嵌入式操作系统

5 BootLoader 举例

陈香兰（xlanchen@ustc.edu.cn）

计算机应用教研室 @ 计算机学院
嵌入式系统实验室 @ 苏州研究院
中国科学技术大学

/media/SAMSUNG/work/6 实验室相关

Outline

Outline

u-boot （Universal Boot）

- uboot 是在 ppcboot 以及 armboot 的基础上发展而来的
- 支持很多处理器，比如 PowerPC、ARM、MIPS、x86

最新的主页

http://www.denx.de/wiki/U-Boot

u-boot 的使用手册，参见

The DENX U-Boot and Linux Guide (DULG) for canyonlands

1. 安装交叉开发环境 ELDK：Embedded Linux Development Kit
2. 通过串口 / 网络连接到目标端
3. 配置、编译并安装 u-boot
4. 配置、编译并安装 Linux
   - 该手册使用 SELF：
     Simple Embedded Linux Framework

Outline

编译 u-boot I

- 下载源码u-boot-2009.08.tar.bz2
- 解压缩

```
tar jvxf u-boot-2009.08.tar.bz2
```

- 编译 u-boot
  - 使用交叉编译器：arm-linux-tools-20061213.tar.gz (gcc 版本为 3.4.4)

make ep7312_config
make all

```
common.a libfdt/libfdt.a api/libapi.a post/libpost.a board/ep7312/libep7312.a --end-
group -L /usr/local/lib/gcc/arm-linux/3.4.4/soft-float -lgcc -Map u-boot.map -o u-bo
ot
arm-linux-objcopy -O srec u-boot u-boot.srec
arm-linux-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
xlanchen@xlanchen-desktop:~/workspace/u-boot-2009.08$
```

编译 u-boot II

- 查看目录中的文件

```
xlanchen@xlanchen-desktop:~/workspace/u-boot-2009.08$ ls
api                          examples        lib_nios     post
board                        fs              lib_nios2    README
CHANGELOG                    include         lib_ppc      rules.mk
CHANGELOG-before-U-Boot-1.1.5 lib_arm        lib_sh       System.map
common                       lib_avr32       lib_sparc    tools
config.mk                    lib_blackfin    MAINTAINERS  u-boot
COPYING                      libfdt          MAKEALL      u-boot.bin
cpu                          lib_generic     Makefile     u-boot.lds
CREDITS                      lib_i386        mkconfig     u-boot.map
disk                         lib_m68k        nand_spl     u-boot.srec
doc                          lib_microblaze  net
drivers                      lib_mips        onenand_ipl
xlanchen@xlanchen-desktop:~/workspace/u-boot-2009.08$
```

- 在 skyeye 上运行 u-boot
  - 拷贝 skyeye-testsuite-1.2.8/u-boot/ep7312/skyeye.conf
  - 运行

编译 u-boot III

skyeye -c skyeye.conf -e u-boot

```
big_endian is false.
arch: arm
cpu info: armv4, arm720t, 41807200, ffffff00, 1
mach info: name ep7312, mach_init addr 0x805abf0
dbct info: Note: DBCT not compiled in. This option will be ignored
uart_mod:0, desc_in:, desc_out:, converter:
SKYEYE: use arm7100 mmu ops
start addr is set to 0xc0f80000 by exec file.



U-Boot 2009.08 ( 9月  21 2009 - 18:29:23)

DRAM:  16 MB
Flash: 16 MB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
EP7312 #
```

Outline

u-boot 源代码 I

- 阅读源码根目录下的 README
  - 了解目录结构和各目录下的内容
  - 了解 u-boot 的配置和编译的命令
  - 知道在什么地方可以找到配置文件中的各个配置信息的含义
- 阅读根目录下的 Makefile
  - 了解配置和编译过程
  - 了解 ARCH、CPU、BORAD 三个层次
    - 例如：arm、arm720t、ep7312
  - 了解编译的产物有哪些
  - 了解目标映像的结构
  - 知道 u-boot 的启动文件：start.S

u-boot 源代码 II

- make ep7312_config

```
3075 ep7312_config»  :»       unconfig
3076 »       @$(MKCONFIG) $(@:_config=) arm arm720t ep7312
```

```
493 unconfig:
494 »       @rm -f $(obj)include/config.h $(obj)include/config.mk \
495 »       »       $(obj)board/*/config.tmp $(obj)board/*/*/config.tmp \
496 »       »       $(obj)include/autoconf.mk $(obj)include/autoconf.mk.dep
```

```
111 MKCONFIG»       := $(SRCTREE)/mkconfig
112 export MKCONFIG
```

- make all

```
294 # Always append ALL so that arch config.mk's can add custom ones
295 ALL += $(obj)u-boot.srec $(obj)u-boot.bin $(obj)System.map $(U_BOOT_NAND) $(U_BOOT
296
297 all:»    »       $(ALL)
```

u-boot 源代码 III

```
302 $(obj)u-boot.srec:»     $(obj)u-boot
303 »       »       $(OBJCOPY) -O srec $< $@
304
305 $(obj)u-boot.bin:»     $(obj)u-boot
306 »       »       $(OBJCOPY) ${OBJCFLAGS} -O binary $< $@
```

```
331 GEN_UBOOT = \
332 »       »       UNDEF_SYM=`$(OBJDUMP) -x $(LIBBOARD) $(LIBS) | \
333 »       »       sed  -n -e 's/.*\($(SYM_PREFIX)__u_boot_cmd_.*\)/-u\1/p'|sort|uniq`;\
334 »       »       cd $(LNDIR) && $(LD) $(LDFLAGS) $$UNDEF_SYM $(__OBJS) \
335 »       »       »       --start-group $(__LIBS) --end-group $(PLATFORM_LIBS) \
336 »       »       »       -Map u-boot.map -o u-boot
337 $(obj)u-boot:»  depend $(SUBDIRS) $(OBJS) $(LIBBOARD) $(LIBS) $(LDSCRIPT) $(obj)u-boot.lds
338 »       »       $(GEN_UBOOT)
339 ifeq ($(CONFIG_KALLSYMS),y)
340 »       »       smap=`$(call SYSTEM_MAP,u-boot) | \
341 »       »       »       awk '$$2 ~ /[tTwW]/ {printf $$1 $$3 "\\\\000"}'` ; \
342 »       »       $(CC) $(CFLAGS) -DSYSTEM_MAP="\"$${smap}\"" \
343 »       »       »       -c common/system_map.c -o $(obj)common/system_map.o
344 »       »       $(GEN_UBOOT) $(obj)common/system_map.o
345 endif
```

