Entités

- **Modules**
- **Ncurses**
- **Qt**

Folder répartition:

./include

-IMonitorModule

-IMonitorDisplay

./src

-Modules

-Cpu

-User

-Ram

-Network

-Ncurses

-Cpu

-User

-Ram

-Network

-Qt

-Common

-Core

-ExecCommand

main_ncurses.cpp

main_qt.cpp

Makefile ncurses

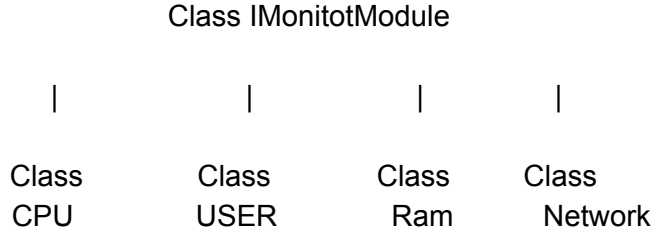
Makefile Qt

IMonitorModule :

Le but de cette partie est d'avoir toutes les données demandés pour notre logiciel. Le principe est d'avoir une classe IMonitorModule, auquel on va lui associer toutes les données nécessaires réparti dans 4 classes (CPU, USER, RAM, Network):

- Monitor Core
- Hostname and UserName
- Operating system and kernel versions
- Date and time
- Cpu
- Ram
- Network

Composition de cette class et de ces filles:



IMPORTANT: pour cette partie, envisager la compilation sur MAC (#ifdef __APPLE__)

Class IMonitorModule

```
{
	public:
		Module();
		virtual Module();

		virtual void Init() = 0; // Initialisera le module
}

Class User : public IMonitorModule
{
	public:
		constructor & destructor
		get & setter
		getTime()
		getUpTime()
	private:
		_userName
		_kernel
		_osVersion
		_machine
		_sysName
}
```

Class CPU : public IMonitorModule

```
{
	Définir les constantes permettant de récupérer les données
	public:
		Constructor & destructor
		get & setter
		int getNbCore() const
		CPUCore &getCore(id)
	private:
		Core **_core
		int _nbCore
}
```

Class Ram : public IMonitorModule

```
{
        public:
                constructor & destructor
                get & setter
                refresh()
        private:
                _maxSwap
                _usedSwap
                _maxRam
                _usedRam
                _buffers
                _cached
}
```

Class Network : public IMonitorModule

```
{
        public:
                constructor & destructor
                get & setter
                refresh()
        private:
                _interfaceActived
                _bytesReceived
                _bytesTransmitted
                _packetsReceived
                _packetsTransmitted
}
```

IMonitorDisplay

Cette partie, concerne la partie ncurses, donc l'affichage dans le terminal. On va donc pouvoir récupérer toutes les informations nécessaire grâce à la class IMonitorModule. Chaque information étant présente dans les classes des modules, doivent être présente. Le design est a choisir. Et ou l'on place les données aussi.

Composition de la class :

Class IMonitorDisplay

|

Class NCurses

Class CPU	Class Network	Class Ram	Class User

Class IMonitorDisplay

```
{
	public:
	private:
}
```

Class NCurses : public IMonitorDisplay

```
{
	public:
		get & setter
	private:
		minWidth
		minHeight
}
```

```
Class displayCPU : public NCurses
{
        public:
                displayCPU(moduleCPU *CPU)
                virtual display(int, int, int, int)
        private:
}
```

```
Class displayNetwork : public NCurses
{
        public:
                displayNetwork(moduleNetwork *Network)
                virtual display(int, int, int, int)
        private:
}
```

```
Class displayRam  : public NCurses
{
        public:
                displayRam(moduleRam *ram)
                virtual display(int, int, int, int)
        private:
}
```

```
Class displayUser : public Ncurses
{
        public:
                displayUser(moduleUser *User)
                virtual display(int, int, int, int)
        private:
}
```

```
class NCursesDisplayUser : public ANCursesDisplay {
protected:
        MonitorModuleUser *user;
public:
        NCursesDisplayUser(MonitorModuleUser *);
        virtual ~NCursesDisplayUser();
        void display(int, int, int, int);
```

exemple de la classe DisplayUser, qui sera a priori semblable pour les autres class Display (CPU, RAM, Network)

CORE ET EXEC COMMAND

La classe Core est a part, elle sera un attribut de la classe CPU.

Module Core (gestion d'un Core: récup)

Class CPUCore

{
	public:
		Constr(id)
		Destr par défault
		std::string getName()
		(mac : sysctl -n machdep.cpu.brand_string)
		(linux: cat /proc/cpuinfo | grep name | head -n 1)
	private:
		int _id
		int _usage
		int _freq
		int _idle
}
Tip: linux → cat /proc/stat | grep cpu
	Boucler sur l'**id** pour faire stringResult +

Exécution d'une commande (renvoie les données à étudier)

Class Command : /common/ Command.cpp/hpp
public:
Cons/Dest par défault
static Exec(std::string command) (accessible PARTOUT)

Exec —> std::system(@param) < 0 return

Ajouter " > /tmp/rush3.rush"

Std::ostream file("/tmp/rush3.rush")
std::sstring ss = file.rdbuff()
remove "/tmp/rush3.rush"

Return ss