Ecole Publique d'Ingénieurs en 3 ans
www.ensicaen.fr

Report

A LICENSE TO KILL

22nd of December 2023,
Version 1.0

Aoûj ELOUNELLI, Abdelmalek BELGHOMARI, Haykel SRIHA, Oussama MANSSOURI,
aouj.elounelli@ecole.ensicaen.fr
haykel.sriha@ecole.ensicaen.fr
abdelmalek.belghomari@ecole.ensicaen.fr
oussama.manssouri@ecole.ennsicaen.fr

Project supervisor : **Alain LEBRET**

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

CONTENTS

TABLE OF FIGURES

LISTING OF TABLES

CONTEXT/INTRODUCTION

In the context of an operating systems project aimed at implementing the knowledge acquired during practical exercises on processes, threads, signals, memory management, etc., we were tasked with designing a spy network in a city composed of spies: three source agents and a case officer, who must operate in this city, which also includes other citizens and a counter-intelligence officer. Therefore, it is necessary to create a simulator consisting of several executables that communicate and have access to the same shared-memory.

The "A License to Kill" project is an academic work that implements various aspects of operating systems in C programming, focused on creating a multi-process application simulating a spy network in a city. The goal is to implement features in operating systems, particularly inter-process communications, threads, and synchronization mechanisms like semaphores and mutexes.

Our project must be, of course, structured, documented, and original (in terms of chosen design) with an emphasis on code quality and the integration of the functionalities of the multi-process simulator.

PROJECT STRUCTURE

1. Design

Given the project's somewhat complex statement, we had to think carefully to find a design that encompasses almost all the conditions present in the statement.

1.1. Needs analysis and preliminary design

While trying to stick to the software development rules, we implemented the state pattern distinctively for each character, ensuring it functioned flawlessly in isolation, particularly in scenarios involving interactions at the mailbox, before integrating character movements. This separation allowed us to fine-tune individual components and verify their interactivity, thereby ensuring that the characters' movements and state transitions were seamlessly integrated and functioned cohesively within the broader simulation framework.

1.2. Technological choices and tools

In terms of programming language, the code provided in the archive is in C, in accordance with the project's requirements. For process and thread management, we used POSIX libraries. Inter-process communications are carried out via message queues, signals and shared memory.

For synchronization, we used *mutexes*, barriers and semaphores. The user interface was developed using the *ncurses* library for a textual interface, offering an live visualization of the simulation.

1.3. Modeling and system architecture

The simulator is divided into several main programs:

- ***spy_simulation*** : Initializes the map grid with an arc consistency algorithm, and starts the different executables as child processes and create theshared memory segment.

- ***citizen_manager*** : Takes care of the citizen threads, managing their movements and interactions.

- ***enemy_spy_network*** and ***counterintelligence_officer*** : Responsible for the actions of spying agents and the counterintelligence officer.

- ***enemy_country*** and ***timer***: Manage the reception of encrypted messages and the tracking of simulation time, respectively.

- ***monitor***: User interface for monitoring the simulation.

2. Development methodology

The project was conducted following an agile methodology. This approach allowed us to work iteratively and incrementally, focusing on team collaboration and adaptability. Regular meetings were organized to track progress, solve problems, and plan the next steps. We also used Git for version management and collaboration.

TASKS DISTRIBUTION

	SPY SIMULATION	CITIZEN MANAGER	ENEMY SPY NETWORK	COUNTER INTELLIGENCE OFFICER	ENEMY COUNTRY	TIMER	MONITOR
BELGHOMARI	X	X	X			X	X
ELOUNELLI	X		X			X	X
SRIHA	X	X		X	X		X
MANSSOURI	X				X		X

Tasks distribution

While this previous distribution might seems « ideal » we haven't been able to strictly apply it and execute our tasks according to it to the fullest. Indeed, since all the different programs of the simulator are, one way or another, related and deeply connected based on the structures used and the methodology behind them, we often found ourselves blocked and had to directly regulate and modify the other programs. For example, every member of the group had to regulate the *spy_simulation* which is responsible for the shared memory since all the simulation is based on this memory.

