Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 2^{nd} semester - 2020/21

Project #2
Semaphores under Unix/Linux
Due: April 16, 2021

Instructor: Dr. Hanna Bullata

Game using Semaphores

We would like to create a multi-processing application that employs semaphores *only*. A parent process will create 6 children processes named $P_0 \ldots P_5$ located initially at locations `A` ... `I` respectively as shown in Figure 1.

The behavior of the whole system is as follows:

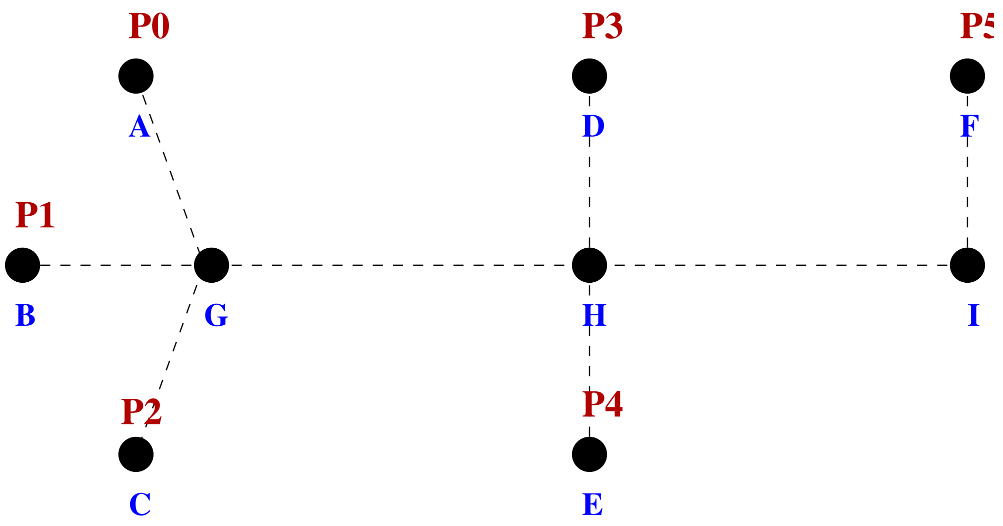

Figure 1: The whole system

1. Upon an order from the parent, processes P_0, P_1 and P_2 start moving towards location `G`.
2. Once P_0, P_1 and P_2 reach location `G`, they *walk together* to location `H`.
3. Once P_0, P_1 and P_2 get to location `H`, processes P_3 and P_4 start moving towards location `H` and at the same time process P_5 starts moving towards location `I`.
4. When *either* P_3 or P_4 gets to location `H`, P_0 starts moving back to location `G`. When both P_3 and P_4 get to location `H`, $P_1 \ldots P_4$ start moving to location `I`.
5. When processes $P_1 \ldots P_5$ reach location `I`, process P_0 starts moving back to location `A`.
6. Once process P_0 gets to location `A`, P_1, P_2, P_3 and P_4 start moving back to location `H` while process P_5 starts moving back to location `F`.
7. Once process P_5 gets to location `F`, process P_3 starts moving back to location `D` while process P_4 starts moving back to location `E`.
8. When either P_3 gets to location `D` or process P_4 gets to location `E`, processes P_1 and P_2 start moving to location `G`.

9. When both process P_3 is back to location `D` *and* process P_4 is back to location `E`, process P_1 starts moving back to location `B` and process P_2 starts moving back to location `C`.

10. Once process P_1 is back to location `B` and process P_2 is back to location `C` (meaning we're back to the original configuration where each process is back to its home location), go back to step **1** above unless the above behavior has been done for 10 times.

11. The parent process must kill all its children, free the semaphores facility it is using and exit.

What you should do

- Write the code for the above system.
- Compile and test your program.
- Check that your program is bug-free. Use the `gdb` debugger in case you are having problems during writing the code (and most probably you will :-). In such a case, compile your code using the `-g` option of the `gcc`.
- Send the zipped folder that contains your source code and your executable(s) before the deadline. If the deadline is reached and you are still having problems with your code, just send it as is!