《十分钟内掌握 epl 语言》

前言

epl 的学习需要您有一点点的 c 语言基础，此外最好有一点区块链的基础认知。因为在已知的各种 xel 资料汇集后，我们发现，如果您具备以上的两个知识储备，

那么会将更容易理解和掌握 epl。

如果您去查看各种整理后的资料，会发现各个的知识点和应用的各部分都比较零碎。此教程的作用也基于此考虑，重点把一些关键的、零碎的问题，穿插起来。

即便如此，我们仍然建议您能够每个文档都看一看，细节查一查。这样虽然过程相对是痛苦漫长的，但是您对 epl 应用的想象力，也将更有依据，对前后几个阶段

的应用，也将更能容易地贯穿前后。

最后，不要幻想没有一点的积累就能够在短时间内掌握一门哪怕很简单的语言。我们也只是尝试性的去把 epl 应用的流程讲清楚，尽量说明白。当然，如前所述，

还是强烈建议您先看一看以下的三篇文章（我们准备的有中文版本），相信只要有基础，不用我们的总结，您也能够掌控整个流程。

epl 概述：https://github.com/kyiss/xel-miner/blob/master/epl-language-notes.md

epl 编程篇:https://github.com/kyiss/wiki/blob/master/e-pl-programming.md

epl 发布任务:https://github.com/kyiss/wiki/blob/master/e-pl-submitting-tasks.md

正常人的语速大概是秒三百字，我们尽量使用 10 分钟的讲解时间，在几千字内，让您明白 epl 是什么，xel 有什么，它们怎么用，具体怎么做等等。。。

认识 ePL 和 XEL 网络

xel 网络是什么

想必您应该是了解 xel 网络（以下简称为 xel）是什么的，在此我就不重复了。但是 xel 能解决什么问题，用在什么地方，我想您应该是模糊的、不明确的、缺少引导的，就如同之前的我一样。只是模糊得知道 xel 是一个分布式算力网络，但是工作机制是什么，有什么作用，都不甚了解。

其实也没那么复杂，xel 简单说，让您可以使用连接于 xel 网络的各个的 miner，激励它们提供闲散的算力，然后汇集到一起，供您调用。算力很抽象，不太好理解，所以得到的算力是什么样的呢?

笔者并非专业人员，也未曾与已存在的同类型项目做过对比，只能凭着自己的认识简单与您聊一聊。在这里，您通过 miner 的算力资源得到的是很多的随机数字，网络内的所有 miners 都在努力地产生更多的随机数，用来匹配你想要的答案。

如果您对算力的认知并不局限于只有加法器，乘法器或者代码中过的各种运算符号，那么，您将很好的理解这种力量。

同时，xel 网络中有一些配套的东西，比如 miner，比如 xeline,比如浏览器，比如钱包等等。利用好这些东西，会让您方便地释放出 xel 网络的力量。

epl 是什么

我们的 ePL 就是为了让你更好地去使用这样的算力，可以想象，许多许多的随机数，可能还包含了地球诞生的原始密码。而您要做的只是去定义它们，解码它们，然后通过它们来获取您要的答案。这个答案可以是一个数字，可以是一个什么结果，可以是一个趋势，也可以是仅仅的一个无实际意义的数字。xel 为您提供了无限的可能性。epl 要做的就是利用这么多的可能性，去匹配您的问题，去洞察世界。

两者之间的关系是什么，如何快速上手

在这里，我们来梳理下 xel 网络的各个部分。首先我们有一个主网，然后一个 miner，这两个是我们将要用到的最重要的部分。还有就是 xeline，一款任务

发布和监测的软件，当然您也可以自己通过提供的.py 文件进行任务的发布（到主网），监测和结果获取，xeline 是集成了这些文件，并提供了一个 GUI，方便您的使用。

请记住一点，忘掉所有各部分功能的细节问题，抛开所有这些可能会困惑您的地方，只盯着如何使用这一个目标。而等您掌握了使用的流程，这些困惑，也将迎刃而解。

所以我们以下的内容都将围绕着主网、miner、epl 这三点。在此，我们再把各部分功能明确一下，epl 是用来写程序的语言，程序是用来描述您任务的功能，

以上的所有数据流向，都是通过主网来完成的。

也请您适当的熟悉下钱包，浏览器，xeline 以及 miner 的输出数据等等。

经验分享

笔者曾经很困惑，困惑于 xel 的应用范围，困惑于对 epl 的编程范围等等，现在回头想想，其实这些大部分的困惑都是自己知识储备的局限性所造成的。在此，我强烈建议您能秉着包容的态度，对待新生的 xel。当前，它还如同一个婴孩，还需要不断的迭代，才能成为您心目中的那个算力王者，请多给它一些时间。

