```
****************************************************************************
*                                                                          *
*                  the nancy grouplespace manager                          *
*                                                                          *
*                       by Geoffrey P. Coco                                *
*                                                                          *
****************************************************************************
```

DEFINITIONS

Nancy is a homebrew database manager designed specifically for the VEOS project.

VEOS data elements are extraordinarily general. These data elements are called 'grouples' and their fields are called 'words'. The precise nature of the grouple is the following: a grouple consists of one or more possible empty words (each stored as a null-terminated string) and an integer which records the number of words in the grouple.

```
      an example grouple:  [ (each) (field) (is a) ('word') ]
```

The information stored in the grouplespace can be shared amongst the distinct parts of an entity as well as other entities through grouple passing. 'Information', here, means *anything* that is not implicit at runtime, which is everything. To reiterate, there are no restrictions on what kind of data can be stored in grouples.

The challenge of nancy is to support a clean, consistent, and powerful language for prototype rule bases and entity communication.

MODUS OPERANDI

Once a VEOS entity has been compiled and linked with the nancy library (see HOW TO.. for more details), remaining nancy setup and configuration occurs at runtime. The rest of this section describes these runtime capabilities.

The grouplespace can be filled one grouple at a time or in bulk. A client can insert a single grouple at any time (see insertGrouple). The same client can use nancy read the contents of a disk file, parse the data into grouples, and store the grouples in the grouplespace (see readGroupleSpace). Nancy can also do the reverse. For example, a client could save the contents of the grouplespace at periodic intervals for backup in case of crash (see writeGroupleSpace).

Nancy maintains client identification through the passing of standard nancy identification records. This client ID record consists of an enitity ID (the unique entity net address), a C primitive name, and a process ID. In some cases, this client ID is used to enforce a client's temporary exclusive rights to data. In some cases, the client ID is used to reference that client's data (keep reading for details). In addition, this client ID mechanism can be adapted to enforce individual and group capabilities.

Next, I must introduce the notion of single grouple selection. Each unique nancy client can select at most one grouple within the grouplespace at a time. To achieve this, maintains selection data for each client. Many nancy functions operate only on a client's selected grouple. For example, killGrouple and checkOutGrouple both perform actions on the client's selected grouple.

A client can remove grouples from the grouplespace one at a time (see killGrouple). Nancy does not support bulk deletions because a given client's delete capabilities may vary from grouple to grouple. A client can do bulk deletions easily by putting killGrouple(...DeleteFromTop...) inside a loop.

Nancy supports the Match/Substitute/Execute paradigm by virtue of it's query and controlled modify capabilities.

A client can request a match of any word from the grouplespace with several powerful parameters (see querySpace). In doing do, the client may select the result, obtain a copy of the result, or determine that no grouples match the query. Sequential searching is possible by selecting each successive query until the end of the grouple space is reached. Nancy also supports wildcard matching. This feature is useful if a client needs to examine each grouple in the grouple space sequentially.

Nancy allows each client to 'check out' an infinite number of grouples. When a grouple has been checked out, it cannot be deleted or modified until the same client has checked it back in again. The client may check in a changed version (or a completely different one) from the originally checked out grouple. This mechanism serves as a safe substitute operation.

Furthermore, the execute operation can be handled by the client. For example, all VEOS clients adhere to the format where the first word of 'execute' grouples contain a function symbol. The client, in this case the VEOS shell, could interpret these 'primitive' names and dispatch the function call.

N O T E : The nancy interface is a client's only safe way to interact with the grouplespace. Any attempt to manipulate the grouplespace other than by making calls to nancy will surely result in death.

