Deadlock Simulator

Purpose

This is a user guide for the MOSS Deadlock Simulator. It explains how to use the simulator and describes the display and the various input files used by and output files produced by the simulator.

Introduction

The deadlock simulator illustrates multiple processes competing for one or more resources to investigate the nature and causes of deadlock conditions and how they can be avoided, detected, and resolved. The simulator includes a graphical user interface that allows the student to step through the "programs" being concurrently "executed" by each of the processes and see which processes are blocked by which resources.

Overview

The deadlock simulator performs the following steps as part of the simulation:

- It creates a specified number of simulated processes.
- For each process, it reads a command file of actions to be performed by that process. Each action is one of the following:
  - compute for a specified length of time
  - request an instance of a specified resource
  - free an instance of a specified resource
  - end or halt the process
- It creates a specified number of instances for each of several resources.
- It "executes" the simulation by considering the commands for each process, one at a time, and either allowing the process to execute, granting it an instance of a requested resource, blocking the process until a requested is available, or ending a process.
- As the execution proceeds, the display is updated to reflect the status of each process, and the number of available instances of each resource.
- At the end of each cycle of execution, the simulator writes a "log" message indicating the time since the beginning of the simulation, the number of available instances of each resource, the number of blocked processes, etc.
- The user may "step" through the simulation to view the action at each cycle, or may "run" the simulation to completion.
- When all processes are halted, the simulation stops.

Running the Simulator

First compile all java files with the **javac** compiler.

To run the program, enter the following command line.

```
$ java deadlock a 2 1 >a.log
```

In this example:

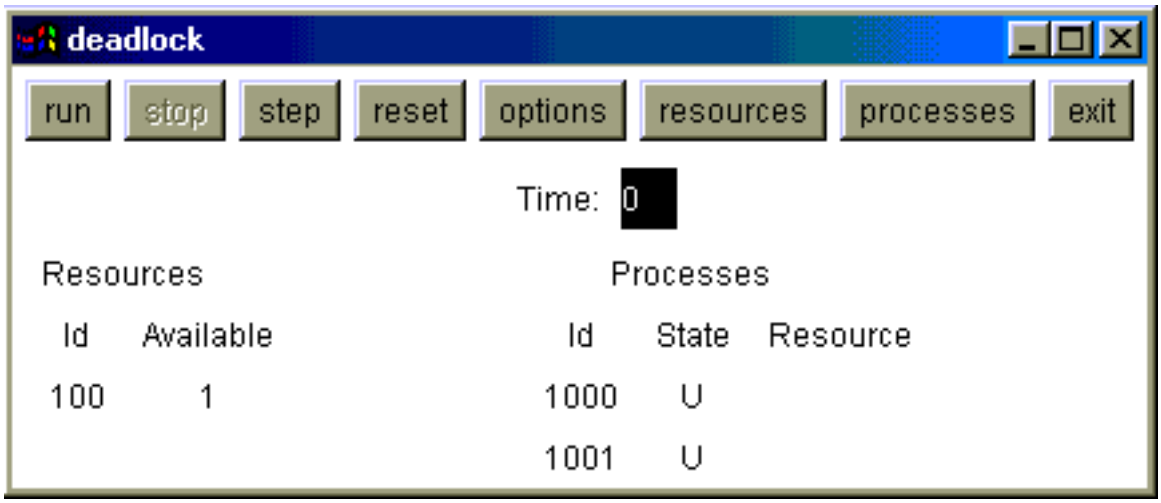
a is the prefix for the names of the files containing the commands, (the actual names of the files are "a0.dat", and "a1.dat"),

2 is the number of processes to be created,

1 is the number of instances to create for the first resource, and

a.log is the name of the output file.

The program will display a window allowing you to run the simulator. You will notice a row of command buttons across the top, and an informational display below. The left side of the information display lists the resources and the number of available instances for each, and the right side lists the processes and the current status for each.



Typically, you will use the `step` button to execute a cycle of the simulation and observe the effect on the resources and processes. When you're done, quit the simulation using the `exit` button.