另外，有心的您应该也发现了，笔者这里并没有过多的讲解细节问题，更多的是一些认知和空间的介绍。所以请见谅，都是无奈之举。对于一个总结性的内容

来说，必须要帮您建立一个宏观的认识之后，才能快速的把整理好的内容传送于您。就如同武林中的功力传承，您可以一步步的童子功练起，也可以机缘之下的外力打通您的任督二脉，然后拥有无限内力，统领江湖。

最后还是要再次的建议您，请多包容，包容。但这并不是说 xel 不好或者说笔者底气不足，并不是这样，笔者只是担心您看不到它好的一面，抛弃了它。。。

第一分钟

如同学习其他新的语言，您要做的第一步就是搭建它的环境，然后看一看它的运行流程等等。在 xel 中，您大多数时间面对的是 epl 的操作，所以，后文主要是

epl 的讲解。捎带的会提一提一些您可能会注意不到的细节问题。

开始前的第一分钟（希望您能的手速能跟上节奏）

一、请准备好 xeline

下载：https://github.com/xel-software/xeline。下载后可以直接运行，打开界面就是完整的 xeline。

注意：

1、当前使用 testnet 进行测试

2、测试网发布任务是免费的，但是要保证您的地址有测试代币 xel，可以通过水龙头获取测试代币。或者联系管理员。

3、任务发布点击左小角的 new tast，界面有 demo 和从文件运行，您可以使用 demo 测试并观察下运行。

4、任务发布成功后，需要等待一段时间，大概几十秒内，等待新的区块到来，然后就能在任务列表内看到您的新任务。

5、多看，多动，不要担心遇到什么问题。）

二、请准备好 Notepad++

1、或者其他文件编辑软件，当前 epl 还没有自己的编辑器，所以还是需要借助常用的软件。

2、按照 epl 语法写代码，保存为.epl 格式就是编译器可以识别的 epl 文件。

三、下载我们的 find_x.epl 文件和 find_x.js

1、 epl 文件保存桌面，js 文件保存于 C:\Users\您的用户名\AppData\Roaming\Xeline 中

2、 .js 文件是为了让您转换您的结果成为您能识别的数据。其实道理很简单，通用型的 xeline 软件，获取的数据各式各样，所以，需要一个回调文件，.js 格式，从结果中提取您要的那个答案。

3、有兴趣的可以查看下回调文件的内细节。请按要求的目录存放。

搭建环境，各组件部分之间的关系

1、请使用 Notepad++打开 find_x.epl 与 find_x.js 文件，查看下内容。

2、请打开 xeline，设置为测试网络，获取一些测试代币

3、如果可以请运行一个 miner（也可以跳过此项）

4、打开在线钱包（用于观察区块数据，当然这里也是可选项）

这里有必要再讲一下 xel、miner、xeline 之间的关系了， xeline 提交的 epl 文件经过自身编译后转为可执行的 c 文件（有待探讨），miner

执行的是 c 文件的 main 函数部分，xel 主网的 Nodes 执行的是 verify 函数部分。简单来说，miner 获取数据，然后 node(节点)进行数据的校验（也就是 verify 函数内容），校验正确，等区块确认后完成奖励发放。

可能您会疑惑，miner 和 node 分别执行的是什么任务，或者任务的哪一部分。这里也简单讲一讲。首先，所有的 miner 和 node 执行的是同样的程序（程序中都包含了 main 和 verify 函数）。这里有一个前提，就是所有的 miner 使用的节点和私钥都是不同的。

可能您还会问，如果都一样，分布式的效率在哪里，复杂任务的分割表现在哪里。我很理解您的这种疑惑，也试着帮您解答。在 xel 的设计中，制定了这样的任务分割策略，加入当前的节点下有很多的 miner，这些 miner 使用的都是同一个节点地址和私钥。那么，这个任务就会自动的分割成

很多小的任务部分。细节部分，可参考 xel 的白皮书和具体测试。当前我们只针对简单任务做一个介绍。

请记住，它们的数据（获取的结果和奖励）都是通过 xel 网络完成。xel 是基础设施，miner 和 xeline 都是运行在它之上的中间件，配套部分。

另外，node（节点）是维持 xel 网络稳定性的根本。

创建，发布第一个 epl 文件，并了解运行流程

抛开所有杂念，开始我们的第一次任务的发布。

任务发布流程为：打开 xeline-选择左下角的 new task 打开任务选择界面-选择文件方式并选择 find_x.epl 文件-等待文件检测完成-提交

等待新的区块来到后，您的任务就会发送给其他的 miner 去执行了。然后点击任务列表，就能看到执行的任务情况了。

以下有几个细节，使用时候请注意下：

1、保证当前账号有一些测试代币，因为任务的发布等等都需要一些发送费用等等。建议您使用水龙头（10s 间隔），账户保留 100xel 的测试代币。

2、等待文件检测完成，正确的 epl 文件，会跳出所有的元参数（后期 epl 程序结构会讲到，暂时不用理会），这里能看到一些费率。这个在 epl 文件内都是设置过的。直接提交就行了。此时，如果您的账户测试代币余额不足以支付任务的预算消耗，那么，也会出现代币不足的提示。

