Institut Mines-Télécom

Didier GUERIOT (Dpt. ITI)
Johanne VINCENT (Dpt. INFO)
Reda BELLAFQIRA (Dpt. ITI)

Didier.Gueriot@imt-atlantique.fr
Johanne.Vincent@imt-atlantique.fr
reda.bellafqira@imt-atlantique.fr

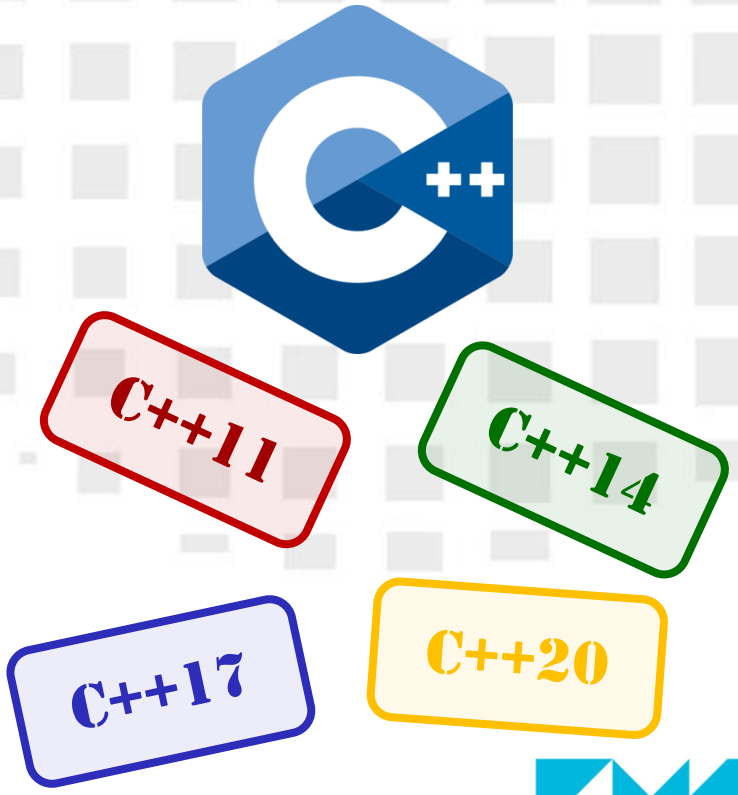
Advanced C++ programming:

from relevant design to efficient implementation

```cpp
#include <iostream>

struct A {
  A()          { foo() ; }
  virtual ~A() { foo() ; }

  virtual void foo() { std::cout << "1" ; }
  void bar()         { foo() ; }
} ;

struct B : public A {
  virtual void foo() { std::cout << "2"; }
} ;

int main() {
  B b ;
  b.bar() ;
}
```

Selon la norme C++17, qu'affiche l'exécution de ce programme ?

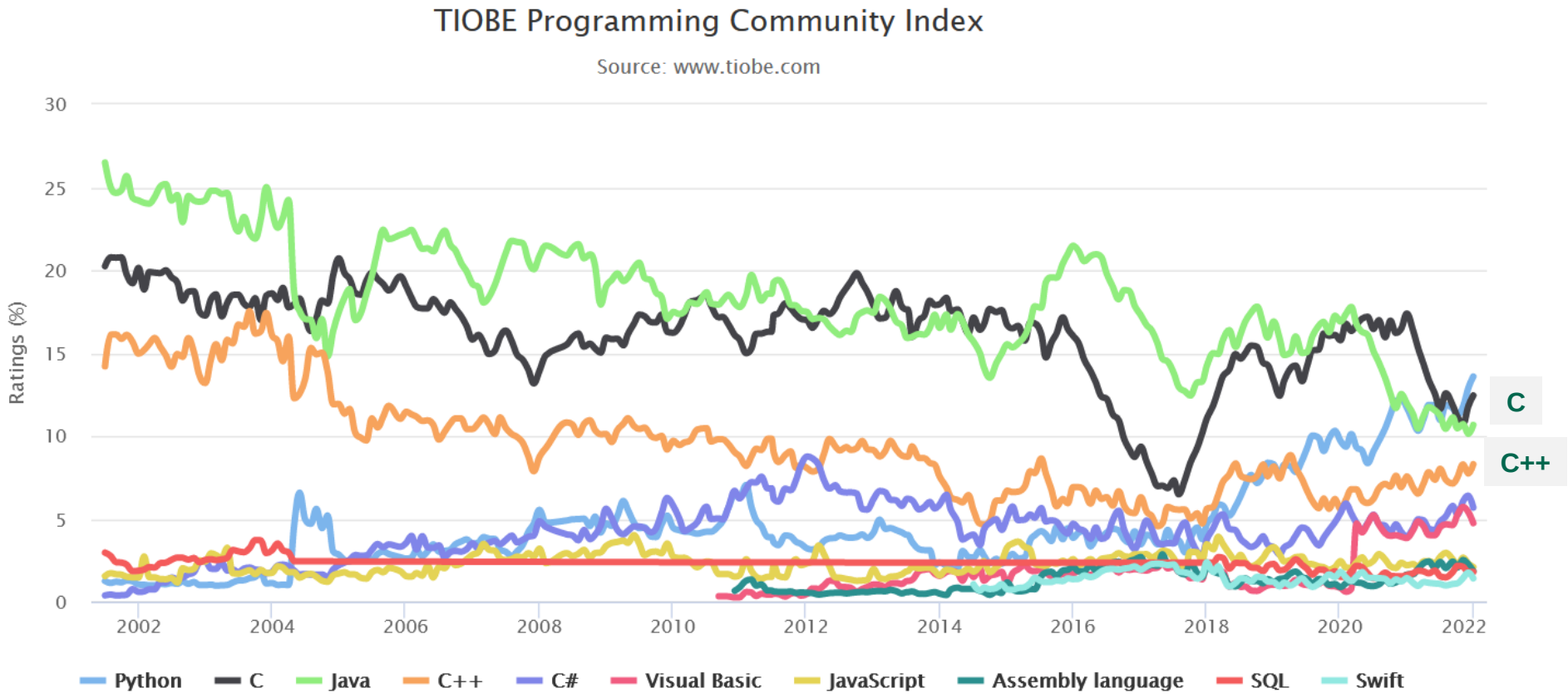
1. La compilation provoque une erreur
2. L'exécution provoque des fuites de mémoire
3. 111
4. 121
5. 222
6. 221
7. 122
8. 212



