

The Select Reference Manual

DSL for array slices in Common Lisp, version 1.0.0

Steven Nunez <steve.nunez@inference.sg>

Copyright © 2017–2018 Steven Nunez

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the section entitled “Copying” is included exactly as in the original.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be translated as well.

Table of Contents

Copying	1
1 Introduction	3
1.1 Selection Specifiers	3
1.1.1 Extensions	3
1.2 Select Semantics	4
2 Systems	5
2.1 select	5
3 Files	7
3.1 Lisp	7
3.1.1 select.asd	7
3.1.2 select/package.lisp	7
3.1.3 select/select-dev.lisp	7
3.1.4 select/select.lisp	8
4 Packages	11
4.1 slct	11
5 Definitions	15
5.1 Exported definitions	15
5.1.1 Functions	15
5.1.2 Generic functions	15
5.1.3 Structures	17
5.2 Internal definitions	18
5.2.1 Macros	18
5.2.2 Functions	18
5.2.3 Generic functions	22
5.2.4 Structures	23
6 Change Log	25
Appendix A Indexes	27
A.1 Concepts	27
A.2 Functions	28
A.3 Variables	30
A.4 Data types	31

Copying

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1 Introduction

Select is a library for taking slices from array-like objects. The most frequently used form is:

```
(select object selection1 selection2 ...)
```

where each *selection* specifies a subset of subscripts along the corresponding axis. The selection specifications are found below.

1.1 Selection Specifiers

Selecting Single Values

A non-negative integer selects the corresponding index, while a negative integer selects an index counting backwards from the last index. For example:

```
(select #(0 1 2 3) 1)           ; => 1
(select #(0 1 2 3) -2)          ; => 2
```

These are called *singleton* slices. Each singleton slice drops the dimension: vectors become atoms, matrices become vectors, *etc.*.

Selecting Ranges

(range start end) selects subscripts *i* where start <= i <= end. When end is nil, the last index is included (cf. subseq). Each boundary is resolved according to the other rules if applicable, so you can use negative integers:

```
(select #(0 1 2 3) (range 1 3)) ; => #(1 2)
(select #(0 1 2 3) (range 1 -1)) ; => #(1 2)
```

Selecting All Subscripts of a Dimension

t selects all subscripts:

```
(select #2A((0 1 2)
             (3 4 5))
      t 1) ; => #(1 4)
```

Selecting with a Sequence

Sequences can be used to make specific selections from the object. For example:

```
(select #(0 1 2 3 4 5 6 7 8 9)
 (vector (range 1 3) 6 (range -2 -1))) ; => #(1 2 3 6 8 9)
(select #(0 1 2) '(2 2 1 0 0))          ; => #(2 2 1 0 0)
```

Using Bit Vectors as a Mask

Bit vectors can be used to select elements of arrays and sequences as well:

```
(select #(0 1 2 3 4) #*00110) ; => #(2 3)
```

1.1.1 Extensions

Section 1.1 describes the core functionality. The semantics can be extended, as you will see in the next section. The extensions in this section are provided by the library and prove useful in practice. Their implementation provides good examples of extending the library.

`including` is convenient if you want the selection to include the end of the range:

```
(select #(0 1 2 3) (including 1 2))
```

```
; => #(1 2), cf. (select ... (range 1 3))
```

`nodrop` is useful if you do not want to drop dimensions:

```
(select #(0 1 2 3) (nodrop 2))
; => #(2), cf. (select ... (range 2 3))
```

`head` and `tail` do the obvious:

```
(select #(0 1 2 3) (head 2))      ; => #(0 1)
(select #(0 1 2 3) (tail 2))     ; => #(2 3)
```

All of these are trivial to implement. If there is something you are missing, you can easily extend `select`. Pull request are welcome.

`ref` is a version of `select` that always returns a single element, so it can only be used with singleton slices.

