X 入门问题锦集

文档库地址 https://xlang.link/documents/index.html

目录:

X 的语言特性是怎样的？

xlang 的语言基础特性如同大多数高级语言，如 Java、c#等。

是否有指针？

解答：xlang 的基础类型中没有指针，也没有用于指针的特殊符号.

但除简单类型（byte, short,char, int, long,double,String）以外的所有对象的内部实现都是指针引用的形式。

创建对象需要使用 new，仅定义的对象值初始化为空指针（nilptr）；

代码示例:

```
//只定义， 则 a 的值初始化为空指针(nilptr);
Object a;

//定义并且创建实例， 则 a 指向被创建的对象实例
Object a = new Object();

// 将 a 的值赋予 b， 则 b 也指向与 a 相同的对象实例
Object b = a;
```

X 的语法是怎样的？

解答：X 的语法是依据 C++和 Java 为蓝本开发的，因此与 java 和 C/C++几乎一致。

以下通过几个示例说明对于 C++ 或者 java 语法主要不同点：

1)X 的程序入口点：

```
class xxx{
    static int main(String[] args) {

    }
};

或:

int main(String[] args) {

}
```

说明:

1.X 的入口点可以是类的静态成员方法，也可以是全局方法。

2.X 的入口点函数返回值类型必须为 int

2) 定义数组对象:

一维数组

```
int []data = new int[3];
```

多维数组

```
int [][]data = new int[3][2];
```

说明:

1.X 的数组变量定义中的 [] 必须在变量标识符之前。

3) 类/接口的继承与实现

```
class MyClass : BaseClass{

};
```

说明:

1.X 的类继承关系使用 ：（冒号）。

2. X 中只允许单继承（无论接口或者基类型）。

4) 成员访问控制

```
class A{
  int a = 0;
};
```

说明:

1. **~~X 中所有成员默认 public 方式访问，暂不可设置访问控制.~~**
2. **Xlang 3.2 版本中增加了完整的成员访问控制**

5) 重写方法

```
class MyClass{
        void func()override{
        }
};
```

说明:

1.X 中的强制方法重写在函数定义后加 override 关键字.

6) 闭包捕获

```
...
Object a;
new Thread(){
    void run(){
        a.func();
    }
}.start();
...
```

说明:

1.X 中闭包捕获的对象无需使用 final 关键字.

7）模板定义

```
...
class MyList<_Type>{
    _Type _My;

    void set(_Type o){
        _My = o;
    }
};

...
```

说明：

1.X 定义模板类无需使用 template 关键字.

2.仅支持定义模板类。

其余说明：

1.X 的类定义后面必须有；（分号）结尾。

2.X 语句结尾的 ；（分号）有且只能有一个， 多余的分号会被编译器报错；

语法定义以及基本数据类型细节请参考:http://xlang.vsylab.com/document/html/index.html

如有更多疑问， 请加入 QQ 群 591392649 探讨。

X 目前有哪些组件可用？

解答：X 的类库分为内置的和第三方类库，

X 系统内置了大多数基础的主流类库，以下为所有内置类库的清单
（所有组件均支持跨平台）：

类库	说明
String	字符串类， 内置强大的字符串类是必不可少的
Vector	X 模板库中的动态数组容器
List	X 模板库中的动态链表容器
Map	X 模板库中的基于红黑树实现的 KV 容器
List.Iterator	X 模板库中的动态链表容器迭代器
Map.Iterator	X 模板库中的 KV 容器迭代器
Timer	定时器类
TimerTask	定时器任务类
Thread	线程类，用于多线程处理
Pattern	正则表达式类
Pattern.Result	正则表达式结果类
InetAddress	网络地址类
HttpRequest	Http（s） 处理相关类
Math	数学相关处理类
Stream	流接口
Base64	base64 加密类
Unsi	通用的基于事件驱动模型的网络服务器接口
StreamSocket	TCP 套接字
DgramSocket	UDP 套接字
XDomNode	XML 处理类
JsonObject	Json 对象处理类
JsonArray	Json 数组处理类
ZipArchive	Zip 文件处理类
ZipEntry	Zip 文件处理相关类
ZipFile	Zip 文件处理相关类
Crypto	对 MD5\SHA1\RSA 等支持的加密类
ICrashHandler	X 程序故障通知接口

Exception	异常类
system	编译器提供的系统底层相关接口

X 官方提供的第三方类库(均支持跨平台):

包 / 类库名	说明
FileSystem 包，File 类	提供基础文件系统处理功能的模块
XPlatform	提供多平台间字符集转换的基础接口
FileStream 包 FileInputStream, FileOutputStream 类	提供文件输出输入流
Runtime 包， OSProcess 类	提供操作系统进程操作相关功能
QT	优秀的跨平台界面库. 提供不同操作系统的界面开发支持
Sql ,Sqlite	提供数据库操作基础接口 Sqlite 数据库操作模块
Log	提供日志处理功能和接口

所有第三方类库的源码均可在 https://github.com/ixlang/xlibrarys 找到。
更多的类库会逐步支持.

如有其他模块的开发需求可在 QQ 群里联系群管或者其他开发人员.

X 中如何使用多线程？

解答：X 提供了多线程处理类 Thread，用户可以任意地方使用 Thread 进行创建线程的操作.