u-boot 源代码 IV

```
169 #########################################################################
170 # U-Boot objects....order is important (i.e. start must be first)
171
172 OBJS  = cpu/$(CPU)/start.o
173 ifeq ($(CPU),i386)
174 OBJS += cpu/$(CPU)/start16.o
175 OBJS += cpu/$(CPU)/resetvec.o
176 endif
177 ifeq ($(CPU),ppc4xx)
178 OBJS += cpu/$(CPU)/resetvec.o
179 endif
180 ifeq ($(CPU),mpc85xx)
181 OBJS += cpu/$(CPU)/resetvec.o
182 endif
183
```

u-boot 源代码 V

```
186 LIBS  = lib_generic/libgeneric.a
187 LIBS += lib_generic/lzma/liblzma.a
188 LIBS += lib_generic/lzo/liblzo.a
189 LIBS += $(shell if [ -f board/$(VENDOR)/common/Makefile ]; then echo \
190 »       "board/$(VENDOR)/common/lib$(VENDOR).a"; fi)
191 LIBS += cpu/$(CPU)/lib$(CPU).a
192 ifdef SOC
193 LIBS += cpu/$(CPU)/$(SOC)/lib$(SOC).a
194 endif
195 ifeq ($(CPU),ixp)
196 LIBS += cpu/ixp/npe/libnpe.a
197 endif
198 LIBS += lib_$(ARCH)/lib$(ARCH).a
199 LIBS += fs/cramfs/libcramfs.a fs/fat/libfat.a fs/fdos/libfdos.a fs/jff
200 »       fs/reiserfs/libreiserfs.a fs/ext2/libext2fs.a fs/yaffs2/libyaf
201 »       fs/ubifs/libubifs.a
202 LIBS += net/libnet.a
203 LIBS += disk/libdisk.a
```

```
288 __OBJS := $(subst $(obj),,$(OBJS))
289 __LIBS := $(subst $(obj),,$(LIBS)) $(subst $(obj),,$(LIBBOARD))
```

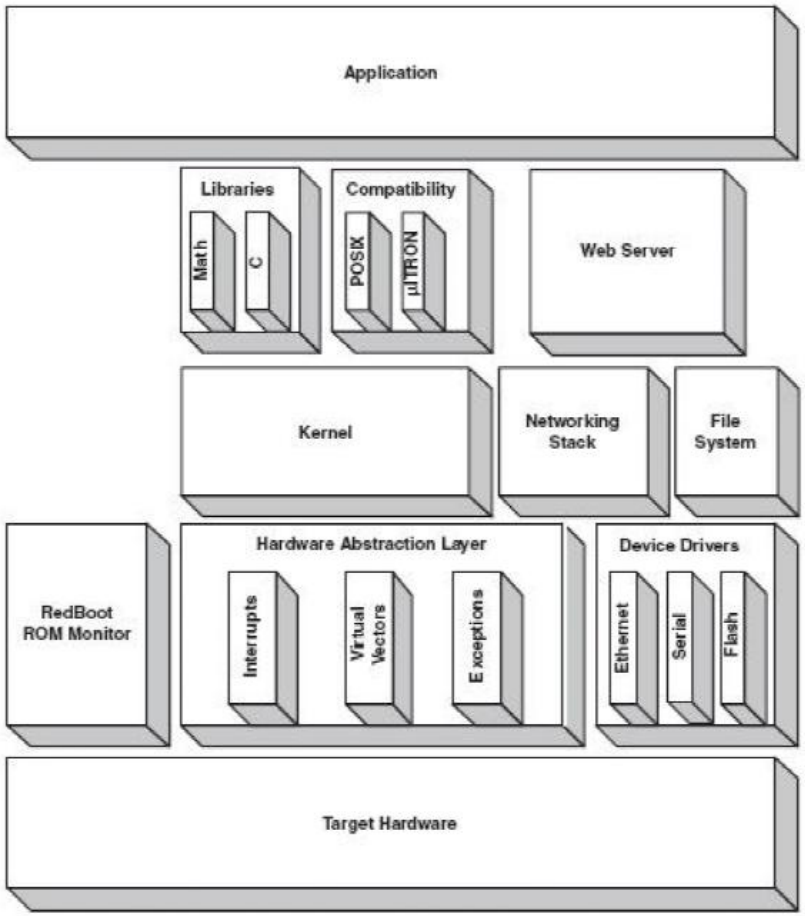
cpu/arm720t/start.S

- 阅读 cpu/arm720t/u-boot.lds，了解 start.S 在 u-boot 中的位置
- 了解 start.S 中的向量表的位置和含义
- 了解 arm 的启动方式
  - reset

Outline

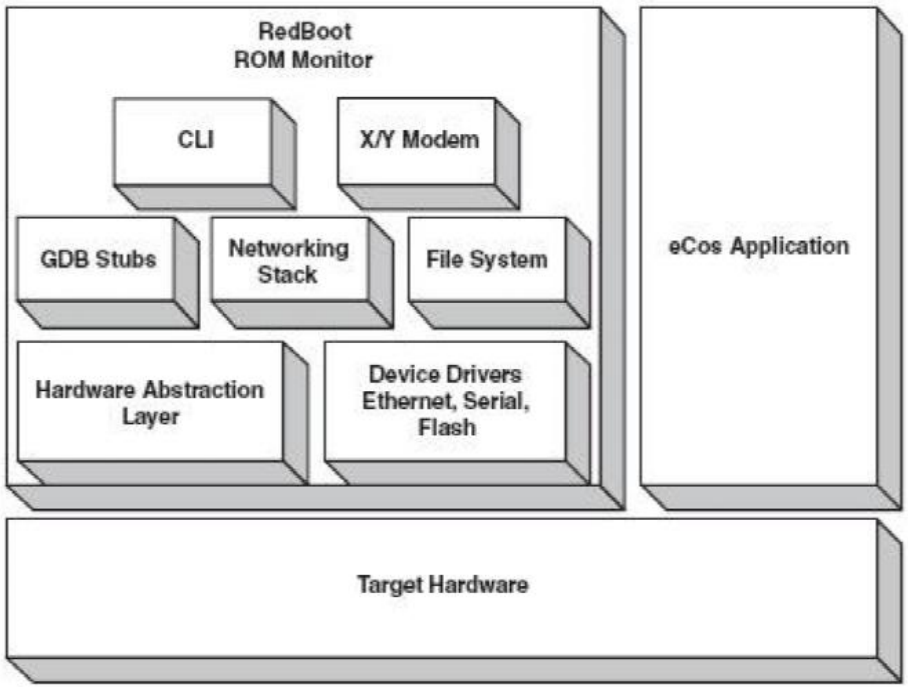
- RedHat Embedded Debug and Bootstrap
  - 是 RedHat 公司的一个标准嵌入式系统引导和调试环境
  - 基于 eCos 操作系统 
    - eCos 是个可配置的操作系统（http://ecos.sourceware.org/）
    - RedBoot 是从 eCos 配置而来，RedBoot User's Guide
  - 支持 ARM 系列、MIPS 系列、PPC 系列等平台
  - 具有较高的可移植性
  - 移植的重点是其体系结构平台相关层
- 目前最新版本为 3.0
  - 文档链接：http://ecos.sourceware.org/docs-3.0/

