K 2.0 Reference Card

SELECTION

`d @ i` **Index (At)**
Returns `i` with atoms replaced by values drawn from `d`, a list, directory or dictionary, according to `i`, indices or symbols as appropriate.

`d . i` **Index (Of)**
Returns items from `d`, a list, directory or dictionary. A vector `i` returns one item-at-depth; otherwise results are cross-sectional. A null value means "all".

`d[i]` **Select items.**

`d[j;k;…]` **Select items at depth.**

STRUCTURAL

`@ x`	**Atom**	Is `x` an atom?
`# x`	**Count**	Number of items.
`^ x`	**Shape**	Extent of rectangularity.
`x # y`	**Reshape**	Make rectangles of atoms or lists.
`, x`	**Enlist**	Form list.
`x _ y`	**Drop**	Drops `x` items from an end of `y`.
	Cut	Break `y` into pieces at indices `x`.
`* x`	**First**	**First item.**
`+ x`	**Flip**	Transpose top two levels of `x`.
`x ! y`	**Rotate**	Move `x` items from one end of `y` to the other.
`| x`	**Reverse**	Reverse order of items.
`x , y`	**Join**	Join atoms or lists together.

ARITHMETIC

`x + y`	**Plus**	`x` **plus** `y`.
`x - y`	**Minus**	`x` **minus** `y`.
`- x`	**Negate**	
`x * y`	**Times**	`x` **times** `y`.
`x % y`	**Divide**	`x` **divided by** `y`. **Note** `0 % 0` **is** `0`.
`% x`	**Reciprocal**	Reciprocal of `x`.
`x ^ y`	**Power**	`x` to the power `y`.
`x ! y`	**Mod**	`y` **residue of** `x`.
`_ x`	**Floor**	Largest integer not greater than `x`.
`x < y`	**Less**	Is `x` less than `y`?
`x > y`	**More**	Is `x` greater than `y`?
`x | y`	**Max**	Larger atom (boolean OR).
`x & y`	**Min**	Smaller atom (boolean AND).
`~ x`	**Not**	`0 = x` for numeric *x*.

COMPARISON

`x = y`	**Equal**	Are atoms of `x` and `y` tolerantly equal?
`x ~ y`	**Match**	Does `x` match `y`?
`x ? y`	**Find**	First location of `y` among items of `x`.

FUNCTION APPLICATION

`f @ x` **Apply Monadic**
Applies monadic function to value. e.g. `{x^2}@3`

`f.x` **Apply**
Applies function of any valence to appropriate value.
e.g. `f0._n` `f5.(a;b;c;d;e)`
Related forms:
`f[j;k;…]` `f[]`

`f ? y` **Inverse**

`?[f;y;x`$_{init}$`]` Finds roots of `y = f[x]` using secant method optionally starting at x_{init}.

`.[f;x;:]` **Trap** Traps errors when `f` is applied to `x`.
`@[f;x;:]` `(0;result)` or
`(1;error text)`

`:[c;expr;f]` **Conditional** `:[`c_1`;`e_1`;`c_2`;`e_2`;… `c_n`;`e_n`;f]`
`do[n;`e_1`;…;`e_n`]`
`if[c;`e_1`;`e_2`;…;`e_n`]`
`while[c;`e_1`;`e_2`;…;`e_n`]`

MODIFICATION

Amend Item

`@[d;i;f;y]` Modify items in a list or dictionary with `f`, in place or as a result. Related forms:
`@[d;i;:;y]` `v[i] : y`
`@[d;i;f]` `v[i] f: y`
`v[i] f:`

Amend at depth

`.[d;i;f;y]` Modifies items-at-depth in a list or dictionary with `f`, in place or as a result. Related forms:
`.[d;i;:;y]` `v[] : y` `v[j;k;…] : y`
`.[d;i;f]` `v[] f: y` `v[j;k;…] f: y`
`v[] f:` `v[j;k;…] f:`

INDICES

`! x` **Enumerate**
Integer vector `0` to `x-1` or the entries in a dictionary or directory on the K-tree.

