Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 2^{nd} semester - 2021/22

Project #1
Signals & fifos under Unix/Linux
Due: April 12, 2022

Instructor: Dr. Hanna Bullata

Primitive War Simulation

We would like to create a multi-processing application that simulates a primitive war between 2 armies using signals and fifos. To simplify things, we'll assume each army is composed of 5 soldiers where each soldier is assigned a static location on the (x, y) plane (assume the battlefield size is 50×50). A parent process is responsible to create the soldiers of the 2 armies and assign them that static location.

The behavior of the whole system should be as follows:

1. Upon creation, the soldiers on both sides will recharge their rifles with bullets. Assume that soldiers are allowed to fire once before the need to recharge their guns. The recharging operation consumes some time. Assume each soldier needs a random amount of time to recharge (the recharge time should be bound between a minimum and a maximum value).
2. When shooting, a soldier will choose to shoot at the enemy soldier that is closest to it. The probability to hurt the enemy soldiers should be random in nature and inversely proportional with the distance (e.g. the bigger the distance the less the likelihood to hurt that soldier).
3. When fired upon, soldiers can be hit in the head, or the neck, or the chest, or the abdomen or the hands or the legs. In order to die, a soldier must be hit by at least 2 bullets in the head, or 3 bullets in the neck or 3 bullets in the chest or 3 bullets in the abdomen or 5 bullets in the hands or legs. When hit by bullets, soldiers have to keep the count of bullets and the location of their injuries in order to decide if they should die (exit the simulation) or stay and continue fighting.
4. When injured, soldiers lose their focus and the ability to inflict damage on enemy soldiers. Assume that each time a soldier is hit by more bullets, that fact deteriorates.
5. Whenever a soldier dies, the enemy soldiers should stop targeting it (because it is a waste). Thus, enemy soldiers must be able to know if an enemy soldier is alive or not.
6. Once all the soldiers of an army die, the parent declares the enemy army as the winner of the round.
7. The parent restarts a new round once step **6.** above is complete by going to step **1.**.
8. The parent process declares the winner of the war once an army has won for a total of 5 rounds.

What you should do

- Write the code for the parent and soldier processes.
- Compile and test your program.

- Check that your program is bug-free and runs as expected. 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!