> **文档版本**
>
> 2018-9-1 第一版
> 2018-9-6 增加Server搭建说明

一些自定义标识

命令：粗斜体
命令中的自定义参数：绿色
文件路径：粗体
网址：斜体
附加说明：【】

大佬

php,js,html,css,mysql 绍伟哥
py,bat 凯哥

要点先记下

1. 客户端：
   unity c# 打包脚本
   shell unity ios 打包脚本
   bat unity android 打包脚本
   客户端 开发流程也是 bat的 ftp服务
   打包串个流程是 python 3.6
   ssh 跳转 http和web交互
   对于unity的修正 就得分版本 自己写点 oc的源码 比如unity keyboard九宫格不支持一类的
   其他的是 c++的 buildtools 和更新打包工具
2. web 好像是 js css html 和mysql 还有一切框架
3. 备份服务器 是 ftp服务
   客户端 开发流程也是 bat的 ftp服务

相关机器

打包机

10.35.51.41 MacOS
10.35.51.40 MacOS
10.35.51.30 Win64【Server】 web工作机器
10.35.51.37 Win64
10.35.51.38 Win64
打包机密码Ms20150409

Patch虚拟机

10.35.49.163 meteorite meteorite_321

SVN服务器

10.35.49.171 publisher tera

FTP

ftp backup Assetbundle：10.35.51.37
ftp backup Package：10.35.51.37

任务流程

1. Windows打包
2. Android打包
3. IOS打包
4. AutoBuild网站
5. 打包脚本

自动打包网站开发环境

- Linux环境

1. CentOS
2. Httpd
3. MySQL
4. PHP7
5. Laravel5.4

~~- Linux上的安装步骤~~
~~1. LNMP一键安装LAMP~~
~~2. PHP Composer下载Laravel5.4源码，拷贝至Apache vhost下~~
~~3. 执行Laravel的artisan生成App Key（command：php artisan key:generate），使用Laravel数据库迁移建立头像数据库~~

