NLS TEXTBOOK

1 June 1980

PREFACE

This document is based on a collection of chapters prepared by the Augmentation Resources Center of Tymshare, Inc. This edition is intended to describe NLS as it exists on the Information Sciences Institute (ISI) systems B, C, D, E, and F. There may be portions of the system or the examples that work a little differently than described in this document, if so please report the discrepancy to LINDA@ISIF.

The original authors, Nina Zolotow, Caroline Rose, and Dirk Van Nouhuys, deserve a great deal of credit for creating this material and its organization.

-- Jon Postel
-- Lynne Sims
-- Linda Sato
1 June 1980

CONTENTS

INTRODUCTION TO THE NLS TEXTBOOK

INTRODUCTION TO THE NLS TEXTBOOK

This textbook introduces the NLS system. It describes what you can do with NLS and how to do it. We also hope that it will help you adapt to a new style of working. By using NLS's special tools and techniques rather than typewriters, pencils, and paper, you will be better able to organize your thoughts as well as your information.

The textbook was written for a very wide range of NLS users. Some are highly technical people, with a great deal of computer experience, who look forward to learning a new computer system with confidence and pleasure. Others are people who have never seen a computer or a computer terminal before; they will be using NLS for the kind of work that they have always done without the aid of a computer.

Just as the backgrounds of our users vary greatly, so do their requirements for NLS documentation. People who are trained by experienced NLS users need a manual they can use for review or reference, while others want to learn NLS on their own by reading some kind of user guide from beginning to end. Still other people want a document that will give them a general idea of NLS's capabilities; they may not necessarily want to use the system or ever sit down at an NLS terminal.

In planning the NLS textbook, we have tried to accommodate as many of these people as possible. The textbook consists of a number of lessons, and their organization is flexible enough so you can read them from beginning to end or skip from one section to another.

The language in the textbook is nontechnical and informal, and we have tried to cover many subjects simply and thoroughly. Our basic approach has been that too much information is better than not enough.

Because learning NLS is similar to learning a foreign language, the textbook is organized like a language textbook. The lesson entitled "Beginning Use of NLS" introduces you to the system as a whole; other lessons give you the information necessary to learn a particular aspect of the system.

The most effective way to use NLS is at an NLS display terminal. For this reason, the lessons assume that you do or will have access to an NLS display. Using NLS at a display terminal is a particularly visual experience--it is often hard to put into words what the result of a certain command will be or how it will feel to work with the system. Therefore, we have included a great many examples that you can follow as exercises so that, if you wish, you can see the results that we describe.

The following section describes in detail the organization of a typical lesson so that you can understand how the various sections relate to each other and can tell which sections will be the most helpful for your particular situation.

THE STRUCTURE OF A LESSON

Every lesson begins with a very brief description of the topics it covers, possibly suggesting the type of work you can do with what you will learn. You are also told what you should already know before starting the lesson. The remainder of each lesson has the structure shown below.

Introduction

This section presents the new concepts and background information you need to know in order to understand the lesson.

Vocabulary

Here you will see an alphabetical list and short definitions of all terms and commands discussed in the lesson. You may want to read through this list to get an overview of the contents of the lesson; however, it is primarily intended to be used for reference. You should refer to it when you are reading the lesson if you encounter a term whose meaning you have forgotten or have never learned. You may also find this list useful when you later refer to the lesson to look up a term or command that you learned from it.

The body of the lesson

The body of a lesson may consist of any number of sections, each under an appropriate heading. These sections contain the primary discussion of the subject of the lesson.

Exercises

This section contains exercises to test what you have learned. Solutions are provided later, following the end of the lesson.

Suggested Projects

Some lessons include this section to suggest ways that you can gain experience in using what you have learned.

Summary

This section briefly summarizes the lesson, emphasizing particularly important concepts. It may refer you to other lessons or documents that might be helpful.

FORMAT OF EXAMPLES

Most examples are shown in two columns, the first indicating what you type and the second showing what you would then see in the command window. The command window is one of the areas of the screen on the display terminal; when you give a command, the command is displayed in this window along with feedback from NLS, such as prompts telling you what you can do next.

In either column of the example, when an entry occupies more than one line, all lines beyond the first are indented slightly to help you see that they are continuation lines. The line length shown depends on the dimensions of the column and may not be the same as what you actually observe at the terminal.

Sometimes the command window shows first one thing and then another, before you type again. In this case, both appear in the second column of the example, one after the other. Note, however, that when you give the command you may hardly notice what is shown first, because what follows it may appear immediately thereafter.

The examples show what you see if you are getting standard command feedback from NLS. You may ask to receive a different type of feedback, more suited to your needs; if you do this, you may see in your command window something slightly different from what is shown in the examples.

The examples often show that you type lowercase letters where you may actually type either lowercase or uppercase or any combination of the two. Where it matters which case you use, this will be noted in the discussion surrounding the example or will be clear from the context. For instance, when you type some text that you want to save, NLS accepts the text exactly as you type it, but when you type a letter of a word in a command, case is ignored.

NOTATION FOR SPECIAL CHARACTERS

The characters you type when you use NLS include not only letters of the alphabet and punctuation, but also some characters that you cannot see, such as spaces and special control characters. Where the NLS textbook shows what you type, you can usually tell where to type a space, such as between words in text. In some cases, however, it may not be clear that you type a space. In these cases, and wherever you are to type any other special character, the textbook uses the notations indicated in the table on the following page.

The first column of this table shows the notation used and the second column tells you what keys or mouse buttons you would press to produce the character represented by the notation. Key names are given here as they appear on the most recently designed NLS keyboard as well as how they may appear on other keyboards. Where applicable, the third column shows a control character that normally will have the same effect as pressing the indicated keys or buttons; note, however, that in some cases you may change the equivalent character from the one shown here.

Notation	Keys or Buttons	Equivalent
<BC>	BACK SPACE CHAR, BACK SPACE, or left mouse button	<CTRL-A> or <CTRL-H>
<BUG>	OK or right mouse button, after pointing with mouse	<CTRL-D>
<BW>	BACK SPACE WORD or left and middle mouse buttons	<CTRL-W>
<CD>	CMD DEL or middle mouse button	<CTRL-X>
<CTRL-x>	CTRL and x simultaneously, where x may be any letter	
<ESC>	ESC	
<HELP>	HELP	<CTRL-Q>
<INS>	INSERT or INSRT	<CTRL-E>
<LF>	LF or LINE FEED	<CTRL-J>
<LIT>	CTRL and V simultaneously	<CTRL-V>
<NULL>	CTRL and N simultaneously	<CTRL-N>
<OK>	OK or right mouse button	<CTRL-D>
<OPT>	OPT'N or OPTION	<CTRL-U>
<RC>	RPT CMD or middle and right mouse buttons	<CTRL-B>
<RET>	RETURN	<CTRL-M>
<SP>	Space	
<TAB>	TAB	<CTRL-I>

NOTATION FOR PROMPTS

When NLS is waiting for you to type something it prompts you with a symbol indicating the kind of thing it expects. For example, "C:" means that NLS expects you to enter a command word.

Notation	Meaning
A	Address
B	Bug - point to something
C	Command Word
L	Level Adjustment
T	Text or Typein
V	Viewspec
/	Alternative, for example A/B means Address or Bug
[ ]	Optional, for example [A] means that Address is an optional argument and that the <OPT> character must be entered before an Address can be entered.
**	Additional command words (usually optional) are available
...	NLS is working on your request.

BEGINNING USE OF NLS

BEGINNING USE OF NLS

This lesson discusses the basic concepts and commands you need to write and edit in NLS, that is, to enter information and make changes to it. Learning these concepts and commands is a vital step towards using the other tools in NLS. Before starting this lesson, you should know how to log in and enter NLS.

INTRODUCTION

NLS is a computer system designed to help people work with information. It includes a wide range of tools, from a simple set of commands for writing, reading, and printing documents to sophisticated methods for retrieving and communicating information.

Why use NLS?

Using NLS to write and edit has many advantages over working with paper, pencils, and typewriters. With NLS it is easier to organize your thoughts as well as your documents and to make changes to what you write. Special tools help you collaborate with other people, send messages, prepare correspondence, and so on. Some unique concepts, which you will soon learn, are the basis of new and effective methods of working.

You will begin by learning the basic writing and editing commands. Although this is only a small part of what NLS offers, you will be able to see some obvious advantages right away. For example, with a simple command, you can change a document by putting in a comma, taking out a phrase, or adding a paragraph, and NLS will automatically adjust the words and paragraphs so that they will look as if they were typed perfectly.

Display and mouse

Why use the display terminal and the mouse? When you try to talk to someone about a particular word on a piece of paper, you use either the simple method of pointing to it or the complex method of saying it is seven lines from the top and five words from the left. In NLS, you can use the mouse to point to what you are talking about. For example, if you are using NLS at a display work station and want to give a command to take out a particular comma displayed on the screen, you can simply use the mouse to point to the exact comma that you want NLS to remove. You can also use the mouse to point to words, sections of text, paragraphs or headings, and groups of paragraphs or headings.

The Base subsystem

NLS is divided into subsystems, which are sets of commands related to particular activities. Normally, the subsystem that is automatically available when you enter NLS is the Base subsystem. It includes the most common commands for doing your everyday work, such as reading, writing, editing, printing, and filing information. You will start learning NLS by working in Base.

VOCABULARY

Base subsystem: A basic set of commands for reading, writing, editing, printing, and controlling files.

<BC>: This stands for Backspace Character. Pressing either the appropriate key on the keyboard or the left mouse button deletes the last character you typed. You can also use <BC> to delete a <BUG> or any step in a command.

bug: To bug means to indicate a character on the screen by using the mouse to point at it and then typing <OK>.

<BUG>: This notation means that you are to bug a character on the screen.

bugmark: The mark displayed on the screen when you bug a character. The bugmark will be a highlighted character.

<BW>: This stands for Backspace Word. Pressing either the appropriate key on the keyboard or the left and middle mouse buttons deletes the last word you typed (plus any spaces, punctuation marks, or other characters following the word).

<CD>: This stands for Command Delete. Pressing either the appropriate key on the keyboard or the middle mouse button cancels any command you have not finished (that is, before you have given the final <OK>). You may then begin a new command.

character: A single letter, number, punctuation mark, space, return character, or special control character. You type characters when giving commands and you store characters in files.

command: An instruction you give to the computer to perform an action. When you give NLS a command, NLS will perform the action after you complete the command with a final <OK>.

command word: A word that NLS knows is part of a command, usually a verb or a noun.

command syntax: The general form of a command.

Create File command: A Base subsystem command that makes a new file in your directory; the file will have the name you specify.

Delete command: A Base subsystem command you can use to remove information, such as a character, a word, or some text.

file: An online work space, the computer's equivalent of a file folder in a filing cabinet, that you can fill with information that you type in or copy from another file. In NLS, you will always work with material in a file, whether you are reading, writing, or editing.

Insert command: A Base subsystem command that lets you add new information to a file, such as a character, a word, some text, or a statement.

invisible character: A character you cannot see on your screen, such as a space or a return character.

Jump Link command: An NLS command you can use to move from one file to another.

Move command: A Base subsystem command to reorder information in a file; for example, you can move one character to follow another.

noise word: When you type a command word, NLS may respond with a word or phrase in parentheses, called "noise words", to help you understand the purpose of the command or what you need to do to complete it.

<OK>: This notation means that you are to press either the appropriate key on the keyboard or the right mouse button, to tell NLS that you have finished giving a command or part of a command.

origin statement: The first statement in every file. It contains information such as the name of the directory, the name of the file, and the identity of the person who last updated the file.

prompt: A series of characters that appears in the command window to tell you what you can do next. Prompts are always one or more uppercase letters followed by a colon (:).

Replace command: A Base subsystem command to remove information, such as a character, a word, or some text, and put new information in its place.

<SP>: This stands for a space, that is, what you type with the space bar on the keyboard. In NLS, a space is an actual character that separates one word from another and that can be inserted, deleted, moved, or copied; it is not emptiness.

statement: The basic unit of information in an NLS file. A statement may be a single character, a word, a title, a heading, some text, or a paragraph. Every character in an NLS file is in a statement.

subsystem: NLS is divided into subsystems, which are sets of commands related to particular activities.

text: A series of adjacent characters, which may include punctuation and spaces, within a statement. A single character may also be considered "text".

Update New command: A Base subsystem command to consolidate recent changes into your file. You may erase changes made between updates with a command discussed in a forthcoming lesson.

visible character: A character you can see on your screen, such as a letter, number, or punctuation mark.

windows: The screen on the display terminal is divided into four areas, called "windows".

command window: When you use a command, the command is displayed in this window along with prompts and noise words. You also see the name of the subsystem you are working in.

file window: This window displays files or parts of files.

status window: This window displays messages to you from NLS or the Executive.

viewspec window: This small window displays characters that tell you what kind of view you have of the file being displayed.

word: A series of letters and/or numbers that are surrounded by spaces, punctuation marks, or any other characters that are not letters or numbers. NLS does not consider the surrounding characters as part of the word.

Note that apostrophy (') is considered as a punctuation mark so that "don't" is two words.

WINDOWS ON THE DISPLAY SCREEN

You start an NLS session by logging in and entering NLS. When you enter NLS, you will see information displayed on your screen. The screen is actually divided into four sections called "windows", and the information displayed in each window has an important function.

status window: This window displays messages to you from NLS or the Executive.

viewspec window: This small window displays characters that tell you what kind of view you have of the file being displayed. For basic information about viewing, see the lesson "Creating and Reading an Organized File".

command window: When you use a command, the command is displayed in this window along with prompts and noise words. You also see the name of the subsystem you are working in.

file window: This window displays files or parts of files; it is generally the largest window on the screen.

COMMANDS

NLS commands use simple English words and were carefully designed to make it easy for you to figure out what you can do next. When you understand the basic form of NLS commands, it will be much easier for you to learn new commands and new subsystems.

Command syntax

The general form of an NLS command is called "command syntax". The word "syntax" means "the arrangement of words in a sentence". NLS commands are like sentences in that they all have a similar form. Every command you will learn in this lesson begins with a verb followed by a noun; for example, the command verb "Move" will be followed by one of three nouns, Character, Text, or Word.

Command recognition

As you learn NLS, it will be extremely helpful for you to pay careful attention to your command window. NLS recognizes the letter "d" as the command word "Delete"; if you look at the command window as you type "d", you will see that when NLS recognizes a command word, it displays the entire word in the command window so you can make sure you gave the right command. Thus, when you type "d", you will see "Delete" in the command window. Most of the time, NLS will recognize a command word after you type the first letter; however, sometimes more than one command may start with the same letter, and you may have to type a space and then one or two letters before NLS will recognize the command.

Note that if you type more than is necessary, the other letters will be taken as specifying the next command word. It is important not to type ahead unless you are sure you know exactly what is necessary.

Prompts and Noise Words

Before you give a command and after you have given part of some commands, such as the verb "Delete" without a noun, you will see "C:" in the command window. "C:" is a prompt; a prompt is one or more uppercase letters, followed by a colon, that tell you what you can do next. In this case the "C" stands for "command word" and means you must type a command word. For example, after typing "d" for the verb "Delete", you may type "c" for the noun "Character"; NLS will recognize the command word after one letter and will then display a new prompt, "B/A:". A slash between letters in a prompt means that you have a choice. Further information about command words and prompts will appear throughout this and other lessons.

The word "at" in parentheses that follows the command word "Character" is called a "noise word". Noise words provide extra information to help you understand a command. In this case, the "(at)" means you now have to think about where the character is that you want to delete.

After typing "dc", you have in your command window a typical NLS command made up of a verb and a noun with prompts and noise words.

Canceling a Command

After you begin a command, you may change your mind or realize that you have made a typing error. To get rid of a command that you have started, use <CD>. After you type <CD>, the command window will display only the subsystem name and the first prompt, "C:", which tells you that NLS is again ready for you to begin a command. For example:

You type:	Command window shows:
d	BASE Delete C:
c	BASE Delete Character (at) B/A:
<CD>	BASE C:

You may also use <BC> to erase only the last command word that you typed.