eCos 架构



http://lh6.ggpht.com/rolle.xu/SDaAqa9yypI/AAAAAAAAAEw/jUYE6htfVNY/s800/ecos.jpg

RedBoot 架构



http://lh5.ggpht.com/rolle.xu/SDaAqK9yyoI/AAAAAAAAAEo/eNK6Vz1uOeY/s800/redboot.jpg

常用命令

- alias – Manipulate command line aliases
- baudrate – Set the baud rate for the system serial console
- cache – Control hardware caches
- channel – Select the system console channel
- cksum – Compute POSIX checksums
- disks – List available disk partitions.
- dump – Display memory.
- help – Display help on available commands
- iopeek – Read I/O location
- iopoke – Write I/O location
- gunzip – Uncompress GZIP compressed data
- ip_address – Set IP addresses
- load – Download programs or data to the RedBoot platform
- mcmp – Compare two segments of memory
- mcopy – Copy memory
- mfill – Fill RAM with a specified pattern
- ping – Verify network connectivity
- reset – Reset the device
- version – Display RedBoot version information

Outline

下载、建立开发环境 I

- 从ftp://ecos.sourceware.org/pub/ecos/上获得 RedBoot
  - 下载 ecos-install.tcl
  - 添加执行权限 chmod +x ecos-install.tcl
  - 运行./ecos-install.tcl
    - 按照提示操作，给出安装目录 /home/xlanchen/workspace/ecos，编译器选择 1,2,3,q

下载、建立开发环境 II

```
Please select a directory for installation
[Default /home/xlanchen/ecos]: /home/xlanchen/workspace/ecos
-------------------------------------------------------------
Available prebuilt GNU tools:

[1]     arm-eabi
[2]     arm-elf (old)
[3]     i386-elf
[4]     m68k-elf
[5]     mipsisa32-elf
[6]     powerpc-eabi
[7]     sh-elf
[q]     Finish selecting GNU tools

("*" indicates tools already selected)
```

下载、建立开发环境 III

```
Directory /home/xlanchen/workspace/ecos does not exist... creating.
Entering /home/xlanchen/workspace/ecos
Retrieving eCos version 3.0
************************************************
Downloads complete.
If you wish to disconnect from the internet you may do so now.
Unpacking ecos-3.0.i386linux.tar.bz2...
Generating /home/xlanchen/workspace/ecos/ecosenv.sh
Generating /home/xlanchen/workspace/ecos/ecosenv.csh
------------------------------------------------------
In future, to establish the correct environment for eCos,
run one of the following commands:
   . /home/xlanchen/workspace/ecos/ecosenv.sh          (for sh/bash users); or
   source /home/xlanchen/workspace/ecos/ecosenv.csh    (for csh/tcsh users)

It is recommended you append these commands to the end of your
shell startup files such as $HOME/.profile or $HOME/.login
------------------------------------------------------
Installation complete!
```

下载、建立开发环境 IV

- 查看 ecos 的目录

```
xlanchen@xlanchen-desktop:~/workspace/ecos$ ls
ecos-3.0  ecosenv.csh  ecosenv.sh  gnutools
xlanchen@xlanchen-desktop:~/workspace/ecos$
```

其中，

```
xlanchen@xlanchen-desktop:~/workspace/ecos$ tree -L 1 gnutools/
gnutools/
|-- arm-eabi
|-- arm-elf
`-- i386-elf

3 directories, 0 files
xlanchen@xlanchen-desktop:~/workspace/ecos$
```

下载、建立开发环境 V

- 运行 ecosenv.sh，建立开发环境

  . ecosenv.sh 或者source ecosenv.sh

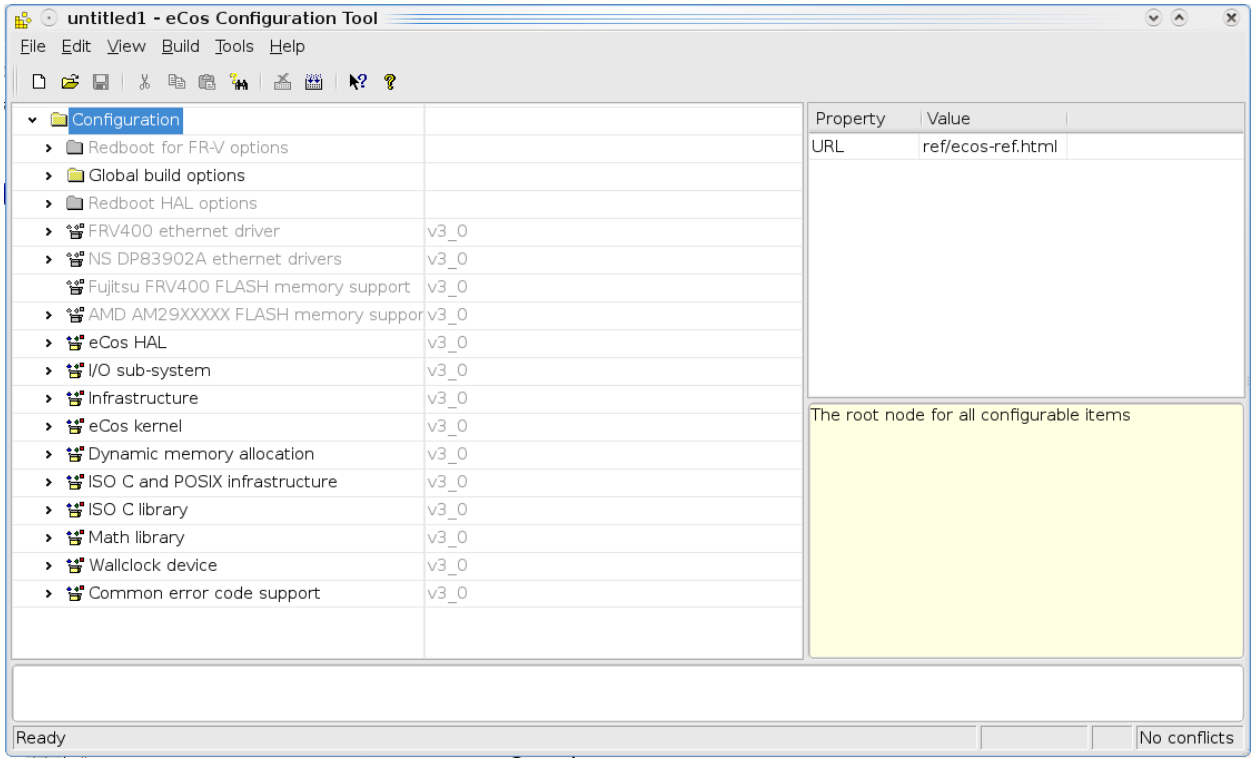
```
# Warning! This is a generated file used by the eCos installer.
# If you edit this file, you may break future upgrades using the installer.

ECOS_REPOSITORY=/home/xlanchen/workspace/ecos/ecos-3.0/packages ; export
ECOS_REPOSITORY

# eCos paths - do not modify this line, it is used by the installer
PATH=/home/xlanchen/workspace/ecos/ecos-3.0/tools/bin:$PATH ; export PATH
# End eCos paths - do not modify this line, it is used by the installer
```

