

LINEUP

FUNCTION

LineUp is a screen editor for editing multiple sequence alignments. You can edit up to 30 sequences simultaneously. New sequences can be typed in by hand or added from existing sequence files. A consensus sequence identifies places where the sequences are in conflict.

DESCRIPTION

LineUp lets you edit several overlapping or aligned sequences simultaneously. LineUp allows you to edit sequences in the context of an alignment to help you see the effect of your changes on the alignment.

As in SeqEd, you can move the cursor with the arrow keys and insert or delete symbols or gaps in the sequences. In LineUp, the cursor can travel from one sequence to another. You can add new sequences by hand or from existing sequence files, and you can move sequences from one position to another.

LineUp provides a surface on which you can arrange and edit many sequences. This surface resembles a piece of graph paper with 31 rows and as many columns as you need. The screen acts as a window behind which the LineUp surface is scrolled.

Sequences can be placed anywhere on the surface as long as two sequences in the same row do not collide. Several sequences can be placed on the same row.

Sequences placed on the LineUp surface become part of a sequence group. A new sequence group is formed by running LineUp with a new sequence group name. Sequences already stored in files can be placed anywhere on the surface with the `Get` command. New sequences (not already in sequence files) can be typed in anywhere on the surface.

When you end a session with LineUp, it writes out each sequence in a file and then writes a list file with the name and position of each sequence in the group. (See Chapter 2, Using Sequence Files and Databases in the User's Guide for more information about list files.) When you edit the group again, the sequences reappear on the LineUp surface where you left them.

You can have a consensus sequence display the dominant character at each column where sequences overlap. The consensus uses uppercase where overlapping sequences are in agreement, lowercase to show disagreement, and periods to show where there is no consensus at all.

EXAMPLE

Here is a session using LineUp to edit the same sequence group displayed in the example session for the Pretty program. First use `Fetch` to copy the files `*.frg` and `picorna.fil` to your default directory.

LineUp

```
% lineup picorna
```

```
R2          Column: 1  Row: 5      No AutoCons  FOSN:  PICORNA  Protein

15: .....ttttgesad.pvtttve....n..yggdt.q...vq
14: .....ttatgesad.pvtttve....n..ygget.q...vq
13: .....ttsagesad.pvtttve....n..ygget.q...iq
12: gvenae.kgvtentna.tadfvapvylpe.nqt.....kv.affynrs...spi.gaftvks.....
11: glgqmlsmi.dntvretvgaatsrdalpn teasgpthskeipaltavetgatnplvsdtvqtrhvq
10: glgqmlsmi.dntvretvgaatsrdalpn teasgpahskeipaltavetgatnplvsdtvqtrhvq
 9: gigdmiegav.egitknalvpptstnslpghkpsgpahskeipaltavetgatnplvsdtvqtrhviq
 8: giedliseva.qgal..tislpkqqdslpdkasgpahskevpaltavetgatnplapsdtvqtrhvq
 7: ..gpvedai.....t..aaigr..vadtvgtgptnseaipaltaaetghtsqvvpqdtmqtrhvkn
 6: glgdeleevivekt.kqtv.asi.....ssgpkhtqkvpiltanetgatmpvlpsdsietrttym
 5: ..npvenyidevlnevlv.....vpinssnpttsnsapaldaaetghtssvqpedvietryvqt

..|.....|.....|.....|.....|.....|.....|.....
 0          10         20         30         40         50         60
```

```
"picorna.fil" successfully loaded.
```

RELATED PROGRAMS

SeqLab is the graphical user interface for the Wisconsin Package. The SeqLab Editor is a powerful and versatile tool for creating, editing and displaying multiple sequence alignments.

SeqEd is an interactive editor for creating and editing individual sequence files.

PileUp creates a multiple sequence alignment from a group of sequences.

Gap uses the algorithm of Needleman and Wunsch to find the alignment of two complete sequences that maximizes the number of matches and minimizes the number of gaps. If you run Gap with the command-line parameters for sequence output, it writes sequence files with the sequences expanded by the addition of gaps.

Pretty displays multiple sequence alignments and calculates a consensus sequence. It does not create the alignment; it simply displays it. Unlike LineUp, Pretty requires that all your sequences start in the same column.

PlotSimilarity plots the running average of the similarity among the sequences in a multiple sequence alignment.

ProfileMake makes a profile from the multiple sequence alignment. ProfileSearch uses the profile to search a database for new sequences with similarity to the group of aligned sequences. ProfileSegments displays optimal alignments between each sequence in the ProfileSearch output list and the group of aligned sequences (represented by the profile consensus). ProfileGap makes optimal alignments between a sequence and a group of aligned sequences represented as a profile.

STARTING OUT

To create a new sequence group, use the LineUp command with a new group name such as `myseqs`. If you use `% lineup myseqs`, LineUp looks in your current directory for the file `myseqs.fil`. If you use the command `% lineup -MSF myseqs`, then LineUp looks for the file `myseqs.msf`. If it doesn't find a file with this name, LineUp starts a new group with one sequence, the consensus, having the same name as the group. (If you do not want to have a consensus sequence in your group, run LineUp with the command-line parameter `-NOCONsensus`.) To construct the group, use the `Get` command to add sequences from existing sequence files or use the `New` command so LineUp lets you type in a new sequence. LineUp prompts you for a unique name of up to ten characters for each new sequence.

EDITING EXISTING GROUPS

You can start LineUp with the name of an existing group. If you have a file of sequence names called `myseqs.fil`, which was created in a previous session with LineUp, use `% lineup myseqs`. If you have a multiple sequence format (MSF) file called `myseqs.msf`, which was created in a previous session with LineUp, use `% lineup -MSF myseqs`. You may specify a file name extension if the default extension LineUp adds is not appropriate. You also can use any single or multiple sequence specification as input to LineUp. Multiple sequences can be specified as a list file or as a sequence specification using a wildcard. (See Chapter 2, *Using Sequence Files and Databases in the User's Guide* for help in specifying sequences.) LineUp loads the sequences into the multiple sequence editor and starts with its window at the left end of the group. You can add more sequences, modify existing ones, delete sequences, rename sequences, and move any sequence to a new position.

SCREEN MODE

In Screen Mode, commands are typically single keystrokes. Except for the search command, Screen Mode commands do not require a `<Return>`.

Entering Sequence Characters

In Screen Mode, the cursor shows your position in one of the sequences in the group. You can insert any valid GCG sequence symbol (see Appendix III) into the sequence by typing the symbol. It is inserted at the cursor.

Deleting Sequence Characters

The `<Delete>` key and `<Ctrl>H` delete the symbols to the left of the cursor, one by one. The remainder of the sequence slides over to fill the gap.

Moving the Cursor Horizontally

To move the cursor to the right one symbol, use the `<Right-arrow>` key; to move to the left, use the `<Left-arrow>` key. Moving the cursor past the end of the sequence moves it to the next sequence on the row.

You can type a number followed by a `<Return>` and the cursor moves to that position in the current row. If you specify a position that is not occupied by a character, the cursor moves to the nearest occupied position.

The `<Left-arrow>` and `<Right-arrow>` keys can be preceded by a number, telling how many symbols to move to the left or right. For example, `10<Right-arrow>` moves ten symbols to the right.

LineUp

You can use the angle brackets to skip 50 characters to the left or right. If you precede the angle bracket by a number, the cursor skips that many characters and continues to do so until you change the number.

Moving the Cursor Vertically

The <Up-arrow> and <Down-arrow> keys move the cursor up or down to the next row.

In contrast with the horizontal arrows, if you precede either the <Up-arrow> or the <Down-arrow> with a number, the cursor moves to the row with that number. For example, 10<Up-arrow> moves to row ten, not up ten rows.

Moving a Whole Sequence

When the cursor is at the left end of a sequence, you can move the sequence to the right with the space bar and to the left with the <Delete> key or <Ctrl>H. If you want to move a sequence to another row or to the other side of a sequence on the same row, you must use the `MOVE` command in Command Mode.

Finding Patterns

To search for a pattern, type a / (slash) in Screen Mode. You are prompted for the sequence pattern you wish to find. LineUp only searches the current sequence. You can repeat the last search by simply using /<Return>.

The command-line parameter `-NUCleotide` or the `NUCleotide` command in Command Mode makes LineUp treat all nucleic acid sequences as circular and finds your pattern even if it wraps from the end of the sequence into the beginning. LineUp uses the same rules for pattern definition and recognition as the `FindPatterns`, `MapPlot`, `Map`, and `MapSort` programs.

The command-line parameter `-PROtein` or the `PROtein` command in Command Mode makes LineUp search linear and disables the nucleic acid ambiguity meanings of the GCG sequence symbols; they also change the way the consensus sequence is defined (see the topic `THE CONSENSUS SEQUENCE` below).

Even if LineUp thinks your sequence is a nucleotide sequence, you can request a perfect match by typing an = right after the /. So if you type /=RTC only RTC is matched, whether you have a protein or a nucleotide sequence.

Leaving Screen Mode

Use <Ctrl>D to leave Screen Mode and enter Command Mode.

Screen Mode Summary

Here is the summary of Screen Mode commands you would see in the on-line help:

Screen Mode

[n] is an optional numeric parameter.

G, A, T, C	- inserts a sequence character
<Delete>	- deletes a sequence character, "drags" a sequence to the left if cursor is at its start

<Ctrl>H	- deletes a sequence character, "drags" a sequence to the left if cursor is at its start
<Space bar>	- "pushes" a sequence to the right if cursor is at its start
/TAACG<Return>	- finds the next occurrence of "TAACG", last pattern is the default when none is specified
[n]<Right-arrow>	- move ahead [n characters]
[n]<Left-arrow>	- move back [n characters]
[n]<Up-arrow>	- move up to next sequence [or to row specified]
[n]<Down-arrow>	- move down to next sequence [or to row specified]
[n]<Return>	- move to column n
1<Return>	- move to start of current sequence
<Ctrl>E	- move to end of current sequence
<Ctrl>R	- redraw the screen
<Ctrl>Z	- enter Command Mode
<Ctrl>I	- push over all seqs starting past current column
<Ctrl>D	- pull over all seqs starting past current column
[n]<	- move 50 [or n] positions to left
[n]>	- move 50 [or n] positions to right

COMMAND MODE

In Command Mode, you enter commands followed by a <Return>.

