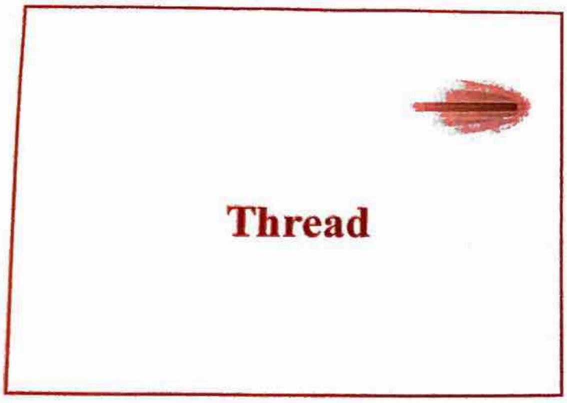
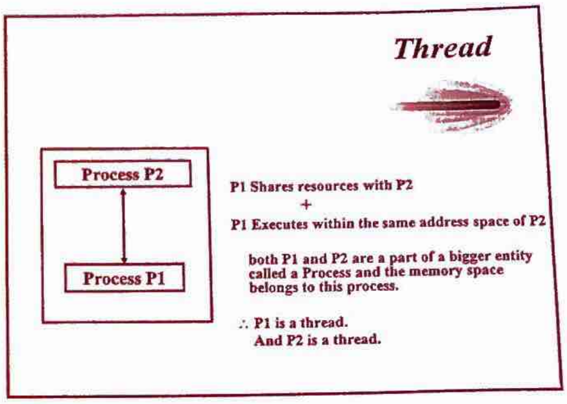


Thread



Thread

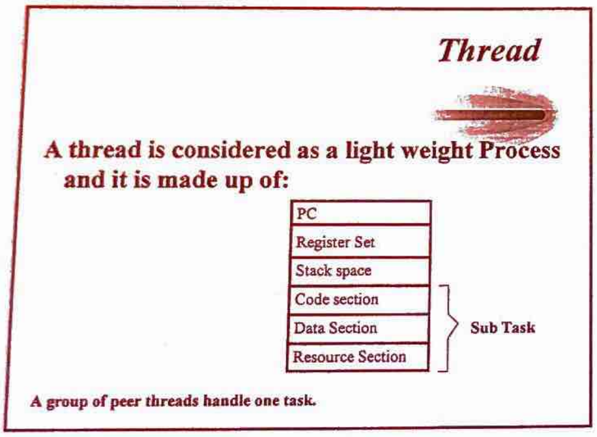
Process P2

Process P1

P1 Shares resources with P2
+
P1 Executes within the same address space of P2

both P1 and P2 are a part of a bigger entity called a Process and the memory space belongs to this process.

∴ P1 is a thread.
And P2 is a thread.



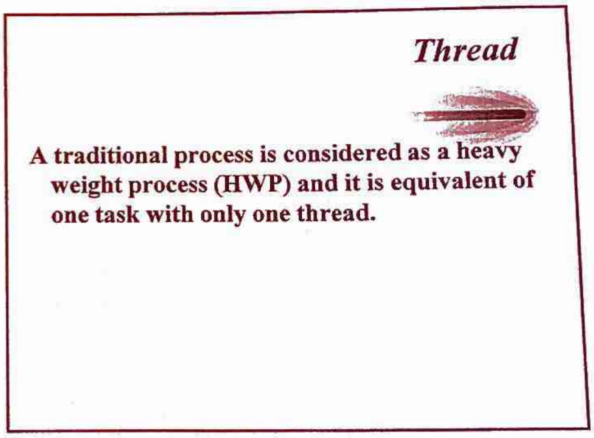
Thread

A thread is considered as a light weight Process and it is made up of:

PC
Register Set
Stack space
Code section
Data Section
Resource Section

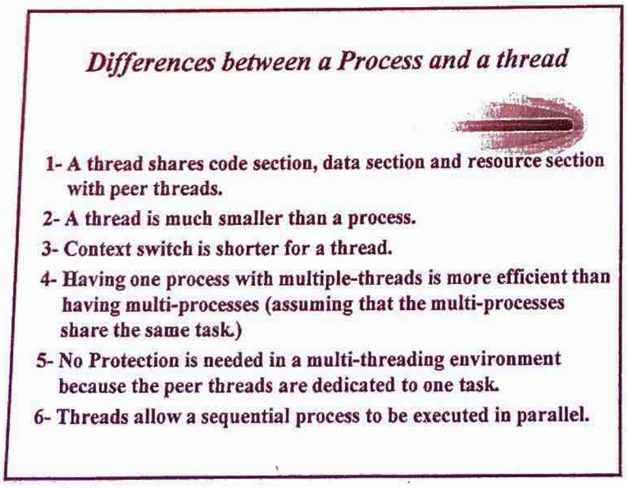
Sub Task

A group of peer threads handle one task.