1.2 Select Semantics

Arguments of `select`, except the first one, are meant to be resolved using `canonical-representation`, in the `select-dev` package. If you want to extend `select`, you should define methods for `canonical-representation`. See the source code for the best examples. Below is a simple example that extends the semantics with ordinal numbers.

```
(defmacro define-ordinal-selection (number)
  (check-type number (integer 0))
  `(defmethod select-dev:canonical-representation
    ((axis integer)
     (slice (eql ',(intern (format nil "~:@(~:~)" number))))))
    (assert (< ,number axis))
    (select-dev:canonical-singleton ,number)))

(define-ordinal-selection 1)
(define-ordinal-selection 2)
(define-ordinal-selection 3)

(select #(0 1 2 3 4 5) (range 'first 'third)) ; => #(1 2)
```

Note the following:

- The value returned by `canonical-representation` needs to be constructed using `canonical-singleton`, `canonical-range`, or `canonical-sequence`. You should not use the internal representation directly as it is subject to change.
- You can assume that `axis` is an integer: this is the default. An object may define a more complex mapping (such as, for example, named rows & columns), but unless a method specialized to that is found, `canonical-representation` will just query its dimension (with `axis-dimension`) and try to find a method that works on integers.
- You need to make sure that the subscript is valid, hence the assertion.

2 Systems

The main system appears first, followed by any subsystem dependency.

2.1 select

Author Steven Nunez <steve.nunez@inference.sg>

Home Page

<http://inference.sg/projects/select/>

Source Control

(:git "git://github.com/symbolics/select")

License MIT

Description

DSL for array slices in Common Lisp.

Long Description

Select is a facility for selecting portions of sequences or arrays. It provides:

An user interface for taking slices (elements selected by the Cartesian product of vectors of subscripts for each axis) of array-like objects. The most important function is 'select'. Unless you want to define a method for this (besides what is already implemented), this is all you need from this library.

An extensible DSL for selecting a subset of valid subscripts. This is useful if, for example, you want to resolve column names in a data frame in your implementation of slice.

A set of utility functions for traversing slices in array-like objects.

Version 1.0.0

Dependencies

- alexandria
- anaphora
- let-plus

Source [select.asd], page 7, (file)

Directory s:/src/select/

Components

- [package.lisp], page 7, (file)
- [select-dev.lisp], page 7, (file)
- [select.lisp], page 8, (file)

3 Files

Files are sorted by type and then listed depth-first from the systems components trees.

3.1 Lisp

3.1.1 select.asd

Location `select.asd`

Systems `[select]`, page 5, (system)

3.1.2 select/package.lisp

Parent `[select]`, page 5, (system)

Location `package.lisp`

Packages `[slct]`, page 11,

3.1.3 select/select-dev.lisp

Dependency

`[package.lisp]`, page 7, (file)

Parent `[select]`, page 5, (system)

Location `select-dev.lisp`

Internal Definitions

- `[all-singleton-representations?]`, page 18, (function)
- `[axis-dimension]`, page 22, (generic function)
- `[canonical-range]`, page 18, (function)
- `[canonical-range]`, page 23, (structure)
- `[canonical-range-end]`, page 18, (function)
- `[(setf canonical-range-end)]`, page 18, (function)
- `[canonical-range-p]`, page 18, (function)
- `[canonical-range-start]`, page 18, (function)
- `[(setf canonical-range-start)]`, page 18, (function)
- `[canonical-representation]`, page 22, (generic function)
- `[canonical-representation]`, page 22, (method)
- `[canonical-representation]`, page 22, (method)
- `[canonical-representation]`, page 22, (method)
- `[canonical-representation]`, page 22, (method)
- `[canonical-representation]`, page 22, (method)
- `[canonical-representation]`, page 22, (method)
- `[canonical-representation]`, page 23, (method)
- `[canonical-representation]`, page 23, (method)
- `[canonical-representations]`, page 18, (function)
- `[canonical-sequence]`, page 19, (function)
- `[canonical-sequence]`, page 23, (structure)
- `[canonical-sequence-p]`, page 19, (function)

- [canonical-sequence-vector], page 19, (function)
- [(setf canonical-sequence-vector)], page 19, (function)
- [canonical-singleton], page 19, (function)
- [column-major-setup], page 19, (function)
- [copy-canonical-range], page 19, (function)
- [copy-canonical-sequence], page 19, (function)
- [make-canonical-range], page 20, (function)
- [make-canonical-sequence], page 20, (function)
- [representation-dimension], page 21, (function)
- [representation-dimensions], page 21, (function)
- [representation-initial-value], page 21, (function)
- [representation-iterator], page 21, (function)
- [row-major-setup], page 21, (function)
- [select-reserved-symbol?], page 21, (function)
- [singleton-representation?], page 21, (function)
- [traverse-representations], page 18, (macro)