Entering Command Mode

Use <Ctrl>D to leave Screen Mode and enter Command Mode.

Editing LineUp Commands

LineUp command editing is modeled on OpenVMS DCL command line editing. The <Left-arrow> and <Right-arrow> keys let you move your cursor around in a command that you have typed so you can insert or delete characters at any position. <Ctrl>E moves the cursor to the end of the line. <Ctrl>U deletes all the characters from the current cursor position to the start of the line.

Editing Previous LineUp Commands

LineUp lets you modify and execute previous commands. The <Up-arrow> key displays previous commands.

Returning to the Screen Mode

If you simply press <Return>, LineUp returns to Screen Mode described above. If you have **-SINGLE**command on the command line or in your command-line initializing file, LineUp returns to Screen Mode immediately after executing each command.

Commands May Be Truncated

Only the capitalized portion of the commands described in the documentation below must be typed.

LineUp

Parameters Are Used With Some Commands

Some commands may be preceded or followed by optional numeric parameters or a file name. The square brackets ([and]) in the documentation below show optional command arguments: *s* and *f* refer to starting and finishing rows or offsets on the surface; *x* and *y* refer to offset and row coordinates. When an optional parameter is omitted, some commands prompt you for the value. Others commands make default assumptions that are explained in each command description.

Missing Position Parameters: Spacewalk

Several commands need position parameters to know where to put a sequence. If these parameters are omitted, LineUp enters a mode, called *Spacewalk*, that allows you to move the cursor anywhere on the surface to select a position for the new sequence. In Spacewalk Mode, the arrow keys and <Return> can be preceded with numbers as in Screen Mode. <Ctrl>D cancels the command when you are in Spacewalk Mode. If you prefer to provide numeric coordinates rather than position the cursor, you can eliminate Spacewalk Mode by using the command-line parameter `-NOSPACewalk` or the command `NOSPacewalk`. You are then prompted for numeric coordinates if you omit them from commands.

Other Missing Parameters

If a required name is omitted or illegal, LineUp prompts you for a name. If you respond with a blank name, LineUp cancels the command.

Working Directory

You must have write privileges in your current working directory to use LineUp; otherwise, LineUp will not accept any name you try to give a sequence.

Default Values for LineUp Prompts

Often when it prompts for sequence or file names, LineUp presents a default value in a manner different from other GCG programs; when the prompt appears, it looks like you have already typed in the default value. You can just press <Return> if you want to accept the default. If you want to change it, proceed as with command-line editing. Make small changes by using the arrow keys to move within the offered response or delete the response with <Ctrl>U and type your desired response.

Command Mode Summary

Here is the summary of Command Mode commands you would see in the on-line help:

Command Mode

x and y represent numbers for column and row.

Only the capitalized part of the command is necessary.

[x,y] Get [filename]	- add sequence [at position x,y] [from filename]
[x,y] New [seqname]	- add empty sequence [at position x,y] [named seqname]
[x,y] MOve [seqname]	- move current or specified sequence [to x,y]
REMove [seqname]	- delete current or specified sequence entirely
REName [old] [new]	- change sequence name (changing consensus name changes the group)
REDraw	- redraw the screen
HEAding [seqname]	- edit documentary heading of current or specified sequence
screen	- enter screen mode (pressing <Return> is sufficient)
NUCleotide	- use nucleotide ambiguity codes in find and consensus
PROtein	- do not use nucleotide ambiguity codes
SPacewalk	- use spacewalk to position sequences
NOSpacewalk	- DO NOT use spacewalk to position sequences
FOSN	- use list file format when writing
MSF	- use multiple sequence format files when writing
[n] SLide	- add n to all sequence columns
[s,f] ROWMove [n]	- move a set of rows (s to f) up or down [n rows]
[s,f] PRint [filename]	- write the sequence group to a Pretty format file
SUMmary [filename]	- write the sequence names and positions in a file or on the terminal screen
GOto [seqname]	- put cursor on start of named sequence
[s,f] CONSensus	- calculate consensus [from s to f]
AUTOconsensus	- automatically calculate consensus (slow)
NOAUTOconsensus	- turn off automatic consensus
FLip	- reverse complement the current group
ZIp [filename]	- align and gap a sequence to the current group
Write [filename]	- write the current sequence group to a file
EXit [filename]	- write the current group to a file and stop
Quit	- quit the editor without writing out the group

COMMAND DESCRIPTIONS

: [x,y] **Get** [filename]

adds the sequence in the specified file to the group at column x in row y. The screen is erased and you are prompted to enter the range and strand. Unlike the **Write** and **Exit** commands, **Get** does not assume any file extension. You must type the file name plus any extension it requires.

LineUp

`:[x,y] New [SeqName]`

adds an empty sequence at column `x` in row `y`.

`:[x,y] Move [SeqName]`

moves the sequence to start at column `x` in row `y`. If the `SeqName` parameter is omitted, the sequence at the current cursor position is moved.

`:REMove [SeqName]`

deletes the entire sequence from the group. If the `SeqName` parameter is omitted, the sequence at the current cursor position is removed.

`:REName [OldName] [NewName]`

changes the name of the sequence. If no names are provided in the command, the sequence at the current cursor position is renamed and you are prompted for the new name. If only one name is provided, it is assumed to be the old name and you are prompted for the new name.

`:REDraw`

redraws your terminal screen. This is useful if noise in the line between your terminal and the computer has changed the screen in some unreasonable way or if a system message appears on your screen.

`:[s] HEAding [SeqName]`

enters Heading Mode to let you view and edit the documentary heading. You can modify any part of the heading. Heading Mode is terminated with `<Ctrl>D`. The optional `SeqName` parameter specifies which sequence heading you want to edit. If omitted, the sequence at the current cursor position is assumed. The optional numeric parameter specifies which line of the heading you want to start editing.

`:screen`

returns your session to Screen Mode. Just pressing `<Return>` also returns you to Screen Mode. (If you prefer to return to Screen Mode after every command is executed, use the command-line parameter `-SINGLEcommand`.)

`:NUCleotide`

sets the sequence type for each sequence in the sequence group to be nucleotide. This enables the nucleic acid ambiguity meanings of the GCG sequence symbols in pattern searches (with `/`) and consensus definition (set the topic `THE CONSENSUS SEQUENCE` below). Also, LineUp treats nucleic acid sequences as circular when searching for a pattern. When the sequences are saved to files with either the `Write` or `EXit` command, they are written as nucleotide sequences if their sequence type is nucleotide.

:**PROtein**

sets the sequence type for each sequence in the sequence group to be protein. This forces LineUp to treat all sequences as linear in pattern searches and does not interpret any sequence characters as nucleotide ambiguity symbols in pattern searches and consensus definition (see the topic **THE CONSENSUS SEQUENCE** below). When the sequences are saved to files with either the **Write** or **EXit** command, they are written as protein sequences if their sequence type is protein.

:**SPacewalk**

enters Spacewalk Mode, which allows you to move the cursor anywhere on the surface to select a position for a new sequence.

:**NOS**Spacewalk

tells LineUp *not* to use Spacewalk Mode but to prompt for numerical surface coordinates.

:**FOSN**

tells LineUp to use the list file format when loading or storing the sequence group.

:**MSF**

tells LineUp to use the multiple sequence format (MSF) file when loading or storing the sequence group.

:**[n] SLide**

shifts all the sequence starting positions by *n*. The coordinate ruler appears to slide under the sequences. *n* can be either a positive or negative number to shift the sequences to the right or left, respectively.

:**[s,f] ROW**Move [*n*]

moves a clump of rows up or down. The sequences on rows numbered from *s* to *f* are moved up *n* rows. Negative values of *n* move the sequences down *n* rows. This command can be used to open a row in the middle of the surface for another sequence. LineUp will not let you move sequences onto rows containing other sequences not simultaneously being moved.

:**[s,f] PR**int [*filename*]

writes a file of the formatted sequence group from position *s* to *f*. The format resembles that of **Pretty**.

:**SUM**mary [*filename*]

writes a list of the names and beginning positions of the sequences loaded into the LineUp editor. This list can go either to a file or to your screen (by typing **Term** for filename).

LineUp

:Goto [SeqName]

moves the cursor to the beginning of the named sequence.

:[s,f] CONsensus

calculates the consensus sequence between positions *s* and *f*. If the optional positions are omitted, the entire consensus is calculated. This command only works when LineUp is not in the Auto Consensus state. (See the topic THE CONSENSUS SEQUENCE below for further details.)

:AUtoconsensus

makes LineUp recalculate the consensus sequence each time there is a change in any of the other sequences. When LineUp is in the Auto Consensus state, the consensus is strictly a function of the other sequences and cannot be changed directly. When the sequence group is large, recomputing the consensus uses a lot of machine time and makes LineUp appear sluggish.

:NOAUtoconsensus

turns off the Auto Consensus state. This allows you to change the consensus directly.

:FLip

makes LineUp reverse and complement all of the nucleic acid sequences in the current group.

:ZIp [filename]

aligns a new sequence to the current consensus.

:Write [filename]

records the current surface configuration in a list file and saves the current version of each sequence in a file if the program is in FOSN mode (see the FILE NAME CONVENTIONS topic below). If the program is in MSF mode, a multiple sequence format (MSF) file is written. If the filename parameter is omitted, LineUp uses the sequence group name specified when the program is initially run. If you specify a file in another directory, all files are created there.

:EXit [filename]

works like the `write` command but stops the session after writing out the sequences. The filename parameter behaves as in the `write` command.

:Quit

terminates a session with LineUp without saving any changes you've made since the last time you used the `write` command.

:Help

shows the commands available in Screen and Command Modes of LineUp.