Thread

A traditional process is considered as a heavy weight process (HWP) and it is equivalent of one task with only one thread.



Differences between a Process and a thread

1- A thread shares code section, data section and resource section with peer threads.

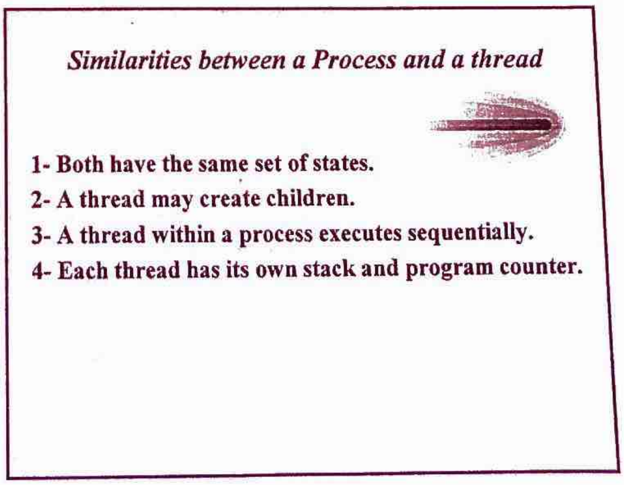
2- A thread is much smaller than a process.

3- Context switch is shorter for a thread.

4- Having one process with multiple-threads is more efficient than having multi-processes (assuming that the multi-processes share the same task.)

5- No Protection is needed in a multi-threading environment because the peer threads are dedicated to one task.

6- Threads allow a sequential process to be executed in parallel.



Similarities between a Process and a thread

1- Both have the same set of states.

2- A thread may create children.

3- A thread within a process executes sequentially.

4- Each thread has its own stack and program counter.

Thread Types

- **User**
- **Kernel**

User Thread

User thread is

created (in user space),

scheduled (within the user space), and

managed by

a user-thread-library (without Kernel intervention).

∴ Kernel is not aware of user-level threads.

User Thread

What is a “user-thread library”?

A system program that is above kernel.

Examples:

P-threads	for POSIX
C-Threads	for MAC
UI-threads	for SOLARIS 2

Kernel Thread

Kernel thread is

created (in the kernel space),

scheduled (within the kernel space), and

managed by

the kernel.

Kernel Thread

kernel threads are used in:

Windows 2000

Windows NT

SOLARIS 2

Digital Unix

Differences in User and Kernel Threads

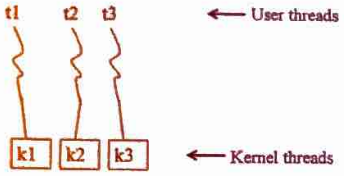
- User threads are faster than Kernel threads *(in terms of creation and scheduling and not execution).*
- *In a single thread hardware,*
  - If a User thread goes to a “wait” state so all of its peers
  - If a Kernel thread goes to a “wait” state, then the Kernel may execute another thread from the peer

Multi-Threading Models

One-to-one

Advantage
- Great parallelism

Disadvantage
- High overhead cost

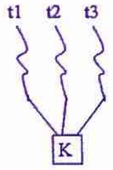


Multi-Threading Models

many-to-one

Advantage
- Low overhead cost

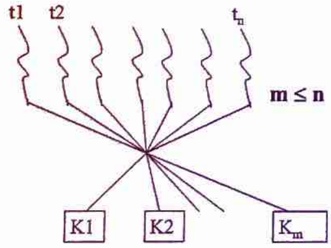
Disadvantage
- No parallelism



Multi-Threading Models

many-to-many

It enjoys, to some degree, the advantages of the other two models:
- Some overhead cost
- Some Parallelism



Threading Issues

- Fork and Exec system calls
- Cancellation
- Signal Handling

Threading Issues

Fork and Exec system calls

Facilitate the creation and execution of a thread.

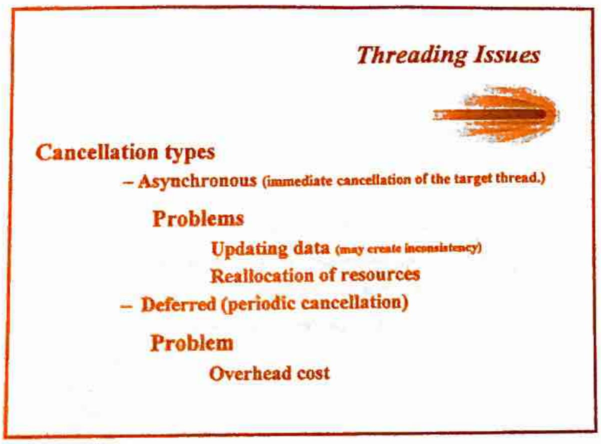
Threading Issues

Cancellation

Termination of a thread before it is completed.

Example

A DBMS retrieving data from a database using several threads. One thread returns the result and thus the other threads must be cancelled.