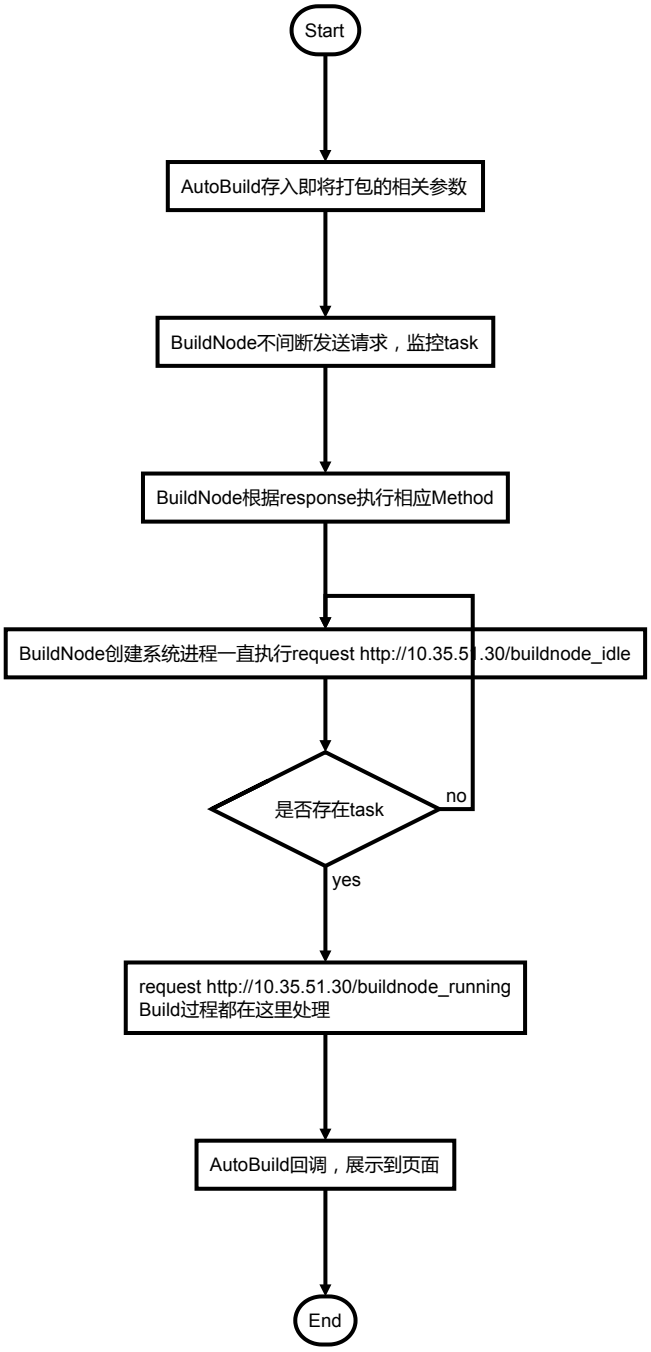
- Windows环境
  WampServer + Laravel

打包过程

简要说明：

每次打包的资源：上一版本的美术资源+当前版本的客户端【原因：美术资源和客户端是两个工程，美术资源每次打包后才会生成ab文件，需要客户端工程】
资源的打包---单独打包

打包web流程：



客户端打包之Windows包

Command Line: ***Unity.exe -projectPath path -batchmode -quit -buildWindows64Player output***
Unity命令参数解析：https://www.cnblogs.com/alongu3d/p/3647485.html

【对于本项目，流程如下】
1.清除旧的文件夹**m1client_path\branch_path\Package**
2.执行命令
unity_path -quit -nographics -projectPath m1client_path\branch_path\UnityProject -project-Tera -executeMethod BuildTools.BuildForWindows -project-Tera -buildTarget Win64 -batchmode m1client_path\branch_path\Package
3.复制客户端文件
将文件夹 **m1client_path\branch_path\GameRes**下的 **\BehaviacData**、**\Configs**、**\Data**、**\Maps**、**\Audio\GeneratedSoundBanks\Windows**复制到**m1client_path\branch_path\Package\GameRes**下与之对应的文件夹中，忽略 **.meta**文件
4.编译lua，执行命令
m1client_path\branch_path\Tools\lua_compiler\cmd_compile.bat
【cmd_compile.bat】 ***脚本所在路径lua.exe cmd_compile.lua GameRes路径 %SOURCE_SUB_DIR% %DEST_BASE_DIR%***
5.将lua编译后的文件从**m1client_path\branch_path\Tools\lua_compiler\Output\lua**复制到**m1client_path\branch_path\Package\GameRes\Lua**

6.下载assetbundles文件
使用TeraUtility.ftp_download，放到**m1client_path\branch_path\Package\GameRes\AssetBundles\Windows**下

客户端打包之Android包

一般流程：Unity通过Android打包插件直接打Android包

【对于本项目，流程如下】
1.清空分支文件夹**m1client_path\branch_path\Package**
2.清空基础资源文件夹res_base_path：**m1client_path\branch_path\UnityProject\Assets\StreamingAssets\res_base**
3.文件列表生成fileListGenerator_path：**m1client_path\branch_path\FileListGenerator.exe**执行命令 ***fileListGenerator_path res_base_path***
FileListGenerator.exe待研究
4.删除一些文件夹
m1client_path\branch_path\UnityProject\Assets\Plugins\iOS
m1client_path\branch_path\UnityProject\Assets\Plugins\x86
m1client_path\branch_path\UnityProject\Assets\Plugins\x86_64
m1client_path\branch_path\UnityProject\Assets\Wwise\Deployment\Plugins\Windows
m1client_path\branch_path\UnityProject\Assets\Wwise\Deployment\Plugins\Mac
m1client_path\branch_path\UnityProject\Assets\Wwise\Deployment\Plugins\iOS
5.TeraUtility.execute命令
unity_path -quit -nographics -projectPath m1client_path\branch_path\UnityProject -project-Tera -buildTarget Android -batchmode -executeMethod BuildTools.BuildForAndroid project_all_name package_path
【TeraUtility.execute说明】扔到Python的subprocess，放到子进程中跑

客户端打包之IOS包

一般流程：Unity通过IOS打包插件生成包，由XCode进行Build，需要IOS APP KEY，XCode设置IOS一些权限，生成IOS可执行文件
https://www.cnblogs.com/mzwl/p/6724034.html
证书相关详细见*https://blog.csdn.net/liuxiongtao_1124/article/details/72373800*
【对于本项目，流程如下】
1.Python paramiko，通过SSH协议连接目标MAC机器
2.TeraUtility.ssh_execute(paramiko.SSHClient(),cmd)===>ssh_client.exec_command(cmd)
3.ssh命令清理正在运行的shell脚本
4.Chmod Permission，workspace读写和执行权限 ***chmod -R 777 path***
5.IOS开发证书导入 ***security unlock -p pwd library_path/Keychains/login.keychain***
6.打包
workspace/tera_autobuild.sh project_name branch_path base_version current_version get_ab_download_path() client_revision project_all_name complatform is_smallpack workspace unity_path
7.关闭ssh