HEADING MODE

Heading Mode allows you to view and edit the documentation that precedes the sequence in the sequence file. All headings are lost if you write the sequences into a multiple sequence format (MSF) file.

Entering Heading Mode

To enter Heading Mode, use the **HEAding** command.

Leaving Heading Mode

Use <Ctrl>D to return to Command Mode.

Moving the Cursor

You can move around using the arrow keys. Although the editing window is only twenty lines long, it scrolls over the heading vertically to let you see and modify any part. <Ctrl>E positions the cursor at the end of the current line.

Editing in Heading Mode

Like many text editors, typing inserts text at the cursor and the <Delete> key and <Ctrl>H delete characters to the left of the cursor. <Ctrl>U deletes everything from the current cursor position to the start of the line. Pressing <Return> creates a new line starting at the current position in the heading.

Unlike many text editors, LineUp asks you if you need more storage before letting you edit the Heading. You must enter the maximum number of lines that you expect to have to add. If you are in Heading Mode and find you do not have enough storage for your changes and additions, you can exit Heading Mode and enter it again, specifying some larger number of lines for increased storage.

PROTEIN AND NUCLEOTIDE SEQUENCE GROUPS

LineUp behaves differently depending on whether you are working with a protein or nucleotide sequence group.

If you are working with a nucleotide sequence group, then pattern searches (see "Finding Patterns" under the SCREEN MODE topic) and the consensus definition (see the topic THE CONSENSUS SEQUENCE) assume the IUB nucleotide ambiguity meanings for the GCG sequence symbols. Also, LineUp treats nucleic acid sequences as circular when searching for patterns. When the sequences are saved to files with either the **Write** or **Exit** command, they are written as nucleotide sequences if their sequence type is nucleotide.

If you are working with a protein sequence group, then LineUp treats all sequences as linear in pattern searches and does not interpret any sequence characters as nucleotide ambiguity symbols in pattern searches and the consensus definition. When the sequences are saved to files with either the **Write** or **Exit** command, they are written as protein sequences if their sequence type is protein.

LineUp

By default, if the first sequence entered into the LineUp editor screen is from an existing sequence file, then the type of that sequence determines the type for the entire group. If the first sequence in a sequence group is entered interactively from the keyboard, then LineUp sets the sequence type for the entire sequence group to be protein, by default. LineUp indicates the type of the sequence group (protein or nucleotide) in the upper-right corner of the editor screen.

You can specify the sequence type for the entire group from the command line with the `-PROtein` and `-NUCleotide` command-line parameters. Once you are viewing the editor screen, you can change the sequence type for the entire sequence group with the `PROtein` and `NUCleotide` command mode commands.

THE CONSENSUS SEQUENCE

An optional consensus sequence can be generated as a function of the rest of the sequences in a sequence group, or, like any other sequence, typed in by you.

By default, new sequence groups contain an empty consensus at row 0 unless LineUp is run with the `-NOCONsensus` command-line parameter.

If the sequence group has no consensus, you can create one using the `New` command and giving the new sequence the same name as the sequence group. The `CONsensus` command and the `AUTOconsensus` commands now work on the row you have designated for the consensus with the `New` command.

When a consensus sequence is generated by LineUp, either by issuing the `CONsensus` command or `AUTOconsensus` command, each consensus character in the consensus sequence is replaced with a character that is a function of the other characters in its column. If all the characters in the column are the same letter and at least one of them is uppercase, the consensus character is the uppercase equivalent of that letter. If there is more than one letter in the column, but one occurs more frequently than any other, or if all letters are the same, but none are uppercase, then the consensus character is the lowercase of that letter. Otherwise, the consensus character is a dot (.).

The consensus definition also depends on whether LineUp is working with a nucleotide or protein sequence group. If a protein sequence group is loaded into the multiple sequence editor, the above description is complete. If a nucleotide sequence group is loaded into the editor, the ambiguity codes are ignored for the purpose of consensus definition. This treats all ambiguity codes as though they were the code 'N.' LineUp indicates the type of the sequence group (protein or nucleotide) in the upper-right corner of the editor screen.

The consensus sequence is distinguished by having the same name as the sequence group. If you rename the consensus sequence with the `REName` command, the name of the sequence group changes as well. (You can rename the group even if you have no consensus sequence.) Conversely, if you specify a file name in a `Write` or `EXit` command, this changes the name for the sequence group being saved and also changes the consensus sequence name.

The consensus sequence is unique in that, because it will likely extend to all columns determined by the other sequences, no other sequence may share its row. You can delete the consensus sequence from your group, and you can later create a new consensus sequence. However, an existing sequence cannot become the consensus sequence, either through the `REName` or `Get` commands.

PULL-OVER AND PUSH-OVER

If you use LineUp with sequence groups that have different sequences starting at several different columns, a problem will arise when you make deletions or insertions to whole columns. For example, suppose you are assembling a group of sequences with LineUp. When a new sequence is added, you may decide a previous sequence reading was incorrect. You may decide to delete a base from an old sequence near the left end of the assembly. The tail of that sequence slides left one column, destroying its alignment with any sequence starting to the right of the deletion site.

The general problem is that insertions or deletions cause shifts of register between sequences. Those sequences that overlap with the changed column appear to need adjustment. But those sequences that start down to the right do not appear misaligned.

LineUp warns you of potential alignment problems by producing a warning sign at the top of the screen. The sign says either `PUSH-OVER WARNING` or `PULL-OVER WARNING`, depending on whether there was an insertion or deletion. The warning is only displayed if there are other sequences that start to the right of your change. To make it easy for you to correct the alignment problem, LineUp provides you with screen mode commands to *PULLOVER* the sequences starting to the right of the cursor (`<Ctrl>P` for deletion), or to *PUSHOVER* (`<Ctrl>I` for insertion). You must make the decision whether the deletion or insertion requires adjustments and then ensure that the adjustments are correct. It is not recommended that you blindly trust the warning sign but that you let it remind you of the issue.

MULTIPLE SEQUENCE FORMAT (MSF) FILES

By default, LineUp reads and writes individual sequence files, grouped in a list file (FOSN format). Using the command-line parameter `-MSF` causes LineUp to expect a multiple sequence format (MSF) file when reading a sequence group, and to write out an MSF file when storing a sequence group (MSF format). For instance, the command `% lineup -MSF hsp70` reads the sequences in the file `hsp70.msf` into the LineUp editor and names the sequence group `hsp70`. (See Chapter 2, Using Sequence Files and Databases in the User's Guide for a complete description of MSF files.) When LineUp writes an MSF file, leading gap characters (.) are added to those sequences that do not start at the beginning of the alignment so that all sequences are left-justified in the output file.

The current sequence group format is indicated as either `FOSN:` or `MSF:` on the top line of the screen editor. You can toggle between these two formats using the `FOSN` and `MSF` commands in command mode.

EDITING INDIVIDUAL SEQUENCE FILES

There is no harm in using SeqEd to change a sequence file that has been written by LineUp. Provided the name is the same, the new version is accepted by LineUp. The only restriction on replacing members of a sequence group is that the new members must not overlap with other sequences on the same row. The information where the sequence starts is stored in the list file, so changing the sequence file can only change the length of the sequence. You can change where a sequence starts on the surface by modifying the Offset and Row columns of its entry in the list file using a text editor. If you overlap two sequences on the same row, LineUp refuses to load one of the overlapping sequences.

EMBEDDED COMMENTS

LineUp does not handle embedded comments. LineUp can read files containing embedded comments, but the comments are lost and will not appear in any file written by LineUp.

LineUp

LINEUP AND PRETTY

If your sequences all start at the same column, you can use Pretty to generate a consensus sequence for a sequence group created by LineUp. Pretty uses a more sophisticated algorithm than LineUp to generate a consensus sequence and you have more control over the consensus calculation. However, Pretty can only handle sequence groups whose left ends are aligned.

Pretty and LineUp both know how to read the other's files of sequence names, so you can use Pretty to get a consensus sequence in Pretty format. Then, `% pretty -UGLY` makes a file of sequence names that LineUp can read. However, the consensus sequence defined by Pretty will not be recognized as the consensus sequence of LineUp. It is named Consensus by Pretty, whereas LineUp names its consensus sequence with the sequence group name. This is reasonable, since LineUp will not define the consensus in the same way, so the names should be different.

If you alternate between using Pretty and LineUp on a sequence group having a LineUp consensus sequence, you have to preserve the old sequence group name when doing `Pretty -UGLY` in order to make LineUp recognize the consensus sequence. If you give a new name to the group, the consensus sequence is no longer recognizable, as such, by LineUp.

THE LINEUP DISPLAY

Several indicators for LineUp are displayed on the top row of the screen. The left-most word indicates the name of the sequence on which the cursor currently rests. Next, the cursor's position on the surface is displayed. Then the display shows whether LineUp calculates the consensus automatically every time you add or delete a character. The sequence group name is next, preceded by either FOSN or MSF, indicating the file format to be used for reading and writing the sequence group. Finally, LineUp indicates whether the type of the sequence group is nucleotide or protein.

LineUp frequently displays the `PULL-OVER WARNING` sign (see the `PULL-OVER AND PUSH-OVER` topic above).

The screen provides a window onto the sequence surface. Through this window, 16 of the 31 surface rows can be viewed at one time. As you move your cursor near the top row of the window, for example, if there are occupied surface rows past the top of the window the surface is scrolled down, letting you see more lines at the top of the window and fewer at the bottom.

When there are more rows in use than can be displayed at once, some rows are hidden above or below the window. When this happens a '+' is displayed next to the top or bottom row number indicating hidden rows in that direction.

Although the window also scrolls horizontally, there is no analogous sign indicating that you cannot see the whole length of the surface.

FILE NAME CONVENTIONS

When you save a sequence group using FOSN format, the name given to the FOSN is made up of the sequence group name followed by the extension `.fil`. The sequence file names are the sequence names used in LineUp and the file extension `.frg`. When you save a sequence group using MSF format, the name given to the MSF file is made up of the sequence group name followed by the extension `.msf`.

