Price $3.00

INTRODUCTION TO THE AUGMENT TEXTBOOK SERIES

Copyright Tymshare Inc., May 1980
20705 Valley Green Drive
Cupertino, California 95014

This document was prepared and composed for printing with AUGMENT by the Office Automation Division of Tymshare, Inc.

AUGMENT Journal Number 48743
Tymshare Document Number 1820

CONTENTS

INTRODUCTION

The AUGMENT Textbook Series describes Tymshare's AUGMENT system. It tells what you can do with AUGMENT and how to do it. We also hope that it will help you develop a new style of working. By using AUGMENT's special tools and techniques rather than typewriters, pencils, and paper, you will be better able to organize your thoughts and written information and communicate them to others. AUGMENT is a versatile system that can help with many different kinds of work; which aspects of it you use will depend on your particular job.

The textbook was written for a very wide range of AUGMENT users. Some are highly technical people, with a great deal of computer experience, who look forward to learning a new system with confidence and pleasure. Others are people who have never seen a computer or a terminal before; they will be using AUGMENT 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 AUGMENT documentation. People who are trained by Tymshare staff need a manual they can use for review or reference, while others want to learn AUGMENT on their own by reading some kind of user guide from beginning to end. Most people need to learn only certain skills beyond the basic ones and do not want to have to wade through a lot of documentation that is not relevant to their work. Still other people want a document that will give them only a general idea of AUGMENT's capabilities; they may not necessarily want to use the system or ever sit down at an AUGMENT work station.

In planning the AUGMENT textbook, we have tried to accommodate as many of these people as possible. The textbook consists of a series of individually-bound lessons. There is an "Introduction to AUGMENT", which gives a general idea of what it is like to learn and use the system. There are basic lessons that teach skills everyone needs to know, starting with "Beginning Use of AUGMENT". There are also more advanced lessons and lessons on specialized topics; often these build upon more basic knowledge taught in other lessons. You do not have to read the lessons that are irrelevant to your work situation or that cover information you already know. The AUGMENT service people who train and advise you will recommend the set of lessons you need. We will continue to add new lessons to the series.

The language in the textbook is simple, nontechnical, and informal. We use a detailed and thorough approach, building carefully upon the basics in sequences we have tested through many hours of training users in AUGMENT. We define, explain, illustrate, and summarize the functions and concepts we present. We include a great many examples that you can follow as exercises, and often suggest further exercises at the end of a lesson. Such thorough treatment may be more than some readers need, but it helps ensure that others will not be confused by too little exposition. The more advanced or specialized a lesson, the less thoroughly we guide the user through the information.

The textbook is useful not only as a learning guide but also for reference. Though it is by no means a complete reference manual, it does include most of what users need to know to do their work in AUGMENT. The organization and layout of each lesson are designed to help readers skim and skip around, both when they first read the lesson and when they refer back to it in search of a specific fact. For example, AUGMENT terms and commands are often noted in the margin beside the paragraphs that discuss them, and the "Vocabulary" section at the end of a lesson lists terms and commands in alphabetical order, describes them briefly, and indicates where they are discussed in the lesson.

The textbook is written primarily for people who do or will have access to a standard AUGMENT display terminal. The features of this terminal include specially marked keys and a pointing device called a "mouse". We have adapted AUGMENT to some display units other than the standard AUGMENT display. Of course, keys and pointing features available on such equipment will be different. You will need to supplement the textbook lessons with our documentation on how to use AUGMENT at that particular terminal. Furthermore, AUGMENT can be used at a typewriter–like terminal; users of AUGMENT at these terminals should be sure to read the special "typewriter mode" lessons.

Because learning AUGMENT is similar to learning a language, the lessons in the textbook are organized like lessons in a foreign language textbook. The following section describes in detail the organization of a typical lesson so you can understand how the various sections relate to each other and can tell which sections will be the most helpful to you.

THE STRUCTURE OF A LESSON

Each lesson begins with a very brief description of the topics it covers, perhaps 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, and possibly the name of a lesson in which you can find it. Following a table of contents, 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.

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. Main headings also appear as running headers at the top of the pages to aid you in flipping back through the material. Illustrations referred to in the lesson may be interspersed with text or may follow on a separate sheet.

Exercises. This section contains exercises to test what you have learned. Solutions are provided at 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.

List of Commands. The AUGMENT commands discussed in the lesson are listed in this section. You are shown all the steps in each command, including both what you type and the words AUGMENT displays to help you fill out the command. You may want to read through this list to review the steps in the commands you have just learned or refer to it later if you have forgotten how to give a command. More information on how the commands are shown appears later in this introduction.

Vocabulary. Here you will find an alphabetical list and short definitions of terms and commands encountered in the lesson. It includes all new terms and commands introduced in the lesson and possibly also some important terms that you should have known before starting the lesson but may have forgotten. You should refer to this section when you are reading the lesson if you encounter an unfamiliar term. You may want to read through it after the body of the lesson to review what you have just learned, or refer to it later if you have forgotten the meaning of a term or the function of a command. For terms and commands that are discussed in the lesson, we include page numbers directing you to the discussions.

Solutions to Exercises. You may check your responses to the exercises.

NOTATION FOR SPECIAL CHARACTERS: < >

The characters you type when you use AUGMENT include not only letters, numbers, and punctuation, but also some characters that you cannot see, such as spaces and characters having a special control function. Where an AUGMENT textbook lesson shows what you type, you can usually tell where to type a space, such as between words in a paragraph. 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 a special notation: an uppercase word or abbreviation enclosed in angle brackets, such as <OK>.

