University *of* Dayton

Department of Computer Science

CPS 475/575

Secure Application Development

Lecture 7 – Concurrent Programming & Concurrent Programming hands-on in Java

Phu Phung

2/4/2020

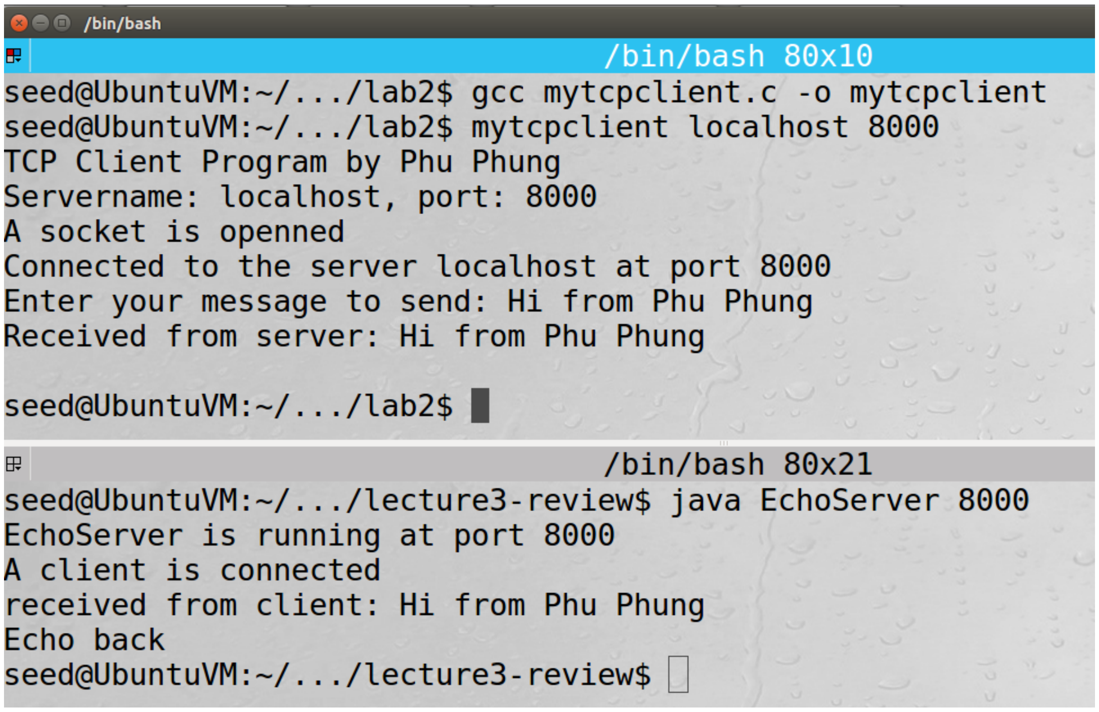
Today’s agenda

- Concurrent Programming
- Introduction to Lab 3
- Concurrent Programming with multi-threading in Java

Review: the EchoServer.java program
(introduced in Lecture 3 and tested in Lab 2)

EchoServer.java with mytcpclient.c – Demo

- The server (EchoServer.java) accepts one client and exits the application if the client exits



/bin/bash

/bin/bash 80x10

```
seed@UbuntuVM:~/.../lab2$ gcc mytcpclient.c -o mytcpclient
seed@UbuntuVM:~/.../lab2$ mytcpclient localhost 8000
TCP Client Program by Phu Phung
Servername: localhost, port: 8000
A socket is openned
Connected to the server localhost at port 8000
Enter your message to send: Hi from Phu Phung
Received from server: Hi from Phu Phung

seed@UbuntuVM:~/.../lab2$ 
```

/bin/bash 80x21

```
seed@UbuntuVM:~/.../lecture3-review$ java EchoServer 8000
EchoServer is running at port 8000
A client is connected
received from client: Hi from Phu Phung
Echo back
seed@UbuntuVM:~/.../lecture3-review$ 
```

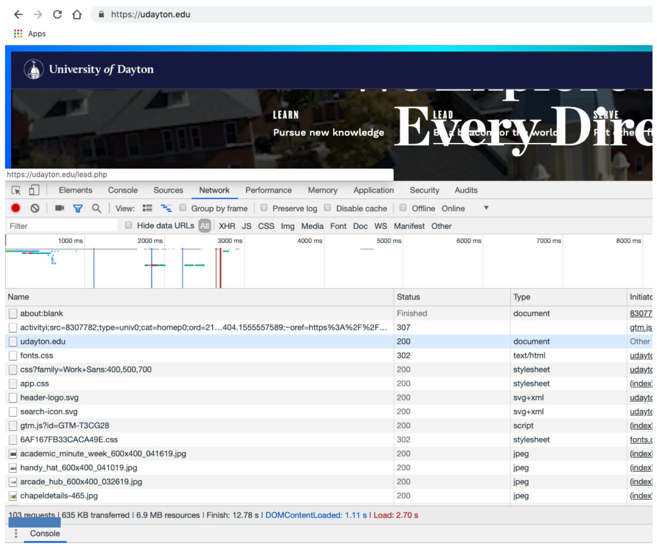
Review: the EchoServer.java program
(introduced in Lecture 3 and tested in Lab 2)