HOW TO USE NANCY FROM WITHIN A VEOS ENTITY PRIMITIVE

1. Include world.h.

2. Make nancy function calls directly from C.

3. Link with: shell.o,
   with options: -lnancy.

IMPLEMENTAION

The following text has been copied from the nancy library source files and nancy.h

```
/*************************************************************************
 *                      powerful metafunctions                             *
 ************************************************************************/

/*************************************************************************
 postDataAsGrouple

Util to transform given list of string data into a grouple and insert into
the grouplespace.  First param (iWords) is number of strings; subsequent
params are the individual strings themselves.

postDataAsGrouple() appends the new grouple to the logical end of the
grouplespace
*/

TVeosErr postDataAsGrouple(int iWords, ...)


/*************************************************************************
setupDataGrouple

Use this function to initialize a single data grouple in the grouplespace.
Upon success, a grouple with only the given keyword will exist in the
grouplespace.

The caller should then use changeDataGrouple to maintain the existing data
grouple.
*/

TVeosErr setupDataGrouple(char *sKeyWord)
```

```
/***********************************************************************
changeDataGrouple

Use this function to maintain a single data grouple with a unique name.
Caller may use setupDataGrouple() to create the data grouple beforehand.

Different from replaceDataGrouple in that caller can change specified word(s)
in the unique grouple without passing a new grouple.

Caller may make several changes at once.  iPairs parameter specifies how many
changes to make.  Pass each change as a index/word pair described below.

pairs: (int iWordIndex, char *pNewWord)

To make a change, pass the grouple word index followed by the replacement
word. The word may be anything.  If nil is passed as a word, that word will
be deleted from the grouple and all subsequent words' indices will decrease
by one.
*/

TVeosErr changeDataGrouple(char *sKeyWord, TPWhoInfo pClient, int iPairs,
...)


/***********************************************************************
replaceDataGrouple

Use this function to maintain a single data grouple with a unique name.
Caller may use setupDataGrouple() to create the data grouple beforehand.

Pass a grouple which should replace the unique one given by sKeyWord.
*/

TVeosErr replaceDataGrouple(char     *sKeyWord,
                      TPWhoInfo     pClient,
                      TPGrouple     pNewGrouple)


/***********************************************************************
takedownDataGrouple

The compliment to setupDataGrouple().  Removes the client's data grouple from
the grouplespace.
*/

TVeosErr takedownDataGrouple(char *sKeyWord, TPWhoInfo pClient)
```

```
/***************************************************************************
writeGroupleSpace

print the entire contents of the grouple space from head to tail to the given
stream
*/

TVeosErr writeGroupleSpace(FILE *stream)


/***************************************************************************
readGroupleFile

Resolve the given filename into a file stream and dispatch readGroupleStream
 */

TVeosErr readGroupleFile(char *sFileName)


/***************************************************************************
readGroupleStream

Parse the given stream of characters into grouples until EOF is read.  Insert
each grouple as it is read into the grouplespace.

The following describes acceptable syntax for the stream:

The stream is terminated by EOF.

Anything before, between, or after grouples (delimited by '[' ']') is
ignored.

Line continuations are allowed by using '\' before the newline.  Comments
within a grouple are allowed with '#'.  The rest of the line after the '#' is
ignored.  See the header for fetchGrouple() for exact syntax of a grouple.
*/

TVeosErr readGroupleStream(FILE *stream)
```

```
/****************************************************************************
*                        the primitive nancy functions
****************************************************************************/


/****************************************************************************
initGroupleSpace

A client must make exactly one successful call to initGroupleSpace before
using the nancy utilities.  Do not try to make calls to nancy if
initGroupleSpace fails.

Caller can pass the name of a text file.  The data in the file will be parsed
into grouples and placed in the grouplespace in the order they are found in
the file. This file should be in the form described in readGroupleStream()'s