The Command Line

The general form of the command line is

> $ java deadlock *file-name-prefix initial-number-of-processes initial-available-for-resource ...*

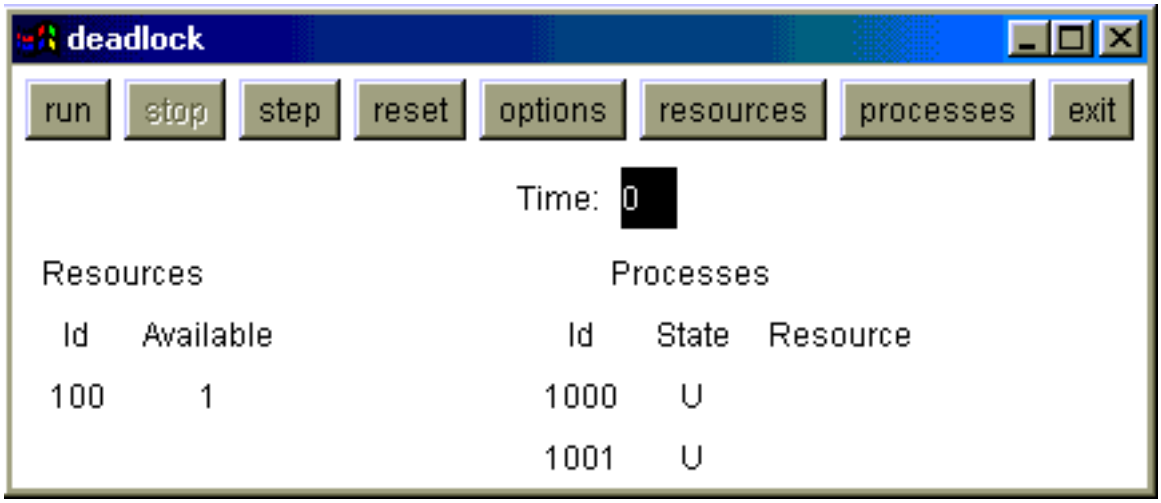
where

Parameter	Description
file-name-prefix	Specifies the name prefix for the process command files. The default command file names are generated from this prefix, followed by the number of the process, followed by ".dat" (e.g, "a0.dat", "a1.dat" if "a" is the prefix). The actual names of the files may be entered or modified in the Processes Dialog (see below).

initial-number-of-processes	Specifies the number of processes to create for the simulation. This should be a non-negative number, usually greater than one. This number may also be entered or modified using the Options Dialog (see below).
initial-available-for-resource...	Specifies the initial number of instances available for each resource. This should be a sequence of non-negative numbers. For example, "2 1 2 0" indicates that there are four resources, and there are initially two instances of resource 0, one instance of resource 1, two instances of resource 2, and zero instances of resource 3. The *number* of resources may also be entered or modified using the Options Dialog (see below). The initial number of instances *available* for each resource may be entered or modified using the Resources Dialog (see below).

The Control Panel

The main control panel for the simulator includes a row of command buttons, and an informational display.



The buttons:

Button	Description
`run`	runs the simulation to completion. Note that the simulation pauses and updates the screen between each step.
`stop`	stops the simulation if it is running. This button is only active if the run button has been pressed.
`step`	runs a single setup of the simulation and updates the display.
`reset`	initializes the simulator and starts from the initial values for each process and resource.
`options`	allows you to change various options for the simulator, including the number of resources and the number of processes.
`resources`	allows you to change the configuration for each resource, including the initial and current number of instances available for each resource.
`processes`	allows you to change the configuration for each process, including current state and the name of the command file for that process.

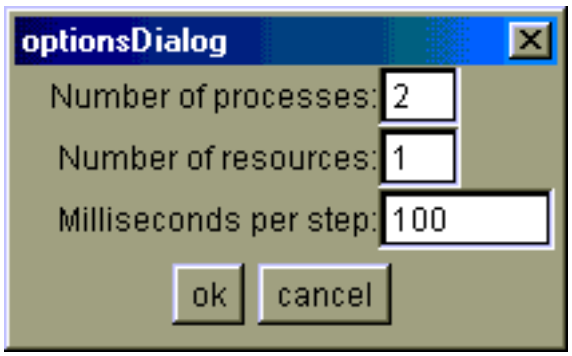
`exit`	exits the simulation.

