

Zentralinstitut für Angewandte Mathematik  
D-52425 Jülich, Tel. (02461) 61-6402

Informationszentrum, Tel. (02461) 61-6658

Referenzkarte

KFA-ZAM-RFK-0011

U. Schmidt

15.11.94

## Korn Shell Programming

Note: In general brackets are used to indicate that the information can be omitted. If brackets are part of the clause they are marked by \*.

### Invocation and initialization

- (1) **ksh** [*options*] [*filename*] [*args*]  
A new shell is created and if *filename* is specified, the commands in *filename* will be executed.
- (2) **filename** [*args*]  
A new shell is created and the commands in *filename* are executed.
- (3) **.** *filename*  
The commands in *filename* are executed within the current shell.

Initialization files:

<b>/etc/profile</b>	System default shell startup. Executed only at login time.
<b>\$HOME/.profile</b>	User's shell startup. Executed only at login time.
<b>\$ENV</b>	Name of ksh startup file. Executed each time a new shell is created. Normally the name of a file, which then calls <b>\$HOME/.kshrc</b>

### Command syntax

<b>#</b>	Following text on the line is comment
<b>\</b>	Use \ as last character to indicate that a continuation line is following
<b>cmd1; cmd2</b>	Run cmd1, then run cmd2
<b>cmd1 &amp;&amp; cmd2</b>	Run cmd1, then run cmd2 only, if cmd1 succeeded
<b>cmd1    cmd2</b>	Run cmd1, then run cmd2 only, if cmd1 failed
<b>cmd1   cmd2</b>	Define a pipeline

### User defined shell variables

#### Simple variables:

Assignment: **NAME=value**  
Reference: **\$NAME**

#### Array variables:

Assignment: **NAME[index]=value\***  
Reference: **\$NAME[index]\***

#### Commands:

To use variables in subsequently called shells, they must be *exported*.  
**export** *NAME[=value]* Add variable *NAME* to export list  
**env** List all exported variables

### Special \$ variables

<b>\$0</b>	Name of command/script currently executed
<b>\$n</b> or <b>\${nn}</b>	Value of parameter at position <i>n</i> ; for <i>n</i> greater 9, enclose <i>nn</i> in braces
<b>*</b>	String with all positional parameters
<b>#</b>	Number of positional parameters
<b>?</b>	Exit status of last executed command
<b>\$\$</b>	Process ID of actual shell. Used to generate unique file names.

### Shell environment variables used by Korn shell

<b>HOME</b>	User's home directory
<b>PATH</b>	List of names of directories to search for executable commands
<b>PS1</b>	User's initial prompt string
<b>PS3</b>	Prompt string for select; default '#?'
<b>PS4</b>	Prompt string for set -xv; default '+'
<b>USER</b>	User's login name
<b>TERM</b>	terminal type
<b>DISPLAY</b>	X11 server display

### Shell environment variables set by Korn shell

<b>LINENO</b>	Line number of the current script
<b>OPTARG</b>	Value of last option processed by getopt
<b>OPTIND</b>	Index of last option processed by getopt
<b>PPID</b>	Process ID of the parent shell
<b>PWD</b>	Current working directory
<b>RANDOM</b>	A random number between 0 and 32767
<b>REPLY</b>	Set by the select command

### Shell Parameter substitution

**\${#name}**  
**\${#name[index]}\***  
Length of the value of variable *name* or *\$n*, or array element with index *index*

**\${#name[\*]}\***  
Number of defined elements in the array *name*

**\${name:=value}**  
If *name* isn't null, then it is used; otherwise *value* is used and assigned to *name*.

**\${name:-value}**  
If *name* isn't null, then it is used; otherwise *value* is used but not assigned to *name*.

**\${name:+value}**  
If *name* is null, then null is used; otherwise *value* is used but not assigned to *name*.

**\${name#value}**  
**\${name##value}**  
Search for pattern is done from left to right (beginning) and the rightmost string without pattern is substituted. The 1. form splits at the 1., the 2. form at the last occurrence.

**\${name%value}**  
**\${name%%value}**  
Search for pattern is done from right to left (ending) and the leftmost string without pattern is substituted. The 1. form splits at the 1., the 2. form at the last occurrence.

### Shell arithmetic

- (1) **expr** *expression*  
**name=`expr expression`**  
Bourne compatible. Each element of *expression* must be separated by blanks.
- (2) **typeset -i** *name*  
**name=expression**  
The elements of *expression* must be coded as one string without blanks.

*Expression* is a combination of *terms* and *operators*, with or without parenthesis. Each *term* may be an integer variable or integer constant.

**term op term [ op term [...]]**  
arithmetic operators are: \* / % + - << >>  
comparison operators are: < <= > >= == !=  
logical operators are: & ^ | && ||

Notice that some operators must be quoted or backslashed to avoid confusion with the wild card characters or shell symbols for I/O redirection.

