

# Advanced C++ programming: from relevant design to efficient implementation

```
#include <iostream>

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

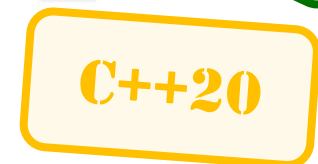
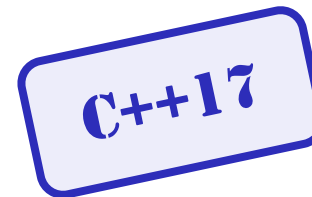
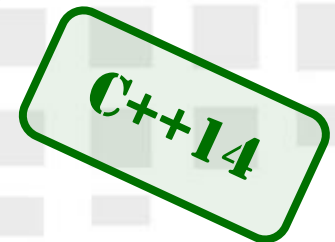
    virtual void foo() { std::cout << "1" ; }
    void bar()      { foo() ; }
};

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



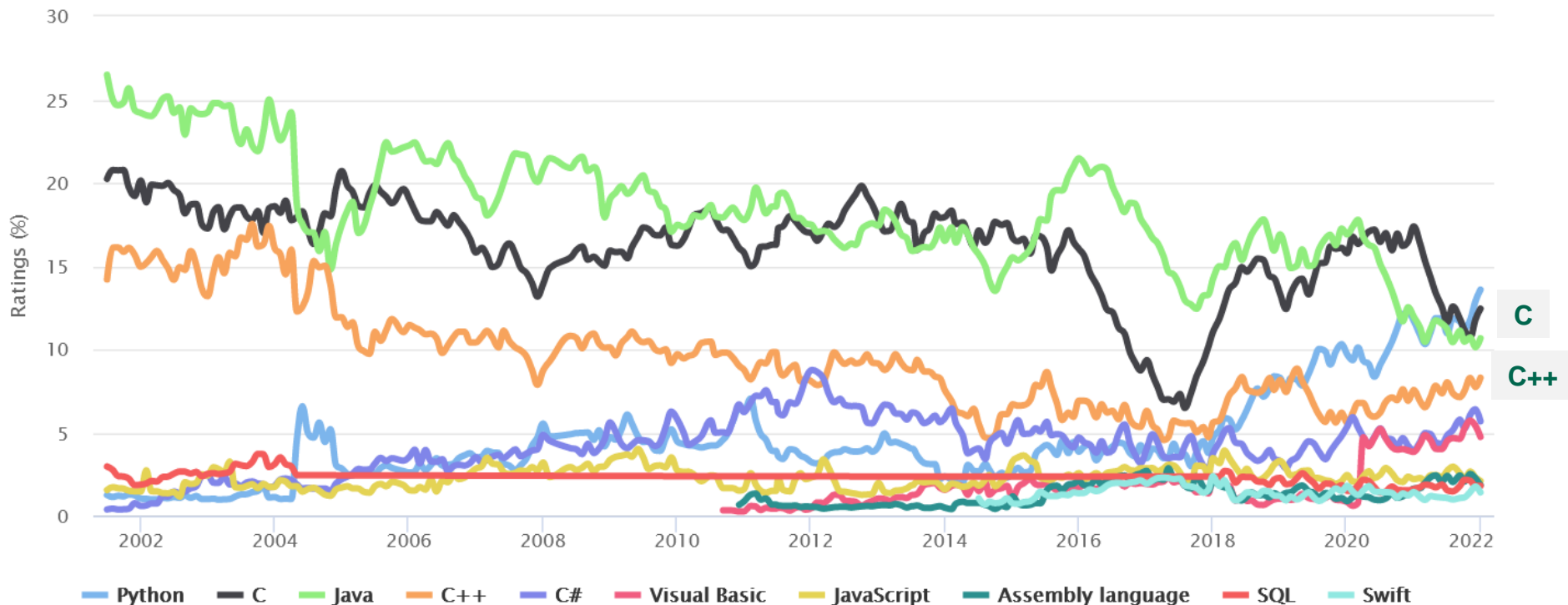
# TIOBE Index

- Computer languages “popularity”

- Definition : <http://www.tiobe.com/tiobe-index/programming-languages-definition/>

## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



# Teaching Unit Objectives

- **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)

## • A QuantLib Class (.h)