- 查看 ecosenv.sh 是否运行成功，

```
xlanchen@xlanchen-desktop:~/workspace/ecos$ . ecosenv.sh
xlanchen@xlanchen-desktop:~/workspace/ecos$ echo $PATH
/home/xlanchen/workspace/ecos/gnutools/arm-elf/bin:/home/xlanch
en/workspace/ecos/ecos-3.0/tools/bin:/home/xlanchen/workspace/R
TEMS/i386-rtems-4.7/bin:/usr/local/sbin:/usr/local/bin:/usr/sbi
n:/usr/bin:/sbin:/bin:/usr/games
xlanchen@xlanchen-desktop:~/workspace/ecos$
```

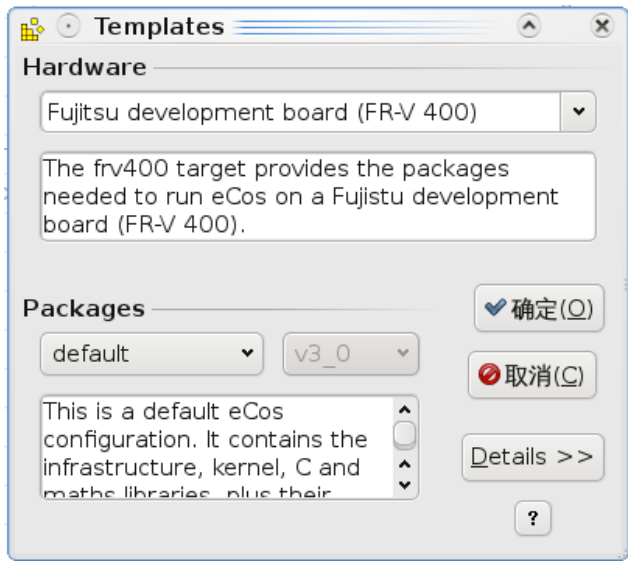
为 pc 配置、编译 redboot I

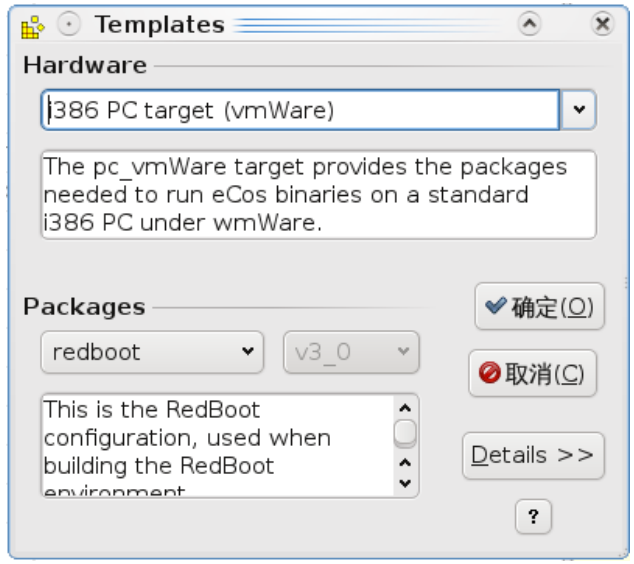
- 使用配置工具~/workspace/ecos/ecos-3.0/tools/bin/configtool



为 pc 配置、编译 redboot II

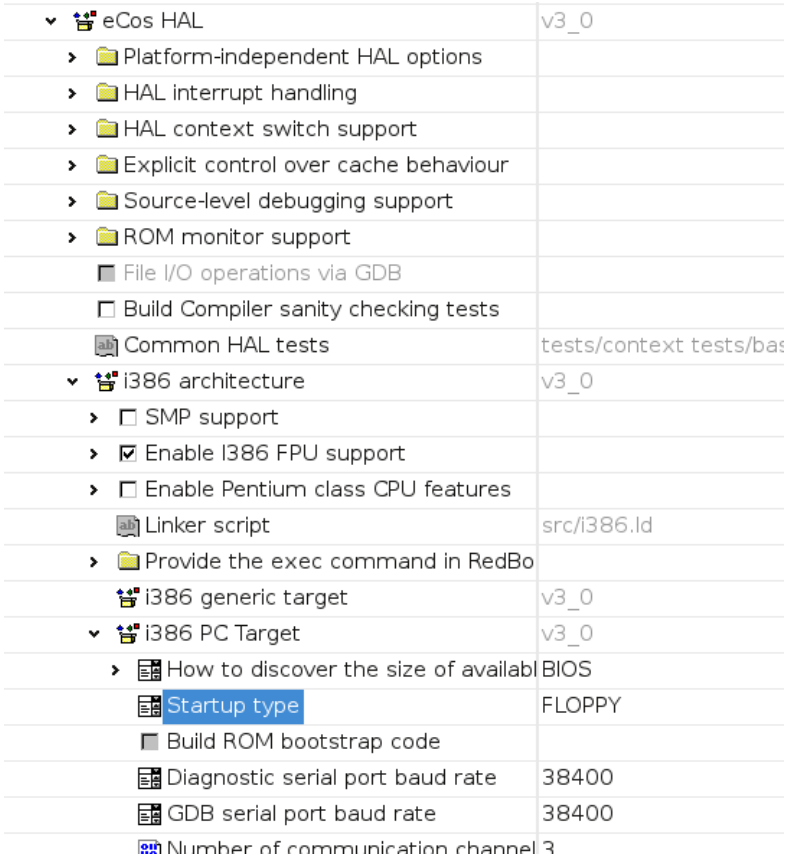
- 在 build 菜单中选择 Templates 得到如下左边的界面。其中，Hardware 选择“i386 PC target (vmWare)”；Packages 选择“redboot”。确定。(若报冲突，确认即可)