header.

Pass nil for no file.
*/

TVeosErr initGroupleSpace(FILE *sFileName)


/****************************************************************************
querySpace

The grouplespace user's only mechanism for matching on grouples in the space.

sMatchWord may contain the wildcard '*'.  I.E. the word "*" matches every
grouple.

Caller can pass nil for address of ptr to grouple copy (result of query) if
they don't need a copy at present.  If the resulting grouple becomes
selected, caller can also get a copy by calling checkOutGrouple() after
querySpace().

If caller passes select flags other than those defined in nancy.h,
VEOS_Nancy_DontChangeSelect becomes the default flag; similarly, search flags
default to VEOS_Nancy_SearchFromTop; and search direction flags default to
VEOS_Nancy_SearchForward.

Caller must pass a caller ID block only when selecting the result of the
query or when searching from current selection.
*/
```

```
TVeosErr querySpace(char        *sMatchWord,
                THGrouple       hGroupleCopy,
                int             iSelectFlags,
                int             iSearchFlags,
                int             iSearchDirection,
                TPWhoInfo       pCallerID)

/** selection flags **/
VEOS_Nancy_SelectResult
VEOS_Nancy_DontChangeSelect

/** search flags **/
VEOS_Nancy_SearchFromSelect
VEOS_Nancy_SearchFromTop
VEOS_Nancy_SearchFromBottom

/** search direction flags **/
VEOS_Nancy_SearchBackward
VEOS_Nancy_SearchForward


/****************************************************************************
unselectGrouple

Release select lock on caller's selected grouple.  Once a client is done with
the 'selection' capability, its a good idea to unselect so other clients can
change or delete that grouple.  If a client needs a grouple kept under the
client's control for an extended period of time, the client should use the
checkOut mechanism.
*/

TVeosErr unselectGrouple(TPWhoInfo pCallerID);


/****************************************************************************
checkOutGrouple

Client can use checkOutGrouple() to ensure that no one will write or delete
to the caller's selected grouple until the same caller checks it in again.
Caller can pass nil for address of ptr to grouple copy if they already have a
copy.
*/

TVeosErr checkOutGrouple(TPWhoInfo pCallerID, THGrouple hDestGrouple);
```

```
/***************************************************************************
 * checkInGrouple

Counterpart to checkOutGrouple().  Use it to 'unlock' the write and delete
protection that checkOutGrouple() previously enforced.  Also, use this
function to  make changes to existing grouples.  Pass nil for new grouple if
no changes.
*/

TVeosErr checkInGrouple(TPGrouple   pOldGrouple,
                  TPGrouple   pNewGrouple,
                  TPWhoInfo   pCallerID);


/***************************************************************************
 * insertGrouple

Use this function to add new grouples to the grouplespace.  Caller can pass
nil for caller ID unless they are inserting before caller's selected grouple.
*/

TVeosErr insertGrouple(TPGrouple    pSrcGrouple,
                   int        iInsertFlags,
                   TPWhoInfo  pCallerID);

/** insert flags **/
VEOS_Nancy_InsertBeforeSelect
VEOS_Nancy_InsertAtTop
VEOS_Nancy_Append


/***************************************************************************
killGrouple

Use this function to delete grouples from the grouplespace.  Caller can
delete their selected grouple, or for bulk deletions, killGrouple() also
supports deletions from the top of the grouplespace.

In order for killGrouple to succeed, the grouple can not be checked out by
any one, and the grouple cannot be selected by anyone other than the caller.
Furthermore, if the caller has the top grouple selected, and      requests a
delete from TOP, rather than a delete SELECTED, killGrouple will not perform
the delete.  In this case, the caller must first unselect the grouple, then
delete TOP, or they can simply delete their selected grouple.

 Upon return from a successful selected-grouple delete, the preceeding
grouple becomes selected, unless the node to be deleted was the head node.
*/
```

```
TVeosErr killGrouple(TPWhoInfo pCallerID, int iDeleteFlags);

/** delete flags **/
VEOS_Nancy_DeleteFromTop
VEOS_Nancy_DeleteFromBottom
VEOS_Nancy_DeleteSelected


/**********************************************************************
                  nancy data structure utils                          *
*********************************************************************/


/**********************************************************************
newGrouple

llocate a grouple record and the word list array (not the words themselves).
