

☉ Shell Variables

\$0	Name of script
\$1 - \$10	Positional parameters #1 - #10
\$#	Number of positional parameters
\"\$*"	All parameters (single word)
"\$@"	All parameters (separate strings)
\${#*}	Number of parameters
\${#@}	
\$?	Return value
\$\$	Process ID (PID)
\$-	Flags (using <code>set</code>)
_	Last argument of previous command
!	PID of last job run in background

* Must be quoted, otherwise it defaults to "\$@".

☉ Substitution/Expansion

\${VAR}	Value of VAR
\${VAR-word}	If VAR <i>not set</i> , use word
\${VAR:-word}	If VAR <i>not set</i> or <i>null</i> , use word
\${VAR:=word}	Assign default value.
\${VAR=word}	(position./spec.param. may not be assigned)
\${VAR:?ERR_MSG}	Display Error Message if <i>null</i> or <i>unset</i>
\${VAR?ERR_MSG}	
\${VAR:+word}	Use Alternate: <i>null/unset</i> - nothing, else - word
\${VAR:offset}	Substring
\${VAR:offset:length}	Substring with size length
\${!VAR[@]}	List Array Keys
\${!VAR[*]}	
\${#VAR}	Parameter length
\${VAR#match}	Remove prefix
\${VAR##match}	Remove prefix (longest)
\${VAR%match}	Remove suffix
\${VAR%%match}	Remove suffix (longest)
\${VAR/pattern/str}	Replace first match of pattern with str

☉ Substitution/Expansion (cont)

\${VAR/pattern/str}	Replace all match of pattern with str
\${VAR^pattern}	Uppercase match
\${VAR^^pattern}	Uppercase all
\${VAR, pattern}	Lowercase match
\${VAR,, pattern}	Lowercase all
\${!VAR*}	Match all vars beginning with VAR
\${!VAR@}	

☉ HISTORY MANIPULATION

!!	Last command
!foo	Last command containing foo
^foo^bar^	Last command containing foo, but substitute bar
!!:0	Last command word
!!:^	Last command's first argument
!\$	Last command's last argument
!!:*	Last command's arguments
!!:x-y	Arguments x to y of last command
C-s	search forwards in history
C-r	search backwards in history

C - Control Key

☉ DIRECTORIES

~-	Previous working directory
pushd tmp	Push tmp && cd tmp
popd	Pop && cd

☉ GLOBBING

ls a [b-dx] e	Globs abe, ace, ade, axe
ls a \{c,b1\} e	Globs ace, able
\$(ls)	`ls` (but nestable!)

☉ Redirections

cmd > file	Redirect the standard output (stdout) of cmd to a file.
cmd 1> file	
cmd 2> file	
cmd >> file	
cmd 2>> file	
cmd &> file	



🕒 Redirections (cont)

```
cmd > file 2>&1
```

```
cmd > /dev/null
```

```
cmd 2> /dev/null
```

```
cmd &> /dev/null
```

```
cmd < file
```

```
cmd << EOL
```

```
foo
```

```
bar
```

```
EOL
```

```
cmd <<- EOL
```

```
<tab>foo
```

```
<tab><tab>bar
```

```
EOL
```

```
cmd <<< "string"
```

```
exec 2> file
```

```
exec 3< file
```

```
exec 3> file
```

```
exec 3<> file
```

```
exec 3>&-
```

```
exec 4>&3
```

```
exec 4>&3-
```

```
echo "foo" >&3
```

```
cat <&3
```

```
(cmd1; cmd2) > file
```

```
{ cmd1; cmd2; } > file
```

```
exec 3<> /dev/tcp/host/port
```

```
exec 3<> /dev/udp/host/port
```

```
cmd <(cmd1)
```

```
cmd < <(cmd1)
```

```
cmd <(cmd1) <(cmd2)
```

```
cmd1 >(cmd2)
```

```
cmd1 | cmd2
```

```
cmd1 |& cmd2
```

```
cmd | tee file
```

```
exec {filew}> file
```

```
cmd 3>&1 1>&2 2>&3
```

```
cmd > >(cmd1) 2> >(cmd2)
```

```
cmd1 | cmd2 | cmd3 | cmd4
```

```
echo ${PIPESTATUS[@]}
```