These file name extensions are the defaults for LineUp, but you can specify your own by using the command-line parameters `-FOSNExtension`, `-FRAGExtension`, and `-MSFExtension` (see below). You can override these choices when you specify an output file name; if you include a file extension, it is used in lieu of that given on the command line or the default.

SYSTEM CRASH OR HANGUP

The current version of LineUp cannot recover from a system crash. If you are disconnected from LineUp, you lose *everything* you have done since the last time you saved the group using the `Write` or `EXit` commands. Therefore, we recommend that you save your work frequently using the `Write` command so that little is lost in the event of a crash.

RESTRICTIONS

ACKNOWLEDGEMENTS

LineUp was designed and implemented by Dr. William Winsborough. We are very grateful for the collaboration of Drs. William Boorstein and Lynn Manseau of the UW Department of Physiological Chemistry.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `-CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% lineup [-INfile1=]picorna`

Prompted Parameters: None

Local Data Files:

`set.keys` (must be in your current working directory to be used)

`-MATRix=blosum62.cmp` assigns the scoring matrix for Zipping proteins
`-MATRix=swgapdna.cmp` assigns the scoring matrix for Zipping nucleic acids

Optional Parameters:

`-MSF` reads and writes sequence groups in MSF format
`-SINGLEcommnd` automatically returns to screen mode after each command
`-PROtein` sets sequence type to protein, and sets `find` to search for perfect symbol matches
`-NUCleotide` sets sequence type to nucleotide, and sets `find` to allow nucleotide ambiguity code matches
`-CONSR0W=0` sets the consensus row for a new sequence group
`-NOCONsensus` starts new sequence groups without a consensus row
`-LINesize=50` sets line length for output with the `PRint` command
`-BL0cksize=10` sets block length for output with the `PRint` command
`-FRAGExtension=frg` sets the file extension for each sequence when using FOSN format
`-CONSExtension=con` sets the file extension for the consensus when using FOSN format

LineUp

`-FOSNExtension=fil` sets the file extension for the list file when using FOSN format
`-MSFExtension=msf` sets the file extension for the multiple sequence file when using MSF format

LOCAL DATA FILES

The files described below supply auxiliary data to this program. The program automatically reads them from a public data directory unless you either 1) have a data file with exactly the same name in your current working directory; or 2) name a file on the command line with an expression like `-DATA1=myfile.dat`. For more information see Chapter 4, Using Data Files in the User's Guide.

Customizing Your Keyboard With SetKeys

You can use the program `SetKeys` to create a `set.keys` file that tells the `SeqEd`, `GelEnter`, `LineUp`, `GelAssemble`, and `SeqLab` sequence editors how to interpret the letters you type at the terminal. When entering gel readings, it is useful to have the symbols for G, A, T, and C under the fingers of one hand in the same positions as the lanes in your gel. `SeqEd`, `GelEnter`, `LineUp`, `GelAssemble`, and the `SeqLab` sequence editor automatically read the file `set.keys` if it is present in your local directory. If `set.keys` is absent, or if the sequence type is set to Protein (in `SeqEd` and `LineUp` only) the terminal keys retain their conventional meanings.

If you have a `set.keys` file in your directory, `SeqEd`, `GelEnter`, `LineUp`, and `GelAssemble` only respond to the keys that it redefines. You can edit the file `set.keys` with a text editor if some of the keys you want to use are not in it. Any keys not mentioned in `set.keys` appear to be dead in these sequence editors. In the `SeqLab` sequence editor, keys that are not redefined retain their normal meanings.

Several keys are vital for the control of `SeqEd`, `LineUp`, `GelEnter`, and `GelAssemble`; this means you are not allowed to redefine the keys for `/`, `[`, `]`, `{`, `}`, `(`, `)`, `:`, `,`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `0`, `<Ctrl>R`, `<Ctrl>D`, `<Ctrl>H`, `<Return>`, and `<Ctrl>E`.

Local Scoring Matrices

This program reads one or more scoring matrices for the comparison of sequence characters. The program automatically reads the program's default scoring matrix in a public data directory unless you either 1) have a data file with exactly the same name as the program default scoring matrix in your current working directory; or 2) have a data file with exactly the same name as the program default scoring matrix in the directory with the logical name `MyData`; or 3) name a file on the command line with an expression like `-MATRIX=mymatrix.cmp`. If you don't include a directory specification when you name a file with `-MATRIX`, the program searches for the file first in your local directory, then in the directory with the logical name `MyData`, then in the public data directory with the logical name `GenMoreData`, and finally in the public data directory with the logical name `GenRunData`. For more information see "Using a Special Kind of Data File: A Scoring Matrix" in Chapter 4, Using Data Files in the User's Guide.

When you use the `:ZIP` command to align a new sequence to the current consensus, `LineUp` reads a scoring matrix file containing values for every possible comparison between sequence symbols. By default, `LineUp` reads the file `swgapdna.cmp` for nucleotide sequence alignments and `blosum62.cmp` for protein sequence alignments.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-MATRix=mymatrix.cmp

allows you to specify a scoring matrix file name other than the program default. If you don't include a directory specification when you name a file with **-MATRix**, the program searches for the file first in your local directory, then in the directory with the logical name MyData, then in the public data directory with the logical name GenMoreData, and finally in the public data directory with the logical name GenRunData.

For more information see the Local Scoring Matrices section.

-MSF

sets LineUp to use MSF format. LineUp reads a sequence group from an MSF (multiple sequence format) file and writes an MSF file when storing a sequence group. The default FOSN format reads and writes individual sequence files, grouped in a list file. (See Chapter 2, Using Sequence Files and Databases in the User's Guide for a complete description of list files and MSF files.)

-SINGlecommand

sets LineUp to return automatically to Screen Mode after every command in Command Mode. **-NOSINGlecommand** is the default.

-PROtein and -NUCleotide

sets the sequence type for each sequence in the sequence group to be either protein or nucleotide. By default, if the first sequence in a sequence group is read from an existing sequence file, then the type of that sequence determines the type for the entire group. Also by default, if the first sequence in a sequence group is entered interactively from the keyboard, then LineUp sets the sequence type for the entire sequence group to be protein.

You can change the sequence type for the entire group when you are in LineUp with the **PROtein** and **NUCleotide** commands. **PROtein** tells LineUp to make pattern searches using perfect symbol matches. When LineUp is in the nucleotide state, if you type **/GARC** in Screen Mode, either of the patterns GAAC or GAGC is found. In the protein state, LineUp treats sequences as linear and will not find patterns that start at the end and continue into the beginning. In the nucleotide state, sequences are searched as though they are circular.

The automatic consensus definition is also different in the protein state than in the nucleotide state. In the nucleotide state, ambiguity codes make no contribution to the consensus. They are treated as if they were all Ns and are ignored. In the protein state, all characters have the same status.

-CONSR0W=0

tells LineUp on which row to put the consensus in a new sequence group. This command is only in effect if **-NOCONsensus** is *not* on the command line. The default is row 0.

LineUp

-CONsensus and **-NOCONsensus**

tells LineUp whether new files should start with a consensus sequence in the group. (Remember that you can create or remove the consensus sequence at anytime, so this is only a matter of convenience.) The default is **-CONsensus**.

-LINesize=n

sets the line length for *pretty*-style output created by the **PRint** command. The value must be in the range from 10 to 110. The default value is 50.

-BLocksize=n

sets the block length (number of bases between spaces) for *pretty*-style output created by the **PRint** command. The range of n must be 1 to line size. The default value is 10.

-FRAGEXtension=frg

sets the file extension that LineUp uses when reading and writing sequence files while in the FOSN format state. Do *not* include the dot separating the file from the extension. The default value is 'frg'.

-CONSExtension=con

sets the file extension that LineUp uses when reading and writing consensus sequence files while in the FOSN format state. Do *not* include the dot separating the file from the extension. The default value is 'con'.

-FOSNEXtension=fil

sets the file extension that LineUp uses when reading and writing list files while in the FOSN format state. Do *not* include the dot separating the file from the extension. The default value is 'fil'.

-MSFEXtension=msf

sets the file extension that LineUp uses when reading and writing multiple sequence format files while in the MSF format state. Do *not* include the dot separating the file from the extension. The default value is 'msf'.

Printed: December 1, 1998 10:52 (1162)

LISTFILE

FUNCTION

ListFile prints a file on a printer attached to your terminal's pass-through printer port.

DESCRIPTION

This utility simply turns on the terminal's pass-through printer port and prints a text file on an ASCII printer (such as a DEC LA-50) connected there.

EXAMPLE

Here is a session using ListFile to print the file `gamma.seq` on our ASCII printer.

```
% listfile
LISTFILE what file(s) ? gamma.seq
%
```

OUTPUT

The printer port is turned on, and `gamma.seq` is printed on the printer.

RELATED PROGRAMS

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are the Wisconsin Package file utilities programs.

RESTRICTIONS

ListFile is only for printing text files, not for making plots. The printer must be turned on. The terminal and printer must be set to the same baud rate.

SUGGESTIONS

If you use ListFile frequently and you find yourself repeatedly typing the same command modifiers, you can save yourself this effort by creating a foreign command in your `gcmymyinit.com` file. See "Customizing Your Login" in Chapter 1, Getting Started in the User's Guide for help defining foreign commands.

<CTRL>C

If you need to stop this program, use <Ctrl>C to reset your terminal and session as gracefully as possible. Searches and comparisons write out the results from the part of the search that is complete when you use <Ctrl>C.

