتدقيق الاملائي spill check، واجراء عملية الحفظ saving في وقت واحد.
- اولا :اذا كانت هذه العمليات لا يمكن ان تتم في نفس الوقت أي يجب ان ينتهي من الطباعه ثم بامكانه التدقيق وبعد الانتهاء منه يكون بامكانه الحفظ فهذا يعنى ان جميع هذه العمليات الثلاث تتم في thread واحد , وهذا هو معنى ان تكون عمليات تتم في مسار تنفيذ واحد single threaded process
- ثانيا : اذا كانت هذه العمليات تتم في نفس الوقت فهذا يعنى ان البرنامج يحتوى على ثلاث مسارات تنفيذيه ال thread الاولى للطباعه والثانيه للتدقيق الاملائي والثالثه للحفظ , وهذا هو معنى ان تكون عمليات تتم في مسارات تنفيذيه متعدده multithreaded proces.

Single Threaded Approaches نظرية المسار الواحد

- A single execution path per process, in which the concept of a thread is not recognized, is referred to as a single-threaded approach. يتم تنفيذ مسار واحد فقط ضمن العملية الواحدة
- MS-DOS, some versions of UNIX supported only this type of process. MS-DOS, بعض إصدارات UNIX تدعم فقط هذا النوع من العمليات

Single Threaded Approaches

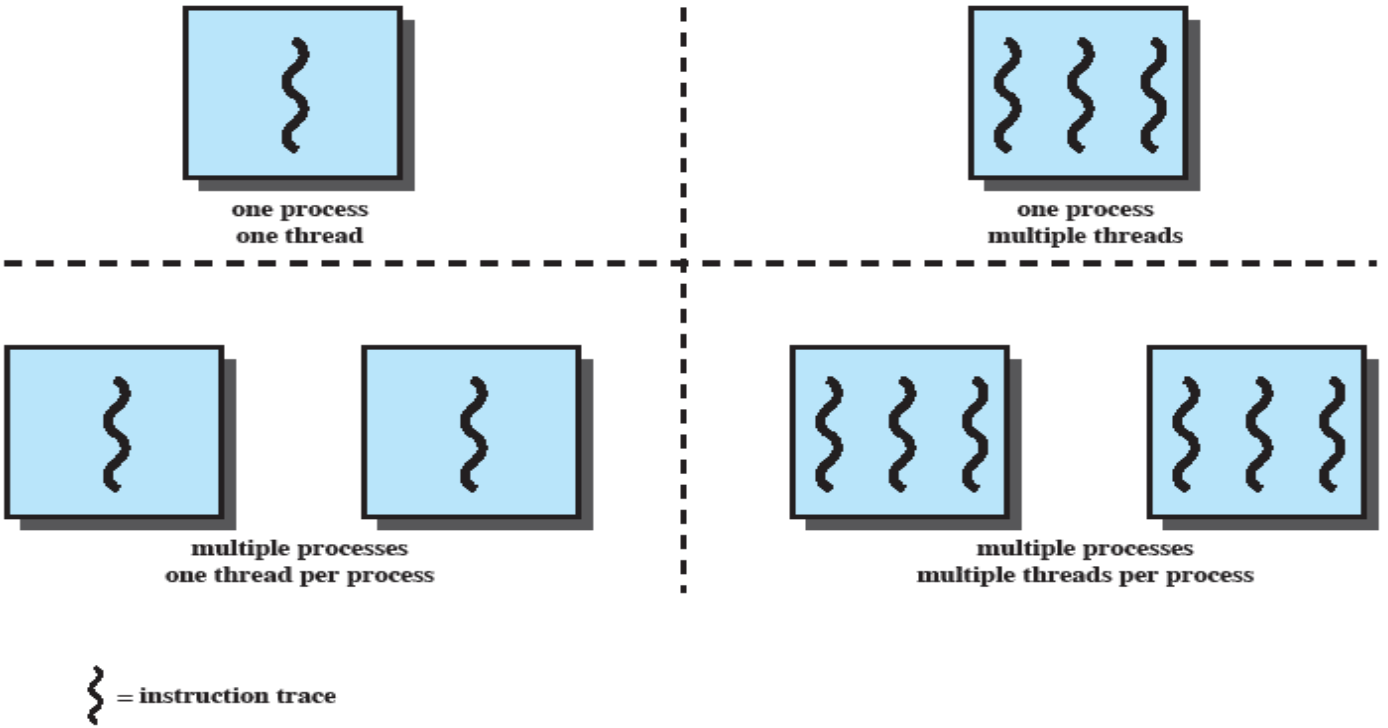


Figure 4.1 Threads and Processes [ANDE97]

Multithreaded نظرية المسارات المتعددة Approaches

- A Java run-time environment is a system of *one* process with multiple threads; Windows, some UNIXes, support *multiple* multithreaded processes.
- هي نظام لعملية واحدة ذات خيوط Javaبيئة وقت تشغيل عمليات UNIXesوبعض Windowsمتعددة. يدعم متعددة مؤشرات الترابط

نظرية المسارات المتعددة Multithreaded Approaches

The right half of Figure 4.1 depicts multithreaded approaches. A Java run-time environment is an example of a system of one process with multiple threads. Of interest in this section is the use of multiple processes, each of which supports multiple threads. This approach is taken in Windows, Solaris, and many modern versions of UNIX, among others. In this section we give a general description of multithreading; the details of the Windows, Solaris, and Linux approaches are discussed later in this chapter.

يصور النصف الأيمن من الشكل 4.1 مناهج متعددة مؤشرات الترابط. تعد بيئة وقت تشغيل Javaمثالا على نظام عملية واحدة ذات مؤشرات ترابط متعددة. من المهم في هذا القسم استخدام عمليات متعددة ، كل منها يدعم خيوط متعددة. يتم اتباع هذا النهج في Windowsو Solarisوالعديد من الإصدارات الحديثة من UNIXوغيرها. في هذا القسم نقدم وصفا عاما لتعدد مؤشرات الترابط ؛ تتم مناقشة تفاصيل مناهج Windowsو