3.1.4 select/select.lisp

Dependency

[select-dev.lisp], page 7, (file)

Parent

[select], page 5, (system)

Location

select.lisp

Exported Definitions

- [head], page 15, (function)
- [including], page 15, (function)
- [including], page 17, (structure)
- [mask], page 15, (generic function)
- [mask], page 15, (method)
- [nodrop], page 15, (function)
- [nodrop], page 17, (structure)
- [range], page 15, (function)
- [range], page 17, (structure)
- [ref], page 15, (generic function)
- [ref], page 16, (method)
- [(setf ref)], page 16, (method)
- [(setf ref)], page 16, (generic function)
- [select], page 16, (generic function)
- [select], page 16, (method)
- [select], page 16, (method)
- [(setf select)], page 16, (method)
- [(setf select)], page 16, (generic function)
- [tail], page 15, (function)
- [which], page 16, (generic function)

- [which], page 16, (method)

Internal Definitions

- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [copy-including], page 19, (function)
- [copy-nodrop], page 19, (function)
- [copy-range], page 19, (function)
- [including-end], page 20, (function)
- [(setf including-end)], page 20, (function)
- [including-p], page 20, (function)
- [including-start], page 20, (function)
- [(setf including-start)], page 20, (function)
- [make-including], page 20, (function)
- [make-nodrop], page 20, (function)
- [make-range], page 20, (function)
- [nodrop-index], page 20, (function)
- [(setf nodrop-index)], page 20, (function)
- [nodrop-p], page 20, (function)
- [range-end], page 20, (function)
- [(setf range-end)], page 20, (function)
- [range-p], page 21, (function)
- [range-start], page 21, (function)
- [(setf range-start)], page 21, (function)

4 Packages

Packages are listed by definition order.

4.1 `slct`

SELECT is a facility for selecting portions of sequences or arrays.

Source `[package.lisp]`, page 7, (file)

Nickname `select`

Use List

- `let-plus`
- `anaphora`
- `alexandria.0.dev`
- `common-lisp`

Exported Definitions

- `[head]`, page 15, (function)
- `[including]`, page 15, (function)
- `[including]`, page 17, (structure)
- `[mask]`, page 15, (generic function)
- `[mask]`, page 15, (method)
- `[nodrop]`, page 15, (function)
- `[nodrop]`, page 17, (structure)
- `[range]`, page 15, (function)
- `[range]`, page 17, (structure)
- `[ref]`, page 15, (generic function)
- `[ref]`, page 16, (method)
- `[(setf ref)]`, page 16, (method)
- `[(setf ref)]`, page 16, (generic function)
- `[select]`, page 16, (generic function)
- `[select]`, page 16, (method)
- `[select]`, page 16, (method)
- `[(setf select)]`, page 16, (method)
- `[(setf select)]`, page 16, (generic function)
- `[tail]`, page 15, (function)
- `[which]`, page 16, (generic function)
- `[which]`, page 16, (method)

Internal Definitions

- `[all-singleton-representations?]`, page 18, (function)
- `[axis-dimension]`, page 22, (generic function)
- `[canonical-range]`, page 18, (function)
- `[canonical-range]`, page 23, (structure)
- `[canonical-range-end]`, page 18, (function)
- `[(setf canonical-range-end)]`, page 18, (function)