ListFile

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `-CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% listfile [-INfile=]gamma.seq -Default`

Prompted Parameters: None

Local Data Files: None

Optional Parameters:

<code>-NOHEAding</code>	suppresses the heading at the top of the listing
<code>-PAGE=64</code>	sets the number of lines to print on each page
<code>-FORMFeeds</code>	sends a form feed instead of blank lines at end of page
<code>-BLAnklines=4</code>	sets the number of blank lines sent at the end of each page
<code>-NOMARgin</code>	doesn't send blank lines or a form feed after each page
<code>-ENDLINes=14</code>	sets the number of blank lines at end of printout
<code>-NOENDFORM</code>	suppresses form feed at end of printout
<code>-FILL=80</code>	compresses text and sets the length of the compressed lines
<code>-SPAcEs=1</code>	sets the number of spaces between words
<code>-NOPASSthrough</code>	suppresses the characters that turn the printer port on and off
<code>[-OUTfile=]RRX6:</code>	directs output to another terminal or file

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

`-NOHEAding`

suppresses the heading at the top of the print-out.

`-PAGE=64`

number of lines to print on each page.

`-FORMFeeds`

sends a form feed instead of several blank lines at the end of each page. Sending blank lines or a form feed at the end of each page positions the paper to begin printing on the next page.

-BLAnklines=4

sets the total number of blank lines to send at the end of each page if **-FORMFeeds** is not on the command line. Sending blank lines or a form feed at the end of each page positions the paper to begin printing on the next page.

-NOMARgin

doesn't send blank lines or a form feed at the end of each page This causes printing to be continuous across the page break, without a margin at the top and bottom of each page.

-ENDLINes=14

sets the number of blank lines at the end of the listing. The first page is shortened on the assumption that there were the same number of blank lines at the beginning.

-NOENDFORM

suppresses the form feed at the end of the printout.

-FILl=80

compresses the listing onto as little paper as possible. The words from the file are compressed together and shown on completely filled lines. You can set the length of the compressed lines with the optional parameter value.

-SPAcEs=1

lets you set the number of spaces separating each word if you are using the **-FILl** parameter.

-NOPASsthrough

suppresses the characters that turn the printer port on and off. The file appears on your screen.

-OUTfile=/dev/tty11

directs output to another terminal, device, queue, or file. The file will not have standard carriage control.

Printed: December 1, 1998 10:52 (1162)

LOOKUP

FUNCTION

LookUp identifies sequence database entries by name, accession number, author, organism, keyword, title, reference, feature, definition, length, or date. The output is a list of sequences.

DESCRIPTION

LookUp uses the Sequence Retrieval System (SRS) created by Dr. Thure Etzold to identify sequences in sequence databases (CABIOS 9(1); 49-57 (1993)). For example, you can find all of the protein sequences published by a particular author or all of the sequences whose annotation contains a particular word.

The expressions you use to find sequences in a database are known as *queries*. LookUp presents a form on your screen that lets you enter the elements of your query. Then LookUp finds all the sequences that contain those elements. The output of LookUp is a list file that can be used as input to any GCG programs that accept multiple sequence input.

EXAMPLE

Here is a session with LookUp that finds the sequences in the PIR database that were published by any author whose last name starts with *Smithies*.

```
% lookup
```

```
LOOKUP in what sequence libraries:
```

- a) sptrembl
- b) pir
- c) embl
- d) genbank
- e) em_tags
- f) gb_tags
- g) All libraries

- q) quit

```
Please choose one or more (* h *): b
```

LookUp

Complete the query form below:

```
      All text:
      Definition:
      Author: smithies<Ctrl>D
      Keyword:
      Sequence name:
      Accession number:
      Organism:
      Reference:
      Title:
      Feature:
On or after (dd-mmm-yyyy):          On or before (dd-mmm-yyyy):
Shortest sequence length:          Longest sequence length:

      Inter-field operator:  AND          Form of output list:  Whole Entries
```

Press <Ctrl>D to continue.

Searching pir

16 entries were found.

Do you wish to:

- 1) write out this list to a file
 - 2) preview the results
 - 3) refine the query
 - 4) choose different libraries
- q) quit

Please choose one (* 1 *):

What should I call the output file (* lookup.list *) ?

16 entries were written to "lookup.list"

%

OUTPUT

LookUp writes a list file naming the sequences which conform to your query. Associated with each sequence in the list file is an ID number. If you use this list file to specify the search set for another session with LookUp (for example with `-INfile=@lookup.list`), the ID numbers help LookUp quickly find the entries in the database.

```
!!SEQUENCE_LIST 1.0
LOOKUP in: pir of: "[SQ-AUT: smithies*]"

16 entries  October 22, 1998 15:03 ..

PIR1:HPHUR ! ID: 310d0003
! haptoglobin-related protein precursor - human
PIR1:UDHUP2 ! ID: 0c130003
! cystatin SN precursor - human

////////////////////////////////////

PIR4:I78580 ! ID: 27a90103
! hemoglobin gamma-G - human (fragment)
PIR4:I58221 ! ID: 28a90103
! hemoglobin gamma-A chain - human (fragment)
```

INPUT FILES

In most cases the search set for LookUp is an entire database. On the command line, this is specified like `-LIBRARY=pir`. *Note that this usage is different from that used by other GCG programs, which specify databases with a wildcard expression such as PIR:**. Alternatively, the search set can be specified by a list file created by a previous LookUp session. This is done by placing a parameter such as `-INFILE=@lookup.list` on the command line. Any sequences in the list that are not indexed for use with LookUp are ignored.

RELATED PROGRAMS

StringSearch identifies sequences by searching for character patterns such as "globin" or "human" in the sequence documentation. Names identifies GCG data files and sequence entries by name. It can show you what set of sequences is implied by any sequence specification. FindPatterns identifies sequences that contain short patterns like GAATTC or YRYRYRYR. You can define the patterns ambiguously and allow mismatches. You can provide the patterns in a file or simply type them in from the terminal.

BLAST searches one or more nucleic acid or protein databases for sequences similar to one or more query sequences of any type. BLAST can produce gapped alignments for the matches it finds. NetBLAST searches for sequences similar to a query sequence. The query and the database searched can be either peptide or nucleic acid in any combination. NetBLAST can search only databases maintained at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, USA. The programs FastA, TFASTA, FASTX, TFASTX, and SSEARCH can also be used to search databases or sequence sets local to your installation for sequences that are similar to a query sequence.

RESTRICTIONS

You can never be certain that the list of sequences in an output list contains every sequence of interest. Usually this is because of inconsistent annotation within the databases. See the CONSIDERATIONS topic for more information about this problem.

LookUp is still experimental, so please check your results carefully!

LookUp

The Wisconsin Package™ cuts sequences in GenBank that are longer than 350,000 bases into fragments of 110,000 bases each. Queries that make finds in such fragmented sequences return only the first fragment in the series of fragments. See the VERY LONG SEQUENCES topic for more information.

If you search both protein and nucleotide databases in the same session of LookUp, your output list will usually contain sequences of both types. Most GCG programs that analyze multiple sequences do not support lists of mixed sequence type, and so these lists are not suitable for input to programs such as PileUp, WordSearch, FastA, FrameSearch, etc.

If you use a list file that is the output from another GCG program to specify the search set for LookUp (for example with `-INfile=@lookup.list`), the program ignores the sequences that are not indexed for use with LookUp.

Most of the advanced features of GCG list files are not supported by LookUp. In particular, you cannot include a reference to another list within a list. You cannot include sequences specified by accession number. You cannot include sequences that are specified ambiguously. For instance, the specification `GenBank:PP*` has no meaning to LookUp. Note that the specification `Viral:PPV` is also ambiguous, as this could refer either to `EM_Vi:PPV` or `GB_Vi:PPV`. The specification `GB_Vi:PPV` is allowed, since this plum pox virus sequence is indexed for LookUp.

Very ambiguous queries do not always work. For example, if you search for all sequences whose names start with *hum*, LookUp loops endlessly.

Not all fields are present in every database. For example, PIR does not have a Feature or Date index, and SWISS-PROT does not have a Title index.

ABOUT DATABASES

What is a Database?

A database is a structured way to represent a group of things that have common attributes. Most sequence databases consist of different fields such as accession number, definition, author, etc., that are filled with appropriate values like *U01317, Human beta globin region on chromosome 11, Smithies*, etc. Fields are grouped together into larger units referred to as entries. LookUp identifies sequences in GenBank, PIR, SWISS-PROT, and EMBL based on the values found in the different fields associated with each sequence entry.

What is an Index?

An indexed database has one or more of its fields organized into a data structure that allows rapid searching. An indexed field is like the index of a book. The subjects in the book are organized alphabetically into an index at the back of the book. When you look up a subject in the index, you will find the page numbers where that subject is mentioned in the body of the book. Likewise in an indexed database, if there were an author index, you could find all of the entries in the database where Smithies is one of the authors just by looking up the name in the index.

What Fields in the Databases Can I Search?

The Sequence Retrieval System (SRS) on which LookUp is based has indices for each of several fields that usually occur in the annotation of a sequence database. These indexed fields are: accession number, author, date, definition, feature, keyword, length, entry name, organism, reference, and title. Each of these indices is described in detail under the INDEXED FIELDS topic below.

THE QUERY FORM

The query form has a line for each field that has been indexed for retrieval with LookUp. You can search for values in one or more fields, and LookUp finds all the sequences containing those values. To move the cursor from field to field, use the <Up-Arrow>, <Down-Arrow>, or <Return> keys.

The field on the query form that is labeled "Form of output list" toggles between the values Whole entries (the default) and Fragments when the cursor is positioned in the field and you press the <Space Bar>. You may want to use the Fragments value if you are searching the Features index for a particular feature that occurs in the sequences. LookUp can represent that feature more precisely by showing its beginning and ending positions and, for nucleic acid sequences, the strand. See the FRAGMENT OUTPUT topic below.

WRITING QUERIES

Keep the following guidelines in mind as you write LookUp queries.

Logical Operators

You can type one or more values on each line of the query form. There are three logical operators that let you combine values in different ways:

- **AND.** Use `&` to specify AND. `A & B` means find all entries that contain both A and B.
- **OR.** Use `|` to specify OR. `A | B` means find all entries that contain either A or B.
- **BUT-NOT.** Use `!` to specify BUT-NOT. `A ! B` means find all entries that contain A but do not contain B. Notice that BUT-NOT is order dependent. `A but not B` would find a completely different set of entries from `B but not A`.