Since there is sometimes more than one way you can type a special character, and since the key labels may differ on different models of terminals, the textbook uses a standard notation for each special character. It uses *only* this notation; except for occasional references to buttons on the mouse, it does not tell you exactly what to press. To learn what keys or mouse buttons to press at the standard AUGMENT display terminal, see the table in Figure 1.

NOTE: This table also includes some key labels that appear on older models of the standard AUGMENT display terminal. If you will be working at a nonstandard display terminal, see the corresponding table in the document on how to use that particular terminal, and if you will be working at a typewriter–like terminal, see the lessons written about using AUGMENT in typewriter mode.

The first column of the table in Figure 1 shows the notation we use. The second column lists the keys or mouse buttons you would press to produce the character represented by the notation or describes what you would do. Where applicable, the third column shows another special character that normally will have the same effect as pressing the indicated keys or buttons; note, however, that in some cases it is possible to change the equivalent character from the one shown here.

Refer to Figure 1 when you encounter unfamiliar notation in a lesson. You do not have to understand the functions of the special characters or how to use the mouse at this point; you will learn about these in detail later.

Also for convenient reference, the table in Figure 2 shows the buttons on the mouse and the functions and notation for the characters you get when you press those buttons. The circles represent the three buttons; filled–in circles represent buttons pressed. You will learn how to use these functions later.

FORMAT OF EXAMPLES

Most examples of doing work in AUGMENT 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 AUGMENT, such as prompts telling you what you can do next.

NOTATION	KEYS OR BUTTONS	EQUIVALENT
<BC>	BACK SPACE CHAR, BACK SPACE, or left mouse button	<CTRL-A> or <CTRL-H>
<MARK>	Point with mouse and press right mouse button or OK key	
<BW>	BACK SPACE WORD or left and middle mouse buttons	<CTRL-W>
<CD>	CMD DELETE or middle mouse button	<CTRL-X>
<CTRL-x>	Hold down CTRL while typing the letter x in either lowercase or uppercase (where x may be any letter)	
<ESC>	ESC or left and right mouse buttons	
<HELP>	HELP	<CTRL-Q>
<INS>	INSERT or INSRT	<CTRL-E>
<LF>	LF or LINE FEED	<CTRL-J>
<LIT>	LITERAL	<CTRL-V>
<LOCAL-RESET>	LOCAL RESET	
<NULL>	NULL	<CTRL-N>
<OK>	OK or right mouse button	<CTRL-D>
<OPT>	OPT'N or OPTION	<CTRL-U>
<RC>	REPEAT CMD or middle and right mouse buttons	<CTRL-B>
<RET>	RETURN	<CTRL-M>
<SP>	Space bar	
<TAB>	TAB	<CTRL-I>

FIGURE 1: SPECIAL CHARACTERS ON THE STANDARD AUGMENT DISPLAY TERMINAL

BUTTONS	FUNCTION	NOTATION
○●○	Command Delete	⟨CD⟩
●○○	Backspace Character	⟨BC⟩
○○●	OK	⟨OK⟩
●●○	Backspace Word	⟨BW⟩
○●●	Repeat Command	⟨RC⟩
●○●	Escape	⟨ESC⟩

FIGURE 2: MOUSE BUTTONS

In either column of an example, when what you type or see cannot fit on one line, all lines beyond the first are indented slightly to show that they are continuation lines. The line length in the example depends on the width 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 AUGMENT. You may ask to receive a different type of feedback, more suited to your needs; in this case, 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, AUGMENT accepts the text exactly as you type it, but when you type a letter of a word in a command, case is ignored.

Occasionally, when the details of the interaction are not important to the current discussion, examples are shown in a more concise way. The information that would otherwise appear in two columns is combined into one, showing the words in the command, some of the feedback you get, and the responses you type, in the order you would see them.

FORMAT OF THE LIST OF COMMANDS

Every AUGMENT command discussed in a lesson is listed in the "List of Commands" section. Similar commands are grouped together. For each command, we show all the steps or choices of steps you have learned in the lesson, in the order of their occurrence in the command, and we may also list some easy extensions to what was discussed. The following paragraphs describe how we represent the steps in these commands. This description is included here for your reference when reading the "List of Commands" sections; it contains some special terms that you will learn later.

A step in a command may be something you type or something AUGMENT displays. Some simple steps and the way we show them are as follows:

> A "command word", which you begin and AUGMENT recognizes and finishes, is represented by a word with only its initial letter capitalized, such as: Delete
>
> "Noise words", which AUGMENT displays in parentheses to help you along, appear in parentheses, such as: (to be named)

Special characters that you type are represented by their standard notation, within angle brackets, such as: <OK> (See Figure 1.)

At some points in commands, we use uppercase words to represent what you can do and then define or describe those words in a table at the end of the commands list. Often the word represents choices you have at that point in the command; it is like a "variable" for which you may select any of the choices listed for it. In the table defining these special words, we use some of the same representations as discussed above. For example:

LOCATION	Prompted by "M/A:" For M you may <MARK>. For A you may type an address, ending with <OK>.
STRUCTURE	Statement or Branch or Group or Plex

The word "or" indicates that you may choose one or the other. Slash (/) also means "or". As shown in the example above, the prompt you get for LOCATION tells you that you have a choice; our definition describes what to do in either case and includes some special characters (which you can look up in Figure 1). The choices listed for STRUCTURE are command words; that is, you may choose one of those command words where STRUCTURE appears as a step in a command.