Multithreaded Approaches

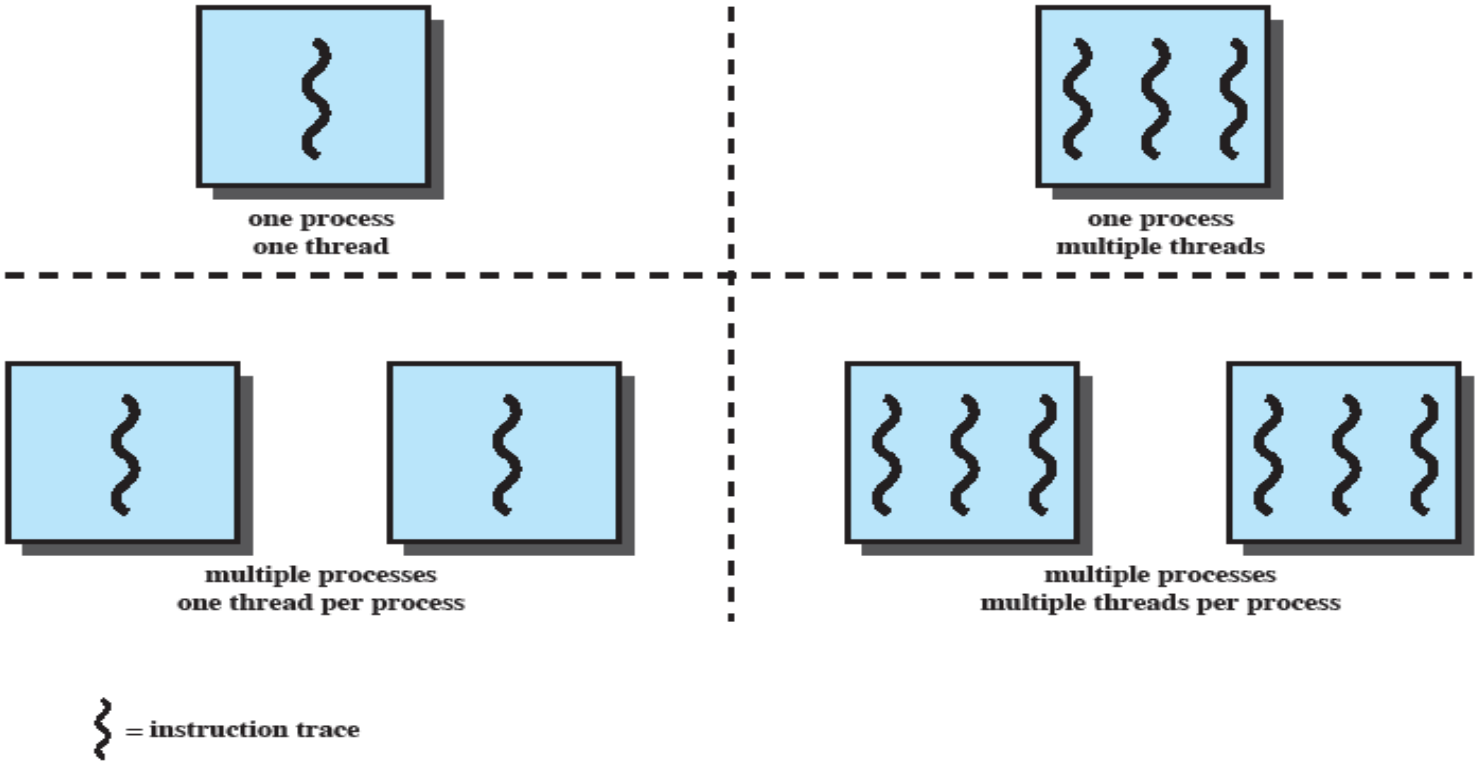


Figure 4.1 Threads and Processes [ANDE97]

Threads vs. Processes

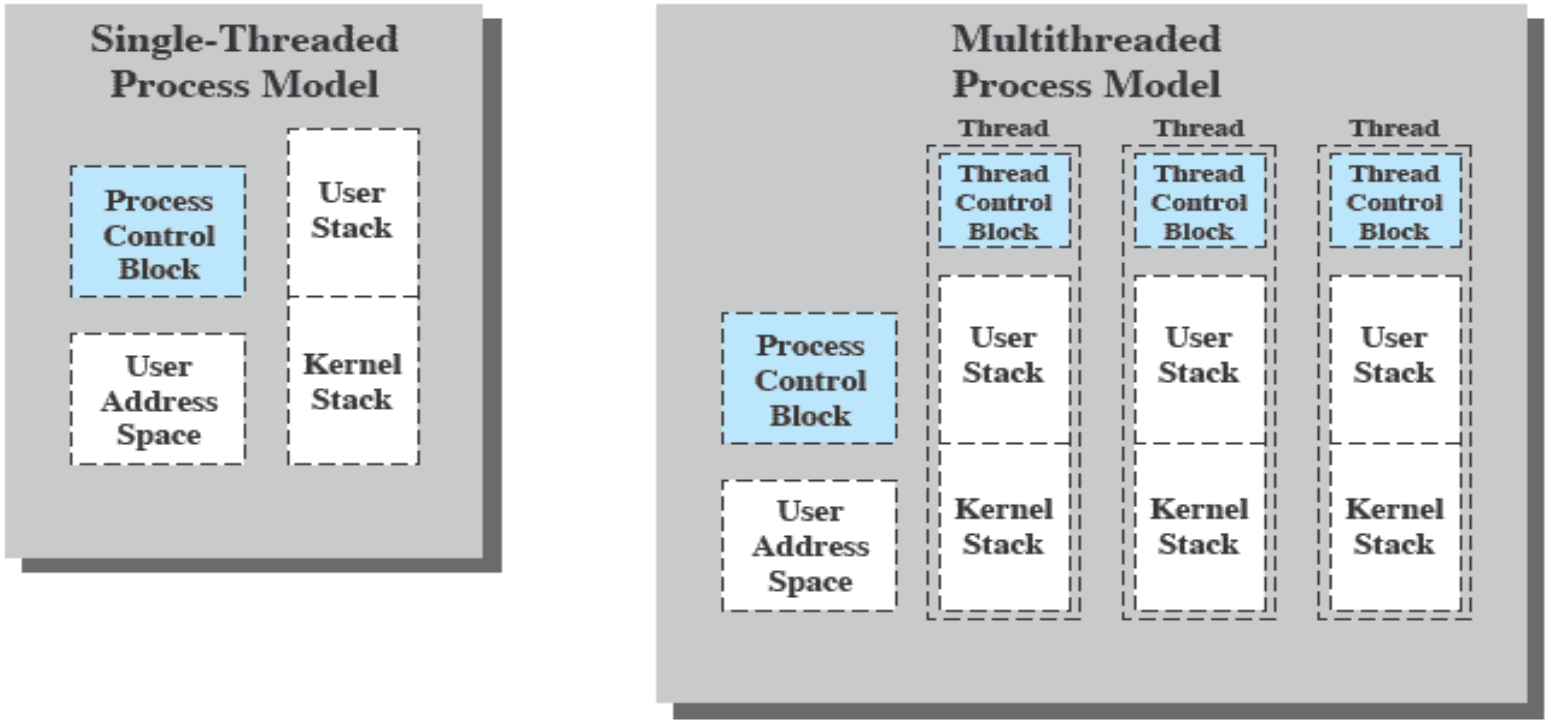


Figure 4.2 Single Threaded and Multithreaded Process Models

Threads vs. Processes

يوضح الشكل 4.2 التمييز بين الخيوط والعمليات من وجهة نظر إدارة العمليات. في نموذج عملية أحادي الخيط (أي لا يوجد مفهوم مميز لمؤشر الترابط) ، يتضمن تمثيل العملية كتلة التحكم في العملية ومساحة عنوان المستخدم ، بالإضافة إلى مكدسات المستخدم والنواة لإدارة سلوك الاستدعاء / الإرجاع لتنفيذ العملية. أثناء تشغيل العملية ، فإنه يتحكم في سجلات المعالج. يتم حفظ محتويات هذه السجلات عندما لا تكون العملية قيد التشغيل. في بيئة متعددة مؤشرات الترابط ، لا يزال هناك كتلة تحكم عملية واحدة ومساحة عنوان مستخدم مرتبطة بالعملية ، ولكن يوجد الآن مكدسات منفصلة لكل مؤشر ترابط ، بالإضافة إلى كتلة تحكم منفصلة لكل مؤشر ترابط يحتوي على قيم السجل والأولوية ومعلومات الحالة الأخرى المتعلقة بمؤشر الترابط.

Threads vs. Processes الفرق بين العملية والمسار التنفيذي

Sr. No	Process	Thread
1	Process is called heavy weight process. العملية ثقيلة الوزن	Thread is called light weight process. المسار التنفيذي خفيف الوزن
2	Process switching needs interface with operating system. تبديل التنفيذ يحتاج تدخل نظام التشغيل	Thread switching does not need to call a operating system and cause an interrupt to the Kernel.
3	In multiple process implementation each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes. تتشارك في المصادر
4	If one server process is blocked no other server process can execute until the first process unblocked. اذا تم ايقاف احدى العمليات يتم ايقاف الجميع	While one server thread is blocked and waiting, second thread in the same task could run.