MAKING A NEW WORK AREA: THE CREATE FILE COMMAND

This lesson teaches you how to write and edit a letter in NLS. To enter information in NLS and keep it separate from your initial file, you need to make a new file in which to store the information. To make a new file, use the Create File command. In general, when you create a file, you should name it something that will be easy to remember when you want to look at it in the future.

space before commands

Because more than one command in Base begins with "c", you have to type "<SP>cr" for NLS to recognize "Create"; then type "f" for "File". You will see the prompt "B/T/[A]:". Since "T" means you can give the name of the new file by typing it, you can then type the name. Follow the name by <OK> to tell NLS you have finished typing it. When you type <OK>, NLS will carry out the command. For example, this is how you would create a file named Dracula:

You type:	Command window shows:
<SP>cr	BASE Create C:
f	BASE Create File B/T/[A]:
dracula	BASE Create File dracula
<OK>	BASE C:

The origin statement

When you create a file, NLS assumes you want to work in it and automatically displays it. You will see a file that is empty except for a heading at the top of the file window. This heading is the origin statement of the file, and every file has one. The origin statement contains information about the file such as the name of the directory and the name of the file. It also contains the identity of the person who created the file or, if the file has been updated, the person who last updated it.

POINTING: BUGGING WITH THE MOUSE

To use the basic writing and editing commands in NLS, you must first learn how to point or "bug" with the mouse. Hold the mouse firmly but not tightly with your wrist on the table so you can touch the buttons on top of the mouse with your fingers. Roll the mouse across the table and watch the cursor (traveling mark) move correspondingly on the screen. To bug a specific character, move the cursor under it and type <OK> by pressing the right mouse button; we show this process as <BUG> in examples. You can bug a character whenever you see the letter "B" in a prompt. At all other times moving the cursor has no effect.

The character you bug will be replaced on the screen with a highlighted rectangle; this is called a "bugmark". If you accidentally try to bug an empty position (that is, where there is no character), NLS will mark a nearby character. It is good practice to look at the bugmark on your screen so that you can tell exactly what character you have bugged.

If you see that you have bugged the wrong character, you can use <BC> to erase the <BUG>, by pressing the left mouse button, and then bug the correct character.

WRITING: THE INSERT STATEMENT COMMAND

Using statements in NLS

You enter information in an NLS file by using the Insert Statement command. The statement is the basic unit of information in NLS. For example, if you write a letter in NLS, every section of the letter should be a separate statement; the salutation would be one statement, the paragraphs in the body would be separate statements, and the closing would be another statement. When you add information to a new file, you add it in the form of statements following the origin statement.

After making a file named Dracula with the Create File command, you could practice using the Insert Statement command by writing the letter from Dracula below.

My Friend,

Welcome to the Carpathains. I am expecting you anxiously. At three tomorrow the diligence will start for Bukovina; a place for you on it is kept. At the Borgo Pass, my carriage will be awaiting you and will bring you to me. I trust your journey from London has been a happy one, and that you will enjoy your visit in my beauteous land.

Your Friend

Dracula

Each section in the letter should be a separate statement. In the examples we show a blank line between statements to make clear the division of the text into statements. In these instructions the statements will appear on your screen without blank lines separating them. Type "is" to begin the Insert Statement command and then bug any character in the origin statement to show that you want your new statement to follow it. When you see the "L/T/[A]:" prompt, respond to the "T" choice by typing "My Friend,<OK>".

```
You type:        Command window shows:

i                BASE Insert C:
s                BASE Insert Statement (to follow) B/A:
<BUG>            BASE Insert Statement (to follow)  L:
My Friend,       BASE Insert Statement (to follow)  My
                  Friend,
<OK>             BASE C:
```

For the second statement, type "is" for "Insert Statement" and bug one of the characters in the new statement, so the paragraph will follow the salutation. When typing in the long paragraph, you do not have to type a return at the end of each line, since NLS will automatically continue onto the next line. Simply type all the words with the spaces between them. Try to type the paragraph exactly as it is shown; later in this lesson you will learn how to improve it by using editing commands. If you make mistakes while you are typing, you can use <BC> to erase the last character you typed and <BW> to erase the last word. If you type <BC> or <BW> several times, the last several characters or words you typed will be erased. Type <OK> when you have finished the paragraph.

You don't need carriage returns.

```
You type:       Command window shows:

i               BASE Insert C:
s               BASE Insert Statement (to follow) B/A:
<BUG>           BASE Insert Statement (to follow)  L:
<OK>            BASE Insert Statement (to follow) B/T/[A]:
Welcome...      BASE Insert Statement (to follow)  Welcome
 land.           to the Carpathains.  I am expecting you
                 anxiously.  At three tomorrow the diligence
                 will start for Bukovina; a place for you on
                 it is kept.  At the Borgo Pass, my carriage
                 will be awaiting you and will bring you to
                 me. I trust your journey from London has
                 been a happy one, and that you will enjoy
                 your visit in my beauteous land.
<OK>            BASE C:
```

Complete the letter by adding the last two statements (the closing and the name). Use the Insert Statement command and bug the statement that you want the new statement to follow. If you accidentally type the wrong command, use <CD> to cancel the command, and then start over.

EDITING TEXT WITH DELETE, MOVE, REPLACE, AND INSERT

After you enter information in an NLS file, you may want to correct errors or make changes to it. Delete, Move, and Replace are three important command verbs for editing text. The Insert verb, which you have learned to use to add statements to a file, also lets you add text within statements.

Deleting

"Delete" enables you to remove information from a file without leaving an empty space, as though the information you deleted was never there. For example, if you delete the second "r" in "perrfect", NLS will close up the empty space so the word will be "perfect".

Moving

"Move" allows you to reorder information in a file; NLS will adjust the text to compensate for the move. For example, if you moved the word "that" in "a phrase makes that sense" to follow the word "phrase", NLS would adjust the phrase to "a phrase that makes sense".

Replacing

"Replace" lets you remove information and put other information its place; that is, it combines Delete and Insert into one verb. If the new information you add is shorter or longer than the old information, NLS will compensate. For example, if you replace the word "good" with the word "terrific" in the phrase "a good sentence", NLS will adjust the text so you have "a terrific sentence".

Character, Text, and Word are three important command nouns for the parts of statements that you can delete, move, replace, or insert.

What Text is

"Character" is a single letter, number, punctuation mark, space, return character, or special control character. "Text" is any series of characters, which may include punctuation and spaces, within a statement. In a command, you point to text by bugging the beginning character and the ending character.

What a Word is

"Word" is any series of letters and/or numbers surrounded by spaces, punctuation marks, or any other characters that are not letters or numbers. (It does not have to be spelled correctly or mean something in English or any other language) NLS does not consider the surrounding characters as part of the word. In a command, you point to a word by bugging any character in the word. You may also bug the space between two words; NLS will consider the two words as one. Bugging between words is often very useful in editing, but be careful not to bug a space unless you want to affect both of the words around it.

Note that hyphenated words and contractions are not considered single words. For example, "inter-office" and "don't" are each considered to be two words.

How do you decide when to use which of these nouns? When you want to delete, move, replace, or insert a single letter, number, or punctuation mark, use "Character". When you want to delete, move, replace, or insert a series of characters, whether it is three characters within a long word or 25 characters that make up six words, use "Text". If you want to delete, move, replace, or insert one or two words surrounded by spaces or punctuation marks, use "Word" so NLS will know how to adjust the spacing.

Note that when new information is typed in (see T prompt) it can be more than one word or character even when one of those command nouns is used.

To help you practice using Delete, Move, Replace, and Insert with Character, Text, and Word, we have provided a list of corrections you can make to the letter from Dracula and have suggested a way of making each correction. For the first five corrections, we have shown the details of what you type and what you see on your screen when you use the command. After making the first five corrections, you should be familiar enough with the form of the commands to be able to finish the rest of the corrections without seeing the details.

Note that there is often more than one way to accomplish a task in NLS. In the following we show one way for each task.

1. Remove the comma that follows "Borgo Pass" by using the Delete Character command and bugging the comma. NLS will automatically close up the space between "Pass" and "my".

```
You type:      Command window shows:

d              BASE Delete C:
c              BASE Delete Character (at) B/A:
<BUG>          BASE Delete Character (at)  OK:
<OK>           BASE C:
```

2. Remove the word "be" between the words "will" and "awaiting". Use the Delete Word command and bug either character in the word. Notice how the space is adjusted when the word is deleted.

```
You type:      Command window shows:

d              BASE Delete C:
w              BASE Delete Word (at) B/A:
<BUG>          BASE Delete Word (at)  OK:
<OK>           BASE C:
```

3. Change the word "awaiting" to "await" by deleting the "ing". Use the Delete Text command. Bug the "i" as the first character of the text and then bug the "g" as the last character of the text.

```
You type:      Command window shows:

d              BASE Delete C:
t              BASE Delete Text (at) B/A:
<BUG>          BASE Delete Text (at) (through) B/A:
<BUG>          BASE Delete Text (at) (through)  OK:
<OK>           BASE C:
```

4. Correct the spelling of "Carpathains" to "Carpathians" by moving the third "a" to follow the "i". Use the Move Character command. Bug the third "a" as the character to be moved and bug the "i" as the character it should follow.

```
  You type:      Command window shows:

  m              BASE Move C:
  c              BASE Move Character (from) B/A/[T]:
  <BUG>          BASE Move Character (from)
                  (to follow) B/A:
  <BUG>          BASE Move Character (from)
                  (to follow)  OK:
  <OK>           BASE C:
```

5. Change the phrase "a place for you on it is kept" to "a place on it is kept for you". Use the Move Text command. For the text that you want to move, bug the space before "for you" as the first character of the text and the "u" in "you" as the last character of the text. For the character you want the text you are moving to follow, bug the "t" in "it".

```
  You type:      Command window shows:

  m              BASE Move C:
  t              BASE Move Text (from) B/A/[T]:
  <BUG>          BASE Move Text (from) (through) B/A/[T]:
  <BUG>          BASE Move Text (from) (through)
                  (to follow) B/A:
  <BUG>          BASE Move Text (from) (through)
                  (to follow)  OK:
  <OK>           BASE C:
```

6. Change the phrase "I am expecting you anxiously" to "I am anxiously expecting you". Use the Move Word command to move "anxiously" to follow "I am" by bugging any character in "anxiously" and then either character in "am".

7. Use the Replace Character command to replace "F" in "Your Friend" with "f" by bugging the "F" and typing or bugging any "f".

8. Change the word "beauteous" to "beautiful". Use the Replace Text command. Bug the "e" in "eous" as the first character of the text you want to replace and then the "s" as the last character of the text. Type the new text "iful" and then <OK>.

9. Use the Replace Word command to replace "visit" with "stay" by bugging any character in "visit" and typing "stay".

10. Add a comma after "Your friend" to make "Your friend,". Use the Insert Character command and bug the "d" as the character the new character should follow; then type or bug a

comma. Note that if you bug a comma as the character to be inserted, the comma that you bug is only copied. It will not be changed in any way.

11. Add a new sentence "Sleep well tonight." to follow the sentence "I am anxiously expecting you.". Use the Insert Text command and bug the period you want the new sentence to follow. Then type "<SP><SP>Sleep well tonight." and end with <OK>. You need to type the two spaces so there will be two spaces between the old sentence and the new sentence.

12. Change the phrase "I trust your journey" to "I trust that your journey". Use the Insert Word command and bug any character in the word "trust" as the word you want the new word to follow, then type "that" as the new word and end with <OK>.

If you see any other typing errors, use the editing commands you have just learned to correct them.

What to bug

Note that when you give a command to move or insert a character or some text, you tell NLS where to put the character or text by bugging the character it should follow; when you want to move or insert a word, you bug the word it should follow. You may also move or insert a word to follow a punctuation mark; simply bug the punctuation mark, and NLS will place the word after it, with a space as usual before the word.

UPDATING A FILE: THE UPDATE NEW COMMAND

Consolidating changes

Any time you have made a great many changes to a file or when you do not plan to work on it in the near future, you should update the file. In NLS, you can remove all the changes you made to a file since the last update; only when you update your file are the changes completely integrated into it.

You type:	Command window shows:
u	BASE Update C:
f	BASE Update File OK/C:
<OK>	BASE C:

READING A FILE: THE JUMP LINK COMMAND

Suppose you leave NLS after creating, editing, and updating a file as described in this lesson, and you later want to see the file again. The next time you enter NLS your initial file will be displayed; if you want to see a different file, you need to use a command to get to it. To reach any file in your directory, use the Jump Link command. Simply type "jl" and then type the name of the file, a comma, and then <OK>. For example, this is how you could see the file named Dracula after logging in and entering NLS:

```
  You type:      Command window shows:

  j              BASE Jump (to) B/C:
  l              BASE Jump (to) Link B/T/[A]:
  dracula,       BASE Jump (to) Link dracula,
  <OK>           BASE C:
```

EXERCISES

1. Write the following answer to Dracula's letter and keep it separate from any other information that is stored in your directory:

> Carpathians -- at last I've arrived! Thank you for your kind welcome.
>
> <your name>

Now add the statement "Dracula," before the first statement you added above.

2. Use a single command to make each of the following corrections to the letter you wrote in Exercise 1:

> (a) Add "The" before "Carpathians".
>
> (b) Change "at last I've arrived" to "I've arrived at last", without using the command word "Text". Now use the same command to change it back again.
>
> (c) Change the single space between the two sentences to two spaces, without typing a space or moving the cursor more than once.

SUMMARY

Several basic NLS concepts were introduced in this lesson: bugging with the mouse, using the verb-noun pattern to combine various command words, reading and responding to prompts, and typing only enough characters in a command word for NLS to recognize the command. These features are consistent in NLS. The mouse is used for all work in display NLS, and all the NLS subsystems use the same basic syntax for commands. Learning these elements will make it easy to learn the rest of NLS.

This lesson has also taught you some basic commands for writing and editing in NLS. You should now know how to create a file, add statements to it, and edit a statement in various ways such as removing or adding a character or a word. You have learned how to update a file you have worked on and how to see it again the next time you enter NLS. We recommend that you now read the lesson "Creating and Reading an Organized File".

SOLUTIONS TO EXERCISES

1. Use the Create File command and give the file a name that is different from the name of any other file in your directory. Using the Insert Statement command and bugging the origin statement of the new file, type in the statement that begins with "Carpathians". Then use the Insert Statement command again, bugging the statement you just added and typing in the statement consisting of your name. To add a statement before the first statement you added, you must do exactly what you did to enter that statement, that is, use the Insert Statement command and bug the origin statement; type "Dracula," as the text of this statement.

2. (a) The simplest way to do this is to give the Replace Character command and replace the "C" by "The C".

(b) Use the Move Word command. Bug the space between "at" and "last" to indicate the word to moved, and bug any character in "arrived" to indicate the word it should follow. To change it back again, bug the space between "at" and "last" and then bug the second "-". Note that if you bug the space between "I've" and "arrived", you will be referring to "ve arrived", because the word ends at the punctuation mark.

(c) Use the Insert Character command and bug the space following the first sentence as both the character to insert and the character it should follow.

GETTING INFORMATION AND SERVICES

GETTING INFORMATION AND SERVICES

This lesson teaches you how to learn about NLS by asking questions of the system and how to request assistance and services from system staff personnel by using message sending facilities. To understand this lesson, you should know how to log in and enter NLS and should have some working experience with elementary NLS commands, described in the lesson "Beginning Use of NLS".

INTRODUCTION

NLS was specially designed so that you can learn as you work. NLS commands use simple English words, and their basic, consistent form makes it easy for you to figure out what you can do next. Almost all commands begin with a verb followed by a noun, and prompts and noise words give you general information about what you can do next. However, there are many times when you will need more information. For example, when you see a "C:" prompt, you know that you should type a command word--but what command word? Or perhaps you know there is a Transpose command, but you are not sure what it does. You can use NLS while you are working online to find out answers to questions like these as well as more general questions about NLS terms and procedures.

