Computer Science 4500
Operating Systems
Fall 2018
Programming Assignment 2

Due on Thursday, October 25, 2018

Introduction

One technique for dealing with deadlock is called "detect and recover." In this scheme, some procedure is used to identify when a deadlock occurs, and then another procedure is used to deal with the blocked processes. One technique to identify a deadlock is to maintain a resource graph that identifies all processes, all resources, and the relationships between them (that is, which processes exclusively own which resources, and which processes are blocked waiting for which resources). In this assignment you will write a program that simulates the execution of multiple processes with respect to their resource allocation and resource release requests, and detect when (or if) deadlock occurs. If deadlock occurs, your program will then display a list of the processes and resources involved in the deadlock.

To simulate the behavior of the processes in the system, your program will use input that describes the sequence of actions in which each process requests, uses, and releases resources. Each of the *NP* processes will be identified by a unique positive integer in the range 1 to *NP*. The behavior of each process is described by a sequence of pairs of an action letter followed by a number *AN*, where *A* indicates the action the process is to take, and *N* depends on the particular action. The various actions permitted are as follows:

- If *A* is **L** (Lock), then the process (say process *P*) requests exclusive use of the resource identified by *N*. (The *NR* resources in the system are identified by integers in the range 1 to *NR*.) If the resource is available when process *P* requests its use, then process *P* is given exclusive use of the resource and is allowed to continue. If the resource is not available, process *P* is blocked and placed at the end of the first-in first-out queue of processes waiting on the resource. Process *P* remains on the queue until the resource becomes available and it (process *P*) is at the head of the queue of waiting processes. Thus the first process to block while requesting use of a resource is the process that is moved to the ready state when the resource again becomes available.
- If *A* is **U** (Unlock), process *P* relinquishes its exclusive use of the resource identified by *N*. This action will also move a single blocked process (say process *Q*) from the head of the queue of processes waiting on this resource to the end of the system's ready queue. Note that immediately after this waiting process (*Q*) moves to the end of the ready queue, the process that just released the resource (*P*) will be moved to the end of the ready queue, just after the process that was at the head of the waiting queue (*Q*).
- If *A* is **C** (Compute), then the process "computes" for *N* units of time, maintaining exclusive use of any resources it may have previously obtained but not yet released.

N will never be smaller than 1; that is, there will never be any vacuous "compute" actions in a process. Of course, the "computation" is really fictional, and will just cause the system time to increase by *N*.

A process terminates execution after it has successfully executed each of the actions specified for it in the input data. Note that we make the (unusual) assumption that if a process terminates still having exclusive use of one or more resources, then those resources are forever unavailable (at least until the next simulation)!

For example, suppose process 1 is described by the following sequence of pairs:

L1 L2 C10 U1 U2

The process will first request exclusive use of resource 1, then exclusive use of resource 2. Next it will "compute" for 10 units of time (maintaining exclusive use of resources 1 and 2), after which it will relinquish its use of resources 1 and 2 (in that order).

There will be no more than 10 processes in the system in any simulation, and we will assume there is just a single CPU. Each process will have its own sequence of integer pairs describing its execution, and there will be no more than 25 pairs for any process. There will be no more than 20 resources in any simulation.

Each successful request for the exclusive use or release of a resource by a process requires one time unit to complete, and a "compute" action requires the number of time units specified by *N* for that action. The example shown above will require a total of 14 time units to complete. Since all processes are assumed to be in the ready queue (in increasing process number order: 1, 2, …, *NP*) at the beginning of a simulation, and the first time unit is numbered 0, the process in the example above will terminate at time 13. (You might argue that it really terminates at time 14, but the interpretation in the instructor's solution is that it terminates at time 13.)

The execution of the sequences describing each process is done in a round-robin manner. One unit of time is given to each process, in turn, starting with process 1, then process 2, …, and then process *NP*. Execution then continues with process 1 again (assuming it didn't block or terminate during its previous "execution"). Of course, when a process completes execution, it is no longer given any units of execution time. If a process becomes blocked, it does not receive an units of execution time until it becomes unblocked. If a process *P* requests a resource *R*, but is not able to immediately obtain exclusive use of *R* (because another process *Q* is using it), then process *P* is blocked while waiting for the resource *R* to become available. When process *Q* releases resource *R*, the first process waiting for the resource (that is, the first process in the queue of processes waiting for resource *R*) is moved to the end of the queue of ready processes. Note that this does not guarantee that process *P* will successfully obtain the resource when it next runs, since another ready process (ahead of *P* in the ready queue) may run and request exclusive use of the resource before *P* is selected for execution.

Let's look at a complete example. Suppose we have three processes, with execution sequences as follows:

Process 1: L1 L2 C2 U1 U2
Process 2: L1 L2 C2 U1 U2
Process 3: L3 C5 U3 C2

Simulation 2 in the sample input and output (below) shows the sequence in which these process actions would occur, and the virtual time (starting at 0) when they would occur.