Square brackets enclose steps in a command that you may skip completely or may choose by first typing the special character <OPT>. For example:

Force (case) [Upper (for)] STRUCTURE (at) LOCATION <OK>

means that you can either omit the bracketed option, as in:

Force (case) STRUCTURE (at) LOCATION <OK>

or include it as follows:

Force (case) <OPT> Upper (for) STRUCTURE (at) LOCATION <OK>

Price $3.00

INTRODUCTION TO AUGMENT

Copyright Tymshare Inc., May 1980
20705 Valley Green Drive
Cupertino, California 95014

This document was prepared and composed for printing with AUGMENT by the Office Automation Division of Tymshare, Inc.

AUGMENT Journal Number 48746
Tymshare Document Number 1838

This introduction is for people who will be learning AUGMENT, to give them an overview so they will have a general idea of what their experience will be like. It also describes a number of the tools AUGMENT offers, so that prospective users will know the many ways AUGMENT can help them in their work. People who will not themselves use AUGMENT, but who will be working with people who do, may also benefit from reading this introductory information.

The introduction must serve several types of readers. Some are readers who know very little about computers; they will find helpful general information in the BACKGROUND section. Readers who know more about using computers may want to skip that section. Note that this introduction does not provide technical details about computer operation or programming.

CONTENTS

BACKGROUND

Computers help you mold and move information.

A computer is an electronic device that can accept information from a user, work with the information, and provide results. Computers provide us with an amazing range of tools. Whether you want to solve a math problem, reserve a seat on an airplane, or write a memo, the computer offers a tool that helps you do what you want done. AUGMENT is a set of tools that enables people who work with textual information to use the power of computers. In the case of AUGMENT, it is most important that computers can store text the way filing cabinets or books do and then help you freely copy or change it and deliver it to others. AUGMENT services include access to a computer, equipment that displays information and what you do with it, and the introduction of new skills and working methods to help users take advantage of these tools.

It is worth stopping for a moment to consider what it means to say that computers offer tools. People have been using tools since the days of stone axes; with tools we have been able to build beyond the strength of our bodies, understand things we cannot observe, and create systems that free us from menial tasks.

Many tools from one source

Some inventions have opened up the possibility of a wide range of related tools. For example, we speak of the invention of the wheel, but when man began to harness rotation, we were able to have not only wheels, but engines to turn them, mills, drills, watches, and so on. Like rotation, computing is now the basis of many tools.

Familiar control of complex systems

Some very complex tools have been made to feel simple and familiar to the person using them. A car is a complex system with thousands of parts and a very subtle design, yet we can use a car with one steering wheel, some pedals and knobs, and a few weeks training. Moreover, the car is really only part of a more complex system, including roads, gas stations, street signs, and so on, that we employ casually.

People have learned how to apply a computer's power in many ways that have changed the world, but they have only begun to understand how to make a computer simple to use. At Tymshare's Office Automation Division, we have been working for many years to make computers simple to use. We have succeeded enough to put in your hands more power for many kinds of work than you can get any other way.

AUGMENT, a versatile system of tools

Although computers have the potential for many uses, people often adapt them for a narrow purpose, such as producing a certain kind of billing or taking certain kinds of reservations. At the Office Automation Division, we harness computers for the large number of jobs related to working with text. AUGMENT was not designed to solve a specific problem, but rather to help you effectively accomplish many everyday tasks, such as planning and writing documents, exchanging memos and ideas with co-workers, and organizing thoughts and information. In this sense AUGMENT is not one tool itself, but a

"system" of coordinated tools like the system that makes highway transportation possible.

Of course, "everyday" tasks vary from one project to the next, as well as for individual employees, departments, and organizations. Often, you may not even know in advance what kind of help you will need when you are thinking, writing, or trying to find information. AUGMENT is built to accommodate diverse and unexpected needs. This diversity is the main reason it is not possible to describe AUGMENT in a single document or for you to learn it in a single training session; you must read appropriate documents or receive additional training as you apply AUGMENT to new tasks.

Commands for different tasks work the same way.

Computers can only respond as they have been designed to respond. In order to get the most from AUGMENT, you must learn what the computer can and cannot do for you, and you must learn the instructions or "commands" to give the computer to make it do what you want. We have designed simple commands to control what the computer does, and there are many such commands to correspond to the many different functions that you need for your everyday work. However, we have carefully designed the different commands to work in the same way as much as possible so that someone who knows how to use AUGMENT for one job can easily learn what it takes to do another. Once you learn the basics, extending your use of AUGMENT will become like using any of the other familiar tools you employ for your daily work.

WHAT IS IT LIKE TO USE AUGMENT?

Viewing Information

For the most part, users of AUGMENT do their work on a television-like screen. Although you can get your information in printed form, the main way you see what you are doing is by looking at the screen.

Seeing less to see more

AUGMENT enables you to rapidly change how you see information, giving you a repertoire of special views which take the place of thumbing through a book and which expand your power to skim information. For example, most AUGMENT files are arranged with headings and subheadings in an outline format, and you may display outline numbers on your screen or not, at the touch of a key; this is a change of view. With equal ease you may display only the major headings, for example, the chapter headings of a book you are scanning on your screen; this is another change of view. Or you may show only paragraphs that contain certain words.

These view changes, aimed at making up for the small window you have into your large world of information, supplement and extend your power to skim and pick. They not only help you with reading and searching for information, but also make it easier to do other common types of work with files, such as writing, editing, and updating. The ability to change views is an important difference between AUGMENT and other text-handling systems.

