CS 370: Operating Systems
[Threads]

Computer Science
Colorado State University

** Lecture slides created by: Shrdeep Pallickara

Instructor: Louis-Noel Pouchet
Spring 2024
Topics covered in this lecture

- Background
- Rationale for threads
- Thread model
- Benefits of multithreaded programming
THREADS
Some background on threading

- Exploited to make programs easier to write
  - Split programs into separate tasks

- Took off when GUls became standard
  - User perceives better performance
    - Programs did not run faster: this was an illusion
    - Dedicated thread to service input OR display output

- Growing trend to exploit available processors on a machine
What are threads?

- Miniprocesses or lightweight processes
- Why would anyone want to have a kind of process within a process?
The main reason for using threads

- In many applications *multiple activities* are going on at once
  - Some of these may block from time to time

- Decompose application into multiple sequential threads
  - Running in *quasi-parallel*
Isn’t this precisely the argument for processes?

- Yes, but there is a new dimension …
- Threads have the ability to share the address space (and all of its data) among themselves
- For several applications
  - Processes (with their separate address spaces) don’t work
Threads are also lighter weight than processes

- **Faster** to create and destroy than processes
- In many systems thread creation is 10-100 times faster
- When number of threads needed changes dynamically and rapidly?
  - Lightweight property is very useful
Threads:
The performance argument

- When all threads are CPU bound all the time?
  - Additional threads would likely yield no performance gain

- But when there is substantial computing and substantial I/O
  - Having threads allows activities to overlap
  - Speeds up the application possibly
AN EXAMPLE APPLICATION
WORD PROCESSOR
Our Word Processor

- Displays document being created on the screen
- Document formatted exactly as it will appear on a printed page
Let’s take a look at someone editing a 800-page document

- User deletes one sentence from Page-1 of a 800-page document

- Now user wants to make a change on page 600
  - Either go to that page or search for term that only appears there
Word processor *does not know* what’s the first line on page 600

Word processor has to **reformat** entire book up to page 600

Threads could help here …
Suppose the word processor is written as a 2-threaded program

- One thread **interacts** with the user
- The second thread handles **formatting** in the background
- As soon as the sentence is deleted
  - Interactive thread tells formatter thread to format the book
While we are at it, why not add a third thread?

- Automatically save file every few minutes
- Handle disk backups *without interfering* with the other 2 threads
What if the program were single threaded?

- Whenever disk backup started
  - Commands from keyboard/mouse would be **ignored** till backup was finished
  - User perceives sluggish performance

- Alternatively, keyboard/mouse events could **interrupt** the disk backup
  - Good performance
  - Complex, interrupt-driven programming
With 3 threads the programming model is simpler

- First thread interacts with the user
- Second thread reformats when told to
- Third thread writes contents of RAM on to disk periodically
Three separate processes WOULD work here

- **All three** threads need to operate on document

- By having 3 threads instead of 3 processes
  - ① The threads share a **common memory**
  - ② Have access to document being edited

- Using processes would require setting up shared memory space, synchronizations, IPC etc. Doable, but much more tedious
  - Tend to use threads when working on the same data within the process
Applications are typically implemented as a process with multiple threads of control

- Perform different tasks in the application
  - Web browser
    - Thread A: Render images and text
    - Thread B: Fetch network data

- Assist in the performance of several similar tasks
  - Web Server: Manages requests for web content
    - Single threaded model: One client at a time
      - Poor response times
    - Multithreaded model: Multiple clients served concurrently
To continue, go to this PDF

- [https://www.cs.colostate.edu/~cs370/Spring24/lectures/CS370-L7-Threads.pdf](https://www.cs.colostate.edu/~cs370/Spring24/lectures/CS370-L7-Threads.pdf)

- Today’s lecture is unfortunately over 2 slide sets, due to a last-minute powerpoint issue...!
The contents of this slide-set are based on the following references