ايجابيات المسارات Advantages of Threads

1. Thread minimize context switching time. يوفر في وقت التنفيذ
2. Use of threads provides concurrency within a process. يدعم التزامن
3. Efficient communication. فعالة في التواصل
4. Economy.اقتصادية
5. Utilization of multiprocessor architectures . تدعم استخدام اكثر من معالج

Thread Execution States حالات تنفيذ المسار

The key states for a thread are:

- Running
- Ready
- Blocked

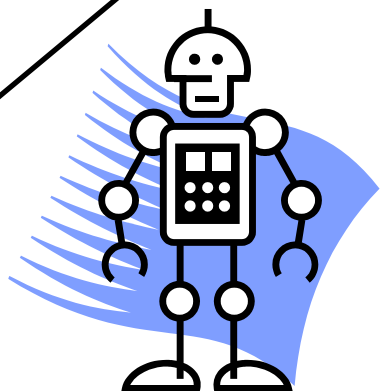
Thread operations associated with a change in thread state are:
العمليات المرتبطة بحالات المسار

- create انشاء
- Block ايقاف
- Unblock ازالة الايقاف
- Finish انهاء

Types of Threads انواع المسارات

User Level Thread
(ULT) على مستوى
المستخدم

Kernel level Thread
(KLT) على مستوى النواة



Types of Threadsانواع المسارات

هناك فئتان عريضتان من تنفيذ الخيوط : -

- خيوط مستوى المستخدم (ULTs)

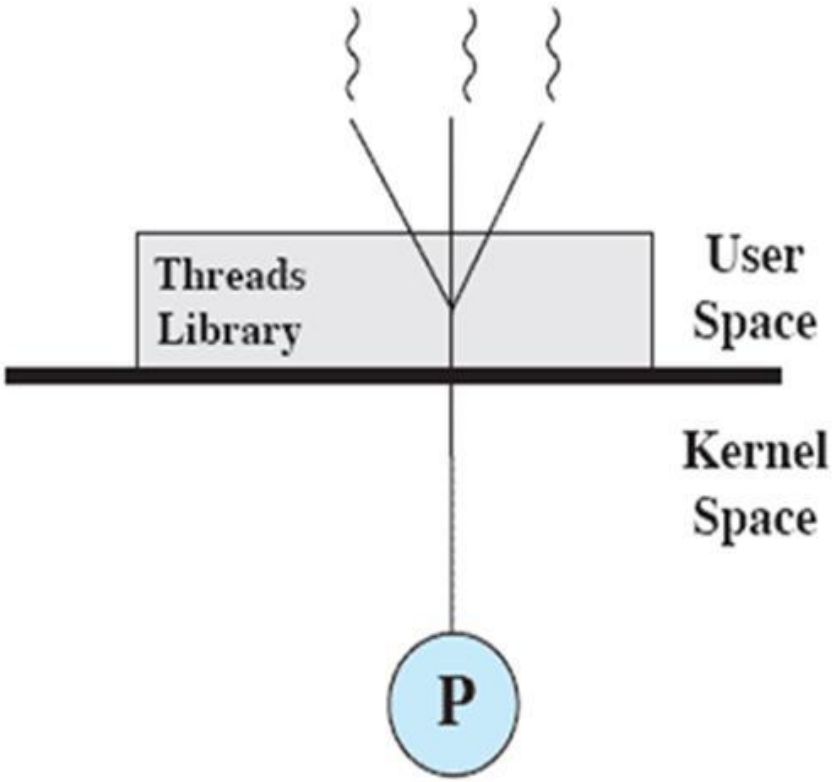
- خيوط مستوى النواة .(KLTs)

-يُشار إلى الأخيرة أيضًا في الأدبيات باسم الخيوط

المدعومة بالنواة أو العمليات الخفيفة الوزن.

User-Level Threads (ULTs)

- Thread management is done by the application ادارة المسار من قبل البرنامج التطبيقي
- The kernel is not aware of the existence of threads
- النواة لا تعرف عن وجود Threads

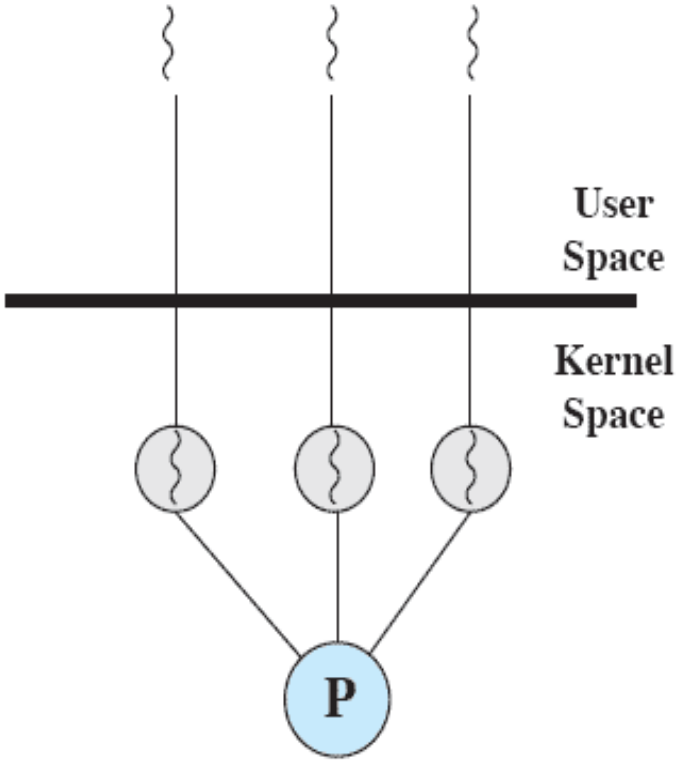


(a) Pure user-level

الخيوط على مستوى المستخدم في مرفق ULTالخالص، يتم تنفيذ كل عمل إدارة الخيوط بواسطة التطبيق ولا يدرك النواة وجود الخيوط .يوضح الشكل 4.5أ نهج ULTالخالص .

الخيوط على مستوى المستخدمفي مرفق ULTالخالص، يتم تنفيذ كل عمل إدارة الخيوط بواسطة التطبيق ولا يدرك النواة وجود الخيوط .يوضح الشكل 4.5أ نهج ULT الخالص .يمكن برمجة أي تطبيق ليكون متعدد الخيوط باستخدام مكتبة خيوط، وهي عبارة عن حزمة من الروتينات لإدارة ULT.تحتوي مكتبة الخيوط على كود لإنشاء الخيوط وتدميرها، وتمرير الرسائل والبيانات بين الخيوط، وجدولة تنفيذ الخيوط، وحفظ سياقات الخيوط واستعادتها.بشكل افتراضي، يبدأ التطبيق بخيط واحد ويبدأ في التشغيل في ذلك الخيط .يتم تخصيص هذا التطبيق وخيطه لعملية واحدة يديرها النواة .في أي وقت يتم فيه تشغيل التطبيق)تكون العملية في حالة التشغيل(، قد يقوم التطبيق بإنشاء خيط جديد لتشغيله داخل نفس العملية .يتم إنشاء الخيط عن طريق استدعاء أداة إنشاء الخيط في مكتبة الخيوط .يتم تمرير التحكم إلى تلك الأداة عن طريق استدعاء إجراء . تقوم مكتبة الخيوط بإنشاء بنية بيانات للخيط الجديد ثم تمرر التحكم إلى أحد الخيوط ضمن هذه العملية الموجودة في حالة الاستعداد، باستخدام بعض خوارزميات الجدولة . عند تمرير التحكم إلى المكتبة، يتم حفظ سياق الخيط الحالي، وعند تمرير التحكم من المكتبة إلى خيط، يتم استعادة سياق هذا الخيط .يتكون السياق بشكل أساسي من محتويات

Kernel-Level Threads (KLTs)



(b) Pure kernel-level

- Thread management is done by the kernel (could call them K*M*T) ادارة المسار من قبل النواة
  - no thread management is done by the application لا يتم إدارة مؤشر الترابط بواسطة التطبيق
    - **Windows** is an example of this approach

الخيوط على مستوى النواة

في مرفق KLTالخالص، يتم تنفيذ كل عمل إدارة الخيوط بواسطة النواة .لا يوجد كود لإدارة الخيوط على مستوى التطبيق، بل واجهة برمجة تطبيقات (API)لمرفق خيوط النواة .يعد Windowsمثالاً على هذا النهج .يوضح الشكل 4.5ب نهج KLTالخالص .تحافظ النواة على معلومات السياق للعملية ككل وللخيوط الفردية داخل العملية.

