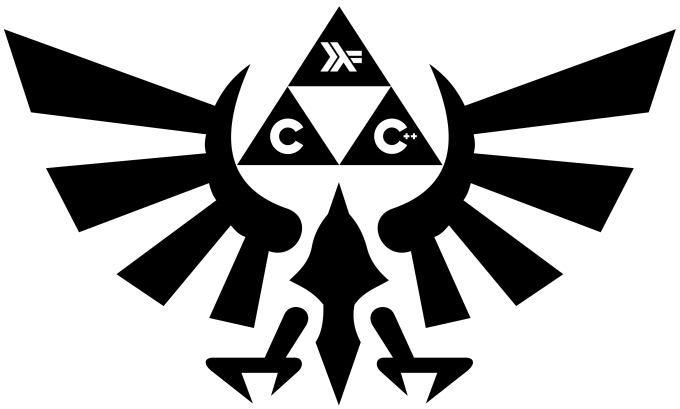
B3 - Paradigms Seminar

B-PDG-300

Rush 3

GNU Krell Monitors



2.2

Rush 3

binary name: MyGKrellm
language: C++
compilation: via Makefile, including re, clean and fclean rules

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

The `*alloc`, `free`, `*printf`, `open` and `fopen` functions, as well as the `using namespace` keyword, are forbidden in C++.

*GNU Krell Monitors (GKrellM) is a system monitor software based on the GTK+ toolkit that creates a single process **stack of system monitors**. It can be used to monitor the status of CPUs, main memory, hard disks, network interfaces, local and remote mailboxes, and many other things.*

Let's be clear: this rush does not aim to make you completely re-implement GKrellM.

Or does it?

However, you are expected to create a clone of it. The subject is purposefully the least restrictive of the three rushes, so that you can feel free to enjoy yourselves and add whatever you want to your program. However, **dat ain't no party**, and a few mandatory steps must be validated first.

This rush consists in three core steps and a bonus step. Each step must be done entirely and perfectly before the next is started, as they are increasingly difficult. You might need to redesign your first step once you completed the second one.

You'll be assessed as much on your architecture as your features.

Your system monitor can be seen as a module container. Modules are displayed as **a vertical stack**, on top of each others. It is possible to add and remove the available modules so that users can adapt the information to their needs, or even reorder them however they want.

GKrellM doesn't need privileged rights to run correctly, and neither should your monitor.

Once properly configured and with a sexy skin, **GKrellM** is a very nice system monitor. But before we get to that point, there's work to be done!

It must be possible to start your system monitor either in "text" or in "graphical" mode, with the same functionalities. In "text" mode, your monitor must be displayed on your terminal using the **nCurses** library. In "graphical" mode, your monitor must be displayed in a graphical window using the **SFML** library.

Keep this constraint in mind from the first line of code you write. Oh, and did you know **nCurses** can handle mouse events ?

Whatever mode it is started in, the visual quality and ergonomy of your monitor will have a non-negligible impact on your grade. For instance, some data are better represented by a numeric value or words, while a histogram or curve is more suited for others. **Be imaginative.**

GKrellM has an awesome skin system. What about yours ?

The architecture of your monitor code is entirely up to you. However, you must use at least the two following interfaces:

- `Krell::IModule`: describes the behavior of a module of your monitor
- `Krell::IDisplay`: describes a display mode of your monitor