```

1  /* -*- mode: c++; tab-width: 4; indent-tabs-mode: nil; c-basic-offset: 4 -*- */
2
3  /*
4   Copyright (C) 2002, 2003 Ferdinando Ametrano
5   Copyright (C) 2003, 2004, 2005, 2006 StatPro Italia srl
6
7   This file is part of QuantLib, a free-software/open-source library
8   for financial quantitative analysts and developers - http://quantlib.org/
9
10  QuantLib is free software: you can redistribute it and/or modify it
11  under the terms of the QuantLib license. You should have received a
12  copy of the license along with this program; if not, please email
13  <quantlib-dev@lists.sf.net>. The license is also available online at
14  <http://quantlib.org/license.shtml>.
15
16  This program is distributed in the hope that it will be useful, but WITHOUT
17  ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
18  FOR A PARTICULAR PURPOSE. See the license for more details.
19  */
20
21  /*! \file blackatmvolcurve.hpp
22   \brief Black at-the-money (no-smile) volatility curve base class
23  */
24
25  #ifndef quantlib_black_atm_vol_curve_hpp
26  #define quantlib_black_atm_vol_curve_hpp
27
28  #include <ql/termstructures/voltermstructure.hpp>
29  #include <ql/patterns/visitor.hpp>
30
31  namespace QuantLib {
32
33      /*! Black at-the-money (no-smile) volatility curve
34       *! This abstract class defines the interface of concrete
35       Black at-the-money (no-smile) volatility curves which will be
36       derived from this one.
37
38       Volatilities are assumed to be expressed on an annual basis.
39       */
40
41      class BlackAtmVolCurve : public VolatilityTermStructure {
42      public:
43          /*! \name Constructors
44           See the TermStructure documentation for issues regarding
45           constructors.
46           */
47          /*! default constructor
48           *! \warning term structures initialized by means of this
49           constructor must manage their own reference date
50           by overriding the referenceDate() method.
51           */
52          BlackAtmVolCurve(BusinessDayConvention bdc = Following,
53                          const DayCounter& dc = DayCounter());
54          /*! initialize with a fixed reference date
55          BlackAtmVolCurve(const Date& referenceDate,
56                          const Calendar& cal = Calendar(),
57                          BusinessDayConvention bdc = Following,
58                          const DayCounter& dc = DayCounter());
59          /*! calculate the reference date based on the global evaluation date
60          BlackAtmVolCurve(Natural settlementDays,
61                          const Calendar&,
62                          BusinessDayConvention bdc = Following,
63                          const DayCounter& dc = DayCounter());
64          */
65          virtual ~BlackAtmVolCurve() {}
66          /*! \name Black at-the-money spot volatility
67          */
68          /*! spot at-the-money volatility
69          Volatility atmVol(const Period& optionTenor,
70                          bool extrapolate = false) const;
71          /*! spot at-the-money volatility
72          Volatility atmVol(const Date& maturity,
73                          bool extrapolate = false) const;
74          /*! spot at-the-money volatility
75          Volatility atmVol(Time maturity,
76                          bool extrapolate = false) const;
77          /*! spot at-the-money variance
78          Real atmVariance(const Period& optionTenor,
79                          bool extrapolate = false) const;
80          /*! spot at-the-money variance
81          Real atmVariance(const Date& maturity,
82                          bool extrapolate = false) const;
83
84          *! \name Visitability
85          */
86          virtual void accept(AcyclicVisitor&);
87          *! \name Calculations
88          These methods must be implemented in derived classes to perform
89          the actual volatility calculations. When they are called,
90          range check has already been performed; therefore, they must
91          assume that extrapolation is required.
92          */
93          virtual Real atmVarianceImpl(Time t) const = 0;
94          virtual Volatility atmVolImpl(Time t) const = 0;
95      };
96  #endif

```



# Examples (1/3)

## • A QuantLib Class (.cpp)

```

1  /* -*- mode: c++; tab-width: 4; indent-tabs-mode: nil; c-basic-offset: 4 -*- */
2
3  /*
4   Copyright (C) 2007 Ferdinando Ametrano
5
6   This file is part of QuantLib, a free-software/open-source library
7   for financial quantitative analysts and developers - http://quantlib.org/
8
9   QuantLib is free software: you can redistribute it and/or modify it
10  under the terms of the QuantLib license. You should have received a
11  copy of the license along with this program; if not, please email
12  <quantlib-dev@lists.sf.net>. The license is also available online at
13  <http://quantlib.org/license.shtml>.
14
15  This program is distributed in the hope that it will be useful, but WITHOUT
16  ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
17  FOR A PARTICULAR PURPOSE. See the license for more details.
18  */
19
20  #include <ql/experimental/volatility/blackatmvolcurve.hpp>
21
22  namespace QuantLib {
23
24      BlackAtmVolCurve::BlackAtmVolCurve(BusinessDayConvention bdc,
25                                         const DayCounter& dc)
26      : VolatilityTermStructure(bdc, dc) {}
27
28      BlackAtmVolCurve::BlackAtmVolCurve(const Date& refDate,
29                                         const Calendar& cal,
30                                         BusinessDayConvention bdc,
31                                         const DayCounter& dc)
32      : VolatilityTermStructure(refDate, cal, bdc, dc) {}
33
34      BlackAtmVolCurve::BlackAtmVolCurve(Natural settlDays,
35                                         const Calendar& cal,
36                                         BusinessDayConvention bdc,
37                                         const DayCounter& dc)
38      : VolatilityTermStructure(settlDays, cal, bdc, dc) {}
39
40      Volatility BlackAtmVolCurve::atmVol(const Period& optionTenor,
41                                         bool extrapolate) const {
42          Date d = optionDateFromTenor(optionTenor);
43          return atmVol(d, extrapolate);
44      }
45  
```