Kernel-Level Threads (KLTs)

الخيوط على مستوى النواةفي مرفق KLTالخالص، يتم تنفيذ كل عمل إدارة الخيوط بواسطة النواة .لا يوجد كود لإدارة الخيوط على مستوى التطبيق، بل واجهة برمجة تطبيقات (API)لمرفق خيوط النواة .يعد Windowsمثالاً على هذا النهج .يوضح الشكل 4.5ب نهج KLTالخالص .تحافظ النواة على معلومات السياق للعملية ككل وللخيوط الفردية داخل العملية.

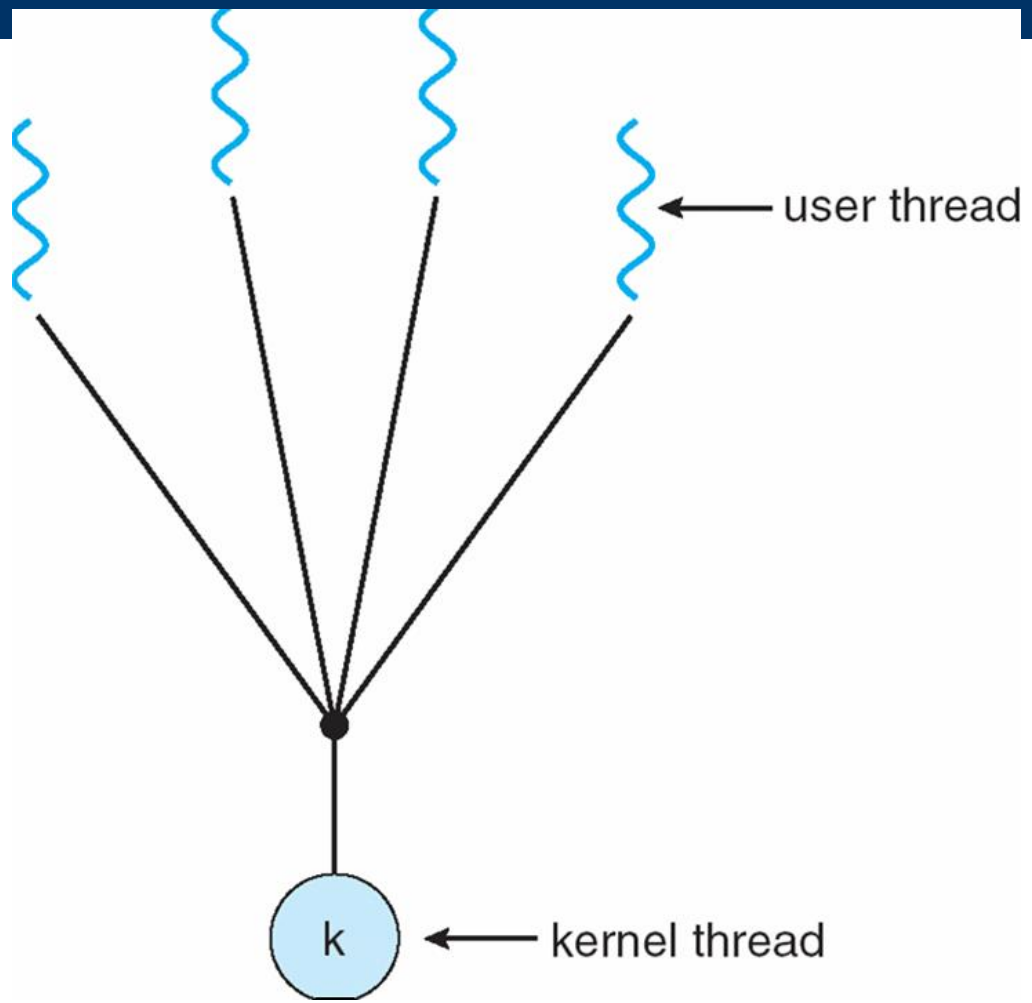
Multithreading Modelsنماذج تعدد المسارات

- Many-to-Oneكثير لواحد
- One-to-Oneواحد لواحد
- Many-to-Manyكثير لكثير

Many-to-One

- **Many user-level** threads mapped to **single** kernel thread.
- تم تعيين العديد من مؤشرات الترابط على مستوى المستخدم إلى مؤشر ترابط واحد. kernel

Many-to-One Model

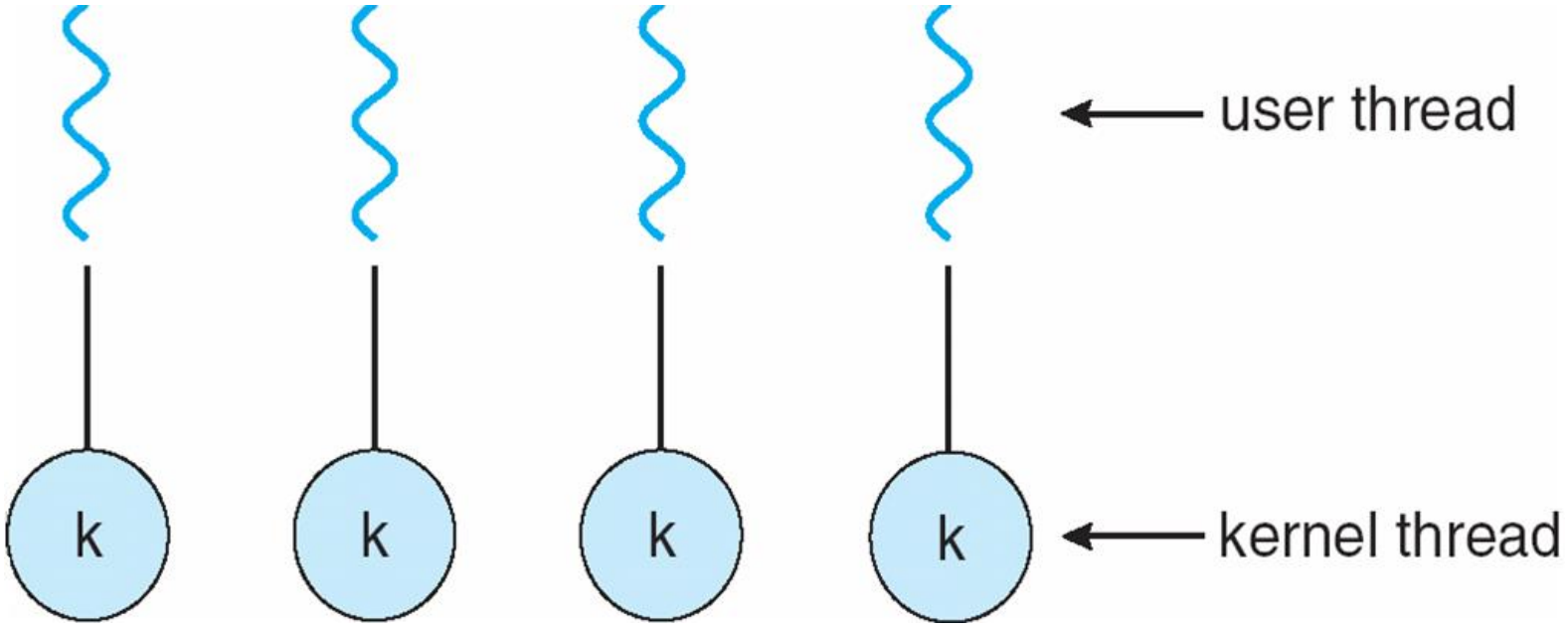


One-to-One

- **Each user-level** thread maps to **kernel thread**
- يتم تعيين كل مؤشر ترابط على مستوى المستخدم إلى مؤشر ترابطkernel