The informational display:

Field	Description
`Time:`	number of "milliseconds" since the start of the simulation.
`Resource Id:`	A number which identifies the particular resource. Resources are numbered starting with zero.
`Resource Available:`	The number of instances available for the particular resource. This is a non-negative number.
`Process Id:`	A number which identifies the particular process. Processes are numbered starting with zero.
`Process State:`	The current state of the process. This may be U (unknown), C (computable), W (waiting), or H (halted). At the beginning of the simulation, all processes have U status. While a process is computable, it has a C status. If it requests a resource which is unavailable, it enters W status until the resource becomes available. When a process has completed all the commands in its command file or performs a halt command, it enters H status.
`Process Resource:`	The resource for which this process is waiting, if any. This field only has a value if the process is in W status.

The Options Dialog Box

The Options Dialog Box allows you to set general options for the simulator.



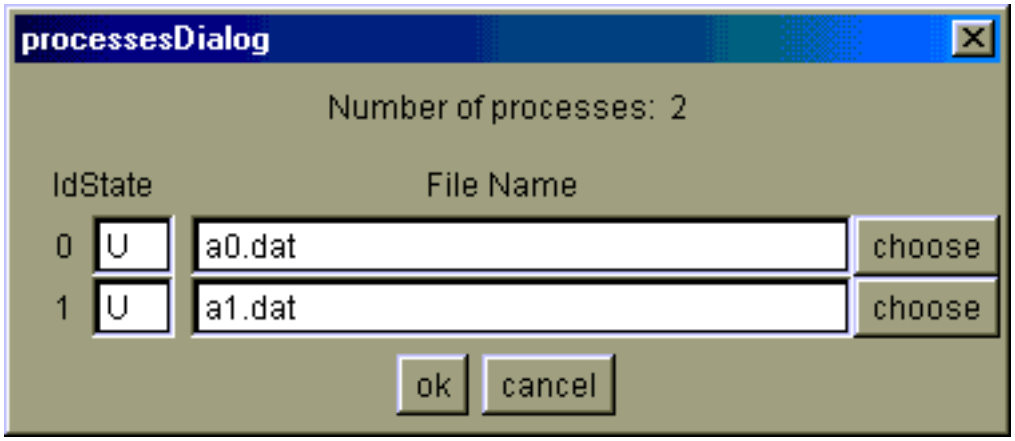
The options:

Field	Description
`Number of Processes:`	The number of processes to use in the simulation. This should be a non-negative number, usually at least two. Although the program does not enforce a limit, you may not be able to view more than about 10 processes on the informational display on your display screen. The initial value for this option is obtained from the second parameter on the command line, or zero, if not specified. Keep in mind that each process should have a process command file. To set properties for individual processes, use the Processes Dialog (see below).

Number of Resources:	The number of resources available in the simulation. This should be a non-negative number, usually at least one. Although the program does not enforce a limit, you may not be able to view more than about 10 resources on the informational display on your display screen. The initial value for this option is obtained from the number of initial instances for each resource specified on the command line (see above), or zero, if none are specified. This number should be one more than the largest resource number mentioned in any of the process command files for the simulation. To set properties for individual resources, use the Resources Dialog (see below).
Milliseconds per step:	The number of real-time milliseconds to pause between each cycle of the simulator in "run" mode. This is the pause between cycles when you hit the run button. The default value is 1000 milliseconds, or, one second.

The Processes Dialog Box

The Processes Dialog Box allows you to enter or modify properties for each process.

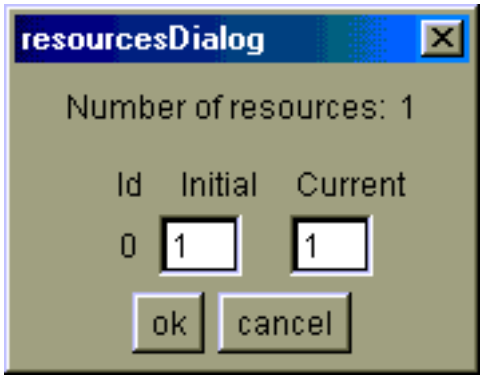


The process properties:

Field	Description
Number of Processes:	The number of processes in the simulation. To change this value, use the Options Dialog (see above).
Process Id	The id number for the process. These numbers are used to identify each process and are assigned by the simulator, starting with zero. These numbers cannot be changed.
Process File Name	The name of the file from which process commands are read. This may be any valid filename. For convenience, there is a choose button which allows you to browse the file system to choose the file. By default, the name is the prefix string, followed by the process number, followed by ".dat".

The Resources Dialog Box

The Resources Dialog Box allows you to enter and modify properties for each resource.