- [canonical-range-p], page 18, (function)
- [canonical-range-start], page 18, (function)
- [(setf canonical-range-start)], page 18, (function)
- [canonical-representation], page 22, (generic function)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 22, (method)
- [canonical-representation], page 23, (method)
- [canonical-representation], page 23, (method)
- [canonical-representations], page 18, (function)
- [canonical-sequence], page 19, (function)
- [canonical-sequence], page 23, (structure)
- [canonical-sequence-p], page 19, (function)
- [canonical-sequence-vector], page 19, (function)
- [(setf canonical-sequence-vector)], page 19, (function)
- [canonical-singleton], page 19, (function)
- [column-major-setup], page 19, (function)
- [copy-canonical-range], page 19, (function)
- [copy-canonical-sequence], page 19, (function)
- [copy-including], page 19, (function)
- [copy-nodrop], page 19, (function)
- [copy-range], page 19, (function)
- [including-end], page 20, (function)
- [(setf including-end)], page 20, (function)
- [including-p], page 20, (function)
- [including-start], page 20, (function)
- [(setf including-start)], page 20, (function)
- [make-canonical-range], page 20, (function)
- [make-canonical-sequence], page 20, (function)
- [make-including], page 20, (function)
- [make-nodrop], page 20, (function)
- [make-range], page 20, (function)
- [nodrop-index], page 20, (function)
- [(setf nodrop-index)], page 20, (function)
- [nodrop-p], page 20, (function)
- [range-end], page 20, (function)
- [(setf range-end)], page 20, (function)

- `[range-p]`, page 21, (function)
- `[range-start]`, page 21, (function)
- `[(setf range-start)]`, page 21, (function)
- `[representation-dimension]`, page 21, (function)
- `[representation-dimensions]`, page 21, (function)
- `[representation-initial-value]`, page 21, (function)
- `[representation-iterator]`, page 21, (function)
- `[row-major-setup]`, page 21, (function)
- `[select-reserved-symbol?]`, page 21, (function)
- `[singleton-representation?]`, page 21, (function)
- `[traverse-representations]`, page 18, (macro)

5 Definitions

Definitions are sorted by export status, category, package, and then by lexicographic order.

5.1 Exported definitions

5.1.1 Functions

head *COUNT* [Function]
First *COUNT* indexes.

Package [slct], page 11,
Source [select.lisp], page 8, (file)

including *START END* [Function]
Range, including both ends.

Package [slct], page 11,
Source [select.lisp], page 8, (file)

nodrop *INDEX* [Function]
Select a single index, but do not drop a dimension.

Package [slct], page 11,
Source [select.lisp], page 8, (file)

range *START END* [Function]
Range, including *START*, excluding *END*.

Package [slct], page 11,
Source [select.lisp], page 8, (file)

tail *COUNT* [Function]
Last *COUNT* indexes.

Package [slct], page 11,
Source [select.lisp], page 8, (file)

5.1.2 Generic functions

mask *PREDICATE SEQUENCE* [Generic Function]
Map sequence into a simple-bit-vector, using 1 when *PREDICATE* yields true, 0 otherwise.

Package [slct], page 11,
Source [select.lisp], page 8, (file)

Methods

mask *PREDICATE* (*SEQUENCE* *sequence*) [Method]

ref *OBJECT &rest SUBSCRIPTS* [Generic Function]
Return the element of *OBJECT* specified by *SUBSCRIPTS*.

Package [slct], page 11,
Source [select.lisp], page 8, (file)
Writer [(setf ref)], page 16, (generic function)

Methods

ref (*ARRAY array*) **&rest** *SUBSCRIPTS* [Method]

(setf ref) *VALUE OBJECT &rest SUBSCRIPTS* [Generic Function]
Stores *VALUE* into the place specified by *SUBSCRIPTS*.

Package [slct], page 11,

Source [select.lisp], page 8, (file)

Reader [ref], page 15, (generic function)

Methods

(setf ref) *VALUE (ARRAY array) &rest SUBSCRIPTS* [Method]

select *OBJECT &rest SELECTIONS* [Generic Function]
Return the slices of *OBJECT* specified by *SELECTIONS*.

Package [slct], page 11,

Source [select.lisp], page 8, (file)

Writer [(setf select)], page 16, (generic function)

Methods

select (*LST list*) **&rest** *SELECTIONS* [Method]
Select from *LST* the subscripts or range specified in *SELECTIONS*. *SELECTIONS* must be a *VECTOR*, *LIST* or *RANGE*.

select (*ARRAY array*) **&rest** *SELECTIONS* [Method]
Return the *SELECTIONS* in the given *ARRAY*.

(setf select) *VALUE OBJECT &rest SELECTIONS* [Generic Function]
Stores *VALUES* into the locations given by *SELECTIONS*.

Package [slct], page 11,

Source [select.lisp], page 8, (file)

Reader [select], page 16, (generic function)

Methods

(setf select) *VALUE (ARRAY array) &rest SELECTIONS* [Method]

which *PREDICATE SEQUENCE* [Generic Function]
Return an index of the positions in *SEQUENCE* which satisfy *PREDICATE*.