为 pc 配置、编译 redboot III

- 在修改启动选项为 floppy



- 创建一个目录 virtualbox-i386，将配置结果保存到这个目录下，命名为“i386.ecc”

为 pc 配置、编译 redboot IV

- 进入 virtualbox-i386 目录，可以看到：

```
xlanchen@xlanchen-desktop:~/workspace/ecos/virtualbox-i386$ ls
i386_build  i386.ecc  i386_install
xlanchen@xlanchen-desktop:~/workspace/ecos/virtualbox-i386$
```

  其中，ecc 是 eCos Configuration 的缩写，表示配置文件

- 进入 i386_build，运行 make
- make 结束后，到 i386_install 目录下，可以看到生成了 redboot.bin 文件和 redboot.elf 文件

```
xlanchen@xlanchen-desktop:~/workspace/ecos/virtualbox-i386/i386_install$ tree bin
bin
|-- redboot.bin
`-- redboot.elf

0 directories, 2 files
xlanchen@xlanchen-desktop:~/workspace/ecos/virtualbox-i386/i386_install$
```

在 virtualbox 上运行 redboot I

- 安装 virtualbox

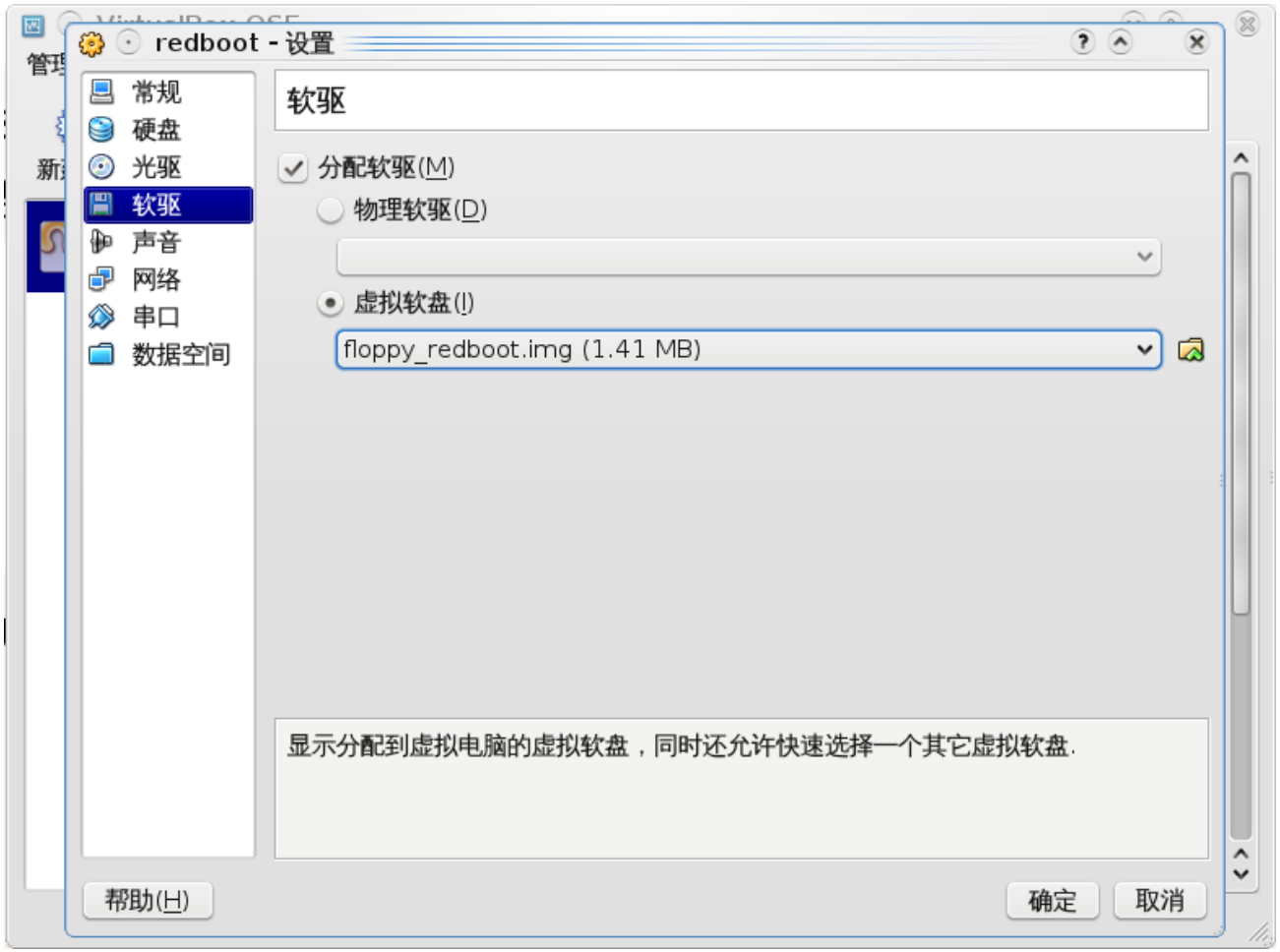
  sudo apt-get install virtualbox

- 为 redboot.bin 生成软盘镜像 floppy_redboot.img

```
dd conv=sync if=redboot.bin of=floppy_redboot.img bs=1440k
```

