2.2 命令行解析

向shell提示输入命令时，shell读取输入行并解析命令行，然后分解为令牌（token）。

“解析（parsing）”是指将一连串的字符串分解为有意义的令牌，然后创建由令牌组成的解析树（parse tree）。

Token又分解为空格和TAB，命令行以新行结束。Shell先解析第一个词是内置（build-in）命令还是位于磁盘的可执行程序。若为内置命令，则在内部执行，否则就检索指定给PATH变量的目录，以此检索程序的位置。

2.3 指令类型

Shell执行别名、函数、内置命令或磁盘（存储设备）等可执行程序的命令。别名是指能在C、TC、Bash Shell中执行的命令简称。函数用于Bourne shell和Bash Shell，shell的内存中定义着别名和函数。

执行命令时，shell解析命令行，首先判断第一个词是内置命令还是存在于磁盘的执行命令。此时，判断为内置命令则直接执行，判断为磁盘的执行命令则执行fork（）系统调用创建子进程，用子进程来检索定义在PATH变量中的路径以查找命令，设置重定向、管道、命令替换、后台处理的文件描述符。子进程运行期间暂停（wait）父进程。

2.5 变量

Shell变量通常以大写字母定义，有两种变量类型：全局变量和局部变量。

2.5.1.1全局变量

所有shell都可以使用全局变量或者环境变量，使用env命令或者是printenv命令显示环境变量。

2.5.1.2局部变量

局部变量只能够用于当前的shell。直接运行set内置命令时，显示包括环境变量的所有变量和函数目录。此时显示的是排序的状态。

2.5.2 生成变量

变量默认使用大写字母，可以包含数字，但是不能以数字开头，生成变量时要注意的是在“=”号的前后不能有空格。使用unset来取消变量的定义。

2.5.3 局部变量转为环境变量

在当前shell中生成的变量只能在当前shell中使用，即局部变量。只有将该局部变量转为环境变量，才能由子进程使用。此时使用的命令是export，export以后该变量就会进入env中。然后使用unset就可以取消变量的设置，env中就不会在出现该变量。

在当前的shell中可以使用bash命令来启动子shell，子shell会继承父shell的环境变量，在子shell中对变量进行改变不会改变父shell中的值。

2.5.4 预定义变量

Bourne shell 和Bash shell都以相同的方法定义变量，都有一些预定义的变量。

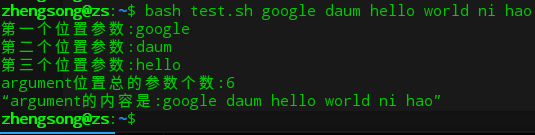
2.5.5 特殊参数变量

Shell可以执行几种特殊参数变量：

|  |  |
| --- | --- |
| $\* | 该参数变量是由1开始的位置参数的扩展，以双引号分隔，以由IFS特殊变量的第一个字符区分的每一个值扩展1个词，并具有总参数值。若IFS为非空或解除，则以空格分隔参数 |
| $@ | 该参数变量是由1开始的位置参数的扩展，若使用双引号扩展，则每一个参数扩展为1个区分词，并具有总参数值 |
| $# | 该参数变量表示十进制数位置参数的总数 |
| $? | 具有最后执行的前台管道行的终止状态 |
| $- | 执行以后立即通过set内置命令或shell自身（如-i标签）扩展为已设置的当前选项标签 |
| $$ | 拥有当前shell的进程ID |
| $! | 拥有后台运行的最新进程ID |
| $0 | 拥有shell或者shell脚本名称 |
| $\_ | Shell运行时设置该变量，拥有使用并执行参数目录的shell脚本的绝对路径，若之前已经执行某个命令，则具有已执行命令中最后一个参数的绝对路径。 |









2.6 Bourne shell

2.6.1 shbang行

执行Bourne shell脚本时，在shell脚本的第一行输入#!/bin/sh,使内核将该文件识别为Bourne shell脚本。

2.6.3 通配符

\*、？、[]用于文件名的扩展，< , > , 2>, >>, | , 字符用于标准IO。使用(‘、”)可以防止解析器解析字符。

注意：在当前shell脚本中设置的局部变量，当前脚本退出执行以后，该变量就失效了。全局变量或者说是环境变量可以用在正在运行的shell和由该shell生成的任何子进程当中，但是在脚本中设置的变量就算是export了，当脚本执行完成以后，该变量也会失效。在shell脚本中设置环境变量以后，应该要用source setenv.sh来将其加到环境中，而不是简单的执行该shell脚本。

2.6.8 读取用户的输入信息

Read命令用于读取用户的输入信息，并向read命令右侧变量分配用户输入的值。可以输入多个变量。

2.6.9 参数

参数通过命令行传递给脚本、脚本中接收每一个值时使用位置参数。与参数有关的是$1、$2、$3...用来表示参数。$\*用来显示所有参数的位置。$#显示位置参数的个数。

2.6.10 数组

Bourne shell支持参数，使用内置的set命令生成词目录，每一个词分配于每个位置顺序。可以使用9个位置参数。内置shift命令向目录左侧的第一个词移动。每一个词从1开始访问每个位置的值。

1. 数组定义，Shell使用一对括号表示数组，数组元素之间使用“空格”分隔。
2. 数组元素的读取，格式：${数组名[下标]}，下标从零开始，下标为\*或@代表整个数组的内容。
3. 数组的遍历，for each
4. 数组的长度，格式：${#数组名[\*/@]}
5. 赋值，格式：数组名[下标]=值，若下标不存在，则新增数组元素；若该下标的值已经存在，则覆盖该值
6. 分片，格式：$[数组名[\*/@]:起始位:长度]，截取部分数组，返回字符串，中间用空格分隔；将结果使用“（）”，则得到新的切片数组
7. 替换元素，格式：${数组名[\*/@]/查找字符/替换字符}，不会修改原数组；如需修改的数组，将结果使用“（）”赋给新数组
8. 删除数组，格式：unset数组名（删除整个数组），unset数组[下标]（清除单个元素）





2.6.11 替换命令

可以在命令中使用反单引号（``）,将Linux命令的结果分隔为变量或者用作字符串

变量替换可以根据变量的状态（是否为空，是否定义来改变值）

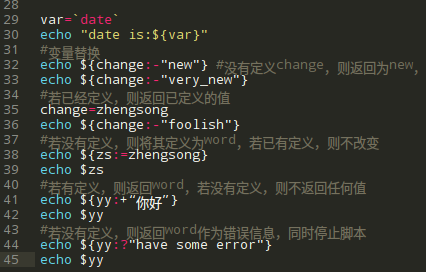
${var:-word}当var没有定义的时候，返回word，否则返回var

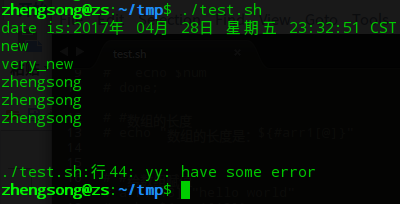
${var:=word}当var没有定义的时候，给var赋值word，否则不改变var的值

${var:+word}当var有定义的时候，返回word，但是不改变原定义，如果没有定义不返回任何值。

${var:?word}当var没有定义的时候，返回word作为错误信

printk(KERN\_ERR "usb-serial: %s - registering generic "息，同时停止脚本





2.15.8 变量扩展修饰符