Saving, Finding, and Sharing Information

Whenever you add or copy information in AUGMENT, the information goes into a "file", a work area in the computer like a file folder in a filing cabinet. There are no odd bits of paper (or diskettes or cassettes) to keep track of. You may put the information in an existing file or name a new one. These files are safe, easy to find, and easy to share.

Safe

When they first begin to work with a system like AUGMENT, most people tend to print paper copies of their work because the version in the computer seems unreal and unreliable. But while paper can be lost or damaged, the file system is always orderly, and your files are safe even though a system problem may occasionally keep you away from your work. Furthermore, the files are regularly copied to magnetic tape, so you know they have been duplicated in storage separate from the computer. Soon most people begin to feel that the electronic record is more reliable. For example, having made notes at a meeting, they want to hurry and get the notes on the computer where they will be safe; then they can throw the paper away.

Accessible

At any time, you can see on the screen a list of the names of files you have made, files that belong to others (if they permit it), or files created for public reading. This list may be alphabetic, reverse alphabetic, in order of writing or reading, or one of several other orders you may choose. You can learn the last time you looked at a file, the

size of the file, and other information to help you keep your files organized. You may draw one or more files by name onto your screen and then do the equivalent of thumbing through them by using the viewing tools discussed above.

Easy to share

Information you are working with resides on a computer that you share with other members of your organization. This means that, if the person who controls the information allows it, you can look at someone else's file as easily as your own, regardless of where in the world he or she happens to be. (See Figure 1.) Different organizations use sharing in different ways. Organizations often collect files into special "directories", which can be thought of as the computer's equivalent of filing cabinets, for sharing purposes. The power of the system to oversee and select a wide range of information gives you a very special feeling of being in touch with what you need to know to do your work.

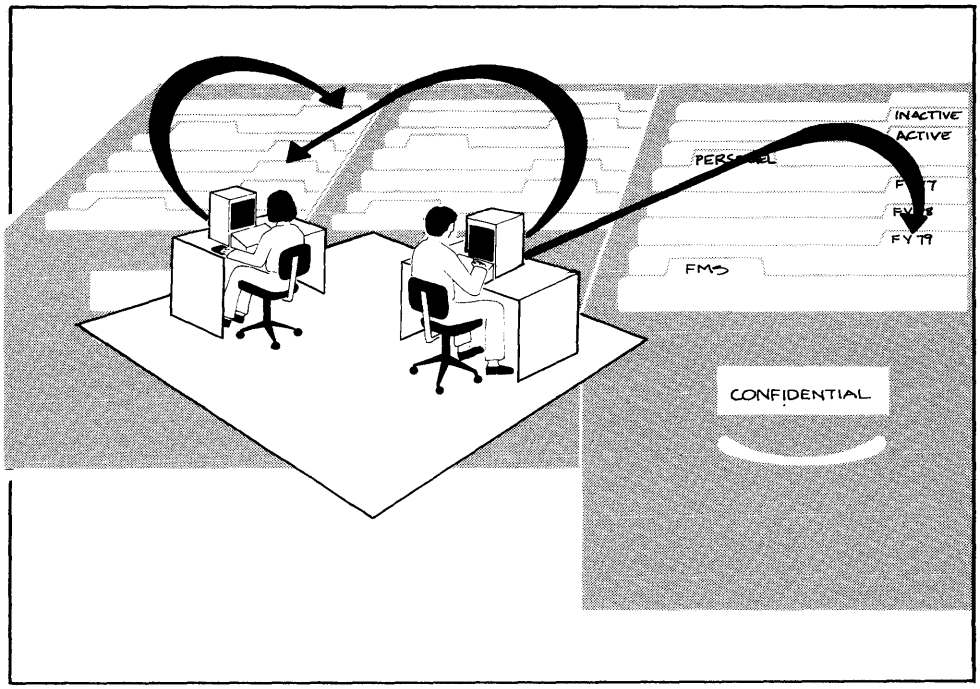


FIGURE 1: SHARING FILES

What is Learning AUGMENT Like?

Learning step-by-step

For most people, learning to use a powerful computer system like AUGMENT is like learning tennis or learning to drive a car. You flounder in the first hour; some hours of practice are necessary before you begin to feel comfortable with it; and you can continue to learn refinements for a long time. For people familiar with word-processing systems, the learning time in AUGMENT is about the same as the learning time in a typical word-processing system *for the same function.*

Of course since AUGMENT has more functions, people who employ those functions go on learning. Once you have mastered some basic set of AUGMENT skills, learning new functions goes faster because the general form of the command is always the same.

Some of our users apply AUGMENT in very particular ways, for example, bulk typing, keeping up a calendar, or sending messages. Most people can become proficient at such applications in a few hours.

How is Writing Different?

Copying from past work

All systems based on electronic memories make it easier to write, in the sense of producing a lot of pages, because they make it easy to copy. Typical word-processing systems make it easy to copy pages from a modest-sized collection of previously written work, whereas a system like AUGMENT, based on a large, flexible file system, makes it easy to copy selectively from *years* of past work.

The number of times a finger presses a key in proportion to each character finally printed is a measure of efficiency in the publications business. Four keystrokes per character is common where typewriters are used, for example, once when originally typed, once when the draft is cleaned up, once in response to the boss's suggestions, and once for camera-ready copy. A good word-processing system, where only revisions are retyped and camera-ready copy is automatic, commonly reduces this figure to two keystrokes or slightly less.

Less than one keystroke per character

For many routine reports, memos, or proposals, the figure in AUGMENT is less than one keystroke per character since more material is copied than revised. As you become experienced at writing in AUGMENT, you will learn to effectively use your searching tools (catalogs and viewing tools as well as automatic searches for content) and to make revisions carefully to ensure that copied words make sense in the new context.

