

# CSCE 240: Advanced Programming Techniques

Lecture 27: On-request Topic: Threading,  
Common Project Coordination Discussion

Lecture 28: Project Presentations

---

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

18<sup>TH</sup> AND 20<sup>TH</sup> APRIL 2023

***Carolinian Creed: “I will practice personal and academic integrity.”***

**Credits:** Some material reused with permission of Dr. Jeremy Lewis.  
Others used as cited with thanks.

# Organization of Lectures 27 and 28

---

- Introduction Section
  - Recap of Lecture 26 – Quiz 2
- Main Section
  - Class 27
    - Threading
    - Common project: discussion and coordination
  - Class 28: Project Presentation
- Concluding Section
  - About next lecture – Lecture 29 – Concluding Lecture
  - Ask me anything

# Introduction Section

---

# Recap of Lecture 26

- Quiz 2 based on roles of common project

- Responsibilities for final chatbot
  - R1: Obtain content (PA1)
  - R2: Data integration and optimization (PA2)
  - R3: Handle user interaction (PA3, PA4)
  - R4: Show results and statistics (P5)
  - R5: Integrate – build final system
  - R6: Test – evaluate system and drive improvements by R1-R5.

- Prog 1: extract data from the district [\[prog1-extractor\]](#)
- Prog 2: process it (extracted data) based on questions [\[prog2processor\]](#)
- Prog 3: make content available in a command-line interface [\[prog3-ui\]](#)
- Prog 4: handle any user query [\[prog4-userintent2querymapper\]](#)
- Prog 5: report statistics on interaction of a session, across sessions [\[prog5-sessionlogger\]](#)

# Main Section

---

# On-request Topic: Threading

---

# Concept: Computer Architecture – Number of Instructions and Data

- Single Instruction, Single Data – SISD
  - Basic computer architecture; Example: early personal computers
- Single Instruction, Multiple Data – SIMD
  - Example: Vector processing machine (Cray)
- Multiple Instruction, Single Data – MISD
  - Example: real-time systems
- Multiple Instruction, Multiple Data – MIMD
  - Example: gaming console, modern computers

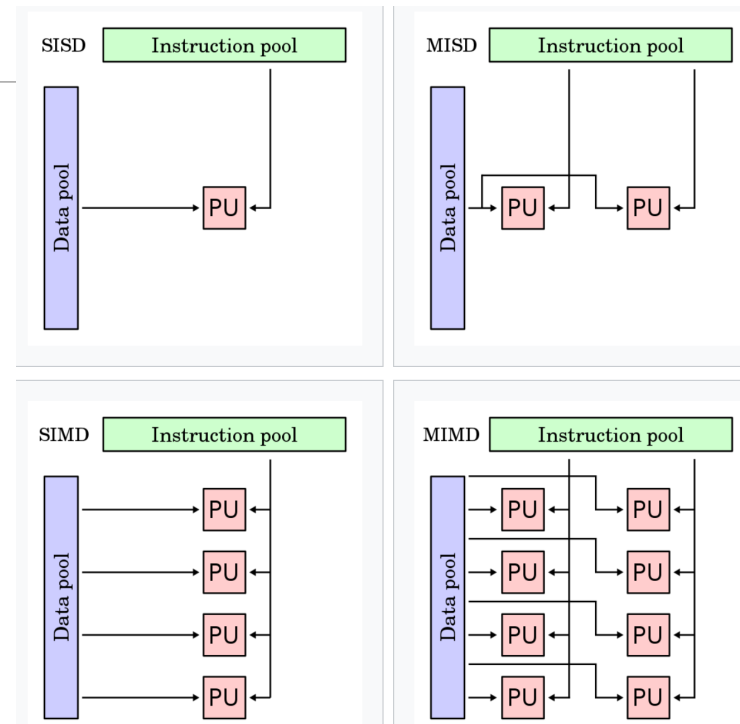


Image credit: [https://en.wikipedia.org/wiki/Flynn%27s\\_taxonomy](https://en.wikipedia.org/wiki/Flynn%27s_taxonomy)

# Concept: Threading

---

- Unit of work on a computer are executed by
  - Program – set of instructions, often run by a single process on a hardware.
  - Processes – unit of work running on a hardware. The same program can be run by multiple processes on a suitable hardware.
  - Threads – a process can be run using multiple threads.
    - Any unit of work that can be done
    - Often running on the same hardware simultaneously
    - Can be in the states: new, ready, running, waiting, terminated, and suspended.
- Related term: parallel or concurrent programming

Credit: <https://www.geeksforgeeks.org/difference-between-process-and-thread/>



# Benefits of Multi-Threaded Programming

---

- Easier programming for applications that are inherently concurrent. Example: UI
- Faster execution in many cases. Example: numeric computation – max, recurrence relations

Code example: [https://github.com/biplav-s/course-adv-proglang-s23/blob/main/sample-code/CandC%2B%2B/Class27 Threading/src/Class27 Threading.cpp](https://github.com/biplav-s/course-adv-proglang-s23/blob/main/sample-code/CandC%2B%2B/Class27%20Threading/src/Class27%20Threading.cpp)

# Illustration: Parallelization With Multi-Threaded Programming

---

•

---

## Algorithm 2: Fibonacci numbers (non-parallel)

---

```
FIB(n)
if  $n \leq 1$  then
|   return  $n$ 
else
|    $x = \text{FIB}(n-1)$ 
|    $y = \text{FIB}(n-2)$ 
|   return  $(x+y)$ 
end
```

Image credit: <https://www.baeldung.com/cs/multithreaded-algorithms>

---

## Algorithm 3: Fibonacci numbers (parallel)

---

```
P-FIB(n)
if  $n \leq 1$  then
|   return  $n$ 
else
|    $x = \text{spawn } P\text{-FIB}(n-1)$  // parallel execution
|    $y = \text{spawn } P\text{-FIB}(n-2)$  // parallel execution
|   sync // wait for results of  $x$  and  $y$ 
|   return  $x + y$ 
end
```

# Discussion and Considerations

---

- How many threads to create?
  - Too little: may not benefit, easier programming
  - Too many: wastes resources, harder programming
- When to create?
  - Fixed at beginning
  - Dynamically decide at runtime
- Debugging multi-threaded programs is a challenge
  - Knowledge of computer architecture is necessary

# Project Coordination and Discussion

---

# Apr 20, 2023: Project Presentation

---

# Project Presentation Instructions

---

- Scope: diseases, data sources, language
- Specification: Prog. Language, code organization, reuse of which student's code
  - Discussion by PAs: PA1, .. , PA6
  - Discussion by roles: R1-R6
- Live demo required
  - Video: suggested to have before
- Experience developing the chatbot
- After class
  - Course paper

# About Next Lecture – Lecture 27

---

# Lectures 27-28: Project Presentations

---

- Shared code in new repo in Github by R5
- Slides to be used from common chatbot GitHub

23	Apr 4 (Tu)	Advanced: Templates	
24	Apr 6 (Th)	AI / ML and Programming	Prog 5 – end
25	Apr 11 (Tu)	Project code summary – student presentation for reuse Review material for Quiz 2	HW 6 due Prog 6 – assembling start
26	Apr 13 (Th)	In class test	Quiz 2 – In class
27	Apr 18 (Tu)	Multi-threading, Project discussion	
28	Apr 20 (Th)	Project presentation	Prog 6 - due Last day of class
	Apr 25 (Tu)		Reading Day
29	May 2 (Tu)	9am – Final Overview	Examination