XLISP Functions to Support Access to C Structures

First, a macro to define a C Record:

(DEFINE-CRECORD name (field-definition...))

Where field-definition is:

(fieldname type &optional size)

Where type is one of:

char, uchar, short, ushort, int, uint, long, ulong, ptr

Instead of a single field, a field-definition can be a list of field

definitions. In this case, each field in the list occupies the same

space in the crecord and the size of the union field is the maximum

of the sizes of each of the component fields.

DEFINE-CRECORD creates the following macros:

(MAKE-name &optional size) --> cpointer

(name-ADDRESS rec i) --> cpointer

and for each field:

(name-fieldname rec &optional i) --> fieldvalue

(name-fieldname-ADDRESS rec) --> cpointer to field

(SET-name-fieldname! rec value &optional i)

as well as the following variables:

name-SIZE for the size of the entire record

name-fieldname-OFFSET for the offset to the start of each field

Note: If the optional i parameter is supplied for one of the SET- macros,

it's position is reversed with value so the calling syntax is really:

(SET-name-fieldname! rec i value)

For example:

(define-crecord foo

((a ptr)

(b long 10)))

defines the macros:

(MAKE-FOO &optional size)

(FOO-ADDRESS rec i)

(FOO-A rec &optional i)

(FOO-B-ADDRESS rec &optional i)

(SET-FOO-A! rec value &optional i)

(FOO-B rec &optional i)

(FOO-B-ADDRESS rec &optional i)

(SET-FOO-B! rec value &optional i)

and the variables:

FOO-SIZE

FOO-A-OFFSET

FOO-B-OFFSET

Supporting functions:

(ALLOCATE-CMEMORY size) --> cpointer

(FREE-CMEMORY cpointer)

(GET-CRECORD-FIELD cpointer offset type-id)

(GET-CRECORD-FIELD-ADDRESS cpointer offset) --> cpointer

(SET-CRECORD-FIELD! cpointer offset type-id value)

(GET-CRECORD-STRING cpointer offset length) --> string

(SET-CRECORD-STRING! cpointer offset length string)

(GET-CRECORD-TYPE-SIZE type-id) --> size

(NULL-POINTER? cpointer) --> #f if pointer has been freed

(CRECORD-TYPE type-symbol) --> type-id