- Examples
  - Windows NT/XP/2000
  - Linux
  - Solaris 9 and later

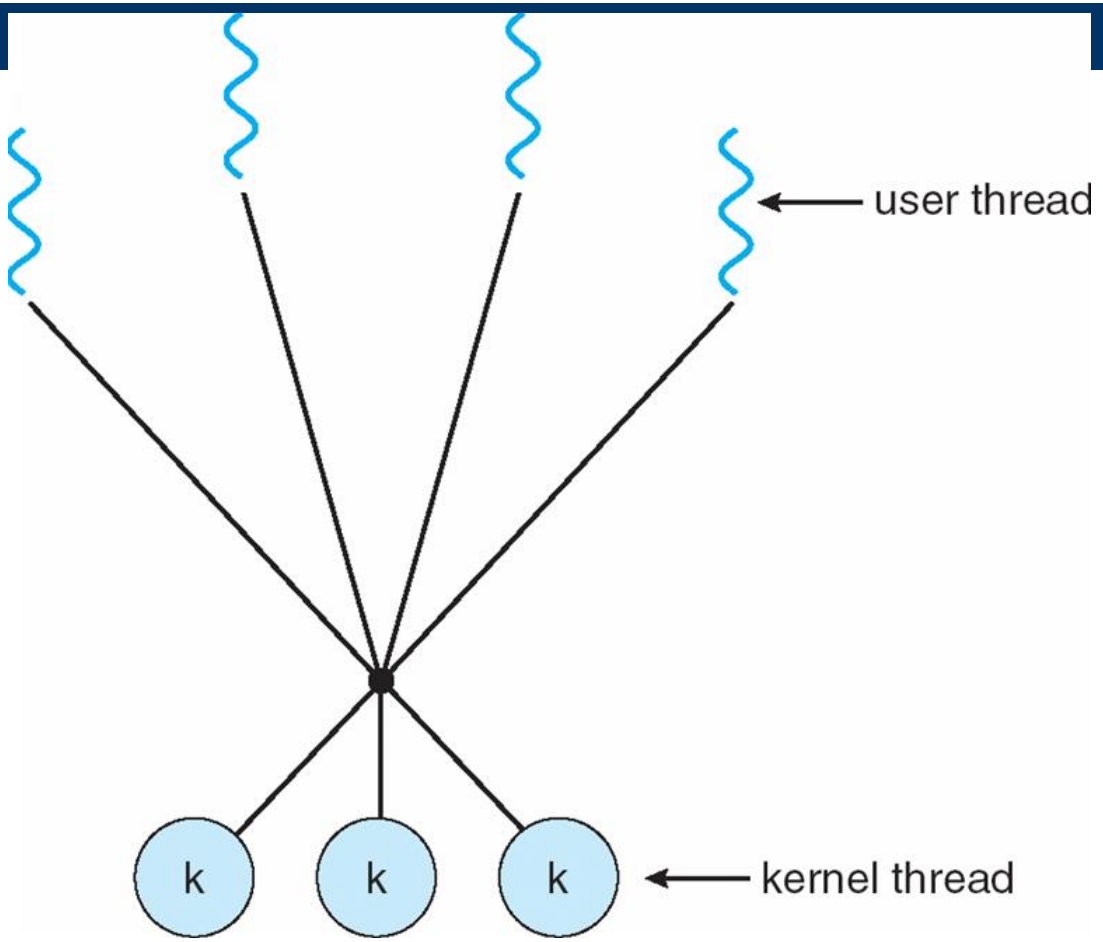
One-to-one Model



Many-to-Many Model

- Allows many user level threads to be mapped to many kernel threads
- يسمح بتعيين العديد من مؤشرات الترابط على مستوى المستخدم إلى العديد من مؤشرات ترابطkernel
- Allows the operating system to create a sufficient number of kernel threads
- يسمح لنظام التشغيل بإنشاء عدد كاف من مؤشرات ترابطkernel
- **Windows** NT/2000 with the *ThreadFiber* package

Many-to-Many Model



Lecture Five

المحاضرة الخامسة

التزامن(Concurrency: Mutual Exclusion and Synchronization)

Multiple Processes

- The central themes of operating system design are all concerned with the management of processes and threads: تهتم جميع الموضوعات المركزية لتصميم نظام التشغيل بإدارة العمليات والخيوط:
  - Multiprogrammingالبرمجة المتعددة
  - Multiprocessingالمعالجة المتعددة
  - Distributed Processingالمعالجة الموزعة
- Big Issue is Concurrency (التزامن)
  - Managing the interaction of all of these processes
  - إدارة التفاعل بين كل هذه العمليات

Multiple Processes

- Multiprogramming:
  - The management of multiple processes within a uniprocessor system. (إدارة أكثر من عملية في معالج واحد)
- Multiprocessing:
  -The management of multiple processes within a multiprocessor. (إدارة أكثر من عملية في أكثر من معالج)
- Distributed processing:
  -The management of multiple processes executing on multiple, distributed computer systems. (إدارة أكثر من عملية في معالجات موزعة)

Multiple Processes

- **Concurrency (التزامن)**
- **القدرة على القيام بعدة مهام في وقت واحد، مثلا الكمبيوتر عن تشغيله يقوم بالعديد من العمليات من تشغيل نظام التشغيل وتشغيل مكافحة الفيروسات والعديد من البرامج في وقت واحد.**
- **التزامن مهم جدا لتمكين أنظمة الكمبيوتر من الاستجابة بسرعة اكبر للمستخدمين، أي انه يحسن في الخدمة للمستخدمين.**
  تخيل لو أنك فتحت ملف وورد وأردت أن تفتح معه مثلا برنامج ريل بلاير، لو لم يكن تزامن لن تستطيع، فلا بد من أن تغلق الوورد أولا ثم تفتح الريل بلير.

Concurrency

Concurrency arises in: (ينشأ التزامن في)

- Multiple applications تطبيقات متعددة
  - Sharing time مشاركة الوقت
- Structured applications التطبيقات المهيكلة
  - Extension of modular design تمديد التصميم المعياري
- Operating system structure هيكل نظام التشغيل
  - OS themselves implemented as a set of processes or threads نظام التشغيل نفسه نفذت كمجموعة من العمليات أو المواضيع

