huangzhen@bird vccap $ echo ${CROOT}

/vob/huangzhen/casa

基础知识：

|  |
| --- |
| make执行过程  1.读入所有的Makefile  2.读入被include 的其他 Makefile  3.初始化文件中的变量。  4.推导隐晦规则，并分析所有规则。  5.为所有目标文件创建依赖关系链。  6.根据依赖关系，决定哪些目标要重新生成。  7.执行生成命令 |

|  |
| --- |
| target ... : prerequisites ...  command  ...  ...  目标：依赖  命令  目标依赖于依赖，当依赖发送变化时，执行命令 |
| 举个例子  cmtscp:  $(MAKE) -C build/cots cmtscp-package  Target是”cmtscp”  Prerequisites为空，表示无论如何command都会被执行  Command是”$(MAKE) -C build/cots cmtscp-package” ，表示进入build/cots，makefile文件为该路径下的Makefile，相当于执行make cmtscp-packge |

|  |
| --- |
| Gcc命令解析  -c 对源代码进行预处理、编译、汇编，但不执行链接，产生的是源代码的目标文件(\*.o)  -I 指定头文件包含目录  -L 指定库文件目录,假定在/home/jie/lib目录下动态库文件libadd.so和静态库文件libadd.a  -l 指定库的名字  -Wall 产生尽可能多的警告信息  -Werror 将所有的警告当成错误进行处理 |

Makefile文件分析

Makefile文件定义了一系列的规则。

当我们执行make时，TARGET是all。

当我们执行make 100G时，TARGET是100G。

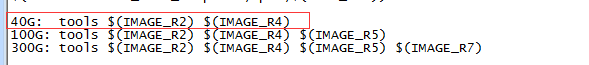
当我们执行make cmts-docker时，TARGET是cmts-docker。

Makefile会根据这个TARGET去寻找规则。如果没有这个TARGET会提示并中止：



在/vob/hz/vccap的Makefile文件中找到规则，





40G的依赖为tools image\_r2.gz image\_r4.gz

|  |
| --- |
| Tool  $(call MAKE\_rule\_template,tools,$(TOOLS\_DIR))  调函数MAKE\_rule\_template，传入两个参数 参数1为tools 参数2为$(TOOLS\_DIR)  define MAKE\_rule\_template  $(1): ;  $$(MAKE) -C $(2)  endef  转化成  target ... : prerequisites ...  command  ...  ...  为  tools: ;  MAKE -C $(TOOLS\_DIR)  相当于执行 MAKE -C $(TOOLS\_DIR)  指明跳转到tools目录下读取那里的Makefile 编译tools  找到tools目录并进入：TOOLS\_DIR=$(CROOT)/tools  前缀 @ 只输出命令执行的结果, 出错的话停止执行  为当前uepi\_psp.c uepi\_debug.c创建软连接 目标文件为$(uepi\_SRCDIR)/uepi\_psp.c $(uepi\_SRCDIR)/uepi\_debug.c  gcc -o gzipcmntcrc3 gzipcmntcrc3.c -I.  gcc -o rphystream uepi\_encap.c rphystream.c uepi\_psp.c uepi\_debug.c -I. -I$(uepi\_SRCDIR) -I$(dcts\_SRCDIR)/include  编译tools之后会生成gzipcmntcrc3 rphystream |

