BASH REFERENCE

CONTENTS

Ailasing 6
Air.hmte Eca. . Jua it on 8
Ar ays 3 1
Bae . . Epm. . sion 6
Bulit-h. Commad. s 6 1
Commad. Line Arg.mets 3
Commad. Sbs.tit uit.on 8
Conditi. ona l Exp. es. ions 5 1
Cont.ol Commad. s 4 1
Déf.niti. on.s 2
Ecu . . it on. Orde 3 1
FieSp .ilt ing 8
Fin .ct.ions 2 1
History.Sbst tiuit.on 5
Inpu .t.Outp.ut 3 1
Invo. ea it ead . . Sta .up 3
Jdo. Idad . . Jdo .Con.t.ob 24
Opit.on SSo . ste 22
Opit.on SSo . shpt 23
Opit.on SSo . tet 21
Pa.tme 9
Re. -Déf.ned . Vai.ba.le 0 1
Res S . . . . ub.stti uit.on 8
Re. mpit.ing 4
Quo ing 6
Rea dl.ne 25
Rea dl.ne .Die it ves 25
Rea dl.ne Key .Binding 25
Rea dl.ne Vai.ba.le 26
Rest.it ed .ba sh 2
Siga .lsa nd. Tra .s 3 1
Spe .ial Cha a 24
Tie .Sbst tiuit.on 6
Vai.ba Ass. ig.me nt 9
Vai.ba Bea . .mes 9
Vai.ba Se. ub.stti uit.on 7

This efee rd wa. wtir e y Arnold Rbb insWe thak ChPa mey (ba sh's maintaine)r fro ise lp

OTHER SOFTWARE PRODUCTS:

Spe ial ed Systems Consu tlants, nc
(206) FOR-UNIX / (206) 782-7733
FAX: (206) 782-7191
E-mail: sales@ssc.com
URL: http://www.ssc.com

Linux Jou rnal — The e mLue inuNg ize
The ical Bookad CDs
SAMBA: ntegtar tugN Xad Wind ws
Sle ll Tutoria ks, SH Refee
Vi & Emas Refees TM, utoria

© Cop yirgt 1 pe cial Sel ystems Consutlants, nc.
P.O. B 5554 9 Sea ttle, WA 9 8 15 -64 9
All Rig ts Reserved

DEFINITIONS

This sbc ile se s i n .2 d0 ba sh.

See at l ype fa. sbel la fi y the meaning
- Séir Ba dl issel for ompute input
- *Serifa Italic* issel o indiate se inpu tal for synta pc lae lo lass, ach *variable* or *cmd*.
- Séir Fa ma issel fo laa o yte

blank – separator between words. Blanks consist of one o mo pacal tra ba la ra te . Inadi ition, words a e mina tb y a y of le following ba ta:
; & ( ) | <> spe nta wil ne

command – a series of *words*.

list – one or more *pipelines*. Cate sepa ; , &, && , || ap it onal ly te te mina ; , &.

n – an integer.

name – a variable, alias, function or command name.

keyword – a reserved word in the ba sh lage Key ca ial only a fae ; ne wil ne, a fe ao the kr y wa ice tra io the cr onta.

pat – a ba sh pa See Patterns.

pipeline – a command or multiple commands connected by pa ipe (|)

string – a collection of characters treated as a unit.

substitution – the process of replacing parts of the co mmad il ne wti h, ffg ete la ing. vai ba le wti h tis va lue ba sh pe fro ms may sbs tit uit on s. This sbel il sts le m in the ro t r le ye a pe fro me d

word – a generic argument; a word. Quoting may be ne es a ily fict on a ins pe ial ba se.

RESTRICTED bash

If ba sh is inok ba sh or wti h the **-r** opi ton , it *restricted*. The fo llowing a it on sa ot llowed in a est ict ed sle ll:

ag idg nie to wy ti d
set ing ose ing SSHELL or PATH
sp ing nta me fo mmad s lact on a ni /
sp ing pa th ame tact on a ins / fo le . co mmad
imp it ing fun ctions om to v ni on me t
pa s ing SSHELLOPTS sta up
el nie it ing ou tput wtiah y of >, >|, <>, >&, &>, or >>
sp ing ex nto un da ffi ot mmad
add ing de le ting lit-co mmad s wti ena e
sp ing **command -p** to ypa s sta ict ed PATH
sp ing set -r or set o e e tt ict ed

The se est ict ion sa e ine ff e *after* e cu ting all sta tup lif e , allowing te tbo le sta tup lif e full co nt ol r se t ing the est ict ed v ni on me t. n(pt ice , e t ict ed sle ll sa ot ed much , a ty da ffi ot l se t up co re lt)y.

Error Reporting

If you if nd a r o in this e fe me le if st po t, ti we wli send you rf e co py of a y of co de. le se wtir e, ed le t on ic ma il to bg s@ssc om .

COMMAND LINE ARGUMENTS

ba sh ae pt l ne lp ion s o se , ad the ad ti on a l one le ta ed GNU-style log it on s lo wn le low .

$ ba sh [*options*] *args*

–	ed s pit op ro es ing
--	ed s pit op ro es ing
-c *cmd*	e cu te *cmd* e fa tr ea ds co mmad rf om lif ea me ni if st en ty of *args* ad fo md a pa ste a c h)
-D	pi nt a ll ob o lep st ing tha t pe y a $ o st du . This imp ies **-n**, no co mmad sa e cu te l
-i	si et nte ive mode
-r	se e t ict ed mode
-s	ea o mmad s rf om stdi e(fa) l
--dump-po-strings	sa se **-D** bu op in GNU e t te fo ma
--dump-strings	sa se **-D**
--help	displaya ea lp me sa ti su es fy l
--login	al ta k log in sle ll
--noediting	do t use the ea dline il bra y. a e o mmad s wh n inte ive
--noprofile	do no t re a da y oft le initi a za ion lif e . See Invocation And Startup, elov
--norc	do no t re el . ~/ ba sh fi inte ive S. e Invocation And Startup, elov
--posix	fo llow the IEEE POSIX 1 2 . sta nd a d
--rcfile *file*	use *file* ins ea d . ~/ ba sh fi inte ive
--restricted	sa se **-r**
--verbose	sa se **set -v**
--version	pi nt e ion informa ion on stdou a d ti su es fy l

INVOCATION AND STARTUP

The e e ways ba sh run s: no rmal nte ive, normal no ni-nte ive , a sh, in POSIX mode , or ino vd a s ll .

