laSalle
UNIVERSITAT RAMON LLULL

Operative Systems

Session 8: Message Queues

2024-2025

27/11/24

Objective

In this session more complex techniques of operative systems will begin to be used. This session will focus on the creation of message queues to manage messages and data in an environment with more than one thread of execution.

Motivation

Specifically, in this session, the student must train:

- Creation of message queues (*msgget*)
- Closing message queues (*msgctl*)
- Sending messages(*msgsnd*)
- Receiving messages (*msgrcv*)
- Configuration of a message queue

Previous documentation:

It is recommended to read the following references:

SALVADOR, J. (2013). *Programació en UNIX (per a practiques de Sistemes Operatius)*. Páginas 56 - 71.

DNI Appointment System

Due to how complicated and time-consuming it can be to make an appointment to renew the DNI, the Government of Spain has commissioned Operating Systems students to develop a system that simulates the process of requesting appointments for the DNI. This system consists of two programs that must communicate with each other through a message protocol: an administration program, in charge of managing appointment reservations, and a user program, which simulates the request for appointments by people.

Program Overview

The Administration Program ("S8_administration.c") acts as the central server that manages available appointments. This program is kept running waiting for requests from users and manages the status of appointment bookings.

The User Program ("S8_person.c") simulates a person requesting an appointment to renew the DNI. This program interacts with the management program by sending reservation requests and receiving available hours.

Communication protocol between the two programs

The two programs communicate with each other using messages. Each message follows a specific protocol to ensure that the correct information is exchanged. Messages sent can be of two main types:

1. REQUEST_TIMES: The user sends this message to the administration program to request the available times for an appointment.
2. RESERVE: The user sends this message indicating the time they wish to book.

The administration program responds with one of the following messages:

1. CONFIRMED: The requested time is available and has been booked.
2. NOT_AVAILABLE: The requested time is already occupied and cannot be booked.

Each message has a specific purpose and its format depends on the protocol implemented in the system.

How the system works

The system follows the following flow:

1. **Administration Program *"S8_administration.c"*:** This program acts as the server that manages the appointment booking system. When launched, the program performs the following actions:

   · **Loading previous appointments:** When you start, the program checks if there is a file called appointments.dat that contains the current status of the appointments. This file stores previously booked appointments. If the file exists, its contents are uploaded to the system. If it does not exist, it will be created when you make your first booking. The format of this file is free, but it should allow you to save and load the status of the appointments in future executions.

   · **Request management:** The program expects requests from users, which can be of two types:

   · **REQUEST_TIMES:** When the program receives this request, it sends the user the available times to book an appointment. The available hours are: 09:00, 10:00, 11:00, 12:00, 13:00, 14:00, 15:00, 16:00. One hour is available if you have less than two bookings made.
   · **RESERVE:** When the program receives a reservation request, it checks to see if the requested time is available:

   · If the time is available, the program sends the CONFIRMED message to the user and updates the appointments.dat file to record the new reservation.

   ·If the time is not available, the program sends the NOT_AVAILABLE message to the user, indicating that the time is busy.

   The program remains running until it receives a signal from *Ctrl+C*, at which point it terminates in a controlled manner and frees up resources.

1. **User Program "S8_person.c":**
   This program simulates a person who wants to book an appointment to renew their DNI. The process that follows is as follows:

   · **Validation of the DNI:** When starting the program, the user is asked to enter their ID. This is validated through the following process:
      · The first eight digits of the DNI are taken.
      · The index is calculated by dividing these numbers by 23 and getting the rest.
      · The rest is used as an index to access a list of letters associated with the DNIs.
      · If the calculated letter matches that of the ID card entered, it is valid. If it does not match, the user is asked to re-enter the ID until it is valid.

   · **Request for available hours:** Once the ID has been validated, the user program sends a REQUEST_TIMES message to the administration program to request the available hours.

   · **Reception of available hours:** The administration program responds by sending the list of available hours to the user, who displays them on his terminal.

