NAME

MASE – Multiple Aligned Sequence Editor.

SYNOPSIS

Don Faulkner and Jerzy (Jurek) Jurka of the Molecular Biology Computer Research Resource have designed and implemented an editor for genetic sequences. *MASE* was designed to simplify the manipulation of **aligned** sequence sets. It contains functionality for many of the common tasks related to this goal. In addition, *MASE* was designed with a very modular approach, making additions or modifications to the code straightforward.

MASE was published in ‘‘Trends in Biochemical Sciences’’, vol. 13, pp. 321-322, 1988

DESCRIPTION

Several routines are used throughout *MASE.* Some familiarity with these routines should make your work more efficient.

REGULAR EXPRESSIONS

Several of the *MASE* functions use *REGULAR EXPRESSIONS.* These are ‘‘wild card’’ patterns related to search strings in editors. The regular expression handler in *MASE* is from GNU-EMACS, and is Copyright (C) 1988,
The Free Software Foundation, Inc. (This copyright applies only to the regular expression handler.) The syntax and capabilities of the regular expression handler are discussed in another section of this document.

MENU INTERFACE

MASE provides a common method input all responses. It exists in two ‘‘flavors’’ - ‘‘menu’’ and ‘‘non-menu’’.

When the ‘‘menu’’ mode is active, there is a finite set of acceptable responses. This mode may be recognized by the appearance of a reverse video ‘‘M’’ at the left of the prompt (the prompt is always terminated by an arrow (‘‘ -> ’’). To select from the menu, you may type part of the name or browse the list with the arrow keys. *MASE* uses an incremental completion method for menu selections. This works like a little bit of ‘‘ESP’’ - it figures out what you will be typing next (to resolve any unambiguous portions of the string.) When the text displayed reflects the function or value you desire, terminate your input with a carriage return. If you know some portion of the command name you desire, but not the FIRST part (incremental completion requires that you know the first part), type an asterisk, then the part of the command name you know. The first command which has that string in it will be selected. If more than one matched this wild card specification, you can go up and down the list of matches with M-p and M-n (previous and next, respectively).

When the non-menu mode is active, one may type input freely. One may retrieve previous selections by using the up and down arrows (if your arrows don’t work, contact your *MASE* administrator for help).

There are several special command keys that are available in the menu modes. These are:

Up History
    UP ARROW, C-K, or C-P
    Retrieve earlier responses

Down History
    DOWN ARROW or C-N
    Retrieve later response

Cursor Backup
    LEFT ARROW, C-B, or C-H (backspace)
    Position cursor in the current string

Cursor Forward
    RIGHT ARROW, C-F, or C-L
    Position cursor in the current string

Delete Back
    <DELETE>
    Delete character behind cursor

Erase Entry
    C-U
    Erase the entire line

Start of Entry
    C-A
    Position cursor at the beginning of the entry

End of Entry
    C-E
    Position cursor at the end of the entry

Expand Aliases
    M-a
    Expand the aliases in the current string. *MASE* currently has no method to introduce items into this alias list, however!

Go to previous wild-card match
    M-p

Go to next wild-card match
    M-n

Expand File Names
    M-f
    Expand the current selection as file names. These file names will be appended to the history list.

    For many purposes, one may want to expand an ambiguous file specification to a list of files. This subroutine performs this function. The expansion is executed using the GNU-EMACS regular expression handler (regex-gnu(3)) to test for matches, modified to be more compatible with the name expansion done within the shells (sh(1) and csh(1)).

    The file name is first parsed into components using slashes (‘‘/’’) as a delimiter. The wild card aspects apply only to the components - no wild card component may cross the boundary of a slash. If the file name does not specify an absolute path (i.e. it does not begin with a period (‘‘.’’) or a slash (‘‘/’’)), then each of the components of your **‘‘DATA’’** environmental variable will be prepended to the filename in turn before the expansions are attempted. The **‘‘DATA’’** environment may be set via a command similar to ‘‘DATA=:/seq/nbrf-12:/seq/gb-52.nih:.:; export DATA" from the Bourne shell (sh(1)), or ‘‘setenv DATA /seq/nbrf-12:/seq/gb-52.nih:.:" from the C shell (csh(1)). As you can see, the directories to be searched are separated by colons (‘‘:’’). The elements of the **‘‘DATA’’** environment will *NOT* be expanded (i.e. they may not contain wild cards).

    If the file name has an at sign (‘‘@’’) as it’s first character, the file will be regarded as an indirect file - a list of other files. Each element in this indirect list will be passed recursively through this mechanism. Thus, the indirect elements may refer to other indirects, and may have wild cards. Wild cards allowed are:

‘‘*’’ Matches any string, including the null string.

‘‘?’’ Matches any single character.

‘‘[...]’’ Matches any one of the characters enclosed. A pair of characters separated by ‘‘-’’ matches any character lexically between the pair.

If the current text entry was ‘‘*.seq’’ and you were to hit M-f, all files that matched ‘‘*.seq’’ in all directories specified by your **‘‘DATA’’** environment would be added to the history list, which you could then select among via cursor keys.

Help

C-? (mapped for (‘‘?’’))
Display short help message regarding the current menu item.

Apropos

M-?
Run *APROPOS* to find about all things (commands, variables . . .) that have a specific key string

Edit Help

M-e
Edit the file containing the short help message associated with this menu item. Edit the file using the editor specified in your **‘‘EDITOR’’** environment, or vi(1). This command is reserved for the use of the *MASE* administrator.

