

# Shell Information

*#!/bin/dash*

first line of file

*command > filename*

write output to *filename*

*command >> file*

append output to *filename*

*command 2> filename*

write stderr to *filename*

*command 2>&1*

write stderr to stdout

*command > file 2>&1*

write stdout and stderr to *filename*

*command < filename*

input from *filename*

*command << EOF*

heredoc until *EOF*

*command<sub>1</sub> | command<sub>2</sub>*

pipe output from *command<sub>1</sub>*  
as input to *command<sub>2</sub>*

*command<sub>1</sub> && command<sub>2</sub>*

execute *command<sub>2</sub>* if *command<sub>1</sub>*  
has exit status zero

*command<sub>1</sub> || command<sub>2</sub>*

execute *command<sub>2</sub>* if *command<sub>1</sub>*  
does not have exit status zero

`$(expression)`  
    *expression* evaluated as arithmetic

`$#` = count of command-line arguments  
`$0` = name of currently executing command  
`$1,$2,$3,...,$9,${10},...,${255}` = command-line arguments  
`@` = list of all command-line arguments  
`*` = list of all command-line arguments  
`?` = exit status of previous command

`read varName`  
    sets value of variable *varName* to  
    next line read from `stdin`

`'str' = str`  
`"str" = str with variables interpolated`  
`'command' = output of command as string`  
`$(command) = output of command as string`

Zero exit status means true/successful  
Non-zero exit status means false/failure

`test expression`  
`[ expression ]`  
    returns *expression* result as exit status  
    integer operators: `-lt,-gt,-eq,-ne,-ge,-le`  
    string operators: `=, -z, -n`  
    file operators: `-d, -e, -f, -s, -nt`

`exit Number`  
    terminate script with exit status *Number*

`if Commanda ; then`  
    *Commands<sub>1</sub>*  
`elif Commandb ; then`  
    *Commands<sub>2</sub>*  
`else`  
    *Commands<sub>3</sub>*;

```

fi

case Word in
  Pattern1) Commands1 ;;
  Pattern2) Commands2 ;;
  ...
  *)          Commandsn ;;
esac

while Command ; do
  Commands
done

for var in Word1 Word2 ...
do
  Commands
done

# Display lines from file
count=0
while read line
do
  count=$((count + 1))
  echo "Line $count: $line"
done <file

# Interactively rm files in current dir
for f in *
do
  echo -n "Remove $f? "
  read answer
  if test $answer = y
  then
    echo $f
  fi
done

```

## Regular Expressions

Atomic Patterns:

letters, digits, punctuation (except those below)

match any occurrence of themselves

`\. \* \+ \? \| \^ \$ \[ \]`

match any occurrence of the second character

`.` (dot)

matches any single character

`(pattern)`

matches *pattern*

Anchors:

`^pattern`

matches *pattern* at the start of a line

`pattern$`

matches *pattern* at the end of a line

Selection:

`[charList]`

matches any single character in *charList*

`[^charList]`

matches any single character not in *charList*

`pattern1|pattern2|pattern3|...`

matches any of the *pattern*<sub>*i*</sub>s

*charLists* use *c*<sub>1</sub>-*c*<sub>2</sub> to denote char ranges, and meta-characters lose their special meaning inside *charLists*

Repetition:

`pattern?`

zero or one occurrences of *pattern*

`pattern*`

zero or more occurrences of *pattern*

`pattern+`

one or more occurrences of *pattern*

`\w` matches alphanumeric, including `'_'`

`\s` matches whitespace

`\d` matches numeric

`\b` word boundary

*pattern*{*N*,*M*}

matches *N* to *M* occurrences of *pattern*