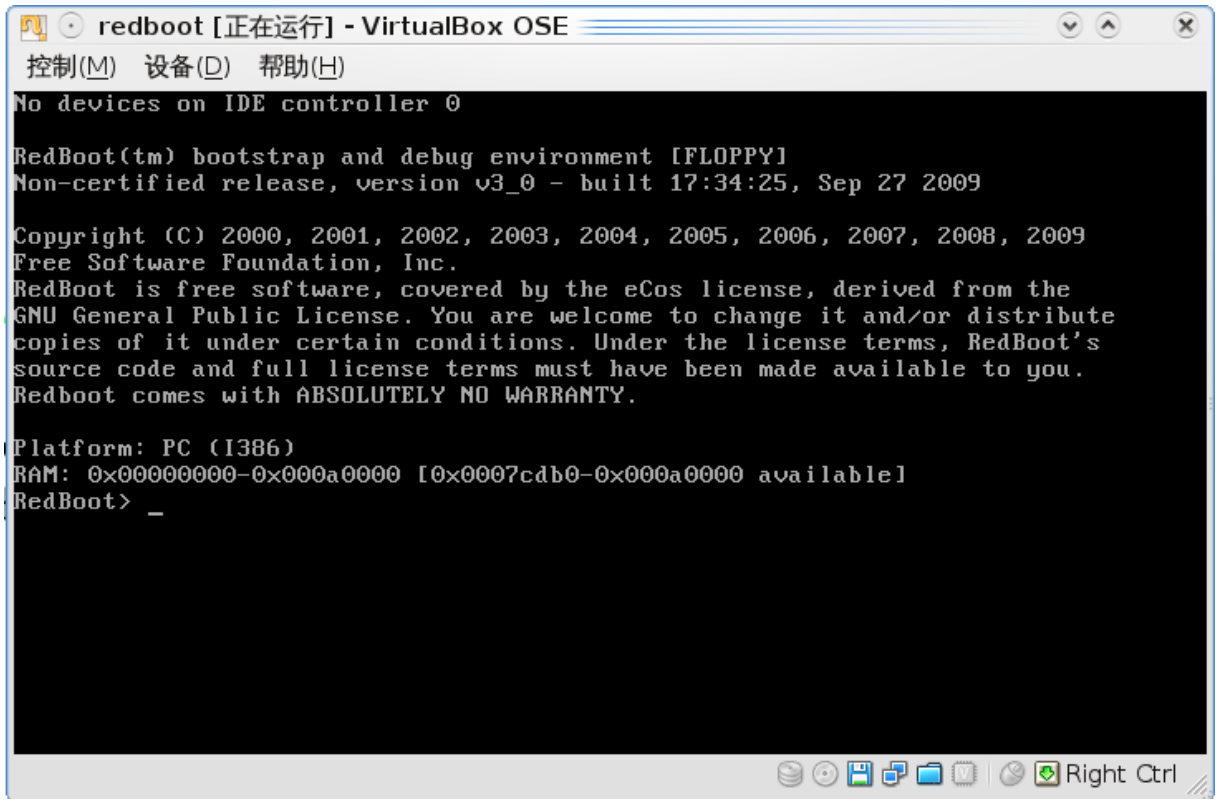
- 运行 virtualbox，创建一个虚拟机 redboot，配置软盘，使用上面生成的 floppy_redboot.img

在 virtualbox 上运行 redboot II



在 virtualbox 上运行 redboot III

- 运行虚拟机 redboot

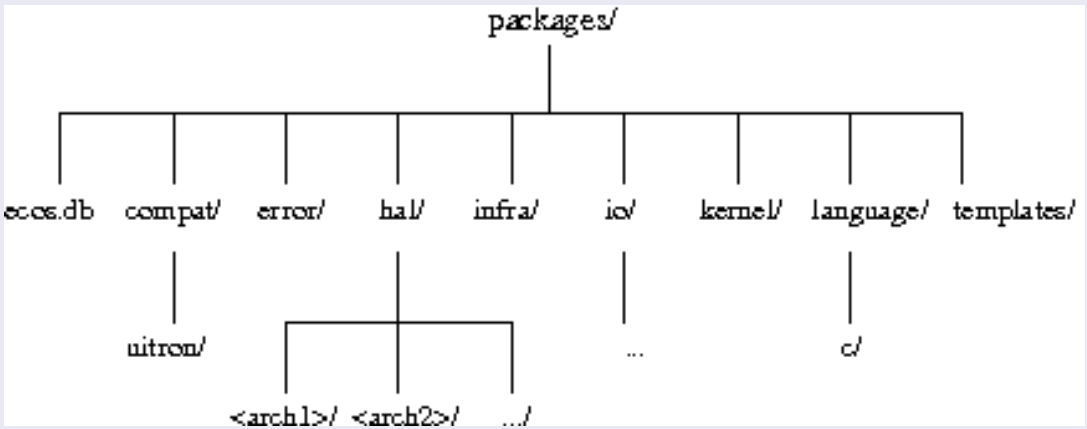


Outline

目录结构分析 I

- 参见
  The eCos Component Writer's Guide

packages 目录层次结构



目录结构分析 II

```
ecos-3.0/packages/
|-- ChangeLog
|-- NEWS
|-- compat
|-- cygmon
|-- devs
|-- ecos.db
|-- ecosadmin.tcl
|-- error
|-- fs
|-- hal
|-- infra
|-- io
|-- isoinfra
|-- kernel
|-- language
|-- net
|-- pkgconf
|-- redboot
|-- services
`-- templates
```

- hal 目录是体系结构相关部分
- io 目录中是一些驱动
- language 目录中提供一些语言支持库，例如 C 库
- template 是指为某种目的 build 而建立的相应模板配置，比如选取一些包，配置一些选项等等

- **ecos.db**是一个数据库文件，配置工具从这个文件中获得各种信息
- 观察 ecos.db

目录结构分析 III

- 观察 hal 目录

```
ecos-3.0/packages/hal
|-- arm
|-- calmrisc16
|-- calmrisc32
|-- common
|-- cortexm
|-- fr30
|-- frv
|-- h8300
|-- i386
|-- m68k
|-- mips
|-- mn10300
|-- powerpc
|-- sh
|-- sparc
|-- sparclite
|-- synth
`-- v85x
```

目录结构分析 IV

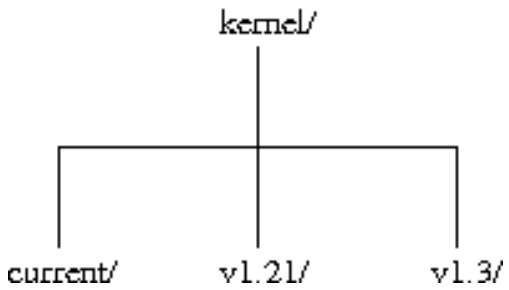
- 观察 hal/i386 目录

```
ecos-3.0/packages/hal/i386/
|-- arch
|-- generic
|-- pc
`-- pcmb
```