1. Normal nte ive Log in sle ll s r un co mmad s ni e / tc / pofi . The if st of . ~/ ba sh p ofi „ ~/ ba sh log in , ad . ~/ po fi tha it sho d ise ute l . This s g si skipp el fi **--noprofile** is sel

Upon log ut ba sh runs . ~/ ba sh log ut if i ex t ists.

Inte ive no l-log in sle ll se ue . ~/ ba sh , if ti ex ists. le **--rcfile** *ifile* op ton ag le lif te si ise l

2. Normal no ni-nte ive : No ni-nte ive sle ll sob va ri a le, mmad , ad tir h me it s: bs tti uit on the *value* of ENV , ad tfi le e s ut na me a ex isting lif te, la ft lie ise ute l

INVOCATION AND STARTUP (continued)

3. nvok ba sh: nte ive log in sle ll sr ea d e cu te e / tc / po fi ad . ~/ po fi fi le y ex ist. The se lif ea ek ipp el fi **--noprofile** is sel . nte ive sle ll se pa d ENV a e cu te la ft lie fi e ists. Noni-nte ive sle ll so bo rt ea d y sta up lif e. Af ter le sta up lif e a e te l ba sh en P OSIX mode

P4. OSIX mode : Wh n sta el wti h **--posix**, inte ive sle ll se pa d ENV a e cu te lg ive n lif e. No

l 5. nvok a s lo l : If run rf om s lo l an d it nvok sh, ba sh es . ~/ ba sh . The **--norc** op ton skips this ste p, ad the **--rcfile** op ton ag le lif e, bu s lo l sta lly do e s no t pass the se op ton s o n to le sta ll invok s.

If SSHELLOPTS ex ists in the v ni on me sta up ba sh ena le s the ive op ton s.

PROMPTING

Wh n inte ive, ba sh displays t lp ir mary ad se nda y po mpt st ings, PS1 ad PS2 . ba sh ex pa d s le fo llowing ea pe sequen es in the l es to f le se t ings.

\a	a ASCII BEL ha ta te l)
\d	the da e in "We kDa MoY on thDa y" fo ma
\e	a ASCII ea pe ha ta te l 033)
\h	the stra me up to le if st o t (.)
\H	the full o stra me
\n	a ne wil ne
\r	a a rig te um
\s	nha me of le sle ll ha ea me (bs)
\t	the it me in 24 -ho r HH:MM:SS fo ma
\T	the it me in 12h -ho r HH:MM:SS fo ma
\u	the sér s us er na me
\v	the si on of ba sh e(,g. 2.02)
\V	the si on a d pa tch lee l of ba sh e(,g. 2.02 .0)
\w	the cu re n t wo k id g ni e y
\W	the se na me of le cu re n t wo k id g ni e ty
\!	the isto y nu mbe r of his ommad
\#	the co mmad nu mb e for his ommad
\$	a # if the ff e i t e UID is 0 , o the rwi se a $
\@	the it me in 12h -ho r am/ m fo ma
\\	a ba sk lsh
\nnn	the ha ta ed ing ch a l va lue *nnn*
\[	sta t a sequen ce f rom -pi nt ing ha ta s
\]	en sequen ce f rom -pi nt ing ha ta s

The isto y nu mbe r i s the nu mb e for le co mmad ni the isto y il st, wh ich may inc lude co mmad s ete rf om tha isto y if le. The co mmad nu mb e r i st le nu mb e r of his co mmad s tta ing rf om the if st co mmad run y the cu re n t i no ea it on t f le sle ll.

The fa t l va lue PS1 is '\s-\v\$ '

HISTORY SUBSTITUTION

History expansion is implicit in shells. History expansion allows the user to use words from previous input lines, with optional editing, to construct new commands. This is described in the shell's history manual.

History substitution is specified by a *event designator* (which selects the previous command line), a *word designator* (which selects words from the chosen line), and optional *modifiers* (which manipulate the selected words). Colons separating the designators may be omitted if the word designator begins with a ^, $, *, -, or %. Each modifier is preceded by a colon. The history variable specifies the start of history expansion, the substitution character, and also the comment character. A history substitution may be preceded by an unescaped & in order to prevent it from being expanded (see the description of the shell's history manual for details).

The event designator:

!	start a history substitution
!*n*	command line *n*
!-*n*	current line minus *n* (*n* previous)
!!	the previous command
!*str*	most recent command line starting with *str*
!?*str*[?]	most recent command line containing *str*
!#	the entire command line typed so far
^*old*^*new*^	quick substitution: repeat last command replacing *old* with *new*

The word designator:

0	the zeroth word
n	word *n*
^	the first argument, i.e. word 1
$	the last argument
%	the word matched by the most recent ?*str*? search
x-*y*	words *x* through *y*. -*y* is short for 0-*y*
*	all but word 0, i.e. 1-$
*n**	words *n* through $
n-	words *n* through $, but omitting $

The modifiers:

e	remove all but the suffix from a file name
g	make changes global, apply the following modifier to each word
h	remove the last part of a file name, leaving the "head"
p	print the new command but do not execute it
q	quote the substituted words, escaping further substitutions
r	remove the last suffix from a file name
s/*old*/*new*/	substitute *new* for *old* in the text. Any delimiter may be used. An & in the replacement text is replaced by *old*. If *old* is empty, select the last *old* or the most recent ?*str*? search. A history reference may be given as the empty string.
t	remove all but the last part of a file name, leaving the "tail"
x	like q, but break into words at spaces, tabs, and newlines
&	repeat the last substitution

QUOTING

c	quote the character *c*
`` `...` ``	old type command substitution
`"..."`	selectively inhibit interpretation of quote characters (see below)
`'...'`	inhibit interpretation of quote characters (see below)
`$'...'`	selectively inhibit interpretation of quote characters (see below)
`$"..."`	selectively inhibit interpretation of quote characters, single quotes are not special (see below)

Within double quotes, the following sequences are processed:

\a	alert (bell)
\b	backspace
\c	suppress trailing newline
\e	escape
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab
ddd	octal value *ddd*
\x*hhh*	hex value *hhh*
\\	backslash

ALIASING

alias *name*=*value*...

