

## **Computer Assignment #4**

**Due date:** 25/10/85

*Note:*

- All of assignments should be done by groups of one or two person! There is no extra bonus for one-person groups.
- Ask us if you have any question or grey area about the assignment.
- All programs must be written in C/C++ (under Linux operating system of course!)
- You **must** use *make* to build your programs.
- You can find the *Linux-Programming-Unleashed & Linux Advanced Programming* books in the <http://ce.sharif.edu/courses/85-86/1/ce424/index.php/section/resources/file/resources>
- Please send your CA source code to [ce424a@gmail.com](mailto:ce424a@gmail.com) before delivering it in the following format:
  - Subject: Computer Assignment2
  - Body: first and last name and student ID of each member.
  - Attachment: Source code: family1\_family2.zip

### **Part I: Bakery Algorithm**

Write a program implementing Bakery Algorithm for synchronization problem for more than 2 threads, presented in your text book. You must not use C/C++'s synchronization methods. You should also write a simple multithread program for testing your solution. (For example each thread increment common field in an infinite while and write it to screen.)

### **PART II: Bankers Algorithm**

In this exercise you should write a program which simulates bankers Algorithm. As you know this algorithms helps for deadlock avoidance in resource allocation. Once your program executed, you must load information about resources and processes from an input file. Here is the input file format:

#### **Input file format:**

```
ProcessesNum    ResourcesTypeNum
Resources:
resource1Num    resource2Num    resource3Num ...
Max:
Max matrix
Allocated:
Allocated matrix
```

**Sample File:**

```
3    3
Resource:
9    9    12
Max:
5    7    10
7    3    2
3    3    3
Allocated:
3    6    9
4    1    0
1    1    2
```

Note that the Allocation Matrix part is optional. And processes' ID starts at 1. (Here 1, 2, and 3).

**Commands:**

- **kill:** This command causes resources allocated for this process to be released and the process be terminated.

**Example:**

```
/>kill processID
Resources were released:
R1    R2    R3
3     6     9
/>
```

- **req:** using this command you can make a new resource request for a specific process for allocating or releasing.

**Example:**

```
/>req 2
request: 2 0 2
System has not enough resources for this request.
```

```
/>req 2
request: 2 0 1
This request could cause a deadlock.
```

```
/>req 3
request: 1 0 1
The request was accepted.
```

```
/>req 1
request: -2 -2 -2
The request was accepted.
```

Note the last request means that release 2 resources from each type of resources which allocated for process number 1.

- **safeseq:** shows a safe sequence for finishing all process.
- **showmaxneed:** shows the maximum needed Matrix.
- **showallocation:** shows the current allocation Matrix.
- **quit:** terminates the programs.