Faculdade de Engenharia da Universidade do Porto

RAKOON

Final Project
Laboratório de Computadores

Class 17 - Group1

Students:

Adriana Almeida up202109752
Duarte Assunção up202208319
Lucas Bessa up202208396
Pedro Gorobey up202210292

Index

Introduction

Our team developed Rakoon, a simulation of a desktop environment (like KDE or GNOME) intended to be a user-friendly interface, which is achieved through the combination of various devices available in the PC hardware.

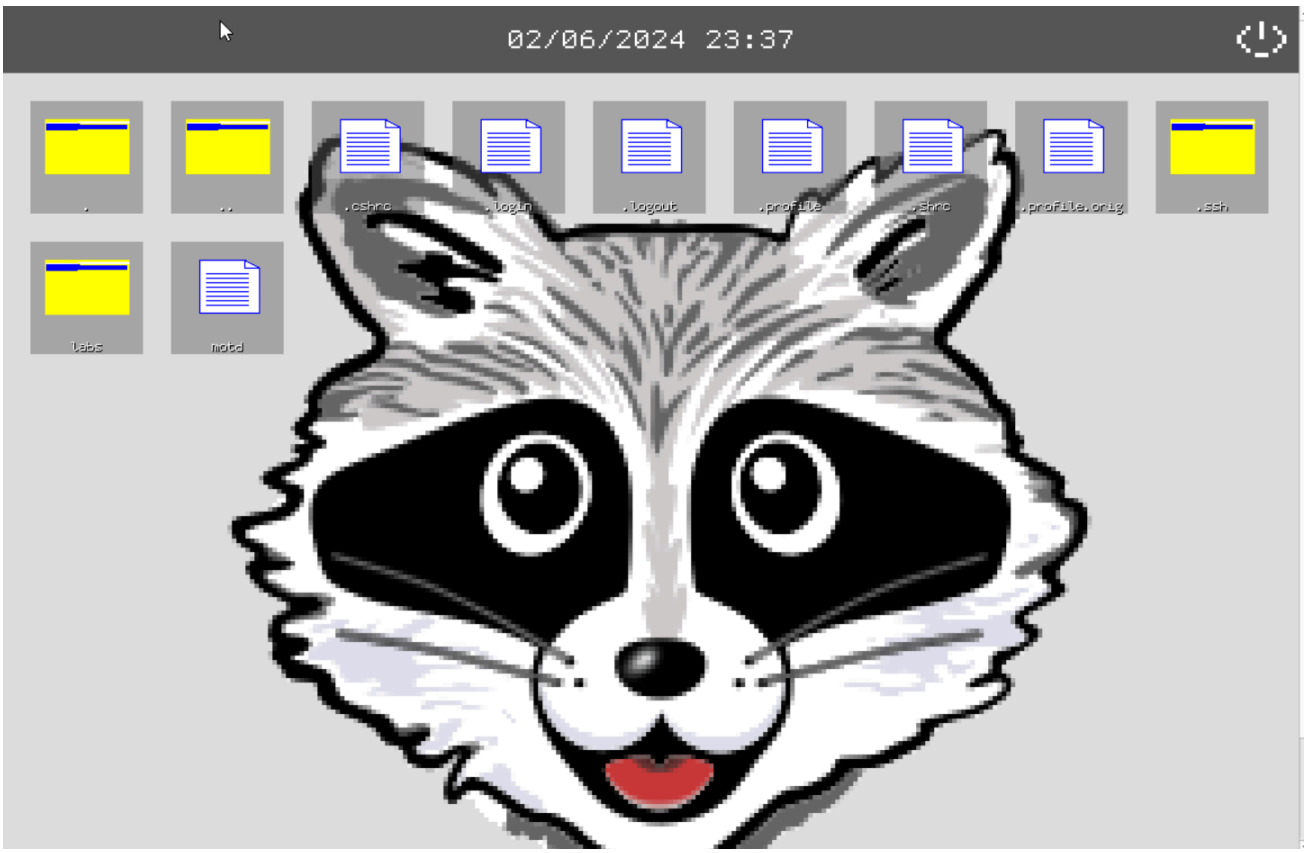
Rakoon also includes the implementation of a chat, that allows users to write and send messages, a file transfer feature and a file explorer, where users can intuitively navigate through the directories on their PC and open supported files. This enhances the overall user experience by providing accessibility and efficient functionalities.

User’s Instructions

★ Run the Program

- ./main

Initially the user enters the desktop environment.
The root directories and files are displayed on the screen.
There is a top bar displaying the real time and a power off button.
Pressing the ESC key terminates the program.



