Producer and Consumer, Synchronization

Raspberry Pi Setup
Assignment Review

- You are supposed to implement a solution to Producer and Consumer problem, using a circular FIFO buffer.
- There can be one or more than one consumers. Same for producers.
- Each producer is supposed to generate a certain number of elements. Elements will be randomly generated uppercase letters, 'A'…'Z'. It also keeps track of the sum of all elements it produced.
- Consumers are supposed to consume the elements, produced by the Producers. Each consumer will keep its own sum of elements that it consumed.
- All Producers and Consumers report the item produced/consumed along with the index and timestamp with nanosecond resolution.
Producer.java

- A Producer will produce the total number of elements which is passed as the second argument to it.
- A seed is used to set the random number generator to generate the same sequence every time the same seed is given. It is passed as the fourth argument to it.
- Generate a random uppercase letter 'A'...'Z' and insert into buffer.
- A producer cannot insert an element into the buffer when the buffer is full.
- If an item is inserted into buffer, it is added to the checksum the Producer is keeping.
Consumer.java

- A Consumer consumes an element from the buffer.
- Each Consumer will consume the (total number of elements / number of consumers) elements.
- A Consumer cannot consume an element when the buffer is empty.
- Once a Consumer consumes an element from the buffer successfully, it adds that number to the checksum it is keeping.
Buffer.java

- Buffer.java contains the circular FIFO buffer that will be used among all the producers and the consumers.
- It also has the required functions that is used to insert or remove an element, and it returns the appropriate values.
- It may additionally have, other functions such as isFull(), isEmpty(), etc. depending on your implementation.
Coordinator.java

• This acts as the Coordinator program for this assignment.
• It creates one instance of the buffer, creates the required threads of producers, creates the required number of threads of consumers, and then waits for all of them to finish.
Synchronization in Java

- Java has inbuilt monitors
  - Allows threads to have mutual exclusion
  - Allows threads the ability to wait (block) for a condition to become true
- Built in thread class can be extended and used
  - Instantiate and use myThread.start()
  - @Override run() to change what a thread does
- Signalling is done using
  - wait()
  - notify() or notifyAll()
public class PhilosopherThread extends Thread {
    @Override
    public void run() {
        //The entry point for each thread
    }
}
Creating and Starting threads

PhilosopherThread Socrates = new PhilosopherThread(table, seat);
Socrates.start(); //begins our Socrates thread and invokes the run() method
Synchronization
Synchronized methods

• A piece of logic marked with synchronized becomes a synchronized block, allowing only one thread to execute at any given time.

```java
public synchronized void pickup(int i) throws InterruptedException {
    //Synchronized code goes in here
}
```
wait(), notify() and notifyAll()

- wait()
  - Causes current thread to wait until another thread invokes the notify() or notifyAll() method
- notify()
  - notify() wakes up one thread waiting for the lock
- notifyAll()
  - The notifyAll() method wakes up all the threads waiting for the lock; the JVM selects one of the threads from the list of threads waiting for the lock and wakes that thread up
Makefile

COMPILER= javac
JRE= java
FILES= DiningPhilosophers.java PhilosopherTable.java PhilosopherThread.java
EXE= DiningPhilosophers

all:
   $(COMPILER) $(FILES)
run:
   $(JRE) $(EXE)
clean:
   rm *.class
JFLAGS = -g
JC = javac
.SUFFIXES: .java .class
.java.class: $(JC) $(JFLAGS) $*.java
CLASSES = DiningPhilosophers.java \
  PhilosopherTable.java \
  PhilosopherThread.java
default: classes
classes: $(CLASSES:.java=.class)
clean: rm *.class
CS 370

Raspberry Pi

- Kevin Bruhwiler and Abhishek Yeluri
Topics

• Intro to Raspberry Pi
• Setting up a Raspberry Pi
• Term Project Requirements
• Term Project Expectations
• Helpful Links
Example Projects

- Retro Gaming Console
- RC Car
- Facial Recognition
- Weather Station
- Telescope
Why Raspberry Pi’s

- Small and Portable
- Cheap
- Well-Documented
- Versatile
- Support for many peripherals (thanks to Linux)

Third Best Selling Computer Brand in the World
Raspberry Pi Models

Raspberry Pi 3 Model B+
- 1.4GHz 64-bit quad-core processor
- dual-band wireless LAN
- Bluetooth 4.2/BLE
- faster Ethernet
- Power-over-Ethernet support (with separate PoE HAT)
- Raspberry Pi 4 - Even more memory
Raspberry Pi Setup

Can connect to monitor, keyboard, mouse

Usable as a normal desktop

Optionally use ssh instead of a monitor
Raspberry Pi Operating Systems

Expect most groups to use Raspbian (officially supported OS)

Other options are available - some OS’s for specific use cases
Raspberry Pi Operating Systems

GPIO Pin mapping

Used to send signals to external devices (usually)

Alternatively, communicate with other devices over WiFi or mesh network
GPIO Libraries

Python/C

- RPi.GPIO (Python)
  - RPi.GPIO code samples
- RPIO.GPIO (Python)
- wiringPi (Python/C)
- pigpio (Python/C/JavaScript)
- gpiozero (Python)
- bcm2835 (C)
Programming Languages

Basically any language will work (Python, C, Java, C++, Javascript, Ruby, Lisp, Rust, R, etc…)

Most projects done in Python or C
Term Project Requirements

Project must involve:

- A single board computer (Raspberry Pi)
  - With WiFi capability + operating system
- Communication with at least one other computer
  - Another board, desktop, assistant, etc.
- At least one sensing or interacting device
  - Heat sensor, motion detector, camera, motor, controller, etc...
Term Project TODO

- Team Composition and Proposal (done)
- Progress Report (due tomorrow)
- Final Report and Demo
  - Report: 1500 - 2500 words
  - Code
  - 10 - 15 Minute Demo
- Presentation
- Peer Review
Term Project Expectations

- Originality
  - Several groups with similar projects (temperature sensors, plant waterers, etc...)
  - Come up with a unique selling point
    - Find similar projects online, then do something different

- Thoroughness
  - Think about the evaluations you’re performing - design careful experiments and control for variables
  - Try to learn something you couldn’t have guessed
Helpful Links

- **Help Guides**
  - Setup instructions
  - SSH with Raspberry Pi’s
  - Help videos
  - FAQ’s
  - Embedded Linux wiki

- **Forums and Tutorials**
  - Raspberry Pi [forums](#) / [projects](#)
  - Hackaday Projects
  - Adafruit Learning Guides
  - Raspberry Pi subreddit
Thank you - Questions?
Acknowledgements

• Contributions of current and past instructors and TAs is acknowledged.