Aliases are not processed when a command is quoted, not when the alias is created. Aliases may contain any non-special character, but usually begin with an alphanumeric, and may end with a space, in which case the next word is also checked for alias expansion. Aliases may be created for shell keywords, but not for POSIX mode.

BRACE EXPANSION

Brace expansion is implicit in shells. A word must contain at least one unquoted left brace and one unquoted comma-separated items inside the braces, or a sequence expression. Examples:

```
$ mkdir /usr/{ghost,share,include,bin}
```

TILDE SUBSTITUTION

~	substitute $HOME
~*user*	substitute *user*'s home directory
~+	substitute PWD
~-	substitute OLDPWD
~*n*	substitute ${DIRSTACK[*n*]}. A leading + or - is allowed. If *n* is negative, it counts from the end of the stack.

Tilde substitution happens after expansion. It is done for words that begin with ~ and for variable assignments.

In variable assignments, it is also done after any : in the value. Like substitution in the middle of a word, tilde substitution happens after expansion. This means for {$*name*/*op*/*word*}, *word* will be subject to tilde substitution, but only if the original value of the variable being substituted began with a ~.

VARIABLE SUBSTITUTION

$*name*	value of variable *name*
${*name*}	same as $*name*
${*name*-*word*}	use variable *name* if set, else use *word*
${*name*=*word*}	assign *word* to *name* if unset
${*name*?*word*}	use *name* if set, otherwise print *word* and exit (if non-interactive) shell
${*name*+*word*}	use *word* if *name* is set, otherwise use nothing
${*name*[*n*]}	element *n* in array *name*
${#*name*}	length of variable *name*
${#*name*[*]}	number of elements in array *name*
${#*name*[@]}	number of elements in array *name*
${*name*#*pat*}	remove shortest matching string from beginning of *name* that matches *pat*
${*name*##*pat*}	remove longest matching string from beginning of *name* that matches *pat*
${*name*%*pat*}	remove shortest matching string from end of *name* that matches *pat*
${*name*%%*pat*}	remove longest matching string from end of *name* that matches *pat*
${*name*:*start*}	
${*name*:*start*:*length*}	substring of *name* starting at character *start* (zero-based, or from the end if negative) and up to *length* characters long (or to the end if no *length*). E.g. if positive *start* counts from the beginning, if -1 from the end, if * or @ the positional parameters, and if array indexed by * or @, *start* and *length* indicate the range of array indices and number of elements. *start* and *length* are arithmetic expressions
${*name*/*pattern*/*string*}	variable *name* with first match of *pattern* replaced with *string*
${*name*/*pattern*}	variable *name* with first match of *pattern* deleted
${*name*//*pattern*/*string*}	variable *name* with every match of *pattern* replaced with *string*
${*name*##/*pattern*/*string*}	variable *name* with match of *pattern* replaced with *string*; match occurs at beginning
${*name*%%/*pattern*/*string*}	variable *name* with match of *pattern* replaced with *string*; match occurs at end

Note for -, =, ?, and +, using *name*: instead of *name* tests whether the *name* is assigned (non-NULL); using *name* tests only whether the *name* is set.

For #, ##, %, %%, /, //, /#/, and /%/, when *name* is * or @ and any index of * or @, the substitution operation is applied to each element.

ARITHMETIC EVALUATION

Arithmetic evaluation is done with the let builtin command, the (( )) command and the $(( )) expansion form.

Arithmetic is done using long integers. **typeset -i** integer variables are limited to [*base*#]*n* where *base* is a decimal number between two and 64 and *n* is in that base. The digits are 0-9, a-z, A-Z, _ and @ (leading 0 or 0x or 0X also accepted for hexadecimal).

The following operators are supported, with the same precedence and associativity as the C language:

+ -	unary plus and minus
! ~	logical and bitwise negation
**	exponentiation (not in C)
* / %	multiply, divide, modulus
+ -	addition, subtraction
<< >>	left and right bitshift
< <= > >=	comparisons
== !=	equality, inequality
&	bitwise AND
^	bitwise XOR
|	bitwise OR
&&	logical AND, short-circuiting
||	logical OR, short-circuiting
? :	trinary conditional
= += -= *= /= %= &= |= ^= <<= >>=	assignment

Integer constants with a leading 0 are octal, and with a leading 0x or 0X are hexadecimal. Otherwise, constants have the form [*base*#]*n*. Variables may also be used; their values are converted to integers.

COMMAND SUBSTITUTION

($ *command*)	new form
`` `command` ``	old form

Run *command*, substitute the standard output, with trailing newlines removed. Field Splitting) The result is split into words (see Field Splitting). The new form simplifies command parsing rule.

$((*expression*)) arithmetic substitution

The *expression* is subject to the same expansions as if it were an argument to the let command.

PROCESS SUBSTITUTION

cmd <(*list1*) (*list2*)

Runs *list1* and *list2* asynchronously with their standard input or output (respectively) connected to named pipes (FIFO files in the file system) or open files in /dev/fd. These files are then passed as arguments to *cmd*, which is expected to open them, read or write them, and finally close them. This is only supported on systems with named pipes (FIFOs) or /dev/fd files.

FIELD SPLITTING

Quoted text becomes one word. Otherwise, the result of expansion is scanned for any characters in $IFS separating words. Multiple white space characters in $IFS delimit empty words, while multiple non-white space characters in $IFS delimit words. When $IFS is not set, the default value is used. Leading and trailing $IFS white space characters are removed, and initial sequences of $IFS white space characters surrounded by any amount of $IFS white space characters delimit fields. If $IFS is NULL, the shell does not do field splitting.

