Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

Sistemas de Operação

(Academic year of 2017/18)

assignment #2 December, 2017

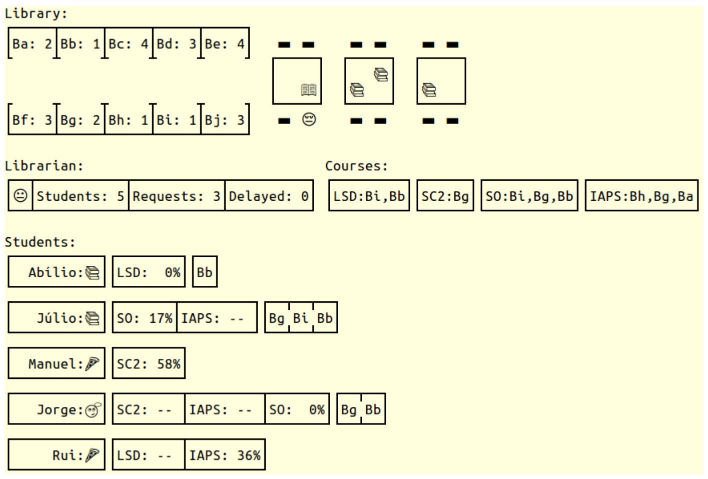
University library simulation

A group of students has entered the University keen to study a group of course units. The University is composed by a library, possessing a list of (randomly defined) books; a set of (randomly defined) course units, which (randomly) prescribe a list of required bibliography.

The library is composed of a bookshelf containing unrequisited books, and a group a tables (which a given number of seats), in which students may study.

Students have a very austere and rigorous life in which, in addition to studying, they also can eat, sleep and have some (little) fun. All the study is done inside the library, and requires the requisition of a random lists of books which are part of the actual course unit bibliography. The whole course is done by doing one course unit at a time (until it is completed). After book requisition, the student sits in an empty (and clean) library seat studying those books (for a random amount of time). Being a little bit sloppy, after this study is completed, the student leaves the books in the library tables (expecting that someone else – a librarian – collects and stores the books into their place).

A predefined given algorithm is used to estimate the contribution of the study for the completion of a course unit.



```
Library:
Ba: 2 Bb: 1 Bc: 4 Bd: 3 Be: 4

Bf: 3 Bg: 2 Bh: 1 Bi: 1 Bj: 3

Librarian:                                 Courses:
Students: 5 Requests: 3 Delayed: 0         LSD:Bi,Bb  SC2:Bg  SO:Bi,Bg,Bb  IAPS:Bh,Bg,Ba

Students:
  Abílio:  LSD:  0%   Bb
   Júlio:  SO: 17% IAPS: --   Bg Bi Bb
  Manuel:  SC2: 58%
   Jorge:  SC2: -- IAPS: -- SO:  0%   Bg Bb
     Rui:  LSD: -- IAPS: 36%
```

Another active entity (i.e to which an asynchronous autonomous program expressing its life should be attached), is the librarian. The librarian, passes its working days accepting and handling requests. He accepts: book requisition, book recollection (i.e. clean tables and return the books to their shelves), termination, and student enrollment and disenrollment. The librarian has also some life beyond its work, so he also eats, sleeps and have some fun. He has a rigid day schedule, with two working periods (morning and afternoon). In each working period he only accepts a fixed amount of requests, after which he collects all books abandoned in the library tables and puts them in their place on the bookshelves. To solve liveness problems, the librarian may be required to collect books from the library table at other times than the one prescribed. Sometimes, he librarian is unable to comply with a request (for instance, when books are at the time unavailable for requisition). When this happens, the request is delayed until it can me met (the student should wait until the request is complied).

Student and librarian active entities are delayed a random amount of time units in their activities (bounded by the simulation parameters). Time units are converted to absolute time by a provided timer module.

Source code is given that implements many (sequential) aspects of the simulation (log module, random functions, global parameters, program argument handling, etc.), and should be carefully analyzed and understood.

Log entity

Note that an active entity should be created and attached to the log module. To ensure a safe implementation of the visualization of the simulation dynamics, there should be only one thread/process in charge to it[1].

Support code

To ease the development of this assignment, a fair amount of working source code is provided (look into it carefully). In particular, wrapping modules for the usage of all concurrent libraries (POSIX threads, System V IPC, and POSIX semaphores) are provided. These modules have functions with similar prototypes than those in the library, but with automatic error handling either by throwing exceptions (with `errno` value), or aborting the process with proper error messages. Hence, there will be no need for error checking when using these modules.

Assignment

Devise a possible solution and write two simulations for this problem, one using POSIX threads, mutexes and condition variables, and the other using UNIX processes, shared memory and semaphores (System V or POSIX).

The solutions should be implemented in C++ and be run in Linux.

A sequential log module is given to observe the dynamics of the simulation in a clear and precise way. This module has two modes of operation – line and window – that allows alternative observing of the simulation (line mode, generates a list of lines describing the state of the simulation parts; and window mode is exemplified in the presented figure).

[1]Graphical handling in languages such as Java's use the same approach to the problem.

Simulation parameters

The simulation should be configurable with the following parameters (minimum/maximum parameters define the interval for a random choice in each case):

name	option	meaning
library:		
NUM_BOOKS	-b <N> --num-books <N>	number of different books in library
BOOK_COPIES	-C <MIN>,<MAX> --book-copies <MIN>,<MAX>	minimum and maximum number book copies
NUM_SEATS	-S <NTABLES>,<SEATS_PER_TABLE> --num-seats <NTABLES>,<SEATS_PER_TABLE>	number of tables and seats per table (must be an even number)
course units:		
NUM_COURSE_UNITS	-c <N> --courses <N>	number of course units
MAX_BOOKS_PER_COURSE	-m <N> --max-books-per-course <N>	maximum number of books per course
BOOK_STUDY_TIME_UNITS	-1 <MIN>,<MAX> --book-study-time-units <MIN>,<MAX>	min./max. required study time units for book within the course
students:		
NUM_STUDENTS	-s <N> --num-students <N>	number of students
STUDY_TIME_UNITS	-2 <MIN>,<MAX> --study-time-units <MIN>,<MAX>	min./max. study time units in each library visit
librarian:		
REQUESTS_PER_PERIOD	-r <MIN>,<MAX> --requests-per-period <MIN>,<MAX>	min./max. attend requests in each period
HANDLE_REQUEST_TIME_UNITS	-3 <MIN>,<MAX> --handle-request-time-units <MIN>,<MAX>	min./max. time units to handle a request
students/librarian:		
EAT_TIME_UNITS	-4 <MIN>,<MAX> --eat-time-units <MIN>,<MAX>	min./max. eat time units
SLEEP_TIME_UNITS	-5 <MIN>,<MAX> --sleep-time-units <MIN>,<MAX>	min./max. sleep time units
FUN_TIME_UNITS	-6 <MIN>,<MAX> --fun-study-time-units <MIN>,<MAX>	min./max. fun time units

Grading criteria

For grading the following features will be considered (either as a penalization or as bonus):

- Safety: no race condition can occur, and the state of the simulation should be consistent as a whole (representing valid states).
- Liveness: the simulation should always proceed.
- Maximum asynchronous concurrent activity (for example, a student should not interfere with the life of other student if no shared resources – such as books or seats – are used).
- Minimization of the activity of the librarian (i.e. besides its life normal schedule, the librarian should only be required to do things if necessary).

The simulation should be tested with extreme values of resources (though enough to ensure progress), to gain confidence on its correctness.