```

46  Volatility BlackAtmVolCurve::atmVol(const Date& d,
47                                     bool extrapolate) const {
48      Time t = timeFromReference(d);
49      return atmVol(t, extrapolate);
50  }
51
52  Volatility BlackAtmVolCurve::atmVol(Time t,
53                                     bool extrapolate) const {
54      checkRange(t, extrapolate);
55      return atmVolImpl(t);
56  }
57
58  Real BlackAtmVolCurve::atmVariance(const Period& optionTenor,
59                                    bool extrapolate) const {
60      Date d = optionDateFromTenor(optionTenor);
61      return atmVariance(d, extrapolate);
62  }
63
64  Real BlackAtmVolCurve::atmVariance(const Date& d,
65                                    bool extrapolate) const {
66      Time t = timeFromReference(d);
67      return atmVariance(t, extrapolate);
68  }
69
70  Real BlackAtmVolCurve::atmVariance(Time t,
71                                    bool extrapolate) const {
72      checkRange(t, extrapolate);
73      return atmVarianceImpl(t);
74  }
75
76  void BlackAtmVolCurve::accept(AcyclicVisitor& v) {
77      Visitor<BlackAtmVolCurve*> v1 =
78          dynamic_cast<Visitor<BlackAtmVolCurve*>>(&v);
79      if (v1 != 0)
80          v1->visit(*this);
81      else
82          QL_FAIL("not a BlackAtmVolCurve visitor");
83  }
84
85  }

```



# Examples (2/3)

## • “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)

- “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);
}
```

C++11

C++14

C++17

C++20



# The teaching unit's philosophy (1/3)

- **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)

- **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)

- **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 (2<sup>nd</sup> 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

## • Overview: detailed content on Moodle

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
<b>01/03</b>	<b>Submission of design pattern deliverables</b>		
<b>03/03</b>	<b>Presentation of the « Design patterns »</b>	<b>2</b>	
03/03	« Case study » Modeling, Feedback		4
10/03	Implementation		2
10/03	Implementation		3
17/03	Integration, tests		3
<b>23/03</b>	<b>Submission of « Case Study » deliverables</b>		
<b>24/03</b>	<b>Presentation of the « Case Study »</b>	<b>2</b>	<b>1</b>







Best practices in software engineering

Design & implementation in C++

Case study



# Targeted skills

- 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

- **Practical information**

- Work in groups (24 registered ► 6 groups of 4)
- Evaluation grids available on Moodle