VARIABLES

The main editing mode of *MASE* references the *INTERNAL VARIABLES ‘‘COLUMN-HIGHLIGHT’’*, *‘‘DISPLAY-POSITION’’*, *‘‘HIGHLIGHT-DIFFERENCES’’*, and *‘‘LOCK-WINDOWS’’*. They are used as follows:

COLUMN-HIGHLIGHT

If set to some non-zero integer, every *COLUMN-HIGHLIGHT*’th column will be displayed in reverse video. This is to serve as a rule. For example, if you set *COLUMN-HIGHLIGHT* to ‘‘5’’, then columns numbered 5, 10, 15, 20, 25 ... would be highlighted.

DISPLAY-POSITION

When set to non-zero, the position of the cursor in the current sequence will be printed at the bottom of the screen, both in ‘‘real’’ column numbers and in ‘‘effective’’ column numbers (ignoring gaps).

HIGHLIGHT-DIFFERENCES

Highlight letters that differ from this specified reference sequence. Sequence ‘‘0’’ refers to the consensus sequence.

LOCK-WINDOWS

When set, this will cause the two windows to shift in complete synchrony. This mode may not be fully functional.

STRING CONVERSION

The routines *OUTPUT* (for output mapping strings), *MAP* (only from within *TAKE* files), and *BIND* (again, only from *TAKE* files) require one to be able to specify strings containing control and other weird characters. *STRING CONVERSION* is used to convert from printable, editable strings into these control-laden strings. The strings are processed as follows. There are several backslash escapes. These were inspired by the ‘‘C’’ programming language. They are:

\n a new line (hex 0xa, decimal 10, octal 012)

\b a back space (hex 0x8, decimal 8, octal 010)

\f a form feed (hex 0xc, decimal 12, octal 014)

\e an escape (hex 0x1b, decimal 27, octal 033)

\s a space (hex 0x20, decimal 32, octal 040) (this is useful to prevent a string break)

\M Meta next – the high bit of the next character will be set

\Z Control Meta next – the high bit of the next character will be set, and the fifth and sixth bits will be cleared.

\### Enter a number in octal. Up to three digits will be ‘‘eaten’’ - it is easiest and least confusing if you always use exactly three digits, as in ‘‘\002’’ for octal 2.

One may prefix a character with a caret (‘‘ˆ’’) to controlize it (clear the fifth and sixth bits). Thus, could be used to specify a control-A. Note that there is overlap: **‘‘ˆJ’’, ‘‘ˆj’’, ‘‘\n’’,** and **‘‘\012’’** are all exactly equivalent.

Also note that if one desires to use these character conversions to program keys for *MAP* and *BIND,* one can call these routines from *INTERPRET-LINE;* the strings will be passed through *STRING-CONVERSION* before being handed to *BIND* and *MAP.*

Change Bars

Throughout the manual, you will notice some characters in the right margin. These are **change bars.** They serve to indicate sections that have changed since the last release of the manual. A plus sign (‘‘+’’) indicates lines that have been added. A minus sign (‘‘-’’) indicates where lines have been deleted. A vertical bar (‘‘|’’) indicates lines that have been modified. Thus, one who is familiar with a version of *MASE* should have fewer problems with the manual as new releases are available.

Anchored positions

There are provisions for setting *Anchor Points* within sequences, represented by periods (‘‘.’’). +
This permits one to edit the alignment within a specific domain without affecting the align- +
ment in subsequent domains. This is implemented through several routines: *Anchor-Check,* +
Anchor-Create, Anchor-Synchronize, DEL-Compensated-Backward, DEL-Compensated- +
Forward, INS-Compensated-Backward, and *INS-Compensated-Forward.* These functions were +
conceived by Drs. Jerzy Jurka and Ela Holsztynska. These functions (mainly *Anchor-* +
Synchronize) may be useful for automatic generation of alignments. +

Locus name display

The locus names are displayed on the left side of the screen. The number associated with +
each is the ‘‘locus number’’ associated with that sequnces. These numbers reflect the locus’s +
current position; they can be changed by several functions. If the locus has been modified, +
the name will be displayed in reverse video (see *Saving your work).* If the first column (the +
character at the left-hand margin) is an asterisk (‘‘*’’), it means that that the last *PATTERN-* +
HIGHLIGHT matched somewhere in that sequence. This provides an at-a-glance indicator of +
which sequences matched a given pattern. +

EXAMPLES

FILES

SEE ALSO

INTRINSIC FUNCTIONS

For a discussion of the functions within *MASE.*

INTERNAL VARIABLES

For a discussion of variables used by *MASE* and its functions to modify their behavior.

STARTUP FILES
For a discussion on customizing *MASE* behavior.

FILE FORMATS
For a discussion of input file formats required by various functions within *MASE.*

BUGS

MASE should be considered an application in development; there may be random glitches throughout; proceed with caution.

Since *MASE* is modular, and far from mature, any suggestions would be most appreciated (please use the *GRIPE* function within *MASE* to register your comments.

Since *MASE* is a complex program, and is evolving, certain elements of the documentation may easily become out of date; the printed documentation may be out of date compared to the functions and the on-line documentation (accessed via help and help within menus).

MASE tries to handle window size changes. Sometimes, it works. Often, the display will be ‘‘not quite right’’. Occasionally, *MASE* will crash and core dump. The solution? For now, resize the window **before** you start *MASE.* Eventually, there will be a proper solution for this problem...