- tera_autobuild.sh：即shell脚本化的MAC上IOS打包过程，流程如上述链接
  BuildTools(冲哥写的Unity IOS打包插件)用来设置打包平台 设置SDK（目前应该是屏蔽状态）和设置基础版本号和更新版本号

客户端资源打包

目的：生成ab文件
unity5打包机制下，一种资源打ab和资源管理的方案 ***https://www.cnblogs.com/Tearix/p/6941650.html***

【本项目的美术资源打包，流程如下】
导出路径：**m1res4build_path\branch_path\TERAMobile\Export\AssetBundles\platform_type**
1.如果目标路径存在，就svn.cleanup()【***svn cleanup %s --username%s --password %s***】,svn.revert()【***svn revert -R %s --username %s --password %s***】
如果不存在svn.checkout(cur_version)【***svn co %s -r %s %s --username %s --password %s***】美术资源到**m1res4build_path\branch_path**,svn.revert()确保资源版本一致
2.如果基础版本==当前版本,TeraUtility.execute命令
unity_path -quit -nographics -projectPath m1res4build_path\branch_path\TERAMobile -executeMethod BuildScript.AutoBuildBasicAssetBundle4platform_type -buildTarget build_target -batchmode check_autobuild_timestamp
如果基础版本!=当前版本，TeraUtility.ftp_download下载ab资源到**m1res4build_path\branch_path\TERAMobile\Export\AssetBundles\platform_type**文件夹，TeraUtility.execute命令
unity_path -quit -nographics -projectPath m1res4build_path\branch_path\TERAMobile -executeMethod BuildScript.AutoBuildUpdateBasicAssetBundle4platform_type -buildTarget build_target -batchmode check_autobuild_timestamp
3.判断是否存在**tera_workspace_path/___*timestamp*【TeraUtility.get_timestamp()】** 文件夹是否存在，存在(表示Build成功了)则移除，不存在则TeraUtility.failed_exit()

AutoBuild网站解析

1. 网站前端框架
   bootstrap 页面框架
   Laravel PHP开发框架
   JQuery js框架
2. Laravel源码目录结构含义：*https://blog.csdn.net/u014665013/article/details/78008511*
   Model层【数据库表的映射实例】：**app/**
   Control层【主要逻辑】：**app/Http/Controllers/**
   View层【页面展示，.blade.php】：**resources/views/**
   Route【地址和逻辑的对应关系】：**routes/**
   Laravel基础服务启动后，请求传给路由（墙：**app\Http\Kernel.php** ）
3. Build过程
   1.【view.button】build_confirm
   2.【param】build_url:
   buildcmd_start?svn_name=svn_name&build_type=build_target&svnRevClient=client_svnRev&svnRevArt=art_svnRev&platform=platforms&v1=v1&v2=v2&v3=v3&xer=xer&bidbid
   3.【control】MainCtrl.buildcmd3
   4.【model】buildtask（save in database ）
   5.【route】workstation
   6.【control】MainCtrl.workstation2
   7.【外部】参数存入数据库后，BuildNode.exe发送请求
   8.【control】BuildNodeCtrl.buildnode_idle、BuildNodeCtrl.buildnode_running、BuildNodeCtrl.buildnode_running_log
   9.【回调】结果反馈到control和view