- **Communication tools**

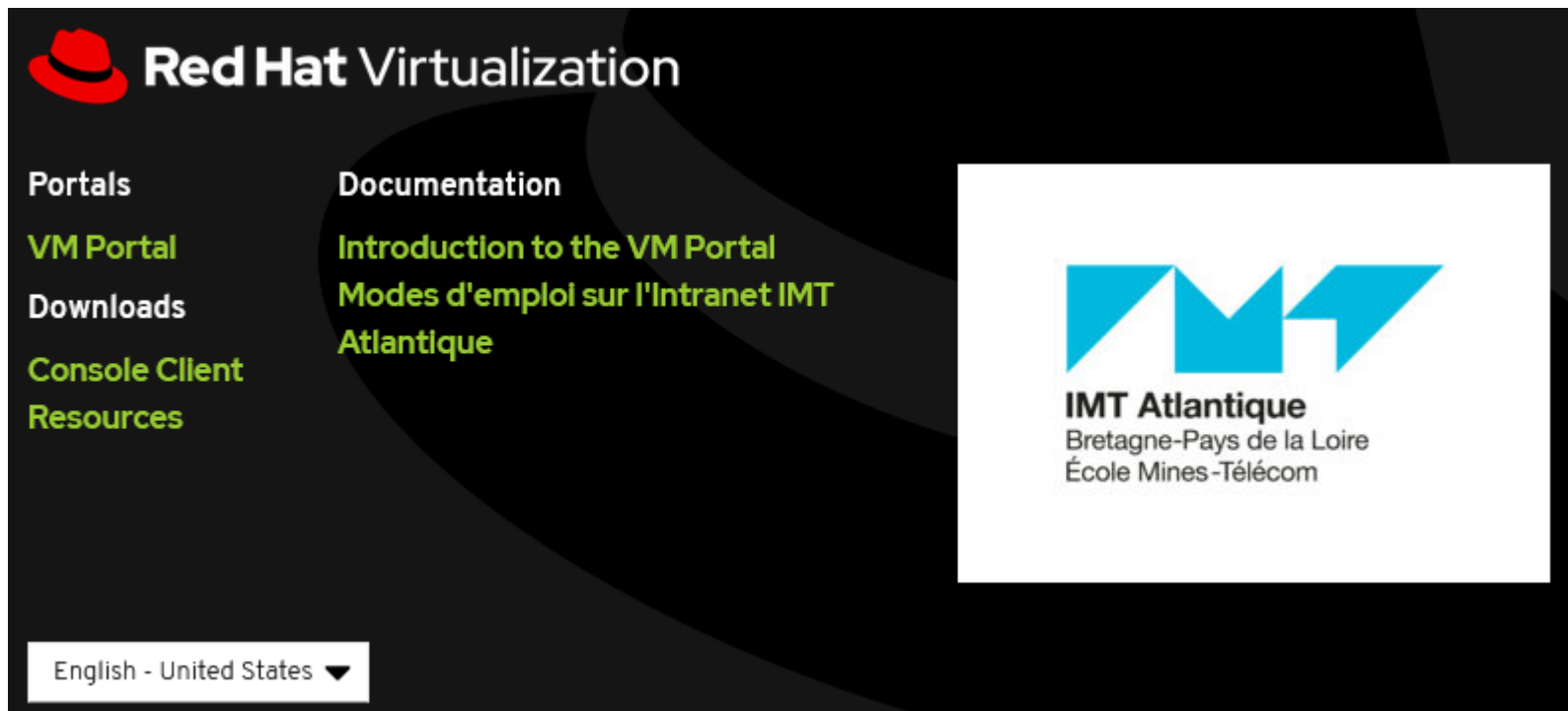
- « Moodle » ► reference page + general forums
- « Discord » ► group work (Practical work, exercises, Case Study)
  - 1 group = 1 chat room
- « Discord » ► additional support in asynchronous mode (during the entire course)
- In case of practical or organizational difficulties...
- Do not wait
- E-mail to **Didier.Gueriot@imt-atlantique.fr** AND **Johanne.Vincent@imt-atlantique.fr**




# Virtual machines in practice

- **Virtual machines**

- All information available on Moodle
- VM portal : <https://vdi.imt-atlantique.fr/ovirt-engine/>



# Virtual machines in practice

 **Red Hat Virtualization**

Virtual Machines

14 Results

Name	Status	OS
SALLE-Campux-I_poste-??_	Off	UBUNTU 22.04 JAMMY JELLYFISH LTS
SALLE-Campux-J_poste-??_	Off	UBUNTU 22.04 JAMMY JELLYFISH LTS
TP-Advanced-CPP	Off	UBUNTU 22.04 JAMMY JELLYFISH LTS
TP-GestionProjet-Planific...	Off	WINDOWS 10 X64

**1 - Run the VM**

Run



# Virtual machines in practice



TP-Advanced-CPP

 Running


SPICE Console ▾

2 – Click on SPICE Console

3 – Open it with the remote-viewer

Opening console.vv

You have chosen to open:

 console.vvwhich is: VirtViewer connection file  
from: <https://vdi.imt-atlantique.fr>

What should Firefox do with this file?

☒ Open with remote-viewer.exe (default) ▾☐ Save File☐ Do this automatically for files like this from now on.