Interleaving and Overlapping Processes

Multiprogramming

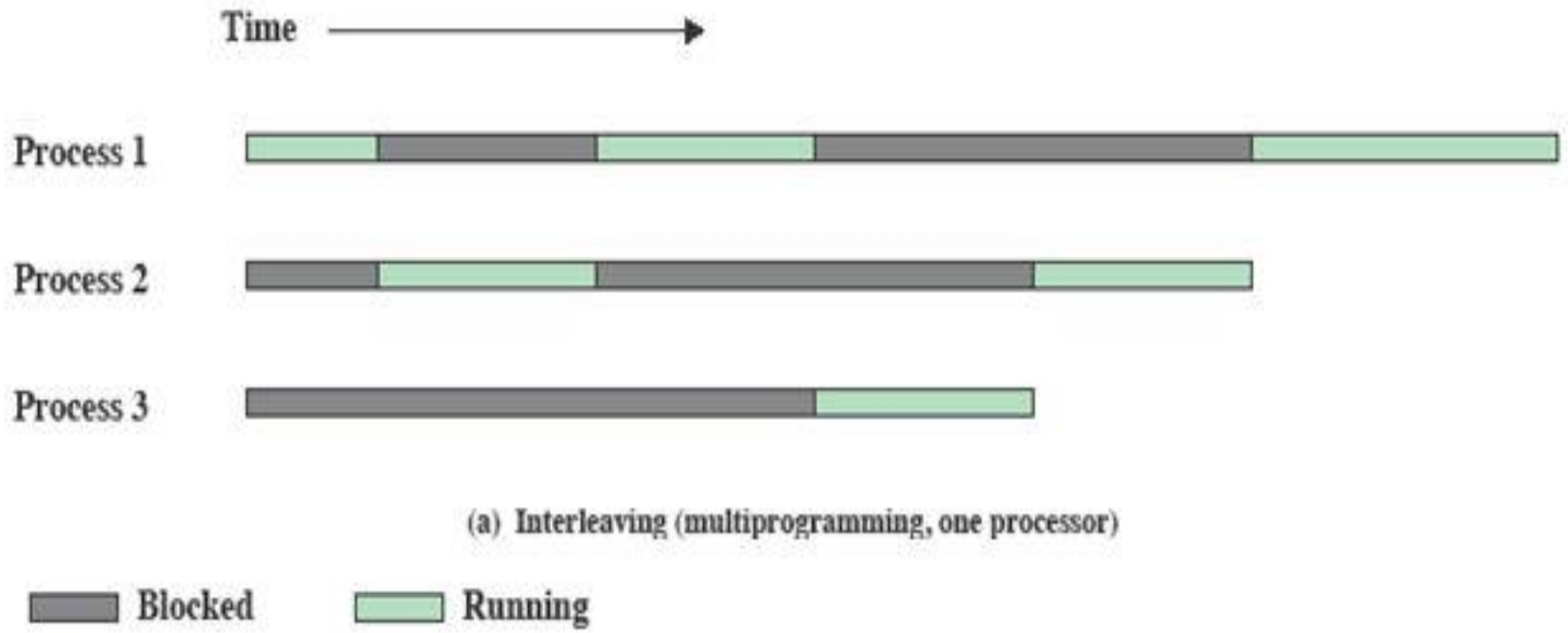


Figure 2.12 Multiprogramming and Multiprocessing

Interleaving and Overlapping Processes

Multiprocessing

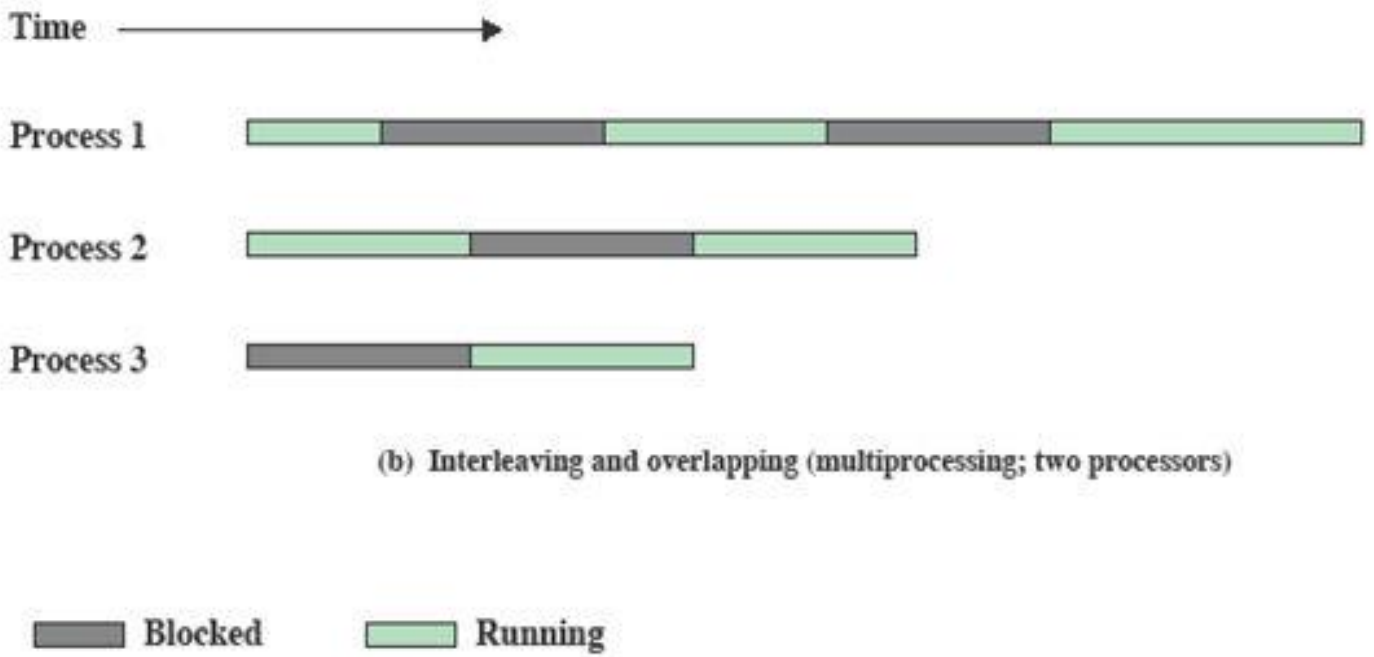


Figure 2.12 Multiprogramming and Multiprocessing

Difficulties of Concurrency

1. The sharing of global resources. (المشاركة في الموارد)

- For example, if two processes both make use of the same global variable and both perform reads and writes on that variable, then the order in which the various reads and writes are executed is critical.

على سبيل المثال ، إذا كانت عمليتان تستخدمان نفس المتغير العام وكلاهما يقوم بالقراءة والكتابة على هذا المتغير ، فإن الترتيب الذي يتم به تنفيذ عمليات القراءة والكتابة المختلفة أمر بالغ الأهمية

Difficulties of Concurrency

2. It is difficult for the operating system to manage the allocation of resources optimally.

(من الصعب إدارة الموارد بالشكل الأمثل)

3. It is very difficult to locate a programming error. (من الصعب تحديد الخطأ في البرمجة)

Enforce Single Access

- If we enforce a rule that only one process may enter the function at a time then:
- فرض قاعدة أن تدخل عملية واحدة فقط في الوقت نفسه لل function
- P1 & P2 run on separate processors
- P1 enters **echo** first,
  - P2 tries to enter but is blocked (تمنع) – P2 suspends
- P1 completes execution
  - P2 resumes and executes echo

Race Condition (سباق الحالة)

- A race condition occurs when(يحدث السباق عندما)
  - Multiple processes or threads read and write data items.
  - So that the final result depends on the order of execution of the processes.
  - تحدث حالة سباق عندما (يحدث السباق عندما) عمليات أو مؤشرات ترابط متعددة قراءة وكتابة عناصر البيانات. بحيث تعتمد النتيجة النهائية على ترتيب تنفيذ العمليات.
- The output depends on who finishes the race last.
- (النتيجة أو القيمة النهائية تعتمد على الترتيب الذي تم به

Operating System Concerns

- What design and management issues are raised by the existence of concurrency?
- The OS must
  - Keep track of various processes
  - Allocate and de-allocate resources
  - Protect the data and resources against interference by other processes.
  - Ensure that the processes and outputs are independent of the processing speed

Operating System Concerns

- ما هي قضايا التصميم والإدارة التي يثيرها وجود التزامن؟ يجب على نظام التشغيل تتبع العمليات المختلفة تخصيص الموارد وإلغاء تخصيصها حماية البيانات والموارد من تدخل العمليات الأخرى. تأكد من أن العمليات والمخرجات مستقلة عن سرعة المعالجة

Process Interaction

- Process Interaction can be defined as:
  - Processes unaware of each other Processes.
  - Indirectly aware of each other Processes.
  - Directly aware of each other.

يمكن تعريف تفاعل العملية على النحو التالي: –

العمليات غير مدركة لبعضها البعض العمليات. تدرك بشكل غير مباشر من بعضها البعض العمليات. ندرك مباشرة من بعضهم البعض

Mutual Exclusion (الاستبعاد المتبادل)

- **Need for mutual exclusion.**
  - Suppose two or more processes require access to a single nonsharable resource, such as a printer.

افترض أن عمليتين أو أكثر تتطلبان الوصول إلى مورد واحد غير قابل للمشاركة، مثل طابعة.

  - Each process will be sending commands to the I/O device, receiving status information, sending data, and/or receiving data. سترسل كل عملية أوامر إلى جهاز الإدخال / الإخراج ، وتلقي معلومات الحالة ، وإرسال البيانات ، و / أو استقبال البيانات.

Mutual Exclusion (الاستبعاد المتبادل)

- We will refer to such a resource as a **critical resource**, and the portion of the program that uses it a **critical section** of the program.

سنشير إلى هذا المورد كمورد مهم ، وجزء البرنامج الذي يستخدمه قسم مهم من البرنامج.

- It is important that only one program at a time be allowed in its critical section. من المهم السماح ببرنامج واحد فقط في كل مرة في قسمه الحرج.

Lecture Six

المحاضرة السادسة

(Deadlock)

Deadlock

- In a multiprogramming environment, many processes compete for a limited number of resources.
- A process requests resources; if the resources are not available at that time, the process enters a wait state.
- It may happen that waiting processes will never again change state, because the resources they have requested are held by other waiting processes.
- This situation is called **deadlock (الجمود)**.