## Conditional expressions

### ... for files; true if

<b>-a file</b>	<i>file</i> exists
<b>-d file</b>	<i>file</i> exists and is a directory
<b>-f file</b>	<i>file</i> exists and is an ordinary file
<b>-r file</b>	<i>file</i> exists and is readable
<b>-s file</b>	<i>file</i> exists and has a size greater than 0
<b>-w file</b>	<i>file</i> exists and is writable
<b>-x file</b>	<i>file</i> exists and is executable
<b>-L file</b>	<i>file</i> exists and is a symbolic link
<b>-O file</b>	<i>file</i> exists and owned by user
<b>file1 -nt file2</b>	<i>file1</i> exists and is newer than <i>file2</i>
<b>file1 -ot file2</b>	<i>file1</i> exists and is older than <i>file2</i>

### ... for strings: true if

<b>-n str</b>	<i>string</i> length is greater than 0
<b>-z str</b>	<i>string</i> length is zero
<b>str1 = str2</b>	<i>string1</i> matches <i>string2</i>
<b>str1 != str2</b>	<i>string1</i> does not match <i>string2</i>
<b>str1 &lt; str2</b>	<i>string1</i> comes before <i>string2</i> (ASCII)
<b>str1 &gt; str2</b>	<i>string1</i> comes after <i>string2</i> (ASCII)

### ... for integer values: true if

<b>n1 -eq n2</b>	<i>n1</i> is equal to <i>n2</i>
<b>n1 -ne n2</b>	<i>n1</i> is not equal to <i>n2</i>
<b>n1 -lt n2</b>	<i>n1</i> is less than <i>n2</i>
<b>n1 -le n2</b>	<i>n1</i> is less than or equal to <i>n2</i>
<b>n1 -gt n2</b>	<i>n1</i> is greater than <i>n2</i>
<b>n1 -ge n2</b>	<i>n1</i> is greater than or equal to <i>n2</i>

## Wild card characters and pattern

<b>*</b>	Any string, including nullstring
<b>?</b>	Any single character
<b>[abc]</b>	Any of the enclosed characters abc
<b>[a-z]</b>	Any of the enclosed characters in the range a through z

## Control statements

### ... for conditional expressions

<b>test expression</b> or <b>[ expression ]*</b>	<i>expression</i> may be a compound expression with following operators: ! (NOT), -a (AND) or -o (OR).
<b>[[ expression ]*</b>	<i>expression</i> may be a compound expression with following operators: ! (NOT), && (AND) or    (OR).
<b>if list1; then list2; [else list3;] fi</b>	
<b>if list1; then list2; elif list3; [elif list4; [...]] [else listn;] fi</b>	
<b>case word in [ ([] pattern1 [ ] pattern2) list ;;] [...] esac</b>	
<b>select var [ in words ]; do list; done</b>	

### ... for loops

<b>for var [ in words ]; do list; done</b>	
<b>while list1; do list2; done</b>	
<b>until list1; do list2; done</b>	

### ... for functions

<b>function_identifier ( ) { list; }</b>	
--	--

## Command line editing (History)

<ESC> k [k] [j] Scrolling within shell history to retrieve commands. **k**: fetch previous command, up; **j**: fetch next command, down. If there are problems on terminal emulation within the net, use **set -o vi**.

Command mode (activated by <ESC>)

<b>h</b>	move cursor back, left, one character
<b>l</b>	move cursor forward, right, one character
<b>w</b>	move cursor to begin of next word
<b>b</b>	move cursor to begin of previous word
<b>x</b>	delete character at cursor
<b>D</b>	delete rest of line starting at cursor

Input mode

<b>i   &gt;</b>	insert characters before   after cursor
<b>r</b>	replace one character at cursor
<b>R</b>	replace characters starting at cursor
<b>A</b>	append characters at end of line

for more information see vi-RFK-0010

## Builtin commands

<b>:</b>	Nullstatement
<b>alias [-tx] [name[=value]]</b>	List or define an abbreviation
<b>break [n]</b>	Escape from loops or case
<b>cd [dirname]</b>	Change the current directory
<b>continue [n]</b>	Start next iteration of a loop
<b>echo [arg]</b>	Write arguments to standard output
<b>eval [arg]</b>	Evaluate arguments
<b>exit [n]</b>	Exit from current shell
<b>getopts optstr name [arg]</b>	Parse command line options and arguments
<b>let expression</b> or <b>((expression))</b>	Evaluate an arithmetic expression
<b>pwd</b>	Print working directory
<b>read [arg]</b>	Read a line from standard input
<b>readonly [name]</b>	Prevent alteration of selected variables
<b>return [n]</b>	Return from a shell function
<b>set [options] [arg]</b>	Set shell options and \$n variables by arguments; 'set -xv' enables tracing and uses PS4
<b>shift [n]</b>	Shift arguments left
<b>test expression</b> or <b>[ expression ]*</b>	Evaluate a conditional expression
<b>trap [omd] [signal ...]</b>	Specify exceptional condition handling
<b>type [name]</b>	Identify a command
<b>typeset options [name=[value]]</b>	Define characteristics of a variable; <i>options</i> may be <b>-i</b> for integer, <b>-l</b> for lowercase, <b>-u</b> for uppercase, <b>-Ln</b> for left substring, <b>-Rn</b> for right substring
<b>unalias name</b>	Drop an alias
<b>umask [ nnn ]</b>	Set or display the file creation mask
<b>unset name</b>	Drop a shell variable

## Information at KFA

Information Centre / Dispatch:  
Phone: +49-2461-61-6658 or 5642  
Telefax: +49-2461-61-2810  
E-Mail: zam.beratung@kfa-juelich.de