PA TTERNS

?	msta ing te la ta inlif ea me
*	m0a mte inlif ea me
[*chars*]	mta ah y of *chars*
	pa(ri sepa atd y a – mta aag
![*chars*]	mtah y e e pt *chars*
^[*chars*]	mtah y e e pt *chars*

tfl le e tglo opit on o shopt iss tet le fdl owing e tedd mta ing faili ite may be sed

?(*pat-list*)	opit onalhy atah y tof tpe t ers
*(*pat-list*)	m0a mre f a y tof tpe t ers
+(*pat-list*)	om1a mre f a y tof tpe t ers
@(*pat-list*)	mtah ltyo1 f a y tof tpe t ers
!(*pat-list*)	mta ah ythinb at y tof tpe t ers

pat-list is a il st of oe r mpatsepd \|.

The POSIX [[*c*=]] ad .[[]]*c*. nota ionsfa me-weigh la ra tead ll ta ing elemen tsad . The nota ion:[[*class*]]: de if nes lacte lase:

alnum	alph au mer	lowe	lower cae
alpha	alph ae cit	p int	pir nta e
blank	spe t r a	pun c	pun ct ua ti on
cntrl	o nt o	spe	whit e pa e
digit	de cimla	up e	up er cae
xdigit	ron -spas	xd igit	ke cimla

The shopt opit ons affe t pat ern mta ing

dotglob incl ud edif e wb se name eg in wit .
nocaseglob ig oe e wa mta ing
nullglob e mve pa t erns tha t don t' mtah

Whenpad ing lif ea me, . ad . . ea ig ed lif ea me mta ing tpe t ers ri $ GLOBI GNORE are also ig ored la ding . mus t le sup il ed in the pat ern mta lif ea me tha t eg in wit . . However, set ing GLOBI GNORE enbla le the dotglob opit on. nclude * . ni GLOBI GNORE ddtg fatl ioar

VA RIA BLE NA MES

Vari ab lea mea e mapo If d, igit sa nd und sre . The y may not a tw ith igit The e si ro il mti on the leg th fva ir ab e na me, ad the cae lof te is igifi a nt

VA RIA BLE ASS IGN MENT

Assig men ts o im ig ia le und erg ti hmeti ea lua it onea as ig men ts e ne tf e fdl owing fo ms.

name=*word*	set *name* to *word*
name[*index*]= *word*	set element *index* of ar ay *name* to *word*
name=(*word* .)	siet nd e ad r ay *name* to *words*
name=([*num*]= *word* .)	set give n ind ies f ar ay *name* to *words*

PR E-DEFINED VA RIA BLES

$*n*	use sition a lpa a me $n, n \le 9$
${ *n*}	use sition a lpa a me *n*
$*	all po sit ional pa a me
$@	all po sit ional pa a me
"$*"	equ iva len t2 $1 $ . . . "
"$@"	equ iva len t2 "$1" "$ . . .
$#	num ber of itiona l pa a metes
$–	opit ons so e ll bgo set
$?	va lue re urnd y last om mad
$$	pess num ber of ll
$!	pessr u mbe r of a t ba k ground cmd
$_	name f prog ram inv ri onmen ta sta tup . Value f a t posit ional ag meit n last mmad . Name f liw lmailif eri MA LP ATH
$BA SH_ARGC ume	eab le use o f si ngle-wod o mmad $ o matsh ted jdo s for fogd ing . Wta te a t, the word mus t ea t ly t a h the o mmad sed to a t t the jdo W hi a va le subs t ing , the typed word ea to ae subs t ing of le o mmad, ile %?*string*
$BA SH	fullifna me sed o inv k la sh
$BA SH_ENV	im o mal non i -nte ive sellls o nly va lue is va ir ab le, mmadad ati hmeics ubsti tuted for pa th of sta tup lif (See Invocation And Startup)
$BA SH_V ER SION	ve sion of la sh
$BA SH_VERSINFO [0]	the major vesion num ber (rele ase)
$BA SH_VERSINFO [1]	the mino vesion num ber (ve sion)
$BA SH_VERSINFO [2]	pa ch leve
$BA SH_VERSINFO [3]	the build ve sion
$BA SH_VERSINFO [4]	the ele ase sta us
$BA SH_VERSINFO [5]	same as $MA C HTYPE
$C DPATH	sea ch pa h for cd o mmad
$D IRST ACK[*]	ar a y va ir ab le o nta ining the pushd and popd dire o y sta k
$ENV	in inte ive POSIX mde sellls, whe invoked sh, the va lue is va ir ab le, o mmadand ati hmeitc subs tituted fpa th stf ta up fif
$EU ID	the ffe ti ve use r id (read-on ly)
$F CEDIT	de fau lt edit o le of o mmad fc, de fau lt va le
$F IGNORE	o lon -sepa ated il st of suf ifes giving the set of lif ea mes to ig ore whe o ing lif ea me o mple tion u sing ea d il ne
$G LOBI GNORE	o lon -sepa ated il st of pa t erns giving the set of lif ea mes to ig ore whe o ing ta mta hing
$G ROU PS[*]	ead only a ay va ir ab le wti h the il st of groups the ... long to
$h istchars	ha t t ha to nt o sh -style histo y (de fau lt #! S) e History Substitution

PR E-DEFINED VA RIA BLES (continued)