Of course, there will inevitably be a time when you have a question that NLS cannot answer or when you are not even sure what kind of problem you have, and what you would really like to do is contact some person who is knowledgeable about NLS. The ACTION service, described in this lesson, provides an easy, effective method for getting advice or help from the system staff. You can also use ACTION to register a complaint, a suggestion, or a compliment.

VOCABULARY

ACTION: A service provided by the system staff that enables users to request assistance and register complaints, suggestions, or compliments.

Help command: An NLS command to get information about commands, terms, and procedures.

menu item: A subtopic, to guide you to related information, listed under a Help description.

<NULL>: This notation represents a special character that means "nothing" or "none".

question mark (?): When typed after any prompt, question mark will show you what you can do next.

Message subsystem: The subsystem you can use to send messages and documents to other users.

LEARNING WHAT YOU CAN DO NEXT: QUESTION MARK

The simplest way to learn about NLS while you work is by using question mark. Any time you are working with NLS (except in the middle of typing in text), you can type a question mark (?) to see a list of all the things you can do next.

? after C:

When typed after a "C:" prompt, question mark shows you all the command words that you can use at that point. For example, if you are working in the Base subsystem and you type "?" after "BASE C:", NLS will list all the command words that begin Base subsystem commands. One of these words is "Insert". If you type "i" to begin an Insert command, you will see another "C:" prompt; if you then type "?", NLS will list the command words that can follow Insert, such as Character, Word, and so on.

You will observe that as question mark displays a list, your command window expands into your file window. You will temporarily be unable to see what is at the beginning of your file window; however, the information in the file will not be affected.

After you have studied the list of command words, you can type a character to begin one of them; your command window and file window will return to their previous sizes and NLS will continue as if you had not used question mark.

<> before a command word

In the following example, notice that "<>" precedes some command words. This means that you must type a space before you begin that word.

```
  You type:      Command window shows:

  i              BASE Insert C:
  ?              BASE Insert ?
                  Current alternatives are:
                    Branch        Link       <>Time
                    Character     Number       Visible
                    Date          Plex         Word
                    Edge        <>Sendmail     <CTRL-Q>: HELP
                    Group         Statement    <CTRL-S>: SYNTAX
                    Invisible     Text
                    ---
  w              BASE Insert Word (to follow) B/A:
```

Removing the list

After you have seen the list of command words, you may want to remove it from your screen before continuing. To do this, type a character that does not begin a command word and NLS will return you to where you were before you used question mark. If instead you choose not to continue, you can type <CD> to cancel the command.

At some steps in commands, NLS is waiting for you to bug something on the screen with your mouse, to type in some text, or to give the location of something, such as a file; that is, the next step is not a command word. In this case, question mark will show you what NLS expects by listing instructions that explain your choices. You can then follow one of the instructions or type <CD> to cancel the command. For example:

```
  You type:      Command window shows:

  i              BASE Insert C:
  w              BASE Insert Word (to follow) B/A:
  ?              BASE Insert Word (to follow) ?
                  Please specify a WORD by
                    BUG or ADDRESS
                    or <CTRL-Q> for HELP, or <CTRL-S>

                    for  Type <OK> to continue.
  <CD>           BASE C:
```

Question mark is useful for reminding yourself about rarely used commands as well as for learning entirely new ones. You can combine the information you get from noise words, prompts, and question mark to move from one step in a command to the next.

? as text

NOTE: If you are typing in text that has a "?" within it, there is no problem. However, if you want to type text that begins with "?" (for example, if you want to insert a question mark at the end of a sentence with the Insert Character command), NLS will respond by listing your current alternatives. When you want "?" to be taken as text, type <LIT> before the question mark.

When NLS says ?

If you enter a command letter which does not match any of the current alternatives, NLS will echo a "?". In such a case simply enter the correct alternative and continue.

COMPLETE INFORMATION ABOUT NLS: THE HELP COMMAND

What you can learn from Help

If you are not satisfied with what you can learn with question mark, you can use the Help command while you are working online to get the most detailed, up-to-date information on NLS that is available. Without interrupting your work in progress, you can look up a definition of any NLS command or term, or a description of NLS procedures with advice on how to accomplish a particular task. You can use Help to find the answers to specific questions, or you can just browse through a general subject area.

How to Find What You Are Looking For

To learn more about NLS, begin by typing "h" for Help. When you see the "OK/T/[A]:" prompt, you can type a specific term or command for Help to look up or, if you don't have a specific term in mind, you can simply type <OK>.

Typing terms for Help to look up

To look up a specific term or command, type it after the "OK/T/[A]:" prompt and then type <OK>. For example, you can look up a term by typing "statement" followed by <OK> or a command by typing "insert statement" followed by <OK>. You may type the terms in uppercase or lowercase, and they may be one word or a number of words separated by spaces.

As Help looks up a term, you will see the message "(Searching HELP file)" in your command window. Depending on where the information is stored, you may also see the message "searching index" in your status window. When Help finds the definition, the message "(Searching HELP file)" will clear, and you will see a "</T:" prompt. So that your work in progress will not be disturbed, Help will create a special window below the command window to display the definition to you; when you have finished using Help, your file window will look as it did before you gave the Help command. Note also that Help temporarily changes your viewspecs; do not be disturbed if you see changes in your viewspec window while you are using Help.

Leaving Help

To return to your work in progress after reading one or more definitions, simply type <CD> to end the Help command.

The following example shows how you could use Help to find the definition of "question mark" while you work in the Base subsystem.

```
You type:           Command window shows:

h                   BASE Help  OK/T/[A]:

question mark<OK>    BASE Help  question mark
                    (Searching HELP file)...
                    </T:

                     question

                     Questionmark:  (?)
                       Typing a question mark at any point in
                     an NLS command will show you the NLS
                     command alternatives available at that
                     point.  After the list has printed
                     you can go on as if you had not typed
                     questionmark.  See also:  CTRL-Q, help.

<CD>                BASE C:
```

NOTE: The term that is defined may not always be exactly the same one that you typed. This is because the Help information is arranged information so that if you type a term that has a similar meaning or spelling as a term known to Help, you will get the definition for the similar term (as in the example above, where you typed "question mark" and got the definition for "questionmark").

When Help doesn't find a term

If the term you type is not like any term defined in Help or if you make a typing error so that Help does not recognize it, you will see the term, as you typed it, followed by a question mark in your status window. You can then try typing a different term or the correct spelling after the "</T:" prompt.

Learning about a subsystem

When you type only <OK> after the "OK/T/[A]:" prompt that you see when you give the Help command, Help displays a definition of the subsystem that you have been working in. This is to provide you with general information and a list of topics to guide you to the particular information you need. For example, if you were working in the Base subsystem and typed "h" and then <OK>, you would see this:

```
You type:       Command window shows:

h               BASE Help  OK/T/[A]:

<OK>            BASE Help
                (Searching HELP file)...
                </T:

                 Base Subsystem
                   Base is the home subsystem in NLS.
                 It has commands that allow you to read,
                 write, and modify information online and
                 output it to hardcopy, among other things.
                    1. how to use the Base subsystem
                    2. commands in Base
```

How to Read a Help Description

The descriptions that Help provides are one or two paragraphs, short enough to fit on your display screen. The paragraphs are written in simple, nontechnical language. They cover as much information as possible and indicate related subjects and commands for you to look up.

Definitions of commands

Help includes a definition of every NLS command. The form of these definitions is always the same. The first line of the definition shows the syntax of the command, followed by a functional description of the command. If you do not know how to read command syntax, use Help to look up the syntax terms, such as SOURCE, DESTINATION, and CONTENT. Some commands have unique syntax terms that represent choices special to that command, and these are defined below the definition of the command. If you look up a command verb that can be followed by any of several command nouns, you will see a list of the various combinations of verb and noun and you can choose an appropriate definition.

Menu

Every Help description that has relevant subtopics will have a list below it, called a "menu". For example, the definition of the Base subsystem shown in the example above has a menu with two items, "1. how to use the Base subsystem" and "2. commands in Base". To see more information about a subject presented in one of the menu items, simply type its number followed by <OK>.

Long menus

When a menu is too long to fit on the screen, you will see the message "(do you want to see the rest of the menu) Y/N:", and you can read the rest of the menu by typing "y" for "yes". You should jot down the number of any menu items that interest you, before continuing to the rest of the menu. You can ask to see an item from anywhere in the menu, even if it is not being displayed. If you do not wish to read the rest of the menu, type "n" for "no". You may then type the number of one of the menu items, type an entirely new term for Help to look up, or type <CD> to return to your work in progress.

See and See also

Besides including menus of subtopics, many Help descriptions contain suggestions of related concepts and commands, with "See" and "See also" followed by a list of terms. The terms are separated by commas; when two or more words are separated only by spaces, they should be considered a single term. To see more about any of these terms, simply type the term after the "</T:" prompt and end with <OK>.

The following example shows how you could use Help to look up a term, choose a menu item to get more information, look up a related term, and then return to your work in progress.

```
  You type:    Command window shows:

  h            BASE Help  OK/T/[A]:
  subsystem<OK> BASE Help > OK/T/[A]: subsystem
               (Searching HELP file)
               (HELP) < >/T:
               SUBSYSTEMS IN NLS
                  When SUBSYSTEM is found in a command
               syntax expression, you may use a
               subsystem name as a command word.
               The following is a list of all NLS
               subsystems.  Some of them are automatically
               loaded for you when you enter NLS; others
               are easily loaded by the user.  Otherwise,
               all NLS subsystems work in the same way.
               See going, or the individual subystem,
               below, for more information.

               1. AFMFormat:  formatting Air
                  Force manuals
               2. Base:  reading, writing,
                  modifying, filing, and printing
               3. Calculator:  doing simple
                  arithmetic
               4. Decimal:  formatting files in
                  the Air Force decimal format
               5. Format:  providing aids for
                  using Output Processor
                  directives
```

```
                6. Graphics:  creating and
                   modifying diagrams
                7. Hyphen:  adding words to the
                   hyphenation dictionary
                8. Letter:  formatting a business
                   letter
                9. Message:  interface to SNDMSG
               10. Modify:  special purpose
                   editing commands
               11. Programs:  compiling, loading,
                   debugging, managing, and adding
                   tools to the AKW
             (do you want to see the rest of the
             menu?) > Y/N:
y                 Yes
             HELP > </T:
               12. Proof:  checking COM formats on
                   a Tektronix screen
               13. Publish:  generating sections
                   of documents
               14. Sendmail:  locating current
                   idents
               15. Executive System:  file
                   handling, other tools
2<OK>             2
                (HELP) > </T:

                   Base:  reading, writing, modifying, filing,
                     and printing
                  By default, you are in the Base subsystem
                  when you enter NLS.

                  Base subsystem
                     Base is the home subsystem in NLS.  It has
                   commands that allow you to read, write, and
                   modify information online and output it to
                   hardcopy, among other things.
message<OK>     message
                (Searching HELP file)
                (HELP) > </T:
                message - one of the following:
                   1. a message sent through the NLS:
                      Sendmail system
                   2. a message sent through the
                      Executive System (TENEX or
                      TOPS-20) SNDMSG
                   3. one's file of Executive System
```

```
                    (TENEX or TOPS-20) messages
                 4. the Message Subsystem for handling
                    SNDMSGes via NLS
  <CD>          BASE C:
```

< and ^

NOTE: If you type question mark after the "</T:" prompt, you will see that two of your alternatives are left angle bracket (<) and up arrow (^). Typing a left angle bracket followed by <OK> will recall the previous display of information. Typing an up arrow followed by <OK> will display the information one level up in the outline of Help information.

Why use Help?

Many people find that working with Help is awkward at first--they are used to turning to a person or a familiar reference manual when they need assistance. The experience of using Help is different from thumbing through a traditional reference manual. The response is sometimes rather slow because Help has to search through a great deal of information, and, of course, Help is in no way as flexible as a person. However, it is worthwhile to work with Help. With practice, using Help becomes easy and comfortable and gives you access to a great wealth of information without your ever having to leave the terminal. The ability to use Help will make you a more self-sufficent NLS user; you will be able to find answers when you need them and take on new tasks with more confidence.

You may occasionally have trouble getting an answer that is easy for you to understand, because there are many different kinds of NLS users; some users know a lot about computers, whereas many are using them for the first time. We have tried to anticipate the terms you need, but we have not always succeeded. If you have suggestions for terms, please send them to Feedback by the method described below.

COMMUNICATING WITH PEOPLE: ACTION

You can send a telegram-like message to the system staff at any time to ask a question, ask for assistance, or offer suggestions, complaints, or compliments. This service is called ACTION, which accepts all messages and returns an answer.

What ACTION does

When questions reach ACTION, an system staff member answers them. When ACTION receives a request for services a system staff member performs the services and informs the requestor of the outcome or explains why the service was not performed. Reports of bugs are passed on to programmers and the planned disposition reported back to the user. Suggestions are sent to the appropriate people and acknowledgments are sent to the user.

Sending a Message to Action: The Message Subsystem

To send a message to Action, you can use the Message subsystem. The Message Subsystem enables you to handle Executive System (TENEX or TOPS-20) SNDMSG communications through NLS. With this subsystem you can move your messages into NLS, sort messages, and automatically send messages through SNDMSG via NLS. To reach the Message subsystem, type "e" for "Execute" then "p" for "Program". This will allow you to load the message program by typing "l" for load and "p" for "Program" followed by "message." Terminate your command by typing <OK>. After the system messages "Loading User Program" and "Subsystem MESSAGE Now Available (Attached)" appear you can type "g" for "Goto" and then "m" for "Message" followed by <OK>. "MESSAGE C:" will appear in your command window when Message is ready for a command.

```
   You type:       Command window shows:

   e               BASE Execute (command in) C:
   p               BASE Execute (command in) Programs
                   PROGRAMS C:
   l               PROGRAMS Load C:
   p               PROGRAMS Load Program B/T/[A]:
   message<OK>     PROGRAMS Load Program message > ...
                   BASE C:
   gm<OK>          BASE Goto (subsystem) Message
                   MESSAGE C:
```

The Message command "Send" prompts you to provide information for a message, and then sends an Executive System SNDMSG to the people whose usernames or idents you specify. To send a branch of the NLS file you have loaded to Action you type "s" for "Send" then "b" for "Branch". The network address ACTION should be specified following the prompt: (To:). The Message system will prompt you for a carbon copy list and a title. The message system wil verify the distribution list and give you the opportunity to add to the list before sending the mail. Note that the command pays attention to viewspecs.

```
  You type:          Command window shows:

  s                  MESSAGE Send C:
  b                  MESSAGE Send Branch (at) B/A:
  <BUG>              MESSAGE Send Branch (at) > V:
  <OK>               (To:) B/T/A:
  action<OK>         (To:)action
                     (Cc:) B/T/A:
  <NULL>             (Subject:) B/T/A:
  Help<OK>           (Subject:)  B/T/A: Help
                     verifyng distribution list
                     To:     ACTION@
                     Type <OK> to continue.
                     (Send the message now?(Type N to add to
                     list)) Y/N:
  y                  Yes ^ ...
                     ACTION@
                     Delivered.
                     MESSAGE C:
  q<OK>              MESSAGE Quit OK/C:
                     BASE C:
```

Sending your message is equivalent to dropping a letter in a mail box, not to delivery. The Execuive mailer checks for messages frequently but the mail may not reach Action until shortly after it was sent.

The Message command "Copy" copies your MAIL.TXT file to the address you point to. It changes the format to make the messages easy to handle in NLS.

```
  You type:          Command window shows:

  c                  MESSAGE Copy C:
  m<OK>              MESSAGE Copy Message (File) OK/T/[A]:
  <OK>               MESSAGE Copy Message (File)
                     "(MAIL.TXT)" (to follow) B/A:
  <BUG>              Copy Message (FILE) "(MAIL>TXT)" L:
  <OK>               MESSAGE C:
```

For further information on the message subsystem see <userguides,message,>.

EXERCISES

1. How can you find out what you can delete?

2. The word "Visible" may appear when you use question mark; how do you find out what it means?

3. How would you ask to have your password changed?

SUMMARY

There are several ways to get information or assistance when you are using NLS. At any point in a command, you may type a question mark and NLS will list your present alternatives. The Help command enables you to ask for definitions of terms and other information. You may easily send a brief message or question to ACTION with the Message subsystem, and you will get an answer.