The resource properties:

Field	Description
`Number of resources:`	The number of resources available in the simulation. To change this value, use the Options Dialog (see above).
`Resource Id`	The id number assigned to the resource. This number is used to identify the resource and is assigned by the simulator and cannot be changed. This is the number which appears in the R (request resource) and F (free resource) commands in the process command files.
`Resource Initial`	The initial number of available instances of the resource. This number is used when the simulator starts or is reset.
`Resource Current`	The current number of available instances of the resource. This number may be changed during the simulation to see the effect it may have on processes waiting for the resource.

The Process Command Files

The process command files for the simulator specifies a sequence operations to be performed by the process or processes which use the file. There are four operations defined C (compute), R (request resource), F (free resource) and H (halt).

Operation	Description
`C` *msec*	Compute for the specified number of milliseconds (cycles).
`R` *resource-id*	Request an instance of the specified resource. If none are available, block the process until the resource becomes available. The resource id should be a non-negative number *less than* the total number of resources available.
`F` *resource-id*	Free an instance of the specified resource. This is usually a resource that was previously requested by the process. The resource id should be a non-negative number *less than* the total number of resources available.
`H`	Halt the process. This is usually the last operation in the file. Any commands which follow it in the file are ignored. Any file that does not end with this operation is implicitly halted.

Sample Process Command Files

The "a0.dat" input file looks like this:

```
/*
a0.dat

The "a" collection of process data files is meant to simulate
two processes competing for a single resource.  If you run
the simulator with one resource available, one of the processes
will block until the other is done using the resource.
*/
C 10    // compute for 10 milliseconds
R 0     // request resource 0
C 10    // compute for 10 milliseconds
F 0     // free resource 0
H       // halt process
```

Note that the "a1.dat" file is identical. In other words, both files request the same resources at approximately the same time.

The Output File

The output file contains a log of the simulation since the simulation started.

The output file contains one line per cycle executed. The format of each line is:

time = t available = r_0 r_1 ... r_n blocked = n

where

t
: is the number of milliseconds since the start of the simulation,

r_i
: is the number of available instances of each resource, and

n
: is the number of blocked processes.

Sample Output

The output file "a.log" looks something like this:

```
time = 0 available = 1 blocked = 0
time = 1 available = 1 blocked = 0
time = 2 available = 1 blocked = 0
time = 3 available = 1 blocked = 0
time = 4 available = 1 blocked = 0
time = 5 available = 1 blocked = 0
time = 6 available = 1 blocked = 0
time = 7 available = 1 blocked = 0
time = 8 available = 1 blocked = 0
time = 9 available = 1 blocked = 0
time = 10 available = 0 blocked = 1
time = 11 available = 0 blocked = 1
time = 12 available = 0 blocked = 1
time = 13 available = 0 blocked = 1
time = 14 available = 0 blocked = 1
time = 15 available = 0 blocked = 1
time = 16 available = 0 blocked = 1
time = 17 available = 0 blocked = 1
```

```
time = 18 available = 0 blocked = 1
time = 19 available = 0 blocked = 1
time = 20 available = 0 blocked = 0
time = 21 available = 0 blocked = 0
time = 22 available = 0 blocked = 0
time = 23 available = 0 blocked = 0
time = 24 available = 0 blocked = 0
time = 25 available = 0 blocked = 0
time = 26 available = 0 blocked = 0
time = 27 available = 0 blocked = 0
time = 28 available = 0 blocked = 0
time = 29 available = 0 blocked = 0
time = 30 available = 1 blocked = 0
```

In this example, the simulation runs for a total of 30 "milliseconds" and then halts. During the simulation, all processes are computable for 10 milliseconds. During the next 10 milliseconds, the one instance of the resource is allocated to one process, while the other process is blocked. During the final 10 milliseconds, the first process frees the resource, but it is immediately allocated by the second process, which then continues to compute, unblocked, to the end of the simulation.

Suggested Exercises

1. Try running the deadlock simulator using the following command:

   ```
   java deadlock a 2 2
   ```

   Explain why a deadlock does *not* occur.

2. There are two additional process command files ("b0.dat" and "b1.dat") in the distribution. Run the deadlock simulator with this command:

   ```
   java deadlock b 2 1 1
   ```

What happens?

Now try this.

```
java deadlock b 2 1 2
```

Why does the first command result in a deadlock but the second does not? Explain your answer in terms of what is going on in the process command files, b0.dat and b1.dat.