`< x` **Grade Up and Down**

`> x` Permutations for ascending or descending order.

`= x` **Group**
List of indices which group matching items together.

`? x` **Range** Unique items.

`& x` **Where** Indices replicated `x` times.

DICTIONARIES

`. x` **Make/Unmake dictionary**
If `x` is list of items (Symbol ; Value ; optional attribute dictionary), result is a dictionary.
If *x* is a dictionary, result is a list of items of this form.

`~ x` **Attribute** Handle of variable's attribute directory.

Assignment `v` becomes

`v : y`	`y`
`v f: y`	`f[v;y]`
`v f:`	`f[v]`
`v :: y`	Global definition.

TRANSLATION

`$ x`	**Format**	Monadic, character representation.
`x $ y`		Dyadic, character rep. determined by integer or float left argument.
`x $ y`	Form	`0` and `0.0` convert characters to numeric. `` ` `` produces symbols. `{}` executes the expression `y`.
`. x`	**Evaluate**	Value of expression or variable.
`d @ x`		… executed in `d`, a dictionary on the K-tree.

EACH

`f' x`	**Each**	Apply `f` to each item of argument(s).
`x f' y`		Use `f'[x;y;z;…]` for higher valence.
`x f\: y`	**Each left**	Apply `f` dyadicaly to each item of `x` with `y`
`x f/: y`	**Each right**	Apply `f` dyadically to `x` with each item of `y`
`x f': y`	**Each pair**	Apply `f` dyadically to consecutive pairs of `y`, starting with `x` (if present).

OVER (/) AND SCAN (\)

`f/ x`	**Monadic** `f`	Apply `f` repeatedly to `x` until no change,
`n f/ x`		`n` times, or until `t` applied to result
`t f/ x`		gives `0`.
`f/[x;y;z;…]`	**Higher valence** `f`	Apply `f` repeatedly to `x` with items of other arguments.

`f[…f[f[x;`y_0`;`z_0`;…];`y_1`;`z_1`;…]…;`y_n`;`z_n`;…]`

For dyadic `f`,
`x f/ y` is `f/[x;y]`
`f/x` is `f/[`x_0`;1_x]`.

Scan (\) Assembles intermediate results of over iterations.

Mike Powell [mikepowell@home.com] © Sept 2000

K 2.0 Reference Card

UNIX K PROCESS

`k script.k -i 1234`[`</dev/null` disables input].

WINDOWS NT 4.0 K PROCESS

`k script.k -i 1234 -h 5678`[`kr` disables input].

ENVIRONMENT VARIABLES

KFONT	Data font (monospaced).
KLFONT	Label font.
KCOLOR	Data object colour.
KSWAP	UNIX Swap directory (default is /var/ktmp).

ATTRIBUTES

`v..a`	Arrangement
`v..bg`	Background colour
`v..fg`	Foreground colour
`v..c`	Class
`v..k`	Click
`v..kk`	Double click
`v..d`	Dependency
`v..e`	Editable
`v..f`	Format
`v..h`	Help
`v..l`	Label
`v..o`	Option list
`v..t`	Trigger
`v..u`	Update
`v..g`	Validation
`v..x`	Width
`v..y`	Height

SYSTEM VARIABLES

`_d`	Current directory.
`_v`	Current global variable under amendment.
`_t`	Current time.
`_h`	Host process (machine name).
`_p`	Host process (port).
`_i`	Shell command parameters; Location of current amendment.
`_w`	Handle of message source.
`_u`	Name of message sender.
`_n`	Nil.
`_f`	Self.