Package [slct], page 11,

Source [select.lisp], page 8, (file)

Methods

which *PREDICATE (SEQUENCE sequence)* [Method]

5.1.3 Structures

including ()	[Structure]
Range, including both ends.	
Package	[slct], page 11,
Source	[select.lisp], page 8, (file)
Direct superclasses	
	structure-object (structure)
Direct methods	
	[canonical-representation], page 22, (method)
Direct slots	
start	[Slot]
Readers	[including-start], page 20, (function)
Writers	[(setf including-start)], page 20, (function)
end	[Slot]
Readers	[including-end], page 20, (function)
Writers	[(setf including-end)], page 20, (function)
nodrop ()	[Structure]
Select a single index, but don't drop a dimension.	
Package	[slct], page 11,
Source	[select.lisp], page 8, (file)
Direct superclasses	
	structure-object (structure)
Direct methods	
	[canonical-representation], page 22, (method)
Direct slots	
index	[Slot]
Readers	[nodrop-index], page 20, (function)
Writers	[(setf nodrop-index)], page 20, (function)
range ()	[Structure]
Range, including start, excluding end.	
Package	[slct], page 11,
Source	[select.lisp], page 8, (file)
Direct superclasses	
	structure-object (structure)
Direct methods	
	[canonical-representation], page 22, (method)
Direct slots	
start	[Slot]
Readers	[range-start], page 21, (function)
Writers	[(setf range-start)], page 21, (function)

end [Slot]
Readers [range-end], page 20, (function)
Writers [(setf range-end)], page 20, (function)

5.2 Internal definitions

5.2.1 Macros

traverse-representations (*SUBSCRIPTS REPRESENTATIONS &key* [Macro]
INDEX SETUP) &body BODY

Loops over all possible subscripts in REPRESENTATIONS, making them available in SUBSCRIPTS during the execution of BODY. The iterator is constructed using the function SETUP (see for example ROW-MAJOR-SETUP). When INDEX is given, a variable with that name is provided, containing an index that counts iterations.

Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)

5.2.2 Functions

all-singleton-representations? *REPRESENTATIONS* [Function]
 Test if all canonical representations are singletons.

Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)

canonical-range *START END* [Function]
 Canonical representation of a contiguous set of array indices from START (inclusive) to END (exclusive).

Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)

canonical-range-end *INSTANCE* [Function]
 (setf canonical-range-end) *VALUE INSTANCE* [Function]

Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)

canonical-range-p *OBJECT* [Function]

Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)

canonical-range-start *INSTANCE* [Function]
 (setf canonical-range-start) *VALUE INSTANCE* [Function]

Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)

canonical-representations *AXES SELECTIONS* [Function]
 Return the canonical representations of SELECTIONS given the corresponding AXES, checking for matching length.

Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)

canonical-sequence *SEQUENCE* [Function]

Canonical representation of array indexes from canonical-sequence *SEQUENCE*.

May share structure. Vectors of the upgraded type of (SIMPLE-ARRAY ARRAY-INDEX (*)) are preferred for efficiency, otherwise they are coerced.

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

canonical-sequence-p *OBJECT* [Function]

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

canonical-sequence-vector *INSTANCE* [Function]

(setf canonical-sequence-vector) *VALUE INSTANCE* [Function]

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

canonical-singleton *INDEX* [Function]

Canonical representation of a singleton index (a nonnegative integer, which is a valid array index).

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

column-major-setup *REPRESENTATIONS TERMINATOR* [Function]

Return SUBSCRIPTS (a list) and ITERATOR (a closure, no arguments) that increments the contents of SUBSCRIPTS in column-major order. TERMINATOR is called when all subscripts have been visited.