BuildNode工程--------脚本调用过程

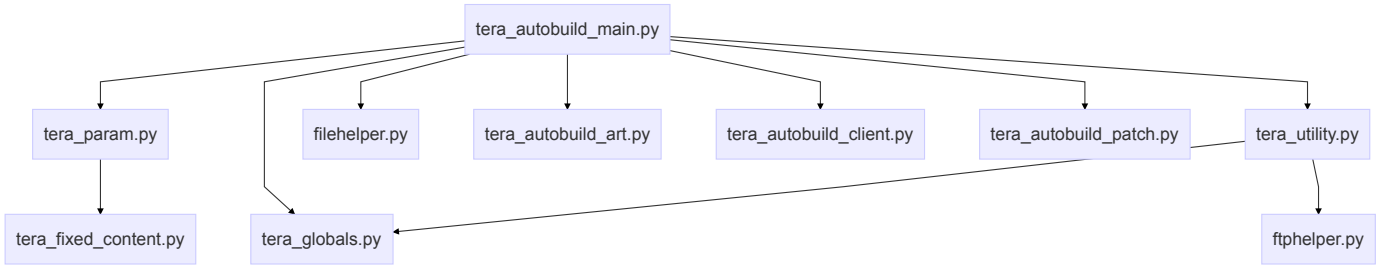
1. 文件路径及URL
   python_path = @"**X:/Python_Env/Anaconda3/python.exe**";
   python_build_script_path = @"**X:/Python_Workspace/tera_autobuild_main.py**";
   python_upload_log_script_path = @"**X:/Python_Workspace/tera_post_log.py**";
   build_log_path = @"**X:/Python_Workspace/out.log**";
   killbat = @"**X:/Python_Workspace/taskkill_before.bat**";
   python_workspace = @"**X:/Python_Workspace**";
   kIdleURL = ***http://10.35.51.30/buildnode_idle***
   kBuildURL = ***http://10.35.51.30/buildnode_running***
   kUploadLogURL = ***http://10.35.51.30/buildnode_running_log***
2. 状态码
   // 1: noting, keep alive
   // 2: dispatch build task, change to running
   // 3: inform build task finish, change to idle
   // 4: stop build task
   // 5: stop build node
   // 6: status change to build, for restart
   // 7: log upload error
   // 254: paramter error, close
   // 255: unregistered, close
   根据状态码开启对应的处理进程
3. 单线程、Http访问
   1.IdleProc() 请求 ***http://10.35.51.30/buildnode_idle*** 返回状态码
   2.BuildProc() 请求 ***http://10.35.51.30/buildnode_running*** 返回状态码
   3.LogProc() 请求 ***http://10.35.51.30/buildnode_running_log***
   【执行脚本】**X:/Python_Workspace/tera_post_log.py X:/Python_Workspace/out.log**
   4.根据1和2状态码选择对应流程

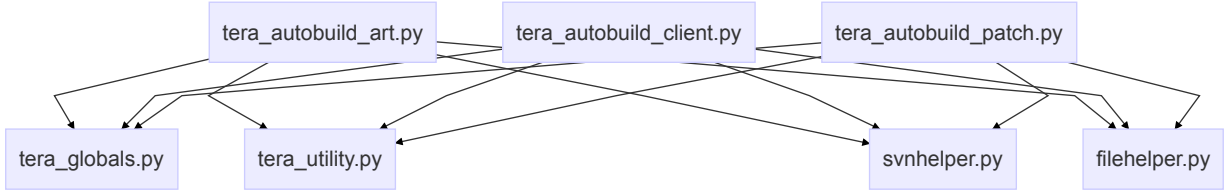
Code	Method	执行脚本
1	break	
2	ToBuild(jsonData)	**X:/Python_Workspace/taskkill_before.bat、X:/Python_Workspace/tera_autobuild_main.py**
3	ToIdle()	

Code	Method	执行脚本
4	StopBuild()	**cmd /c taskkill /F /IM python.exe[Unity.exe][svn.exe]**
5	Exit(5)	
6	SyncBuildStatus((int)jsonData["build_id"])	
254	Exit(254)	
255	Exit(255)	