TIOBE Index

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- Computer languages “popularity”
  - Definition : `http://www.tiobe.com/tiobe-index/programming-languages-definition/`



Teaching Unit Objectives

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **Be able to understand / modify / extend C++ code from a library (QuantLib, ...)**
  - To acquire a certain degree of proficiency in C++
  - Be able to reuse / adapt proven solution mechanisms (design patterns)
- **Be familiar with the computer science principles implemented in the C++ language**
  - Select and justify your implementation choices
  - Application to different domains (finance, image, ML, ...)

▶ To be able to transpose these concepts into other languages...

Examples (1/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- A QuantLib Class (.h)

```
/* -*- mode: c++; tab-width: 4; indent-tabs-mode: nil; c-basic-offset: 4 -*- */

/*
 Copyright (C) 2002, 2003 Ferdinando Ametrano
 Copyright (C) 2003, 2004, 2005, 2006 StatPro Italia srl

 This file is part of QuantLib, a free-software/open-source library
 for financial quantitative analysts and developers - http://quantlib.org/

 QuantLib is free software: you can redistribute it and/or modify it
 under the terms of the QuantLib license.  You should have received a
 copy of the license along with this program; if not, please email
 <quantlib-dev@lists.sf.net>. The license is also available online at
 <http://quantlib.org/license.shtml>.

 This program is distributed in the hope that it will be useful, but WITHOUT
 ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
 FOR A PARTICULAR PURPOSE.  See the license for more details.
*/

/*! \file blackatmvolcurve.hpp
    \brief Black at-the-money (no-smile) volatility curve base class
*/

#ifndef quantlib_black_atm_vol_curve_hpp
#define quantlib_black_atm_vol_curve_hpp

#include <ql/termstructures/voltermstructure.hpp>
#include <ql/patterns/visitor.hpp>

namespace QuantLib {

    //! Black at-the-money (no-smile) volatility curve
    /*! This abstract class defines the interface of concrete
        Black at-the-money (no-smile) volatility curves which will be
        derived from this one.

        Volatilities are assumed to be expressed on an annual basis.
    */
    class BlackAtmVolCurve : public VolatilityTermStructure {
      public:
        /*! \name Constructors
            See the TermStructure documentation for issues regarding
            constructors.
        */
        //@{
        //! default constructor
        /*! \warning term structures initialized by means of this
                     constructor must manage their own reference date
                     by overriding the referenceDate() method.
        */
        BlackAtmVolCurve(BusinessDayConvention bdc = Following,
                         const DayCounter& dc = DayCounter());
        //! initialize with a fixed reference date
        BlackAtmVolCurve(const Date& referenceDate,
                         const Calendar& cal = Calendar(),
                         BusinessDayConvention bdc = Following,
                         const DayCounter& dc = DayCounter());
        //! calculate the reference date based on the global evaluation date
        BlackAtmVolCurve(Natural settlementDays,
                         const Calendar&,
                         BusinessDayConvention bdc = Following,
                         const DayCounter& dc = DayCounter());
        //@}
        virtual ~BlackAtmVolCurve() {}
        //! \name Black at-the-money spot volatility
        //@{
        //! spot at-the-money volatility
        Volatility atmVol(const Period& optionTenor,
                          bool extrapolate = false) const;
        //! spot at-the-money volatility
        Volatility atmVol(const Date& maturity,
                          bool extrapolate = false) const;
        //! spot at-the-money volatility
        Volatility atmVol(Time maturity,
                          bool extrapolate = false) const;
        //! spot at-the-money variance
        Real atmVariance(const Period& optionTenor,
                         bool extrapolate = false) const;
        //! spot at-the-money variance
        Real atmVariance(const Date& maturity,
                         bool extrapolate = false) const;
        //! \name Visitability
        //@{
        virtual void accept(AcyclicVisitor&);
        //@}
      protected:
        /*! \name Calculations

            These methods must be implemented in derived classes to perform
            the actual volatility calculations. When they are called,
            range check has already been performed; therefore, they must
            assume that extrapolation is required.
        */
        //@{
        //! spot at-the-money variance calculation
        virtual Real atmVarianceImpl(Time t) const = 0;
        //! spot at-the-money volatility calculation
        virtual Volatility atmVolImpl(Time t) const = 0;
        //@}
    };

}

#endif
```