NLS is a large system that includes many powerful tools and techniques. Question mark, Help, and ACTION enable you to learn about NLS as you work and to become a self-sufficient and creative user. Even if you know enough to get your work done, you can always learn new commands and more effective methods, as well as surprising tricks--this means that you can use NLS for a wide range of tasks as well as satisfy your sense of curiosity and play.

SOLUTIONS TO EXERCISES

1. At "BASE C:", type "d" for "Delete" and then type a question mark. You can then use one of the command nouns or type <CD> to cancel the command.

2. At "BASE C:", type "h" for "Help" and then type "visible", followed by <OK>. When you have finished reading the definition, type <CD>.

3. Use the Goto command to reach the Message subsystem. When in Message, use the Send command, as described in this lesson, to send a message with a title of your choice and the text from an NLS file as the message. Use the Quit command to return to the Base subsystem.

CREATING AND READING AN ORGANIZED FILE

CREATING AND READING AN ORGANIZED FILE

This lesson teaches you how to create and read an organized file in NLS with the Base subsystem. You should already know how to log in and create simple NLS files.

INTRODUCTION

In this lesson, you will learn how to write a document in NLS and then read it. Writing in NLS means adding text to a file by typing at your keyboard or by copying text that is already stored in the computer. A valuable difference between writing documents in NLS and writing them on typewriters is the unique way that text in NLS files can be structured. The paragraphs and headings are organized into an outline that tells NLS the relationships between the statements. For example, the first chapter title would be at position 1 in the outline, the first subheading in that chapter would be at position 1a, and the first paragraph following that subheading would be at position 1a1. This outline form is called "hierarchical structure".

This lesson will teach you more about hierarchical structure. You will see how it helps you organize your thoughts and your writing, and how it makes it possible for you to look at your online document from many different points of view.

Reading in NLS is much more flexible than reading text that is printed in a book. With a book, you can either read each section straight through, from beginning to end, or you can scan quickly across pages, picking out a sentence here and there. In NLS you can read the paragraphs on your screen, one after the other, but there are also many different and useful ways to look at an online document. For example, you can see only the headings, or only the headings and the first line of every paragraph; the other lines will still be there, but you do not see them until you need them.

VOCABULARY

Back: A command word that refers to the statement immediately before the statement you indicate, regardless of level.

hierarchical structure: The structure of NLS files; an outline form that shows the relationship between the statements.

<INS>: This stands for Insert. Pressing the appropriate key on the keyboard puts you in insert mode.

insert mode: You enter this mode when you type <INS> in place of the final <OK> at the end of the Insert Statement command. In insert mode, you can continually add statements until you type <CD>.

Insert Statement command: A Base subsystem command that lets you add statements to a file.

Jump command: An NLS command to move from one point in a file to another, or from one file to another.

level: A number that indicates how far up or down a statement is in the hierarchical structure of a file.

level clipping: Using viewspecs to display a limited number of levels in a file.

line clipping: Using viewspecs to display a limited number of lines for each statement.

Next: A command word that refers to the statement immediately following the statement you indicate, regardless of level.

origin statement: The first statement in every file. When you create a file, NLS automatically makes an origin statement containing the name of the file and other pertinent information. The origin statement is at level 0, that is, one level above the first-level statements; it is the only statement that can be at this level.

predecessor: The predecessor of a statement is the preceding statement that is at the same level and has the same upstatement. Not every statement has a predecessor.

statement number: A series of alternating numbers and letters that indicates the exact position of a statement within the hierarchical structure of a file.

structure: The arrangement of statements in a file. NLS files have a hierarchical structure.

substatement: A substatement of a statement is any statement that is one level below it. Not every statement has a substatement.

substructure: All of the statements one or more levels below a particular statement. Not every statement has substructure.

successor: The successor of a statement is the next statement that is at the same level and has the same upstatement. Not every statement has a successor.

upstatement: The upstatement of a statement is the statement that is one level above it. Every statement except the origin statement has an upstatement.

viewspecs: Single-letter specifications of how you see your file. For example, with one viewspec you will see blank lines between statements, and with another you will see the statements without blank lines between them.

ORGANIZED FILES: HIERARCHICAL STRUCTURE

Humankind has classified most of the things in the world, including the world itself, into hierarchies. The world is divided into continents, the continents are divided into countries, countries are divided into states, states into counties, and so on. When we refer to hierarchical structure in NLS files, we mean the arrangement of statements into an outline form that shows the relationships between the statements. This is like any natural hierarchy, because you arrange the statements in the outline according to their content.

We can use the example of the world to show how you would express a natural hierarchy with NLS structure. Each item in the following outline is a separate statement. Note that although each statement shown here is only one line long, a statement in a hierarchy can also be an entire paragraph. NLS uses identing to show the relationships between the statements. Each statement that is "below" another statement in the hierarchy is indented three character positions from that statement.

```
World

     Africa

        Ethiopia

        Kenya

        Zambia

     Antarctica

     Asia

        India

        Japan

        Thailand

     Australia

        Australia

        New Zealand
```

Europe

  Austria

  Ireland

  Spain

North America

  Canada

  Cuba

  Mexico

South America

  Argentina

  Brazil

  Chile

Level

We could fill in this outline by including all the countries under each continent and then adding states under countries, counties under states, and so on, until we reached individual buildings or people in them. The hierarchy in the example has three levels; the World statement is at level 1, the continents are one level below it, at level 2, and the countries are at level 3. If we added states under countries, the hierarchy would then have four levels.

Substatement and upstatement

Two useful NLS terms for describing the relationships between statements in a file are "substatement" and "upstatement". A substatement of a statement is any statement that is one level below it. In the preceding example, Ethiopia, Kenya, and Zambia are all substatements of Africa, and Antarctica has no substatements. The upstatement of a statement is the statement that is one level above it. For example, the upstatement for Ethiopia, Kenya, or Zambia is Africa, and the upstatement for Antarctica is World.

Substructure

We use the term "substructure" to refer to all of the statements that are one or more levels below a particular statement. In the preceding example, all of the continents and the countries below each continent are the substructure of the World statement. The two countries Australia and New Zealand, below the continent Australia, are the substructure of

Australia. Since Antarctica has no countries below it, it has no substructure, and since none of the countries have statements below them, none of them have substructure.

This lesson shows how you would create a table of contents, another familiar form of hierarchical structure. The table of contents structure is especially appropriate because many people use NLS for producing hierarchically structured documents. To create a file named Gulliver containing the partial table of contents shown on the following page, you would first give the Create File command and type "gulliver" as the name of the file. When you do this, NLS automatically makes an origin statement containing information about the file. You would then be ready to insert the table of contents following the origin statement.

A Letter from Capt. Gulliver to his Cousin Sympson

The Publisher to the Reader

The Contents

Part I. A Voyage to Lilliput

Chap. II. The Emperor of Lilliput, attended by several of the nobility, comes to see the author in his confinement.

Chap. III. The author diverts the Emperor and his nobility of both sexes, in a very uncommon manner.

Part II. A Voyage to Brobdingnag

Chap. I. A great storm described, the long-boat sent to fetch water, the author goes with it to discover the country. A description of the inhabitants.

Chap. II. A description of the farmer's daughter. The author carried to a market-town, and then to the metropolis.

Chap. III. The author sent for to court. The Queen buys him of his master the farmer and presents him to the King.

Part III. A Voyage to Laputa, Balnibarbi, Glubbdubdrib, Luggnagg, and Japan

Chap. I. The author sets out on his third voyage, is taken by pirates. He is received into Laputa.

Chap. II. The humours and dispositions of the Laputians described.

Chap. III. A phenomenon solved by modern philosophy and astronomy. The Laputians' great improvements in the latter.

Notes and Comments

WRITING: INSERT MODE

When you want to type a series of statements such as the table of contents on the preceding page, with each new statement directly following the last one, use insert mode to enter the statements. Once you are in insert mode, you simply type a statement and end it with <OK>, then type the next statement and end that with <OK>, and so on. Each statement will follow the last one you typed. To enter insert mode, type <INS> in place of the final <OK> at the end of the Insert Statement command that you use to enter the first statement. To leave insert mode, type <CD> just as you would to cancel any other command.

When you use the Insert Statement command, you see the prompt "L:"; the "L" in this prompt means that you can indicate the relative level of the statement you are inserting, that is, its level relative to the statement it follows. This enables you to arrange the statements into a hierarchy as you type them. Type "d<OK>" after the prompt if you want the statement to be down one level from the one it follows or "u<OK>" if you want it to be up one level. If you want the statement to be up more than one level, simply type another "u" for each level; for example, "uuu<OK>" would make the statement be up three levels from the statement it follows. If you want the statement to be at the same level as the statement it follows, type <OK> after the "L:" prompt or simply begin typing the statement.

NOTE: You cannot insert a statement to be down more than one level from the one it follows.

Inserting to follow the origin statement

When you begin inserting statements into a newly created file, the first statement that you type will follow the origin statement. The origin statement is the upstatement of all the first-level statements that you add to the file; in other words, the origin statement is at level 0 and all of the other statements in the file are the substructure of the origin statement. However, you do not need to indicate the level of the statement that follows the origin statement; since any statement that you insert to follow the origin statement must be at level 1, NLS automatically assumes level 1 in this case.

After giving the Create File command to create a file named Gulliver, you could use insert mode to add to it the table of contents, adjusting the level of each statement after the "L/T/[A]:" prompt when necessary. Entering the first two statements would look like this:

```
You type:         Command window shows:

i                 BASE Insert C:
s                 BASE Insert Statement (to follow) B/A:
<BUG>             BASE Insert Statement (to follow)  L:
<OK>              BASE Insert Statement (to follow) B/T/[A]:
A Letter...       BASE Insert Statement (to follow)  L:
                   A Letter from Capt. Gulliver
 Sympson           to his Cousin Sympson
<INS>             L:
<OK>              B/T/[A]:
The Publisher     The Publisher to the Reader
 ...Reader<OK>    L:
<OK>              B/T/[A]:
```

Entering the next three statements would look like this:

```
You type:         Command window shows:

The Contents      The Contents
 <OK>             L:
d<OK>             "d" B/T/[A]:
Part I...         Part I.  A Voyage to Lilliput
 <OK>             L:
d<OK>             "d" B/T/[A]:
Chap. II...       Chap. II.  The Emperor of Lilliput, attended
 <OK>              by several of the nobility, comes to see
                   the author in his confinement.
                  L:
```

You could continue in this way to enter the entire table of contents; however, you would not see all of it in the file window. It is important for you to realize that as long as you type each statement and follow it with <OK>, it will be added to your file even though you cannot see it being added. Of course, you can always see the statement that you are typing in the command window.

Notice that since "Notes and Comments" is at the highest level in the hierarchy, it is two levels up from the statement it follows (Chap. III of Part III). Thus, to add the statement "Notes and Comments" two levels up from the statement it follows, you would type "uu<OK>" after the "L:" prompt. Then, after inserting that final statement, you would type <CD> after the "L:" prompt to leave insert mode.

INSERTING INDIVIDUAL STATEMENTS

After using insert mode to type a series of statements into a file, you may find that you want to add statements to what you typed. You can use the Insert Statement command to add one or more statements anywhere in a file. Just indicate which statement you want the new statement or statements to follow. To add one statement, end the Insert Statement command with <OK>; to add more than one statement, enter insert mode as usual by typing <INS> in place of the final <OK>. For example, to add Chap. I to Part I in the table of contents, you could use the Insert Statement command, bug the Part I statement, and add the Chap. I statement to follow it, down one level.

```
You type:       Command window shows:

i               BASE Insert C:
s               BASE Insert Statement (to follow) B/A:
<BUG>           BASE Insert Statement (to follow)  L:
d<OK>           BASE Insert Statement (to follow) "d"
                 B/T/[A]:
Chap. I...      BASE Insert Statement (to follow) "d" Chap.
                 I. The author is shipwrecked and swims for
                 his life; gets safe on shore in the country
                 of Lilliput, is made a prisoner, and carried
                 up the country.
<OK>            BASE C:
```

Inserting to follow a statement with substructure

When you insert a statement to follow another at the same level and the statement it is to follow has substructure, the statement you are inserting will come after the substructure. For example, if you added a Part IV statement to the table of contents and indicated that it should follow Part III at the same level, Part IV would come after Chap. III of Part III, one level higher than Chap. III and at the same level as Part III. Note that there is often more than one way to add a statement at a particular position in a hierarchically structured file. For example, if you inserted a Part IV statement to follow Chap. III and indicated that it should be up one level, the result would be the same as if you inserted it to follow Part III at the same level.

READING: THE JUMP COMMAND

If you have added statements to a file at a place that is not displayed on your screen, you will probably want to look at those statements. To read part of an NLS file that is not currently displayed on your screen, you can use one of the many forms of the Jump command to find the statement you want to read and display it at the top of the file window. The NLS command verb for reading is called "Jump" because you can leap from one statement to any other without displaying any of the statements between them. When you use the Jump command verb, you will see a prompt "B/C:" that means you have to tell what point in a file you want to jump to by bugging something or by using a command word.

Jumping to a statement you bug

The simplest way to indicate which statement you want to jump to is to bug it. This allows you to read more of what follows. After you bug the statement, you will see a "V:" prompt, which means that you may change viewspecs (as described later in this lesson). To keep the same view, simply type <OK>. The statement you bugged will then move to the top of your file window and you will see the next series of statements that follow it. For example, after inserting the table of contents into the Gulliver file, you could read beyond what shows on your screen by using Jump and bugging any character in the last statement in your file window.

You type:	Command window shows:
j	BASE Jump (to) B/C:
<BUG>	BASE Jump (to) V:
<OK>	BASE C:

Jump Origin

After reading the entire table of contents, you could return to the beginning of the file by using the Jump Origin command. This command displays the origin statement of your file at the top of your file window.

You type:	Command window shows:
j	BASE Jump (to) B/C:
o	BASE Jump (to) Origin B/A:
<BUG>	BASE Jump (to) Origin V:
<OK>	BASE C:

JUMPING WITH STRUCTURE: NEXT, BACK, SUCCESSOR, AND PREDECESSOR

Four important command words that you can use with the verb "Jump" are Next, Back, Successor, and Predecessor. Like "upstatement" and "substatement", these terms describe structural relationships between statements in an NLS file.

Next and Back

"Next" is the statement immediately following the statement that you indicate and "Back" is the statement immediately before the one you indicate. These command words can be used with "Jump" to display a statement not currently in the file window. For example, to see the statement following the last statement in your file window, you can give the Jump Next command and bug the statement. Since there is another command word that can follow Jump and begins with "n", you must type "<SP>n" for "Next".

You type:	Command window shows:
j	BASE Jump (to) B/C:
<SP>n	BASE Jump (to) Next B/A:
<BUG>	BASE Jump (to) Next V:
<OK>	BASE C:

If you then give the Jump Back command and bug the statement at the top of your file window, you will see the immediately preceding statement.

You type:	Command window shows:
j	BASE Jump (to) B/C:
b	BASE Jump (to) Back B/A:
<BUG>	BASE Jump (to) Back V:
<OK>	BASE C:

Successor and predecessor

The "successor" of a statement is the next statement that is at the same level and has the same upstatement; not every statement has a successor. The "predecessor" of a statement is the preceding statement that is at the same level and has the same upstatement; not every statement has a predecessor. To see various sections of a document that are at the same level, you can Jump to the successors and predecessors of the statements. For example, if the Part II statement of the table of contents were showing on your screen and you used Jump to display the successor of Part II, you would see Part III at the top of your file window.

You type:	Command window shows:
j	BASE Jump (to) B/C:
s	BASE Jump (to) Successor B/A:
<BUG>	BASE Jump (to) Successor V:
<OK>	BASE C:

If you then used Jump to display the predecessor of Part III, you would see Part II at the top of your file window.