The set of processes in simulation 2 does not deadlock, but obviously other sequences can (see the samples for several such sequences). To detect deadlock, your program must build a resource graph (with nodes for processes and resources, and directed edges indicating resource ownership and pending resource requests), and test the graph for the existence of a cycle after each resource allocation request is made by a process. This test must be performed after each successful or unsuccessful allocation.

The Assignment

Write a program to carry out the simulation of multiple processes executing the sequences of resource requests and execution specified by the input data, checking after each resource allocation request for the existence of deadlock by attempting to find a cycle in the resource graph. If no deadlocks occur during execution, indicate the time when each process executes its last action and the total execution time required for each process. If a deadlock does occur, indicate that fact, the time when the deadlock was detected, and the identification of the processes and resources involved in the circular wait. The samples shown below illustrate the desired output format.

Turn in your assignments on time through Canvas. You can also turn your assignment in up to 24 hours late for a 20% penalty, or up to 48 hours late for 30% penalty. After 48 hours past the deadline, your assignment will not be graded and you will receive a zero.

The Trace Facility

The instructor’s solution to the assignment includes a trace facility that may help you understand the sequence of events in a simulation. The trace facility is turned on if there is any command line argument(s) in addition to the name of the file containing the executable program.

The trace output consists of two types of information. Each time an action is attempted, the trace output will display the simulation time, the process, the action, and the parameter (resource number or computation time) involved. As you can see from the sample output (below), the computation time parameter will continually be reduced as the process make progress toward completion.

The second type of information – shown in parentheses in the sample output – indicates when a resource is allocated, when an allocation was unsuccessful, when a resource was deallocated (released), when a process was unblocked (awakened) because a resource on which it was waiting was deallocated, and finally when a process terminated. This should all be obvious from examination of the sample output.

The Input Data

The input data will contain only integers and action letters (e.g., L1, L2) separated by whitespace (blanks, tabs, or end of line characters). The first two integers will specify *NP* and *NR* (The number of process/resource). For each process there then follows an integer indicating the number of integer pairs in its execution sequence, immediately followed by that many pairs of action letters/integers. This sequence repeats (giving data for additional simulations) until *NP* is 0 and *NR* is 0. Sample input and output appears below.

Details

You may use any algorithm you wish to detect deadlock, but that described in the textbook is recommended (that is, looking for a cycle in the resource graph after each allocation request).

All processes will have at least one execution item.

No invalid execution sequences will appear. In particular, no process will attempt to release a resource it does not already possess, and no process will attempt to obtain a resource it already possesses. (There is only one unit of each resource available.)

Sample Input and Output

The sample input and output shown below was produced from the instructor's solution for this assignment. The executable form of this solution is available in the file **`/home/peggy/csci4500/Fall2018/prog2`**, and the sample data shown here is available in the file **`/home/peggy/csci4500/Fall2018/prog2.input`**. The program reads the input data from the standard input and writes its results to the standard output. Any arbitrary command line argument to the program will cause it to display the trace information shown in the output (between the simulation number heading line, and the line indicating if there was successful completion of the processes or deadlock). No extraneous output (e.g. debugging information) should appear in your output, and tracing information should not appear in the output from your solution.

Sample Input

```
1 2
5   L1 L2 C10 U1 U2
3 3
5   L1 L2 C2 U1 U2
5   L1 L2 C2 U1 U2
4   L3 C5 U3 C2
2 2
5   L1 L2 C2 U1 U2
5   L2 L1 C2 U2 U1
3 3
6   L1 C3 L2 C3 U1 U2
6   L2 C3 L3 C3 U2 U3
6   L3 C3 L1 C3 U3 U1
7 6
7   L1 C5 L2 C5 U1 U2 C3
4   C3 L3 C5 U3
5   C3 L2 C2 U2 C3
6   L4 C6 L3 U4 C2 U3
6   L3 C5 L5 U3 U5 C2
7   L6 C5 L2 C2 U2 U6 C3
7   L5 C2 L4 C5 U5 C2 U4
4 4
9   L1 L2 L3 L4 C2 U1 U2 U3 U4
9   L2 L3 L4 L1 C2 U2 U1 U4 U3
9   L3 L4 L1 L2 C2 U1 U2 U3 U4
```

```
9   L4 L1 L2 L3 C2 U2 U1 U4 U3
0 0
```

Expected Output for the Sample Input (without trace)

```
Simulation 1
All processes successfully terminated.
Process 1: run time = 14, ended at 14

Simulation 2
All processes successfully terminated.
Process 1: run time = 6, ended at 12
Process 2: run time = 6, ended at 21
Process 3: run time = 9, ended at 19

Simulation 3
Deadlock detected at time 2 involving...
    Processes 1, 2
    Resources 2, 1

Simulation 4
Deadlock detected at time 12 involving...
    Processes 1, 2, 3
    Resources 2, 3, 1

Simulation 5
Deadlock detected at time 40 involving...
    Processes 4, 5, 7
    Resources 3, 5, 4

Simulation 6
Deadlock detected at time 4 involving...
    Processes 1, 2, 3, 4
    Resources 2, 3, 4, 1
```