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

copy-canonical-range *INSTANCE* [Function]

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

copy-canonical-sequence *INSTANCE* [Function]

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

copy-including *INSTANCE* [Function]

Package [slct], page 11,

Source [select.lisp], page 8, (file)

copy-nodrop *INSTANCE* [Function]

Package [slct], page 11,

Source [select.lisp], page 8, (file)

copy-range *INSTANCE* [Function]

Package [slct], page 11,

Source [select.lisp], page 8, (file)

including-end <i>INSTANCE</i>	[Function]
(setf including-end) <i>VALUE INSTANCE</i>	[Function]
Package [slct], page 11,	
Source [select.lisp], page 8, (file)	
including-p <i>OBJECT</i>	[Function]
Package [slct], page 11,	
Source [select.lisp], page 8, (file)	
including-start <i>INSTANCE</i>	[Function]
(setf including-start) <i>VALUE INSTANCE</i>	[Function]
Package [slct], page 11,	
Source [select.lisp], page 8, (file)	
make-canonical-range &key (<i>START START</i>) (<i>END END</i>)	[Function]
Package [slct], page 11,	
Source [select-dev.lisp], page 7, (file)	
make-canonical-sequence &key (<i>VECTOR VECTOR</i>)	[Function]
Package [slct], page 11,	
Source [select-dev.lisp], page 7, (file)	
make-including &key (<i>START START</i>) (<i>END END</i>)	[Function]
Package [slct], page 11,	
Source [select.lisp], page 8, (file)	
make-nodrop &key (<i>INDEX INDEX</i>)	[Function]
Package [slct], page 11,	
Source [select.lisp], page 8, (file)	
make-range &key (<i>START START</i>) (<i>END END</i>)	[Function]
Package [slct], page 11,	
Source [select.lisp], page 8, (file)	
nodrop-index <i>INSTANCE</i>	[Function]
(setf nodrop-index) <i>VALUE INSTANCE</i>	[Function]
Package [slct], page 11,	
Source [select.lisp], page 8, (file)	
nodrop-p <i>OBJECT</i>	[Function]
Package [slct], page 11,	
Source [select.lisp], page 8, (file)	
range-end <i>INSTANCE</i>	[Function]
(setf range-end) <i>VALUE INSTANCE</i>	[Function]
Package [slct], page 11,	
Source [select.lisp], page 8, (file)	

- range-p** *OBJECT* [Function]
Package [slct], page 11,
Source [select.lisp], page 8, (file)
- range-start** *INSTANCE* [Function]
(setf range-start) *VALUE INSTANCE* [Function]
Package [slct], page 11,
Source [select.lisp], page 8, (file)
- representation-dimension** *REPRESENTATION* [Function]
 Return the dimension of a canonical-representation, or NIL for singleton selections (they are dropped).
Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)
- representation-dimensions** *REPRESENTATIONS* [Function]
 Return a list for the dimensions of canonical representations, dropping singletons.
Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)
- representation-initial-value** *REPRESENTATION* [Function]
 Initial value for iteration.
Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)
- representation-iterator** *REPRESENTATION CARRY CONS* [Function]
 Return a closure that sets the car of CONS to the next value each time it is called, resetting and calling CARRY when it reaches the end of its range.
Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)
- row-major-setup** *REPRESENTATIONS TERMINATOR* [Function]
 Return SUBSCRIPTS (a list) and ITERATOR (a closure, no arguments) that increments the contents of SUBSCRIPTS in row-major order. TERMINATOR is called when all subscripts have been visited.
Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)
- select-reserved-symbol?** *SYMBOL* [Function]
 Test if SYMBOL has special semantics for SELECTION.
Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)
- singleton-representation?** *REPRESENTATION* [Function]
 Test if a canonical REPRESENTATION is a singleton.
Package [slct], page 11,
Source [select-dev.lisp], page 7, (file)

5.2.3 Generic functions

axis-dimension *AXIS* [Generic Function]

Return the dimension of axis. Needs to be defined for non-integer axes.

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

canonical-representation *AXIS SELECTION* [Generic Function]

Canonical representation of SELECTION, given information in AXIS. The default methods use dimensions as AXIS.

Each selection needs to be resolved into a canonical representation, which is either a singleton, a range, or a sequence of subscripts. They should only be constructed with the corresponding CANONICAL-SINGLETON, CANONICAL-RANGE and CANONICAL-SEQUENCE functions.

@c(CANONICAL-REPRESENTATION) needs to ensure that the represented subscripts are valid for the axis.

Unless a specialized method is found, the dimension of the axis is queried with AXIS-DIMENSION and resolution is attempted using the latter. Methods that resolve symbols should test them with SELECT-RESERVED-SYMBOL? and use CALL-NEXT-METHOD.

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

Methods

canonical-representation *AXIS (SELECTION nodrop)* [Method]

The canonical representation for NODROP.

Source [select.lisp], page 8, (file)

canonical-representation *AXIS (SELECTION range)* [Method]

The canonical representation for RANGE.

Source [select.lisp], page 8, (file)

canonical-representation *AXIS (SELECTION including)* [Method]

The canonical representation for INCLUDING.

Source [select.lisp], page 8, (file)

canonical-representation *AXIS SELECTION* [Method]

canonical-representation *AXIS (CANONICAL-RANGE canonical-range)* [Method]

canonical-representation *AXIS (CANONICAL-SEQUENCE canonical-sequence)* [Method]

canonical-representation *(AXIS integer) (SLICE null)* [Method]

canonical-representation *(AXIS integer) (SELECTION integer)* [Method]

canonical-representation *AXIS (SELECTION sequence)* [Method]

`canonical-representation` (*AXIS* integer) (*SELECTION* (eq1 t)) [Method]

`canonical-representation` *AXIS* (*SELECTION* bit-vector) [Method]

5.2.4 Structures

`canonical-range` () [Structure]
 Canonical representation of a contiguous set of array indices from *START* (inclusive) to *END* (exclusive).

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

Direct superclasses
 structure-object (structure)

Direct methods
 [canonical-representation], page 22, (method)

Direct slots

`start` [Slot]

Type alexandria.0.dev:array-index

Readers [canonical-range-start], page 18, (function)

Writers [(setf canonical-range-start)], page 18, (function)

`end` [Slot]

Type alexandria.0.dev:array-index

Readers [canonical-range-end], page 18, (function)

Writers [(setf canonical-range-end)], page 18, (function)

`canonical-sequence` () [Structure]
 Canonical representation of a sequence of array indexes.

Package [slct], page 11,

Source [select-dev.lisp], page 7, (file)

Direct superclasses
 structure-object (structure)

Direct methods
 [canonical-representation], page 22, (method)

Direct slots

`vector` [Slot]

Type (simple-array alexandria.0.dev:array-index (*))

Readers [canonical-sequence-vector], page 19, (function)

Writers [(setf canonical-sequence-vector)], page 19, (function)

6 Change Log

Select was originally called slice (<https://github.com/tpapp/cl-slice>) and written by Tamas K. Papp. Since it was abandoned in 2017 (<https://tpapp.github.io/post/orphaned-lisp-libraries/>), I have taken it over to be part of a rebooted Common Lisp statistics library. Changes in this version include:

Documentation Improvements

- Move to HTML based documentation system
- Docs now on github.io

Test Improvements

- Ported to FiveAM and refactored
- Improved test coverage
- Added failure messages to aid debugging
- Added tests for selection iteration

Enhancements

- Renamed to 'cons' to 'range'
- Range now handles (range x x) => nil
- Selections work identically on sequences; previously differed between lists and vectors
- Selections may be specified using a list; previously could only be a vector
- Sequence selections now honor fill-pointer, if any

Bug Fixes

- Range now handles END = (length <sequence>)
- Selecting from a list no longer drops dimension

Appendix A Indexes

A.1 Concepts

F

File, Lisp, <code>select.asd</code>	7
File, Lisp, <code>select/package.lisp</code>	7
File, Lisp, <code>select/select-dev.lisp</code>	7
File, Lisp, <code>select/select.lisp</code>	8

L

Lisp File, <code>select.asd</code>	7
Lisp File, <code>select/package.lisp</code>	7
Lisp File, <code>select/select-dev.lisp</code>	7
Lisp File, <code>select/select.lisp</code>	8

S

<code>select.asd</code>	7
<code>select/package.lisp</code>	7
<code>select/select-dev.lisp</code>	7
<code>select/select.lisp</code>	8

A.2 Functions

(
(setf canonical-range-end)	18
(setf canonical-range-start)	18
(setf canonical-sequence-vector)	19
(setf including-end)	20
(setf including-start)	20
(setf nodrop-index)	20
(setf range-end)	20
(setf range-start)	21
(setf ref)	16
(setf select)	16

A

all-singleton-representations?	18
axis-dimension	22

C

canonical-range	18
canonical-range-end	18
canonical-range-p	18
canonical-range-start	18
canonical-representation	22, 23
canonical-representations	18
canonical-sequence	19
canonical-sequence-p	19
canonical-sequence-vector	19
canonical-singleton	19
column-major-setup	19
copy-canonical-range	19
copy-canonical-sequence	19
copy-including	19
copy-nodrop	19
copy-range	19

F

Function, (setf canonical-range-end)	18
Function, (setf canonical-range-start)	18
Function, (setf canonical-sequence-vector)	19
Function, (setf including-end)	20
Function, (setf including-start)	20
Function, (setf nodrop-index)	20
Function, (setf range-end)	20
Function, (setf range-start)	21
Function, all-singleton-representations?	18
Function, canonical-range	18
Function, canonical-range-end	18
Function, canonical-range-p	18
Function, canonical-range-start	18
Function, canonical-representations	18
Function, canonical-sequence	19
Function, canonical-sequence-p	19
Function, canonical-sequence-vector	19
Function, canonical-singleton	19
Function, column-major-setup	19
Function, copy-canonical-range	19
Function, copy-canonical-sequence	19
Function, copy-including	19
Function, copy-nodrop	19

Function, copy-range	19
Function, head	15
Function, including	15
Function, including-end	20
Function, including-p	20
Function, including-start	20
Function, make-canonical-range	20
Function, make-canonical-sequence	20
Function, make-including	20
Function, make-nodrop	20
Function, make-range	20
Function, nodrop	15
Function, nodrop-index	20
Function, nodrop-p	20
Function, range	15
Function, range-end	20
Function, range-p	21
Function, range-start	21
Function, representation-dimension	21
Function, representation-dimensions	21
Function, representation-initial-value	21
Function, representation-iterator	21
Function, row-major-setup	21
Function, select-reserved-symbol?	21
Function, singleton-representation?	21
Function, tail	15

G

Generic Function, (setf ref)	16
Generic Function, (setf select)	16
Generic Function, axis-dimension	22
Generic Function, canonical-representation	22
Generic Function, mask	15
Generic Function, ref	15
Generic Function, select	16
Generic Function, which	16

H

head	15
------	----

I

including	15
including-end	20
including-p	20
including-start	20

M

Macro, <code>traverse-representations</code>	18
<code>make-canonical-range</code>	20
<code>make-canonical-sequence</code>	20
<code>make-including</code>	20
<code>make-nodrop</code>	20
<code>make-range</code>	20
<code>mask</code>	15
Method, <code>(setf ref)</code>	16
Method, <code>(setf select)</code>	16
Method, <code>canonical-representation</code>	22, 23
Method, <code>mask</code>	15
Method, <code>ref</code>	16
Method, <code>select</code>	16
Method, <code>which</code>	16

N

<code>nodrop</code>	15
<code>nodrop-index</code>	20
<code>nodrop-p</code>	20

R

<code>range</code>	15
<code>range-end</code>	20
<code>range-p</code>	21
<code>range-start</code>	21
<code>ref</code>	15, 16
<code>representation-dimension</code>	21
<code>representation-dimensions</code>	21
<code>representation-initial-value</code>	21
<code>representation-iterator</code>	21
<code>row-major-setup</code>	21

S

<code>select</code>	16
<code>select-reserved-symbol?</code>	21
<code>singleton-representation?</code>	21

T

<code>tail</code>	15
<code>traverse-representations</code>	18

W

<code>which</code>	16
--------------------------	----

A.3 Variables

E

`end` 17, 18, 23

I

`index` 17

S

`Slot, end` 17, 18, 23

`Slot, index` 17

`Slot, start` 17, 23

`Slot, vector` 23

`start` 17, 23

V

`vector` 23

A.4 Data types

C

canonical-range.....	23
canonical-sequence.....	23

I

including.....	17
----------------	----

N

nodrop.....	17
-------------	----

P

Package, slct.....	11
--------------------	----

R

range.....	17
------------	----

S

select.....	5
slct.....	11
Structure, canonical-range.....	23
Structure, canonical-sequence.....	23
Structure, including.....	17
Structure, nodrop.....	17
Structure, range.....	17
System, select.....	5