- This server program
  - Accepts one client and exits the application if the client exits
  - Is not a typical server application
- A typical server application
  - Can handle multiple clients at a same time (concurrently)

Why and How?

A realistic motivating example

- Consider the scenario when you access the website https://udayton.edu
  - How many HTTP requests do you think your browser will send to the server?
    - ~100
  - Do you think that the browser will do these steps in sequence, i.e., complete one request then send another?
    - If so, it will slow down the web
      - These should be concurrent
        - » The key idea of concurrent programming

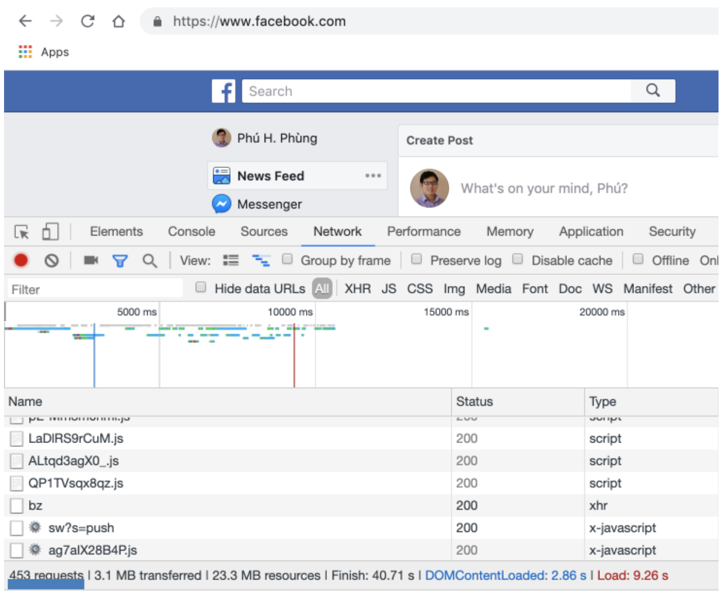

https://udayton.edu
Apps
University of Dayton
LEARN
Pursue new knowledge
LEAD
SERVE
Every Dire
https://udayton.edu/lead.php
Elements Console Sources Network Performance Memory Application Security Audits
View: Group by frame Preserve log Disable cache Offline Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
1000 ms 2000 ms 3000 ms 4000 ms 5000 ms 6000 ms 7000 ms 8000 ms
Console

Name	Status	Type	Initiato
about:blank	Finished	document	83077
activityi;src=8307782;type=univ0;cat=homep0;ord=21…404.1555557589;~oref=https%3A%2F%2F…	307		gtm.js
udayton.edu	200	document	Other
fonts.css	302	text/html	udayto
css?family=Work+Sans:400,500,700	200	stylesheet	udayto
app.css	200	stylesheet	(index)
header-logo.svg	200	svg+xml	udayto
search-icon.svg	200	svg+xml	udayto
gtm.js?id=GTM-T3CG28	200	script	(index)
6AF167FB33CACA49E.css	302	stylesheet	fonts.c
academic_minute_week_600x400_041619.jpg	200	jpeg	(index)
handy_hat_600x400_041019.jpg	200	jpeg	(index)
arcade_hub_600x400_032619.jpg	200	jpeg	(index)
chapeldetails-465.jpg	200	jpeg	(index)

103 requests | 635 KB transferred | 6.9 MB resources | Finish: 12.78 s | DOMContentLoaded: 1.11 s | Load: 2.70 s

Another similar example

- Consider the Facebook server
  - Each client (browser) sent > 400 requests in a single access
  - The server need to handle hundreds of thousands of clients at the same time
- Sequential programming is infeasible


https://www.facebook.com
Apps
Search
Phú H. Phùng
Create Post
News Feed
Messenger
What's on your mind, Phú?
Elements Console Sources Network Performance Memory Application Security
View: Group by frame Preserve log Disable cache Offline
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
5000 ms 10000 ms 15000 ms 20000 ms

