Basics

March 19, 2020

1 About

This notebook demonstrates features of Cling, Xeus-Cling and Jupyter Notebook.

2 Hello World

[1]:
```
#include <iostream>

std::cout << "Hello World" << std::endl;
```

```
Hello World
```

- No main() necessary
- each statement is in global space
- some statements are forbidden in global space, like function calls ...
- ... but Cling handles such situations and transforms these statements internally

3 Global and Local Variables

[2]:
```
// global variable
int g1 = 1;
```

[3]:
```
// local variable
{
    int l1 = 2;
}
```

[4]:
```
std::cout << l1 << std::endl;
```

```
input_line_11:2:15: error: use of undeclared identifier
'l1'
 std::cout << l1 << std::endl;
              ^


        Interpreter Error:
```

[5]:
```
std::cout << g1 << std::endl;
{
    // hide global variable
    int g1 = 3;
    std::cout << g1 << std::endl;
}
std::cout << g1 << std::endl;
```

```
1
3
1
```

4 Standard C++ Features

[6]:
```
int fd1(int k){
    return k + 2;
}
```

[7]:
```
std::cout << fd1(3) << std::endl;
```

```
5
```

[8]:
```
class Cd1 {
  int a;
  int b;

public:
    Cd1(int a, int b) : a(a), b(b) {}
    int sum(){
        return a + b;
    }
};
```

[9]:
```
Cd1 cd1(4, 7);
std::cout << cd1.sum() << std::endl;
```

```
11
```

5 Non-Linear Program Flow

[15]:
```
std::cout << non_lin_var << std::endl;
```

```
4
```

[13]:
```
++non_lin_var;
```

[11]:
```
int non_lin_var = 3;
```

6 Persistent Memory

[16]:
```
int k = 0;
```

[20]:
```
for (int end = k + 5; k < end; ++k){
    std::cout << k << std::endl;
}
```

```
7
8
9
10
11
```

[19]:
```
k -= 3;
```

7 Template Specialization

[21]:
```
#include <chrono>
constexpr int dim = 512;
```

[22]:
```
float * A = new float[dim * dim];
float * B = new float[dim * dim];
float * C = new float[dim * dim];
```

[23]:
```
for(int i = 0; i < dim; ++i){
    A[i] = static_cast<float>(i);
    B[i] = static_cast<float>(i);
}
```

[24]:
```
void var_matmul(float const * const A, float const * const B, float * const C,␣
 ↪const int dim)
{
    float sum = 0.f;
    for (int i = 0; i < dim; ++i) {
      for (int j = 0; j < dim; ++j) {
        for (int k = 0; k < dim; ++k) {
          sum += A[i * dim + k] * B[k * dim + j];
        }
        C[ i * dim + j] = sum;
        sum = 0.f;
      }
    }
}
```

[25]:
```
template<int dim>
void t_matmul(float const * const A, float const * const B, float * const C)
{
    float sum = 0.f;
    for (int i = 0; i < dim; ++i) {
      for (int j = 0; j < dim; ++j) {
        for (int k = 0; k < dim; ++k) {
          sum += A[i * dim + k] * B[k * dim + j];
        }
        C[ i * dim + j] = sum;
        sum = 0.f;
      }
    }
}
```

[26]:
```
var_matmul(A, B, C, dim);
t_matmul<dim>(A,B,C);
```

[27]:
```
{
    std::chrono::time_point<std::chrono::high_resolution_clock> v_start, v_end,␣
 ↪t_start, t_end;

    v_start = std::chrono::high_resolution_clock::now();
    var_matmul(A, B, C, dim);
    v_end = std::chrono::high_resolution_clock::now();

    t_start = std::chrono::high_resolution_clock::now();
```

```
    t_matmul<dim>(A, B, C);
    t_end = std::chrono::high_resolution_clock::now();

    std::chrono::duration<double> v_diff = v_end - v_start;
    std::chrono::duration<double> t_diff = t_end - t_start;

    std::cout << "var_matmul: " << v_diff.count() << "s" << std::endl
              << "t_matmul: " << t_diff.count() << "s" << std::endl;
}
```

```
var_matmul: 0.540087s
t_matmul: 0.530416s
```

