csh

https://www.jb51.net/article/57770.htm

https://www.computerhope.com/unix/ucsh.htm

https://docslib.org/doc/12944229/c-shell-cookbook

https://starlink.eao.hawaii.edu/devdocs/sc4.pdf ---

cshell

1. csh

```
#
./myscript     #
/home/xxx/myscript    #

# source xxx
source ./myscript    #  source

#  source xxx              shell           alias
              alias       shell
#       shell

#                    .cshrc                        shell

#--               shell                     alias

#csh

#                                          #!/bin/tcsh
```

2. csh

21   1

a.csh

```
#!/bin/csh
```

```
set a = ${#argv}

echo "argv number = $a"

echo $0 $1 $2 $3; #              3
echo "argv: $argv[1] $argv[2] $argv[3]"; # $argv[1]   $1

echo ${0}    #
```

```
    :
./a.csh 1 2 3

argv number = 3
./a.csh 1 2 3
argv: 1 2 3
./a.csh
```

22   2

```
#!/bin/csh

set files = ~/*; #    home
foreach tmp ($files)
    if (-f $tmp) then
        set bname = `basename $tmp`
        echo "normal file: $tmp, basename = $bname"
    else if(-d $tmp) then
        echo "dir: $tmp, basename = $bname"
    endif
    sleep 1; #   1s
    sleep 1;
end
```

3

```
set x = 5
echo $x;  #    5
echo ${x}; #    5
echo ${x}kg; #        x    kg,        5kg
echo $%x; #$%x

$#name  name
${#argv}   argv
$#argv,    argv
$argv[1],  argv    1               arg
$argv[0]   argv    0
```

```
set i = 0
@ i = $i + 1 #
@ i = $i - 1 #
@ i = $i * 2 #
@ i = $i / 2 #
#    C++
@ i++ #
@ i-- #
@ i+=2 #    2
@ i-=2 #    2
@ i*=2 #    2
echo $i
```

3.1

- argv　　　　　　　　$0　shell　$1　　　　$argv[1]
- cwd,
- path　path
- home　　　　~,　　home
- prompt　

3.2 cshell prompt

```
prompt        The string which is printed before reading each command from
the terminal. prompt may
include any of the following formatting sequences (+), which are replaced by
the given information:

%/            The current working directory.
%?            The current working directory, but with one's home directory
represented by '?' and other users' home directories represented by '?user'
as per Filename substitution. '?user' substitution happens only if the shell
has already used '?user' inapathname in the current session.

%c[[0]n], %.[[0]n]    The trailing component of the current working
directory, or n trailing components if a digit n is given. If n begins with
'0', the number of skipped components precede the trailing component(s) in
the format '/<skipped>trailing'. If the ellipsis shell variable is set,
skipped components are represented by an ellipsis so the whole becomes
'...trailing'. '?' substitution is done as in '%?' above, but the '?'
component is ignored when counting trailing components.

%C Like %c,   but without '?' substitution.
%h, %!, !     The current history event number.
%M            The full hostname.
%m            The hostname up to the first '.'.
%S (%s)       Start (stop) standout mode.
%B (%b)       Start (stop) boldfacing mode.
%U (%u)       Start (stop) underline mode.
```

```
%t, %@           The time of day in 12-hour AM/PM format.
%T     Like '%t', but in 24-hour format (but see the ampm shell variable).
%p     The 'precise' time of day in 12-hour AM/PM format, with seconds.
%P     Like '%p', but in 24-hour format (but see the ampm shell variable).
\c     c is parsed as in bindkey.
?c     c is parsed as in bindkey.
%%     A single '%'.
%n     The user name.
%d     The weekday in 'Day' format.
%D     The day in 'dd' format.
%w     The month in 'Mon' format.
%W     The month in 'mm' format.
%y     The year in 'yy' format.
%Y     The year in 'yyyy' format.
%l     The shell's tty.
%L     Clears from the end of the prompt to end of the display or the end of
the line.
%$     Expands the shell or environment variable name immediately after the
'$'.
%#     '>' (or the first character of the promptchars shell variable) for
normal users, '#' (or the second character of promptchars) for the
superuser.
%{string%}
Includes string as a literal escape sequence. It should be used only to
change terminal
attributes and should not move the cursor location. This cannot be the last
sequence in
prompt.
%?     The return code of the command executed just before the prompt.
%R     In prompt2, the status of the parser. In prompt3, the corrected
string. In history, the history string.  '%B', '%S', '%U' and '%{string%}'
are available in only eight-bit-clean shells; see the version shell
variable.


The bold, standout and underline sequences are often used to distinguish a
superuser shell. For
example,
> set prompt = "%m [%h] %B[%@]%b [%/] you rang? "
tut [37] [2:54pm] [/usr/accts/sys] you rang? _

If '%t', '%@', '%T', '%p', or '%P' is used, and noding is not set, then
print 'DING!' on the change of hour (i.e, ':00' minutes) instead of the
actual time.

Set by default to '%# ' in interactive shells.
```