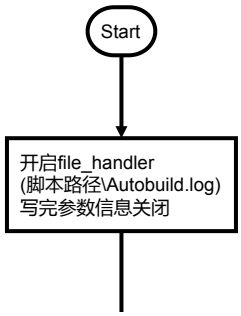
脚本解析

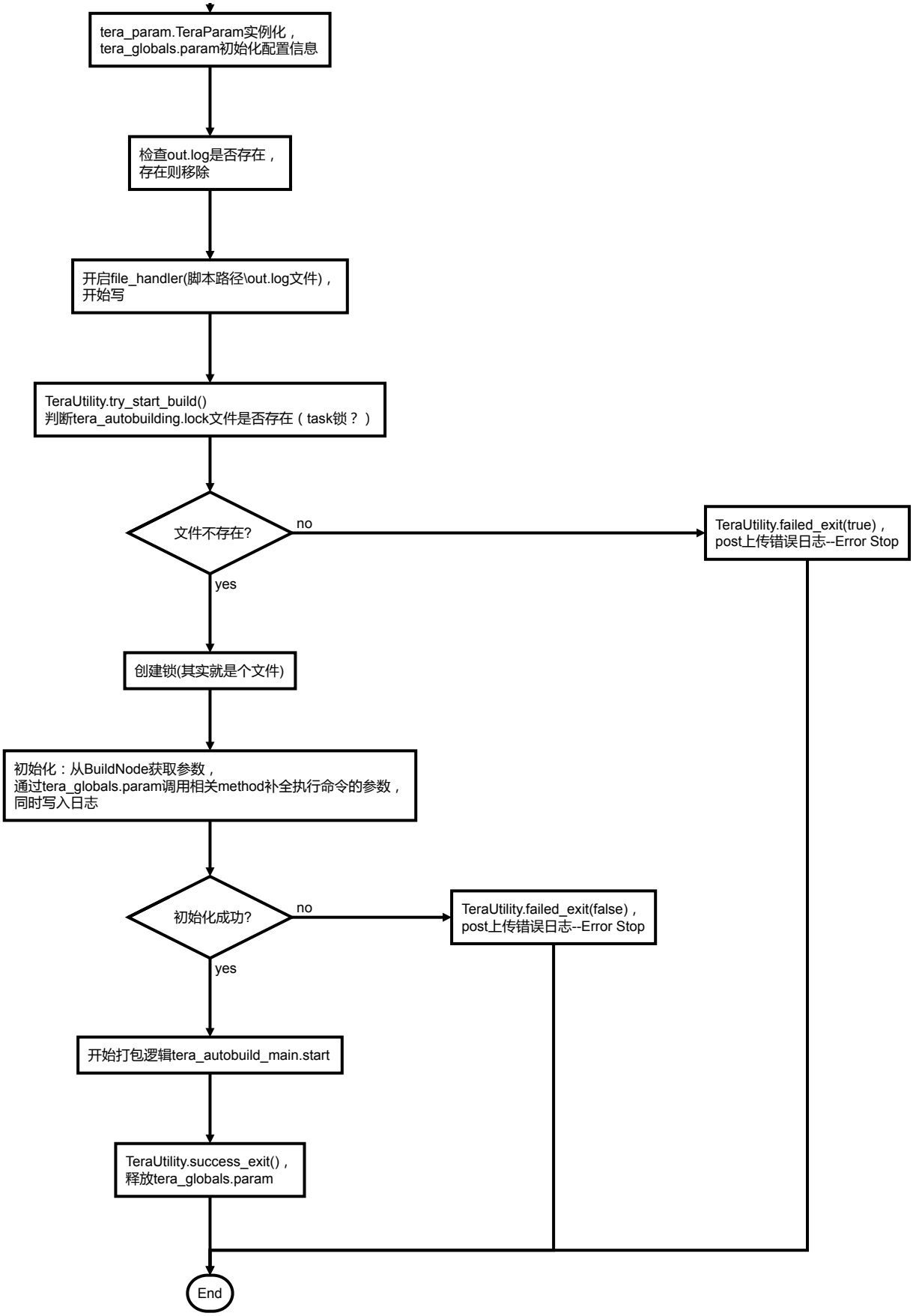
1. 位置：**\Python_Workspace**
   配置：
2. 脚本作用
   filehelper.py_：文件和文件夹处理相关，创建、删除等等
   ftphelper.py_:FTP相关的Method
   PermissionFix.py_
   svnhelper.py_SVN相关的Method
   taskkill_before.bat
   tera_autobuild.sh:IOS打包shell脚本
   tera_autobuild_art.py：美术资源打包（ ）
   tera_autobuild_client.py：客户端打包（Android、IOS、Windows包）
   tera_autobuild_main.py：自动打包脚本
   tera_autobuild_patch.py：分支打包相关method
   tera_fixed_content.py：打包的相关配置，各种环境和路径
   tera_globals.p：日志写入
   tera_param.py：读入tera_fixed_content的配置，转化成命令行参数
   tera_post_log.py：日志上传
   tera_utility.py：ssh执行、zip压缩、ftp上传下载、日志记录、锁、流程的退出
3. 脚本调用关系