Examples (1/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- A QuantLib Class (.cpp)

```
/* -*- mode: c++; tab-width: 4; indent-tabs-mode: nil; c-basic-offset: 4 -*- */

/*
 Copyright (C) 2007 Ferdinando Ametrano

 This file is part of QuantLib, a free-software/open-source library
 for financial quantitative analysts and developers - http://quantlib.org/

 QuantLib is free software: you can redistribute it and/or modify it
 under the terms of the QuantLib license.  You should have received a
 copy of the license along with this program; if not, please email
 <quantlib-dev@lists.sf.net>. The license is also available online at
 <http://quantlib.org/license.shtml>.

 This program is distributed in the hope that it will be useful, but WITHOUT
 ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
 FOR A PARTICULAR PURPOSE.  See the license for more details.
*/

#include <ql/experimental/volatility/blackatmvolcurve.hpp>

namespace QuantLib {

    BlackAtmVolCurve::BlackAtmVolCurve(BusinessDayConvention bdc,
                                       const DayCounter& dc)
    : VolatilityTermStructure(bdc, dc) {}

    BlackAtmVolCurve::BlackAtmVolCurve(const Date& refDate,
                                       const Calendar& cal,
                                       BusinessDayConvention bdc,
                                       const DayCounter& dc)
    : VolatilityTermStructure(refDate, cal, bdc, dc) {}

    BlackAtmVolCurve::BlackAtmVolCurve(Natural settlDays,
                                       const Calendar& cal,
                                       BusinessDayConvention bdc,
                                       const DayCounter& dc)
    : VolatilityTermStructure(settlDays, cal, bdc, dc) {}

    Volatility BlackAtmVolCurve::atmVol(const Period& optionTenor,
                                        bool extrapolate) const {
        Date d = optionDateFromTenor(optionTenor);
        return atmVol(d, extrapolate);
    }

    Volatility BlackAtmVolCurve::atmVol(const Date& d,
                                        bool extrapolate) const {
        Time t = timeFromReference(d);
        return atmVol(t, extrapolate);
    }

    Volatility BlackAtmVolCurve::atmVol(Time t,
                                        bool extrapolate) const {
        checkRange(t, extrapolate);
        return atmVolImpl(t);
    }

    Real BlackAtmVolCurve::atmVariance(const Period& optionTenor,
                                       bool extrapolate) const {
        Date d = optionDateFromTenor(optionTenor);
        return atmVariance(d, extrapolate);
    }

    Real BlackAtmVolCurve::atmVariance(const Date& d,
                                       bool extrapolate) const {
        Time t = timeFromReference(d);
        return atmVariance(t, extrapolate);
    }

    Real BlackAtmVolCurve::atmVariance(Time t,
                                       bool extrapolate) const {
        checkRange(t, extrapolate);
        return atmVarianceImpl(t);
    }

    void BlackAtmVolCurve::accept(AcyclicVisitor& v) {
        Visitor<BlackAtmVolCurve>* v1 =
            dynamic_cast<Visitor<BlackAtmVolCurve>*>(&v);
        if (v1 != 0)
            v1->visit(*this);
        else
            QL_FAIL("not a BlackAtmVolCurve visitor");
    }

}
```

Examples (2/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- "Visitor" Design Pattern

The visitor pattern allows generic algorithms to be implemented without modifying the objects on which they operate and supports different actions for each type of object without the need for dynamic casting.

```
1  class element_concrete_1;
2  class element_concrete_2;
3
4  class visitor
5  {
6    public:
7      virtual void visit(element_concrete_1& el) = 0;
8      virtual void visit(element_concrete_2& el) = 0;
9  };
10
11 class visitor_concrete : public visitor
12 {
13   public:
14     virtual void visit(element_concrete_1& el) override
15     {
16       // Do something with el
17     };
18
19     virtual void visit(element_concrete_2& el) override
20     {
21       // Do something with el
22     };
23 };
24
```

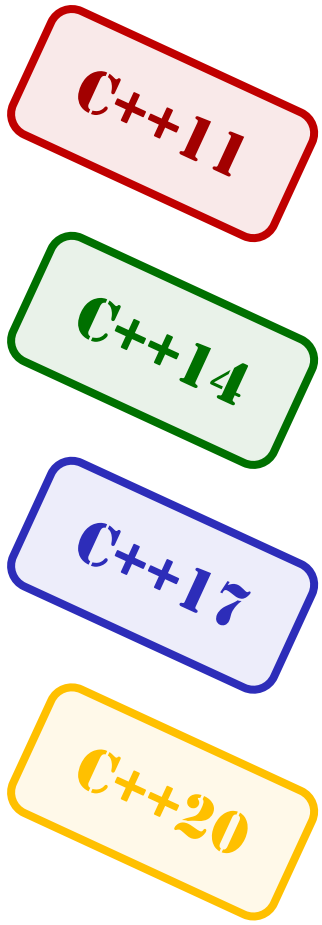
```
26 class element
27 {
28   public:
29     virtual void accept(visitor& v) = 0;
30 };
31
32 class element_concrete_1 : public element
33 {
34   public:
35     virtual void accept(visitor& v) override
36     {
37       v.visit(*this);
38     }
39 };
40
41 class element_concrete_2 : public element
42 {
43   public:
44     virtual void accept(visitor& v) override
45     {
46       v.visit(*this);
47     }
48 };
```

