Comake2使用详解

1、适用用户范围：

第一次使用comake2工具

使用comake2搭建环境：这时用户不需要关心COMAKE文件细节，和第一次使用comake2工具时的情形差不多，第一次使用comake2工具初始化环境

$mkdir --parent ps/se/ac/make

$cd ps/se/ac/make

$comake2 -S

$comake2 -UB

comake2 -S //初始化COMAKE文件

comake2 -S表示从平台最新基线版本获取依赖列表；comake2 -S -r 1.0.1.0表示从平台1.0.1.0版本获取依赖列表 友情提示：如果你的代码库里已有COMAKE文件，可跳过这一步；如果你的模块路径是ps/se/ac/make，请cd ps/se/ac/make再执行comake2 -S命令

comake2 -UB //下载并编译依赖代码

2、comake2命令

$ comake2 -h

comake[com make]能够自动帮助用户搭建环境,并且生成Makefile工具.

程序会读取目录下面的COMAKE文件,产生Makefile和环境.用户需要提供这个COMAKE文件.

参数:

-h --help 查看帮助

-D --debug 开启debug选项[默认不打开].-D -D可以查看更多调试信息.

-S --scratch 创建一个默认的COMAKE文件

-r --revision 从平台检出模块cvspath指定的TAG对应的依赖列表，配合-S使用，如-S -r 1.0.0.0

-E --export-configs 导出模块的4位版本依赖,存放在COMAKE.CONFIGS下面.比如-E public/ub@1.0.0.0

-W --watch-configs 查看本地依赖模块.-W -W可以查看模块引入来源.-W -W -W可以查看依赖模块的依赖.

-I --import-files 在解释COMAKE文件之前导入模块

-C --change directory 切换到directory下面执行[默认当前目录]

-Q --quiet 安静模式[默认不打开]

-U --update-configs 更新环境

-B --build-configs 构建环境

-F --force 构建环境时强制进行[默认不进行]

-e --export-local-configs 导出本地环境到CONFIGS.SCM文件

-f --scmfile= 重现编译环境

-d --devdiff 存在本地修改的共同开发依赖列表（多模块共同开发时适用）

-J --make-thread-number= 如果模块使用COMAKE生成的Makefile的话,编译线程数[默认是4]

-j --modules-thread-number= 并发下载、编译模块的线程数[默认是1]

-K --keep-going 构建/更新环境中途出错的话,忽略错误继续[已废弃]

-P --pretreatment 生成Makefile时不进行预处理[默认进行预处理]

-O --quot-all-deps 生成Makefile时引用所有头文件依赖[默认过滤目录外依赖]

--parent-module 支持父子模块添加进编译依赖

--no-recursive 不递归生成每个目录下面的Makefile[默认情况下是递归生成]

--no-revert 不恢复依赖模块的本地修改，配合-U使用[默认恢复]

--time-compile-link 计时编译和链接时间[默认不打开]
--recache 强制更新comake2缓存的依赖列表
--old-da 使用2.1.2及以前的依赖打平策略
--new-da 最新打平策略
在help中未列出的参数如下：
--dump-da=da|dc|df|ds|dm 查看依赖完整关系图（分别以不同的形式组织，All|Collected|Flatten|Sorted|Map）
-A --scmaudit 导出依赖树
--old-api
--conf 指定config文件 CONF_FILE
--dd= 和平台依赖模块进行对比 # --dd and --dump-cfgs share same mode when analyzing deps.
--dump-cfgs 列出所有依赖
--dump-cfgs-scmpf= 列出平台依赖
--get-remote-revision 与--dump-cfgs、--dd=等一起使用，获取依赖的代码当前最新版本
--redirect-file= 与--dump-cfgs、--dd=等一起使用，将输出重定位到文件里
--show-unstable-urls 与--dump-cfgs、--dd=等一起使用，列出不稳定的编译依赖
--warn-downgrade-only 与--dump-cfgs、--dd=等一起使用，列出低版本
--permissive-guilty-workroot 容忍错误的工作环境（只提示 不退出）