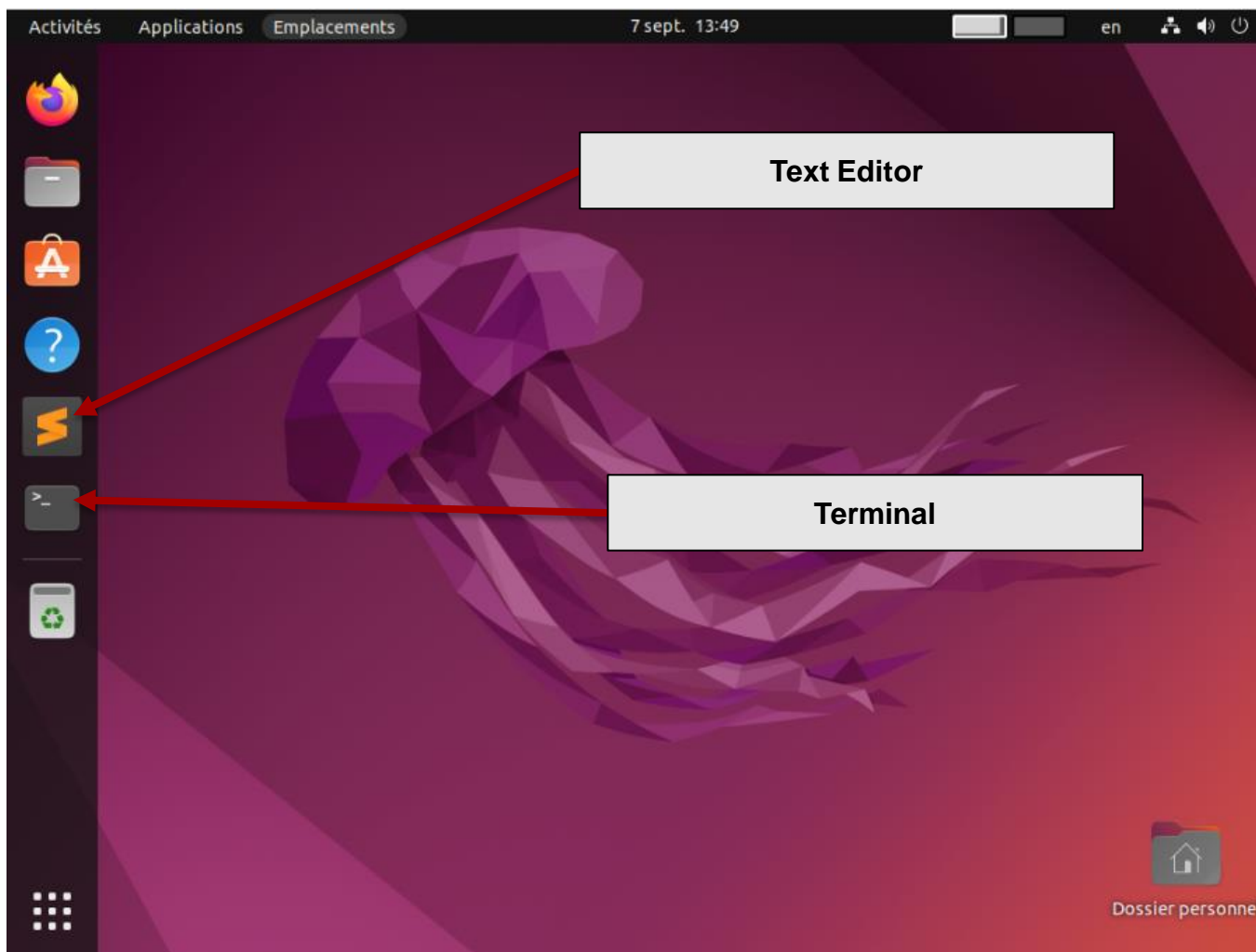
OK

Cancel

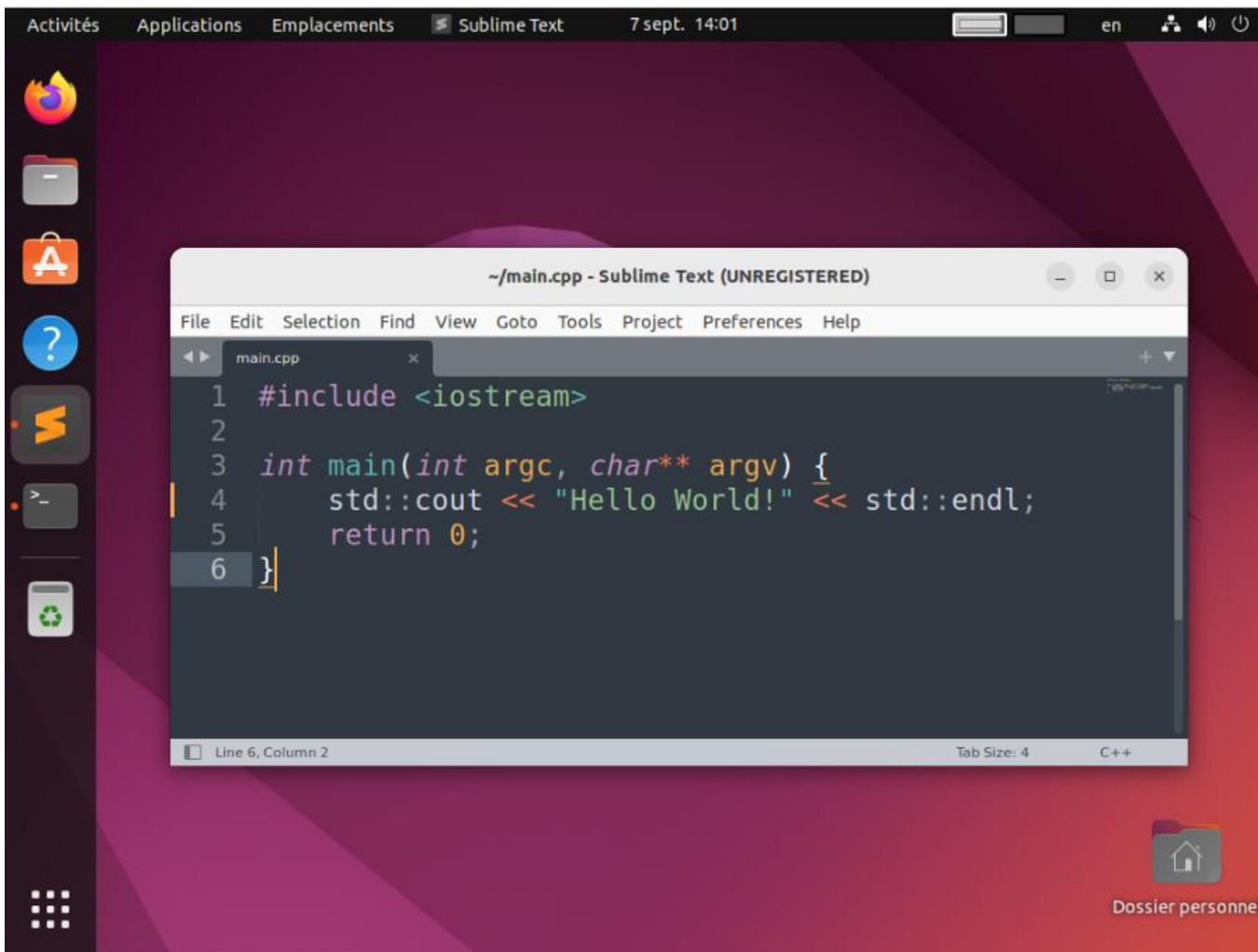




# Virtual machines in practice



# Virtual machines in practice



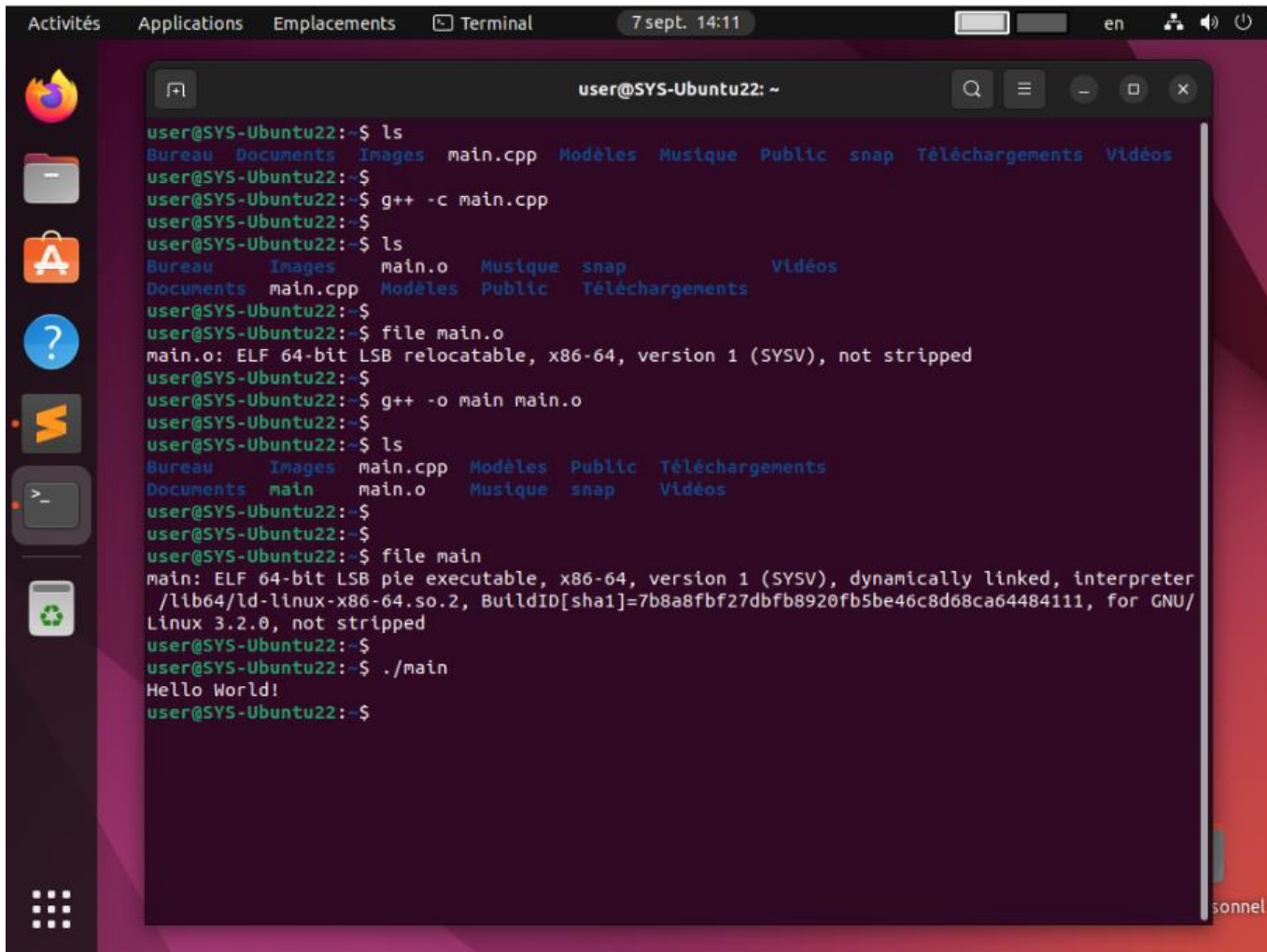
The screenshot shows a Linux desktop environment with a dark theme. The top panel displays the system menu with 'Activités', 'Applications', 'Emplacements', and 'Sublime Text' (which is active). The date and time '7 sept. 14:01' are shown on the right. The left sidebar contains icons for Firefox, Files, App Store, a question mark, a terminal, and a trash can. The main window is a Sublime Text editor titled '~/.main.cpp - Sublime Text (UNREGISTERED)'. It contains a C++ program with the following code:

```
1 #include <iostream>
2
3 int main(int argc, char** argv) {
4     std::cout << "Hello World!" << std::endl;
5     return 0;
6 }
```