Threading Issues

Cancellation types

- Asynchronous (immediate cancellation of the target thread.)

Problems

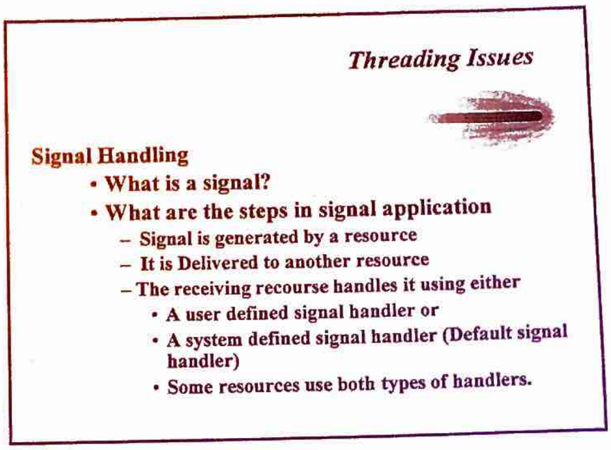
Updating data (may create inconsistency)

Reallocation of resources

- Deferred (periodic cancellation)

Problem

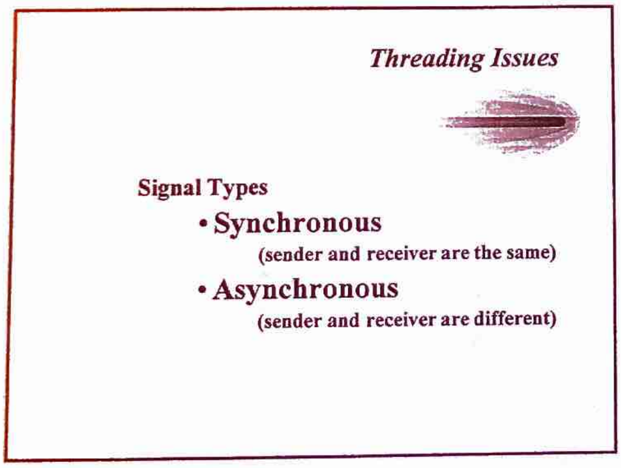
Overhead cost



Threading Issues

Signal Handling

- What is a signal?
- What are the steps in signal application
  - Signal is generated by a resource
  - It is Delivered to another resource
  - The receiving recourse handles it using either
    - A user defined signal handler or
    - A system defined signal handler (Default signal handler)
    - Some resources use both types of handlers.



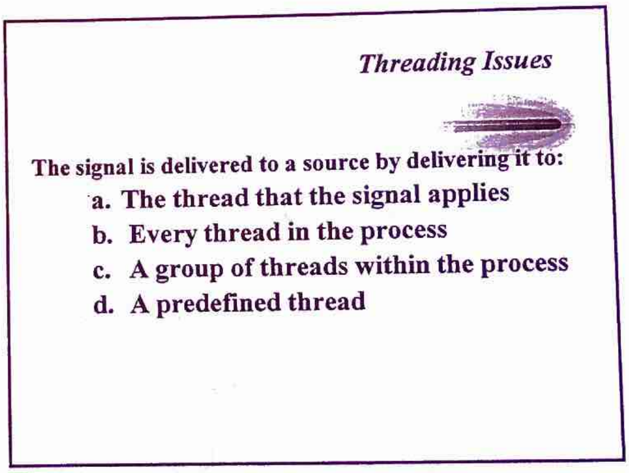
Threading Issues

Signal Types

- Synchronous

  (sender and receiver are the same)
- Asynchronous

  (sender and receiver are different)



Threading Issues

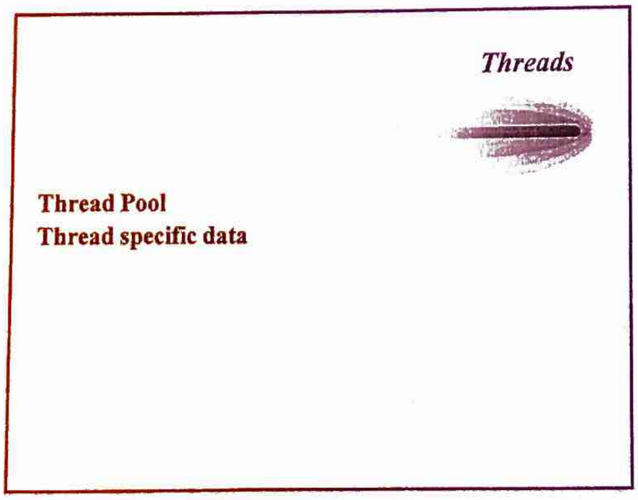
The signal is delivered to a source by delivering it to:

a. The thread that the signal applies

b. Every thread in the process

c. A group of threads within the process

d. A predefined thread



Threads

Thread Pool

Thread specific data