TensorFlow 内核剖析

TensorFlow Internals

刘光聪

This page is intentionally left blank.

前言

C++ Programming Style 是一份关于 C++ 的代码规范，为 C++ 程序员提供程序设计的参考和建议。

规范总结了我们长期在编码实践中遇到的一些宝贵经验，其中部分规则、原则和建议摘自社区软件设计大师的论文、报告、书籍和博客，并引用了一些典型的源代码。

约定

约定	说明
原则	坚持遵守的指导思想
规则	必须遵守的约定
建议	可以考虑的约定
正例	原则、规则、建议所给出的正确例子
反例	原则、规则、建议所给出的错误例子

表 1 规范约定

宏定义

为了提高代码的表现力，规范中使用了一部分实用的宏定义。为了避免歧义，这里列出了部分较为重要的宏定义供参考和查阅。

示例代码 1 cub/base/Default.h

```
#ifndef GKOQWPRT_1038935_NCVBNMZHJS_8909603
#define GKOQWPRT_1038935_NCVBNMZHJS_8909603

namespace details
{
   template <typename T>
   struct DefaultValue
   {
      static T value()
      {
         return T();
      }
   };

   template <typename T>
   struct DefaultValue<T*>
   {
       static T* value()
       {
           return 0;
       }
   };

   template <typename T>
   struct DefaultValue<const T*>
   {
       static T* value()
       {
           return 0;
       }
```

```
    };

    template <>
    struct DefaultValue<void>
    {
        static void value()
        {
        }
    };
}

#define DEFAULT(type, method)  \
    virtual type method { return ::details::DefaultValue<type>::value(); }

#endif
```

DEFAULT 对于定义空实现的 virtual 函数非常方便。需要注意的是，DEFAULT 所有计算都是发生在编译时，并没有任何运行时成本。

示例代码 2 cub/base/Keywords.h

```
def func():
  return 0
```

示例代码 3 cub/base/Keywords.h

```
#ifndef H16274882_9153_4DB2_A2E2_F23D4CCB9381
#define H16274882_9153_4DB2_A2E2_F23D4CCB9381

#include <cub/base/Config.h>
#include <cub/base/Default.h>

#define ABSTRACT(...) virtual __VA_ARGS__ = 0

#if __SUPPORT_VIRTUAL_OVERRIDE
#   define OVERRIDE(...) virtual __VA_ARGS__ override
#else
#   define OVERRIDE(...) virtual __VA_ARGS__
#endif

#define EXTENDS(...) , ##__VA_ARGS__
#define IMPLEMENTS(...) EXTENDS(__VA_ARGS__)

#endif
```

Config.h 提供了编译器支持 C++11 特性的配置信息。ABSTRACT, OVERRIDE, EXTENDS, IMPLEMENTS 等关键字，使得其他阵营的程序员，也能看懂规范中大部分 C++ 的代码，极大地改善了 C++ 的表现力。

示例代码 4 cub/base/Role.h

```
#ifndef HF95EF112_D6C6_4DB0_8C1A_BE5A6CF8E3F1
#define HF95EF112_D6C6_4DB0_8C1A_BE5A6CF8E3F1
```

```
#include <cub/base/Keywords.h>

namespace details
{
   template <typename T>
   struct Role
   {
      virtual ~Role() {}
   };
}

#define DEFINE_ROLE(type) struct type : ::details::Role<type>

#endif
```

通过 DEFINE_ROLE 的宏定义来实现对接口的定义，可以消除子类对虚拟析构函数的重复实现，消除代码重复。

作者

本规范主要由刘光聪，王博，尉刚强，赵永刚撰写，并由袁英杰审校。由于时间的仓促，规范错误再所难免。如果您发现了错误，或者有更好的意见，请第一时间联系 COCK(C++ Optimal Construction Kit)，我们将非常感激！

This page is intentionally left blank.

目录

This page is intentionally left blank.

1 前言

C++ Programming Style 是一份关于 C++ 的代码规范，为 C++ 程序员提供程序设计的参考和建议。

规范总结了我们长期在编码实践中遇到的一些宝贵经验，其中部分规则、原则和建议摘自社区软件设计大师的论文、报告、书籍和博客，并引用了一些典型的源代码。

约定

约定	说明
原则	坚持遵守的指导思想
规则	必须遵守的约定
建议	可以考虑的约定
正例	原则、规则、建议所给出的正确例子
反例	原则、规则、建议所给出的错误例子

表 1.1 规范约定

宏定义

为了提高代码的表现力，规范中使用了一部分实用的宏定义。为了避免歧义，这里列出了部分较为重要的宏定义供参考和查阅。

示例代码 1-1 cub/base/Default.h

```
#ifndef GKOQWPRT_1038935_NCVBNMZHJS_8909603
#define GKOQWPRT_1038935_NCVBNMZHJS_8909603

namespace details
{
   template <typename T>
   struct DefaultValue
   {
      static T value()
      {
         return T();
      }
   };

   template <typename T>
   struct DefaultValue<T*>
   {
```

```
        static T* value()
        {
            return 0;
        }
    };

    template <typename T>
    struct DefaultValue<const T*>
    {
        static T* value()
        {
            return 0;
        }
    };

    template <>
    struct DefaultValue<void>
    {
       static void value()
       {
       }
    };
}

#define DEFAULT(type, method)  \
    virtual type method { return ::details::DefaultValue<type>::value(); }

#endif
```

DEFAULT 对于定义空实现的 virtual 函数非常方便。需要注意的是, DEFAULT 所有计算都是发生在编译时，并没有任何运行时成本。

示例代码 1-2 cub/base/Keywords.h

```
def func():
  return 0
```

示例代码 1-3 cub/base/Keywords.h

```
#ifndef H16274882_9153_4DB2_A2E2_F23D4CCB9381
#define H16274882_9153_4DB2_A2E2_F23D4CCB9381

#include <cub/base/Config.h>
#include <cub/base/Default.h>

#define ABSTRACT(...) virtual __VA_ARGS__ = 0

#if __SUPPORT_VIRTUAL_OVERRIDE
#   define OVERRIDE(...) virtual __VA_ARGS__ override
#else
#   define OVERRIDE(...) virtual __VA_ARGS__
#endif

#define EXTENDS(...) , ##__VA_ARGS__
#define IMPLEMENTS(...) EXTENDS(__VA_ARGS__)

#endif
```

Config.h 提供了编译器支持 C++11 特性的配置信息。ABSTRACT, OVERRIDE, EXTENDS, IMPLEMENTS 等关键字，使得其他阵营的程序员，也能看懂

规范中大部分 C++ 的代码，极大地改善了 C++ 的表现力。

示例代码 1-4 cub/base/Role.h

```
#ifndef HF95EF112_D6C6_4DB0_8C1A_BE5A6CF8E3F1
#define HF95EF112_D6C6_4DB0_8C1A_BE5A6CF8E3F1

#include <cub/base/Keywords.h>

namespace details
{
   template <typename T>
   struct Role
   {
      virtual ~Role() {}
   };
}

#define DEFINE_ROLE(type) struct type : ::details::Role<type>

#endif
```

通过 DEFINE_ROLE 的宏定义来实现对接口的定义，可以消除子类对虚拟析构函数的重复实现，消除代码重复。

作者

本规范主要由刘光聪，王博，尉刚强，赵永刚撰写，并由袁英杰审校。由于时间的仓促，规范错误再所难免。如果您发现了错误，或者有更好的意见，请第一时间联系 COCK(C++ Optimal Construction Kit)，我们将非常感激！

This page is intentionally left blank.