- 观察 hal/arm 目录

```
ecos-3.0/packages/hal/arm
|-- aeb
|-- aim711
|-- arch
|-- arm9
|-- at91
|-- cma230
|-- e7t
|-- ebsa285
|-- edb7xxx
|-- gps4020
|-- integrator
|-- lpc24xx
|-- lpc2xxx
|-- mac7100
|-- pid
|-- sa11x0
|-- snds
`-- xscale
```

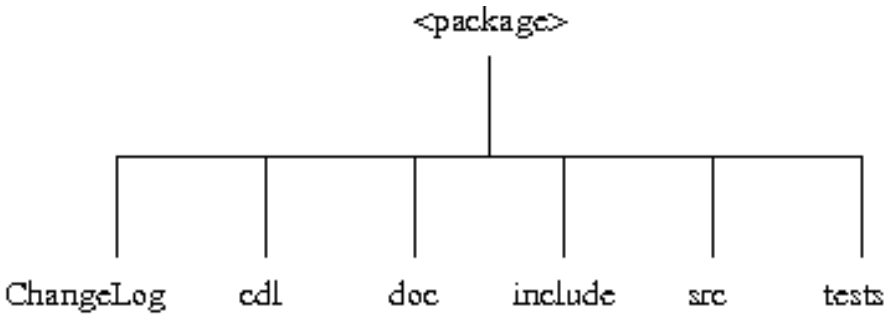
Package 的版本



- 一般 current 最新，多用于 ecos 开发者
- 我们所使用的版本

```
ecos-3.0/packages/kernel/
`-- v3_0
```

版本下的通用目录结构



- 对比 kernel/v3_0 的目录结构

```
ecos-3.0/packages/kernel/v3_0/
|-- ChangeLog
|-- cdl
|-- doc
|-- host
|-- include
|-- src
`-- tests
```

源文件

- 库源文件
- 链接描述文件（*.ld）
- 外部头文件
- 文档（doc 目录）
- 测试用例（tests 目录）
- CDL 脚本（组件定义语言 CDL，Component Definition Language）
  - CDL 是 eCos 组件框架的一个关键部分。
    - eCos 中所有的包都必须具有至少一个 CDL 脚本对其进行描述。
    - CDL 脚本包含了该包中所有配置选项的详细信息，并提供了如何对该包进行编译的信息。
- ChangeLog

i386 相关 I

```
ecos-3.0/packages/hal/i386/
|-- arch
|-- generic
|-- pc
`-- pcmb
```

- arch：为 i386 处理器体系结构提供支持
- generic：为普通 IA32 处理器提供支持，从 80386→ 最新的 Pentium/Athlon。
- pcmb 中的 mb 是指 MotherBoard，即提供对 PC 主板的支持
- pc：提供标准的 pc 目标机的支持

i386 相关 II

```
ecos-3.0/packages/hal/i386/arch/v3_0/
|-- ChangeLog
|-- cdl
|   `-- hal_i386.cdl
|-- include
|   |-- arch.inc
|   |-- basetype.h
|   |-- hal_arch.h
|   |-- hal_cache.h
|   |-- hal_intr.h
|   |-- hal_io.h
|   |-- hal_smp.h
|   |-- i386.inc
|   `-- i386_stub.h
`-- src
    |-- context.S
    |-- hal_misc.c
    |-- hal_syscall.c
    |-- i386.ld
    |-- i386_stub.c
    |-- redboot_linux_exec.c
    `-- vectors.S
```

```
ecos-3.0/packages/hal/i386/generic/v3_0/
|-- ChangeLog
|-- cdl
|   `-- hal_i386_generic.cdl
|-- include
|   |-- var_arch.h
|   |-- var_intr.h
|   `-- variant.inc
`-- src
    `-- var_misc.c
```

i386 相关 III

```
ecos-3.0/packages/hal/i386/pc/v3_0/
|-- ChangeLog
|-- cdl
|   `-- hal_i386_pc.cdl
|-- doc
|   `-- RELEASENOTES.txt
|-- include
|   |-- hal_diag.h
|   |-- pkgconf
|   |-- platform.inc
|   |-- plf_arch.h
|   |-- plf_intr.h
|   |-- plf_io.h
|   |-- plf_misc.h
|   `-- plf_stub.h
|-- misc
|   |-- menu.lst
|   |-- redboot_FLOPPY.ecm
|   |-- redboot_FLOPPY_D850GB.ecm
|   |-- redboot_FLOPPY_SMP.ecm
|   |-- redboot_GRUB.ecm
|   `-- redboot_ROM.ecm
`-- src
    |-- PKGconf.mak
    |-- hal_diag.c
    |-- plf_misc.c
    |-- plf_stub.c
    |-- romboot.S
    `-- romboot.ld
```

```
ecos-3.0/packages/hal/i386/pcmb/
`-- v3_0
    |-- ChangeLog
    |-- cdl
    |   `-- hal_i386_pcmb.cdl
    |-- include
    |   |-- pcmb.inc
    |   |-- pcmb_intr.h
    |   |-- pcmb_io.h
    |   `-- pcmb_serial.h
    |-- src
    |   |-- pcmb_misc.c
    |   |-- pcmb_screen.c
    |   |-- pcmb_serial.c
    |   `-- pcmb_smp.c
    `-- support
        `-- gfxmode.c
```

redboot.bin 的生成 I

- 观察 ecos-3.0/packages/hal/i386pc/v3_0/cdl/hal_i386_pc.cdl，寻找 redboot.bin

```
cdl_option CYGBLD_BUILD_REDBOOT_BIN_FLOPPY {
    display       "Build Redboot FLOPPY binary image"
    active_if     CYGBLD_BUILD_REDBOOT
    active_if     { CYG_HAL_STARTUP == "FLOPPY" }
    calculated    1
    no_define
    description "This option enables the conversion of the Redboot
                 ELF image to a binary image suitable for
                 copying to a floppy disk."

    make -priority 325 {
        <PREFIX>/bin/redboot.bin : <PREFIX>/bin/redboot.elf
        $(OBJCOPY) -O binary $< $@
    }
}
```

redboot.bin 的生成 II

- 观察 ecos-3.0/packages/redboot/v3_0/cdl/redboot.cdl，寻找 redboot.elf

```
        make -priority 320 {
            <PREFIX>/bin/redboot.elf : $(PREFIX)/lib/target.ld
$(PREFIX)/lib/vectors.o $(PREFIX)/lib/libtarget.a $(PREFIX)/lib/libextras.a
                    @sh -c "mkdir -p $(dir $@)"
                    $(CC) -c $(INCLUDE_PATH) $(ACTUAL_CFLAGS) -o
$(PREFIX)/lib/version.o $(REPOSITORY)/$(PACKAGE)/src/version.c
                    $(CC) $(LDFLAGS) -L$(PREFIX)/lib -Ttarget.ld -o $@
$(PREFIX)/lib/version.o
        }
```

