laSalle
UNIVERSITAT RAMON LLULL

Operating Systems 2023-2024

Session 8: Semaphores I 24/11/22

Goals

In previous sessions, we worked on the concept of threads and forks for parallelism and concurrency of tasks. The problem with working with shared resources and multi-process systems is that two processes try to access the same resource simultaneously, thus generating unpredictable or random results. On the other hand, sometimes the correct sequencing of tasks (synchronization) must also be guaranteed. The goal of this session is to introduce Operating Systems students to one of the most common synchronization and mutual exclusion mechanisms: **semaphores**. With the development of this session, it is intended that the student clarify the theoretical concepts of the use of semaphores and solve different problems.

Motivation

More specifically, with this session, the student will exercise:

- Semaphore creation (*SEM_constructor, SEM_init*)
- Control and destruction of semaphores (*SEM_destructor*)
- Operations on semaphores (*SEM_signal, SEM_wait*)
- Our custom library semaphore.h
- Use of the mutex library to use semaphores with threads.

Previous documentation

To complete this session, it is recommended to read the following references:

SALVADOR, J. (2014). *Programació en UNIX per a pràctiques de Sistemes Operatius*.

STEVENS, W. R. & RAGO S.A. (2008). Advanced Programming in the UNIX Environment, 2nd edition.

TANENBAUM, A. S. (2009). Modern Operating Systems, 3rd Edition.

Underwater explorers

In the context of underwater exploration, we are faced with the challenge of investigating unknown sea depths. For this purpose, we have decided to run a simulation in which different teams of underwater explorers compete to be the first to discover treasures hidden in the seabed.

This underwater competition will take place in an area with different depth levels, from the shallowest to the deepest. The explorers must follow a set of established rules:

- Each team of explorers must descend in order from the shallowest to the deepest level.
- Only one team may be present at each sea level at a time.
- When a team arrives at a level that has not been previously explored, they must report their discovery and set up an exploration camp. This takes 2 seconds.
- If the level has already been explored, the explorers will conduct a brief 1 second inspection before proceeding.
- The first team to reach the deepest level will win the underwater competition. The simulation will conclude when all teams have returned to the surface.

The depth levels to be explored are as follows:

Surface -> Level 1 -> Level 2 -> Level 3 -> Level 4 -> Maximum Depth.

The names of the participating teams are in a text file called "scientis.txt" in the following format:

Team Name 1
Team Name 2
...

Execution example

```
a.navarrosoto@montserrat:>./S8

Team xavi discovered Surface!
Team aaron reached Surface
Team xavi discovered Level 1!
Team male reached Surface
Team aaron reached Level 1
Team xavi discovered Level 2!
Team male reached Level 1
Team aaron reached Level 2
Team xavi discovered Level 3!
Team male reached Level 2
Team aaron reached Level 3
Team xavi discovered Level 4!
Team male reached Level 3
Team aaron reached Level 4
Team male reached Level 4
```

WARNING: screen messages may be displayed in a different order or misformatted due to screen sharing between main program and threads.

Considerations

- It can be considered that the user will perform the actions correctly. That is, he/she will always make the participant dance before proceeding to kick him/she out and will always make all the participants dance and kick them out before proceeding to send the 'F' to terminate the program.
- The use of global variables is not allowed, except for those that you think are essential.
- All participants must be created using threads.
- Recursively programming is not allowed. All software has to be iterative.
- You don't need to control Ctrl+C.
- Check not to leave shared resources (semaphores, zombie processes, etc.) in the system. Remember to use ipcs, ipcrm, and the provided scripts. And freeing all the dynamic memory.
- It must be ensured that at the end of the execution all the processes have ended correctly without leaving zombie or orphaned processes.
- All inputs and outputs must be done using file descriptors. **The use of printf, scanf, FILE*, getchar or similar is NOT allowed.**
- **The use of the functions "system", "popen", or from the same family is NOT allowed**.
- **You must compile using -Wall and -Wextra flags.**
- **Any deliverable containing warnings or errors will be directly discarded.**
- At the start of the .c you must include a comment with your logins, names, and surnames.
- **For submitting the session, you must hand over a file "S8.c", and deliver it through eStudy platform.**