$HIS TCMD	histo y num ber of e cur ent o mmad
$HIS TCONTROL	wti ah lue ig e pace , do ot ente ilnes t ha t eg in wti space into e sto y lif e. Wti ah lue ig e dups, do not e il ne tha ma ches t la t p e ious il ne. Use ig e both to o mbine tpo it ons
$HIS TFILE	whe re o mmad istory is ed
$HIS TFILESIZE	ma imum num ber of ines to o p in $HIS TFILE
$HIS TIGNORE	o lon -sepa ated il st of pat erns; if the cur ent il ne mta a y tof te m, ht e il ne is o t e ed in the isto ylif e & epeats the last histo y il ne Pat erns mus t mta h the who le il ne
$HIS TSIZE	num be r of pre ious o mmads to ep a ila le whle la sh is un ing
$HOME	ho me ire o y for cd o mmad va lue sed fo lie sion
$HOSTFILE	lif e in fo ma e/ c ho sts e for ho stna me mple ion
$HOSTNA ME	na me of e cur ent ho st
$HOSTTYPE	st ing escir bing the cur ent ho st
$IFS	if eld sepa to (space, ta , new il ne)
$IG NOREEOF	fo inte ive sellls, nu mbe r of o nsec itve EOFs tha t mus t be ente ed fore la sh a tua lly e its
$INPUTRC	na me of ea il ne sta tup lif e, ove r ides .~/ inputrc
$LANG	na me cf ur ent loa e
$LC_ALL	cur ent loa le; ides $LANG and othe LC_ va ir ab les
$LC_COLLATE	cur ent loa le for llat ion, incl ud es o tr ing esu ts of lif ea me e pa sion
$LC_CTYPE	cur ent loa le for a a te las fun ctions (see Patterns)
$LC_ ME SSAGES	cur ent loa le for tans ating "$. " st ings
$LINENO	il ne num ber of e ing e cuted ni script or fun ction
$MA C HTYPE	a st ing in GNU *cpu-company-system* format esc ibing the ma hine run ing la sh
$MA IL	name of a mailif e, ay
$MA IL CHECK	che k fo mai e y *n* seo nds (60 de fau lt)
$MA ILP ATH	lif ea mes o e k for wm ai; use : sepato *filename* may be fdl owed by ?*message*; $_ ni *message* is mai lif ea me. Ove r ides $MA IL
$OLDPW D	pe ious wo ing ire o y
$OPT ARG	va le l f ast g men t pesse by getopts
$OPT ERR	if set to 0, display error mesages from getopts (de fau lt 1)
$OPT IND	inde l f ast g men t pesse by getopts
$OSTYPE	st ing d escir bing the ope ta ing system run ing la sh
$PATH	o mmad sea ch pa h
$PIP ESTATUS [*]	a a y va ir ab le o nta ining e ti sta us va lues f om pesse in the most e ntly e e cuted foground ipeil ne
$PP ID	pess id s f e'll s pa ent
$PRO MPT _COMM AND	o mmad to run efo e pir mary pro mpt
$PS1	pir mary pro mpt st ing (\s–\v\$)
$PS2	seo nda y pro mpt ing (>)
$PS3	sele t o mmad p rompt st ing (#?)
$PS4	t a ing o mpt ing (+)
$PW D	cur en t wo ing ire o y
$RAN DOM	sta ime 'ti s ed, 0 – 32767
$REPLY	set by the sele t and ead o mmads
$SECONDS	num ber of econds ince ll inv o a it on
$SHELL	na me of his ell
$SHELLOPTS	o lon -sepa ated il st of e ena led sell opit ons of set –o
$SHLVL	inc e men ted y one for ea h sub la sh
$TIMEFORMAT	fo mat ing for time ke yw rd Spe ia o nst ucts introdu e by %.

%[*p*][l]R	elap sed
%[*p*][l]U	use r CPU
%[*p*][l]S	system CPU
%P	CPU pe ent g
%%	il te al %

Opit onal *p* give s the e ision, the nu mber of dig for de cimal po int; ti mus t be be twe en 0 and 3. Opit onal l pod uces a long er fo mat, ni the form *MM*m*SS.FF*s

$TMOUT	num ber of seconds to wait dui rng p o mpt be fore e minait ng
$UID	the real use r id (read-on ly)

FUN CTI ONS

Fun ctions un in the same pess the ail ng script, and share pen lif ea nd cur ent dire to y. They ass ther i pa a me itae script vai $1 , $2 and so on. $0 does not chage e turn may be sed inside a fun ction . script Fun ctions share ps wti h the pa ent ipt, e e pt ea DEBUG . Fun ctions may be e ive , ad may be loa l vair ab les , declad ing e la , loa , or type se . Fun ctions may ep ted inde env ri onmen t wti **export –f**.

INPU T/ OUTPUT

Re dri e it on sa ohe ltf ro ig t atfpe ipe sa se up . Defa ltlif de so rptoa stdi ad stdo . Ei de so rptdae e nodu lose -on -e c

& *word*	send stdu ad stde r o *word*
>& *word*	send stdu ad stde r o *word*
[*n*]<*file*	se *file* fo iput
[*n*]>*file*	se *file* fut
[*n*]>| *file*	ilek >bu ot ies moc ldoe
[*n*]>> *file*	ilek > budped o *file* fi et ist
[*n*]<> *file*	ope *file* fo wtir (e de ffat l 0)
[*n*]<& *m*	dp il a t e iputf tilde sorpto *m*
[*n*]>& *m*	dp ile *m*
[*n*]<&–	close iputf tilde sorpto
[*n*]>&–	close utputf tilde sorpto
[*n*]<< *word*	

impu ct ome rf om the shell sorptpt, e t ail ne wti *word* a EOF on iput . fl a y of *word* isqo tel , no adl iti ona l proes ing isdoe iput by the sh.ll Otle rwise

• do vair a be, mm ad , air hmeitc bo stti uti ons
• ig ed wil ns
• use \ to e \, $, ad if st la de *word*

[*n*]<<–*word* a loe , but wti h la ding ta ig ored

Of & ad >& , the if st pde
>*word* 2>&1 .