Examples (3/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- "Modern C++" ...

```
template<typename F>
auto async(F&& func) -> std::future<decltype(func())> {
    typedef decltype(func()) result_type;

    auto promise = std::promise<result_type>();
    auto future = promise.get_future();

    std::thread(std::bind([=](std::promise<result_type>& promise)
{
        try  {
            // Note: will not work with std::promise<void>
            //     (needs some meta-template programming )
            promise.set_value(func());
        }
        catch(...)  {
            promise.set_exception(std::current_exception());
        }
    } ,  std::move(promise))).detach();

    return std::move(future);
}
```



The teaching unit's philosophy (1/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- C++ in practice
  - ▶ Be able to code, an object-oriented solution by taking advantage of the concepts & characteristics of the C++ language.
  - Organization (first 5 weeks) :
    - 30' lecture ▶ important notions
    - 30' practice ▶ few targeted questions about the previous notions
    - 15' quiz or synthesis
    - Moodle ▶ **Entry point** for external resources
    - Discord ▶ **Entry point** for **all** your questions / help with debugging
  - Additional support (for the whole TU, **asynchronous**) :
    - Discord ▶ to answer **all** your questions / help with debugging
    - Discord ▶ quiz (1 question per day)

The teaching unit's philosophy (2/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **Good practices in software engineering**
  - ▶ Be able to formalize solution elements to answer a problem (via UML)
  - ▶ Be able to reuse / adapt proven solution mechanisms (design patterns)
  - Organization (over 2 weeks at the beginning of the TU)
    - 30' lecture ▶ important notions
    - 30' practice ▶ few targeted questions about the previous notions
    - 15' quiz or synthesis
  - **Evaluation** :
    - **Analysis of a design pattern** ▶ to understand how it works (static / dynamic), its interest and **code it in C++** on a simple concrete example.

The teaching unit's philosophy (3/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **Solve a concrete problem**
  - From a very basic existing simulation and new "client" needs, add high-level behavior and functionalities
  - Design, C++ implementation, tests and restitution
  - Application
    - Ecosystem simulation
    - Subject available on Moodle
  - Organization (2nd half of TU)
    - Modeling (2 weeks) ▶ UML + explications
    - Implementation (3 weeks) ▶ sources + demo + presentation
  - Evaluation
    - Oral defense (with discussion of the achieved results)
    - Code source analysis

Outline & Evaluation

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- Overview: detailed content on Moodle

Best practices in software engineering
Design & implementation in C++
Case study

Date	Content	Lecture	Practice
03/02	Introduction, C++ (1/9 to 3/9)	2	2
10/02	UML (1/3 & 2/3), C++ (4/9 & 5/9)	2	2
17/02	C++ (6/9 to 8/9), UML (3/3)	2,5	1,5
24/02	C++ (9/9), Design Pattern implementation	0,5	1,5
01/03	**Submission of design pattern deliverables**		
03/03	Presentation of the « Design patterns »	**2**	
03/03	« Case study » Modeling, Feedback		4
10/03	Implementation		2
10/03	Implementation		3
17/03	Integration, tests		3
23/03	**Submission of « Case Study » deliverables**		
24/03	Presentation of the « Case Study »	**2**	1

Targeted skills

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- CG1 – Comprendre et analyser, synthétiser un problème et/ou une situation complexes (N4)

- CG3 – Concevoir et réaliser des systèmes et d organisations (N4)

- CG9 – Communiquer (N3)

- Validation rule of the teaching unit
  - At least 2 validated skills **AND** more than 5 tokens

