<https://blog.csdn.net/yiweibin/article/details/5386761>

在我现在的游戏项目中，基本上每天都要代码，各种游戏资源需要更新。而且每次从SVN服务器上更新代码后都要编译好久。另外资源的更新也是一件很麻烦的事情，因为我们的所有游戏资源都是统一放在一个FTP上面，每个版本发布之后都会把最新的游戏资源放在里面。每次从FTP上把好几G的数据更新下来很是费时间。于是我在想能不能写个小程序让这些都自动执行，即能够设定一个时间。例如每天的凌晨从FTP上把资源更新下来，然后再从SVN上把最新的代码更新下来，最后让程序自动编译。每一步都有详细的log记录，第二天上班后就可以取到最新的版本了。

实际我的上述方法就是Daily Build的雏形。Daily Build又称“nightly build”，是将一个软件项目的所有最新代码取出，从头开始编译、链接，用安装软件包将链接好的软件安装好，运行安装后的软件，运行测试软件包对主要功能进行测试（smoke test），发现错误并报告错误的完整过程。我的上述想法只是没有自动安装和测试的步骤。

其实实现起来也不是很难，最简单的方法就是写个批处理，然后把批处理程序挂到控制面板的“Scheduled Tasks”中。这种土办法自己要玩玩还行，真要用到项目中去还必须使用专业的软件工具。我在网上搜了一下，发现一款名叫Visual Build的软件还不错，以后有机会要好好介绍一下。

下面还是稍微介绍下每日构建吧：

在微软软件开发中，每日构建是最重要的过程之一，被称为微软产品开发的“心跳”。简单来看，每天构建系统将整个产品解决方案完整构建一遍，生成的目标文件和安装文件被放置在一个共享位置。接着，安装文件被自动部署到release server上，随后可以自动运行BVT（build verification test），并将所有结果寄送每个team member的信箱。

微软有一套完善的内部系统来完成整个自动化流程，以及流程管理、reporting等工作，而如果我们没有这套系统，也想实现完整的daily build流程，该怎么做呢？

在VS.NET2003时代，IDE可以控制整个方案的构建，但是所有的构建逻辑被IDE控制，对于开发人员来说，整个构建流程就像一个黑箱，很难修改和管理。当然可以使用PreBuildEvent和PostBuildEvent来控制，但是这些event都写在csproj/vbproj文件中，不便于修改，不适于扩展。而且使用IDE来做每日构建的话，要求构建系统本身装有VS.NET，这会带来额外的成本。另一种办法是使用NAnt，通过XML配置文件，来管理构建流程，这会使得整个流程易于修改，可扩展，并且不要求构建系统安装IDE，只需要有framework即可。问题是使用NAnt必须额外写一堆复杂的XML配置文件，足以让很多developer看了头疼。

VS.NET2005中引入了一套全新的构建系统：MSBuild。简单来讲，MSBuild可以直接读取csproj文件，控制csc/vbc等编译器，生成整个方案。实际上，VS2005的IDE自身就是调用MSBuild来完成编译的，这与VS2003有很大的不同。并且由于VS2005的csproj文件服从MSBuild的配置schema，因此我们可以直接使用csproj，稍稍修改一下，就能组织起完整的构建流程了。

虽然这是一个非常简单的过程，但却有非常重要的意义：

1、能最小化集成风险

项目组可能遇到的一个很大的风险是，项目组成员根据不同的系统功能各自开发不同的代码，但是当这些代码集成为一个系统的时候，也许系统完成不了预期的功能。这种风险的发生取决于项目中的这种不兼容性多久才被发现，由于程序界面已经发生了变化，或者系统的主要部分已经被重新设计和重新实现了，相应的排错工作将非常困难和耗时。极端情况下，集成的错误可能回导致项目被取消掉。每日构造和冒烟测试可以使这种集成错误变得非常小，而且便于解决，防止了很多集成问题的产生。

2、能减小产品低质量的风险

       这种风险是和集成不成功、集成出错相关联的。每天对集成的代码做一些少量的冒烟测试，即可杜绝项目中那些基本的质量问题。通过这种方式，使系统达到一种周知的良好状态，维护这样的系统可以防止系统逐步恶化到耗费大量时间排查质量问题的地步。

3、能简单化错误诊断

当系统每天都进行build和测试时，系统任何一天发生的错误都能够变得十分精细，便于排查。比如在17日系统还运行正常，18日就出错了，那么只需要检查这两次build之间的代码变化就可以了。

4、能极大鼓舞项目组的士气

看到产品的不断成长，能够极大的鼓舞项目组的士气，有时甚至不管这个产品到底用来做什么。开发人员可能会为系统显示了一个矩形而感到激动。通过每日构造，产品每天进步一点点，保证项目士气的持续高涨。

