**Multithreading**

Table of Contents

[Process (which has least one thread) 1](#_Toc423620375)

[Threads (aka Thread of Execution) 1](#_Toc423620376)

[Multithreaded program : (a program uses more than more thread at some time) 1](#_Toc423620377)

[Thread states 2](#_Toc423620378)

[Processes vs Threads 2](#_Toc423620379)

[Execution 2](#_Toc423620380)

[Parallelism & Speedup (say 2 threads) 3](#_Toc423620381)

[Multithreading Bug (race condition) 3](#_Toc423620382)

## Process (which has least one thread)

• three major components:

1. Executable program
2. Data created/needed by the program
3. The execution context of the program (file opened, resources allocated..)

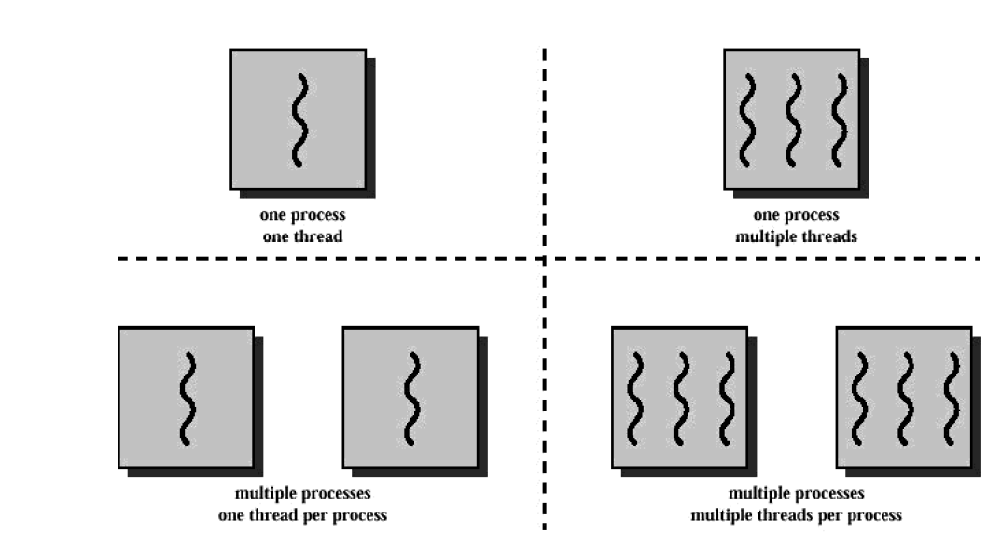
## Threads (aka Thread of Execution)

• means: a sequence of executable commands that can be scheduled to run on the CPU.

## Multithreaded program : (a program uses more than more thread at some time)

• Why? An Important Example: Time-Consuming Background task

FTP transfer, if single thread, then you cannot use UI until the file transfer to finish



## Thread states

• Executing

• Ready

• Blocked

## Processes vs Threads

**Pro for Threads:**

• take less time to create/terminate a thread than a process

• switching between threads is faster (threads in same process communicate easily since they share same memory/files)

**Con for Threads:**

• no protection between threads in the same process

• if any thread encounters an error, whole process is terminated by OS

## Execution

CPU run multi-threaded program by switching the different tasks (thread 1 would execute for 20 ms, then thread 2 for 20 ms, then thread 3 for 20 ms, then back to thread 1 for 20 ms.). To the user, it seems that they run in parallel.

2 ways

• Co-operative Multithreading (e.g. Mac OS 9)

* thread themselves are responsible for yielding the CPU at appropriate time
* Benefits:
  + if every program plays nicely, it can be very efficient
* Shortcomings:
  + some program might never release CPU, could make system unusable.

• Pre-emptive Multirheading (accepted standard for Multithreading)

* the OS manages when each thread runs and when it yields the CPU to another thread.
* Every thread acts as it’s only thread in the system, and it just executes.

## Parallelism & Speedup (say 2 threads)

* Fully parallelized (speed = 2 times)
* Partially parallelized (speed <= 2 times)
* Cannot be parallelized (speed remains unchanged)

## Multithreading Bug (race condition)