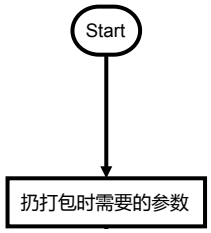


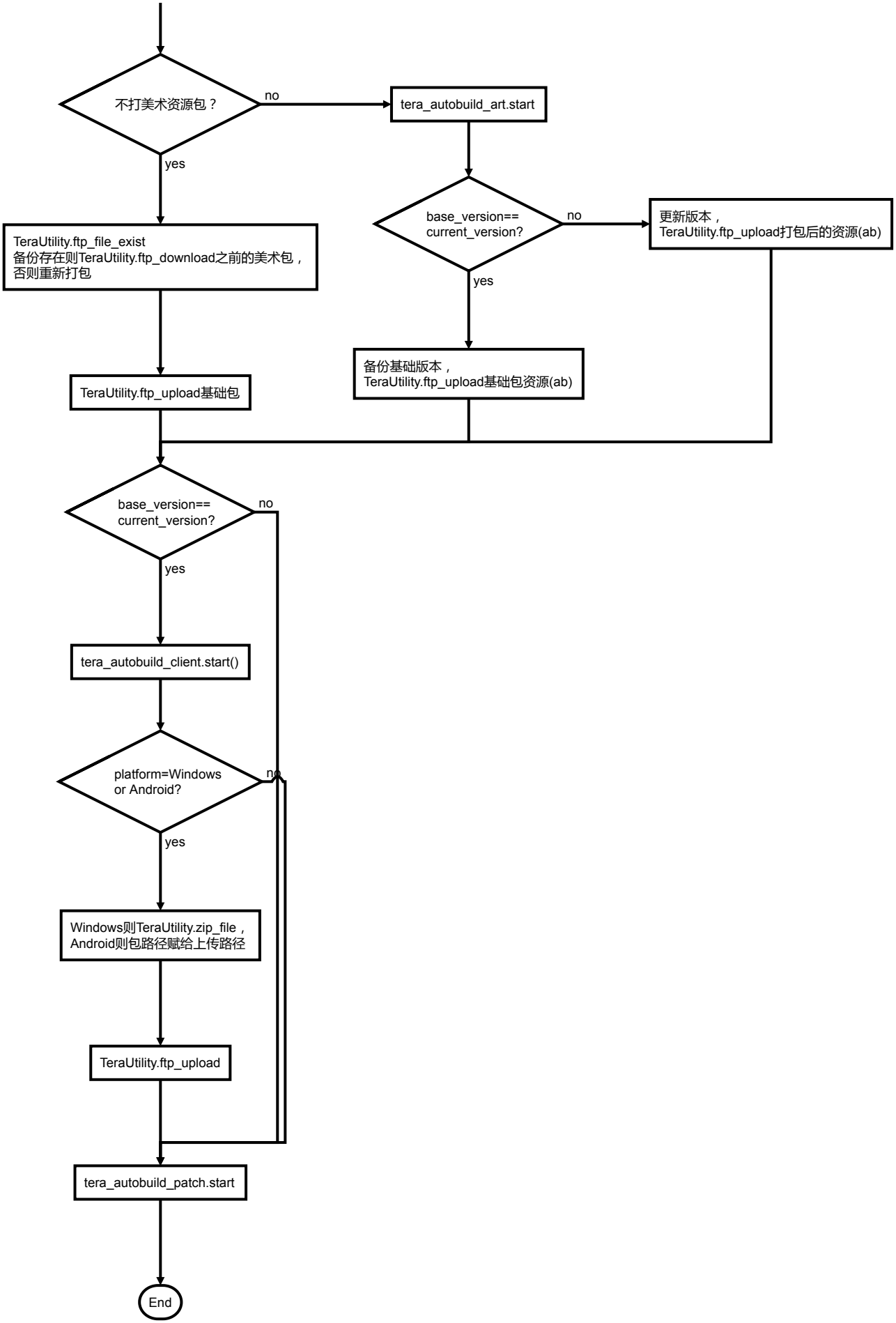
4. 自动打包脚本解析
   tera_autobuild_main.main()





打包逻辑tera_autobuild_main.start()