进行每日构造和冒烟测试

虽然说这是一个简单枯燥的活，每天进行build，每天进行测试，但也有着一些值得注意的细节：

1、每天坚持

每日构造，最重要的就是“每日”。如Jim McCarthy所说，把每日构造看作是项目的“心跳”，没有“心跳”的话，项目也就死了(Dynamics of Software Development, Microsoft Press, 1995)。Michael Cusumano and Richard W. Selby描述了另外一种隐含的比喻，把每日构造比作项目的“同步脉冲”(Microsoft Secrets, The Free Press, 1995)。 不同开发人员写的代码在他们的“脉冲”之间肯定都会存在“同步”的差异，但是必须有这样一个“同步脉冲”，使得这些代码能够组合为一个整体。当项目组坚持每天把这些不同的“脉冲”组合到一起的时候，开发人员脱离整体的情况就会得到极大程度的杜绝。

有些项目组把这一过程简化为“每周build一次”。这样带来的问题是，某一次build失败后，可能要回溯好几周才能找到原因。如果这种情况发生的话，已经得不到经常build带来的好处了。

2、严格检查每一次build

要保证每一次build的成功，就必须保证build后的结果（也可称为build）是可以正常运行的，如果build不可运行，那么本次build被认为是不成功的，同时应该将修复此次build的工作提高到项目组最高级别来处理。

对于如何衡量一个build，每一个项目组都会定义一些自己的标准，这些标准需要设定一个严格的质量级别来处理那些特别严重的缺陷，同时也需要具有一定的伸缩性来忽略掉那些微不足道的缺陷，一些不适当的关心也许会使整个过程举步为艰。

一个好的build起码应该具备以下条件：

●能够成功编译所有的文件、库，以及其它相关组件；

●能够成功链接所有的文件、库，以及其它相关组件；

●不能存在任何使得系统无法运行或者运行出错的高级别故障；

●当然，必须通过冒烟测试

3、每天进行冒烟测试

冒烟测试应该是对整个系统流程从输入到输出的完整测试。测试不必是面面俱到的，但是应该能够发现系统中较大的问题。冒烟测试应该是足够充分的，通过了冒烟测试的build就可以认为是经过充分测试、足够稳定的。

不进行冒烟测试的build是没有太大价值的。冒烟测试就像一个哨兵，在阻止着产品质量恶化和集成问题的产生，不进行冒烟测试，每日构造可能会变成浪费时间的练习。

冒烟测试必须随着系统的扩充而扩充。最初，冒烟测试可能是非常简单的，比如验证系统是否会打印“Hello World”，随着系统功能的扩充，冒烟测试需要越来越充分。最初的冒烟测试也许只需要几秒钟来执行，逐渐地，测试可能会花费30分钟，1小时，甚至更长。

4、建立一个专门的build小组

    在很多项目组，维护每日构造，并更新冒烟测试用例，会耗费一个人工作的大部分时间。因此在一些大的项目中，这项工作独立成不止一个人来完成的全职工作。比如在 Windows NT 3.0的研发中，就有一个由四个全职人员组成的专门的build小组(Pascal Zachary, Showstopper!, The Free Press, 1994)。

5、为build增加修订，如果这样做有意义的话

    一般开发人员不会每天都经常向系统中快速的增加实际的代码，通常是每隔几天，他们在开发好完成某个功能的一套代码以后，然后集成到整个系统中。

6、规定一些导致build失败的惩罚措施

很多执行每日构造的项目组都会规定一些惩罚措施，来惩罚那些导致build失败的行为。从最开始，项目组成员就清楚的知道，build的正常执行是项目组的头等大事。一个失败的build是项目组的意外，无法成为项目组工作的准则。必须坚持：导致build失败的同事，必须停下手中的工作，首先来解决build失败的问题。如果一个项目组的build经常失败的话，久而久之的，再来谈build的正确性就没有意义了。

有种轻松的惩罚措施，能够突出解决问题的优先性。Some groups give out lollipops to each "sucker" who breaks the build. This developer then has to tape the sucker to his office door until he fixes the problem. 有些项目组会惩罚犯错的同事戴上山羊角，或者向一个项目基金捐献5块钱。

有些项目组对此的惩罚就有点残酷了。微软的开发人员，在一些知名度很高、很重要的产品如Windows NT，Windows 95，Excel等产品后期研发中，被要求随时带着寻呼机，如果你的代码导致build失败的话，即使是凌晨3点钟，也会要求你立即来处理这个问题。

7、即使在压力下也需坚持每日构造和冒烟测试

当项目进度的压力越来越大时，维护每日构造的工作看起来有些浪费时间，但是恰恰相反。在压力之下，开发人员丢掉一些平时的规定，会采用一些设计和实现的捷径，这在平时压力较小的环境下一般时不会用的。代码的review和单元测试也可能会比平时粗心一些，这些代码的状态变化也会比平时快很多。

为防止这种情况的出现，每日构造会坚持相关的规定，让压力下的项目保持在正轨上。代码仍然每天在不断变化，但是构造过程使得这种变化每天都可控。

谁能够从每日构造这种过程中得到好处呢？一些开发人员会抗议说，由于他们的项目太大，每天进行build是没有实际意义的。但是为什么现在最复杂的软件项目组却能够成功的执行每日构造的制度呢？本文首发时，Windows NT包括了560万行代码、分布在4万个源程序文件中，项目组仍然可以坚持每日构造。