Operating Systems 2 - Assignment Semaphores - 2018

Learning objective

In this assignment you learn how to write a concurrent application in C by means of semaphores.

Assignment

Requirement: the following problem has to be solved by means of Semaphores, without further use of other synchronisation possibilities.

Case: Santa Claus and his elves

It's Christmas time and Santa Claus is busy preparing all the presents to give out on Christmas day. Of course he has help in the form of dozens of elves in their customary red and green suits. In the days before Christmas all the children will make their wish list, which have to be collected by all the **collecting-elves**. Santa has also a number of **work-elves**, who buy, sort and assemble all the presents based on the wish lists. All elves have a nice green suit with red details like shoes and gloves.

To let everything run smoothly, Santa has drawn up some guidelines:

- The **collecting-elves** are sent out to collect the wish list, this takes some time. When they return they report to Santa Claus.
- The **work-elves** buy, sort and assemble all the presents based on the wish lists and stop working regularly to report to Santa Claus to discuss the work progress or participate in a meeting.
- Santa sometimes starts a **collecting-meeting** to hand over the wish lists from the collecting-elves to the work-elves. In this meeting the lists and presents to buy, sort and assemble are discussed. This important meeting can only start if one *work-elf* and at least *three collecting-elves* are *available* (that means, they have reported to Santa Claus). All available collecting-elves will participate in the meeting.
- If *two work-elves* are *available*, but a collecting-meeting cannot start (yet), because not enough *collecting-elves* are available, a **work-meeting** starts between Santa and the two work-elves to discuss matters, after which they resume their work (there's always plenty to do)
- If enough elves are present that both meetings could start, the collecting-meeting always has priority, and all *work-elves* not needed anymore resume their work.
- Santa also needs his rest, so after each meeting he will take short nap, after which he will see if any elves have reported to him so a meeting can possibly start.
- All elves who report to Santa always wait till the next meeting or until they are dismissed.

Task

Implement a simulation of the above system in C, you may use your own laptop and C IDE but it should be possible to port it to, and run on, a Raspberry Pi. Use only semaphores for synchronisation and the correct functioning

Hint: Thread synchronisation by means of Semaphores always uses shared memory, to share information between threads. So find out first which information you probably have to keep track of to simulate the case above.
Develop your solution by determining:

- which activities (threads) there are in your system,
- what an activity actually does,
- where activities have to wait for each other (synchronisation)
- which data you need to let the system run correctly

Additional features (not obligatory to pass)

For those of you who want a bigger challenge (and consequently a possible higher grade), improve your solution by adding some or all additional guidelines below:

1. Some work-elves have more seniority and dress themselves in a **red** suit (with corresponding green details of course) so they can be more easily spotted when their expertise is needed. A collecting meeting now must always consist of a red work-elf and, as before, three collecting-elves.
2. If a work-elf reports to Santa Claus while a meeting is in progress (you cannot join halfway!), or Santa is taking a nap, he[1] will wait a short time to see if a meeting can start, if not, he resumes work again, and after some working time reports back again.
3. Consider that it is very busy and Santa doesn't have time to take a nap after each meeting, that is after one meeting he almost immediatly checks if another meeting can start, if not, he takes his nap. But make sure he will take his naps at least after every four meetings.

To submit at blackboard:

1. A report in which you describe the approach of your problem. Clearly specify the threads you need, the life cycle of the threads and the way in which the threads communicate/synchronise with each other.
   Also specify the way how you tested your program, so that you can make it plausible in a convincing way that your solution is a correct implementation of the case (this means of course also that you should test your program in a convincing way).
2. An ***exported*** zip file with the name Surname1Surname2Semaphore (in which the surnames are of course the surnames of the group members).
3. Deadline: see blackboard.

[1] Could be also a he