Conclusion

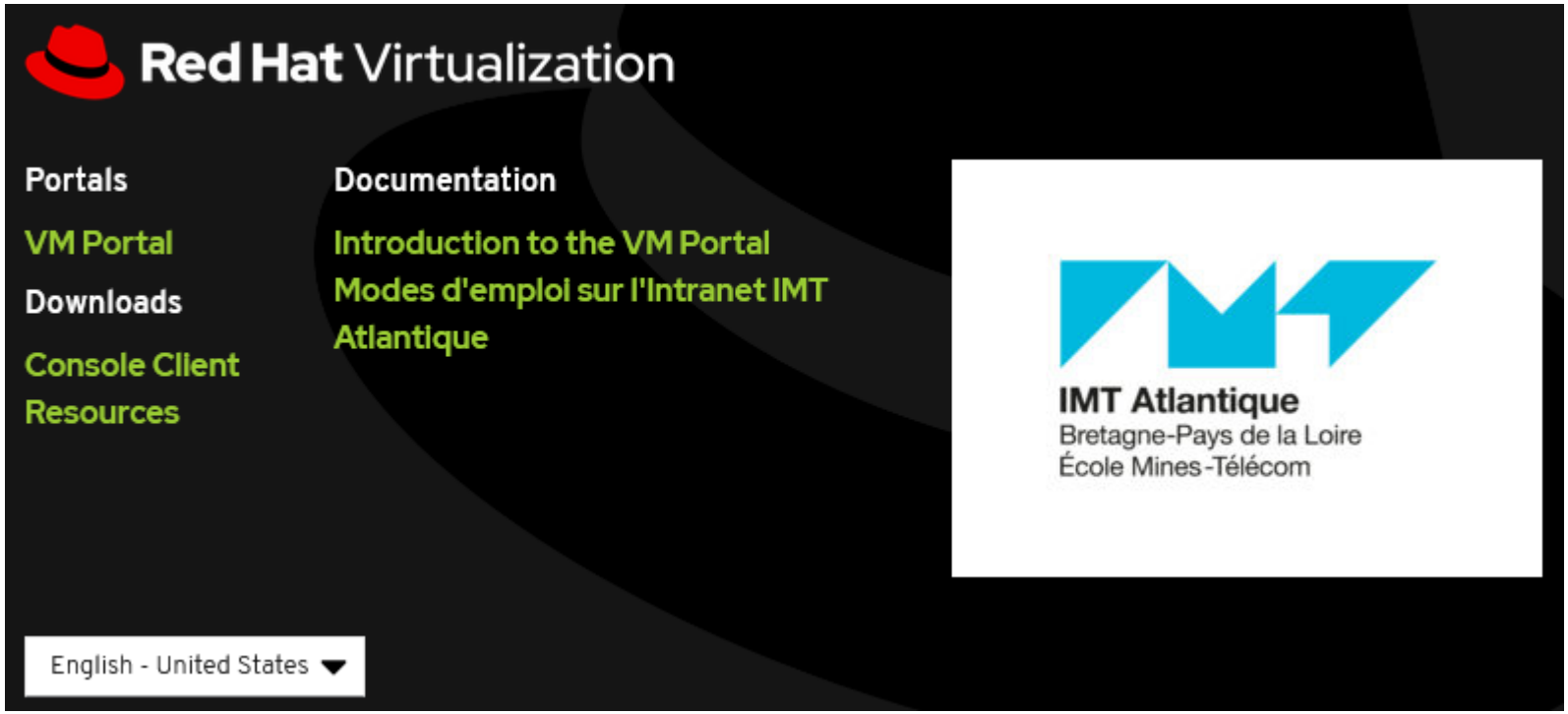
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **Practical information**
  - Work in groups (24 registered ▶ 6 groups of 4)
  - Evaluation grids available on Moodle
- **Communication tools**
  - « Moodle » ▶ reference page + general forums
  - « Discord » ▶ group work (Practical work, exercises, Case Study)
    - 1 group = 1 chat room
  - « Discord » ▶ additional support in asynchronous mode (during the entire course)
  - In case of practical or organizational difficulties...
  - Do not wait
  - E-mail to **Didier.Gueriot@imt-atlantique.fr** AND **Johanne.Vincent@imt-atlantique.fr**

Virtual machines in practice

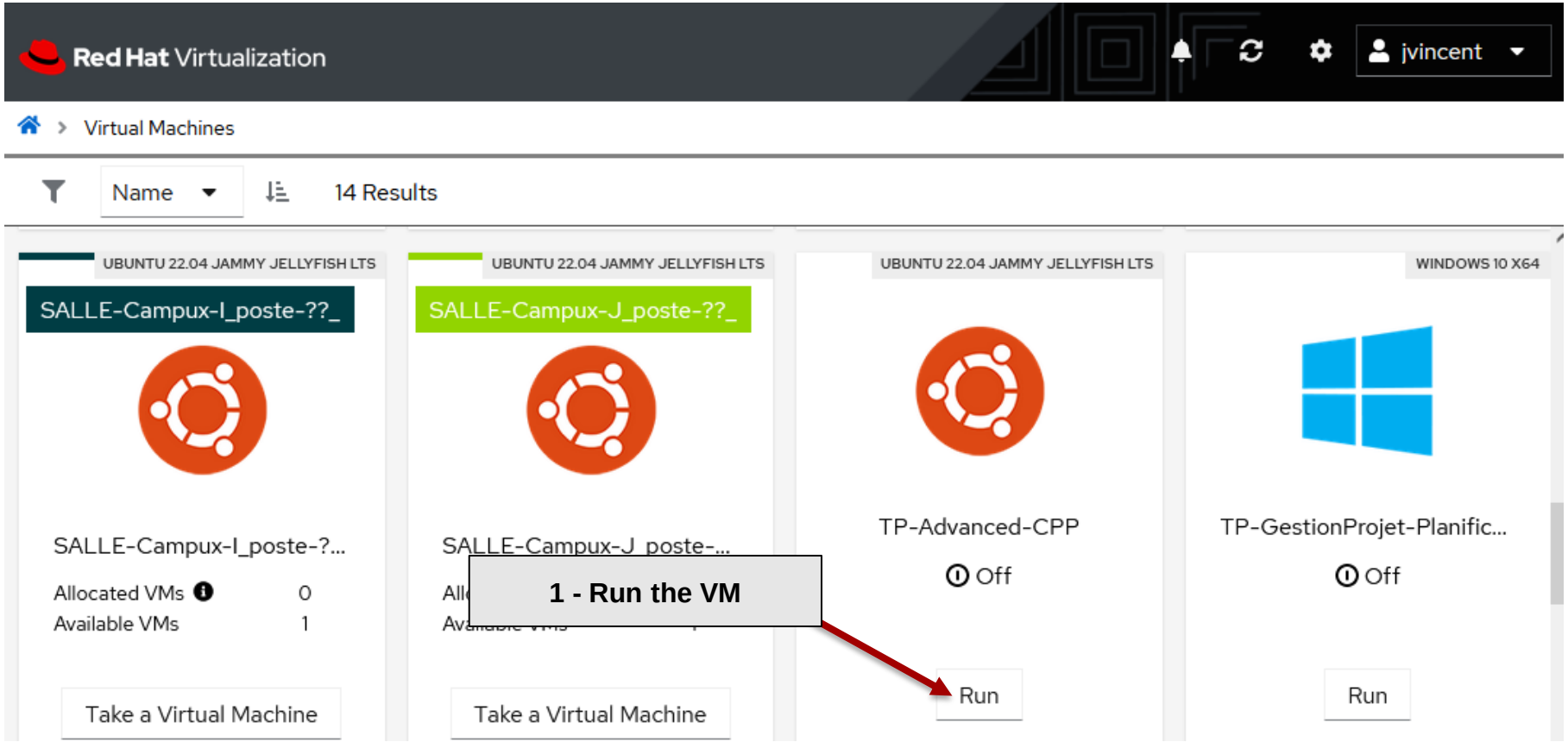
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **Virtual machines**
  - All information available on Moodle
  - VM portal : https://vdi.imt-atlantique.fr/ovirt-engine/