初始化COMAKE：comake2 -S
搭建环境：基本命令是comake2 -UB，其中-U是下载代码，-B是编译模块
并发搭建环境：comake2 -UB -j 4表示并发4线程下载和编译模块
单线程编译：comake2 -B -J 1表示使用make命令编译某个依赖模块时不进行并发（如该依赖模块的Makefile不支持并发，则适用于该情况）；
强制重新搭建环境：comake2 -UB -F，即使依赖已经编译过，也会重新编译
不revert本地修改：comake2 -UB --no-revert
生成Makefile：comake2 [-P]，-P会跳过预处理环节，加快生成Makefile的过程
查看依赖完整关系图：comake2 -U --dump-da=da
查看本地修改的编译依赖模块：comake2 -d
和平台依赖进行对比：comake2 --dd=1.0.0.0
重现编译环境：comake2 --scmfile=模块名.COMAKE.CONFIGS.SCM
配置文件COMAKE语法
#coding:gbk

工作路径.层数=CVS路径的层数
WORKROOT('../../')
使用硬链接copy.
CopyUsingHardLink(True)
支持32位/64位平台编译
ENABLE_MULTI_LIBS(True)
C预处理器参数.
CPPFLAGS('-D_GNU_SOURCE -D__STDC_LIMIT_MACROS -DVERSION=\\\"1.9.8.7\\\"')
为32位目标编译指定额外的预处理参数
CPPFLAGS_32('-D_XOPEN_SOURE=500')
C编译参数.
CFLAGS('-g -pipe -W -Wall -fPIC')
C++编译参数.

CXXFLAGS('-g -pipe -W -Wall -fPIC')
IDL编译参数
IDLFLAGS('--compack')
UBRPC编译参数
UBRPCFLAGS('--compack')
头文件路径.
INCPATHS('. ./include ./output ./output/include')
使用库
LIBS('./libci-tools.a')
链接参数.
LDFLAGS('-lpthread -lcrypto -lrt')
依赖模块
CONFIGS('lib2/ullib')
为32位/64位指定不同的依赖路径.
CONFIGS_32('lib2/ullib')
CONFIGS_64('lib2-64/ullib')
user_sources=''
user_headers=''
可执行文件
Application('ci-tools',Sources(user_sources))
静态库
StaticLibrary('ci-tools',Sources(user_sources),HeaderFiles(user_headers))
共享库
SharedLibrary('ci-tools',Sources(user_sources),HeaderFiles(user_headers))
子目录
Directory('demo')
编译标签
元素名称 作用 example 注意事项
WORKROOT 当前工作路径 WORKROOT('../../') 最好使用相对路径
MakeThreadNumber 使用多少个线程进行make MakeThreadNumber(4) 相当于执行使用make -j[废弃]
CopyUsingHardLink copy使用硬链接 CopyUsingHardLink(True)
ENABLE_MULTI_LIBS 多环境编译 ENABLE_MULTI_LIBS(True) 支持多环境编译，生成32位/64位通用的Makefile
CPPFLAGS C预处理参数 CPPFLAGS('-DGNU_SOURCE')
CFLAGS C编译参数 CFLAGS('-g -Wall','-O2')
CXXFLAGS C++编译参数 CXXFLAGS('-g -Wall','-O2')
INCPATHS搜索头文件路径 INCPATHS('. ./include')
这里不要加上-I前缀.
INCPATHS允许使用$开头,代表路径相对于WORKROOT.比如INCPATHS('$public/ub')
LIBS 链接使用的库文件 LIBS('./libmock.a')
LDFLAGS 链接参数 LDFLAGS('-lpthread -lcrypto')
IDLFLAGS IDL编译参数 IDLFLAGS('--compack --ns=mock') 实际上是mcy的编译参数
UBRPCFLAGS UBRPCGEN编译参数 UBRPCFLAGS('--compack')
PROTOFLAGS proto编译参数 PROTOFLAGS('--proto_path=./')
PROTOC protobuf路径 PROTOC('../../thirdsrc/protobuf/install/bin/protoc')
ImportConfigsFrom 从其他目录的COMAKE继承编译依赖 ImportConfigsFrom('..')
ReplaceExtNameWith 替换文件后缀名 ReplaceExtNameWith('x.cpp y.cpp','.cpp','.o')

GetEnv 获得环境变量 GetEnv('USER') 如果没有特定key的话,返回'undefined'
BuildVersion 获取编译版本 BuildVersion() 等效于GetEnv('COMAKE2_BUILD_VERSION')
CCHECKFLAGS ccheck参数 CCHECKFLAGS('-c cc.conf')
PCLINTFLAGS pclint参数 PCLINTFLAGS('-c pclint.conf')