- 观察 ecos-3.0/packages/hal/i386/arch/v3_0/cdl/hal_i386.cdl，寻找 vectors.o

```
make {
    <PREFIX>/lib/vectors.o : <PACKAGE>/src/vectors.S
    $(CC) -Wp,-MD,vectors.tmp $(INCLUDE_PATH) $(CFLAGS) -c -o $@ $<
    @echo $@ ": \\" > $(notdir $@).deps
    @tail -n +2 vectors.tmp >> $(notdir $@).deps
    @echo >> $(notdir $@).deps
    @rm vectors.tmp
}
```

redboot.bin 的生成 III

- 观察生成的 virtualbox-i386/i386_build/hal/i386/arch/v3_0/vectors.o.deps 文件

```
/home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/lib/vectors.o : \
 /home/xlanchen/workspace/ecos/ecos-3.0/packages/hal/i386/arch/v3_0/src/vectors.S \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/pkgconf/system.h \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/pkgconf/hal.h \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/pkgconf/hal_i386.h \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/pkgconf/hal_i386_pc.h \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/pkgconf/hal_i386_pcmb.h \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/cyg/hal/arch.inc \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/cyg/hal/i386.inc \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/cyg/hal/variant.inc \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/cyg/hal/platform.inc \
  /home/xlanchen/workspace/ecos/virtualbox-i386/i386_install/include/cyg/hal/pcmb.inc
```

启动流程 I

- i386 PC （vmWare）
- ./ecos-3.0/packages/hal/i386/arch/v3_0/src/vectors.S →cyg_start (阅读：开始 → 结束)

```
        .text
        .globl _start

_start:
        hal_cpu_init
        hal_smp_init
        hal_diag_init
        hal_mmu_init
        hal_memc_init
        hal_intc_init
        hal_cache_init
        hal_timer_init
```

```
#ifdef CYGDBG_HAL_DEBUG_GDB_INITIAL_BREAK
        .extern breakpoint
        call breakpoint
#endif
        .extern cyg_start
        call    cyg_start

        # Hmm.  Not expecting to return from cyg_start.
1:      hlt
        jmp     1b
```

- ./ecos-3.0/packages/hal/i386/pcmb/v3_0/include/pcmb.inc : hal_cpu_init
- ./ecos-3.0/packages/redboot/v3_0/src/main.c : cyg_start

启动流程 II

- 关于 hal_cpu_init，有三个文件提供对 hal_cpu_init 的定义，按照依赖的顺序
  1. arch/v3_0/include/arch.inc

```
#ifndef CYGPKG_HAL_I386_CPU_INIT_DEFINED
        # Initialize CPU
        .macro hal_cpu_init
        .endm
#endif /* !CYGPKG_HAL_I386_CPU_INIT_DEFINED */
```

  2. pc/v3_0/include/platform.inc，此文件内部首先包含了 pcmb.inc。然后有定义：

启动流程 III

```
#ifndef CYGPKG_HAL_I386_CPU_INIT_DEFINED
»
#define CYGPKG_HAL_I386_CPU_INIT_DEFINED

##=============================================================================
## ROM and GRUB startup
##
## Although these two startup types are, on the face of it, very different,
## the actual work that needs to be done here for them both is much the same.
## In both cases the system has been initialized in real mode and the transition
## to 32 bit protected mode has been done. Here all we need to do is set up
## our own GDT and IDT, reload the segment registers and proceed.

#if defined(CYG_HAL_STARTUP_ROM) || defined(CYG_HAL_STARTUP_GRUB)

»       .macro» hal_cpu_init
```

3. pcmb/v3_0/include/pcmb.inc

```
#ifndef CYGPKG_HAL_I386_CPU_INIT_DEFINED
»
#ifdef CYG_HAL_STARTUP_FLOPPY

#define CYGPKG_HAL_I386_CPU_INIT_DEFINED

»       .macro »hal_cpu_init
```

启动流程 IV

- 显然上述 pcmb.inc 中的定义将会起作用，阅读之。
  - 准备栈
  - 加载 redboot
  - 关中断
  - 加载 GDT 表和 IDT 表
  - 切换到保护模式
  - 重置 eflags 寄存器为全 0

小结

1. u-boot
   - u-boot 简介
   - 编译 u-boot
   - 简单分析 u-boot 源码
2. RedBoot
   - RedBoot 简介
   - RedBoot 的下载、编译和运行
   - RedBoot 的简单分析
3. 小结和作业

作业：

1. 嵌入式 Linux 的软件层次。
2. 常用的嵌入式 GUI。
3. BIOS
4. 什么是 BootLaoder，其主要任务是什么？有哪些工作（操作）模式。
5. 列举 5 个 BootLoader。

Project4

- 安装 dosbox，在 dosbox 中成功编译并运行 μC/OS-II
  - 通过阅读 μC/OS-II 的源代码，掌握 μC/OS-II 的编译过程和代码组成
  - 了解各个目录下的代码在 μC/OS-II 中的作用
  - 分析 μC/OS-II 是如何在 dosbox 中运行成功的
- 安装 bochs，在 bochs 中成功引导运行 μC/OS-II 的 EX1_x86L
  - 提示：
    - 参考 linux-2.4.18 中的启动代码，编写 16 位代码作为 bootsect，加载 μC/OS-II 内核，进入保护模式，跳转到 μC/OS-II 内核入口运行
    - 改写 μC/OS-II 中的汇编代码
    - 设置好中断向量表
    - 通过直接写 VGA（0xB80000 开始），25 行 $\times$80 列，来输出信息 (格式：双字符 =ASCII 码 + 颜色)
  - 要求：
    - 添加的文件或者修改的文件使用独立的目录 / 子目录
    - 尽可能不要修改 μCOS-II 目录下的文件，但可以拷贝出来修改
    - 提供使用手册
    - 以成功运行 EX1_x86L 为标准
- 给出详细的实验报告

Thanks !

The end.