8 Including and Linking

8.1 Preparation: create a shared library

[28]:
```
%%file foo.hpp
#pragma once

namespace foo {
    int bar();
}
```

```
Overwriting foo.hpp
```

[29]:
```
%%file foo.cpp
#include "foo.hpp"

int foo::bar() { return 42; }
```

```
Overwriting foo.cpp
```

[30]:
```
!gcc -shared foo.cpp -o foo.so
```

8.2 Call Functionality of the Library

[31]:
```
foo::bar()
```

```
input_line_36:2:2: error: use of undeclared identifier
'foo'
 foo::bar()
 ^
```

```
Interpreter Error:
```

[32]:
```
#include "foo.hpp"
```

[33]:
```
foo::bar()
```

```
IncrementalExecutor::executeFunction: symbol '_ZN3foo3barEv' unresolved while
linking [cling interface function]!
You are probably missing the definition of foo::bar()
Maybe you need to load the corresponding shared library?
```

```
Interpreter Error:
```

[34]:
```
#pragma cling(load "foo.so")
```

[35]:
```
foo::bar()
```

[35]:
```
42
```

9 REPL Object Representation

[36]:
```
"Hello World"
```

[36]:
```
"Hello World"
```

[37]:
```
int i1 = 3;
```

[38]:
```
i1
```

[38]:
```
3
```

[39]:
```
int fi1(){
    return 42;
}
```

[40]:
```
fi1()
```

[40]:
```
42
```

10 Reflection

- values of variables
- type of a variable (cling kernel only)
- memory address
- enum completion (cling kernel only)
- interpreter environment

[41]:
```
struct S {
    int a = 3;
    float b = 6,f;
} s;
```

[42]:
```
s
```

[42]:
```
@0x7fa5267b9054
```

[43]:
```
s.a
```

[43]:
```
3
```

[44]:
```
#include "cling/Interpreter/Interpreter.h"
```

[45]:
```
gCling->getDefaultOptLevel()
```

[45]:
```
0
```

[46]:
```
gCling->setDefaultOptLevel(3)
```

11 Redefinition

[47]:
```
#include "cling/Interpreter/Interpreter.h"
gCling->allowRedefinition();
gCling->isRedefinitionAllowed();
```

[48]:
```
int func(){
    return 43;
}
```

[49]:
```
func()
```

[49]:
```
43
```

[50]:
```
int func(){
    return 42;
```

```
}
```

[51]:
```
func()
```

[51]:
```
42
```

[52]:
```
class class1 {
    int a = 3;
    int b = 4;
public:
    int func() {
        return a + b;
    }
};
```

[53]:
```
{
    class1 c;
    std::cout << c.func() << std::endl;
}
```

```
7
```

[54]:
```
class class1 {
    int a = 3;
    int b = 40;
public:
    int func() {
        return a + b;
    }
};
```

[55]:
```
{
    class1 c;
    std::cout << c.func() << std::endl;
}
```

```
43
```

12 I/O through Web Elements

[56]:
```
#include <string>
#include <fstream>

#include "xtl/xbase64.hpp"
#include "xeus/xjson.hpp"
```

[57]:
```
void display_image(const std::string filename){
    std::ifstream fin(filename, std::ios::binary);
    std::stringstream buffer;
    buffer << fin.rdbuf();
    // memory objects for output in the web browser

    xeus::xjson mine;

    xeus::get_interpreter().clear_output(true);

    mine["image/png"] = xtl::base64encode(buffer.str());
    xeus::get_interpreter().display_data(
        std::move(mine),
        xeus::xjson::object(),
        xeus::xjson::object());
}
```

[58]:
```
display_image("pictures/conclusion_basics.png");
```

Conclusion

Changed Behavior

- Simplified syntax
- Non-linear flow
- Persistent memory
- Template parameters at runtime
- linking at runtime

New Features

- REPL Object Representation
- Reflection
- Redefinition
- I/O with web elements
- Features of Jupyter Notebook