SYSTEM FUNCTIONS

`x _bin y`	Binary search for y in sorted x [`_binl` for each y].
`x _di y`	Delete items with indices y from x.
`x _dv y`	Delete value y from x [`_dvl` for each y].
`x _draw y`	x random selections from y.
`_gtime x`	GMT or local time (`_ltime`) as `yyyymmdd hhmmss`.
`_ic x`	Integer from char and char from integer (`_ci`).
`_jd x`	Julian day from `hhhhmmdd` and date from Julian (`_dj`).
`x _lsq y`	Least squares solution, W^j, of $Y_{ij}W^j=X_i$.
`x _dot y`	Dot product (`+/x*y`).
`x _mul y`	Matrix multiply.
`_inv x`	Matrix inverse.
`x _in y`	Is x in y? (`_lin` tests each x).
`x _sv y`	Evaluate items of y in base x.
`x _sm y`	Are the strings in x found in y?
`x _ss y`	Find all occurrences of strings of y in strings of x.
`_ssr[x;y;z]`	Replace substrings y found in x with z.
`x _vs y`	Expand items of y in base x.

MATH FUNCTIONS

`_abs x`	Absolute value.
`_floor x`	Integer part without comparison tolerance.
`_sin x`	Sine (inverse is `_arcsin`).
`_cos x`	Cosine (inverse is `_arccos`).
`_tan x`	Tangent (inverse is `_arctan`).
`_sinh x`	Hyperbolic sine.
`_cosh x`	Hyperbolic cosine.
`_tanh x`	Hyperbolic tangent
`_exp x`	Exponential (inverse is `_log`).
`_sqr x`	Square (inverse is `_sqrt`).

I/O & COMMUNICATION

`0: f`	Load file f as text; new lines mark items.
`f 0: x`	Save text x in file f; new lines inserted.
`` ` 0: x ``	Write text to session log.
`(s;w) 0: f`	Load text file f as fields. s is one of "IFCS" or blank; w is lengths. f is a filename or (filename;offset;length).
`1: f`	Load contents of K-file f as mapped data.
`2: f`	Copy contents of K-file f as data.
`f 1: x`	Save x in K-file f.
`(s;w) 1: f`	Load binary file f as fields. s is a C datatype, one of "`cbsifdCS`" or blank; w is lengths. f is a filename or (filename;offset;length).
`c i: f`	Load entire file f as characters (c), integers (i) or floats (d). f is a filename or (filename ; offset ; length).
`f 2: (e;t)`	Link object code.
`3: (n;p)`	Get communication handle of process p on machine n.
`3: h`	Close communication with partner whose handle is h.
`t 3: x`	Remote set.
`t 4: x`	Remote get.
`4: x`	Internal data type.
`5: x`	Executable form.
`f 5: c`	Synchronized file append.

CONTROLS & DEBUGGING

`\`	Abort.
`/text`	Comment.
`:`	Resume after stop.
`: x`	Resume with value; or Return value.
`' x`	Signal (or `'` alone).
`\ x`	Stop/Trace.

COMMANDS

`\.`	Attributes
`\b [char]`	Set or report break flag (s = stop, t=trace, n=none).
`\c b`	Console on (`1`) or off (`0`).
`\d [name]`	Set or report current directory.
`\v [dir]`	Entries in `dir`. [`^` is parent; ~ is attribute dir].
`\e [bool]`	Set or report error flag.
`\\`	Exit.
`Ctrl c`	Interrupt execution.
`\i [name]`	Invalid names.
`\l file`	Load script.
`\m`	Windows nonstandard commands `(i,h,c,f or l)`.
`\text`	`Text` passed to OS for evaluation.
`\p [digits]`	Set or report print precision.
`\r [seed]`	Set or report random seed.
`\kr file`	Make a K runtime program.
`\t [digits]`	Set or report timer.
`\s file`	Step through execution of file.
`\_`	System names.
`\t expr`	Measure execution time for expression.
`\w`	Workspace size (used, allocated, mapped).

HELP

`\`	Command summary.
`\0`	Help for I/O and data representation.
`\+`	Help for verbs.
`\'`	Help for adverbs.
`\:`	Help for assignment, functions & controls.