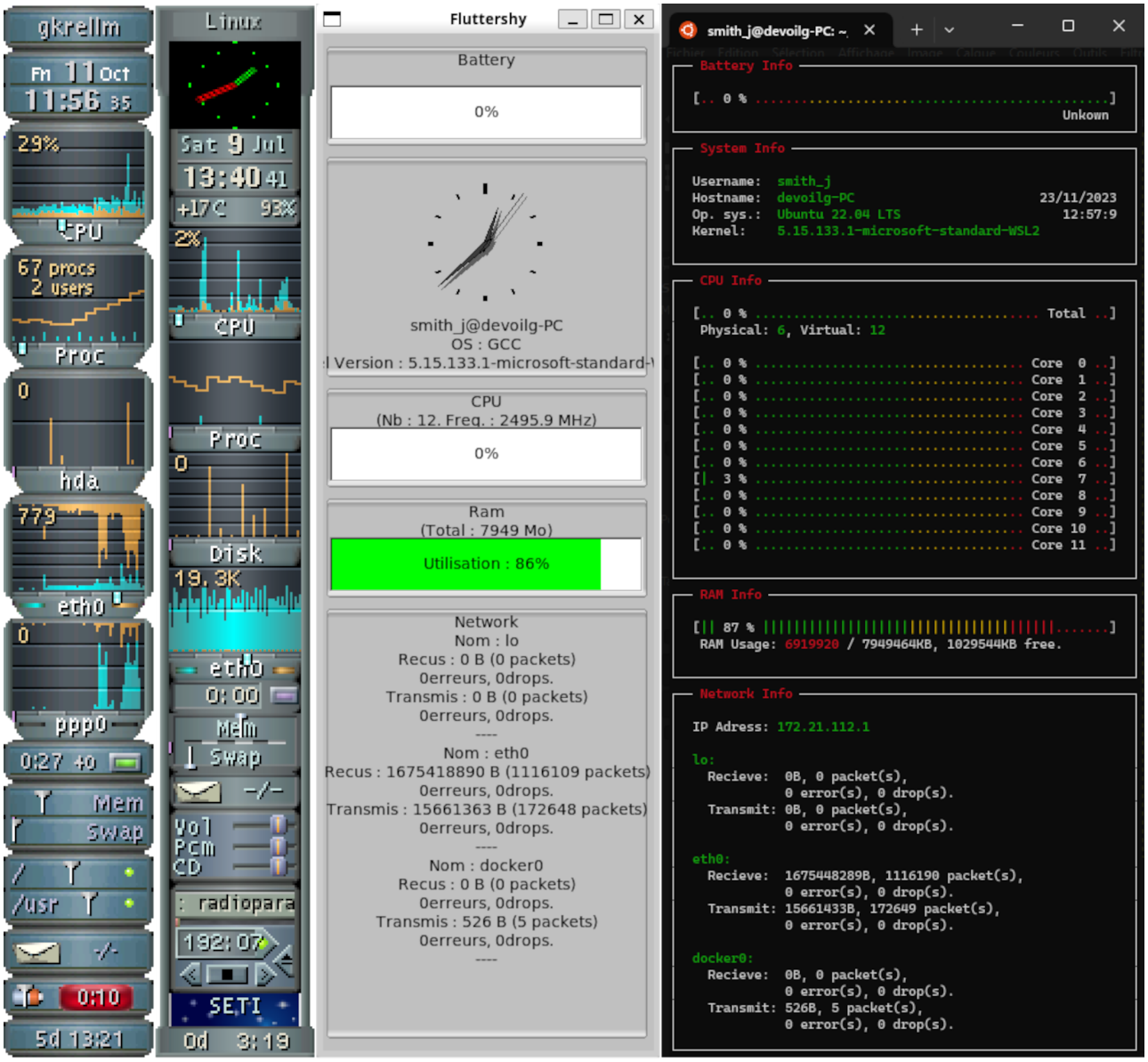
The content of these interfaces is up to you. Take great care to design your code. You may want to take some time to wonder why we force you to use them. Again, we don't need something visually fancy. A simplier solution is often better.

These interfaces will allow modularity of your program. As your modules and displays are always referenced through their interfaces, it can be switched flawlessly :

- Beginner: modules and displays are configurable at compilation time
- Intermediate: modules and displays are configurable on the command line
- Advanced: modules and displays are configurable and can be switched, added or removed at runtime

Be brave. Aim for **Advanced** features.

EXAMPLES



GKrellM (2 skins) MyGKrellM GTK & nCurses examples

See how GKrellM ***stacks*** its modules ? That's what we want !

Steps

Step 1 : Monitor Core

Create the core of your program:

- It **must** orchestrate your modules and displays
- Modules and displays **should** be used through their interfaces
- It **could** be able to change modules and/or displays at runtime.

Step 2 : Modules and nCurses

You must implement the following modules:

- Hostname and username module
- Operating system name and kernel version module
- Date and time module

You must implement nCurses text display. Something simple, graphs are bonuses.

Step 3 : More modules and SFML

You must implement the following modules:

- CPU module (model, frequency, number of cores, activity…)
- RAM module

You must implement SFML graphical display. Something simple, graphs are bonuses.

Bonus

Modules :

- Network load module
- Battery module
- As many **useful** modules as possible

Display :

- Display datas with graphs (bar charts, percentage bar, line graph, histogram…)
- GTK display
- SDL2 diplay
- Qt5 display
- Unreal Engine 5.3 display