You type:	Command window shows:
j	BASE Jump (to) B/C:
p	BASE Jump (to) Predecessor B/A:
<BUG>	BASE Jump (to) Predecessor V:
<OK>	BASE C:

NOTE: If you try to jump to the Next, Back, successor, or predecessor of a statement that does not have one, the statement you indicate will appear at the top of your file window.

CHANGING VIEWS: VIEWSPECS

NLS enables you to look at your file from several different points of view. For example, you can display your file with blank lines between the statements. When you view your file with blank lines, it does not mean that NLS puts empty spaces into your file; NLS merely displays your file in a different way.

"Viewspecs" are single-letter specifications that determine what kind of view you have of your file. Some of the viewspecs that are currently controlling your view are displayed in the viewspec window. When you enter NLS, certain viewspecs are automatically in effect, such as those that let you see all lines of all statements in your file. To change your view, you can use the Set Viewspecs command. Type "<SP>se" for "Set" and "v" for "Viewspecs". You will then see a "V:" prompt, which means that you can type one or more viewspecs followed by <OK>. Uppercase viewspecs produce different results than lowercase viewspecs, so be sure to type the proper case. Set Viewspecs

Viewing Blank Lines between Statements

Viewspecs y and z control the blank lines between statements.

Viewspec:	Means:
y	Blank lines between statements on
z	Blank lines between statements off

Viewspecs y and z

For example, to see your file with blank lines between the statements, you can use the Set Viewspecs command as follows:

You type:	Command window shows:
<SP>se	BASE Set C:
v	BASE Set Viewspecs V:
y	BASE Set Viewspecs "y" V:
<OK>	BASE C:

Many people find that files are easier to read with viewspec y. You can change your view in other ways by setting other viewspecs while keeping the blank lines between statements on.

Viewing Statement Numbers

Viewspecs not only enable you to control the way you see your file, they also enable you to display helpful information that NLS maintains about each statement; for example, you can display your file with statement numbers. A statement number is a series of alternating numbers and letters that indicates the exact position of a statement within the hierarchical structure of a file. The first statement after the origin statement is always statement 1. The first substatement after statement 1 is statement 1a, and the successor of statement 1 is statement 2. To see the statement numbers for your file, you can use the Set Viewspecs command with viewspec m. Viewspec m

You type:	Command window shows:
<SP>se	BASE Set C:
v	BASE Set Viewspecs V:
m	BASE Set Viewspecs "m" V:
<OK>	BASE C:

By viewing your file with statement numbers on, you can determine the level of a particular statement. Note that a statement number is not part of a statement; you cannot use editing commands to delete or change it.

To see your file again without the statement numbers, use the Set Viewspecs command with viewspec n. Viewspec n

Viewspec:	Means:
m	Statement numbers on
n	Statement numbers off

Level and Line Clipping Viewspecs

Viewspec d

To get an overall picture of a book, a reader often looks at a list of the chapter titles if one has been specially prepared. In NLS, you can easily display only the main headings in any document by using viewspecs to display a limited number of levels in the file. This is called "level clipping". For example, you can use the Set Viewspecs command with viewspec d to "show first level only".

The viewspec window on your screen keeps you informed of how many levels are being displayed. Notice that if you set viewspec d, you will see "1 ALL" in the viewspec window. This means that one level and all lines are being displayed.

Viewspec b

If you want to see more than one level, you can display the two highest levels, the three highest levels, or any number of levels up to 64. After setting viewspec d, you can set viewspec b, which means "show one level more"; you will then see two levels of statements in the file window and "2 ALL" in the viewspec window. To display three levels, you can add another viewspec b for "show one level more", for four levels, you can add still another viewspec b, and so on. Thus, if you were viewing a file with all levels and all lines and you wanted to see only the top three levels, you could set your viewspecs as follows:

```
  You type:        Command window shows:

  <SP>se          BASE  Set C:
  v               BASE  Set Viewspecs V:
  dbb             BASE  Set Viewspecs "dbb" V:
  <OK>            BASE C:
```

Viewspec a

Viewspec a means "show one level less". For example, if three levels were being displayed and you wanted to see only two, you could set viewspec a.

It is important to understand that level clipping viewspecs display a number of levels more or less than the number that is in effect according to your previous viewspecs, not necessarily a number of levels more or less than what you see on your screen. For example, ALL levels is 64 levels, so if you set viewspec a when you have ALL levels showing, you will have 63 levels; this will change your view only if the part of the file you are looking at contains a statement at level 64.

NOTE: No matter what levels are to be displayed according to your viewspecs, NLS will always show the statement that is at the top of the file window and any other statements following it at the same level. For example, if you set viewspec d to "show first level only" when there are two third-level statements at the beginning of your file window, you will see those third-level statements and then only the first-level statements following them. — Top statement below viewspec levels

You can use viewspec c to display all levels. This viewspec is in effect when you enter NLS. — Viewspec c

If you specify two viewspecs that contradict each other, such as viewspec d to show the first level only and viewspec c to show all levels, the viewspec you type last will take effect. For example, if you gave the command "Set Viewspecs dc", viewspec c would take effect and viewspec d would be ignored.

You can use viewspecs to control the number of lines displayed for each statement in the same way that you can control the number of levels. This is called "line clipping". Use viewspec t to display only the first line in each statement, viewspec r to add lines, viewspec q to take them away, and viewspec s to show all lines. Viewspec s is in effect when you enter NLS. — Viewspecs q, r, s, and t

Notice that your viewspec window shows the number of lines currently being displayed. For example, if you set viewspec t to show first lines only while you have level clipping viewspecs to show three levels, you will see "3 1" in your viewspec window.

The following list summarizes the level and line clipping viewspecs:

Viewspec:	Means:
a	Show one level less
b	Show one level more
c	Show all levels
d	Show first level only
q	Show one line less
r	Show one line more
s	Show all lines
t	Show first lines only

Level and line clipping viewspecs can be used together very effectively. For example, you could set viewspecs d, b, and t — Viewspecs w and x to see the first lines of all statements at the two highest levels, for a good overall outline of a document. Two more viewspecs, w and x, combine level and line clipping.

Viewspec:	Means:
w	Show all lines and all levels
x	Show one line and one level only

CHANGING VIEWS AS YOU READ: JUMPING WITH VIEWSPECS

When you give any Jump command other than Jump Link, you will see a "V:" prompt. This prompt enables you to change your viewspecs while you jump. When the statement you jump to is displayed, any viewspecs that you type after the "V:" will take effect. Remember that no matter what levels are to be displayed according to your viewspecs, NLS will always show the statement you jump to and any statements following it at the same level.

Changing viewspecs while you jump is a good way to look for something in a structured file. You could begin by looking at the file with only one level and one line, jump to the heading that interests you and give a viewspec that adds another level, check the subheadings and jump to one with a viewspec that adds another level, and so on, until you find the statement you want to read. You could then jump to that statement with a viewspec to show all lines so you could read the entire paragraph.

EXERCISES

1. What is the upstatement of the Part III statement in the table of contents? What is the upstatement of "Notes and Comments"?

2. What is the Next of "The Contents"? What is the Back?

3. What is the successor of the Part I statement? What is the predecessor?

4. What is the statement number of the Chap. I statement in Part III? What is the statement number of its upstatement?

SUGGESTED PROJECT

To gain more experience with creating and reading structured files, make a new file and type in the management structure of your own department or organization. When the hierarchy of managers and non-managers is complete, use level clipping viewspecs to see the managers at the top levels of the hierarchy.

SUMMARY

The ability to arrange the information in your NLS file into a hierarchical structure provides several important advantages for writing and reading. Working with structured files not only gives you new tools to help you write, it often changes the way you write. When composing a document in NLS, you are encouraged to plan the overall structure of the document, and this often means that a document written in NLS is more carefully organized than one written by hand or on a typewriter. You can use level clipping viewspecs to see the organization of your document at a glance.

When you learn to use editing commands to change a structured file, you will see that it is very easy to reorganize a well-structured document; with one command you can move a chapter from the beginning to the end of a document or delete a section of unnecessary or redundant information without disturbing the rest of the text. The lesson that teaches you how to do this is "Editing: An Intermediate Lesson".

SOLUTIONS TO EXERCISES

1. The upstatement of the Part III statement is "The Contents". Since "Notes and Comments" is a first-level statement, its upstatement is the origin statement.

2. The Part I statement is the Next of "The Contents". "The Publisher to the Reader" is the Back of "The Contents".

3. The successor of the Part I statement is the Part II statement. The Part I statement does not have a predecessor.

4. The Chap. I statement in Part III has the statement number 3C1; its upstatement has the statement number 3C.

EDITING: AN INTERMEDIATE LESSON

EDITING: AN INTERMEDIATE LESSON

This lesson teaches you to revise a document by taking advantage of the hierarchical structure of NLS files. You will learn how to combine basic editing commands with an understanding of structured files and how to have NLS automatically repeat commands for you. Before reading this lesson, you should be familiar with the information presented in the lessons "Beginning Use of NLS" and "Creating and Reading an Organized File".

INTRODUCTION

The hierarchical structure of NLS files makes it possible for you to control logical sections of a document. It is not only easier to read an online document, it is much easier to totally reorganize one. For example, you can move a chapter from the beginning of a document to the end with one simple command. The editing command verbs Delete, Move, Replace, and Insert, which you have already learned to use with Character, Text, and Word, also apply to units of structure, such as Statement. For example, just as you can use the command Delete Character to remove a character, you can use the command Delete Statement to remove an entire paragraph. You will learn two new units of structure and two new editng command verbs in this lesson.

Revising a document with NLS can mean much more than simply correcting typing errors or grammar. You can essentially create a new document by taking an existing document and using editing commands to revise it, without having to retype any of the original text. As you revise, you can use level and line clipping viewspecs to see an overall picture of the organization of the document and can easily use a few editing commands to reorganize it.

VOCABULARY

branch: A statement and all of its substructure (if it has any). A branch may be a single statement that has no statements below it or an origin statement that includes the entire file.

Copy command: A Base subsystem command that reproduces information in a file, such as a character, a word, some text, a statement, a branch, or a group.

file return ring: A list of the last ten files that you displayed.

group: A series of branches that have the same upstatement.

Jump File Return command: An NLS command that enables you to display one of the files that are in your file return ring. When NLS returns you to the file, the statement that you displayed last will be at the top of the file window and the same viewspecs will be in effect.

<RC>: This stands for Repeat Command. Pressing either the appropriate key on the keyboard or the middle and right mouse buttons puts you in repeat mode.

repeat mode: You enter this mode when you type <RC> in place of the final <OK> in a command. The command is repeated up to the step where you have to bug something or type some text. After you give the final <OK> for the repeated command, it will automatically repeat again, and so on, until you type <CD>. You also enter this mode if you type <RC> when NLS is ready for a command, in which case the last command you gave is repeated until you type <CD>.

Transpose command: A Base subsystem command that enables you to make information such as characters, words, text, statements, branches, or groups exchange places.

STRUCTURE: BRANCH AND GROUP

Besides a statement, there are other units of structure that you can change with a single editing command. Two of these structural units are "branch" and "group".

What a branch is

A branch is a statement and all of its substructure (if it has any). A branch can be a single statement that has no statements below it, or a statement with several levels of substatements below it. One branch may have several other branches within it.

What a group is

A group is any series of branches that have the same upstatement. A group may consist of branches that are statements without substatements as well as branches that have several levels of substructure. To point to a group, bug the top statement of the first branch in the group and then bug the top statement of the last branch in the group. Whenever you designate a group, you must always indicate two things: the first branch in the group and the last branch in the group.

To jump to a branch or a group, simply jump to the statement that is at the top of the branch or at the top of the first branch in the group. (There is no "Jump to Branch" or "Jump to Group" command.) You can do this with any of the Jump commands you have already learned. Branch and group are especially important concepts when you edit structured files, because you can use the command words Branch and Group with editing command verbs such as Delete and Move.

EDITING WITH STRUCTURE

The hierarchical structure of NLS files enables you to control sections of structure, such as Statement, Branch, and Group, in the same way you can control Character, Text, and Word. You can use any of the editing command verbs, such as Delete, Move, Replace, and Insert, to change the organization or content of your document. Rather than retype an entire document, you can easily use structure editing commands to remove inaccurate information and organize accurate information so that it makes sense.

Viewspecs do not affect editing.

While you are editing structure, you should take advantage of level and line clipping viewspecs so you can see more of your file as you work. You can show only one or two levels to get an overall idea of the structure of the document; then when you use structure editing commands, the entire statement, branch, or group will be affected even though it is not displayed. For example, you can delete a branch by bugging its topmost statement and NLS will remove all the information in the branch, even if all of it is not displayed.

In more elementary lessons, we suggested in detail how to build practice files; however, because of the large amount of information you can control with structural editing, it would be tedious for you to create a practice file. Instead, to help you practice editing with structure, we have prepared a file named Sports in the NLS-Documentation directory. This file contains an outline of professional sports teams and includes many errors that you can correct with the commands described in this lesson. Since the files in the NLS-Documentation directory are protected so that you can not make any changes to them, you can edit the information in the Sports file only by copying it into a file in your own directory. This lesson shows how you could copy the information into a file of your own and gives many examples of commands as they would be used to revise it.

COPYING

One of the great advantages of using NLS is that you can copy material from other files in your own directory or from files in other directories and then edit whatever you have copied to suit yourself. For example, if you wanted to write a business letter that was similar but not exactly the same as one written online by a co-worker, you could easily copy the letter and then simply make the necessary changes without retyping it.

To copy information from one file to another file or from one part of a file to another, use the command verb "Copy". Follow it by a noun describing what you want to copy, such as Character, Text, Word, Statement, Branch, or Group. The original information remains untouched. The copied information is like any new information in your file; you can delete it, move it, replace it, and even copy it.

When you use Copy to reproduce sections of structure, such as a statement, a branch, or a group, you can place the copied structure anywhere in a file, at any level; you indicate the statement that you want it to follow and the relative level, just as you do when you use the Insert Statement command.

Copying from a File in Another Directory

While there is a simple way to copy a whole file from one directory or filename to another (Copy File), in the following we use a slightly more complex method of obtaining a copy of an existing file, to illustrate a way to copy a portion of a file.

To copy information from the file named Sports in the NLS-Documentation directory into a file of the same name in your own directory, you would first give the Create File command and type "sports" as the name of the file. Your next step would be to look at the file in the NLS-Documentation directory.

Seeing another file

You can use the Jump Link command to display another file that is either in your own directory or in another directory. The "link" that you type is a series of characters indicating where to jump to (and thus what to show in your file window). To see a file that is in another directory, you must type the link as follows: the directory name, a comma, the file name, and another comma. For example, this is how you would display the Sports file in the NLS-Documentation directory:

```
  You type:            Command window shows:

  j                    BASE Jump (to) B/C:
  l                    BASE Jump (to) Link B/T/[A]:
  nls-documentation,   BASE Jump (to) Link nls-documentation,
   sports,              sports,
  <OK>                 BASE C:
```

You would then see the beginning of the file in your file window. To get an overall picture of the contents of the file, you could use the Set Viewspecs command with viewspec d, to show the first level only. If you set viewspec d after jumping to the Sports file, you would see the three main headings in the file, one for each of the sports baseball, football, and basketball.

To copy the three first-level branches into the empty Sports file that you created, you could use the Copy Group command. When you copy material from a file after setting level or line clipping viewspecs, you are copying everything that is in those branches, not just what you see on the screen. So you could have viewspec d on as you copy the branches, and although you would see only the first-level statements in the file, you would be copying everything. Viewspecs control only what you see, not what is actually there.

Copy Group

In the Copy Group command, you specify the group you want to copy by bugging two statements, one for the first branch in the group and one for the last branch in the group. For the first branch to be copied from the Sports file, you would bug one of the characters in the baseball heading, and for the last branch, one of the characters in the basketball heading. To specify what you want the copied group to follow, you would type your directory name, a comma, the file name "sports", another comma, and then <OK>. This tells NLS that you want the group to follow the origin statement of the Sports file in your own directory. After typing <OK>, you would see "OPT/L:", prompting you for the level of the group relative to the statement it will follow; here you would simply type another <OK>.

```
You type:      Command window shows:

c              BASE Copy C:
g              BASE Copy Group (from) B/A/[T]:
<BUG>          BASE Copy Group (from)  (through) B/A/[T]:
<BUG>          BASE Copy Group (from)  (through)
                (to follow) B/A:
directory,     BASE Copy Group (from)  (through)
 sports,        (to follow) directory,sports, A:
<OK>           BASE Copy Group (from)  (through)
                (to follow) directory,sports,  L/[**]:
<OK>           BASE C:
```

Your Sports file would then contain a copy of the group that you bugged.

