si 版本应用程序编写

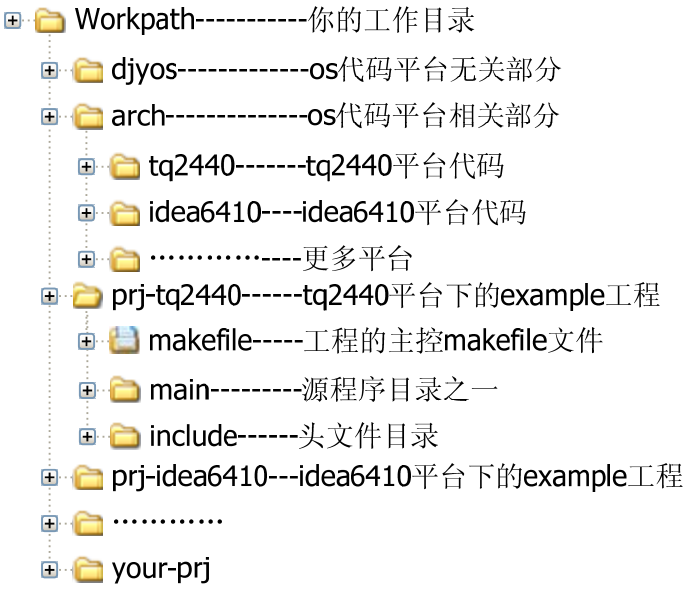
djyos 的 si 版本中，应用程序和操作系统是一起编译到一个可执行映像中的，本文档主要有两部分内容：

1、 如何添加、修改应用程序代码。

2、 编译和烧录应用程序。

1. 创建工程目录

1.1. si 版本的目录结构



si 版本目录结构

这是 djyos 整体目录结构，目录中包含了所有移植平台的代码，并且为每个平台建立了一个 example 目录，该目录下包含一个在该平台下的 example 代码，你可以只下载自己相关的平台代码。

1.2. 从 djy_main 函数开始

在操作系统代码一起提供的 example 工程下，有一个 main.c 文件，main.c 里有一个 djy_main 函数，该函数是应用程序代码的入口。djy_main 的原型是：

```
void djy_main(void);
```

应用程序代码必须提供 djy_main 函数，否则编译会出错。该函数被事件响应函数__djy_main 函数调用，__djy_main 代码如下：

```
void __djy_main(struct event_script *my_event)
{
    djy_main();
}
```

在 djyos 内部，__djy_main 实际上是一个线程的入口函数，在三星系列处理器上，操作系统为该线程分配了 8K 栈空间，也就是说，__djy_main 函数不可以使用超过 8K 的栈空间。测算一个线程需要的栈空间尺寸，请参看《都江堰操作系统与嵌入式系统设计》一书的第 2.8.2 章和第 11.3.4 章。

1.3. 创建工作目录

假定你要创建一个名为“my-prj”的工程，该工程是以 tq2440 开发板为平台的，你只需要复制 prj-tq2440 目录，并把它改名为“my-prj”就可以了。

2. 添加源程序文件

不可在“my-prj”目录下直接添加源程序文件或头文件，这样会使目录变得混乱。djyos 的 makefile 文件也不支持你这样做，除非你修改 makefile 文件。

在 main 目录下添加一个 mysrc.c 文件，你准备好自己的文件后，只需打开 main 目录下的 makefile 文件，把这一行：

source =main.c

改为

source =main.c myrsc.c

如果添加多个文件，则依次把文件名加在后面就可以了，文件名之间用空格隔开，c 文件和汇编文件都一样，但须区分大小写。

3. 添加源程序目录

假定你要在“my-prj”目录下添加一个“my-folder”文件夹，在该文件夹下有一个 mysrc1.c 文件，步骤如下：

1、 打开“my-folder”的父目录“my-prj”下的 makefile 文件，在这句中：

subdir =$(archdir) ../djyos main

添加 my-folder，改为：

subdir =$(archdir) ../djyos main my-folder

如果添加多个目录，依次把目录名加到后面，用空格隔开。

任何目录下添加子目录时，都是修改该子目录的父目录下的 makefile 文件中的 subdir 变量。

2、 从工程目录下任意一个子目录下 copy 一个 makefile 文件，修改这两行：

subdir =

source = mysrc1.c

把本目录的子目录和源程序文件列表加上。

4. 编译工程

特别提示：si 版本是应用程序和操作系统编译在一起的，图 1中的工程目录必须存在，即使 djy_main 函数是一个空函数也必须存在。

本文档假设你已经按照《建立 windows 下 djyos for arm 的编译和调试环境.doc》文件中的方法安装好 gcc 编译工具。

在 windows 下，点“开始-运行”，在运行对话框：

中输入 cmd，点“确定”进入命令行方式，然后进入源程序所在目录（makefile 文件也在这个目录中），**注意，不是进入安装 cygwin 时带的“Cygwin Bash Shell”环境**。在这个目录中执行 make 命令即可编译。假定工程目录是 g:\djyos\si\my-prj，可用下列 4 条命令分别编译不同目标文件：

g:\djyos\si\my-prj>make clean　　删除所有编译结构文件。

g:\djyos\si\my-prj>make debug　　编译产生 debug.elf 文件，未经编译优化，并且包含大量调试信息，可用于仿真调试。

g:\djyos\si\my-prj>make run_inram　编译产生 run_inram.bin，编译器设为 2 级优化，该文件可烧录到 flash 中，运行时自动 copy 到 ram 中执行。

g:\djyos\si\my-prj>make run_inflash　编译产生 run_inflash.bin，编译器设为 2 级优化，该文件可烧录到 nor、flash 中，直接在 norflash 中运行，适合像 cm3 这样的单片机。

编译产生的结果文件：

*.elf：elf 格式的可执行文件，包含符号表，还可包含更多的调试信息。

*.bin：可直接烧录的二进制可执行文件。

4.1. 何时应该使用 make clean 命令

make 有一个特点，就是如果被依赖的文件比目标文件更新的话，就会重新编译目标文件，否则不会重编译。比如两次编译间，a.c 文件没有被编辑过，那么被依赖的文件 a.c 就不比目标文件 a.o 新，因此第二次编译时 a.o 就不会被重新编译，但有时候，我们不希望这种“查新”机制起作用，就要用 make clean 命令把目标文件删掉，强制重新编译。

1、 修改头文件

原理上讲，如果依赖关系做得好，头文件也成为.o 文件的依赖文件的话，头文件修改后，编译器能正确重编译依赖该头文件的目标文件，但事与愿违，有些时候头文件与 c 源文件的依赖关系很复杂，并且在编程过程中会有调整，要完全描述这种依赖关系变得很困难，且 makefile 会被弄得很复杂难读。因此，djyos 的 makefile 不描述头文件和 C 文件的依赖关系，故在修改头文件后，需要执行 make clean 命令删掉所有.o 文件，强迫重新生成所有.o 文件。

2、 目标改变

比如原来执行过：

make debug 编译调试版本

现在要执行

make run_inram 编译可直接执行版本，由于在执行 debug 时，已经产生了.o 文件，而.c 文件又没有修改，故不会重新编译，而是直接把编译 debug 时产生的.o 文件重新连接。而 run_inram 和 debug 的命令行参数并不一样，该.o 文件是按 debug 的命令行参数编译的，不适合 run_inram 使用，必须重新按 run_inram 的参数产生所有.o 文件，故需要在执行 make run_inram 之前用 make clean 删掉所有.o 文件。

3、 源代码版本发布或归档

发布源代码时，实际上只希望发布源代码文件和.bin 文件，就要用 make clean 删掉其他编译产生的文件。