|  |
| --- |
| ARCHIVEDIRS=40G 100G 300G |

|  |
| --- |
| Image  $(foreach archive,$(ARCHIVEDIRS),$(eval $(call MAKE\_rule\_template,$(archive),$(CROOT)/build/archive/$(archive))))  foreach是一个循环语句  $(foreach <var>,<list>,<text>)  函数功能：函数“eval”是一个比较特殊的函数。  使用它可以在Makefile中构造一个可变的规则结构关系（依赖关系链），其中可以使用其它变量和函数。  函数“eval”对它的参数进行展开，展开的结果作为Makefile的一部分，make可以对展开内容进行语法解析。  展开的结果可以包含一个新变量、目标、隐含规则或者是明确规则等。  也就是说此函数的功能主要是：根据其参数的关系、结构，对它们进行替换展开。  解读语法：  例如$(foreach archive,$(ARCHIVEDIRS),$(eval $(call MAKE\_rule\_template,$(archive),$(CROOT)/build/archive/$(archive))))  对每一个在列表ARCHIVEDIRS的元素archive，执行$(call MAKE\_rule\_template,$(archive),$(CROOT)/build/archive/$(archive)操作  即调用函数MAKE\_rule\_template 传入两个参数  define MAKE\_rule\_template  $(1): ;  $$(MAKE) -C $(2)  endef  因为之前定义了ARCHIVEDIRS=40G 100G 300G  所以相当于  40G:  MAKE -C $(CROOT)/build/archive/40G  100G:  MAKE -C $(CROOT)/build/archive/100G  300G:  MAKE -C $(CROOT)/build/archive/300G  进入$(CROOT)/build/archive/40G目录，找到makefile文件(Makefile) 执行make (all) |

|  |
| --- |
| 查看$(CROOT)/build/archive/40G/Makefile  找到目标”all”的规则    两个依赖 tools和image  (tools见上文)  规则    目标：image  依赖：c10g xlp  命令：   1. 重定向image\_r2.gz image\_r4.gz到image\_tmp.gz 2. 执行mksmmimg.sh 参数1为image\_tmp.gz 参数2为image\_40g.gz 3. 删除image\_tmp.gz     目标：xlp  依赖：lcversion  命令：到NETL\_XLP\_DIR=../../netl-xlp目录 执行 make $(plt\_flags)    目标：c10g  依赖：lcversion smmversion $(DTB1) $(DTB2)  命令：到RMI\_G2\_DIR=../../rmi-g2目录执行make (all) |
| 进入casa/build/rmi-g2目录  查看Makefile    目标：all  依赖：tools prep\_kernel smm\_appls lc\_appls smm\_fs lc\_fs image |

|  |
| --- |
| smm\_appls    目标：smm\_appls  依赖：rmi\_common\_deps  命令：make -f smm.mk all  指定makefile文件为smm.mk 执行make all    目标：rmi\_common\_deps  命令：make -f smm.mk glib protobuf-c SSL libevent2  并定义TEMPLATE\_NO\_DEP\_RULES=1 |

|  |
| --- |
| 查看smm.mk    目标：all  依赖：casa pub kerbobj  （发现当前文件smm.mk没有相应的规则，它们在include所包含的文件里。）    Include列表  include $(CROOT)/casa\_global.mk  include $(CROOT)/makerules/smm\_so\_lib.var.rules  include $(CROOT)/makerules/smmmks.mk  include $(CROOT)/makerules/extern/pubbuilds.var.mk  include $(CROOT)/makerules/extern/staticbuild.var.mk  include $(CROOT)/makerules/extern/ssl.var.mk  include $(CROOT)/makerules/extern/glib.var.mk  include $(CROOT)/makerules/extern/strongswan.var.mk  include $(CROOT)/makerules/program\_template.mk  include $(CROOT)/../pub/pub.mk  include $(CROOT)/makerules/extern/strongswan.mk.mk  根据make执行过程第2点和第3点  需要把.mk文件展开并重建规则 |

|  |
| --- |
| 查看casa/makerules/smmmks.mk  # smmmks.mk  # file for all smm/control plane specific .mk file includes and  # PROGRAM and PUB\_BUILDS defines  Include列表（省略）  根据make执行过程第2点和第3点  需要把.mk文件展开并重建规则 |

|  |
| --- |
| 查看casa/mgmt/CLI/cli.mk  发现里面定义了很多变量，这些变量有什么作用呢？  回到smm.mk  看到下面紧接着一行  include $(CROOT)/makerules/program\_template.mk  这是一个模板  casa/makerules/template.README 对program\_template.mk的使用做出了解释  结合program\_template.mk和cli.mk |

|  |
| --- |
| 其他makefile文件类似 |