EXECU TI ONOR DER

All sbs tit uti on sa nd /I O relrie it on sa pe med be de mm ad isa tu ally e cu ted

ba sh mainta insa inte rnal ba sh ta le fo ing ex te nal o mm ad s. Iniit ally this ta le ise mpty As o mm ad sa od y sea c hing tde rie o ie ilsted ni PATH , the yad to the sh ta le

The o mm ad saoch roie r s hell fun ctionf bulit io mm ad s o nd , ad e nal o mm ad s(ifst ni the inte rnal a sh ta le, ad the i PATH) thi d

SIGNALS AND TRAPS

Sig al ad il ng isdoe wti h the t p bulit-o mm ad The *word* g me t o de so rbing o de te tpm e iptf le sig al ssc aned twie by ba sh; on e whe the t p o mm ad ise ued in whe the sig ial sc a t. The qu ote fo le t p o mm ad T. p sa e tel ni sde ig al numbe r. Yo o ct lag the sta ofsa ig tal la t wa ig oed whe the shell stpd

Tr p son DEBUG lap tfoe mm ad sa e tel

Ba k gound mm ad s(tho se foll ow ey &)w lli ig te SIGINT ad SIGQUIT sig alsfi le mo nti o opiton is urned ff . Othe rwise, he y inh eti e les to f the ret ba sh.

ARRAYS

Ar ays ni ba sh ave ilmti so n theu mbe r of eleme ts. Ar ay ind ies a t at 0. Ar ay subsc ir pts a ab e ar hmetie pes ion s. Ar ay eleme ts need no t be o nti gu ou s. ba sh doe sno t ha ssc ita ive a ys.

CONTROL COMMANDS

! *pipeline*
 e cu e *pipeline*. If e ti stu wa non ze ti e fl a ti sta wa ze ti 1
case *word* in [[(]*pat1*[|*pat2*]) *list*;;.] esac
 e cu e *list* asoc ited wti *pat* tha t mta les *word*. Field p ilt ing iso t doe fo *word*. *pat* isa ba sh pa tte Patterns) | isse l o indi a e OR o ndti on . Use la ing (fi e is inside $( )
for *name* [in *words*] do *list*; one
 seque eit ally as ig ea h *word* o *name* ad cu e *list*. fl ni *words* is missing se the sti on al pa rame
[function] *func*() { *list*;}
 deifne fun c tion *func* odysi *list*s(e Functions)
if *list1*; then *list2* [elif *list3* ; then [*list4*] e se *list5*] ;fi
 fi e cu ting *list1* te urnssu ess ful e ti sta, e cu e *list2* else .
select *name* [in *words*] do *list*; one
 pirnt meu *words*, rompt wti PS3 ad ed a il ne rf om stdi , sa ing it i REPLY . If the il ne si tha mbe o wte *name* o ti t wise t *name* o NULL . E e *list*. fl ni *words* is missing se the sti on al pa ame. ba sh a toma ia lly rep ir nts he meu tt the loop
time [–p] *pipeline*
 e cu e *pipeline*; pirnt lap sed, ystem ad se ti me o stde r .
 –p pirnt ime ni PO SIX fo ma
 The TS IMEFORMAT vair a le nt ol s he fo ma the tpfi t –p iso t used ba sh use the lue $' na t\l %3lR\nuse %3lU\nsys\ %3lS' fi the e si na lue fo TS IMEFORMAT
until *list1* do *list2*; one
 ilek whle bu t ng a t e he te minait on test
while *list1* do *list2*; one
 e cu e *list1*. If la t o mm ad ni *list1* lad a su ss ful e ti sta, *list2* foll ow ey *list1*. Repa t unlit a t omm ad ni *list1* te urns a un su ss ful e ti sta
((..))
 a ti hme ie lu a ition , ek le .". "
[[*expression*]]
 eva lu a e *expression*, e urn su ss ful e ti sta fi t ue , un su ss ful fi alse s(e Conditional Expressions li s)
(*list*)
 e cu e *list* ina sub -shell
{*list*;}
 e cu e *list* in the cu rre t shell

CONDITIONAL EXPRESSIONS

Used wti h the [[..]] o mpound o mm ad , whie not pa t ms ion r w sopl ilt ing

string	t ue fi *string* iso t NULL
–a *file*	t ue fi *file* e ist (–e isp e)
–b *file*	t ue fi *file* isa block e ie
–c *file*	t ue fi *file* isa la te e ie
–d *file*	t ue fi *file* isa di e y
–e *file*	t ue fi *file* e ist
–f *file*	t ue fi *file* isa e ifl a
–g *file*	t ue fi *file* la s id ti se
–G *file*	t ue fi *file* gop ise ffe it e gd
–h *file*	t ue fi *file* isa ym boil cil nk
–k *file*	t ue fi *file* la s it k y bti se
–L *file*	t ue fi *file* isa ym boil cil nk
–n *string*	t ue fi *string* la son -ze leg h
–N *file*	t ue fi *file* e ist nd wa modfi ed sine lart ed
–o *option*	t ue fi *option* iso
–O *file*	t ue fi *file* ow ne r se ffe it e ud
–p *file*	t ue fi *file* isa fif (o na mped ipe)
–r *file*	t ue fi *file* is a e ld e
–s *file*	t ue fi *file* la son -ze ie
–S *file*	t ue fi *file* iso t
–t *filedes*	t ue fi *filedes* isa e mina
–u *file*	t ue fi *file* la s te id ti se
–w *file*	t ue fi *file* is wtir a le
–x *file*	t ue fi *file* ise uta le
–z *string*	t ue fi *string* la s e leg h
file1 –nt *file2*	t ue fi *file1* ise we ha *file2* or *file2* doe sno t e ist
file1 –ot *file2*	t ue fi *file1* is lder ha *file2* or *file2* doe sno t e ist
file1 –ef *file2*	t ue fi *file1* ad *file2* sa meif
string == *pattern*	t ue fi *string* mta les *pattern*
string != *pattern*	t ue fi *string* doe sno t mta *pattern*
string1 < *string2*	t ue fi *string1* ise fe *string2*
string1 > *string2*	t ue fi *string1* isa tf e *string2*
exp1 –eq *exp2*	t ue fi *exp1* eq s *exp2*
exp1 –ne *exp2*	t ue fi *exp1* doe sno eq *exp2*
exp1 –lt *exp2*	t ue fi *exp1* is les tha *exp2*
exp1 –gt *exp2*	t ue fi *exp1* isg ha *exp2*
exp1 –le *exp2*	t ue fi *exp1* is les tha eq o l *exp2*
exp1 –ge *exp2*	t ue fi *exp1* isg rea te ha eq o l *exp2*
(*expression*)	t ue fi *expression* is e , gop ing
! *expression*	t ue fi *expression* ise
exp1 && *exp2*	t ue fi *exp1* AND *exp2* a
exp1 || *exp2*	t ue fi *exp1* OR *exp2* is ue

fl *file* is e / fd *n*, the i , e e iso e/ d di e o y li f e sorpto *n* is c le ek d. Othe rwise, e ha e/ fd *n* if e is le ek l d. nu x, Fe BSD, BSDs maylao the s) e info fot he indi ia te d e if de so rpto , is od t f e t a e/ d e ie lif e