Working from the top down

As discussed in connection with viewing, AUGMENT files make it easy to write in outline format. Much business or technical documentation lends itself to working "from the top down", that is, jotting down the major headings, then jotting down the subheadings, and then filling in the contents under them. AUGMENT supports such writing very effectively. Often you jot down a heading and discover you can copy in the body of the text.

Freedom to experiment

In general, AUGMENT makes it easier to experiment, not only with words and phrases, but with organization. It is easy to change a word or a paragraph or the overall organization of a document, see how it looks, and then change it back if you wish.

Communicating and Distributing through Electronic Mail

Electronic mail as a part of AUGMENT takes the place of some telephone calls, allows you in effect to leave notes on the desk of people scattered all over the country, simplifies distribution of notes, memos, reports, and

book-length documents to people on the system, and, if you wish, stores and retrieves these documents as a librarian would.

Comparison with telephone

The telephone calls AUGMENT replaces are those that do not require you to talk back and forth much. AUGMENT mail has several advantages over telephone calls: You never fail to reach the other party, you do not interrupt anyone's work, and you may file a written record automatically.

Easy distribution

The size of an item makes no difference to the system, up to several hundred pages. It is possible to transmit copies of a long document as easily as a brief note, to one person or to a list of people, with just a few keystrokes.

Automatic filing

Unless you mark it "private", the system will file and index each item by words in the title, the author, and other filing keys. Thus an individual and an organization build up a body of past information. At any time you can look in catalogs and locate (usually within a day for old items) all the documents relating to a topic, written by a particular person, or prepared on a certain day. The result of this automatic cataloging is that individuals depend less on private files (because the system is their filing cabinet) and spend less time keeping them up. The organization has a better chance to pool knowledge and make it available to new people.

Changing How You Work

How you adjust to working with a computer depends a great deal on your job, your work habits, your personal style, and the style of your organization. In general, however, it replaces pencil, typewriter, file drawer, desk calculator, and most of the tools associated with an office desk and filing system, plus mail and some of the functions of the telephone and the library. All of these are replaced with something that is swifter and smoother.

Changing the shape of your work

Like other people who have used computers to help them do their regular work, we have found that computers usually enable us to finish faster, and they also change what we do to something larger and more valuable. Often, as people become experienced in AUGMENT, they find something they can do which improves their function but which they did not think was appropriate before. Secretaries may find they can schedule meetings that only their principals could schedule before, managers may find they can skim reports that were totally unread before, and engineers may find they can communicate with associates through electronic mail where they previously worked in a vacuum.

Changing how you use paper

Just as AUGMENT users will in general learn to do their work in different ways, they will use paper in new ways. Most users who work on a screen can produce paper copies of their work with adequate printing quality quite easily, through either printers attached to their work stations or fast central printers. Most people who can easily print make clean drafts frequently, for review by themselves or by others. It

is possible to scan and edit a cleanly printed draft in a way that is not possible either with an old, marked-up draft or with a display screen. Tools exist for setting up simple formats, as for a letter, or complex formats for more formal publications, such as a finished book. AUGMENT tends to turn users from people who make notes and revisions on paper into people who use paper for clean drafts and for distribution to those who do not use AUGMENT.

Changing How You Use AUGMENT

Expanding your capabilities

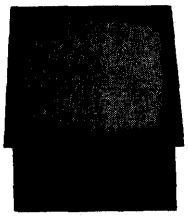
Normally people begin using AUGMENT for one function or only a few functions. For example, managers might use it solely to survey information, exchange messages, and watch their calendars; specialized clerical people, to type smoothly a certain type of information; others to edit and produce some document in one format; and others to search old files and compile new documents based on old. Many people continue in that way, while others, through need or interest, expand their knowledge. The learning process is like acquiring more support systems, in that it often involves expanding your job as well as helping you do it.

Doing more of your work in AUGMENT, or developing new work in AUGMENT, may change how you perceive the system. For example, when you begin editing, you may feel that AUGMENT has many extra commands. As your skill grows, you will eventually use these other commands.

THE AUGMENT SYSTEM

AUGMENT was designed so that you can use the computer comfortably as well as effectively. The instructions that you give to control the system and the responses from the computer have been carefully planned to provide you with as much information as possible without wasting your or the computer's time.

Work Stations



Display

You give your commands to the computer at a "work station". Generally speaking, you use AUGMENT by typing on a keyboard, pointing by moving a mark across a screen with a controlling device, and watching the results on the screen. These pieces of equipment are all part of your work station. A work station may also include a small printer to make paper copies.

The "display" is the television-like device on which you can see the commands that you give to the computer and the computer's responses to your commands, and where you can read documents and information stored in the computer.

The AUGMENT display screen is carefully arranged into different areas, each with a special purpose and each able to change independently of the others. As shown in Figure 2, an area near the top of the screen shows the command you are giving and the response you get from the computer, while a large area below that shows the information you are working with. You can further divide the lower area into several parts so that you can look at a number of different files or at different parts of the same file.

