String Manipulations

String Length

${#string}
expr length $string

These are the equivalent of *strlen()* in *C*.

expr "$string" : '.*'

```
stringZ=abcABC123ABCabc

echo ${#stringZ}                 # 15
echo `expr length $stringZ`      # 15
echo `expr "$stringZ" : '.*'`    # 15
```

Length of Matching Substring at Beginning of String

expr match "$string" '$substring'

$substring is a regular expression.

expr "$string" : '$substring'

$substring is a regular expression.

```
stringZ=abcABC123ABCabc
#       |------|
#       12345678

echo `expr match "$stringZ" 'abc[A-Z]*.2'`   # 8
echo `expr "$stringZ" : 'abc[A-Z]*.2'`       # 8
```

Index

expr index $string $substring

Numerical position in $string of first character in $substring that matches.

```
stringZ=abcABC123ABCabc
#       123456 ...
echo `expr index "$stringZ" C12`             # 6
                                             # C position.

echo `expr index "$stringZ" 1c`              # 3
# 'c' (in #3 position) matches before '1'.
```

This is the near equivalent of *strchr()* in *C*.

Substring Extraction

${string:position}

Extracts substring from *$string* at *$position*.

If the $string parameter is "*" or "@", then this extracts the positional parameters, starting at $position.

${string:position:length}

Extracts *$length* characters of substring from *$string* at *$position*.

```
stringZ=abcABC123ABCabc
#       0123456789.....
#       0-based indexing.

echo ${stringZ:0}                            # abcABC123ABCabc
echo ${stringZ:1}                            # bcABC123ABCabc
echo ${stringZ:7}                            # 23ABCabc

echo ${stringZ:7:3}                          # 23A
                                             # Three characters of
substring.



# Is it possible to index from the right end of the string?

echo ${stringZ:-4}                           # abcABC123ABCabc
# Defaults to full string, as in ${parameter:-default}.
# However . . .

echo ${stringZ:(-4)}                         # Cabc
echo ${stringZ: -4}                          # Cabc
# Now, it works.
# Parentheses or added space "escape" the position parameter.
```

The *position* and *length* arguments can be "parameterized," that is, represented as a variable, rather than as a numerical constant.

If the $string parameter is "*" or "@", then this extracts a maximum of $length positional parameters, starting at $position.

```
echo ${*:2}          # Echoes second and following positional
parameters.
echo ${@:2}          # Same as above.

echo ${*:2:3}        # Echoes three positional parameters, starting
at second.
```

expr substr $string $position $length

Extracts *$length* characters from *$string* starting at *$position*.

```
stringZ=abcABC123ABCabc
#       123456789......
#       1-based indexing.

echo `expr substr $stringZ 1 2`              # ab
echo `expr substr $stringZ 4 3`              # ABC
```

expr match "$string" '\($substring\)'

Extracts *$substring* at beginning of *$string*, where *$substring* is a regular expression.

expr "$string" : '\($substring\)'

Extracts *$substring* at beginning of *$string*, where *$substring* is a regular expression.

```
stringZ=abcABC123ABCabc
#       =======

echo `expr match "$stringZ" '\(.[b-c]*[A-Z]..[0-9]\)'`   # abcABC1
echo `expr "$stringZ" : '\(.[b-c]*[A-Z]..[0-9]\)'`       # abcABC1
echo `expr "$stringZ" : '\(.......\)'`                   # abcABC1
# All of the above forms give an identical result.
```

expr match "$string" '.*\($substring\)'

Extracts *$substring* at *end* of *$string*, where *$substring* is a regular expression.

expr "$string" : '.*\($substring\)'

Extracts *$substring* at *end* of *$string*, where *$substring* is a regular expression.

```
stringZ=abcABC123ABCabc
#               ======

echo `expr match "$stringZ" '.*\([A-C][A-C][A-C][a-c]*\)'`    #
ABCabc
```

```
echo `expr "$stringZ" : '.*\(......\)'`                        #
ABCabc
```

Substring Removal

${string#substring}

Deletes shortest match of *$substring* from *front* of *$string*.

${string##substring}

Deletes longest match of *$substring* from *front* of *$string*.

```
stringZ=abcABC123ABCabc
#       |----|          shortest
#       |----------|    longest

echo ${stringZ#a*C}      # 123ABCabc
# Strip out shortest match between 'a' and 'C'.

echo ${stringZ##a*C}     # abc
# Strip out longest match between 'a' and 'C'.



# You can parameterize the substrings.

X='a*C'

echo ${stringZ#$X}       # 123ABCabc
echo ${stringZ##$X}      # abc
                         # As above.
```

${string%substring}

Deletes shortest match of *$substring* from *back* of *$string*.

For example:

```
# Rename all filenames in $PWD with "TXT" suffix to a "txt" suffix.
# For example, "file1.TXT" becomes "file1.txt" . . .

SUFF=TXT
suff=txt

for i in $(ls *.$SUFF)
do
  mv -f $i ${i%.$SUFF}.$suff
  #  Leave unchanged everything *except* the shortest pattern match
  #+ starting from the right-hand-side of the variable $i . . .
done ###
```

${string%%substring}

Deletes longest match of *$substring* from *back* of *$string*.

```
stringZ=abcABC123ABCabc
#                    ||     shortest
#        |------------|     longest

echo ${stringZ%b*c}      # abcABC123ABCa
# Strip out shortest match between 'b' and 'c', from back of
$stringZ.

echo ${stringZ%%b*c}     # a
# Strip out longest match between 'b' and 'c', from back of $stringZ.
```

This operator is useful for generating filenames.