DIFFICULTIES ENCOUNTERED & IMPLEMENTED SOLUTIONS

This section aims to provide a comprehensive overview of the significant challenges we faced during the development of the "A License to Kill" project. Each difficulty is described in detail, shedding light on the complexities encountered in this intricate simulation of espionage and counterintelligence dynamics.

1. Integration of Collaborative Efforts

One of the most substantial challenges we faced in this project was the integration and debugging of code segments developed independently by various team members. Utilizing GitLab for code sharing and version control, we were tasked with merging distinct coding styles and methodologies. Each member's unique approach to processor implementation was beneficial for innovative solutions but led to significant time devoted to refactoring. A major aspect of this endeavor was aligning different approaches to shared memory management, essential for the codebase's seamless interoperability.

Moreover, the merging of these individual executables unveiled unexpected and unusual behaviors in segments of code that previously functioned correctly. This required us to invest a considerable amount of time in debugging. These challenges not only tested our technical skills but also highlighted the crucial need for effective communication, consistent coding practices, and a proactive approach to anticipating and resolving issues in a collaborative software development environment.

2. Inter-Process Communication Complexity

Our project involved multiple processes operating concurrently, requiring a sophisticated mechanism for Inter-Process Communication (IPC) using a shared memory and. The challenge was to ensure efficient and synchronized communication between the different entities like spies, citizens, and the counter-intelligence officer. The difficulty in IPC led to issues like data inconsistency, process deadlock, and challenges in maintaining the real-time state of the simulation. This has been successfully managed by using a simple semaphore between the timer and the monitor. As a matter of fact, il allows the monitor to update its content for each clock's tick.

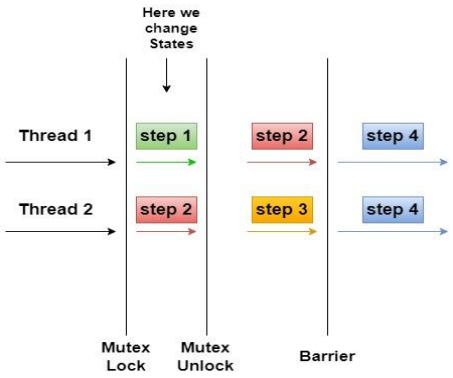
Also, we not only used this semaphore to synchronize the values with the display, but we also initialized a pair of semaphore - *sem_consumer* and sem*_producer* - in order to make the spies and case officer wait for the *citizen_manager* program. Indeed, this program generates the buildings and houses, and then it lets the "consumer" program - enemy_spy_network - assigns the houses to the spies and to case officer once they are created.

3. Concurrency and Synchronization Handling

In our project, handling concurrency and synchronizing access to shared resources amidst multiple threads executing parallel tasks was a complex endeavor. The primary challenge centered around avoiding race conditions to ensure thread-safe operations. This was especially crucial in situations where multiple threads concurrently modified shared data, a scenario that could lead to inconsistent or corrupt states within the simulation.

To address these challenges, we implemented *pthread_mutex* and *pthread_barrier*. The *mutex* locks provided a mechanism to sequence the execution of threads, ensuring orderly access to shared resources. We used these locks to control the flow of execution and prevent race conditions. The barriers, on the other hand, were instrumental in aligning the threads at the end of each simulation round. They ensured that no thread proceeded to the next state until every thread had completed its actions for the current round, thus maintaining the chronological order of events in the simulation.

Here's an example of how we handle our threads, the different colors represent the different states of the threads throughout the time :



4. Agent Behaviour Implementation

Implementing the complex behaviours of various characters using the state pattern presented a significant challenge, particularly for the spies, who exhibit the most unpredictable behaviours. To address this, we conducted a thorough analysis of each character's state pattern before coding. This involved identifying potential states that could follow a given state and determining the conditions necessary for transitioning between these states. For instance, we developed distinct functions and implemented a scheduling system in the shared memory. This system allocated specific time slots for the diverse activities of each character.

