



Lem-IPC

lemipc

*Summary: This project is not about recoding the lemin with threads for each ant.*

*Version: 4*

# Contents

<b>I</b>	<b>Preamble</b>	<b>2</b>
<b>II</b>	<b>Objectives</b>	<b>4</b>
<b>III</b>	<b>General Instructions</b>	<b>5</b>
<b>IV</b>	<b>Mandatory part</b>	<b>6</b>
<b>V</b>	<b>Bonus part</b>	<b>8</b>
<b>VI</b>	<b>Submission and peer-evaluation</b>	<b>9</b>

# Chapter I

## Preamble

If I didn't have you  
Life would be blue  
I'd be Dr. Who without the Tardis  
A candle without a wick  
A Watson without a Crick  
I'd be one of my outfits without a D\*\*ie  
I'd be cheese without the mac  
Jobs without the Wozniak  
I'd be solving exponential equations that use bases not  
found on your calculator making it much harder to crack  
I'd be an atom without a bomb  
A dot without the com  
And I'd probably still live with my mom

And he'd probably still live with his mom

Ever since I met you  
You turned my world around  
You supported all my dreams and all my hopes  
You're like Uranium 235 and I'm Uranium 238  
Almost inseparable isotopes

I couldn't have imagined  
How good my life would get  
From the moment that I met you, Bernadette

If I didn't have you  
Life would be dreary  
I'd be string theory without any string  
I'd be binary code without a one  
A cathode-ray tube without an electron gun  
I'd be "Firefly," "Buffy" and "Avengers" without Joss Whedon  
I'd speak a lot more Klingon Heghlu'meH QaQ jajvam

And he'd definitely still live with his mom

Ever since I met you  
You turned my world around  
You're my best friend and my lover  
We're like changing electric and magnetic fields  
You can't have one without the other

I couldn't have imagined  
How good my life would get  
From the moment that I met you, Bernadette

Oh, we couldn't have imagined  
How good our lives would get  
From the moment that we met you, Bernadette

Kate & Riki

# Chapter II

## Objectives

You must recode the `lemipc` command that does not exist.

```
$> man lemipc  
No manual entry for lemipc
```

The goal of the project is to make two processes communicate and interact.  
The FAQ included in the `faq.txt` file is part of the subject.

# Chapter III

## General Instructions

- Your project must be written in C.
- This project will only be reviewed by humans. You're free to organize and name your files as you will as long as you respect the following instructions.
- You will turn-in a Makefile.
- If you're smart and use your `libft` library, you must copy the sources and the `Makefile` associated to a folder named `libft` which will have to be located at the root of your repo. Your `Makefile` will have to compile the library calling its `Makefile`, and then compile your project.
- You must manage the errors a reasonable way. Your program will not quit unexpectedly (Segmentation fault, etc...).
- Your program cannot have memory leaks.
- You can use all the `libc`, except for anything that might match an element in your (supposedly complete, with `printf`) `libft`. You will then use the version from your own `lib`.

# Chapter IV

## Mandatory part

The executable will be named `lemipc`.

Game rules:

The idea is to have players (gathered in teams) battle on a 2D board. For a team to be victorious, they will have to be the last team on the board.

When a player dies, they disappear from the board.

For a player to be killed, they must be touched by at least 2 players from the same team, that is, one adjacent tile to the tile the target stands on (diagonal works). Of course, the killing team will have to be different from the target's. When a player understands they're surrounded by at least 2 players from another and same team, he must leave the board and end their execution.

One board tile will only take one player at a time.

Technical constraints:

Each client is a process and there must be only one executable, which means the first player creates shared resources (shm, msgq, semaphores).

In the same way, when a player quits, they will make sure they're the last player on board. If so, they will have to clean all the IPCs created by the first player so they don't remain in the memory (`man ipcs(1)`).

The board must be stocked in a shared memory segment (SHM). Each player can check the content of the board, but they will have to respect the constraints tied to the shared resources and the competitive access (semaphores).

A player can only communicate with other players through MSGQ.

On the map, a player can see if a tile is empty or occupied by another player. If so, the player's team number will be displayed. You cannot differentiate players from the same team.

You must display what's going on on the board.

- Whether in text mode. If so, the first player (the one creating the board) displays the board's content, or each player does, or a specific process takes care of it.
- whether it's in graphic mode. If so, the first player creates the display or you can make another executable dedicated to the display.



# Chapter V

## Bonus part

Here are some bonuses that could be very useful :

- A graphical interface (pleasant, beautiful and intuitive)
- Any other bonuses you like that are useful, well implemented and 100% functional



The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.

# Chapter VI

## Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.