Deadlock

- في البيئة متعدد البرمجة، العديد من العمليات تتنافس على عدد محدود من المصادر (resources).
- العملية تطلب المصادر، إذا كانت هذه المصادر غير متوفرة في هذا الوقت، عندها لابد أن تدخل العملية في حالة انتظار (wait).
- ربما يحدث أن العمليات لا تغير من حالتها مرة أخرى لأن المصادر محجوزة لعمليات أخرى هي الأخرى في حالة انتظار (wait).
- هذا ما يطلق عليه **الجمود** (**Deadlock**)

الجمود Deadlock

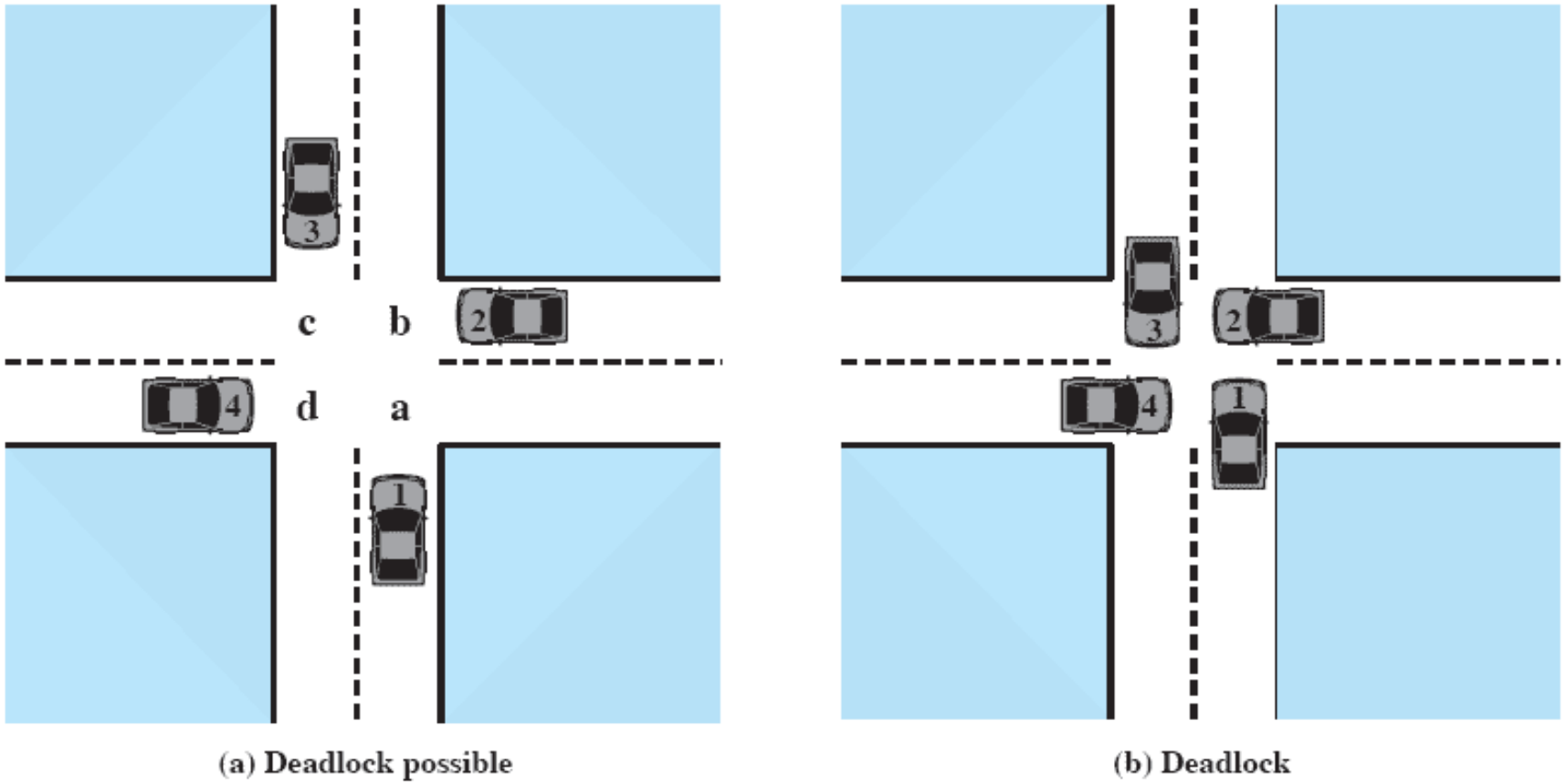


Figure 6.1 Illustration of Deadlock

Conditions for Deadlock

- ***حالة الجمود ممكن أن تظهر في حالة تحقق الشروط الأربعة التالية معاً (simultaneously) في النظام:-***

1. Mutual exclusionالاستبعاد المتبادل
   - Only one process may use a resource at a time قد تستخدم عملية واحدة فقط موردا في كل مرة
2. Hold-and-waitلاحتفاظ والانتظار
   - A process may hold allocated resources while awaiting assignment of others. قد تحتفظ العملية بالموارد المخصصة أثناء انتظار تخصيص موارد أخرى

Conditions for Deadlock

3. No preemption لا استباق
   - No resource can be forcibly removed from a process holding it.

     لا يمكن إزالة أي مورد بالقوة من عملية تحتفظ به.

3. Circular wait الانتظار الدائري
   - A closed chain of processes exists, such that each process holds at least one resource needed by the next process in the chain. توجد سلسلة مغلقة من العمليات ، بحيث تحتوي كل عملية على مورد واحد على الأقل تحتاجه العملية التالية في السلسلة

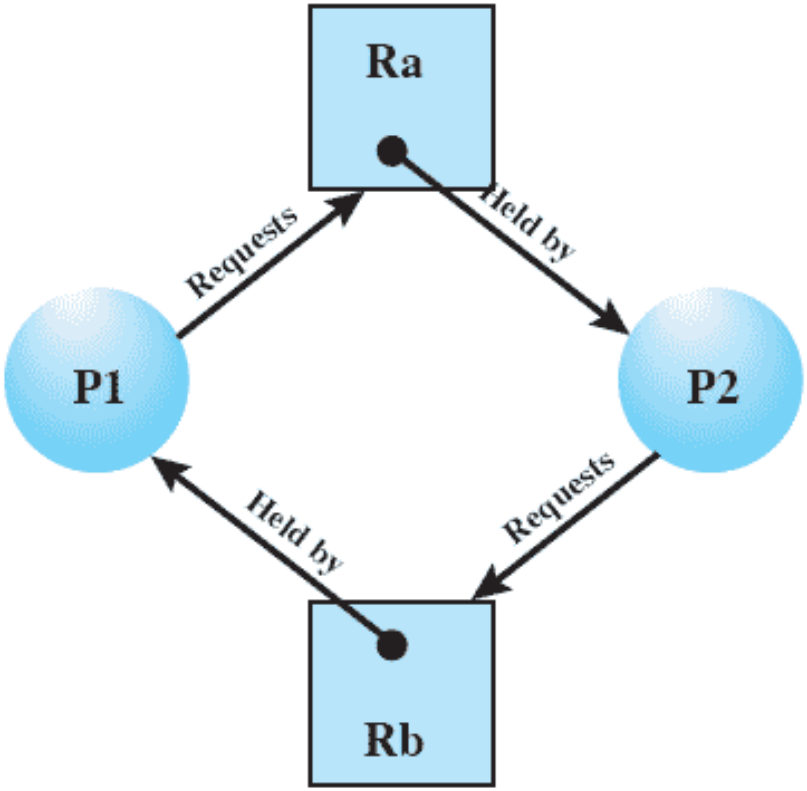
Resource Allocation Graphs الرسوم البيانية لتخصيص الموارد

- Directed graph that depicts a state of the system of resources and processes رسم بياني موجه يصور حالة نظام الموارد والعمليات