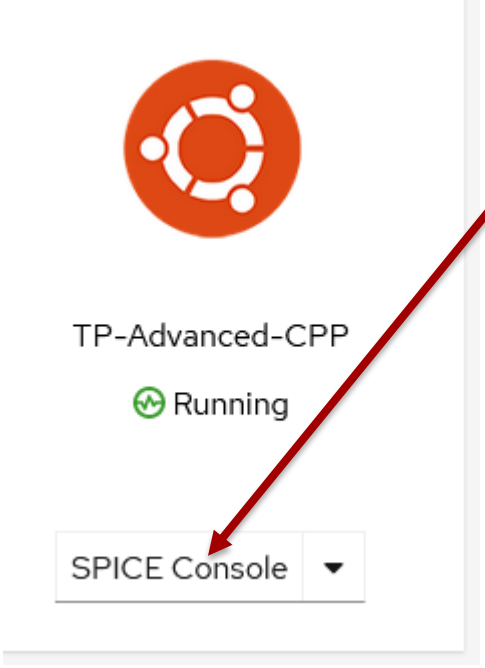
Virtual machines in practice

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom



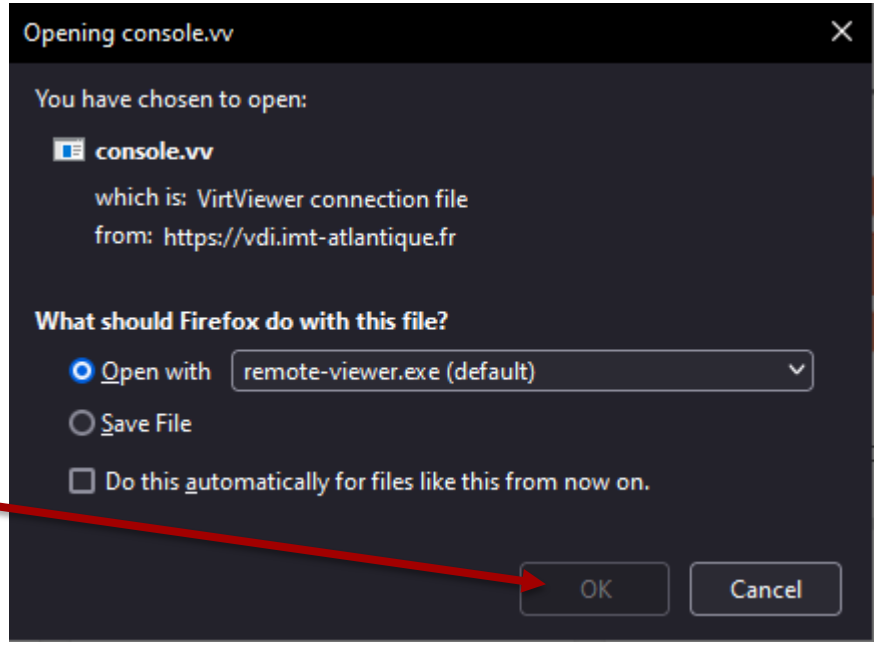
Virtual machines in practice

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom



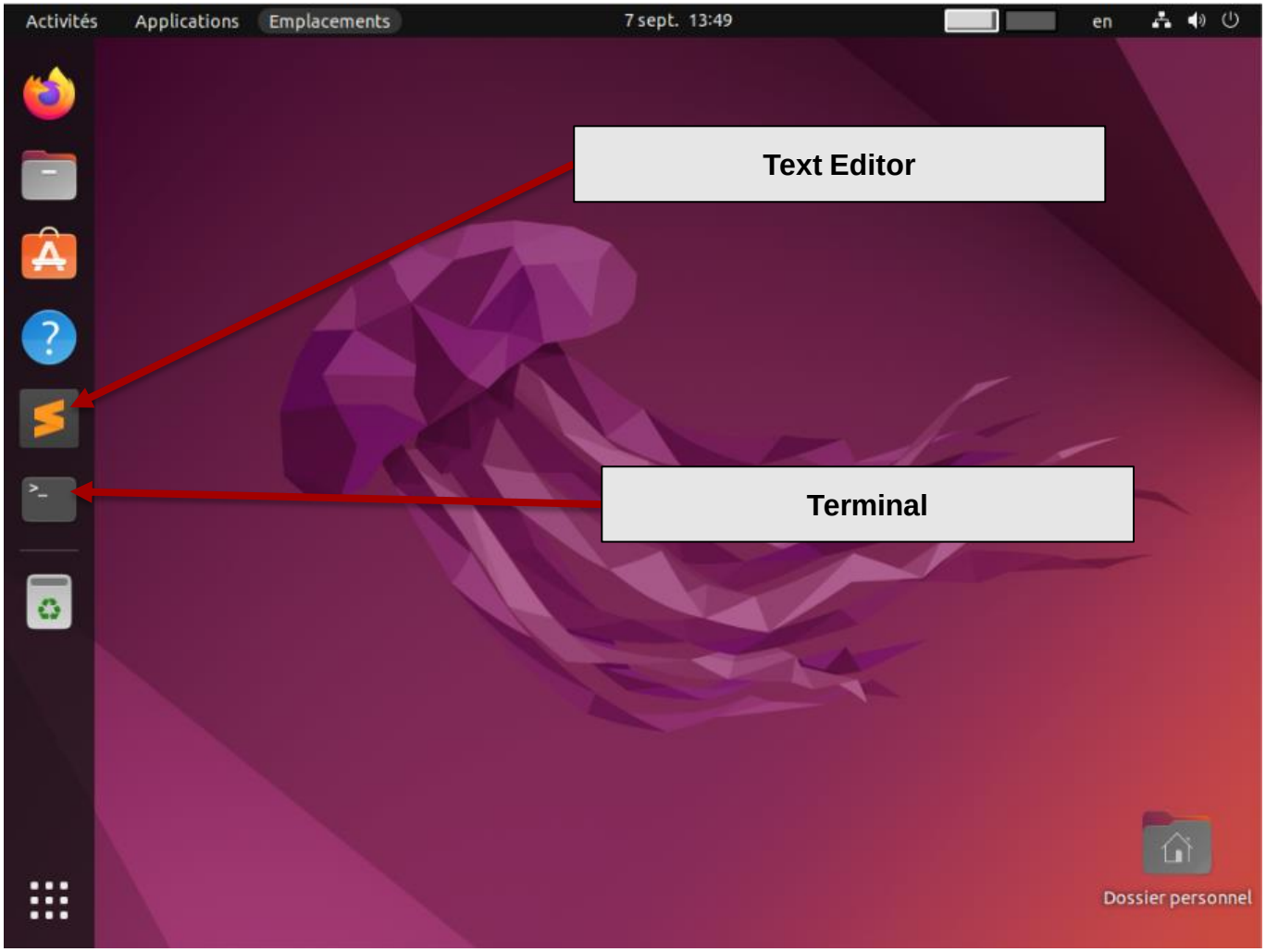
2 – Click on SPICE Console

3 – Open it with the remote-viewer



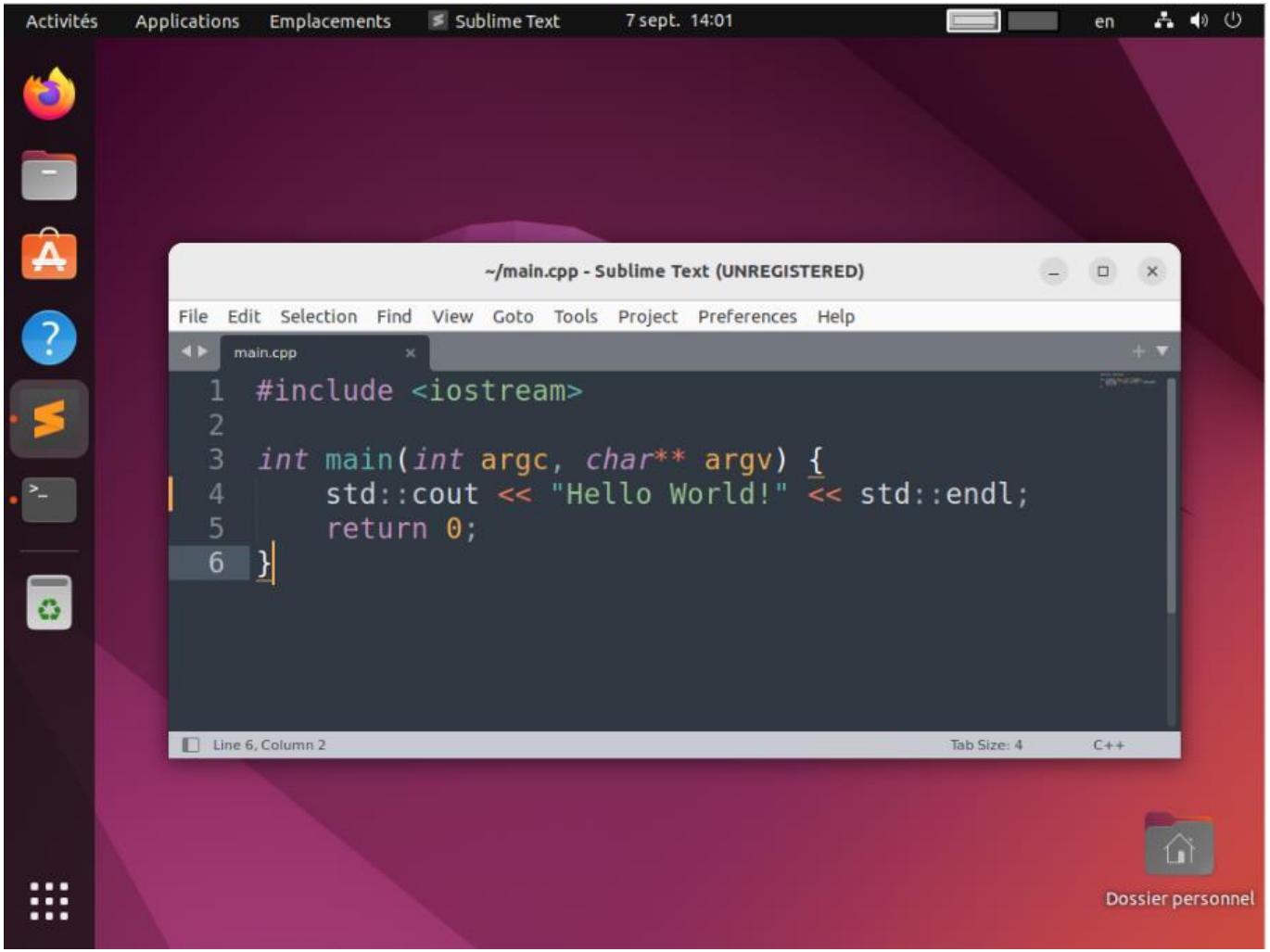
Virtual machines in practice

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom



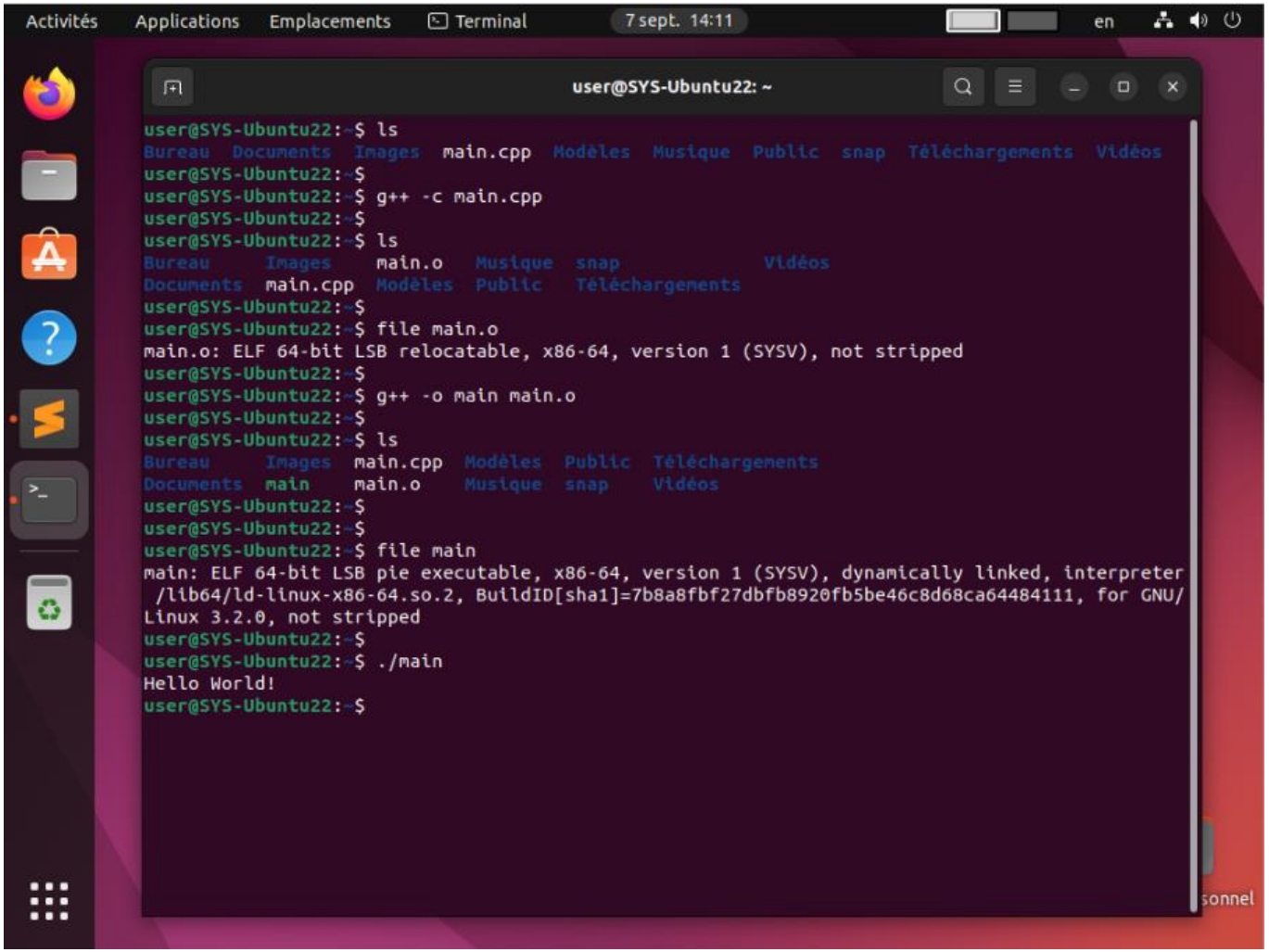
Virtual machines in practice

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom



Virtual machines in practice

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom



Virtual machines in practice

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom