4.2 Makfile 中的变量_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 4.2 Makfile 中的变量涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

Makefile 中使用变量有点类似于 C 语言中的宏定义, 使用该变量相当于内容替换, 使用变量可以使 Makefile 易于维护, 修改起来变得简单。

- 常用的两种变量
  - 普通变量
  - 普通变量，一般是用户自己根据需求定义的变量。变量的定义要做到见名知意。
  - 示例:
  - "=" 是最普通的等号，在 Makefile 中容易搞错赋值等号，使用“=”进行赋值，变量的值是整个 Makefile 中最后被指定的值。
  - ```
    OBJ_A = A
    OBJ_B = $(OBJ_A) B
    OBJ_A = AA

    all:
        @echo "${OBJ_B}"


    #经过上面的赋值后，最后OBJ_B的值是AA B，而不是A B，
    #在make时，会把整个Makefile展开，来决定变量的值
    ```
  - “:=”表示直接赋值，赋予当前位置的值。
  - ```
    OBJ_A := A
    OBJ_B := $(OBJ_A) B
    OBJ_A := AA

    all:
        @echo "${OBJ_B}"


    # 最后BIR_B的值是A B，即根据当前位置进行赋值。因此相当于"="
    # ": ="才是真正意义上的直接赋值
    ```
  - “?=”表示如果该变量没有被赋值，赋值予等号后面的值。
  - ```
    #@VAR := A
    VAR ?= A

    all:
        @echo "${VAR}"
    ```
  - 自动变量
变量名	含义
$@	规则中的目标集合，在模式规则中，如果有多个目标的话，“$@”表示匹配模式中定义的目标集合。
$<	依赖文件集合中的第一个文件.
$^	所有依赖文件的集合，使用空格分开， 如果在依赖文件中有多个重复的文件，会去除重复的依赖文件， 只保留一份。
  - 示例代码:
  - fun.h
  - ```
    #ifndef __FUN_H__
    #define __FUN_H__
    ```

```
#include <stdio.h>

extern int global;

extern void print_value();

#endif
```

- fun.c

```
#include "fun.h"

int global = 20;

void print_value()
{
    printf("global = %d\n",global);
    return;
}
```

- main.c

```
#include "fun.h"

int main()
{
    print_value();
    return 0;
}
```

- Makefile

```
CC := gcc

#$@ :  目标的集合
#$< :  第一个依赖条件
#$^ :  所有的依赖条件

TARGET := main_exec
OBJECT := main.o fun.o

#@ 这里的$^指所有的依赖条件。即$(OBJECT)中main.o fun.o
#@ 这里$@ 指目标集合
#@ 这里$^ 指的是第一个依赖条件
$(TARGET) : $(OBJECT)
        $(CC) $^ -o $@

#@ 以下代码等价于
#@ main.o : main.c
#@   gcc -c main.c -o main.o
#@fun.o : fun.c
#@   gcc -c fun.c -o fun.o
%.o : %.c
        $(CC) -c $< -o $@

clean :
        rm -rf *.o main_exec
```

二. 课后任务

练习：

- 大家写一个链表的代码, 链表需要 linklist.c linklist.h,main.c 要求用 Makefile 自动变量来
- 编写 Makfile。 【作业截图只需要上传 Makefile 格式就行】