代码示例:

```
Thread t = new Thread(){
    void run()override{
        // TODO
    }
};

t.start();

或者使用临时类的方式:

new Thread(){
    void run()override{
        // TODO
    }
}.start();
```

X 中如何进行线程同步？

解答：X 中的任意非简单类型都可以使用同步块语法进行多线程同步功能;
示例:

以下示例为两个线程调用同一个方法 threadProc;

```
Object obj = new Object();


void threadProc(){
    synchronized(obj){
        // 线程安全区域
    }
}

//线程 1
new Thread(){
        void run()override{
            threadProc();
        }
}.start();


//线程 2
new Thread(){
        void run()override{
            threadProc();
        }
}.start();
```

说明：使用 synchronized 语法同一时刻仅有一个线程获取到 obj 对象；
在已获取到对象的线程离开 synchronized 块之前，其他线程都将被阻塞，从而达到同一时刻 synchronized 块内只有一个线程执行操作的目的。
注意：用于同步的对象不能为简单类型或者空指针，即不可为 byte , int , short, long, double, String 等类型.

X 中有没有条件变量/事件通知？

解答：同步块语法可以提供此功能;
示例:

以下示例为两个线程调用同一个方法 threadProc;

```
Object obj = new Object();

//线程 1
new Thread(){
        void run()override{
            synchronized(obj ){
                //进入等待状态 ，并释放 obj 对象以便其他线程获取对象
                obj .wait();
                //wait 的线程被激活， 并重新获取对象 obj
            }
        }
}.start();


//线程 2
new Thread(){
        void run()override{
            synchronized(obj ){
                //通知正在对 obj 进行等待的对象， 激活 wait 的线程
                obj .notify();
            }
        }
}.start();
```

说明：线程调用 wait 之后将释放 obj 对象，以使其他线程有机会获得 obj 对象。

在进行等待的线程被激活之后会立即获取 obj 对象,若此时 obj 对象已被其他线程获得，则进入阻塞状态，直到 obj 的持有者释放对象。

notify 每次只能激活一个等待的线程，而使用 notifyAll 方法可以激活全部的正在等待的线程.

X 中是否具有多态的特性？

解答：X 支持多态， 通过派生类对基类或者接口方法的重写来实现对象以多种状态运作;

```
...
// 用于媒体的播放 、停止、 暂停控制接口
interface MediaController{
    bool play();
};

// 音频播放类 基类为 MediaController
class AudioPlayer : MediaController{
    bool play()override{
        // todo 音频播放
    }
};

// 视频播放类 基类为 AudioPlayer
class VideoPlayer : AudioPlayer {
    bool play()override{
        // todo 视频播放
        return super.stop();
    }
};

MediaController vc = new VideoPlayer();
ac.play();
...
```

X 需要手动释放内存吗？X 的 GC 怎样工作？

解答：X 被设计为无需人工干预的内存自动回收机制，因此无需手动释放内存。

此外,开发者也可以使用 _system_.gc(); 强制进行垃圾清理，但这种做法并不推荐，更好的方式是由系统自动进行控制。

系统自动进行回收有什么好处？

X 的虚拟机在运行时，会对生命周期结束的简单对象立即回收。

而对于生命周期复杂的对象会通过内部的机制（如内存占用、负载等因素）确定最佳回收时间，对于不同生命周期的对象会分别选用最合适的机制来处理，因此不推荐手动的方式干预。

X 能与 C/C++ 或者汇编等语言进行交互吗？

解答: X 可以调用 C/C++ 或者汇编语言编写的动态链接库。
如 windows 系统的 dll ， linux 的 so， 或者 macos 的 dylib。
详情参考文档《X 调用外部 Native 方法》

X 都支持哪些平台？

解答: X 直接运行于操作系统平台依赖虚拟机， 目前已经支持的操作系统平台和版本列表如下:

操作系统/架构	x86	x86_64	armel	armhf	aarch64
Windows 7 ~ Windows10	√	√	x	x	x
Linux	√	√	√	√	√
DARWIN/OSX	¤	√	○	¤	○
Adnroid Native	√	√	√	√	√
linux 类嵌入式设备	○	○	√	√	√

√ ：支持并开放公共版本。
○ ：已支持，但由于开发需求较少，未在公开版本中投放。
¤ ：理论支持，但由于此平台需求所以并未组建对应版本。
x ：不支持。

X 支持生成的可执行文件格式有 PE, ELF, MACHO, 分别含 32 位和 64 位.

Xlang 的优势在哪里？

解答：

1. X 的语法为主流的 C 系语法， 使用上没有任何门槛，java 或者 C++程序员无需在语言上投入成本即可使用，对于不熟悉主流 C 系语言的开发者可以通过 X 更好的学习 Java 或者 C++等语言的语法。

2. X 原生支持跨平台，能生成不同平台上的可执行文件，省去了安装运行环境等一切复杂事务。

3. X 使用了自动的垃圾回收机制，无须担心空指针、野指针等问题，开发过程中能更加专注程序逻辑。

4. X 语言及其开发环境 XStudio 是跨平台的，并且每个平台的运行调试开发都完全一致，避免了跨平台开发时环境不熟悉的尴尬，让你能够更加专注于程序的开发。

5. X 较之现有的跨平台涉及 UI 类应用开发的语言和套件， 有无可比拟的开发效率。

6. X 支持调用操作系统原生态的动态链接库， 如 DLL，SO，Dylib，因此无须担心性能问题，并且可完全利用现有的 C/C++生态支持。

有任何疑问请进入 

QQ 群：591392649，进行交流探讨.