3、如果提交了一个任务文件没有反应，可以试试先提交一个其他文件，然后再回来重新提交一次（此处后期会优化，目前先这样处理，修改后的文件也会遇到这样的问题。）

4、点击任务发布后，您应该就能在左下角看到当前打开的任务数量和当前网络的一些数据，虽然不全面，但是也能做参考使用。

此时，您已经完成了第一步关键的任务发布流程。请多观察下区块数据、xeline 状态、miner 输出（如果您运行了本地的 miner，在 xeline 上会有本地 miner 的一些数据，在 xeline 的左侧）

经验分享

其实，您完全没必要看这么多内容。简单点，只需要打开文件，保存文件，放置到相应的位置，然后就一路下一步的发布任务就可以了。任务发布后，由于当前测试网的转账费率较高(新版的低费率已经在测试了，后期会更新)，所以，强烈建议您账号内预留多一些的测试代币，100 左右吧。

第一个任务，如果您看了代码就会发现，是一个很简单的方程式，之所以用这个是方便您理解程序的结构、系统运行特点。可能您会认为问题过于简单，无意义。还是之前说的，先想下，再质疑，包容一些。假设，我们有无限的 miner，那么，这个问题的解决速度要高于加法器，这样理解，您就会发现个中奥妙。

具体等待时间，不好确定，也许 50 个区块左右，也许 100 个，也许在您限制的时间内，它不会有结果，但是都请耐心等待。

第二分钟

任务已经发布出去了，我们能做的也只有等待。通过以上的了解，我想您已经大概明白了程序的创建，保存和发布。现在我们看下程序细节。

epl 的是基于 C 语言的衍生，发布任务必须通过检测符合 epl 规则之后，才允许转换为执行的 C 语言文件。同时，epl 内部屏蔽了很多 c 语言可能危害

到 miner 的一些地方，比如文件操作，头文件，宏定义及一些其他循环指令。请理解这一点，如果没有这些限制，主网上的那么多 miner，将不受限制地被您的代码所控制，这是不可想象的。当然，epl 的规则不是一成不变的，后期会根据需要，慢慢的增加一些有用且不会影响到安全的一些功能。

epl 程序结构是什么

目前的 epl 文件中（find_x.epl）可以明显看到，程序的最顶部有一些注释符号"//"，请注意，这是必须要存在的，顶端的注释符是为了告诉 xeline 您的一些初始数据（元数据），比如任务的奖励情况，任务的终止条件等等。这里不做详细讲解，可以在 epl 编程篇了解细节。

元数据目前只是 xeline 需要用到，至于通过主网发布的时候是否需要增加，如何增加，需要您自己去发掘了。

元数据之后的程序中可能会用到的一些全局变量的初始化，请注意，epl 当前的操作只允许使用全局变量，并且不允许使用自定义的变量。笔者写到

这里总有点担忧，所以有必要再说以下，当前版本是禁止使用自定义变量的，至于以后您了解整体后，可以试着去做改进，开源项目，这样的操作都是允许的，所以不要质疑当前代码所能操作的空间，限制了自己。

接上文，此处的全局变量初始化是初始化了一个 1000 个无符号整数型的一维数组 u[]，可能您会质疑，为什么是 u[0]，简单来说，这是程序在其内部定义过的，包括其他的一些变量，程序中也有定义，浮点型等等，给您预留了足够的空间。细节可查看 epl 概述篇。

同全局变量的定义类似，您应该已经发现了另外一些其他的参数，比如 m[0]，此处也是内部已经定义过的变量数组，m[]数组是此处的核心，内部存

存放的我们要用到的伪随机数。m[]数组中元素有 12 个，m[0]-m[9]为每次运行产生的伪随机数，m[10]位迭代次数，m[11]为当前迭代的数值。

您只需知道这些细节，就可以了。

详细的内容，笔者这里再次建议您后面看下 epl 编程篇。这里也就不再多说。

我们看下程序的整体结构：

元数据
变量初始化
main()
verify()

完成的 epl 程序（注：xeline 使用的程序文件需要包含元数据）应该包含以上 4 部分，我们已经讲解了元数据和变量初始化，下面将重点讲解下两个函数)。

程序中您会看到 main(verify());　这里也是允许的，在 epl 概述篇中有讲解过 mian 函数内，需要包含的内容有哪些。简单的任务，比如本案例中的因为使用的是简单的任务，所以这样使用是可以的。复杂的任务或者校验逻辑，还是建议您不要这样使用。