Notice that you can add information to a file that is not currently displayed. To display your new Sports file, you could use the Jump Link command, as follows:

```
You type:      Command window shows:

j              BASE Jump (to) B/C:
l              BASE Jump (to) Link B/T/[A]:
sports,        BASE Jump (to) Link sports,
<OK>           BASE C:
```

In general, you can copy material from any file that you can read, and you can read any file that you have access to; a forthcoming lesson will contain information about file access. However, if you copy from a file in someone else's directory and that person is still making changes to the file, you will not be able to read or copy the latest changes. You can only copy from the file as it was the last time it was updated. Note that when you display a file that someone else is editing, NLS will inform you with a message in your status window.

You may not see recent changes.

Copying within a File

Copy Word

To practice copying information from one part of a file to another, you could improve the baseball section of the Sports file. If you set viewspecs to show three levels, you would see an American League heading under Baseball and two headings under American League, one "Eastern Division" and the other simply "Western". To copy the word "Division" from the Eastern Division heading to follow the word "Western", you could type "cw" for "Copy Word", bug "Division", and then bug "Western" as the word that the copied word should follow.

```
  You type:      Command window shows:

  c              BASE Copy C:
  w              BASE Copy Word (from) B/A/[T]:
  <BUG>          BASE Copy Word (from)  (to follow) B/A:
  <BUG>          BASE Copy Word (from)  (to follow)  OK:
  <OK>           BASE C:
```

Copy Statement

You might also notice that we have failed to separate the National League into divisions. You could use the Copy Statement command to copy the division names from the American League. For example, to copy the heading "Eastern Division", you would type "cs" for "Copy Statement", bug Eastern Division, and then bug National League as the statement that the copied statement should follow. After the "OPT/L:" prompt, you would have to type "d" for "down" to indicate that the division heading should be one level down from the league heading.

```
  You type:      Command window shows:

  c              BASE Copy C:
  s              BASE Copy Statement (from) B/A/[T]:
  <BUG>          BASE Copy Statement (from)  (to follow) B/A:
  <BUG>          BASE Copy Statement (from)  (to follow)
                  L/[**]:
  d              BASE Copy Statement (from)  (to follow) "d"
                  L:
  <OK>           BASE C:
```

There would then be an Eastern Division heading under National League. You could use Copy Statement similarly to copy the heading "Western Division" to follow this heading at the same level. Later in this lesson you will learn how to move the teams in the National League into the appropriate divisions.

TRANSPOSING

The command verb "Transpose" enables you to make information such as characters, words, text, statements, branches, or groups exchange places. When you transpose structure, the statements, branches, or groups do not have to be at the same level.

Transpose Statement

For example, if you were to jump to the football heading in the Sports file and had all levels showing (but no blank lines between statements), you would see that in the American Conference of the National Football League, the list of divisions includes "Denver Broncos" and the first team under it is "Western Division". To make these two statements change places, you would type "ts" for "Transpose Statement", bug one of the characters in Denver Broncos, and then bug one in Western Division.

```
You type:      Command window shows:

t              BASE Transpose C:
s              BASE Transpose Statement (at) B/A:
<BUG>          BASE Transpose Statement (at)  (and) B/A:
<BUG>          BASE Transpose Statement (at)  (and)
                OK/[**]:
<OK>           BASE C:
```

Transpose Branch

Suppose you then wanted to reverse the order of the Central Division and the Eastern Division. You would type "tb" for "Transpose Branch", bug one of the characters in Central Division, and then bug one in Eastern Division.

```
You type:      Command window shows:

t              BASE Transpose C:
b              BASE Transpose Branch (at) B/A:
<BUG>          BASE Transpose Branch (at)  (and) B/A:
<BUG>          BASE Transpose Branch (at)  (and)  OK/[**]:
<OK>           BASE C:
```

Transpose Group

To use Transpose Group, you must point to the two groups that you want to exchange places. When you use the Transpose Group command, you will see the "B/A:" prompt four times; each prompt means that you have to indicate a statement. After the first "B/A:" prompt, bug the top statement of the first branch in the first group, and after the second "B/A:" prompt, bug the top statement of the last branch in the first group. After the third "B/A:" prompt, bug the top statement of the first branch in the second group, and after the fourth "B/A:" prompt, bug the top statement of the last branch in the second group. If you watch your prompts and noise words carefully, you will be able to tell what to do next.

DELETING STRUCTURE

The easiest way to remove unnecessary information in an NLS file is to delete it. Using the Delete command verb to remove structure will show quite dramatically how an editing command can affect an entire section.

Delete Statement

For example, in the Sports file, if you read the list of teams under the Eastern Division in the American Conference of the National Football League, you would see that the last team in the list is the Denver Athletes. To remove this team from the list, you would type "ds" for "Delete Statement" and bug the statement.

```
You type:       Command window shows:

d               BASE Delete C:
s               BASE Delete Statement (at) B/A:
<BUG>           BASE Delete Statement (at)  OK/[**]:
<OK>            BASE C:
```

Delete Branch

In the list of divisions in the American Conference of the National Football League, there is an Unknown Division with three unknown teams. The statement "Unknown Division" with its three substatements is a branch. To remove this entire division, you would type "db" for "Delete Branch" and bug one of the characters in the statement "Unknown Division".

```
You type:       Command window shows:

d               BASE Delete C:
b               BASE Delete Branch (at) B/A:
<BUG>           BASE Delete Branch (at) OK/[**]:
<OK>            BASE C:
```

Delete Group

In the list of teams for the Pacific Division in the Western Conference of the National Basketball Association, there is a group of teams that does not belong: the Tacoma Crows, the Salt Lake City Bees, and the Santa Barbara Waves. To remove these three teams with one command, you would type "dg" for "Delete Group", bug one of the characters in Tacoma Crows, and then bug one in Santa Barbara Waves.

```
You type:       Command window shows:

d               BASE Delete C:
g               BASE Delete Group (at) B/A:
<BUG>           BASE Delete Group (at)  (through) B/A:
<BUG>           BASE Delete Group (at)  (through) OK/[**]:
<OK>            BASE C:
```

MOVING STRUCTURE

You can use the Move command verb to transfer structure, such as a statement, a branch, or a group, from one place to another within a file or from one file to another.

Move Statement

For example, in the Sports file, the football team New York Jets is listed under the Central Division but belongs in the Eastern Division. To move this team under the appropriate division, you would type "ms" for "Move Statement", bug one of the characters in New York Jets as the statement you want to move, and then bug one of the characters in New England Patriots as the statement you want New York Jets to follow. After the "OPT/L:" prompt, you must indicate the relative level of the moved statement in the same way that you indicate the relative level after the "L/T/[A]:" prompt when you use the Insert Statement command. To place New York Jets at the same level as New England Patriots, you would simply type <OK> after the "OPT/L:" prompt.

```
You type:       Command window shows:

m               BASE Move C:
s               BASE Move Statement (from) B/A/[T]:
<BUG>           BASE Move Statement (from)  (to follow) B/A:
<BUG>           BASE Move Statement (from)  (to follow)
                 L/[**]:
<OK>            BASE C:
```

Changing the level of a statement

You may not always want to move the structure to a new place in a file; you may want only to change its hierarchical position. You can use Move to change the level of the structure without moving it to another place in the file. For example, although it follows the correct statement, the basketball team New York Nets is at division level and should be at team level under the Atlantic Division. To change its level, you would type "ms" for "Move Statement", bug one of the characters in New York Nets and one of the characters in Philadelphia 76ers, and then, at the "OPT/L:" prompt, type <OK> to indicate that you want New York to be at the same level as Philadelphia.

NOTE: You cannot use the Move Statement command to move a statement that has substructure; such a statement is the beginning of a branch and can be moved only as a branch.

Moving a statement with substructure

To move a branch, type "mb" for "Move Branch", bug the top statement of the branch, and then bug the statement you want the branch to follow. After the "OPT/L:" prompt, indicate if you want the branch to be up, down, or at the same level as the statement it will follow, just as you do when you use the Insert Statement command.

Move Branch

To move a group, type "mg" for "Move Group", bug the top statement of the first branch in the group after the first "B/A:" prompt and the top statement of the last branch in the group after the second "B/A:" prompt. After the third "B/A:" prompt, bug the statement you want the group to follow. Then, after the "OPT/L:" prompt, indicate if you want the branch to be up, down or at the same level as the statement it will follow.

Move Group

To practice using Move, you could move the baseball teams in the National League into the appropriate divisions. The Eastern Division should include Chicago, Montreal, New York, Philadelphia, Pittsburgh, and St. Louis; the Western Division should include Atlanta, Cincinnati, Houston, Los Angeles, San Diego, and San Francisco.

REPLACING STRUCTURE

You can use the Replace command verb to remove structure, such as a statement, a branch, or a group, and put new information in its place.

With the Replace Statement command, you can type or bug a new statement to replace the old one. For example, in the Sports file, you could use the Replace Statement command to change the heading "Baseball" to "Major Baseball Leagues".

Replace Statement

You type:	Command window shows
r	BASE Replace C:
s	BASE Replace Statement (at) B/A:
<BUG>	BASE Replace Statement (at) (by) B/T/[A]:
Major... Leagues	BASE Replace Statement (at) (by) Major Baseball Leagues OK:
<OK>	BASE C:

The Replace Branch command removes a branch from your file and gives you the choice of either bugging a branch to put in its

Replace Branch

place or typing a statement. If you type a statement, it will be added in place of the top statement of the branch that is being replaced.

Replace Group

The Replace Group command removes a group from your file and gives you the choice of either bugging a group to put in its place or typing a statement. If you type a statement, it will be added in place of the top statement of the first branch in the group that is being replaced.

INSERTING STRUCTURE

Bugging statements to insert

If you consider the "B/T/A:" prompt that you see when you use the Insert Statement command, you will see that you can not only type the statement that you want to add to your file, but you can also bug the statement. When you bug the statement, it is copied as if you had used the Copy Statement command. To use the Insert Statement command to copy a statement that you bug, type "is" for "Insert Statement", bug one of the characters in the statement you want the new statement to follow, indicate the relative level of the new statement, and then bug one of the characters in the statement you want to copy. The example below adds the new statement at the same level as the statement you indicate the new statement should follow.

```
You type:      Command window shows:

i              BASE Insert C:
s              BASE Insert Statement (to follow) B/A:
<BUG>          BASE Insert Statement (to follow)  L:
<OK>           BASE Insert Statement (to follow) B/T/[A]:
<BUG>          BASE Insert Statement (to follow) OK:
<OK>           BASE C:
```

Insert Branch

The Insert Branch command adds a branch to your file. You can type the branch if you use <INS> to enter insert mode after typing the first statement, just as when you use the Insert Statement command. Or you can bug a branch to be copied as in the Copy Branch command. To bug the branch, simply bug one of the characters in the top statement of the branch.

Insert Group

The Insert Group command adds a group to your file. You can type the group if you use <INS> to enter insert mode after typing the first statement, just as when you use the Insert Statement command. Or you can bug a group to be copied as in the Copy Group command. To bug the group, bug one of the characters in the top statement of the first branch in the group and then bug one of the characters in the top statement of the last branch in the group.

REPEATING COMMANDS

Repeat mode

You may find that you often need to repeat the same command over and over again. If you type <RC> instead of the final <OK> in a command, you will enter repeat mode. In repeat mode, the last command that you used is repeated up to the step where you have to bug something or type some text, and after you give the final <OK>, the command repeats again. The command will continue to repeat until you type <CD> to stop. For example, if you want to read an entire file by using the Jump command over and over again, you can give the Jump command once and end with <RC> instead of <OK>. You will then be in repeat mode until you type <CD> to leave it.

```
You type:       Command window shows:

j               BASE Jump (to) B/C:
<BUG>           BASE Jump (to)  V:
<RC>            BASE Jump (to) B/C:
<BUG>           BASE Jump (to)  V:
<OK>            BASE Jump (to) B/C:
<BUG>           BASE Jump (to)  V:
<OK>            BASE Jump (to) B/C:
<CD>            BASE C:
```

You also enter repeat mode if you type <RC> when NLS is ready for a command. The last command you gave will be repeated until you type <CD>. For example, if Replace Statement was the last command you gave, you could type <RC> after "BASE C:", and NLS would repeat the Replace Statement command up to the first "B/A:" prompt, where you have to indicate which statement you want to replace.

```
You type:       Command window shows:

rs              BASE Replace Statement (at) B/A:
<BUG>           BASE Replace Statement (at) (by) B/T/[A]:
statement       BASE Replace Statement (at) (by) statement
<OK>            BASE C:
<OK>            BASE Replace Statement (at) B/A:
```

RETURNING TO A FILE: THE JUMP FILE RETURN COMMAND

File return ring

After moving from one file to another, such as from your initial file to a file you want to copy and then from that file to one you want to edit, you can easily return to any of the files you displayed previously. NLS maintains a list of the last ten files you displayed; this list is called the "file return ring". The Jump File Return command enables you to display again one of the files in your file return ring. Every time you jump to another file, it is added the list. When NLS returns you to a previous file, it will display your last view of that file, with the same statement at the top of the file window and the same viewspecs in effect.

When you use the Jump File Return command, NLS reminds you of each file in your file return ring by displaying the directory name and the file name. These "flashbacks" are presented one at a time so you can type "y" for "yes" (meaning you want to return to that particular file) or "n" for "no" (meaning that you do not want to return to that particular file and want to consider the next file on the list). For example, to return to the last file that you displayed, you would type "jfr" for "Jump File Return", follow with <OK>, and then answer "yes" after the first "Y/N:" prompt.

```
You type:       Command window shows:

j               BASE Jump (to) B/C:
f               BASE Jump (to) File B/C:
r               BASE Jump (to) File Return OK:
<OK>            BASE Jump (to) File Return
                < DIRECTORY, FILE.NLS;1, > Y/N:
<OK>            BASE C:
```

When you return to a file in your file return ring, it becomes your current file and is added to the list.

NOTE: Whenever you see a "Y/N:" prompt, you may type <OK> as well as "y" for "yes".

EXERCISES

Use the editing commands introduced in this lesson to correct the disorganized meal plan in the file named Meals in the NLS-Documentation directory.

1. Copy the first week of the meal plan to your own work file.

2. Delete the two extra copies of the plan for Sunday.

3. Use one command to put the days of the week into proper order.

4. Move the dinner plan to its proper place on Saturday.

5. Change the breakfast and dinner meals on Thursday so the day will end with pizza, beer, and ice cream instead of starting with them. Use only one command.

6. Sunday has two dinners that are exactly the same and no lunch. Change one of the dinners to a light meal for lunch.

SUMMARY

When you use editing commands to control logical sections of a structured file and level and line clipping viewspecs to display the overall picture, you have the power to revise or reorganize the file in any way that you want. It is just as easy to delete, move, replace, insert, copy, and transpose statements, groups, branches as it is to control characters, words, and text. You can try making changes just to see if they make sense; if you aren't pleased, it is easy to try another change or even put things back the way they were. If you create logically structured files as you write, it will be much easier for you or a co-worker to revise, reorganize, or find appropriate material to copy.

Revising a document with NLS is a quick and easy task; you can update existing proposals, reports, letters, and so on, rather than start from scratch. Because it is so easy to revise existing documents, be sure to read carefully everything that you copy. Using the same poorly written document over and over again is not an effective use of NLS.

To learn more about structure and how it can help you look at your files as well as edit them, see the lesson "Viewing: An Intermediate Lesson".

SOLUTIONS TO EXERCISES

1. Use the Create File command to create your own work file, the Jump Link command to look at the file in the NLS-Documentation directory, and the Copy Branch command to copy the first week into your file.