Both && ad || es o t oi cu ti Ope ad s o o mpair sope und a ti hme ie lu a it on. Fo == ad != op ae y pa rt o *pattern* ti a a st ing

BUILTIN COMMANDS

The se mm ad sa e el rie tly by the shell. Almost ll e pt –– o ek f opiti on s.

. *file*
source *file*
 ead ae eo mm ad s rf om *file*. fl g me ts ae ti ona l pa a ms. Sea h PATH ; if o ing fondl , ok in the cu e t di e y
: null o mm ad ; e 0 ti sta
[se est
alias [–p] [*name*[=*value*] .]
 c ea te na il a. Wti n o a rg me ts, pirnt all il ae . Wti *name*, irnt il sa le fo *name*
 –p pirnt il a le ae il a
bg [*jobid*]
 pt *jobid* in the k gound
bind [–m *map*] [–lpPsSvV]
bind [–m *map*] [–q *func*] [–r *keyseq*] [–u *func*]
bind [–m *map*] –f *file*
bind [–m *map*] *keyseq:func*
 display ado / r modfi e dil ne fun c tion bind ings. ynta is ame o i .~/ nput
 –f *file* e wbind ingsf om *file*
 –l il st the me o f ll e dil ne fun c tions
 –m *map* se the ym p *map*
 –p il st e dil ne fun c tion sa d ind ing for e-rea ding
 –P il st e dil ne fun c tion sa d ind ing
 –q *func* sh ww lich ys inve *func*
 –r *keyseq* e mo e ind ing s for *keyseq*
 –s il st e dil ne e y seque e a nd ma os for e-rea ding
 –S il st e dil ne e y seque e a nd ma os
 –u *func* e mo e y bind ing s for *func*
 –v il st e dil ne vair a ha me a nd va lus for e-rea ding
 –V il st e dil ne vair a ha me a nd va lus
break [*n*]
 e ti om e clos ing o , whie , unlit or sele t loop. fl *n* is upp iled , e ti om *n*'th clos ing loop
builtin *shell-builtin* [*args* .]
 e cu e *shell-builtin* wti g ie *args* ad e urn sta . Useful fo d y ofsa he ll fun c tion tha e if ne sa bulit -n, e,g. cd
cd [–LP] [*dir*]
 e re t die o y o *dir* ($HOME de fa t.) Do rie o y pa sha c uh sing va le S DPATH
 –L se log ia l pa th for cd .., $PWD (de fa t)
 –P se ysia l pa th for cd .., $PWD
 fl bo ah ge ive n, he la t the o mm ad il ne wins
cd [–LP] –
 e re t die to yo S OLDPWD
command [–pvV] *name* [*arg* .]
 wti ho –v or –V, e cu e *name* wti h rg me ts *arg*
 –p se de fa t sa ch pa th PATH
 –v pirnt ipit on *name*
 –V pirnt ipit on *name*
continue [*n*]
 do e tit e io f e clos ing o , whie , unlit or sele t loop. *n* is upp iled , e *n*'th clos ing loop

BUILT-IN COMMANDS (continued)

declare [±aAfFrix] [−p] *name*[=*value*]
typeset [±aAfFrix] [−p] *name*[=*value*]
set attributes and values of variables. Inside functions, ... of the variable. Using + instead of − turns attribute off. With no names, print every variable's name and its attributes
- **−a** *name* is an array
- **−f** each *name* is a function
- **−F** don't show function definitions (bodies)
- **−i** *name* is an integer ... arithmetic evaluation ... assignment
- **−r** make *names* read-only
- **−x** make *names* for export

dirs [**−clpv**] [+*n*] [−*n*]
display the directory stack
- +*n* show *n*'th entry from left, *n* ≥ 0
- −*n* show *n*'th entry from right, *n* ≥ 0
- **−c** clear the directory stack
- **−l** print long form ... listing
- **−p** print the stack one entry per line
- **−v** print the stack one entry per line, with index numbers

disown [**−ar**] [**−h**] [*job* ...]
with no options, remove named *jobs* from the table of active jobs
- **−a** remove all ... (with **−h**) all jobs
- **−h** mark ... *jobs* ... a SIGHUP when the shell terminates
- **−r** use with **−h** to mark just running jobs

echo [**−eEn**] [*words* ...]
echo *words*; **−−** is not special
- **−e** expand \-escapes (see *echo*(1))
- **−E** never expand \-escapes
- **−n** don't output trailing newline
printf is more portable

enable [**−adnps**] [**−f** *file*] [*name* ...]
enable and disable built-ins, or load and unload ... built-ins from shared library file. Disabling a built-in allows ... file with the same name as built-in
- **−a** print all built-ins, with their status
- **−d** delete built-in loaded with **−f**
- **−f** *file* load new built-in *name* from *file*
- **−n** disable *name*, or print disabled built-ins with no *names*
- **−p** print enabled built-ins
- **−s** print only POSIX special built-ins

eval [*words*]
evaluate *words* and execute result

exec [**−a** *name*] [**−cl**] [*words*]
execute *words* in place of the shell, run ... redirections only ... the shell's open files
- **−a** use *name* for argv[0]
- **−c** clear environment first
- **−l** place a − in argv[0] (like *login*(1))