Special Case: (C shell only) If you are specifying a query on the command line, and the query expression contains the `!` (BUT-NOT) logical operator, you must preface the `!` with a backslash (`\`), for example `-AUTHOR=McDonald!\Strand`. (This does not apply to Korn shell users.) If you are using an init file to specify a query, you must enclose any query expression that contains a `!` in double quotation marks. In this case, you do not include a backslash before a `!`, for example `-AUTHOR="McDonald!Strand"`.

If you type values for more than one index, LookUp finds entries where each field conforms to the values you have typed. This is equivalent to saying that LookUp joins the values on the different lines with the logical operator AND. You can change this to OR by moving the cursor to the field labeled "Inter-field operator" and pressing the <Space Bar>.

Case Insensitivity

All queries are case insensitive. Regardless of whether you type uppercase or lowercase letters, LookUp converts all queries to uppercase.

Parentheses

If more than one logical operator appears in an expression without parentheses, LookUp evaluates the expression from left to right. However, you can group expressions within a query to define the order in which they are performed. Use parentheses to group expressions you want LookUp to evaluate first. For example, when you type `smithies & (slightom | Blechl)` as the value for the Author field, LookUp first searches for sequence entries containing

LookUp

references with Slightom or Blechl as authors. Then, out of those entries it searches for those which also contain Smithies as an author.

Wildcard Extension

LookUp accepts question marks (?) or asterisks (*) as wildcards anywhere within a value. A question mark represents any single character. If you type `s?ith`, you will retrieve entries containing authors named *Smith*, *Slith*, *Sjith*, etc., but not named *Sith*.

An asterisk represents zero or more characters. Typing `*smith*` will retrieve entries with authors named *Smith*, *Hocsmith*, *Smithies*, *Hocsmithels*, etc. Values with leading wildcards, such as `*smith`, significantly reduce the speed of LookUp. Trailing wildcards usually have little effect on performance.

By default, LookUp treats every value in your query as if it ended with an asterisk wildcard. This automatic wildcard extension means that when you type `pseudo`, LookUp treats it as `pseudo*` and will retrieve entries containing the patterns *pseudo*, *pseudo-*, *pseudogene*, *pseudoknot*, etc.

You can turn off automatic wildcard extension for a single value by appending a pound sign (#) to the value, for instance `pseudo#`. LookUp then will find only those entries where the word *pseudo* occurs by itself. You can turn off automatic wildcard extension with `-NOWILdcardextension`. If you have automatic wildcard extension turned off, you can still use `*` to tell LookUp to extend a particular value. When automatic wildcard extension is turned off, the `#` character is treated as a literal part of your query.

Double Quotation Marks

If you are specifying a query on the command line, and the value contains a shell special character, such as a space, #, or &, you must enclose the value in double quotation marks ("), for example `-KEYword="ribosomal proteins"` or `-DEFinition="transport#"`.

Special Case: If you are specifying a query on the command line, and the value has a comma in it, you must enclose the value in single (') AND double quotation marks ("), for example `-AUTHor="'Slightom,J.L.'"`. (Within init files however, use only the double quotation marks.) If you specify a query where the value has a comma or dash in it, you must enclose the value in double quotation marks ("), for example `-AUTHor="Slightom,J.L."`.

INDEXED FIELDS

Below is a description of each indexed field on the LookUp query form. Following the example for each index is the parameter you would use to set a value for this index from the command line.

All text: `globin & duplication` (-ALLtext="globin & duplication")

This index is composed of most indices combined, including: Author, Definition, Feature, Keyword, Organism, Reference, and Title. If you think a word like *globin* or *duplication* might occur in a title, or a definition, or a feature, this index can search all three indices at once without making you type the value for each index separately.

Definition: **globin** (-DEFInition=**globin**)

This index contains each word in the definition of every database entry. The words are each indexed separately, without any regard for the order in which they appear. A definition like *Human beta globin region on chromosome 11* generates completely independent indices for the words *human*, *beta*, *globin*, *region*, *on*, *chromosome*, and *11*. A query value that would likely find this definition would be **human & beta & globin**. Hyphenated terms, such as *beta-globin*, are indexed as two separate words.

Author: **Slightom** (-AUTHor=**Slightom**)

This index contains all of the authors cited in each database entry. Most databases do not use "et al.," so second, third, and fourth authors will usually be present. The index includes the author's surname followed by first and middle initials. No spaces separate the surname and initials. For example, Dr. J. L. Slightom would be indexed as *Slightom,J.L.* If you do not include the initials, LookUp will find all entries with an author whose surname starts with *slightom*.

Keyword: **ribosomal proteins** (-KEYword="**ribosomal proteins**")

This index contains every keyword in each database entry. Unlike most of the fields in LookUp, keyword values may contain spaces, as in *ribosomal proteins*. The discipline of assigning keywords differs greatly from database to database; for example, you cannot be sure that both the organism name and enzyme superfamily name will appear in every entry's keyword list. (See the All text index and also the CONSIDERATIONS topic.)

Entry name: **Humhbb** (-NAME=**Humhbb**)

This index contains all of the sequence names in each database entry. These names (referred to as locus names in GenBank) should be unique, so you should not find more than one entry in a database for any name.

Accession number: **J00179** (-ACCession=**J00179**)

This index contains all of the accession numbers in each database entry. While primary accession numbers are supposed to be unique in each database, secondary accession numbers can appear in more than one sequence.

Organism: **Homo sapiens** (-ORGanism="**Homo sapiens**")

Most sequence databases name the organism from which the sequence is derived. Organism values may contain spaces. In recent years both EMBL and GenBank have used systematic nomenclature whenever possible. If you want to specify a species name, the genus must precede the species (for example *Homo sapiens*). Typing just **sapiens** will find nothing.

The higher-order systematic names like Eukaryota, Animalia, Metazoa, Chordata, Vertebrata, Mammalia, Theria, Eutheria, Primates, Haplorhini, Catarrhini, Hominidae are indexed independently. If your query is not a species name, use only one higher-order systematic name.

LookUp

Reference: `EMBO&6#&523-&1987`

`(-REFERENCE="EMBO&6#&523-&1987")`

*** The Reference index does not work correctly in this release! ***

Each reference is indexed into four independent subfields: journal name, volume number, beginning page number, and year. You can specify values for any or all of these subfields on this line. The order of the subfield values does not matter.

Journal names are indexed exactly as they appear in each database. If the curators of one database call a journal *Nucleic Acids Res.*, the curators of another call the same journal *Nucl. Acids Res.*, and a third database uses *NAR*, these differences will be reflected in the indices used by LookUp. Notice, however, that the expression `(NAR | NUCL) & 1989` would probably find all of the sequences published in 1989 in Nucleic Acids Research.

The volume is a number less than 1950, the date is a number greater than 1950, and the beginning page is a number followed by a hyphen. If you specify values for more than one subfield, you must join the subfields with logical operators (see Logical Operators in the WRITING QUERIES topic).

For most references, specifying both the volume number and starting page number is definitive.

Title: `globin & duplication`

`(-TITLE="globin & duplication")`

This index contains all words in the titles of each citation in the databases. Some databases do not include the title for each citation, so failure to find a word that you think occurs does not imply that the reference of interest is not cited in one of the databases (see the Reference index). The words are indexed without regard for the order in which they first appeared. A title like *A history of the human fetal globin gene duplication* generates independent indices for each separate word: *a*, *history*, *of*, *the*, *human*, *fetal*, *globin*, *gene*, and *duplication*. An expression likely to find this title, if it were present, would be: `globin & duplication & history`.

This index is not available for SWISS-PROT.

Feature: `cds`

`(-FEATURE=cds)`

A feature is a region of a sequence that is identified in the feature table of a sequence database. Associated with each feature is a set of words that may include a gene name, a function, an EC number, etc. Every word associated with each feature is indexed independently without regard for order. If you type `cds` as the value, every coding sequence that is documented by a CDS feature will be found.

You can have the output show where each feature occurs within a sequence by selecting Fragments instead of Whole entries on the query form (see the FRAGMENT OUTPUT topic).

This index is not available for PIR.

On or before: **31-dec-1994** (-**LATest=31-dec-1994**)
 On or after: **1-jan-1994** (-**EARliest=1-jan-1994**)

A Date index contains the date sequences were entered or updated in each database. With these two fields, you can identify sequences that were updated between any two dates. The format for a date value is DD-MMM-YYYY where D, M, and Y stand for day, month, and year respectively. The English abbreviations for the months of the year are: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.

This index is not available for PIR.

Shortest sequence length: **10** (-**SHORtest=10**)
 Longest sequence length: **400** (-**LONGest=400**)

The lengths of every sequence in each database are indexed. You can use these fields to restrict the sequences found to those with certain lengths, for example between 10 and 400 characters.

LIST INPUT

Normally LookUp's search set consists of all of the sequences in one or more of the sequence databases. If you have a list file created by an earlier session with LookUp, you can use this smaller set of sequences as your search set by adding a parameter like `-INfile=@lookup.list` to the command line. (An at sign (@) must precede the name of the list file.)

You should use only list files that were created by earlier sessions with LookUp, as the input list file can contain only sequences that have been indexed for searching with LookUp. You cannot add sequences to the list that were not in the libraries when the LookUp indices were created.

Most of the advanced features of GCG list files are not supported by LookUp. In particular, you cannot include a reference to another list within a list. You cannot include sequences specified by accession number. You cannot include sequences that are specified ambiguously. For instance, the specification `GenBank:PP*` has no meaning to LookUp. Note that the specification `Viral:PPV` is also ambiguous, as this could refer either to `EM_Vi:PPV` or `GB_Vi:PPV`. The specification `GB_Vi:PPV` is allowed, since this plum pox virus sequence is indexed for LookUp.