2. Use the Delete Group command to delete the last two Sundays.

3. Use the Move Group command to move Thursday and Friday to follow Wednesday.

4. Use the Move Branch command to move Dinner to follow Lunch on Saturday. Bug Lunch as the statement you want Dinner to follow and keep it at the same level as Lunch.

5. Use the Transpose Group command to switch the food under Breakfast with the food under Dinner.

6. Use the Replace Branch command to delete the first Dinner and type in a new heading for lunch. Then use the Insert Statement command to add a new meal plan for the lunch.

VIEWING: AN INTERMEDIATE LESSON

VIEWING: AN INTERMEDIATE LESSON

This lesson contains an intermediate-level description of how to read and search for information in NLS files. The techniques introduced here will enable you to read files quickly and flexibly. Skill in viewing files will increase the productivity of your work in NLS.

To understand this lesson, you must be familiar with basic NLS editing and viewing, covered in the lessons "Beginning Use of NLS" and "Creating and Reading an Organized File", and with the structural terms "branch" and "group", discussed in the lesson "Editing: An Intermediate Lesson".

INTRODUCTION

The hierarchical structure of NLS files provides you with several important advantages when you need to look at online information. You can use level and line clipping viewspecs to view the information in many different ways. When you are reading a document in a file, you can easily skip from one section in the document to another according to their logical positions in the hierarchy. This lesson introduces some new structural entities and relationships that will enable you to take further advantage of hierarchical organization when reading files as well as when editing them.

When you want to see a statement that is in the middle of a file, there are many ways to reach it. You can use a command to get directly to a specific location. Or, if you do not know the exact location of a statement, you can give a command to find the statement that includes a particular phrase or you can use viewspecs to scan the file until you find the statement. Using these techniques to look at a file is often called "viewing" rather than "reading" because you can carefully select what you want to see; you do not have to wade through pages of unnecessary information as you do when you are reading a book.

To practice viewing as described in this lesson, use the Jump Link command to display the file named Output-processor in the NLS-Documentation directory. This file contains the online version of the Output Processor Users' Guide.

```
  You type:           Command window shows:

  j                   BASE Jump (to) B/C:
  l                   BASE Jump (to) Link B/T/[A]:
  nls-documentation,  BASE Jump (to) Link nls-documentation,
   output-             output-processor,
   processor,
  <OK>                BASE C:
```

VOCABULARY

address: A location in an NLS file.

Down: A command word that refers to the first substatement, if any, of the statement you indicate.

end of branch: The last statement in a specified branch, regardless of level.

head: The first statement in a plex.

ident: A short series of characters that identifies an individual or a group to NLS.

Jump Return command: An NLS command that enables you to return to one of the statements in the statement return ring for the file you are looking at. When NLS returns you to a previous statement, the statement will again appear at the top of the file window and the same viewspecs will be in effect as when you last viewed the statement.

Jump Content First command: An NLS command that displays the first statement, starting from the beginning of your file, that contains the text you specify for content.

Jump Content Next command: An NLS command that displays the next statement, following the statement at the top of your file window, that contains the text you specify for content.

Jump Word First command: An NLS command that displays the first statement, starting from the beginning of your file, that contains the word(s) you specify for content.

Jump Word Next command: An NLS command that displays the next statement, following the statement at the top of your file window, that contains the word(s) you specify for content.

link: A series of characters that indicates a location in an NLS file.

plex: All the branches that have the same upstatement as the branch that you indicate.

SID: This stands for Statement Identifier, which is a unique number that NLS automatically assigns to every statement as it is added to a file. An SID is always a number that begins with zero. Unlike a statement number, the SID of a particular statement does not change when the structure of the file changes.

statement number: A series of alternating numbers and letters that indicates the exact position of a statement within the hierarchical structure of a file.

statement return ring: A list of the last ten statements you jumped to in a particular file.

statement signature: The ident of the person who last changed a statement and the date and time the change was made. You can use viewspec K to see statement signatures.

tail: The last highest-level statement in a plex.

Up: A command word that refers to the upstatement of the statement you indicate, that is, the statement that is one level above it.

SETTING VIEWSPECS WITH THE MOUSE

As you become more experienced in working with information in NLS files, you will often want to change your view. You can change viewspecs in the Base subsystem by giving the Set Viewspecs command. However, when you are reading or editing a file and want to change viewspecs, you may not want to pause to give the Set Viewspecs command. You can instead use the mouse to change your viewspecs, without giving a command. You can even change your viewspecs with the mouse while you are in the middle of specifying a command.

Which buttons to press

For lowercase viewspecs, simply press the left and middle mouse buttons and hold them down as you type the appropriate lowercase characters on the keyboard or keyset. Do not release the mouse buttons until you have finished typing the viewspecs. For uppercase viewspecs, press all three mouse buttons as you type the viewspec characters; you can type them in lowercase because pressing all three mouse buttons, like pressing a SHIFT key, tells NLS to take them as uppercase.

Viewspecs f and F

When you give the Set Viewspecs command, NLS automatically displays your file with the new viewspecs in effect. However, when you use the mouse to change viewspecs, you must set either viewspec f or viewspec F along with your other viewspecs to make the view change.

Viewspec:	Means:
f	Put viewspecs into effect
F	Recreate the display

For example, to see your file with blank lines between the statements, hold down the left and middle mouse buttons while typing "yf". You will see "BASE "y" in your command window. If you do not set viewspec f or F, the view will change only when you use an editing command that changes it or when you jump to another statement.

NOTE: Using the mouse to change viewspecs works in almost all NLS subsystems, while the Set Viewspecs command is available only in Base.

VIEWING STATEMENT NUMBERS AND SIDS

NLS mantains both a statement number and an SID for every statement. You can use viewspecs to display your file with statement numbers or SIDs. The statement number indicates the exact position of the statement in the hierarchical structure of the file. A statement number is a series of alternating numbers and letters that begins with a number; the origin statement of a file has the statement number 0, the first statement you add to the file will have the statement number 1, its first substatement will have the statement number 1a, and so on. The SID (which stands for Statement Identifier) is a unique number that NLS automatically assigns to each statement as it is added to the file. An SID is always a number that begins with zero; the origin statement has the SID 01, the first statement that you add to the file will have the SID 02, the second will have the SID 03, and so on, regardless of their level.

When you move a statement to another position in the file, the statement number changes to reflect the new position; however, the SID does not change. Once an SID is assigned to a statement, it remains with that statement even if you move the statement or change the text. Note that neither statement numbers nor SIDs are part of the actual statement, so they cannot be deleted or moved.

SIDs do not change.

The combination of viewspec m and viewspec I displays the SIDs in your file. Viewspec m means "statement numbers/SIDs on" and viewspec I means "show SIDs, not statement numbers". If you set viewspec m without specifically setting viewspec I at some point during your work session, NLS will assume that you want to see statement numbers rather than SIDs.

Viewspec I

You type:	Command window shows:
<SP>se	BASE Set C:
v	BASE Set Viewspecs V:
mI	BASE Set Viewspecs "mI"
<OK>	BASE C:

To use the mouse to display SIDs, press the left and middle mouse buttons and hold them down while typing "m", and then press all three mouse buttons and hold them down while typing "if". You will see BASE "mI" in your command window. Remember that when you press all three mouse buttons, the characters you type are taken as uppercase viewspecs, even if you type them in lowercase.

After setting viewspecs m and I, you can see statement numbers rather than SIDs by using viewspec J to "show statement numbers, not SIDs". To turn off either statement numbers or SIDs, use viewspec n for "statement numbers/SIDs off".

Viewspec J

Viewspecs G and H control the position of the statement numbers

Viewspecs G

and SIDs when they are displayed. Viewspec G places "statement numbers/SIDs right" and viewspec H places "statement numbers/SIDs left". There are various advantages to having the statement numbers or SIDs on the right or on the left; you will have to find out which works best for you in different situations. If you do not specifically set viewspec G sometime during your work session, viewspec H will automatically be in effect. For example, to use the mouse to display statement numbers on the right, press the left and middle mouse buttons and hold them down while typing "m", and then press all three mouse buttons and hold them down while typing "gf".

and H

In summary, three pairs of viewspecs control statement numbers and SIDs: m/n, I/J, and G/H.

Viewspec:	Means:
m	Statement numbers/SIDs on
n	Statement numbers/SIDs off
I	Show SIDs, not statement numbers
J	Show statement numbers, not SIDs
G	Statement numbers/SIDs right
H	Statement numbers/SIDs left

It is important to understand how each pair of viewspecs interacts with the others. You cannot show either statement numbers or SIDs unless viewspec m is set. For example, if you set viewspec I or G while viewspec n is in effect, NLS will not display the SIDs. Similarly, when you set viewspec m, the status of I/J and G/H determines the type of numbers that will appear and whether they will appear on the left or the right.

Level and line clipping viewspecs do not conflict in any way with the viewspecs that control statement numbers and SIDs. When you use any combination of level and line clipping viewspecs with statement numbers or SIDs, you will see the statement numbers or SIDs for all the statements that are displayed, but you will not see any for the statements that are not displayed.

ADDRESSING WITH STATEMENT NUMBERS AND SIDS

In NLS commands, you can use statement numbers or SIDs to indicate which statements you want to find or change. For example, when you to want to display a statement at the top of your file window, you need to give the location of the statement. A common method of giving the location of a statement is to bug it, but you cannot bug a statement that is not displayed on your screen. You can, however, use either the statement number or the SID to locate a particular statement.

Addresses and links

A location in an NLS file is called an "address", and the series of characters that you type to indicate an address is called a "link". Just as you use the Jump Link command to move from one file to another, you can use it to move from one statement to another within a file, by typing a statement number or an SID after the "B/T/[A]:" prompt. For example, you can display the third first-level statement at the top of your file window as follows:

```
You type:      Command window shows:

j              BASE Jump (to) B/C:
l              BASE Jump (to) Link B/T/[A]:
3              BASE Jump (to) Link 3
<OK>           BASE C:
```

If you set viewspecs to see SIDs when looking at the Output-processor file, you will notice that statement 3 has the SID 027. To display this statement at any time, even after editing may have changed the position of some statements, you can use the SID in the Jump Link command.

```
You type:      Command window shows:

j              BASE Jump (to) B/C:
l              BASE Jump (to) Link B/T/[A]:
027            BASE Jump (to) Link 027
<OK>           BASE C:
```

Typing and punctuating links

Note that the "B" and "T" in the "B/T/[A]:" prompt mean that you can bug a link or, as shown in the preceding examples, type a link. The information that you can put in a link includes a directory name, a file name, and a statement number or SID. The form of the information you type in the link tells NLS how to interpret it. For example, you have already learned that a comma must always follow a file name in a link. A statement number or SID does not need any special punctuation. More information on links will appear in a forthcoming lesson.

Whether you use statement numbers or SIDs in your links depends a great deal on what you are doing. For instance, if you are writing a document that has four first-level branches and you want to see the paragraph immediately below the fourth heading, you can simply jump to statement 4a. In this case, you can figure out the statement number without looking at it. But if you move or delete several statements, the statement numbers change and it is hard to keep track of them. It is then extremely useful to work with a paper copy of the file printed with SIDs so you always know a specific way to address each statement.

STRUCTURAL RELATIONSHIPS: UP, DOWN, AND END

The more you work with structured files, the more often you want to see a statement because it is up or down from another. For example, after you have displayed a statement with a certain SID, you may want to look up at its heading. NLS provides three command words for such purposes: Up, Down, and End. Like Next, Back, Successor, and Predecessor, these words follow the verb "Jump".

Up and Down

"Up" is the upstatement of the statement that you indicate and "Down" is the first substatement of the statement that you indicate. Every statement in a file has an Up, except for the origin statement, but not every statement has a Down. For example, to see the Up of one of the statements in your file window, you can use the Jump Up command and bug the statement.

You type:	Command window shows:
j	BASE Jump (to) B/C:
u	BASE Jump (to) Up B/A:
<BUG>	BASE Jump (to) V:
<OK>	BASE C:

Similarly, to see the Down of one of the statements in your file window, you can use the Jump Down command and bug the statement. If the statement does not have a Down, the statement that you bug will move to the top of the file window.

You type:	Command window shows:
j	BASE Jump (to) B/C:
d	BASE Jump (to) Down B/A:
<BUG>	BASE Jump (to) V:
<OK>	BASE C:

The "B/A:" prompt means that you can bug a statement or give an address, such as a statement number or an SID. For example, to see the upstatement of the first statement in your file, you can use the Jump Up command and type "1" for the address; this will put the origin statement at the top of the file window.

```
  You type:       Command window shows:

  j               BASE Jump (to) B/C:
  u               BASE Jump (to) Up B/A:
  1               BASE Jump (to) Up  A: 1
  <OK>            BASE Jump (to) Up  V:
  <OK>            BASE C:
```

End of branch

You can also use the command word "End" after "Jump". The Jump End command displays the "end" of a branch, that is, the last statement in the branch that you indicate, regardless of level. This command is extremely useful for checking the last statement in a particular section of a file. For example, to display the very last statement in your file, you can simply jump to the end of branch 0.

```
  You type:       Command window shows:

  j               BASE Jump (to) B/C:
  e               BASE Jump (to) End (of branch) B/A:
  0               BASE Jump (to) End (of branch) A: 0
  <OK>            BASE Jump (to) End (of branch) V:
  <OK>            BASE C:
```

MORE ON STRUCTURE: PLEX, HEAD, AND TAIL

What a plex is

Another structural concept that will help you to view as well as to edit your files is "plex". The word "plex" is a special NLS term derived from the word "plexus", which the dictionary defines as "an interwoven combination of parts in a structure". Although this sounds rather vague, "plex" has a very specific meaning in NLS: A plex is ALL of the branches that have the same upstatement or, in other words, all of the substructure below a particular statement. Like Statement, Branch, and Group, Plex is a command noun for editing; you can use it after editing command verbs such as Delete and Move.

To point to a plex, you can bug any of the highest-level statements in the plex or type one of their statement numbers or SIDs. For example, in the Output-processor file, all of the branches that have the upstatement 3 constitute a plex. You can refer to this plex as the plex at 3a, the plex at 3b, the plex at 3c, and so on, or you can bug any of the statements or give its SID. Note that the plex may begin with a statement before the one that you point to. For example, if you point to the plex at 3c, the first statement in the plex is 3a.

Since all the first-level branches in a file have the same upstatement (the origin statement), all of the statements in the file except the origin statement constitute a plex. This plex is the plex at statement 1, which is exactly the same as the plex at statement 2, statement 3, statement 4, and so on.

Head and tail of plex

Every plex has a head and a tail. The head is the first statement in the plex and the tail is the last highest-level statement in the plex. If a plex consists of only one branch, the head and tail are the same statement. You can use the command words Head and Tail with the verb "Jump". For example, to see the first item in a long list, you can use the Jump Head command or, to see the heading of the last chapter in a report, you can use the Jump Tail command. Specify the plex that you want to see the head or tail of by typing the address of any one of the highest-level statements in the plex or by bugging it.

```
  You type:      Command window shows:

  j              BASE Jump (to) B/C:
  h              BASE Jump (to) Head B/A:
  <BUG>          BASE Jump (to) Head  V:
  <OK>           BASE C:
```

The following example shows how you can display the last first-level statement in the file you are looking at.

```
  You type:      Command window shows:

  j              BASE Jump (to) B/C:
  t              BASE Jump (to) Tail B/A:
  1              BASE Jump (to) Tail A: 1
  <OK>           BASE Jump (to) Tail V:
  <OK>           BASE C:
```

Plex versus group

Note that "plex" and "group" are closely related concepts. A plex consists of all the branches that have the same upstatement, while a group is any series of branches that have the same upstatement. For example, if the branches beginning at statements 3a through 3c constitute a plex, then the branches beginning at the following statements are groups: 3a through 3b; 3b through 3c; and 3a through 3c, that is, the entire plex.