   · **One-Hour Reservation:** The user selects a time, and the user program sends a RESERVE message to the administration program with the selected time. The administration program responds as follows:
      · If the time is available, it sends the CONFIRMED message and the user program shows that the appointment has been successfully booked.
      · If the time is busy, it sends the message NOT_AVAILABLE and the user requests the available times again, repeating the process until they manage to book a time.

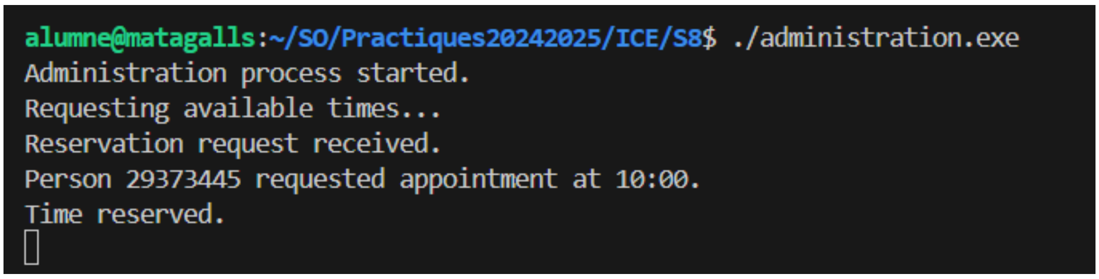
It is important to note that, in order to simplify communication and avoid conflicts, it is assumed that there will not be two instances of the S8_person.c program running simultaneously with the same DNI. In this way, each user has a unique mailbox identified by their ID number, which makes it easier to receive and send personalized messages.