While we meticulously adhered to the predefined activities of each character, our state pattern implementation is not flawless. For example, a spy might not always send a message if they are preoccupied with spotting or shopping. This inconsistency means that we may not consistently receive three messages in the mailbox every time. Nevertheless, apart from the

interactions between characters, most other criteria related to character behavior have been successfully met.

5. Comprehensive Testing and Debugging

Despite thorough preliminary studies of the implementation, we encountered several unforeseen behaviours, both in the monitor and in the state transitions. These issues often arose from irregular movements leading to unexpected outcomes. Throughout most of the project, we relied on GDB and terminal displays for debugging and pinpointing errors. However, once we completed the development of the monitor, it became significantly easier to observe the characters' dynamic movements, greatly simplifying the process of identifying and resolving problems.

➔ The challenges we faced throughout this project were not just obstacles, but powerful catalysts for our personal and professional development. Confronting these issues demanded an in-depth comprehension of system programming, mastery of concurrency management, and a refined approach to software design principles. This journey significantly broadened our skillset, preparing us for future endeavours in increasingly complex projects. In this high-pressure environment, laden with assignments and exams, we also honed our management skills. We explored and applied various Design Patterns and AI tools, such as Arc consistency, and practically implemented Operating System tools we learned about in class. Most importantly, this experience enhanced our capability to manage substantial workloads and navigate the dynamics of team projects. It was a rigorous yet rewarding journey that deeply enriched our understanding and appreciation of software development.

CONCLUSION & FEEDBACK

Oussama: This project presented a significant challenge, yet it proved to be a valuable experience for developing my proficiency in C programming. What stood out for me in this project was the emphasis on teamwork. Collaborating in a group underscored the importance of effective project management. Coordinating efforts among team members to handle different aspects of the project showed me the importance of clear communication and shared responsibility.

Overall, while the project was demanding, it significantly contributed to my skill development and provided valuable insights into the complexities of large-scale software projects.

Haykel: Throughout the duration of this project, I had the opportunity to significantly enhance my proficiency in C programming. This involved mastering complex data structures and adeptly applying Design Patterns in C. Given the immense workload within a constrained timeframe, I honed my coding efficiency, developing swift and accurate responses to the problems I face. Additionally, I improved my project management skills on GitLab. This platform proved to be immensely beneficial, leading me to utilize it extensively for various assignments. Furthermore, I learned to effectively coordinate in a team environment, seamlessly integrating disparate pieces of code into a cohesive whole.

This project stood out as the most demanding yet fulfilling experience during my time at ENSICAEN, offering me a valuable opportunity to tackle and overcome a formidable challenge.

Aoûj: As a key contributor to the *enemy_spy_network* program and the *timer* component in the "A License to Kill" project, I found the experience both challenging and rewarding. Working on the *enemy_spy_network* allowed me to delve deep into intricacies of inter-process communication and threading in a Unix environment, enhancing my skills in managing complex, concurrent processes. The development of the timer component was particularly insightful, as it required precise time management to synchronize various aspects of the simulation. These tasks not only tested my technical abilities but also improved my problem-solving skills and understanding of system-level programming. Overall, this project was a significant milestone in my development journey, providing practical experience in handling real-world programming challenges.

Abdelmalek: Coding the simulation wasn't a piece of cake but it was enjoyable. I learned a lot about operating system tools used in practical work and I was driven by the will to achieve this project. It was tough to use the Design Patterns from our Software Engineering classes and the Algorithms from the AI course while coding in C and trying to synchronize processes all along. However, we managed to make it work. I'm happy to understand how multiprocessor software runs now. It felt like an introduction to the world of video games and how PNJs are working.

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04

ministère éducation nationale enseignement supérieur recherche

Normandie Université

Cti Commission des titres d'ingénieur

cdefi Conférence des Directeurs des Écoles Françaises d'Ingénieurs

European Accreditation of Engineering Programmes EUR-ACE®

CONFÉRENCE DES GRANDES ÉCOLES