VIEWING STATEMENT SIGNATURES

A statement signature, like a statement number or an SID, is information about an individual statement that you can display at any time by setting the appropriate viewspec. The statement signature shows the ident of the person who last changed the statement and the date and time the change was made. Displaying statement signatures is especially useful when more than one person has worked on the same file. It allows you to see which statements were added or changed by a co-worker.

Viewspecs K and L control statement signatures.

Viewspec:	Means:
K	Statement signatures on
L	Statement signatures off

Viewspecs K and L

The statement signatures will appear on the right side of the file window. Statement signatures do not conflict with statement numbers or SIDs. You can display a file with both at the same time; the signatures will appear one line below statement numbers or SIDs being displayed on the right.

NOTE: NLS identifies users by idents. If a user has a directory in his or her own name, the statement signature for that user will show the ident associated with that directory. If a user is one of several who share a directory, the statement signature will contain the ident that the user specified upon entering NLS. If a co-worker is using your directory or has specified your ident upon entering NLS, there is no way NLS can tell that it is not actually you; in this case, the statement signatures on the statements changed by your co-worker will show your ident.

CONTENT SEARCHING

Quite frequently when you want to display a statement, you do not know its statement number or SID but you do have some idea of what appears in the statement. The content searching commands in NLS look for a statement that contains the text or word that you specify. For example, if you want to see the paragraphs in the Output Processor Users' Guide that contain information on headers, you can use content searching commands to display the statements that contain "header". Four content searching commands are: Jump Content First, Jump Content Next, Jump Word First, and Jump Word Next.

Jump Content First

Jump Content First looks for the first statement in your file that contains the text you specify for content. You can specify the text by typing it or bugging it. For example, this is how you could find the first statement that has the text "header" in it:

```
You type:        Command window shows:

j                BASE Jump (to) B/C:
c                BASE Jump (to) Content C:
f                BASE Jump (to) Content First RPT/B/T/[A]:
header           BASE Jump (to) Content First "header"
<OK>             BASE Jump (to) Content First "header" V:
<OK>             BASE C:
```

The text you specify may include any characters except quote marks ("). NLS will search for the text exactly as you indicate it, and it will distinguish uppercase from lowercase characters. Thus, the command shown in the preceding example will not find a statement that contains only "Header" and not "header"; to find such a statement, you would have to specify "Header" for content.

Jump Content Next

Jump Content Next will display the next statement, following the statement at the top of the file window, that contains the text you specify for content. Once you have used any content searching command during an NLS work session, you have the choice of searching for the same content you used last time. After you type the command words, the content you used for the last command appears in the command window, followed by the prompt "RPT/B/T/[A]:". To repeat the search, simply type <RC> after this prompt. For example, if you wanted to see the next occurrence of "header", you could use the Jump Content Next command as follows:

```
You type:        Command window shows:

j                BASE Jump (to) B/C:
c                BASE Jump (to) Content C:
n                BASE Jump (to) Content Next "header"
RPT/B/T/[A]:
<RC>             BASE Jump (to) Content Next  V:
<OK>             BASE C:
```

If you want to search for any other content, simply type or bug the text as usual after the "RPT/B/T/[A]:" prompt.

When you are using Jump Content First and no statement contains the text you have specified, or when you are using Jump Content Next and no statement in the rest of the file contains it, NLS will display a message in the status window. You will see the text in quotes followed by a question mark. For example, if you were searching for "header" and typed "heder" by mistake, you would see the message "heder"=W ?? in the status window.

Jump Word versus Jump Content

Jump Word First and Jump Word Next are the same as Jump Content First and Jump Content Next except they search for a word or a series of words rather than text. For example, if you use a Jump Word command and type "header", NLS will search for the word "header", that is, "header" surrounded by characters that are not letters or numbers; but if you use Jump Content and type "header", NLS will search for the text "header" even if it is within another word, such as in "headers" or "doubleheader".

When using the Jump Word First or Jump Word Next command, you can type or bug the word or, if you have already given a content searching command during the work session, you can use <RC>. For example, to find the next statement that contains a word displayed on your screen, you can use the Jump Word Next command and bug the word. If you specified "header" in the last content searching command you gave, this is what will happen:

```
You type:       Command window shows:

j               BASE Jump (to) B/C:
w               BASE Jump (to) Word C:
n               BASE Jump (to) Word Next "header" RPT/B/T/[A]:
<BUG>           BASE Jump (to) Word Next  V:
<OK>            BASE C:
```

RETURNING TO A STATEMENT: THE JUMP RETURN COMMAND

Statement return ring

After you have jumped from one point in a file to another, you may want to go back to where you jumped from. To help you do this, NLS maintains a list, for each file you display, of the last ten statements you jumped to; this list is called the "statement return ring". The Jump Return command enables you to return to one of the statements in the statement return ring. When you return to a previous statement, the statement will again appear at the top of the file window and the same viewspecs will be in effect as when you last viewed the statement.

When you give the Jump Return command, NLS reminds you of each statement in your statement return ring by displaying the first few characters of the statement. NLS displays these "flashbacks" one at a time so you can type "y" for "yes" (meaning you want to return to that statement) or "n" for "no" (meaning you do not want to return to that statement and want to consider the next statement in the list). For example, to see the next to the last statement, you can give the Jump Return command and answer "no" for the last statement and "yes" for the next to the last statement.

You type:	Command window shows:
j	BASE Jump (to) B/C:
r	BASE Jump (to) Return OK:
<OK>	BASE Jump (to) Return Statement begin Y/N:
n	BASE Jump (to) Return Statement begin Further on back Y/N:
y	BASE Jump (to) Return Statement begin no Further on back OK:
<OK>	BASE C:

When you return to a statement in your statement return ring, it becomes your current statement and is added to the list.

NOTE: Whenever you see a "Y/N:" prompt, you may type <OK> as well as "y" for "yes".

EXERCISE

Find the paragraphs in the Output Processor Users' Guide that include the word "switch". After you find each paragraph, jump to the heading for that section of the document, then return the paragraph you found to the top of your file window before looking for the next paragraph.

SUMMARY

By taking advantage of NLS viewing capabilities, you can freely move through a document of any length without reading or even scanning unnecessary paragraphs. When you understand the relationships between statements in a hierarchical file, you can easily jump from one logical section of the document to another. You can use statement numbers and SIDs to display immediately any particular statement. Content searching commands enable you to find statements that contain specific information. Besides the level and line clipping functions, viewspecs control helpful information about individual statements, such as SIDs and statement signatures.

Viewing NLS files is a important aspect of working in NLS. The ability to see the information in your file in a variety of ways is valuable no matter what subsystem you are using. The Jump commands and the ability to change viewspecs are universal; you will use them for almost every kind of online work.

SOLUTION TO EXERCISE

If you are not already looking at the Output Processor Users' Guide, use the Jump Link command to display the file named Output-procesor in the NLS-Documentation directory, by typing "nls-documentation,output-processor," after the "B/T/[A]:" prompt. Give the Jump Word First command and type "switch" as the word. After you find the first paragraph, give the Jump Up command and bug the paragraph to see the heading for that section of the document. Then use the Jump Return command to return the paragraph you found to the top of your file window. To find the next statement that includes the word "switch", use the Jump Word Next command and type <RC> after the "RPT/B/T/[A]:" prompt. Then give the Jump Up and Jump Return commands as before, followed by another Jump Word Next command, and so on.

NLS VOCABULARY

NLS VOCABULARY

Action: A service provided by the systems staff that enables users to request assistance and register complaints, suggestions, or compliments.

address: A location in an NLS file.

Back: A command word that refers to the statement immediately before the statement you indicate, regardless of level.

Base subsystem: A basic set of commands for reading, writing, editing, printing, and controlling files.

<BC>: This stands for Backspace Character. Pressing either the appropriate key on the keyboard or the left mouse button deletes the last character you typed. You can also use <BC> to delete a <BUG> or any step in a command.

branch: A statement and all of its substructure (if it has any). A branch may be a single statement that has no statements below it or an origin statement that includes the entire file.

Branch command: A Sendmail subsystem command in which you specify a branch that you want to send.

bug: To bug means to indicate a character on the screen by using the mouse to point at it and then typing <OK>.

<BUG>: This notation means that you are to bug a character on the screen.

bugmark: The mark displayed on the screen when you bug a character. The bugmark will be a highlighted character.

<BW>: This stands for Backspace Word. Pressing either the appropriate key on the keyboard or the left and middle mouse buttons deletes the last word you typed (plus any spaces, punctuation marks, or other characters following the word).

<CD>: This stands for Command Delete. Pressing either the appropriate key on the keyboard or the middle mouse button cancels any command you have not finished (that is, before you have given the final <OK>). You may then begin a new command.

character: A single letter, number, punctuation mark, space, return character, or special control character. You type characters when giving commands and you store characters in files.

command: An instruction you give to the computer to perform an action. When you give NLS a command, NLS will perform the action after you complete the command with a final <OK>.

command word: A word that NLS knows is part of a command, usually a verb or a noun.

command syntax: The general form of a command.

Comment command: A Sendmail subsystem command that lets you enter a comment to appear along with the citation for a Journal item.

Copy command: A Base subsystem command that reproduces information in a file, such as a character, a word, some text, a statement, a branch, or a group.

Create File command: A Base subsystem command that makes a new file in your directory; the file will have the name you specify.

Delete command: A Base subsystem command you can use to remove information, such as a character, a word, or some text.

Down: A command word that refers to the first substatement, if any, of the statement you indicate.

end of branch: The last statement in a specified branch, regardless of level.

file: An online work space, the computer's equivalent of a file folder in a filing cabinet, that you can fill with information that you type in or copy from another file. In NLS, you will always work with material in a file, whether you are reading, writing, or editing.

file return ring: A list of the last ten files that you displayed.

Goto command: The NLS command you use to enter another NLS subsystem.

group: A series of branches that have the same upstatement.

Group command: A Sendmail subsystem command in which you specify a group that you want to send.

head: The first statement in a plex.

Help command: An NLS command to get information about commands, terms, and procedures.

hierarchical structure: The structure of NLS files; an outline form that shows the relationship between the statements.

ident: A short series of characters that identifies an individual or a group to NLS.

initial file: The file, named with your ident, that automatically appears when you first enter NLS.

<INS>: This stands for Insert. Pressing the appropriate key on the keyboard puts you in insert mode.

Insert command: A Base subsystem command that lets you add new information to a file, such as a character, a word, some text, or a statement.

insert mode: You enter this mode when you type <INS> in place of the final <OK> at the end of the Insert Statement command. In insert mode, you can continually add statements until you type <CD>.

Jump command: An NLS command to move from one point in a file to another, or from one file to another.

Jump Content First command: An NLS command that displays the first statement, starting from the beginning of your file, that contains the text you specify for content.

Jump Content Next command: An NLS command that displays the next statement, following the statement at the top of your file window, that contains the text you specify for content.

Jump File Return command: An NLS command that enables you to display one of the files that are in your file return ring. When NLS returns you to the file, the statement that you displayed last will be at the top of the file window and the same viewspecs will be in effect.

Jump Return command: An NLS command that enables you to return to one of the statements in the statement return ring for the file you are looking at. When NLS returns you to a previous statement, the statement will again appear at the top of the file window and the same viewspecs will be in effect as when you last viewed the statement.

Jump Word First command: An NLS command that displays the first statement, starting from the beginning of your file, that contains the word(s) you specify for content.

Jump Word Next command: An NLS command that displays the next statement, following the statement at the top of your file window, that contains the word(s) you specify for content.

level: A number that indicates how far up or down a statement is in the hierarchical structure of a file.

level clipping: Using viewspecs to display a limited number of levels in a file.

line clipping: Using viewspecs to display a limited number of lines for each statement.

link: A series of characters that indicates a location in an NLS file.

menu item: A subtopic, to guide you to related information, listed under a Help description.

Move command: A Base subsystem command to reorder information in a file; for example, you can move one character to follow another.

Next: A command word that refers to the statement immediately following the statement you indicate, regardless of level.

noise word: When you type a command word, NLS may respond with a word or phrase in parentheses, called "noise words", to help you understand the purpose of the command or what you need to do to complete it.

<NULL>: This notation represents a special character that means "nothing" or "none".

<OK>: This notation means that you are to press either the appropriate key on the keyboard or the right mouse button, to tell NLS that you have finished giving a command or part of a command.

origin statement: The first statement in every file. When you create a file, NLS automatically makes an origin statement containing the name of the file and other pertinent information. The origin statement is at level 0, that is, one level above the first-level statements; it is the only statement that can be at this level.

plex: All the branches that have the same upstatement as the branch that you indicate.

Plex command: A Sendmail subsystem command in which you specify a plex that you want to send.

predecessor: The predecessor of a statement is the preceding statement that is at the same level and has the same upstatement. Not every statement has a predecessor.

prompt: A series of characters that appears in the command window to tell you what you can do next. Prompts are always one or more uppercase letters followed by a colon (:).

question mark: When typed after any prompt, question mark (?) will show you what you can do next.

Quit command: The NLS command you use to leave the Sendmail subsystem and return to the subsystem you were last working in.

<RC>: This stands for Repeat Command. Pressing either the appropriate key on the keyboard or the middle and right mouse buttons puts you in repeat mode.

repeat mode: You enter this mode when you type <RC> in place of the final <OK> in a command. The command is repeated up to the step where you have to bug something or type some text. After you give the final <OK> for the repeated command, it will automatically repeat again, and so on, until you type <CD>. You also enter this mode if you type <RC> when NLS is ready for a command, in which case the last command you gave is repeated until you type <CD>.

Replace command: A Base subsystem command to remove information, such as a character, a word, or some text, and put new information in its place.

SID: This stands for Statement Identifier, which is a unique number that NLS automatically assigns to every statement as it is added to a file. An SID is always a number that begins with zero. Unlike a statement number, the SID of a particular statement does not change when the structure of the file changes.

<SP>: This stands for a space, that is, what you type with the space bar on the keyboard. In NLS, a space is an actual character that separates one word from another and that can be inserted, deleted, moved, or copied; it is not emptiness.

statement: The basic unit of information in an NLS file. A statement may be a single character, a word, a title, a heading, some text, or a paragraph. Every character in an NLS file is in a statement.

statement number: A series of alternating numbers and letters that indicates the exact position of a statement within the hierarchical structure of a file.

statement return ring: A list of the last ten statements you jumped to in a particular file.

statement signature: The ident of the person who last changed a statement and the date and time the change was made. You can use viewspec K to see statement signatures.

structure: The arrangement of statements in a file. NLS files have a hierarchical structure.

substatement: A substatement of a statement is any statement that is one level below it. Not every statement has a substatement.

substructure: All of the statements one or more levels below a particular statement. Not every statement has substructure.

subsystem: NLS is divided into subsystems, which are sets of commands related to particular activities.

successor: The successor of a statement is the next statement that is at the same level and has the same upstatement. Not every statement has a successor.

tail: The last highest-level statement in a plex.

text: A series of adjacent characters, which may include punctuation and spaces, within a statement. A single character may also be considered "text".

Transpose command: A Base subsystem command that enables you to make information such as characters, words, text, statements, branches, or groups exchange places.

Up: A command word that refers to the upstatement of the statement you indicate, that is, the statement that is one level above it.

Update File command: A Base subsystem command to consolidate recent changes into your file. You may erase changes made between updates with a command discussed in a forthcoming lesson.

upstatement: The upstatement of a statement is the statement that is one level above it. Every statement except the origin statement has an upstatement.

viewspecs: Single-letter specifications of how you see your file. For example, with one viewspec you will see blank lines between statements, and with another you will see the statements without blank lines between them.

visible character: A character you can see on your screen, such as a letter, number, or punctuation mark.

windows: The screen on the display terminal is divided into four areas, called "windows".

> command window: When you use a command, the command is displayed in this window along with prompts and noise words. You also see the name of the subsystem you are working in.
>
> file window: This window displays files or parts of files.
>
> status window: This window displays messages to you from NLS or the Executive.
>
> viewspec window: This small window displays characters that tell you what kind of view you have of the file being displayed.

word: A series of letters and/or numbers that are surrounded by spaces, punctuation marks, or any other characters that are not letters or numbers. NLS does not consider the surrounding characters as part of the word.