Expected Output for the Sample Input (with trace enabled)

```
Simulation 1
0: process 1: L1
    (resource 1 allocated to process 1)
1: process 1: L2
    (resource 2 allocated to process 1)
2: process 1: C10
3: process 1: C9
4: process 1: C8
5: process 1: C7
6: process 1: C6
7: process 1: C5
8: process 1: C4
9: process 1: C3
10: process 1: C2
11: process 1: C1
12: process 1: U1
    (resource 1 released)
13: process 1: U2
    (resource 2 released)
    (process 1 terminated)
All processes successfully terminated.
Process 1: run time = 14, ended at 14

Simulation 2
0: process 1: L1
```

```
    (resource 1 allocated to process 1)
1: process 2: L1
    (resource 1 unavailable)
1: process 3: L3
    (resource 3 allocated to process 3)
2: process 1: L2
    (resource 2 allocated to process 1)
3: process 3: C5
4: process 1: C2
5: process 3: C4
6: process 1: C1
7: process 3: C3
8: process 1: U1
    (resource 1 released)
    (process 2 unblocked)
9: process 3: C2
10: process 2: L1
    (resource 1 allocated to process 2)
11: process 1: U2
    (resource 2 released)
    (process 1 terminated)
12: process 3: C1
13: process 2: L2
    (resource 2 allocated to process 2)
14: process 3: U3
    (resource 3 released)
15: process 2: C2
16: process 3: C2
17: process 2: C1
18: process 3: C1
    (process 3 terminated)
19: process 2: U1
    (resource 1 released)
20: process 2: U2
    (resource 2 released)
    (process 2 terminated)
All processes successfully terminated.
Process 1: run time = 6, ended at 12
Process 2: run time = 6, ended at 21
Process 3: run time = 9, ended at 19

Simulation 3
0: process 1: L1
    (resource 1 allocated to process 1)
1: process 2: L2
    (resource 2 allocated to process 2)
2: process 1: L2
    (resource 2 unavailable)
2: process 2: L1
    (resource 1 unavailable)
Deadlock detected at time 2 involving...
    Processes 1, 2
    Resources 2, 1

Simulation 4
0: process 1: L1
    (resource 1 allocated to process 1)
1: process 2: L2
    (resource 2 allocated to process 2)
2: process 3: L3
    (resource 3 allocated to process 3)
3: process 1: C3
4: process 2: C3
```

```
5: process 3: C3
6: process 1: C2
7: process 2: C2
8: process 3: C2
9: process 1: C1
10: process 2: C1
11: process 3: C1
12: process 1: L2
    (resource 2 unavailable)
12: process 2: L3
    (resource 3 unavailable)
12: process 3: L1
    (resource 1 unavailable)
Deadlock detected at time 12 involving...
    Processes 1, 2, 3
    Resources 2, 3, 1

Simulation 5
0: process 1: L1
    (resource 1 allocated to process 1)
1: process 2: C3
2: process 3: C3
3: process 4: L4
    (resource 4 allocated to process 4)
4: process 5: L3
    (resource 3 allocated to process 5)
5: process 6: L6
    (resource 6 allocated to process 6)
6: process 7: L5
    (resource 5 allocated to process 7)
7: process 1: C5
8: process 2: C2
9: process 3: C2
10: process 4: C6
11: process 5: C5
12: process 6: C5
13: process 7: C2
14: process 1: C4
15: process 2: C1
16: process 3: C1
17: process 4: C5
18: process 5: C4
19: process 6: C4
20: process 7: C1
21: process 1: C3
22: process 2: L3
    (resource 3 unavailable)
22: process 3: L2
    (resource 2 allocated to process 3)
23: process 4: C4
24: process 5: C3
25: process 6: C3
26: process 7: L4
    (resource 4 unavailable)
26: process 1: C2
27: process 3: C2
28: process 4: C3
29: process 5: C2
30: process 6: C2
31: process 1: C1
32: process 3: C1
33: process 4: C2
34: process 5: C1
```

```
35: process 6: C1
36: process 1: L2
    (resource 2 unavailable)
36: process 3: U2
    (resource 2 released)
    (process 1 unblocked)
37: process 4: C1
38: process 5: L5
    (resource 5 unavailable)
38: process 6: L2
    (resource 2 allocated to process 6)
39: process 1: L2
    (resource 2 unavailable)
39: process 3: C3
40: process 4: L3
    (resource 3 unavailable)
Deadlock detected at time 40 involving...
    Processes 4, 5, 7
    Resources 3, 5, 4

Simulation 6
0: process 1: L1
    (resource 1 allocated to process 1)
1: process 2: L2
    (resource 2 allocated to process 2)
2: process 3: L3
    (resource 3 allocated to process 3)
3: process 4: L4
    (resource 4 allocated to process 4)
4: process 1: L2
    (resource 2 unavailable)
4: process 2: L3
    (resource 3 unavailable)
4: process 3: L4
    (resource 4 unavailable)
4: process 4: L1
    (resource 1 unavailable)
Deadlock detected at time 4 involving...
    Processes 1, 2, 3, 4
    Resources 2, 3, 4, 1
```