复杂的任务中，可以分别定义 main 函数和 verify 函数的内容，如之前所述，main 函数主要为 miner 执行，verify 主要为节点校验使用。

但是要注意，严格按照 epl 概述篇中的 main 函数编程规格进行（其实也没什么限制）。

verify()函数内部，必须有检验逻辑，此处逻辑用于选定您想要的答案，如果逻辑设计错误或者没有，那么任务将不会有结果（可能会执行，但是

不会有返回值，浪费手续费）。verify 函数也比较好理解，内部的 verify_bty 用于结果的选定，如果得到的结果符合内部的表达式要求，那么就会返回

符合该表达式的当前的伪随机数 m[x]，请记住，返回的是伪随机数，您的表达式需要校验的是这个表达式，然后通过回调函数，处理的也是这个值。

verify_bty(内部为表达式，真或者假)，verify 函数中必须要有这个语句，它是核心部分。

看上去很简单，应用空间很小，实则不然，如果您通过程序技巧去设计逻辑语句，通过全局变量去传递一些关键数据，那么就会发现它的适用范围很大。同时，本示例中，由于是一个小的任务的校验，所以直接使用了简单的表达式，进行结果匹配。

另外一个 verify_pow，也是至关重要的一个语句，verify 中必须包含，简单说是为了匹配您的算力奖励。此处不需要调整，后期您熟悉后，可以适当改变。

所以，对于刚入门的您来说，您需要调整的只有 verify_bty 这一个语句，就能满足您当前简单任务的测试。当然，您也可以试着去做更强大的一些任务，这里没有什么限制。

在您尝试的过程中，可以通过 xeline 提交文件任务的方式，来校验 epl 文件是否符合规则。（如果发现任务提交后没有提示，可以试着选择另外一个其他文件文件，然后退回再次尝试即可，xeline 的一个 bug）

第三分钟

此时，您的任务已经可以在任务列表里看到了，但是具体的数据细节，请看下面部分。

如何查看各部分的数据

首先，打开您的在线钱包，在区块数据部分，您将会看到您地址的一些转账信息。里面包含了转账笔数、转账费用、转账流向等等，这些都是任务执

行时候需要要存留的痕迹，通过这些数据，您就能查看到当前的一些为您提供算力的节点，您的开销和当前网络上在执行任务的一些类似记录等等。

请试着找到更多痕迹，然后告诉我们。

当然，如果您打开的有一个 miner，您将会看到 miner 运行的一些输出信息，任务 ID，目标值和一些其他输出（不要在意那些其他显示）。将会更好地帮助您理解整个系统的运行。

此处是可选项，对于一个任务发布人员来说，只需要一个 xeline 即可。对于那些需要出售算力的朋友来说，只需要一个 miner 即可。

进阶 epl

通过上面的介绍，我们知道了数据的产生，校验和传送过程，这里将会重点讲解一些细节问题。

产生的是什么数据，范围是多少

数组 m[]伪随机数的生成代码参考 epl 编程篇，您可以使用一个 m[0]，也可以使用 m[0]-m[9]的组合，这样每次的数据范围为 32-320 位（二进制）。

如果您熟练掌握后，可以试着使用数组 u[]或者其他初始化的变量，将数组变量根据需要的灵活组合，也是 epl 的特点之一。

每次只能产生一个结果吗

在实际操作中，根据需要，你可以产生多个结果。此处可以参考 epl 进阶篇。

高级应用介绍

如之前所述，通过灵活的变量组合，设计好校验逻辑，应该能解决您的大部分问题。此处后期会补充一些实际案例，供您参考。

第四分钟

要讲的基本都讲清楚了。后面这点时间留给您慢慢梳理一下。

此处你需要四分钟的时间去梳理下整体，把不清楚的问题简单的写出来

第八分钟

也许你是一个幸运儿，也许你有更强大的算力支持，也许你还需要等待

为您工作的 miner 越多，您的结果出现的越早，当然，也需要一点幸运在里面。偶然中的必然，在这里提现的淋漓尽致。

回头看

xel 很好理解，epl 很好掌握，好了，请准备开始您的表演吧。

第十分钟

恭喜你

也许此时您已将得到了一个答案，也许还在运行，不过都没有关系，您当前已经掌握了所有的基础要领。没有执行完毕的任务，可以取消掉，准备开始运行您自己的任务吧。

最后再说点什么

写教程什么的非笔者所擅长，呆坐了两天才把这些碎碎念的东西敲打出来，笔者翻译了很多 xel 的资料，这一次要写一点自己的东西。

过程中，笔者万分感慨，也终于能体会到开发人员的心境 。也为从无到有，中间要做很多基础资料等等，这样默默无闻，埋头苦干的精神所震撼。希望大家能同笔者一样，感受到 xel 的力量。也希望这份力量与您同在。