... interactive shell ... unless the shopt option ... is set

exit [*n*]
exit with return value *n*. Use $? if no *n*

export [**−fnp**] [*name*[=*value*] ...]
with no arguments, print names and values of exported variables. Otherwise export ... *names* to the environment of commands
- **−f** *names* are functions
- **−n** stop exporting each *name*
- **−p** print export ... before the ... variable

fc [**−e** *editor*] [**−nlr**] [*first* [*last*]]
process ... of commands from *first* to *last* from the last $HISTSIZE commands
- **−e** run *editor* ... if supplied; ... not, use ... FCEDIT, $EDITOR, or vi on commands ... then ...
- **−l** list ... to stdout ... instead of editing
- **−n** don't print line numbers
- **−r** reverse ... commands

fc **−s** [*old*=*new*] [*command*]
substitute *new* for *old* in *command* (or last command) if no *command* ... the ... result

fg [*jobid*]
put *jobid* in the foreground

getopts *optstring name* [*arg* ...]
parse ... arguments ... options (see *bash*(1))

hash [**−r**] [**−p** *file*] [*name*]
with no arguments, print ... the shell ... contents, giving ... found ... file name
- **−p** *file* enter *file* for *name* in the shell table
- **−r** clear the internal hash table

Assign ... PATH also ... the shell table

help [*pattern*]
print help. With *pattern*, print help about all the commands that match *pattern*

history [*n*]
history −anrw [*file*]
history [**−c**]
history −p *arg* ...
history −s *arg* ...
with no options, print the command history. An argument *n* prints only *n* lines. If supplied, use *file* instead of $HISTFILE
- **−a** append ... new history lines to history file
- **−c** clear the history list
- **−n** read ... new history lines in the file into the internal history list
- **−p** perform history substitution ... print the result
- **−r** replace internal history with ... history file
- **−s** place the *args* into the history list ... for later
- **−w** write the internal history to the file

jobs [**−lnprs**] [*jobid* ...]
jobs −x *command* [*args* ...]
list information about jobs
- **−l** also list process id
- **−n** only list stopped ... notified jobs
- **−p** only list process groups
- **−r** only list running jobs
- **−s** only list stopped jobs
- **−x** replace any *jobid* in the command line with the corresponding process ID, and execute the command

kill [**−***sig*] *jobid* ...
kill [**−s** *signame*] [**−n** *signum*] *jobid* ...
send SIGTERM ... given ... signal ... to ... *jobids*. Signals ... may ... with or without the SIG prefix ... GST. ... jobs get a SIGCONT if first ... *sig* is either SIGTERM or SIGHUP

kill −l [*sigs* ...]
list signal names and/or numbers. If *sig* is a number ... exit status, ... the signal that killed the process

let *arg* ...
evaluate each *arg* as an arithmetic expression; if 0 ... other... (see Arithmetic Evaluation)

local [*name*[=*value*] ...]
create a variable with the given value local to a function. With no operands, print all local variables. Must be used inside a function

logout
exit a login shell

popd [**−n**] [+*n*] [−*n*]
remove top entry from the directory stack. With no arguments, remove the top and cd there
- +*n* remove *n*'th entry from left, *n* ≥ 0
- −*n* remove *n*'th entry from right, *n* ≥ 0
- **−n** don't change directory

printf *format* [*arg* ...]
print output like ANSI C printf, with extensions
- %b expand escape sequences in strings
- %q print quoted string that can be re-read

Format ... reused as needed

pushd [**−n**] [*dir*]
pushd [**−n**] [+*n*] [−*n*]
add directory to the directory stack. With no arguments, exchange the top two entries
- +*n* ... entry from left is at the top, *n* ≥ 0
- −*n* ... entry from right is at the top, *n* ≥ 0
- **−n** don't change directory
- *dir* push *dir* on the stack and cd there

pwd [**−LP**]
print working directory name
- **−L** print logical path (default)
- **−P** print physical path

If both given, ... last ... the command line wins

read [**−a** *name*] [**−er**] [**−p** *prompt*] [*names* ...]
read stdin and assign to *names*. $IFS splits input. REPLY is the ... if no *name* given. Exit 0 unless end of file ... seen
- **−a** read words into indexed array *name*
- **−e** use readline if reading from a terminal
- **−p** print *prompt* if reading from a terminal before reading
- **−r** \ at ... is ... no ... to ... line continuation

readonly [**−afp**] [*name*[=*value*] ...]
make *names* read-only; print list if no *names*
- **−a** each *name* must be array
- **−f** each *name* must be function
- **−p** print read-only ... before ... variable

return [*n*]
exit a function ... script with return value *n*. With no *n*, the return status of last command. If not in function or . script, print error message

set [−*options*] [**−o** *option*] [*words*]
set flags and options (see Options To set.) *words* set positional parameters

set [+*options*] [+**o** *option*] [*words*]
unset flags and options

shift [*n*]
rename positional parameters, $*n*+1 ... *n* defaults to 1

shopt [**−opqsu**] [*option* ...]
print or change values of shell options. With no arguments, print shell option information
- **−o** only allow set **−o** options
- **−p** print in reusable ... format
- **−q** quiet mode; exit status indicates option status
- **−s** set (enable) given option; with no options, print ... that are set
- **−u** unset (disable) given option; with no options, print ... that are unset

(see Options To shopt)

suspend [**−f**]
suspend the shell until SIGCONT is received
- **−f** ... suppress ... if login shell

test
evaluate conditional expressions (see Options To test and Conditional Expressions)

times
print cumulative process times

trap [**−lp**] [*word*] [*sigs*]
execute *word* if signal *sigs* received. *sigs* are numbers or signal names with or without "SIG". With no *word* or *sigs*, print traps. With *word*, ... *sigs* to ... defaults. If *word* is "−", reset *sigs* ... If *word* is null string, ignore *sigs*. If *sigs* is 0 or EXIT, run *word* at exit from shell. If *sigs* is DEBUG, run *word* after each command
- **−l** print all signal names and numbers
- **−p** print traps with quoting

type [**−apt**] *name* ...
describe how the shell interprets *name*
- **−a** print all possible interpretations of *name*
- **−p** print the name of file that ... be ... *name* is an internal program
- **−t** print a word describing *name*