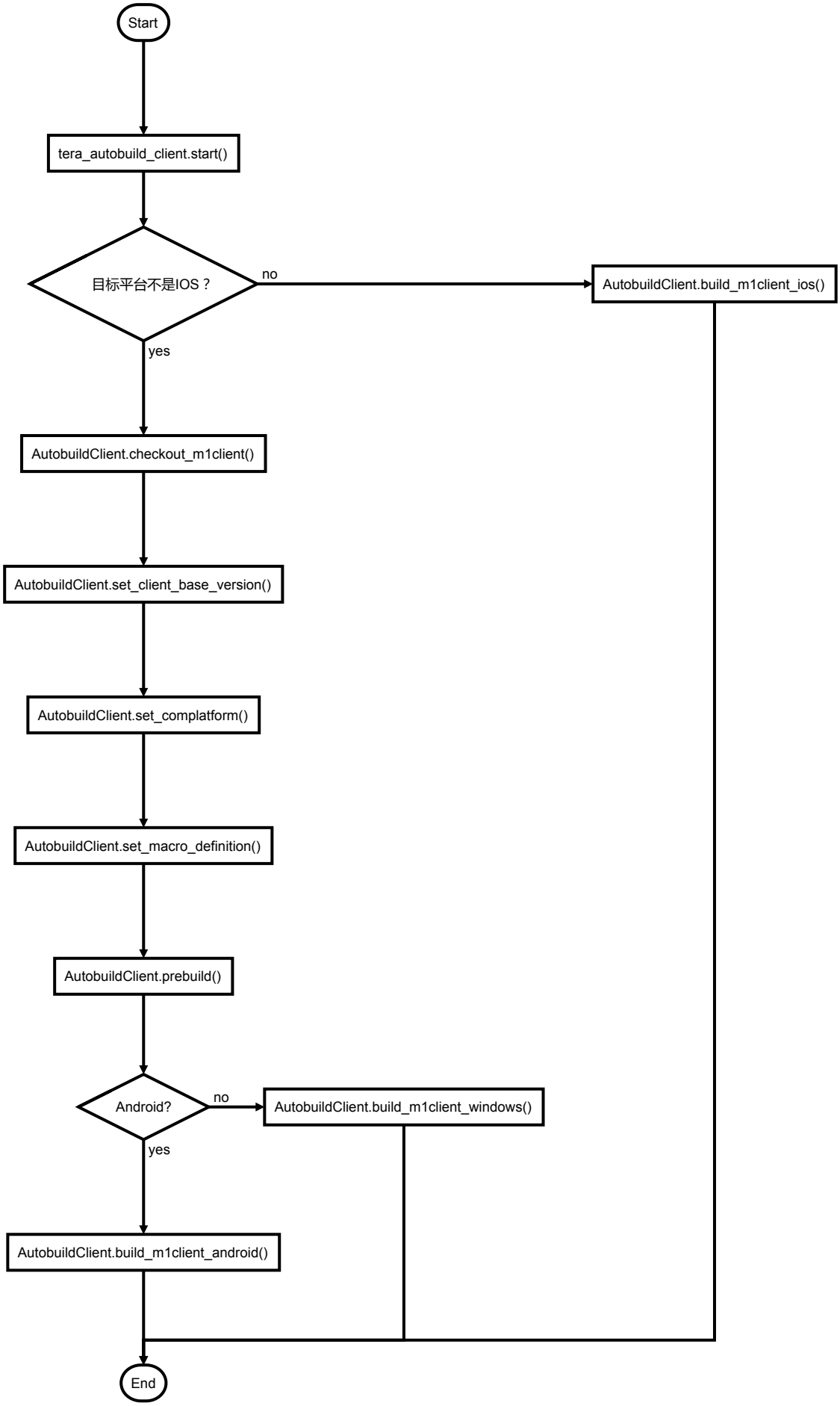
打包流程中的几个点

锁：通过文件的存在与否决定流程走向。【条件--结果】：锁文件存在(running)--post失败日志，锁文件不存在(notrunning)--建文件，继续下一步

锁的生成方式：【条件--结果】：try_start_build(notrunning)--生成，failed_exit(notrunnint)--销毁，success_exit--销毁

exit的时候都是post请求**http://10.35.51.30/buildcmd_finish**

5. 客户端打包流程



6. 补丁(Patch)打包流程描述

1.AutobuildPatch.start()

2.当前版本=基础版本则删除文件夹

workspace_path\patch\branch_path\base_version\platform_type\UpdateResource\LastVersion

workspace_path\patch\branch_path\base_version\platform_type\UpdateResource\LastVersion
清空文件夹（删除再创建的过程）
workspace_path\patch\branch_path\base_version\platform_type\JupGenerate
3.__generate_patch()，转到文件夹**workspace_path\commandtool**当前版本=基础版本？
3.1.【TRUE】__generate_base()生成基础包
workspace_path\commandtool\HobaPackToolsCommand.exe _base_version workspace_path\patch\branch_path\base_version\platform_type\JupGenerate
如果需要生成小包（第一个整资源更新包），则__generate_update(True)
3.2.【FALSE】__generate_update()
4.__generate_update(self, is_small_pack = False)
4.1.【is_small_pack=TRUE】如果需要生成小包则__generate_first_whole_update()，小包体第一个整资源更新包
更新当前版本客户端资源__checkout_m1client_to_path(...)，比对上一版本，进行更新
patch_workspace_path\commandtool\HobaPackToolsCommand.exe _platform_type _base_version _last_version next_version _last_folder _current_folder _output_path _is_smallpack
4.2.【is_small_pack=FALSE】__generate_common_update()，正常更新流程
__checkout_m1client_to_path(...)上一版本，__checkout_m1client_to_path(...)当前版本，比对更新
patch_workspace_path\commandtool\HobaPackToolsCommand.exe _platform_type _base_version _last_version _current_version _last_folder _current_folder _output_path _is_smallpack
5.上传文件
6.配置ftp权限

