总述：

shell 中有七种类型的扩展，他们分别是：花括号扩展(brace expansion)、波浪线扩展(tilde expansion)、参数和变量扩展(parameter and variable expansion)、命令替换(command substitution)、算术扩展(arithmetic expansion)、词的拆分(word splitting)和路径扩展(pathname expansion)。 进程替换(process substitution)只有在个别支持的系统中有。扩展的顺序是：花括号扩展，波浪线扩展，参数，变量和算术扩展还有命令替换(按照从左到右的顺序)，词的拆分，最后是路径扩展。

下面是shell扩展中的--参数和变量扩展(parameter and variable expansion)：

1）${parameter}

①拓展为变量的值。一是为了扩展一个大于9的位置参数，如：${10}。二是为了在变量后直接跟字符串时，不会把字符串看成变量名的一部分，如：${ver}sion。

②间接引用。作为bash 2版本的一个主要升级内容，利用${!variable}可间接引用变量，比evel var1=\$$var2高级了。

2）子串削除。单一符号是最小匹配；两个符号是最大匹配。pattern中的*（星号）统配所有字符。

①${parameter#pattern} ${parameter##pattern} 从parameter的开头位置截掉匹配的pattern。如：

parameter=/dir1/dir2/dir3/my.file.txt

${parameter#*/}： 去掉从开头到第一个 / 的所有字符串：dir1/dir2/dir3/my.file.txt

${parameter##*/}： 去掉从开头到最后一个 / 的所有字符串：my.file.txt

${parameter#/*.}： 去掉从开头到第一个 . 的所有字符串：file.txt

${parameter##/*.}： 去掉从开头到最后一个 . 的所有字符串：txt

注意：#之后开始截取的地方必须是字符串的第一个字符，在这里是 / ，或者用一个星号统配标志字符到开头所有。若标志字符在字符串中有多个，##最大匹配直到parameter中最靠后的一个pattern。

②${parameter%pattern} ${parameter%%pattern} 从parameter的结尾位置截掉匹配的pattern。如：

parameter=/dir1/dir2/dir3/my.file.txt

${parameter%/*}： 去掉最后条 / 及其右边的字符串：/dir1/dir2/dir3

${parameter%%/*}：去掉第一条 / 及其右边的字符串：(空值)

${parameter%.*t}： 去掉最后一个 . 及其右边的字符串：/dir1/dir2/dir3/my.file

${parameter%%.*t}：去掉第一个 . 及其右边的字符串：/dir1/dir2/dir3/my

注意：%之后开始截取的地方必须是字符串的最后一个字符，在这里是 t ，或者用一个星号统配

标志字符到结尾所有。若标志字符在字符串中有多个，%%最大匹配直到parameter中最靠前的一个pattern。

3）子串提取。

①${parameter:offset}，在$parameter中从位置offset处开始提取到最后。如：

${parameter:5}：提取从第五个字符到最后：/dir2/dir3/my.file.txt

${parameter:(-3)}：提取从倒数开始的3个字符：txt，用括号括起来，转义位置这个参数。

${parameter: -8}：提取从倒数开始的8个字符：file.txt，负号前面有个空格，转义位置这个参数。

②${parameter:offset:length}，在$parameter中从位置offset开始提取length长度的子串，不包括第offset个。如：

${parameter:0:5}：提取最左边的 5 个字节：/dir1

${parameter:5:5}：提取第 5 个字节右边的连续 5 个字节：/dir2

4）根据变量状态赋值。利用 ${ } 可针对不同的变量状态赋值。(未定义、空值、非空值)。

①${var-string} 若变量var未定义，则用string作传回值。(空值及非空值时返回变量var值)

②${var:-string} 若变量var未定义或为空，则用string作传回值，var本身的值不变。(非空值时返回变量var值)

③${var=string} 若变量var未定义，则用string作传回值，同时把string值赋给变量var。(空值及非空值时返回变量var值)

④${var:=string} 若变量var未定义或为空，则用string作传回值，同时把string值赋给变量var。(非空值时返回变量var值)

⑤${var+string} 若变量var为空或非空值，均使用string作传回值，var本身的值不变。(未定义时返回变量var值)

⑥${var:+string} 若变量var为非空值，则用string作传回值，var本身的值不变。(未定义或空值时返回变量var值)

⑦${var?string} 若变量var未定义，则把string输出到并准错误中，并从脚本中退出。 (空值及非空值时返回变量var值)

⑧${var:?string} 若变量var未定义或为空，则把string输出到标准错误中，并从脚本中退出。可利用此特性来检查是否设置了变量的值。(非空值时返回变量var值)

5）模式替换：${parameter/pattern/string}。如：

①${parameter/dir/path}：将第一个 dir 替换为 path：/path1/dir2/dir3/my.file.txt

②${parameter//dir/path}：将全部 dir 替换为 path：/path1/path2/path3/my.file.txt

③${parameter/#/dir/-path}：如果开头匹配 /dir 那就把开头替换成-path：-path/dir2/dir3/my.file.txt

④${parameter/%txt/doc}：如果结尾匹配 txt 那就把结尾替换成doc：/dir1/dir2/dir3/my.file.doc

⑤${@/txt/doc}或${*/txt/doc}：如果parameter是@或*，此时会轮询替换所有的位置参数的pattern为string，并列出所有结果。如果parameter数组的下标是@或*，则会轮询替换所有的数组内容的pattern为string，并列出所有的结果。

6）变量长度和个数

①${#parameter} 表示变量值的长度。如： string=string.txt;echo${#string} 可得到 10。

②${#*}和${#@} 表示位置参数的个数

③${#array[*]}和${#array[@]} 表示数组中元素的个数

7）变量大小写转换：${parameter^patten}或${parameter^^patten}和${parameter,patten}或${parameter,,patten}

①${parameter^patten}转换变量开头的第一个字符为大写。两个^^号则转换变量中所有匹配的单个字符。

②${parameter,patten}转换变量开头的第一个字符为小写。两个,,号则转换变量中所有匹配的单个字符。

③如果parameter是*或者@，则转换所有位置参数开头第一个或者是所有匹配的字符，并列出清单。

④如果parameter是array[*]或者array[@]，则转换所有数组元素开头第一个或者是所有匹配的字符，并列出清单。

⑤如果没有patten部分，则转换整个parameter为相应的大小写。

8）位置参数相关

①表示9以后的位置参数也需要带上大括号，例如${10}、${21}。

②利用参数1 2 3 4 5运行脚本，用*或@提取：

${*:2}：提取从第二个开始到最后的参数：2 3 4 5

${@:2}：同上面一样：2 3 4 5

${*:3:3}：提取第3个开始的3个位置参数：3 4 5

③大括号标记法还提供了一种提取从命令行传递到脚本的最后一个位置参数的简单办法，同时需要用上间接引用就表示为：test=${!#}。

9）匹配变量名：${!prefix*}或${!prefix@}。如：

#var1=1;var2=2;var3=3;ttt=4

#echo ${!var*} 结果就是以var开头的所有变量名：var1 var2 var3

10）列出数组下标：${!name[*]}或${!name@}。如：

array=(a asdf afwe gag fddoi wewe);echo ${!array[*]}结果为数组的下标：0 1 2 3 4 5