Synchronization in C

• Synchronization in C uses Mutex.
• The Mutex is provided in pthread.
• A mutex lock is a variable that can be “locked” by only one thread at a time. The thread with the lock is allowed to modify/read protected data. When it is done, it releases the lock.
  - Initialize a mutex lock variable:
    ‣ pthread_mutex_t mutex;
    ‣ pthread_mutex_init (&mutex, NULL);
  - Lock a mutex lock variable: Only one thread will be allowed to do this. The rest of the threads will be forced to wait until the lock is released. Threads will be chosen non-deterministically.
    ‣ pthread_mutex_lock(&mutex);
  - Unlock a mutex lock variable: The thread that has the lock should be the one to unlock it.
    ‣ pthread_mutex_unlock(&mutex);
  - Cleanup a mutex lock variable:
    ‣ pthread_mutex_destroy(&mutex);

Threads in Java

• There are two ways to create threads in Java: extends the Thread class or implement the Runnable interface.
• In both ways, we need to implement run method, which is equivalent to the worker function in C.
• Then, we need to create a thread object and call the start method.

```java
/**
 * HelloRunnable.java
 */
public class HelloRunnable implements Runnable {
    public void run () {
        System.out.println("Hello Runnable!");
    }
    public static void main (String[] args) {
        Thread t = new Thread(new HelloRunnable());
        t.start();
    }
}

/**
 * HelloThread.java
 */
public class HelloThread extends Thread {
    public void run () {
        System.out.println("Hello Thread!");
    }
    public static void main (String[] args) {
        Thread t = new HelloThread();
        t.start();
    }
}
```

• `t.join()` causes the current thread to pause execution until t’s thread terminates.
• `synchronized` method is like the worker function with mutex lock in C.

Logic Programming

• Logic programming emerged as a distinct paradigm in the 1970’s.
• What makes logic programming distinct and cool is that when you programming with logic program languages, you only need to say what you want, not how you want it done.
• A program written in a logic programming language consists of:
  - a set of facts about objects and their relationships
  - a set of rules about objects and their relationships
- a set of queries about objects and their relationship

Prolog

- Prolog and Logic

<table>
<thead>
<tr>
<th>Prolog</th>
<th>Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implication</td>
<td>B → A</td>
</tr>
<tr>
<td>Conjunction</td>
<td>A ∧ B</td>
</tr>
<tr>
<td>Disjunction</td>
<td>A ∨ B</td>
</tr>
<tr>
<td>Not</td>
<td>~A</td>
</tr>
</tbody>
</table>

- Fact
  - Consist of a particular item or a relation between items
  - Should begin with a lowercase letter and end with a full stop
  - Can consist of any letter or number combination, as well as the underscore character
  - Should avoid the characters +, -, *, /, or other mathematical operators
  - Example:

```
tomorrow_is_another_day.
saturday.
no_courses.
happy.
```

- Relation
  - more complicate facts
  - To define a relation, start with the relation name followed by a pair of parentheses with arguments in it.
    ```
    relation(<argument1>, <argument2>, … , <argumentN>).
    ```
  - Relation names must start with a lowercase letter.
  - Can have N arguments, N >= 0.
  - Example:

```
/**
 * Prerequisite of CS courses.
 */

prerequisite(cs231, cs151).
prerequisite(cs232, cs231).
prerequisite(cs333, cs231).
prerequisite(cs421, cs333).
prerequisite().
```

- Variable
  - A string of uppercase letters, lowercase letters, digits and underscore that start either with an uppercase letter or with an underscore.