Operating Systems 2024-2025

Session 10: Semaphores II 11/12/24

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 and mutex for threads**. 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 own library semaphore.h / semaphore_v2.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.

Salle Cars, Ltd

The eco-vehicle company “Salle, Ltd” is about to launch its new model: The Challenger. For that, the company has opened a new factory in Barcelona. They have asked us to implement a software to manage the process of assembling a car, ensuring that each stage is completed correctly before moving on to the next. The objective is to create a program that controls this process in a factory that has five assembly stations, each dedicated to a different part of the car.

The stations are as follows:

1. **Body Station**: Where the body of the car is assembled.
2. **Engine Station**: In charge of installing the engine.
3. **Wheel Station**: Where the wheels are assembled.
4. **Paint Station**: To apply the coat of paint.
5. **Quality Control Station**: To verify that the car has passed correctly through the four previous stations.

Due to budget constraints, the factory can only assemble two cars at a time. If there are more than two cars in production, the additional cars must wait until one of the cars in process has finished.

The process follows a specific order: first the car goes through the Body Station, then the Engine Station, the Wheel Station, the Paint Station and finally arrives at the Quality Control Station.

The program will follow the following steps:

1. **Program Initialization**
   The program must be started by receiving as parameter the number of cars to be manufactured. If this parameter is not given or if the number is less than 1, the following error message should be displayed:
   Usage: ./S10.exe <number of cars>.

2. **Process Creation and Management**
   Each station must run in a separate thread per station. However, the stations must be synchronized: a station cannot start its task until the previous station has completed its work on that car. This ensures that the process follows the correct assembly order.

3. **Simulation of the work at each station**
   Each station simulates the assigned task with a random waiting time between 1 and 5 seconds. During this time, the station will be "working" on the car.
   When a station starts its task, the program should display the following message:

   ```
   [Station name] of car [car number] starting
   ```

   When the station task is finished, it must be printed:

   ```
   [Station name] of car [car number] assembled
   ```

4. **Quality control**
   Once the car has passed through the Body, Engine, Wheels and Paint stations, the Quality Control is carried out.

   At the Quality Control Station, random numbers between 0 and 9 are generated for each of the four phases (Body, Engine, Wheels, Paint). If the number is greater than 5, the station passes QC; otherwise, wait 1 second and repeat the process until the station passes QC.

   For each station that attempts to pass the check, the following message should be displayed depending on whether it passes or fails.

   ```
   Quality control of car [car number] at [station name]
   [failed/passed]
   ```

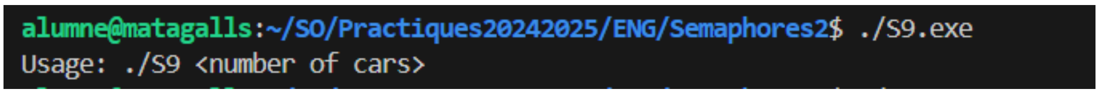
   After all stations have passed quality control, this should be displayed:

   ```
   Quality control of car [car number] passed
   ```

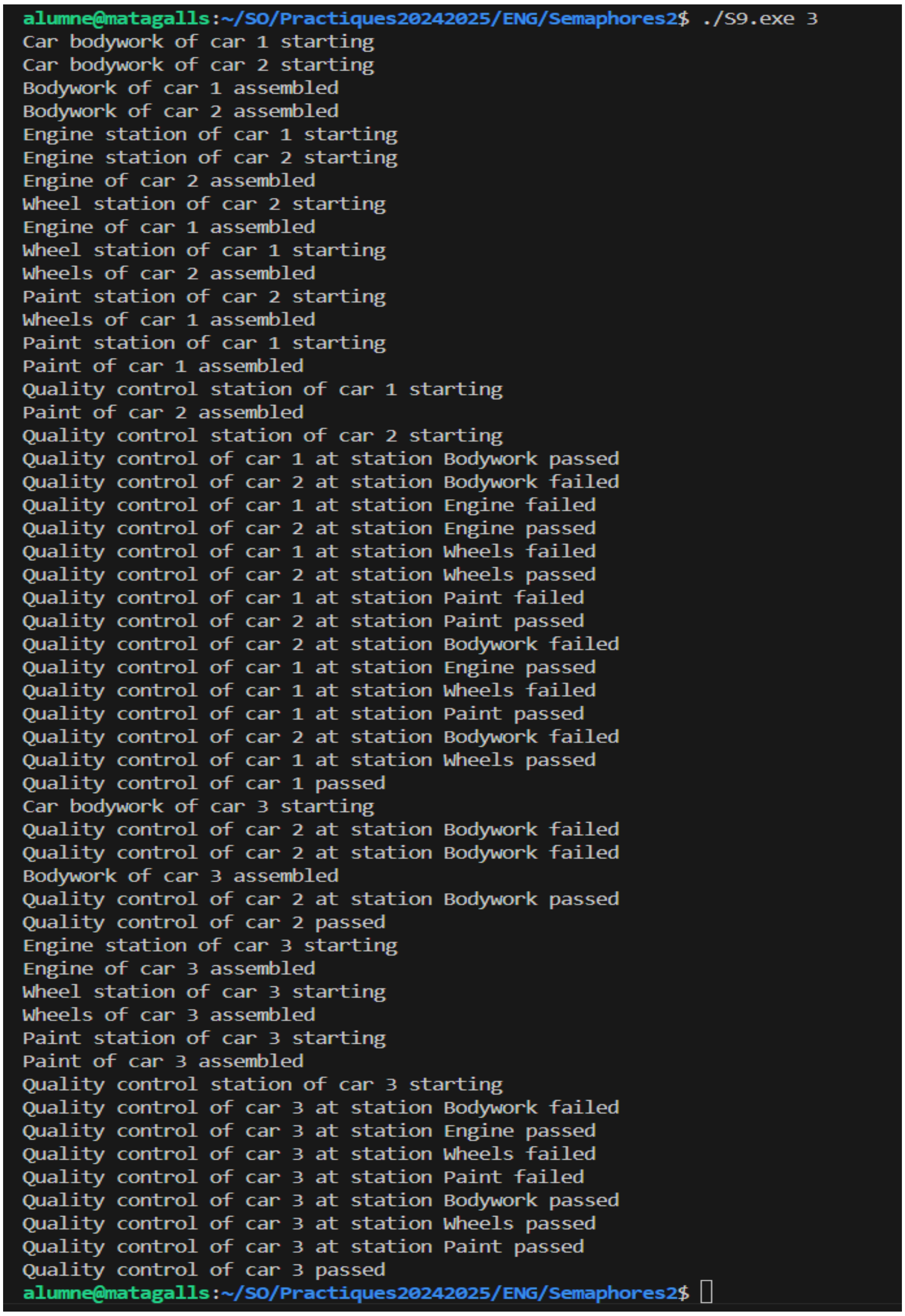
5. **Program end**

   After all cars have completed the assembly tasks and passed the quality checks, the program should automatically terminate, ensuring that all process threads have finished correctly.

Execution Examples



```
alumne@matagalls:~/SO/Practiques20242025/ENG/Semaphores2$ ./S9.exe
Usage: ./S9 <number of cars>
```



```
alumne@matagalls:~/SO/Practiques20242025/ENG/Semaphores2$ ./S9.exe 3
Car bodywork of car 1 starting
Car bodywork of car 2 starting
Bodywork of car 1 assembled
Bodywork of car 2 assembled
Engine station of car 1 starting
Engine station of car 2 starting
Engine of car 2 assembled
Wheel station of car 2 starting
Engine of car 1 assembled
Wheel station of car 1 starting
Wheels of car 2 assembled
Paint station of car 2 starting
Wheels of car 1 assembled
Paint station of car 1 starting
Paint of car 1 assembled
Quality control station of car 1 starting
Paint of car 2 assembled
Quality control station of car 2 starting
Quality control of car 1 at station Bodywork passed
Quality control of car 2 at station Bodywork failed
Quality control of car 1 at station Engine failed
Quality control of car 2 at station Engine passed
Quality control of car 1 at station Wheels failed
Quality control of car 2 at station Wheels passed
Quality control of car 1 at station Paint failed
Quality control of car 2 at station Paint passed
Quality control of car 2 at station Bodywork failed
Quality control of car 1 at station Engine passed
Quality control of car 1 at station Wheels failed
Quality control of car 1 at station Paint passed
Quality control of car 2 at station Bodywork failed
Quality control of car 1 at station Wheels passed
Quality control of car 1 passed
Car bodywork of car 3 starting
Quality control of car 2 at station Bodywork failed
Quality control of car 2 at station Bodywork failed
Bodywork of car 3 assembled
Quality control of car 2 at station Bodywork passed
Quality control of car 2 passed
Engine station of car 3 starting
Engine of car 3 assembled
Wheel station of car 3 starting
Wheels of car 3 assembled
Paint station of car 3 starting
Paint of car 3 assembled
Quality control station of car 3 starting
Quality control of car 3 at station Bodywork failed
Quality control of car 3 at station Engine passed
Quality control of car 3 at station Wheels failed
Quality control of car 3 at station Paint failed
Quality control of car 3 at station Bodywork passed
Quality control of car 3 at station Wheels passed
Quality control of car 3 at station Paint passed
Quality control of car 3 passed
alumne@matagalls:~/SO/Practiques20242025/ENG/Semaphores2$
```

Considerations

- It must be ensured that at the end of the execution all the processes/resources are cleaned up properly.
- The use of global variables is not allowed, except for those that you think are essential.
- All inputs and outputs must be done using file descriptors. **The use of printf, scanf, FILE*, getchar or similar is NOT allowed.**
- **Busy waiting is forbidden.**
- **The use of the functions "system", "popen", or from the same family is NOT allowed**.
- **You must compile using -Wall, -Wextra, and -lpthread 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 "S10.c", and deliver it through eStudy platform.**