The status bar at the bottom of the editor shows 'Line 6, Column 2', 'Tab Size: 4', and 'C++'. In the bottom right corner of the desktop, there is a 'Dossier personnel' icon.



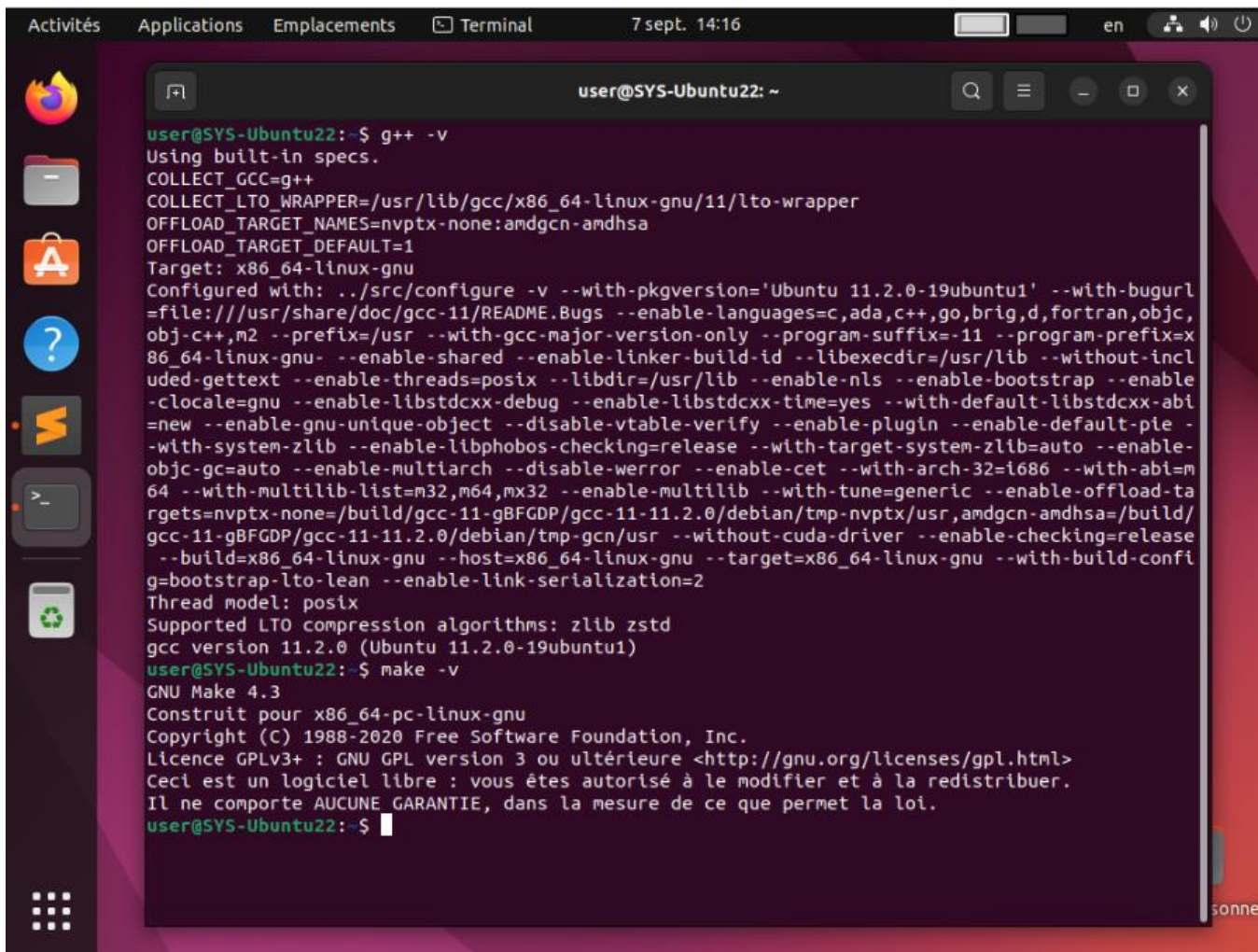
# Virtual machines in practice



```
user@SYS-Ubuntu22: ~  
user@SYS-Ubuntu22:~$ ls  
Bureau Documents Images main.cpp Modèles Musique Public snap Téléchargements Vidéos  
user@SYS-Ubuntu22:~$ g++ -c main.cpp  
user@SYS-Ubuntu22:~$ ls  
Bureau Images main.o Musique snap Vidéos  
Documents main.cpp Modèles Public Téléchargements  
user@SYS-Ubuntu22:~$ file main.o  
main.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripped  
user@SYS-Ubuntu22:~$ g++ -o main main.o  
user@SYS-Ubuntu22:~$ ls  
Bureau Images main.cpp Modèles Public Téléchargements  
Documents main main.o Musique snap Vidéos  
user@SYS-Ubuntu22:~$ file main  
main: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter  
/lib64/ld-linux-x86-64.so.2, BuildID[sha1]=7b8a8fbf27dbfb8920fb5be46c8d68ca64484111, for GNU/  
Linux 3.2.0, not stripped  
user@SYS-Ubuntu22:~$ ./main  
Hello World!  
user@SYS-Ubuntu22:~$
```



# Virtual machines in practice



```

user@SYS-Ubuntu22: ~
user@SYS-Ubuntu22:~$ g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:amdgc-n-amdhsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 11.2.0-19ubuntu1' --with-bugurl=
=file:///usr/share/doc/gcc-11/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,
obj-c++,m2 --prefix=/usr --with-gcc-major-version-only --program-suffix=-11 --program-prefix=x
86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-incl
uded-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-bootstrap --enable
-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi
=new --enable-gnu-unique-object --disable-vtable-verify --enable-plugin --enable-default-pie -
-with-system-zlib --enable-libphobos-checking=release --with-target-system-zlib=auto --enable-
objc-gc=auto --enable-multiarch --disable-werror --enable-cet --with-arch=32=i686 --with-abi=m
64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-ta
rgets=nvptx-none=/build/gcc-11-gBFGDP/gcc-11-11.2.0/debian/tmp-nvptx/usr,amdgc-n-amdhsa=/build/
gcc-11-gBFGDP/gcc-11-11.2.0/debian/tmp-gcn/usr --without-cuda-driver --enable-checking=release
--build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu --with-build-confi
g=bootstrap-lto-lean --enable-link-serialization=2
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 11.2.0 (Ubuntu 11.2.0-19ubuntu1)
user@SYS-Ubuntu22:~$ make -v
GNU Make 4.3
Construit pour x86_64-pc-linux-gnu
Copyright (C) 1988-2020 Free Software Foundation, Inc.
Licence GPLv3+ : GNU GPL version 3 ou ultérieure <http://gnu.org/licenses/gpl.html>
Ceci est un logiciel libre : vous êtes autorisé à le modifier et à la redistribuer.
Il ne comporte AUCUNE GARANTIE, dans la mesure de ce que permet la loi.
user@SYS-Ubuntu22:~$

```