★ File Explorer

- By double clicking in an existent folder that is either in the main desktop environment or in the file explorer the user will access directly the contents of the selected folder.
- The current path is displayed on top of the window
- On the top bar of the window there are arrows with the option to go back or to the front
- The red cross on the top right corner closes the file explorer

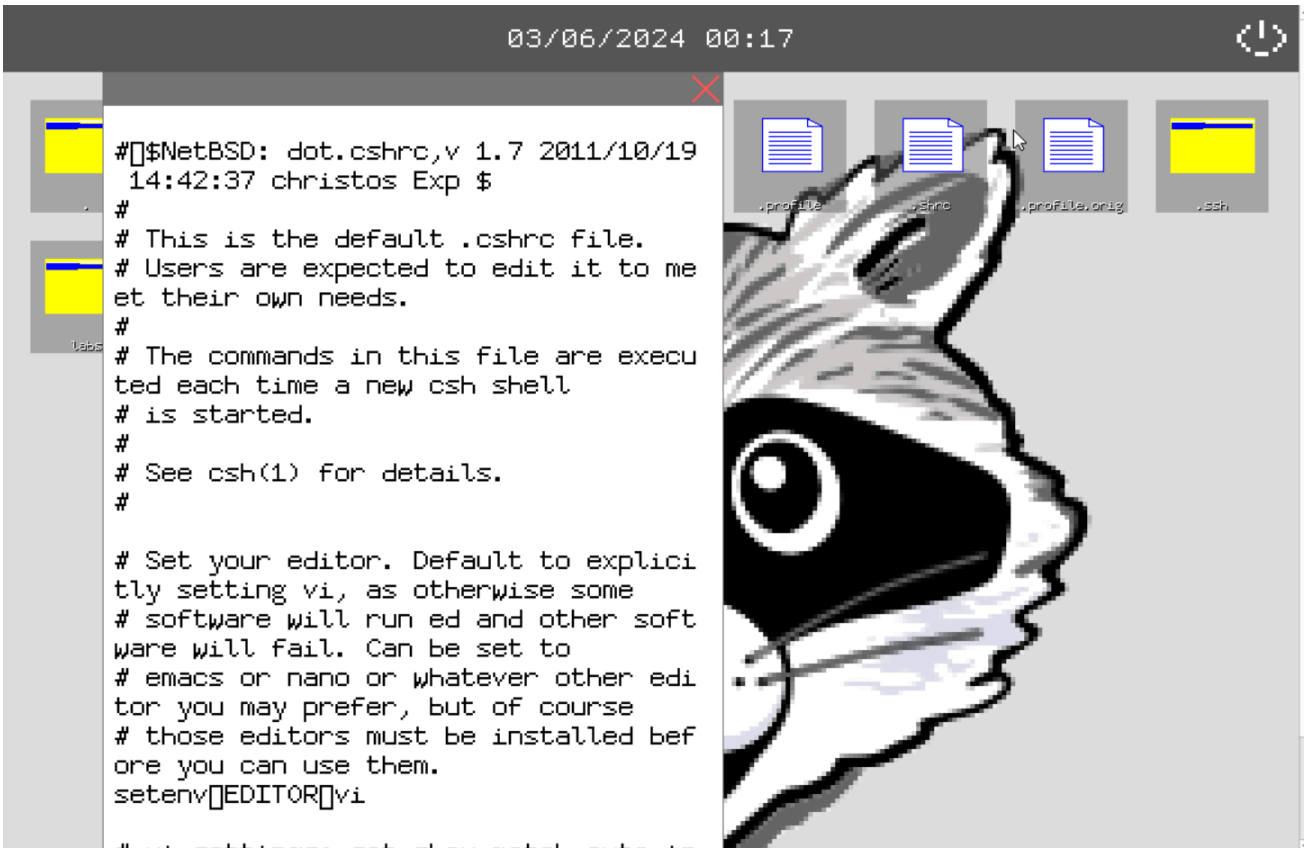
- If the user double clicks on a text document and a File Reader will open, allowing the inspection of the document.
- If the user double clicks on a image and a File Viewer will open and images is displayed.

★ Drawing

- There is no access to this functionality in the Rakoon but by running ./paint in the terminal a Draw application accessible.
- By pressing the mouse left button the user can draw in the created window.
- A simple double click erases the drawings of a certain region

★ File Reader and Viewer

- If the user double clicks on a text document and a File Reader will open, allowing the inspection of the document.
- The up and down buttons in the keyboard allow the user to see the whole page.
- If the user double clicks on a image and a File Viewer will open and images is displayed.



★ Chat

- Each key the user types appears on the screen
- Only numbers, lowercase letters, <Backspace> and <Enter> are allowed

- <Backspace> removes the character behind the cursor
- <Enter> “sends” the message
- Typing “clear” erases the user's screen, but on the other side no message is received
- Typing “quit” or “exit” ends communication on both sides, meaning the chat is terminated
- Sometimes, the initial message might appear distorted
- If your side of the chat closes after a key press, ask the other person to send something while you are in the chat. This should solve the problem

★ File Transfer

As a sender:

- The filename of the source file (the one to be copied) is required
- If the filename is valid (the file exists at that location), the program will wait for the other side to send a start-transfer byte
- After 30 seconds, if no such byte is sent, the program terminates normally
- If such byte is received, the program starts the file transfer
- Due to the serial port constraints, only one byte is sent at a time - even for small files, it can take a while, so try not to transfer files with a massive size
- The number of transferred bytes / total bytes is shown on the screen
- When the file is done transferring, the program sends an end-of-transfer byte and terminates

As a receiver:

- The filename of the destination file (where to copy the file) is required
- The file should not exist (a new one is created instead to prevent overwriting)
- The program then sends a start-transfer byte and continually receives bytes from the serial port
- When the program receives a end-of-transfer byte, it ceases execution and quits

Project Status

Devices

Input Output Device	Purpose	Details	Interruptions
Timer0	Control screen refreshes and small intervals	Default frequency (60Hz)	Yes
Keyboard	Obtain user input through keyboard press	Each scancode is translated to a character Used for text input and for some commands	Yes
Mouse	Obtain user input through mouse movement and button clicks for the cursor	Uses position to move cursor Uses buttons to drag windows Has multiple states to manage different actions/movements	Yes
Graphics	Display the windows with mode 0x14C (direct), resolution 1152x864, 2^{24} colors	Double/Triple Buffering V-Sync Refreshes at a certain frequency if necessary Uses moving objects (windows) Uses a custom font	No
RTC	Display the current time	Used to read the current time Uses alarm to get the date each minute	Yes
Serial Port	Chat and file transfer	Due to some limitations the	Yes

		byte send/receive uses a delay (around 0.3ms) Does not use FIFOs Special bytes (defined with macros) are used	

Functionalities

Functionality	Devices	Implementation
Real date and time display	RTC	Complete
Desktop Environment	Graphics	Complete
Creation of new window	Graphics	Complete
File explorer Window	Graphics, Mouse, Keyboard	Complete
Navigation through directories	Graphics, Mouse	Complete
File Reader	Graphics	Complete
Scroll in File Reader	Keyboard	Complete
Image Viewer	Graphics	Complete
Draw	Mouse, Graphics	Exists, incomplete
Chat	Serial Port	Works but not on Rakoon
File Transfer	Serial Port	Works but not on Rakoon

Devices Specific Usage

The implementations of the drivers found in “proj/src/interrupts”, are completely independent of the desktop environment logic implemented and count with lower level functions crucial for the devices usage.
Located in “proj/app” are the app devices that use the interrupts (implemented separately) to handle the app.

Timer0:

Timer0 was mainly used to control the frame rate and small intervals.
Implementation of Timer0 is in both files, timer_i8254.c and app_timer.c

Relevant implemented functions:

➔ app_timer.c functions handle time for the project, specific windows and the IPC. These functions are used in “proj/app/setup.c”.
- int_rect_t (ipc_timer_handler) ( )

➔ timer_i8245.c functions manage the i8254 timer, more specifically Timer0. This files main functions can initialize the timer, change its frequency, handle interruptions and manage observers.

- int timer_init ( )
- int (timer_set_frequency) (uint8_t timer, uint32_t freq)
- void (timer_int_handler) ( )
- int timer_add_observer(int ticks, void* id, void (*action) (), bool repeat)
- int timer_remove_observer(void * id)

Keyboard:

The keyboard obtains user input and handles the scancode received, translating it to a character. The scancodes macros are defined in “proj/src/interrupts/scancodes.h” and in the file scancodes.c is defined the function to convert the scancode into the proper character.
Depending on the keys pressed and the current scenario it might create a new window, such as the File Explorer, or handle text input enabling the implementation of the chat and the notes.

The proper synchronization between the keyboard and the mouse is also ensured.
Files keyboard.c, kbc_i8042.c and app_keyboard.c contain the most relevant functions for the appropriate interactions to run smoothly. These functions are used in “proj/app/setup.c”.

Relevant implemented functions:

➔ kbc_i8042.h implements functions and helper functions that write commands to the keyboard controller
- int (kbc_mouse_write)(uint8_t byte)

➔ keyboard.c has function crucial to the correct handling of interrupts
- void (kbc_ih) ( )
- int (keystroke_get) (keystroke_t *k)

➔ app_keyboard.c implements for the different contexts the correct handling of the user input
- int_ret_t ipc_keyboard_handler( )

Mouse:

The mouse has many utilities in the project. Moving the cursor inside the desktop environment Selecting a given directory or document is possible by a double click, opening, resizing, moving and closing windows, is also through the mouse.
The mouse packets are used to obtain the position of the mouse and check if the buttons have been pressed.
These implementations can

Relevant implemented functions:

➔ mouse.c file includes a function to properly handle the interrupts
- void (mouse_ih) ( )

➔ app_mouse.c
- int_ret_t ipc_mouse_handler( )

Graphics:

Graphics is used to display the desktop environment, the windows created by the user, the many folders and documents that correspond to drawn XPM images as well as the characters in text writing or reading. Elements defined in XPM format are converted into sprites and used throughout the desktop enhancing the user’s experience.
The project is in video mode, the VBE mode used is 0x14C with resolution 1152x864 and 32 bit color mode, 8 bits for each rgb color.
To allow more efficiency we implemented triple buffering.
Page flipping (ex: drawing of new windows) is achieved with the function vg_refresh( ) implemented in graphics.c.
The directory “proj/src/graphics” contains multiple files contributing to the Graphics implementations.
Relevant implemented functions:

➔ graphics.c contains important function to handle the VBE
- int (vg_start) (uint16_t mode)
- int (vg_refresh) ( )
- int (vg_fill) (uint8_t *from)
- int(vg_clrsrc) ( )
- int (draw_sprite) (window_t *win, const sprite_t +img, int16_t x, int16_t y)

RTC:

The real time clock is used to display the current time on the desktop environment top bar. The folder “lab6” contains the definition and implementation of the lower functions needed

Serial Port:

Serial Port enables communication between two machines. In this project two functionalities were implemented, the communication through text messages and the exchange of files.

Relevant implemented functions:

- serial_port(int argc, char* argv[])
- chat(char* self, char* other)
- file_transfer(uint8_t stance, char* origin, char* destination)

Code Structure

Module Interrupts interrupts.c 3%

In this module the struct int_info_t is used to store information about the interrupts received (hook id, handler and enabled status). The file has functions to enable and disable interrupts and the function int (interrupts_run)( ) that calls the functions defined in separate files to handle Timer0, keyboard, RTC and mouse device.

Module Keyboard keyboard.c 1%

The functions in this file configure the keyboard, interrupts are handled, the scancodes are processed and keys pressed are converted to characters.

Module KBC kbc_i8042.c 1%

This module provides implementations of functions that interact with the keyboard controller helping the implementations of the keyboard and mouse. Includes functions to write and read from the KBC.

Module Mouse mouse.c 1%

Mouse module counts with functions to handle packet processing, data retrieval and interrupts. Definition of mouse_packet_t, an alias of struct packet, makes the code readable.

Module Timer timer_i8254.c 1%

In this module the functions implemented interact with the i8254, specifically focusing on Timer0. It includes functionalities for initializing the timer frequency and observer, handling the interrupts, setting frequency and managing the observers.

Module RTC rtc.c 1%

This module has functions to interact properly with the RTC. Including functions to read and write to the registers, to start and configure the RTC and to enable, disable, handle and identify the interrupts.

Module Graphics graphics.c 3%

In this module a set of functions are defined to handle the operation usinge the VBE . There is initialization of the video mode, buffer handling, and function to draw different elements

Module Colors colors.c 1%

Here is the definition of the function to blend ARGB colors. It counts with the two structures rgb_t and argb_t, both this structures represent color values in different formats.

App Devices Modules

app_keyboard.c 2%
This module, using functions defined previously handles the project specific functionalities related to the keyboard and sends the data to the correct window. The function ‘ipc_keyboard_handler’ is responsible for handling inter process communication keyboard related events.

scancodes.c 1%
Implements the conversion from scancode to character.

app_mouse.c 4%
Here are defined functions to control the app mouse with the IPC and interactions with the graphical interface, with functions to check for double clicks, and mouse state.

app_rtc.c 1%
The functions in ‘file_expl_draw.c’ initialize and draw the directories in the user’s computer
This module is necessary to include the real time clock with the app, using functions defined in the ‘interrupts’ and ‘lab6’ folders. The function ‘rtc_handler’ updates the global variable and sets a flag to update the time.

app_timer.c 1%
File with Timer0 functions to handle the desktop environment interrupts related to frame updates and refreshes as well as the inter process communication.