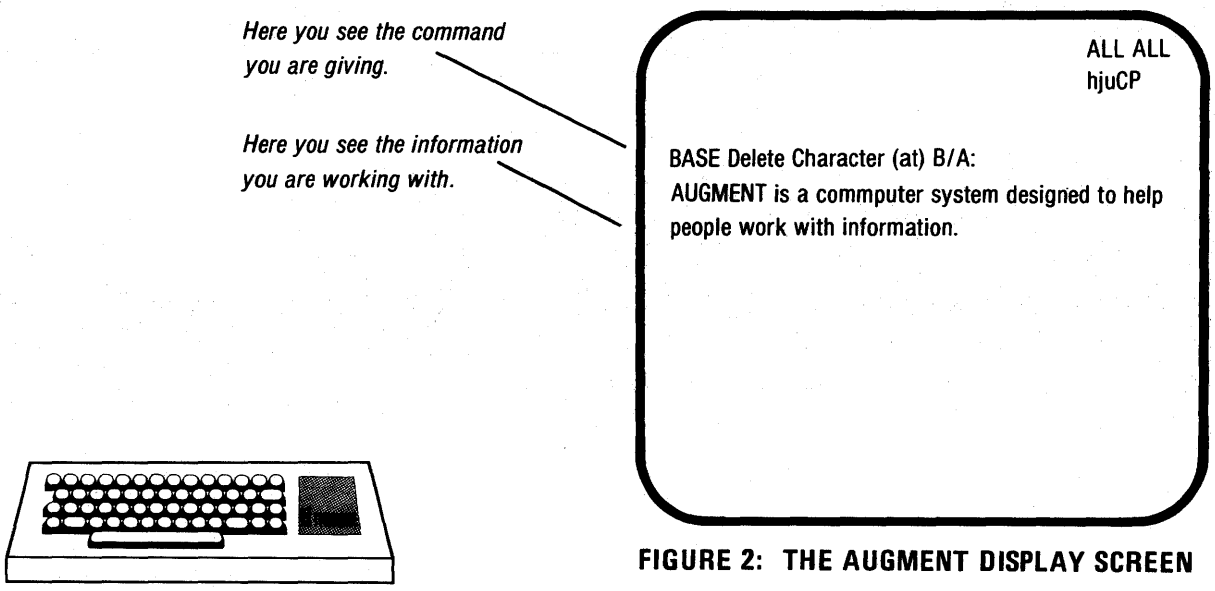
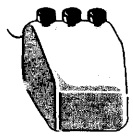


FIGURE 2: THE AUGMENT DISPLAY SCREEN

Keyboard

You type text or enter commands with the keyboard. In the standard AUGMENT work station, the keyboard is detached from the display for convenience. In general, it resembles a typewriter keyboard; it has the standard alphabet, numbers, and punctuation found on a typewriter as well as special AUGMENT function keys.

Pointing with the mouse



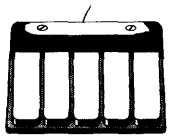
Standard AUGMENT work stations include a "mouse", a hand-sized device with three buttons on the top. When you are working in AUGMENT, you can use your right hand to roll the mouse on the table, moving the "cursor" (the traveling mark) on the display screen correspondingly. The mouse enables you to point directly to what you want to do something to, rather than describe its location. For example, if you want to give a command to remove a particular comma displayed on the screen, you can use the mouse to point to the exact comma. You can also point to words, sections of text, paragraphs or headings, and groups of paragraphs or headings. The buttons on top of the mouse can be used alone or in combination with the keyboard or the keyset (described below) to enter commands and information in AUGMENT.

Other kinds of cursor control

NOTE: We have adapted AUGMENT to some display units other than the standard AUGMENT display so that you can employ the cursor control normal to that unit. For example, instead of pointing with the mouse, you might point by pressing keys on the keyboard, such as special "cursor keys" with arrowheads indicating the direction in which the cursor will move.

Keyset

Standard AUGMENT work stations offer a device with five piano-like keys, called a "keyset", which you can use as a supplement to the keyboard if you wish. The keyset enables you to type commands quickly with your left hand only, while using the mouse with your right hand.



Typewriter work stations

Another common type of work station consists of a device like a typewriter for entering and viewing commands and information. You do not see the results as fast and flexibly as on a display screen; however, you may prefer to use a typewriter work station at times because the equipment is portable and inexpensive and will give you a written record of what you have done. You can use AUGMENT in typewriter mode much the same way as in display mode. The two modes are as similar as possible, to make it easy to switch from one to the other; for example, the commands in typewriter mode are worded the same as in display mode.

Commands

AUGMENT is an "interactive" computer system, which means that you and the computer work together. You give a command, the computer follows the instruction, and you see the result. The computer is then ready for you to give another command; the next command you give may depend on the result of your previous command.

Simple English verbs and nouns

AUGMENT commands use simple English words, and their basic, consistent form makes it easy for you to figure out what to do next. Most commands begin with a verb that names the action, such as Delete or Insert, followed by a noun that names what to do the action to, such as Character, Word, or Text. We have designed the commands this way so that they are like simple English sentences telling someone to "Do this" or "Put that there".

Commanding the computer

You enter a command word usually by typing only the first letter of the word, for example, "d" for "Delete" and "c" for "Character". (See Figure 3.) Occasionally you have to type two or three letters, but you never have to type the full word. The command words always appear in full on your screen, so you can easily check to see whether you have typed the desired command.

Delete

Character

FIGURE 3: LETTERS FOR COMMAND WORDS

Helpful prompting

When you enter a command, AUGMENT helps you fill it out by giving you prompting that helps you understand the command and indicates what you can do next. For example, after you type "dc", you see "Delete Character (at) M/A:". The "(at)" tells you that in the next part of the command you will specify where the character is that you want to delete. The "M/A:" means that you can either use the mouse to point to the character and "mark" it or give an "address" describing the location of the character.