33   **perl**

```
set a = "123"
```

```
set a = `echo $a | perl -pe "s#123#456#"`

echo $a
#       a        456      perl



set cl = "123"
set cl_type = `echo $cl | perl -e '$_ = <>; if (/^$/) { print "none"} elsif
(/^\d\+$/) { print "num"} else { print "else" }'`
echo $cl_type
# cl_type      num
```

4

```
#
set arr = ()

#           3
set myarr = (str1, str2,str3)
echo $myarr[2]
echo $myarr
echo $myarr[*]

set files =  *
```

```
set myarr = ($myarr str4)
```

shift

```
# array test
set ar = (0 1 2 3 4)
echo $#ar
echo ${#ar}
echo $ar[1]   # index [1]
echo $ar[1-3]
echo $ar[*]
shift ar
echo $ar[*]
```

```
5
5
0
0 1 2
0 1 2 3 4
1 2 3 4
```

5. if/else/switch/case

```
if ($n < 5 || 20 <= $n) then
endif

if ($n < 5 && 20 <= $n) then
endif

if(expression)then
   commands
endif

if {(command)} then
   commands
endif


if(expression) then
   commands
else if(expression) then
   commands
else
   commands
endif


switch("$value")
case pattern1:
   commands
   breaksw
case pattern2:
   commands
   breaksw
default:
   commands
   breaksw
endsw
```

6. while/foreach

```
while(expression)
   #      c++
   @ i++ #
   @ i-- #
   @ i+=2 #   2
   @ i-=2 #   2
   @ i*=2 #   2
```

```
  commands
  continue
  break

end

set wordlist = (a b c)
foreach var ($wordlist)
  echo $var
end
```

7.

: -e -f

```
#!/bin/tcsh

#
if (! -e ./abc/abc1) then
endif

#           ,
if (! -e ./*/abc1) then
endif
```

```
1. shell        ,
2. #!/bin/sh
3.
4. myPath="/var/log/httpd/"
5. myFile="/var /log/httpd/access.log"
6.
7. #       -x          $myPath
8. if [ ! -x "$myPath"]; then
9. mkdir "$myPath"
10. fi
11.
12. #      -d          $myPath
13. if [ ! -d "$myPath"]; then
14. mkdir "$myPath"
15. fi
16.
17. #      -f          $myFile
18. if [ ! -f "$myFile" ]; then
19. touch "$myFile"
20. fi
21.
22. #              -n,-n
23. if [ ! -n "$myVar" ]; then
24. echo "$myVar is empty"
```

```
25. exit 0
26. fi
27.
28. #
29. if [ "$var1" = "$var2" ]; then
30. echo '$var1 eq $var2'
31. else
32. echo '$var1 not eq $var2'
33. fi
```

-f -e
Conditional Logic on Files

-a **file** exists.
-b **file** exists and is a block special file.
-c **file** exists and is a character special file.
-d **file** exists and is a directory.
-e **file** exists (just the same **as** -a).
-f **file** exists and is a regular file.
-g **file** exists and has its setgid(2) bit set.
-G **file** exists and has the same group ID **as** this process.
-k **file** exists and has its sticky bit set.
-L **file** exists and is a symbolic link.
-n string length is not zero.
-o Named option is set on.
-O **file** exists and is owned by the user ID of this process.
-p **file** exists and is a first in, first out (FIFO) special **file** or
named pipe.
-r **file** exists and is readable by the current process.
-s **file** exists and has a **size** greater than zero.
-S **file** exists and is a socket.
-t **file** descriptor number fildes is open and associated with a
terminal device.
-u **file** exists and has its setuid(2) bit set.
-w **file** exists and is writable by the current process.
-x **file** exists and is executable by the current process.
-z string length is zero.

    -s   -f

csh