FRAGMENT OUTPUT

LookUp normally writes a list of sequences defined only by their database and entry names. Each element of such a list refers to the whole entry. The field on the query form that is labeled "Form of output list" toggles between the values Whole entries (the default) and Fragments when the cursor is positioned in the field and you press the <Space Bar>. You may want to use the Fragments value if you are searching the Features index for a particular feature that occurs in the sequences. LookUp can represent that feature more precisely by showing its beginning and ending positions and, for nucleic acid sequences, the strand. Features consisting of separate fragments are listed contiguously in the output list file and share the same `Join` name.

Here is some fragment output from a query designed to find complete coding regions for genes encoding xanthine dehydrogenase.

LookUp

```
LOOKUP in: embl,genbank of: "[SQ-ALL: complete* & xanthine* &
                                dehydrogenase*] > [SQ-FTS: cds*]"
```

```
6 features July 3, 1995 14:34 ..
```

```
GB_IN:DROXDHA Begin: 1086 End: 1139 Strand: + Join: DROXDHA-8
GB_IN:DROXDHA Begin: 2164 End: 4776 Strand: + Join: DROXDHA-8
GB_IN:DROXDHA Begin: 4839 End: 5981 Strand: + Join: DROXDHA-8
GB_IN:DROXDHA Begin: 6049 End: 6216 Strand: + Join: DROXDHA-8
GB_IN:DROXDHA Begin: 6284 End: 6334 Strand: + Join: DROXDHA-8
GB_PR:HSU06117 Begin: 64 End: 4065 Strand: + Join: HSU06117-3
GB_PR:HUMXDH Begin: 131 End: 4147 Strand: + Join: HUMXDH-2
GB_PR:HUMXDHA Begin: 58 End: 4059 Strand: + Join: HUMXDHA-2
GB_RO:RATXDHA Begin: 27 End: 4022 Strand: + Join: RATXDHA-2
```

If you are querying LookUp from the command line, you can get this form of output with the **-FRAGments** parameter.

PIR does not support fragment output.

Extracting Features

Each fragment in the fragment output list file is accompanied by `Begin`, `End`, `Strand` and `Join` sequence attributes. You can use the `Assemble` program to extract the features into separate GCG sequence files. All sequences listed contiguously in the list file that share the same `Join` name are concatenated into a single sequence and the resulting sequence file is given the same name as the `Join` name. Fragments listed individually are extracted into separate sequence files, each with the same name as the corresponding `Join` name.

Using the features list file example above, `Assemble` writes five new sequence files. The first file, called `droxdha-8.seg`, contains the assembly from the first five sequence segments in the list file. The second file, `hsu06117-3.seg`, contains the assembly from the sixth sequence segment in the list file. Similarly, the remaining three sequence segments in the list file are extracted into separate sequence files. See the entry for `Assemble` in the Program Manual for more information about extracting features according to sequence attributes in a list file.

You can use the `Translate` program to translate features according to the sequence attributes in a features list file and write each translated sequence into its own GCG sequence file. See the entry for `Translate` in the Program Manual for more information about translating features according to sequence attributes in a list file.

Features with Ambiguous Begin or End Positions

Use **-complete** to ignore features whose start or end positions are not accurately identified. See the **PARAMETER REFERENCE** topic for more information.

CONSIDERATIONS

Note that the databases are inconsistent in their annotation. You can find misspellings as well as differences in hyphenation and the type of information entered in fields. *As a result, you can never be certain that the list of entries in an output list contains every sequence of interest.*

Suppose you wanted to find pseudogenes. You might consider searching nucleotide database entries using `pseudo*` as the value for the Definition field. But you cannot assume that such a search would be exhaustive. The text you choose may not have been used by every annotator who created an entry containing a pseudogene. If the pseudogene were an incidental part of the entry, the annotator may have noted it only in the feature table. For example, the definition for the sequence GB_Pr:Humhbb, which contains two pseudogenes, is *Human beta globin region on chromosome 11*.

In addition, be aware that all databases contain spelling errors; the misspelling *psuedo* occurred 10 times in the definitions of GenBank Release 95.0.

Hyphens in particular are used inconsistently. For example, to find as many entries as possible that are pseudogenes, you should search for `pseudo-gene` as well as `pseudogene`. Another example where inconsistent use of hyphens can cause problems is the globin family. GenBank definition lines may contain the terms *beta-globin*, *beta globin*, and *beta-hemoglobin*. One way to deal with this is to specify just one of the words, since LookUp indexes the words on either side of a hyphen separately. You can also use a leading wildcard. For example, if you type `*globin`, LookUp will retrieve the following members of the globin family: haptoglobin, hemoglobin, haemoglobin, myoglobin, cyanoglobin, plakoglobin, alphaglobin, alpha-globin, alpha-globin-3, alpha-1 globin, beta-min-globin, beta-3-globin, beta-2-globin, beta-H1-globin, beta-B globin, uteroglobin, y2-globin, beta-major globin, zeta-globin, and so on. Note that using leading wildcards significantly reduces the speed of LookUp.

Another consideration is that a value such as `pseudo` may occur in words other than *pseudogene*. In addition to pseudogene sequences, your output list may also contain RNA sequences known to form pseudoknots or sequences from the organism *Pseudomonas*.

VERY LONG SEQUENCES

Future releases of GenBank are expected not to have any sequences longer than 350,000 bases. However as release 10.0 of the Wisconsin Package was being prepared, two sequences longer than 350,000 bases were still present in GenBank proper (GB_Ba:Ecouw67 and GB_Pl:Scchrix) and several dozen such sequences were present in the High Throughput Genome (HTG) division. These sequences are broken into overlapping fragments in the Wisconsin Package. The 372 kilobase sequence Ecouw67, for instance, is divided into four fragments: Ecouw67_0, Ecouw67_1, Ecouw67_2, and Ecouw67_3. Each fragment is 110,000 bases long and overlaps the one following it by 10,000 bases. All of the annotation appears with the first fragment, so LookUp normally returns only the first fragment if your query makes a hit on one of these long sequences. If you are searching for features and you are asking for fragment output, LookUp tries to infer which fragment contains the feature of interest. *If a feature you find spans two of the fragments, it will not be represented correctly.*

SUGGESTIONS

Note that the output list can contain any number of entries and may result in an extremely large output list file.

Become familiar with the format of each database by doing a number of simple queries and looking at the output carefully. The topic ANNOTATING LISTS tells you how to display the original records from each database.

Use the `#` symbol to turn off automatic wildcard extension, thereby reducing the number of entries in your output (see Wildcard Extension in the WRITING QUERIES topic).

If you search both protein and nucleotide databases in a single session, your output list will probably contain sequences of both types. Most GCG programs that do multiple sequence analysis do not

LookUp

support lists of mixed sequence type. For example, mixed lists are not suitable for input to programs such as PileUp, WordSearch, FastA, and FrameSearch. Therefore, if you want to use the output list as input to other Wisconsin Package programs, you should search protein and nucleotide databases separately.

ANNOTATING LISTS

LookUp normally writes a simple list of sequences identified by entry name and definition. The `-ANNOtate` parameter lets you add other annotation from the original sequence record to each sequence in the list to help you identify the sequence and understand how LookUp processed your query.

The values you can use with this parameter correspond to fields that are indexed for LookUp: `ACCession`, `AUTHor`, `DATE`, `DEFInition`, `FEAture`, `NAME`, `KEYword`, `ORGanism`, `REFerence`, and `TITle`. For example, `-ANNOtate=AUTHor` annotates each sequence in an output list with author names.

If you have chosen Whole entries for the "Form of output list" field of the query form when `-ANNOtate=FEAture`, LookUp includes the whole feature table next to each sequence. This can create large output files. If you have chosen Fragments for this field when `-ANNOtate=FEAture`, LookUp includes only the feature of interest.

The date does not appear on a separate line in Genbank, so if you want to see the date for GenBank entries, use `-ANNOtate=NAME` instead of `-ANNOtate=DATE`. Note that LookUp does not support date or reference annotation for PIR.

Annotated lists, like other lists, are compatible with GCG programs that support multiple sequence specifications.

You can turn off annotation altogether with `-NOANNOtate`. LookUp is much faster with annotation turned off.

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `-CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Minimal Syntax: `% lookup [-ALLtext=]globin -Default`

Prompted Parameters:

<code>-LIBrary=pir[,...]</code>	specifies one or more data libraries
<code>-ALLtext=globin</code>	searches all text indices
<code>-DEFInition=globin</code>	searches definition index for one or more words indexed independently, eg. "Globin & Region"
<code>-AUTHor=smithies</code>	searches author index for one or more, e.g. "Smithies, O. & Slightom, J.L."
<code>-KEYword=globin</code>	see document before using keywords
<code>-NAME=hsgg13</code>	searches entry name index
<code>-ACCessionnumber=s12345</code>	searches accession number index

-ORGanism="Homo Sapiens" searches genus and species index
 -REFerence=cell&1981 searches complete reference index
 -TITle=history searches title of citation index
 -FEATure=gamma searches for any word in a feature table
 -SHOrtest=100 finds only sequences of length 100 or more
 -LONgest=400 finds only sequences of length 400 or less
 -EARliest=01-apr-1992 searches for sequences modified on or after
 specified date
 -LATest=30-apr-1992 searches for sequences modified on or before
 specified date
 -MATch=or specifies inter-field logic (AND is default)
 -OUTfile=lookup.list names output file for list of sequences

Local Data Files: None

Optional Parameters:

-NOWILdcardextension turns off automatic wildcard extension
 -INfile=@lookup.list searches in lookup.list instead of libraries
 -ANNotate=feature[,...] shows fields from original annotation in output
 acceptable values include: ACCession, AUTHor,
 DATE, DEFinition, FEATure, NAME, KEYword,
 ORGanism, REFerence, and TITle
 -FRAGments shows features as fragments instead of whole
 entries
 -COMplete shows only features with unambiguous coordinates
 -MONitor shows databases searched and how many hits found

ACKNOWLEDGEMENT

LookUp is a database application that makes use of the Sequence Retrieval System (SRS) written by Dr. Thure Etzold of the European Molecular Biology Laboratory (EMBL). SRS is described in CABIOS 9(1); 49-57 (1993) and Appl. Biosci 9; 59-64 (1993). We are grateful to Thure Etzold, Patrick Argos, and EMBL for making the internals of SRS available for use with LookUp. Scott Rose wrote the LookUp application for GCG and John Devereux wrote the documentation.