> **非本地打包时，.manifest文件的去除原因？**
>
> Unity中，.manifest文件都只是用来做本地依赖关系和增量打包的时候用的；有一个没有后缀名称的AssetBundle文件，包含了所有的依赖关系的总的依赖关系配置文件。
> 加载AssetBundle的时候，只需要那个没有后缀名称的AssetBundle文件，代表的是该项目的所有AssetBundle的依赖关系。

Scheduler.exe

Laravel 计划任务

增加项目分支打包的流程

1. 是否增加新的打包节点（node --- pc_ip、mac_ip），如果是则需在t_buildnode添加记录
2. 修改项目的config文件夹下svntobuild.php文件，添加分支信息
3.

疑问

1. BuildNode的流程为何不集成到web中，作为web后台处理（待个人验证可行性）

优化

凯哥说的资源打包 建立分布式服务器

自动打包完整搭建流程（踩坑记录）

WEB---AutoBuild网站搭建------Windows

https://blog.csdn.net/putin1223/article/details/44993985

1. 数据库导出sql文件命令： ***/bin/mysqldump -u root -p database_name >file_name.sql*** 需要SQL密码
   数据库导入sql文件命令：进入mysql console ***use database_name;source path/file_name.sql***
2. 下载并安装WampServer3.1
   如果mysql密码忘记了请看 *https://blog.csdn.net/wuyan_meixin/article/details/26217087*
   【坑】mysql5.7中password字段变成authentication_string
3. 下载并安装Composer（官网下载或者通过php命令下载）
   3.1. *https://getcomposer.org/*
   3.2. ***php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"***
   php composer-setup.php

php -r "unlink('composer-setup.php');"

【坑】这种方式下php配置中ini文件一定修改，使得extension=php_openssl.dll生效

4. 下载Laravel *https://github.com/johnlui/Learn-Laravel-5* 解压扔到wampserver的www目录下
   ps：使用Composer命令下载一直没反应，据说是可行的。
5. 在Laravel工程目录中执行 ***composer install*** 或者 ***composer update***。install的时候一直failed，update就没报错，原因未知
   修改镜像 ***composer config -g repo.packagist composer https://packagist.phpcomposer.com***
   【坑】Laravel5是需要PHP7支持，因而composer安装的时候，一定要选对PHP的版本
   【坑】还得装Git
6. 配置Apache，**\wamp64\bin\apache\apache2.4.33\conf**下httpd.conf文件，找到该位置，加入server.php
   <IfModule dir_module>
   DirectoryIndex index.php index.php3 index.html index.htm server.php
   </IfModule> **==========**非必须
7. 验证，访问http://localhost/工程名/
   【坑】访问报错，没有key，就需要 ***php artisan key:generate***，执行报错
   【坑】github下载的Laravel源码没有.env文件，导致 ***php artisan key:generate*** 命令报错，就先 ***copy .env.example .evn*** 才解决
8. 将生成后的APP_KEY的值添加到项目config/app.php文件中的APP_KEY位置后（key-value的形式）
9. 即可
10. 针对自动打包的项目
    更改wampserver的项目根目录：

> (1)、修改wamp64/bin/apache/apache2.4.33/conf下http.conf文件，更改对应位置
>
> ```
> DocumentRoot "C:/mywork/copyProject/public"
> <Directory "C:/mywork/copyProject/public/">
> ```
>
> (2)、修改wamp64/bin/apache/apache2.4.33/conf/extra下httpd-vhosts.conf文件，更改对应位置，内容如上
> (3)、修改wamp64/scripts下config.inc.php文件，更改对应位置
>
> ```
> $wwwDir = 'C:/mywork/copyProject/public'
> ```

关于多站点配置，参考*https://www.cnblogs.com/catherineSue/p/6668133.html*

11. 跑通了^_^

WEB---AutoBuild网站搭建------Linux

暂无