Module File Explorer

file_expl.c 5%
Implementation of the file explorer that allows changing directories, going back and forth, selecting elements, etc.

file_expl_draw.c 2%
Functions to load file explorer sprites and draw directory elements.

home.c 2%
A simpler version of the file explorer, it's the first "window" you see when you run the program.

Module Directories directories.c 2%

This module is crucial to the implementations of the file explorer, has it defines function to handle the directories contents, storing the relevant information. In this file are defined structs to represent the directory and assist the information storing. The struct 'vector_dir_item_t' stores 'dir_item_t' items, while the struct 'dir_node_t' is used to link directories, enabling hierarchical storage.

Module Window window.c 4%

This module manages the windows to be created in the desktop environment. Using the buffer the function 'draw_window' can initialize a new windows or close one and clear the buffer used.

Module Shared shared.c 2%

This module has the necessary function to manage the desktop environment and render some very useful items. In this file structs are defined, 'dir_node_t' is used to store information about directories, xpm_image_t is for XPM images, 'window_t' stores windows properties such as width, height and buffer.

Modules in IPC

api.c 4%
This file defines the functions to manage and manipulate the windows of the desktop environment. The ipc uses sockets to interact with the middleman server. The functions implemented here can draw and refresh widows.

client.c 1%
Just contains a function to connect to a specific socket address.

protocol.c 1%
Contains functions to allow the protocol used to work (just defines functions that create shared buffers between the middleman and the client).

server.c 3%
Communicates with its clients through the middleman.

middleman.c 3%
Implementation of the functions that allow the bridge between the client and the server.

Module Image Viewer

cbmp.c
The file implements the important functions to create, read and manipulate the BPM image files.

image_viewer.c
This file allows the creation of a window, where an image will be displayed. To do this correctly functions verify the images dimensions and the window position.

Module Text Viewer text_viewer.c

In this module are implemented the functions that contribute to the display of a document's contents. It starts by creating a new window, reading the lines from the file to display and drawing them on the screen.

Module Main.c 2%

This file is the entry point for the whole project, the first step it takes is executing “lcom_run_proj”. This file initializes the starting middle man for the IPC and forks a new process to “lcom_run_proj”.

Implementation Details

IPC

The main process creates a socket and connects to a middleman’s socket.

The middleman’s purpose is to wait for events sent by the server and send them to the client and wait for requests sent by clients and direct them to the server.

The windows connect to the middleman to be able to communicate with the server.

The middleman also serves the purpose of passing window buffers to the server, since it has a faster and safer response than the client. This is done after the client requests a refresh and the server acknowledges the request.

Serial Port

While making both the chat() and the transfer() functions, a problem arose: subscribing COM1 with both IRQ_REENABLE and IRQ_EXCLUSIVE prevented the reception of interruptions on at least one side of the Serial Port
The reason for this problem is still unknown. A way to circumvent it was eventually found, though.
The Serial Port operates with delays rather than polling or register reads. This means that, whenever the program receives from or sends a byte through the Serial Port, it waits a certain amount of time (0.3ms - 0.4ms) as to prevent overwrite and repeated read of the same byte
The delay in receiving user input is virtually none due to the approach taken: after each scancode (the program ignores break codes), the corresponding character byte is sent through the serial port to the other side, which stores it in a buffer (char[] array). When one of the users clicks <Enter>, it simply signals the other user to print what is in the buffer and then a new line (if there is at least one character in the buffer)
Clicking <Backspace> sends a byte to the other user telling them to remove the last character in the buffer
Typing “clear” and then clicking <Enter> clears the screen of the user who sent it and tells the other one to clear the buffer
Typing “quit” or “exit” and then clicking <Enter> exits the chat on both sides

References

Lib CBMP
https://github.com/mattflow/cbmp
This library was used to read bmp images.

Conclusion

In this project we were able to implement the desktop environment initially planned, even though we had ups and downs along the road and some aspects suffered a few alterations to better fit the actual implementation.
A few features were trickier such as the implementations of the serial port and the IPC.
The biggest troubles we had are related to the merging all the implemented functionalities.