Goals

In the previous sessions, multiple new tools have been used: file descriptors, pipes, sockets, signals, forks, threads, semaphores, shared memory, message queues, etc.

In this last session, the aim is to give a global vision of the tools used by carrying out a globalizing exercise where the most important part will be the design of the solution based on a statement.

Motivation

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

- File descriptors and operations like dup, dup2, execv, execvp, execl.
- Signals.
- Threads and/or forks.
- Pipes.
- Shared Memory.
- Sockets.
- Semaphores and/or mutex.
- Messages Queues.

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.

Space Mission: Cosmic Exploration

In recent years, the space race has advanced significantly, and the La Salle Space agency aims to improve its communication system for its interplanetary missions. The Department of Aerospace Engineering has requested your help to develop a communication algorithm that is efficient and secure for transmitting critical mission data back to Earth.

The algorithm must be capable of processing three types of messages: "EMERGENCY," "REPORT," or "WELCOME." Any other type of message should generate a warning indicating that the message type is not valid for the mission. Each message type must have a different output format, using ASCII symbols to maximize the clarity of the transmitted data, with a minimum size of 6x40 characters.

Given that the La Salle Space team consists of specialists from different fields, the code should be as clear and simple as possible while meeting the following restrictions:

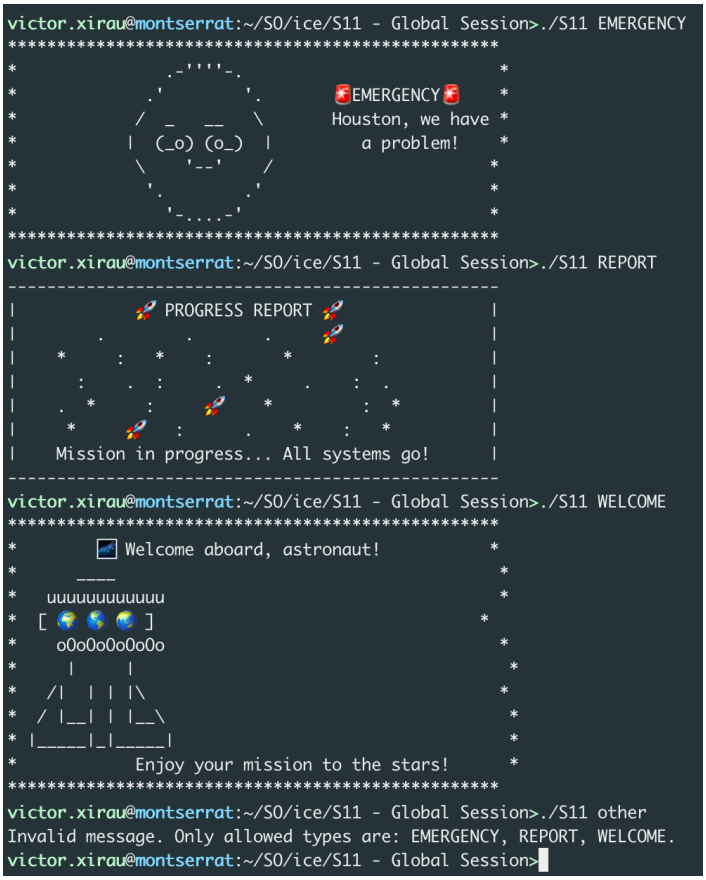
Considerations

- The use of any makefile is not allowed.
- In this session **YOU ARE NOT ALLOWED TO ASK QUESTIONS**
- The use of #defines is not allowed
- The use of any type of variables is not allowed.
- The use of loops (for, while, do while) is not allowed.
- The use of forks or threads is not allowed.
- The use of shared memory or semaphores is not allowed.
- The use of pipes, sockets or message queues is not allowed.
- The use of signals is not allowed.
- The use of text files is not permitted.
- The use of read or write is not allowed
- The use of file descriptors is not allowed
- The use of scanf, FILE*, getchar is not allowed.
- The group names must be commented above.
- It must be compiled using the –Wall and –Wextra flags.
- Any practice that contains warnings or errors when compiling will have a wonderful double duck.
- Any practice that does not follow the restrictions will have a double duck.

Error Handling
The code must properly handle any invalid input or unexpected situations, providing clear and specific error messages.

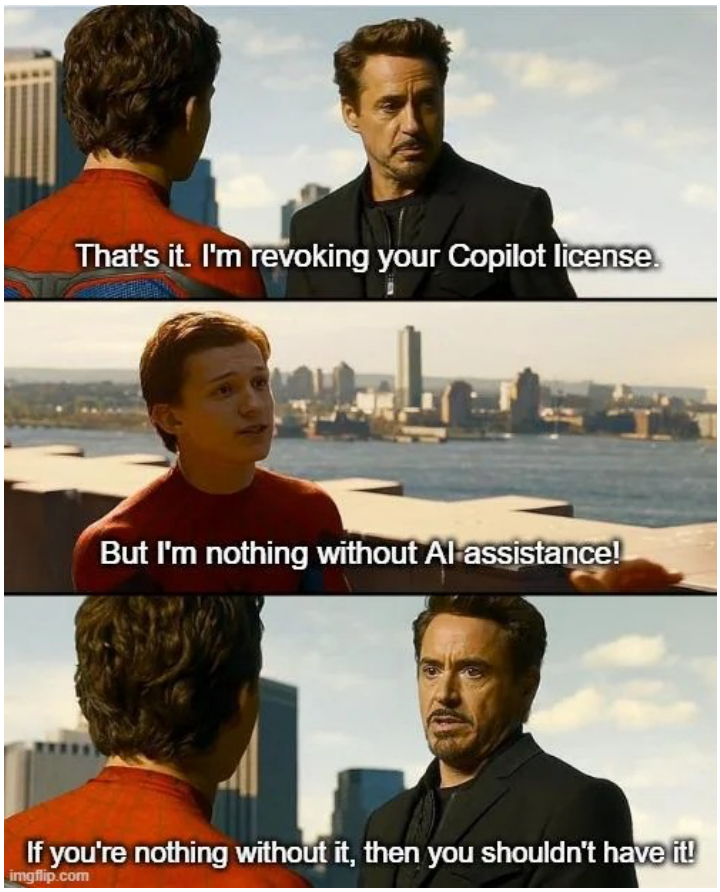
Execution Examples

- When running the command "./s11 EMERGENCY", an ASCII-formatted alert message should be displayed.
- For "./s11 REPORT", a simulated data report should be generated.
- When executing "./s11 WELCOME", an ASCII welcome image for the astronauts should be displayed.



The more original you are with the ASCII image, the higher your grade will be!

Reminder


That's it. I'm revoking your Copilot license.
But I'm nothing without AI assistance!
If you're nothing without it, then you shouldn't have it!
imgflip.com