Resulting ptr is nil unless complete success.
*/

TVeosErr newGrouple(THGrouple hDestGrouple)


/**********************************************************************
disposeGrouple

Deallocate a grouple.  Be sure to pass the address the ptr to the grouple so
that it can be set to nil from within disposeGrouple().  Always return
VEOS_SUCCESS.
*/

TVeosErr disposeGrouple(THGrouple hDeadGrouple)


/**********************************************************************
setGroupleWord

Used to allocate a single grouple word (newGrouple() does not do this!).

This function requires the iWords field of the dest grouple to be accurate.
This field will also be accurate upon return.
*/

TVeosErr setGroupleWord(TPGrouple   pDestGrouple,
                  int       iWhichWord,
                  char      sSrcWord)
```

```
/*************************************************************************
copyGrouple

Duplicate the given grouple into a new grouple (allocated by copyGrouple).
CopyGrouple() will not succeed unless all allocations succeed, and
hDestGrouple is set to nil when an error has occured.
*/

TVeosErr copyGrouple(TPGrouple pSrcGrouple, THGrouple hDestGrouple)


/*************************************************************************
slopDataIntoGrouple
*/

TVeosErr slopDataIntoGrouple(THGrouple hNewGrouple, int iWords, ...)


/*************************************************************************
stringToGrouple

retrieve first grouple found in given string as described in fetchGrouple().
pass back new grouple.  upon success, hNewFinger points to char position just
after endof first grouple string.

string gouples should NOT be delimited by '[' and ']' as streams should be.

 this function is the logical inverse of groupleToString().
*/

TVeosErr stringToGrouple(char *sText, THGrouple hNewGrouple)


/*************************************************************************
groupleToString

generate a single string which contains all the words from the given grouple.
the resulting string will be of the form,  [ (1st grouple word) (2nd) ... ]

this function is the logical inverse of stringToGrouple().

the string buffer must be pre-allocated by the caller.
*/

TVeosErr groupleToString(TPGrouple pTheGrouple, char *sBuffer)
```

```
/************************************************************************
newWhoParamBlock

Allocate the standard caller ID param block.
*/

TVeosErr newWhoParamBlock(THWohoInfo hDestBlock, char *sPrimName)



/************************************************************************
copyWhoParamBlock

Duplicate the given who parameter block.
*/

TVeosErr copyWhoParamBlock(TPWhoInfo pSrcBlock, THWhoInfo hDestBlock)


/************************************************************************
validateID

Compare given caller ID param blocks for equality,

returns VEOS_Nancy_Equal for equality, VEOS_Nancy_Unequal for inequuality
*/

TVeosErr validateID(TPWhoInfo pCaller, TPWhoInfo pReference)


/************************************************************************
 groupleCompare

Compare given grouple for equality

returns VEOS_Nancy_Equal for equality and VEOS_Nancy_Unequal ...
*/

TVeosErr groupleCompare(TPGrouple pLeftGrouple, TPGrouple pRightGrouple)
```

```
/************************************************************************
 printGrouple

a varaiation on VEOS_DisplayTuple() by cowboy Dan Pezely
*/

TVeosErr printGrouple(FILE *stream, TPGrouple pTheGrouple)


/************************************************************************
glomGrouples

grouple concatenate of variable number of grouples.

input form is TPGrouple pDestGrouple, int iHowMany, TPGrouple pSrcGrouple
*/

TVeosErr glomGrouples(TPGrouple pDestGrouple, int iHowMany, TPGrouple
pSrcGrouple, ...)


/************************************************************************
fetchGrouple

A variation on VEOS_GetTuple() by cowboy Dan Pezely

 fetchGrouple() returns VEOS_SUCCESS if a grouple could be read from the
stream.

Pass a ptr to a TVeosErr in pFileStatus.  Resulting value will be VEOS_EOF if
an EOF was read from the input stream.  This value is set regardless of
whether a grouple was read.

The following is the acceptable syntax for a grouple:

Groupes begin with '[' and are terminated by ']' or EOF.
Line continuations are allowed by using '\' before the newline.
Comments within a grouple are allowed with '#'.
The rest of the line after the '#' is ignored.
Words can be explicitly formed by '(' or '{' and ')' or '}' around the text,

            i.e. ( "this" is all oNe [] word ) or ()

Words not formed as above can be terminated by white space or ',' ';' '\n'.
*/

TVeosErr fetchGrouple(FILE *stream, THGrouple hNewGrouple, TVeosErr
pFileStatus)
```