The last step in any command is to type a confirming or "OK" character; this tells AUGMENT to go ahead and carry out the command. If you change your mind any time before you give the final "OK", you can easily cancel the command and start over again.

You don't have to know how to type.

Many AUGMENT users type more slowly and less accurately than the average typist. The system forgives typing errors and allows you to backtrack in commands, and the editing, copying, and spelling checking features help you to be productive without being a good typist.

Questioning a command

AUGMENT was specially designed so that you can learn about commands as you work. The simplest way to do this is by using question mark (?). Any time you are working with AUGMENT (except in the middle of typing text), you can type a question mark to see a list of all the choices you have at that point, such as all the command words you can enter.

File Structure

All the information in AUGMENT files is structured into an outline form. The basic units of information in a file are normally headings or paragraphs, although the user is free to choose how to employ the outline structure. As it normally appears on the screen, indenting shows the structure, and with one of the view changes discussed above you can ask to see outline numbers.

The power of structure

The structure of AUGMENT files helps you organize your thoughts and your documents, and also enables you to locate precisely what you need from a large storehouse of information. The storage area of your computer contains thousands of books worth of information; this information is divided into directories that belong to people or organizations by name, and into files named by the users. The files are further divided into outline form. This strict organization allows you, for example, to copy any paragraph from any other file to your file with a few keystrokes if you have proper access. It also allows control of which users are allowed to read or write on which files. Provisions exist for reaching information stored on other computers.

No fixed lines or pages

People who are familiar with word-processing systems, or with systems based on central computers that address their files by lines, should understand that lines are not a fixed unit in AUGMENT files. The lines you see are arranged to fit the screen or printed page as the information is on its way from storage to your display or printing device. No AUGMENT file is formatted into pages until it is printed. The fact that lines and pages are not fixed allows a great deal of flexibility when it comes to displaying files or to printing the same files on different devices.

Subsystems

AUGMENT is divided into subsystems, which are sets of commands related to particular activities. Normally, the subsystem that is automatically available when you enter AUGMENT is the Base subsystem. It contains the most common commands for doing your everyday work, such as reading, writing, editing, printing, and organizing your files. You can switch to other subsystems to do other common types of work with the same files.

AUGMENT includes an electronic mail subsystem which you can use to send messages and documents to a list of people known to AUGMENT and have them cataloged and stored in the AUGMENT Journal. (See Figure 4.) In a file they see whenever they enter AUGMENT, the recipients will receive a notice of what was sent and a way to get to it immediately, or the item itself if it is short. Mail items can easily be obtained in printed form.

FIGURE 4: ELECTRONIC MAIL

Here are some of the other commonly used subsystems:

Graphics. The Graphics subsystem enables you to create, display, modify, and print line drawings that are stored in AUGMENT files with text. To use Graphics, you need a Graphics work station, consisting of an AUGMENT work station and a Tektronix storage tube display or the equivalent. You can create Graphics diagrams separately or in their appropriate place in a document. They can be printed on a photocomposition device or on a special printer attached to the Graphics work station.

Proof. The Proof subsystem presents pages as closely as possible to how they would appear when printed via a photocomposer (for example, with proportionally spaced letters), to allow preliminary proofing of formats that require graphic arts production (such as a change of type faces or integral illustrations). The Proof subsystem will display the

layout of the page correctly but the type font only approximately, and it will work only on suitable high-resolution display work stations.

Publish. The Publish subsystem provides a number of commands that support document production, such as commands for automating simple punctuation, counting words, establishing the reading grade level, and doing limited indexing.

Spell. Spell is a subsystem that checks AUGMENT files for doubtful spellings. When it finds a word it does not recognize, you can correct the word or accept the word as it is. Whenever possible, Spell will suggest one or more correctly spelled "guesses". Spell keeps track of the words you have corrected, and subsequent occurrences of identical errors will be automatically corrected. When your file contains correctly spelled words that Spell does not recognize, you have the option of inserting them into a dictionary that Spell will use to check the rest of your file. You can build that dictionary into a personal or organizational jargon dictionary.

Table. The Table subsystem allows you to set up tables, edit them in a tabular manner, and do simple arithmetic where extensive computation is not required. You can define a screen-sized part of your file as a table with headings, columns, rows, subheadings, and so on, and then enter information into it without using tabs or typing a lot of spaces. You can edit rows and columns (for example, insert a new row or replace one column with another) and perform simple arithmetic on parts of tables (such as automatic totalling of a row or column, or multiplying columns and storing the results).

Retrieve. The Retrieve subsystem takes advantage of AUGMENT file structure to find and copy information in sophisticated ways. You use it, for example, if you want to get copies of all the paragraphs under certain headings (but not the headings) or if you want all the headings that precede certain information (but not the information).

Conference. The Conference subsystem allows two or more users of any of Tymshare's AUGMENT service computers to display the same information on their screens. They can each point to items separately, and a protocol exists to allow orderly control of commands in making changes.

Encrypt. The Encrypt subsystem allows you to specify a key and then transform your file into a coded image that looks like nonsense characters. Only someone who knows the key can read the file from then on.

Programs. The Programs subsystem contains commands for adding to AUGMENT anything from a simple program, such as one to modify statements containing particular forms of text, to a new subsystem. You can write the program or subsystem in Base and then compile it in Programs. The Programs subsystem also provides access to a powerful debugging tool, and there are several commands for handling special

programming needs. Additional subsystems exist for other programming functions, such as library maintenance of source and compiled code.

New subsystems are constantly being created, some for wide use, others for specific purposes. Subsystems can be created to give AUGMENT users access to programs on other computers.