To run the administrator program:
./administration.exe

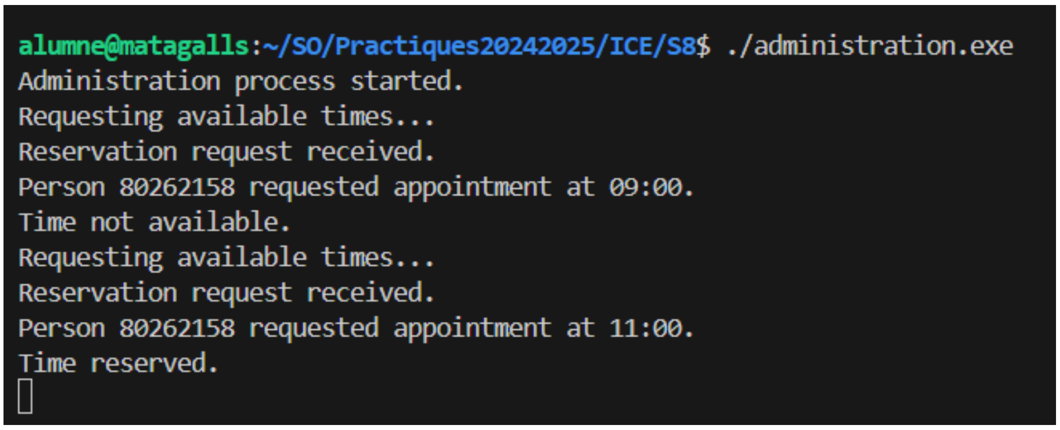
To run the user's program:
./person.exe

Examples of execution

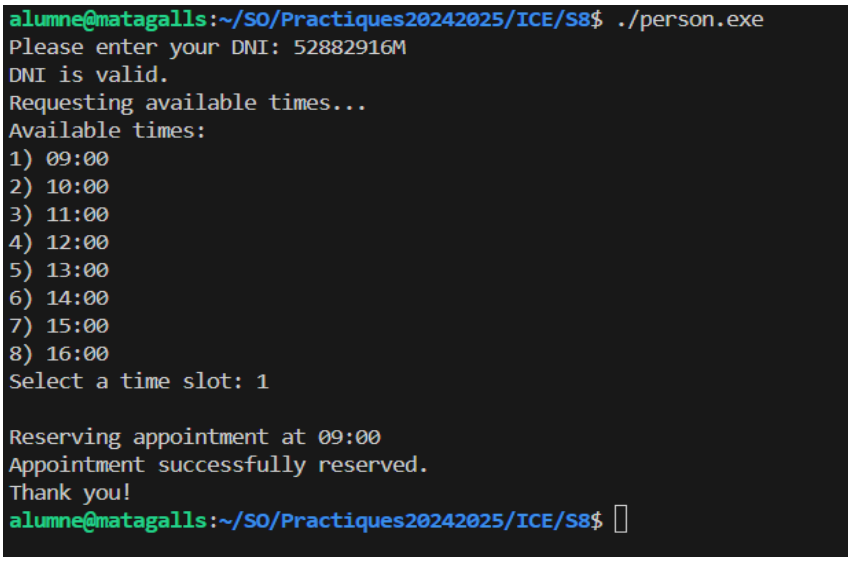
Example of successful management execution:


```
alumne@matagalls:~/SO/Practiques20242025/ICE/S8$ ./administration.exe
Administration process started.
Requesting available times...
Reservation request received.
Person 29373445 requested appointment at 10:00.
Time reserved.
```

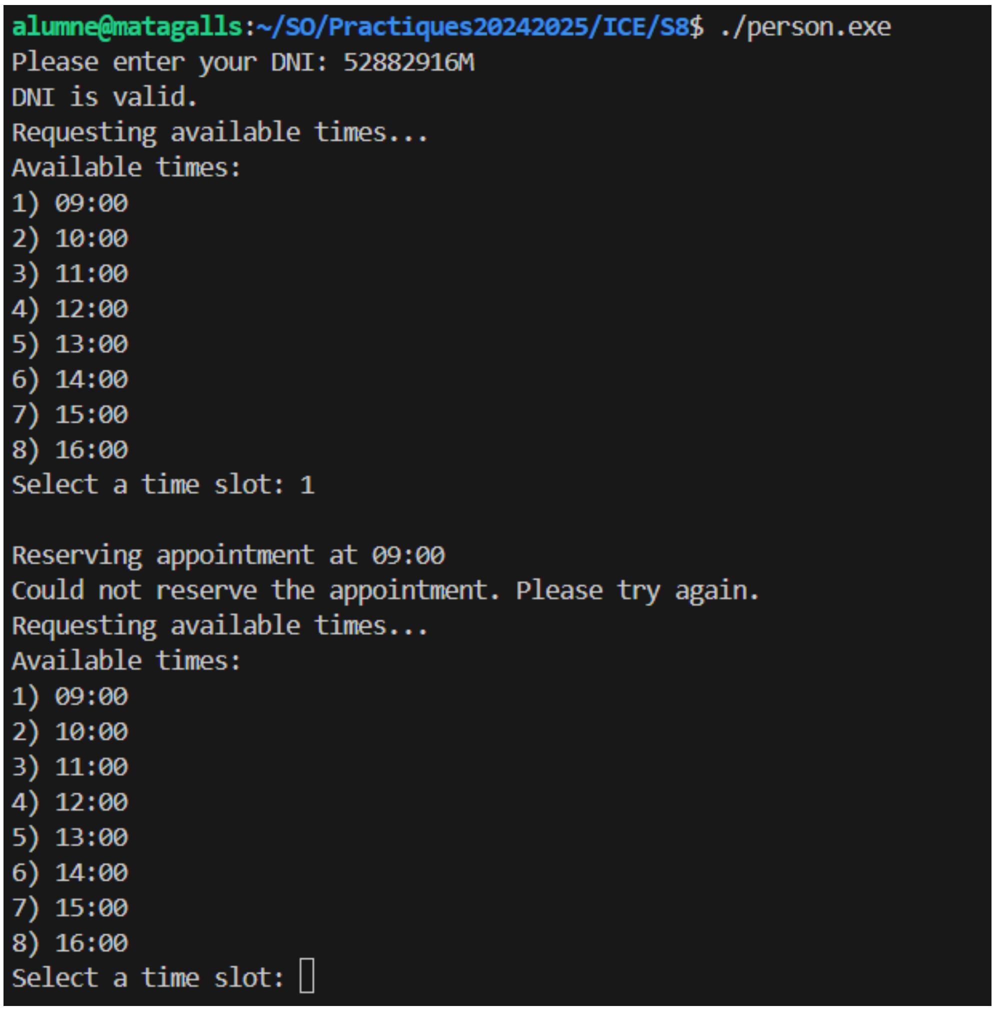
Example of execution when the time the user wants to book is not the correct one:


```
alumne@matagalls:~/SO/Practiques20242025/ICE/S8$ ./administration.exe
Administration process started.
Requesting available times...
Reservation request received.
Person 80262158 requested appointment at 09:00.
Time not available.
Requesting available times...
Reservation request received.
Person 80262158 requested appointment at 11:00.
Time reserved.
```

