laSalle
UNIVERSITAT RAMON LLULL

Operating Systems 2024-2025

Session 6: Pipes and memory c 11/13/24

Objective

In this session you will begin to practice complex techniques specific to operating systems. This session will focus on shared memory usage as well as other tools seen in previous sessions like pipes.

Motivation

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

- Creation of shared memory (shm*get, shmat*).
- Destruction of shared memory (*shmdt*, shm*ctl*).
- Access to the shared memory.
- Creation of pipes (*pipe*)

Previous documentation

To complete this session, it is recommended to read of 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.

laSalle
UNIVERSITAT RAMON LLULL

Operating Systems 2024-2025

Session 6: Pipes and memory c 11/13/24

Underground Network Train Management

Due to the importance of efficiently managing trains and passengers in underground transport networks, Operating Systems students have been commissioned to develop a system that simulates the monitoring and management of trains in an underground network. This system will consist of a single program, which will use fork() to create multiple processes (Control Center and Stations), which will communicate with each other using **pipes** and **shared memory**.

Program Overview

The **Control Centre** is the main process responsible for managing trains and passenger traffic in three simulated stations. This process will remain in execution, waiting to receive reports from the stations on the arrival or departure of trains. Each station updates the number of passengers in the shared memory, and the Control Center will display on screen both the reports received and the updated passenger data.

The program must be implemented in a single file called S6.c. The Control Center will create three child processes that will represent the stations, and these child processes will communicate with the parent process using **pipes**.

Protocol of communication between processes

The communication protocol between the Control Center and the Stations uses pipes. Train events at each station will be reported to the parent process through the following messages:

1. **ARRIVAL_REPORT**: The station sends this message to the Control Center when a train arrives.
2. **DEPARTURE_REPORT**: The station sends this message when a train departs from the station.

Each message will be sent through a dedicated pipe for each station, and the Control Center will be responsible for processing the reports and reading the updated number of passengers from the shared memory.

How the system works

The system follows the following workflow:

1. **Control Center Process (parent process):**
    - **Creating Child Processes**: Create three child processes, one for each station, and establish a dedicated pipe for communication with each station.
    - **Waiting and processing reports**: Remains running, waiting to receive reports from stations through pipes.
    - **Shared memory read**: After receiving each report, it reads the updated number of passengers from the shared memory.
    - **Information Display**: Each time you receive a report, it displays on the screen:
        - The identification of the station that sends the report.
        - The type of event that occurred (arrival or departure of a train).
        - The updated number of passengers at the station.
2. **Station Processes (3 child processes):**
    - **Event simulation**: Each station simulates events of train arrival or departure randomly.
        - Each station will generate a total of **10 train events** .
        - Events will be generated with random time intervals to simulate real-world conditions.
    - **For each event, the station**:
        - **Report Submission**: Sends a report to the Control Center through its dedicated pipe, indicating the type of event (ARRIVAL_REPORT or DEPARTURE_REPORT).
        - **Passenger Update**: Updates the number of passengers in the shared memory corresponding to your station:
            - **Train Arrival**: Increases the number of passengers by a random value between 10 and 50.

- **Train departure**: Decrements the number of passengers by a random value between 10 and 50, making sure that the number of passengers is not negative.

- **Continuation of the simulation**: The station process continues to generate events until all 10 trains have been simulated.
- **Completion Notification**: Upon completion of all 10 events, the station informs the Control Center that it has finished its simulation.

Additional details:

- **Station identification**: Each station has a unique ID (e.g., Station 1, Station 2, and Station 3) that is used in reporting and information visualization.
- **Shared memory**:
  - Each station has a section in shared memory to store its number of passengers.
- **On-screen output**:
  - Control Center displays clear and detailed messages every time you receive a report:

[Control Center] Station 1 – Train arrival. Station 1 now has 120 passengers.

[Control Center] Station 2 - Train departure. Passengers at station: 80.

- **System Termination**:
  - The Control Center remains running until it receives termination notifications from all three stations.
  - After receiving all notifications, Control Center closes the system in an orderly manner.

This system will effectively simulate the management and monitoring of trains in an underground network, applying key concepts of communication between processes and synchronization in operating systems.

Example of execution:

```
user@matagalls-iso:~/SO/Practiques20242025/ICE/S6$ ./S6.exe
[Control Center] Station 0 - Train departure. Station 0 now has 0 passengers.
[Control Center] Station 1 - Train arrival. Station 1 now has 43 passengers.
[Control Center] Station 2 - Train departure. Station 2 now has 0 passengers.
[Control Center] Station 0 - Train departure. Station 0 now has 0 passengers.
[Control Center] Station 1 - Train departure. Station 1 now has 17 passengers.
[Control Center] Station 2 - Train arrival. Station 2 now has 0 passengers.
[Control Center] Station 0 - Train departure. Station 0 now has 0 passengers.
[Control Center] Station 1 - Train departure. Station 1 now has 0 passengers.
[Control Center] Station 2 - Train departure. Station 2 now has 69 passengers.
[Control Center] Station 0 - Train arrival. Station 0 now has 46 passengers.
[Control Center] Station 1 - Train arrival. Station 1 now has 16 passengers.
[Control Center] Station 2 - Train arrival. Station 2 now has 111 passengers.
[Control Center] Station 0 - Train arrival. Station 0 now has 87 passengers.
[Control Center] Station 1 - Train arrival. Station 1 now has 39 passengers.
[Control Center] Station 2 - Train arrival. Station 2 now has 137 passengers.
[Control Center] Station 0 - Train arrival. Station 0 now has 122 passengers.
[Control Center] Station 1 - Train arrival. Station 1 now has 75 passengers.
[Control Center] Station 2 - Train arrival. Station 2 now has 171 passengers.
[Control Center] Station 0 - Train arrival. Station 0 now has 165 passengers.
[Control Center] Station 1 - Train arrival. Station 1 now has 123 passengers.
[Control Center] Station 2 - Train arrival. Station 2 now has 171 passengers.
[Control Center] Station 0 - Train departure. Station 0 now has 117 passengers.
[Control Center] Station 1 - Train arrival. Station 1 now has 155 passengers.
[Control Center] Station 2 - Train departure. Station 2 now has 221 passengers.
[Control Center] Station 0 - Train arrival. Station 0 now has 129 passengers.
[Control Center] Station 1 - Train departure. Station 1 now has 131 passengers.
[Control Center] Station 2 - Train arrival. Station 2 now has 221 passengers.
[Control Center] Station 0 - Train arrival. Station 0 now has 179 passengers.
[Control Center] Station 1 - Train arrival. Station 1 now has 146 passengers.
[Control Center] Station 2 - Train arrival. Station 2 now has 221 passengers.
Control Center: All stations have finished.
user@matagalls-iso:~/SO/Practiques20242025/ICE/S6$
```

laSalle
UNIVERSITAT RAMON LLULL

Operating Systems 2024-2025

Session 6: Pipes and memory c 11/13/24

Considerations

- **There can be no active waiting.**
- **All used Fds and memory must be closed and released.**
- You must handle shared memory safely to avoid racing conditions.
- **The use of "system" or "popen" or similar functions from the same family is not permitted.**
- To carry out this session it is not necessary to use a makefile.
- **The use of global variables only those that are strictly necessary is not allowed.**
- **All entry and exit must be made with file descriptors, the use of printf, scanf, FILE*, getchar, or similar is not allowed.**
- **It must be compiled using the –Wall and –Wextra flags.**
- **Any practice that contains warnings will be inadequate.**
- **The name of the file will be S6.c**
- **The file must have the name, session and the name and login of the students who have done the practice commented at the beginning.**