If you know how to browse the World Wide Web, you can look at SRS under the URLs
<http://srs.ebi.ac.uk:5000/> <http://srs.ebi.ac.uk>

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

-LIBrary=SwissProt,GenBank

searches the SwissProt and GenBank data libraries.

LookUp

-ALLtext=Globin

searches all text indices for the word *globin*. The text indices are Author, Definition, Feature, Keyword, Organism, Reference, and Title. (Note that the Name and Accession Number indices are *not* included.)

-DEFinition=Globin

searches for entries whose definition line contains the word *globin*.

-AUTHor=Smithies

searches for entries derived from publications containing an author whose surname is Smithies.

-KEYword=Globin

searches for entries that contain the word *globin* in their KEYWORDS field.

-NAME=hsgg13

searches for the sequence entry whose name is HSGGL3. Depending on the database, the name may correspond to the LOCUS name, the ID name, the ENTRY name, etc.

-ACCESSIONnumber=S12345

searches for the sequence entry whose accession number is S12345.

-ORGanism="Homo Sapiens"

searches for any sequence entries deriving from the organism *Homo sapiens*. The genus and species names are indexed as a unit. If you want to search on the species name alone, you must preface it with a wild card: **-ORGanism=*Sapiens**.

-REFerence=Cell&1981

searches for entries reported in the journal *Cell* in 1981. (This index does not work correctly in this release.)

-TITLE=History

searches for sequences reported in articles whose name contains the word *history*.

-FEAture=Gamma

searches for sequence entries whose feature table contains the word *gamma*.

-SHORtest=100

searches for sequences containing 100 or more residues.

-LONGest=400

searches for sequences containing 400 or fewer residues.

-EARliest=01-apr-1992

searches for sequence entries that were entered or last modified on or after April 1, 1992

-LATest=30-apr-1992

searches for sequence entries that were entered or last modified on or before April 30, 1992.

-MATCH=OR

specifies the logic to be used to combine index fields (the default is AND).

-NOWILdcardextension

LookUp normally treats all values in your query as if they ended with an asterisk wildcard (See Wildcard Extension in the WRITING QUERIES topic). You can suppress this automatic wildcard extension either by adding a # to the end of any value that you do not want extended or by using this parameter to suppress it for all values. With automatic wildcard extension turned off, you must explicitly append an asterisk to make any particular field value wild.

-INfile=@lookup.list

LookUp can use a list file created during a previous session with LookUp as the search set. A parameter like the one in this example can be used in place of **-LIBrary=swissprot**. If both **-LIBrary** and **-INfile** are used, the program uses the list file and ignores the library parameter.

-ANNotate=AUTHor[...]

LookUp normally writes a simple list of sequences identified by entry name and definition. The **-ANNotate** parameter lets you add other annotation from the original sequence record to each sequence in the list to help you identify the sequence and understand how LookUp processed your query.

The values you can use with this parameter correspond to fields that are indexed for LookUp: **ACC**ession, **AUTH**or, **DATE**, **DEF**inition, **FEA**ture, **NAME**, **KEY**word, **ORG**anism, **REF**erence, and **TIT**le. For example, **-ANNotate=AUTHor** annotates each sequence in an output list with authors.

If you have chosen Whole entries for the "Form of output list" field of the form and **-ANNotate=FEA**ture, LookUp includes the whole feature table next to each sequence. This can create large output files. If you have chosen Fragments for this field and **-ANNotate=FEA**ture, LookUp includes only the feature of interest.

The date does not appear on a separate line in Genbank, so if you want the date for GenBank entries in your output list, use **-ANNotate=NAME** instead of **-ANNotate=DATE**. Note that PIR entries are not indexed for date.

Annotated lists, like other lists, are compatible with GCG programs that support multiple sequence specifications.

LookUp

You can turn off annotation altogether with `-NOANNotate`. LookUp is much faster with annotation turned off.

`-FRAGments`

LookUp normally writes a list file even if you search for sequences in the Feature index. In addition, it can show the exact locations of most features. If you select Fragments from the "Form of output list" field in the form or use `-FRAGments`, LookUp will represent features with their beginning and ending coordinates, the strand on which they are found, and whether they are joined to other features appearing below them in the list. These multi-fragment features can be joined together into a new composite sequence with `Assemble` or `Translate`.

`-COMplete`

Some features have starting and ending positions that are beyond the bounds of the sequence data archived in a particular database entry. In the features table of GenBank and EMBL, these features are represented with ranges that have a `<` before the beginning coordinate and/or a `>` before the ending coordinate. Here is a feature whose beginning lies before the first base stored in the sequence entry:

```
CDS                <1. .81
                   /note="gamma globin; NCBI gi: 386767"
```

There is no way to represent this ambiguous coordinate in a GCG list file, so it is written as if the coordinate were exact. Here is how the feature is normally represented in the list file:

```
GBPR:HUMHBG3E  Begin: 1 End: 81 Strand: + Join: HUMHBG3E-2 !Id: 0200...
!      CDS                <1. .81
!                               /note="gamma globin; NCBI gi: 386767"
```

If you want to keep such features out of your LookUp output, you should use `-COMplete`. The ambiguous features will still appear in your output, but they are printed with an exclamation point preceding their names so that programs that use this list as input will ignore them. Here is how the fragment is represented when you use `-COMplete`:

```
! GBPR:HUMHBG3E  Begin: 1 End: 81 Strand: + Join: HUMHBG3E-2 !Id: 0200...
!      CDS                <1. .81
!                               /note="gamma globin; NCBI gi: 386767"
```

`-MONitor`

This program normally monitors its progress on your screen. However, when you use `-Default` to suppress all program interaction, you also suppress the monitor. You can turn it back on with this parameter. If you are running the program in batch, the monitor will appear in the log file.

LookUp prints a period on your screen every time it writes 100 lines into your output file.

Printed: December 1, 1998 10:52 (1162)

LPRINT

FUNCTION

LPrint prints text file(s) on a PostScript printer connected to LPrintPort.

DESCRIPTION

We use this utility to list text files on our PostScript laser printers. If you have a PostScript printer, your system manager has defined the logical name LPrintPort to be the port to which that printer is connected.

If the logical name LPrintPort is really your terminal, LPrint turns on the terminal's pass-through printer port before printing the file. You can send the output to a disk file if you need to move the file to another machine before printing it.

EXAMPLE

Here is a session using LPrint to print the file gamma.seq on our PostScript laser printer.

```
% lprint
LPRINT what file(s) ? gamma.seq
%
```

OUTPUT

The output is in the Courier font at 12-point scale.

RELATED PROGRAMS

Replace, CompressText, OneCase, ShiftOver, DeTab, ChopUp, LPrint, and ListFile are the Wisconsin Package file utilities programs.

RESTRICTIONS

LPrint is only for printing text files, not for making plots. The printer must be turned on. The terminal and printer must be set to the same baud rate.

FORM FEEDS

Programs like Map can keep clusters of related material from spanning page boundaries by putting a form-feed character on a line by itself whenever the next cluster of material will not fit on the current page. If the first character on any line is a form-feed character, LPrint stops printing on this page and continues printing the material from subsequent lines on the next page. If there are any other characters on the line with the form feed, they are *not* printed.

LPrint

COMMAND-LINE SUMMARY

All parameters for this program may be added to the command line. Use `-CHECK` to view the summary below and to specify parameters before the program executes. In the summary below, the capitalized letters in the parameter names are the letters that you *must* type in order to use the parameter. Square brackets ([and]) enclose parameter values that are optional. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

Syntax: `LPrint [-INfile=]gamma.seq -Default`

Required Parameters: None

Local Data Files: None

Optional Parameters:

<code>-OUTfile=gamma.ps</code>	directs the output to somewhere other than LPrintPort
<code>-COLumns=80</code>	controls the number of characters per line
<code>-NOHEAding</code>	suppresses the heading on each page
<code>-LANdscape</code>	prints lines lengthwise (eleven-inch width)
<code>-MARGin=5</code>	moves the left margin five characters to the right
<code>-PAGE=60</code>	sets the number of lines per page
<code>-BEGin=1</code>	sets the line number where printing begins
<code>-END=10</code>	sets the line number where printing ends
<code>-NUMbering</code>	prints a line number at the beginning of each line
<code>-CTRLD</code>	adds <Ctrl>D at the end of the output

LOCAL DATA FILES

None.

PARAMETER REFERENCE

You can set the parameters listed below from the command line. For more information, see "Using Program Parameters" in Chapter 3, Using Programs in the User's Guide.

`-OUTfile=temp.ps`

directs output to a file or another device instead of to LPrintPort.

`-COLumns=80`

Usually, LPrint shrinks the font to fit the number of characters on the longest line in your first input file. You can modify the font size to support between 40 and 255 characters per line with this parameter.

`-NOHEAding`

Usually, LPrint puts a heading that shows the file name, the date and the page number. You can suppress that heading with this parameter.

-LANDscape

The text orientation can be rotated to print parallel to the long axis of the paper with this parameter.

-MARGin=3

Usually, LPrint prints the first character in the first column (margin = 0). This parameter lets you shift the listing to the right by adding one or more blank characters in the margin. (The number of characters per line of output is the sum of the column and margin settings.)

-PAGE=60

Usually, LPrint prints 60 lines per page on regular listings and 42 lines per page in landscape orientation. You can set the number of lines per page with this parameter.

-BEGin=1

You can make LPrint begin printing the file from a line other than the first line with this parameter.

-END=10

You can make LPrint stop printing the file at a particular line with this parameter.

-NUMbering

makes LPrint print a line number at the beginning of each line.

-CTRLD

sends a <Ctrl>D (^D, ASCII 4) at the end of the stream of PostScript instructions (some terminal servers require this in order to liberate the terminal for use by others).

Printed: December 1, 1998 10:52 (1162)