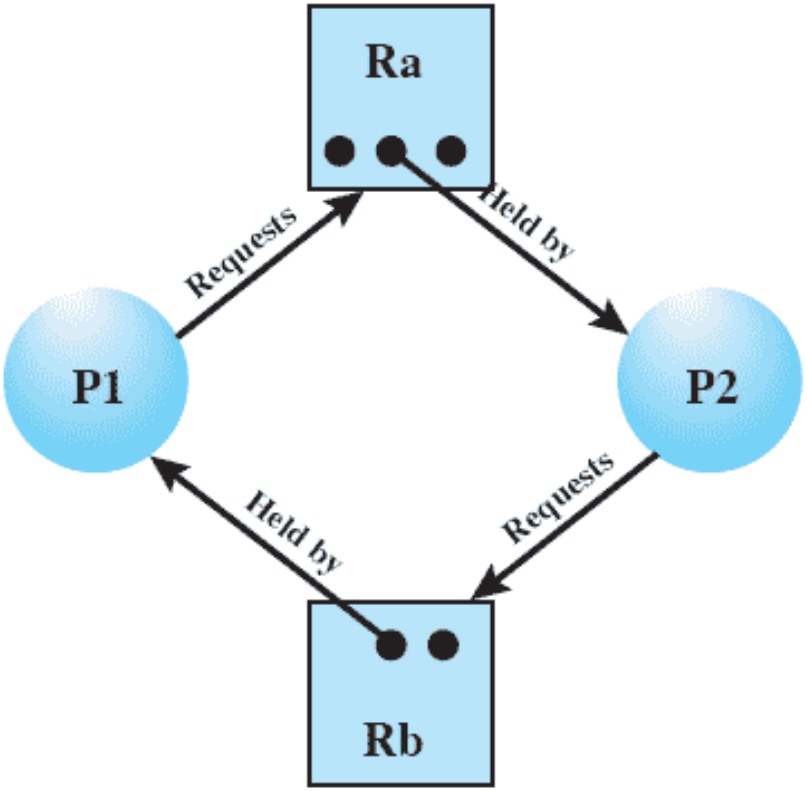
(a) Resouce is requested

(b) Resource is held

Resource Allocation Graphs



(c) Circular wait



(d) No deadlock

Resource Allocation Graphs

الرسوم البيانية لتخصيص الموارد

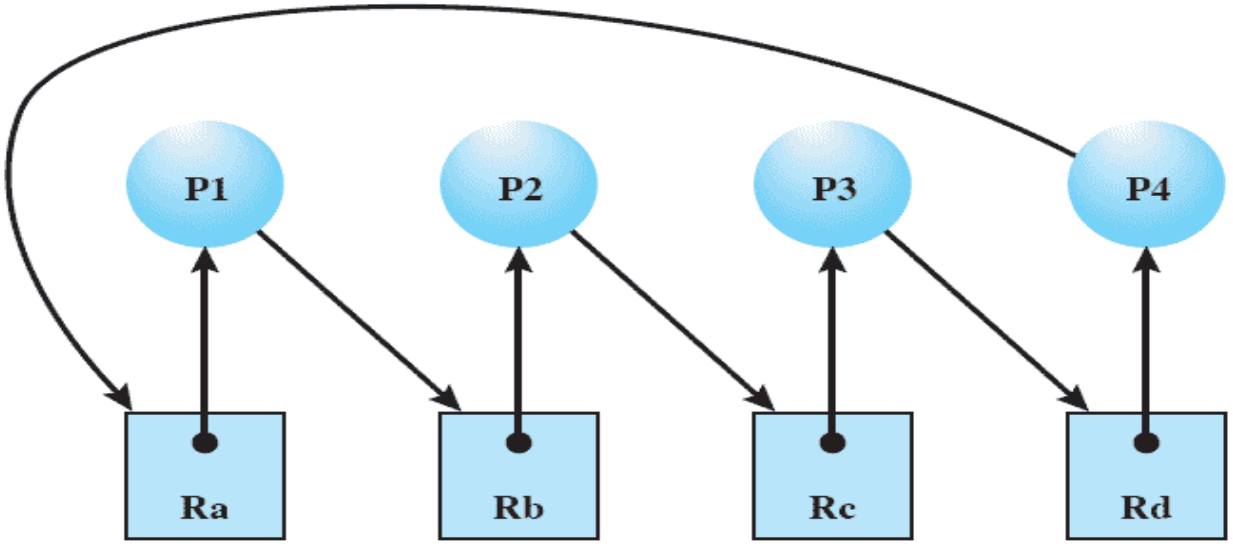


Figure 6.6 Resource Allocation Graph for Figure 6.1b

Possibility of Deadlock احتمال حدوث الجمود

- Mutual Exclusionالاستبعاد المتبادل
- No preemptionلا استباق
- Hold and wait احتفظ وانتظر

Existence of Deadlock حدوث الجمود

- Mutual Exclusionالاستبعاد المتبادل
- No preemptionلا استباق
- Hold and wait احتفظ وانتظر
- Circular waitانتظار دائري

Methods for handling Deadlocks

- ***للتأكد من أن حالة الجمود لن تحدث، على النظام أن يستخدم إحدى الأسلوبين:-***

1. **Deadlock-prevention (منع الجمود)**
2. **Deadlock-avoidance (اجتناب الجمود)**

Deadlock Prevention

2. Hold and Wait
   - Require a process request all of its required resources at one time. تتطلب عملية طلب جميع الموارد المطلوبة لها في وقت واحد

- للتأكد من أن شرط Hold-and-wait لن يتحقق في النظام، يجب أن نضمن أن أي عملية لا تطلب مصادر إضافية طالما أنها تحتفظ بمصادر أخرى.

Methods for handling Deadlocks

- **Deadlock prevention** is a set of methods for ensuring that at least one of the necessary conditions cannot hold.
- أي أن الـ Deadlock prevention هو مجموعة من الطرق للتأكد من أن أحد الشروط الضرورية لن تتحقق.
- **Deadlock avoidance**, on the other hand, requires that the OS be given in advance additional information concerning which resource a process will request and use during lifetime.

Deadlock Prevention

1. Mutual Exclusionالاستبعاد المتبادل
   – The mutual-exclusion condition must hold for non-sharable resources.

- Mutual Exclusion يجب أن يتحقق عند المصادر الغير مشاركة (non-sharable resources).
- على سبيل المثال:
- الطابعة لا تستطيع المشاركة في نفس الوقت مع العديد من العمليات.

Deadlock Prevention

3. No Preemptionلا استباق
   – Process must release resource and request again.

   يجب أن تحرر العملية المورد وتطلب مرة أخرى.

   – OS may preempt a process to require it releases its resources

   قد يستبق نظام التشغيل عملية تتطلب منه إطلاق موارده

4. Circular Waitالانتظار الدائري
   – Define a linear ordering of resource types

   تعريف ترتيب خطي لأنواع الموارد

Deadlock Avoidance

- A decision is made dynamically whether the current resource allocation request will, if granted, potentially lead to a deadlock.
- يتم اتخاذ قرار إذا كان طلب تخصيص الموارد الحالية ممكن أن يؤدي إلى حدوث deadlock.

- Requires knowledge of future process requests.

يتطلب معرفة طلبات العملية المستقبلية.

Recovery from Deadlock إصلاح

- When a detection algorithm determines that a deadlock exists, several alternatives exist.

عندما تحدد خوارزمية الكشف وجود طريق مسدود، توجد عدة بدائل

1. **Process Termination**إنهاء العملية

1. **Resource Preemption**استباق الموارد

Process Termination

- To eliminate deadlocks by aborting a process, we use one of two methods:
- للتخلص من حالات الجمود عن طريق إجهاض عملية ما ، نستخدم إحدى طريقتين

1. Abort all deadlocked processes
   - إنهاء جميع العمليات المسببة ل deadlock
2. Abort one process at a time until the deadlock cycle is eliminated
   - إنهاء عملية واحدة في كل مرة حتى ينتهي deadlock

Process Preemption

- preempt some resources from processes and give these resources to other processes until the deadlock cycle is broken.
  - أخذ بعض الموارد من العمليات وإعطاء هذه الموارد إلى عمليات أخرى حتى ينتهي deadlock

End of Lecture Six