Example of execution when a user books a slot:

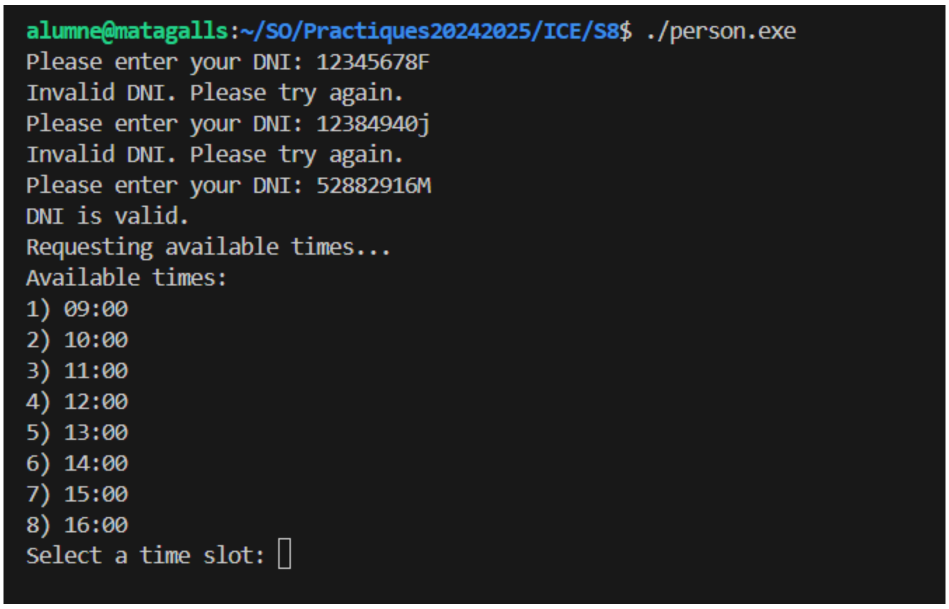

```
alumne@matagalls:~/SO/Practiques20242025/ICE/S8$ ./person.exe
Please enter your DNI: 52882916M
DNI is valid.
Requesting available times...
Available times:
1) 09:00
2) 10:00
3) 11:00
4) 12:00
5) 13:00
6) 14:00
7) 15:00
8) 16:00
Select a time slot: 1

Reserving appointment at 09:00
Appointment successfully reserved.
Thank you!
alumne@matagalls:~/SO/Practiques20242025/ICE/S8$
```

Example when a user requests a slot when there are no more seats available:



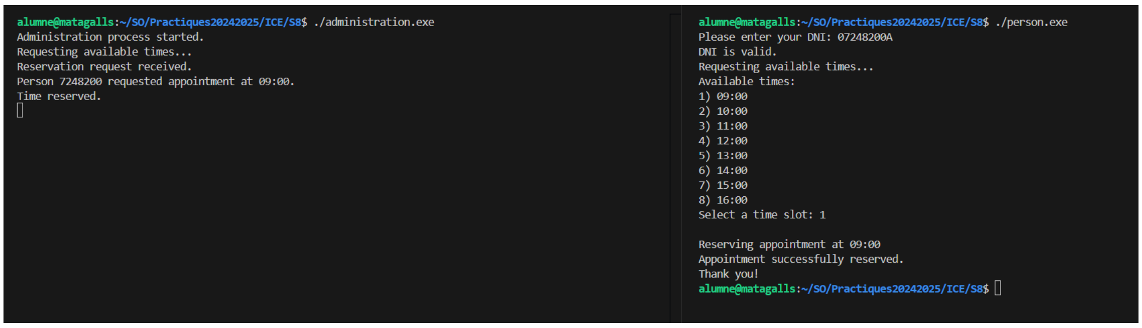
```
alumne@matagalls:~/SO/Practiques20242025/ICE/S8$ ./person.exe
Please enter your DNI: 52882916M
DNI is valid.
Requesting available times...
Available times:
1) 09:00
2) 10:00
3) 11:00
4) 12:00
5) 13:00
6) 14:00
7) 15:00
8) 16:00
Select a time slot: 1

Reserving appointment at 09:00
Could not reserve the appointment. Please try again.
Requesting available times...
Available times:
1) 09:00
2) 10:00
3) 11:00
4) 12:00
5) 13:00
6) 14:00
7) 15:00
8) 16:00
Select a time slot: 
```

Example of when a user enters the wrong DNI:



```
alumne@matagalls:~/SO/Practiques20242025/ICE/S8$ ./person.exe
Please enter your DNI: 12345678F
Invalid DNI. Please try again.
Please enter your DNI: 12384940j
Invalid DNI. Please try again.
Please enter your DNI: 52882916M
DNI is valid.
Requesting available times...
Available times:
1) 09:00
2) 10:00
3) 11:00
4) 12:00
5) 13:00
6) 14:00
7) 15:00
8) 16:00
Select a time slot: 
```

Complete example of the program with a correct operation:



```
alumne@matagalls:~/SO/Practiques20242025/ICE/S8$ ./administration.exe
Administration process started.
Requesting available times...
Reservation request received.
Person 7248200 requested appointment at 09:00.
Time reserved.
```

```
alumne@matagalls:~/SO/Practiques20242025/ICE/S8$ ./person.exe
Please enter your DNI: 07248200A
DNI is valid.
Requesting available times...
Available times:
1) 09:00
2) 10:00
3) 11:00
4) 12:00
5) 13:00
6) 14:00
7) 15:00
8) 16:00
Select a time slot: 1

Reserving appointment at 09:00
Appointment successfully reserved.
Thank you!
alumne@matagalls:~/SO/Practiques20242025/ICE/S8$ 
```

Considerations

1. It can be considered that each user will be connected with a unique and valid ID number.
2. It can be assumed that the management program will always start first.
3. The management program must remain running until it receives a signal from **Ctrl+C**, at which point it must terminate in a controlled manner and release resources.
4. Two files must be submitted: S8_administration.c and S8_person.c.
5. The use or programming of queues, lists, or other complex data structures is not permitted.
6. **Only one (1) message queue can be created for the session solution.**
7. The format of the structure for sending with queues is free. It is guaranteed that the written messages will not exceed 256 characters.
8. **The use of global variables is not allowed, except for those that are essential.**
9. At no time can resources and/or processes be left **unreleased**. Remember to check the resources with *ipcs* and delete the queues not deleted by mistake with *ipcrm -q <id>*.
10. All entry and exit must be made with *file descriptors* , the use of printf, scanf, FILE*, getchar, or similar is not allowed.
11. It must be compiled using the flags –Wall, –Wextra and -lpthread.
12. Any practice that contains *warnings* will be inadequate.
13. The headers of the .c files must have the name, logins of the students and the number of the session.
14. One hour is considered available if you have less than 2 bookings made.
15. The program must validate the DNI following the specified algorithm (calculation of the letter index).