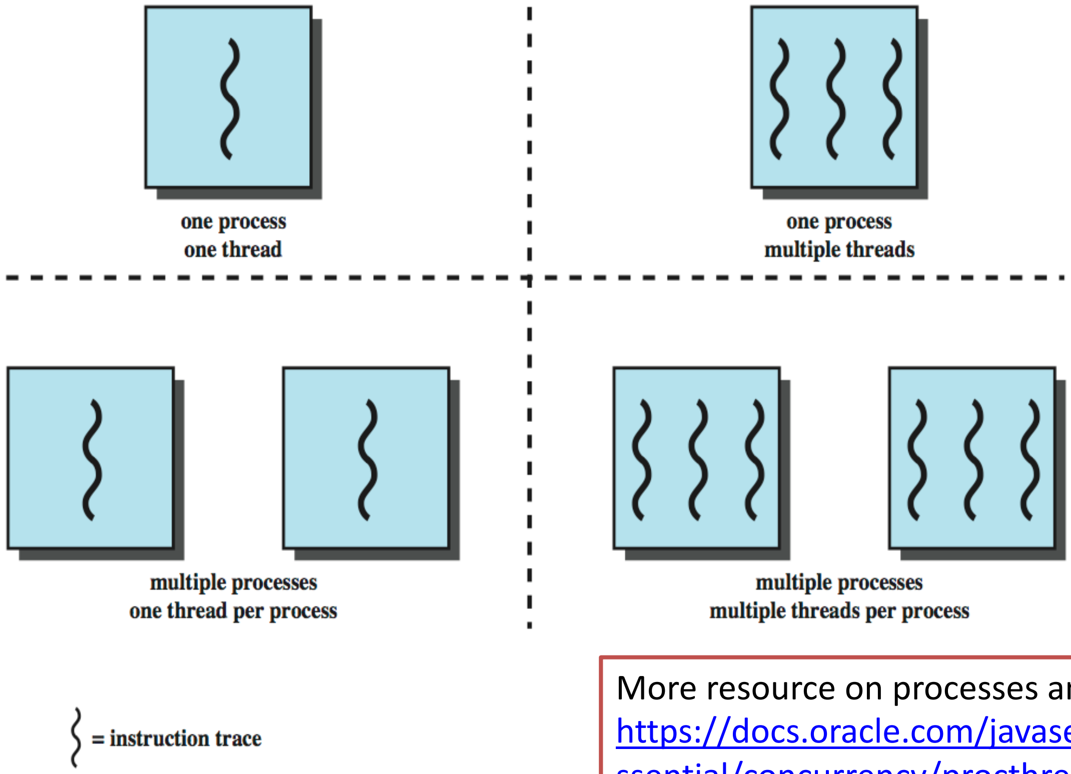
Name	Status	Type
LaDIRS9rCuM.js	200	script
ALtqd3agX0_.js	200	script
QP1TVsqx8qz.js	200	script
bz	200	xhr
sw?s=push	200	x-javascript
ag7alX28B4P.js	200	x-javascript

453 requests | 3.1 MB transferred | 23.3 MB resources | Finish: 40.71 s | DOMContentLoaded: 2.86 s | Load: 9.26 s

Introduction to the Concurrent Programming Concept

- A programming paradigm that supports concurrent computing
  - Multiple computations are executed during overlap time periods
    - i.e., concurrently, in opposite of sequentially
- Normally supported by concurrent programming languages
  - Use language constructs of concurrency, e.g., multi-threading

Review: Threads and Processes



More resource on processes and threads: https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html

Figure credit: Saverio Perugini
Reference: W. Stallings. Operating Systems: Internals and Design Principles

Concurrent Programming with Multi-threading

- Implemented in Programming Languages supported by Operating Systems level
  - Multiple threads in a program (process) can run concurrently
    - Can share the same global resources of the program
    - Handle input/output independently with each other
      - Non-blocking I/O (a single-threaded program blocks on a long-running task)

An approach of Multi-Threaded in Java

- Extending a Thread Class, e.g.,:
  `class EchoServerThread extends Thread {..}`
  - implement a `run()` method, e.g.,:

    ```
    public void run() {..}
    ```
  - create a new Thread object (e.g., `EchoServerThread`) and call `start()` to run the thread, e.g.,:

    ```
    new EchoServerThread().start();
    ```

Multi-Threaded in Java - another approach

- Implementing a Runnable Interface, e.g.,:
  `class EchoServer implements Runnable {..}`
  - implement a run() method: `public void run() {..}`
  - instantiate a Thread object: `new Thread(Runnable obj, ..);`
  - Create the Runnable object (e.g., `EchoServer)` and call start() to run the thread

Example: Multi-Threaded for EchoServer.java

- Let’s use the first approach: create a new Thread class:

```
class EchoServerThread extends Thread {
   public EchoServerThread(/*???*/){
   }
   public void run(){
      System.out.println("A new thread
for client is running");
   }
}
```