An Example of Applying AUGMENT

All these capabilities and subsystems are like an assortment of tool boxes that can be used together on some large project, as building a house may require carpentry tools, plumbing tools, electrical kits, and so on. With some office work, too, no one single-purpose computer system can help much; the broad usefulness of AUGMENT can be put to work in these cases.

Let us take, for example, the case of a team of people using AUGMENT to both gather information and present it in final printed reports. Every tool they need to produce camera-ready copy of the reports is available through AUGMENT. Each person may research information stored and cataloged in the computer, use the system to keep notes and organize material, and communicate with co-workers through the computer (for example, by sending memos that are permanently recorded and cataloged, sharing files, and even "talking" to others by means of the keyboard).

All drafts of the team's reports are easily written and edited in AUGMENT. Any portion of text can be moved or copied within or between files (automating the traditional cut and paste technique) and tables and graphic illustrations can be incorporated into the document. The writers can combine their separate contributions into one file and then use the same file. Several tools enable them to control the document; for example, they can learn the name of the person who made the last change to any paragraph and the date and time the change was made.

A draft at any stage can be permanently recorded in separate storage and cataloged. When the final draft is ready (or at any other time in the life of the document), a writer may experiment with formats for a printer or phototypesetter and may proof photocomposed pages with approximate fonts and type sizes on a graphics display terminal. When the team is satisfied, the document can be sent from the computer to a phototypesetter. The report in its machine form can also be transmitted to or shared by people in other locations having a computer connection to the computer on which it is stored.

DOCUMENTATION

We offer four types of documentation to give you the information you need to do your work and to learn more about what you can do with AUGMENT: the AUGMENT Textbook Series, online help, quick reference cards, and special-purpose documentation.

The AUGMENT Textbook Series

The AUGMENT Textbook Series consists of individual "lessons" which are modeled after foreign language textbook lessons. They address a wide range of users, from people who know little about computers to people who have a great deal of computer experience. The lessons in the Textbook Series give not only the bare bones information, but also a feel for how users can be more effective in their jobs with these tools. Besides the basic skills for all applications of AUGMENT, most lessons teach specific functions, such as editing, printing, setting up tables, and formatting. We believe that the lessons provide a sound basis for learning and cover most of what people need to know to avoid pitfalls and confusion, but they do not include all the details of AUGMENT or seldom-used commands.

Help as you work

Help while you are working on the computer ("online" help) comes in the form of the Help command. This command, available in every AUGMENT subsystem, provides the most complete and up-to-date information about all aspects of AUGMENT. Typing "h" for "Help" and then any AUGMENT term (followed by the usual "OK" character) produces a description of the term, or information pointing out different uses of the term which will lead to a full description of it and to other, related subjects. Users have occasionally learned the system through the use of Help alone.

We make every effort to keep the Help information complete and current. It corresponds to the lengthy and hard-to-read reference manual that goes with many computer systems. We have chosen to maintain the complete reference documentation about AUGMENT on the computer rather than in printed form because it is more quickly accessible by users and easier to keep current. For every new release of AUGMENT that contains changes visible to the users, we update the Help information.

Quick reference cards

We also provide "quick reference cards" showing commonly used groups of commands and features. They are concise and give you a handy way of reminding yourself about how commands work that you have already learned.

Specific topics

Finally, we have various other documents on specific tools or functions, such as: instructions on how to use AUGMENT with certain equipment; a manual describing the "Output Processor" formatting codes for making formatted documents; and guides to help you with particular applications of AUGMENT.

SERVICES FROM TYMSHARE'S OFFICE AUTOMATION DIVISION STAFF

Part of what you get with your AUGMENT service can be help from people. There are many experienced, interested, friendly people at our headquarters and field offices who enjoy helping you to learn AUGMENT and use it better.

Tymshare's Office Automation Division has a staff of trainers, analysts, and programmers experienced in AUGMENT and issues of office automation to give you further help if needed.

Training

If you are reading this, you may be taking a training class shortly. To accommodate the needs of different users and of users at different stages of development, Tymshare offers several types of training given separately to different homogeneous groups. For example, administrative support staff can get a training package that is suited to their particular tasks, uses of the system, and familiarity with the system environment.

Tymshare's training program is designed to raise the user's level of expertise in several stages. Our initial training program is designed for proficiency in fundamental system knowledge and skills, related to some small useful applications. It includes actual classroom instruction, both with and without hands-on experience, directed practice on material presented in class and tutorially, and individual coaching and assistance. The second level of the training program covers advanced topics and skills and is oriented to the applications of specific user groups. You will find that continuing training in the actual details of performing the application tasks, and in further application capabilities, is very important to ensure that users can fully use the power of the system and make it effective in their organization.

Consulting

Analysts are able to study your work and make suggestions about how you might use AUGMENT more efficiently, or to discover applications well suited to AUGMENT. They can also help you organize and implement the people, policies, and procedures that make up the planned application activity, in these ways: by recommending which people should be trained in which function, what equipment should be installed, and where to install it; by specifying the steps in the procedures needed; by helping individuals get started; and by advising on the timing of a staged transition or expansion.

Special programming

Tymshare provides programming to support the special needs of organizations. For example, although AUGMENT is not appropriate for voluminous computation, it is often convenient to have commands modeled on an organization's financial procedures, because with AUGMENT the results are easy to integrate into documents. The differences between business and accounting methods in different organizations usually mean that the commands good for one group are inappropriate for another. Building a special subsystem upon basic AUGMENT procedures, we can easily suit commands and high-level functions to a given organization. (See Figure 5.)

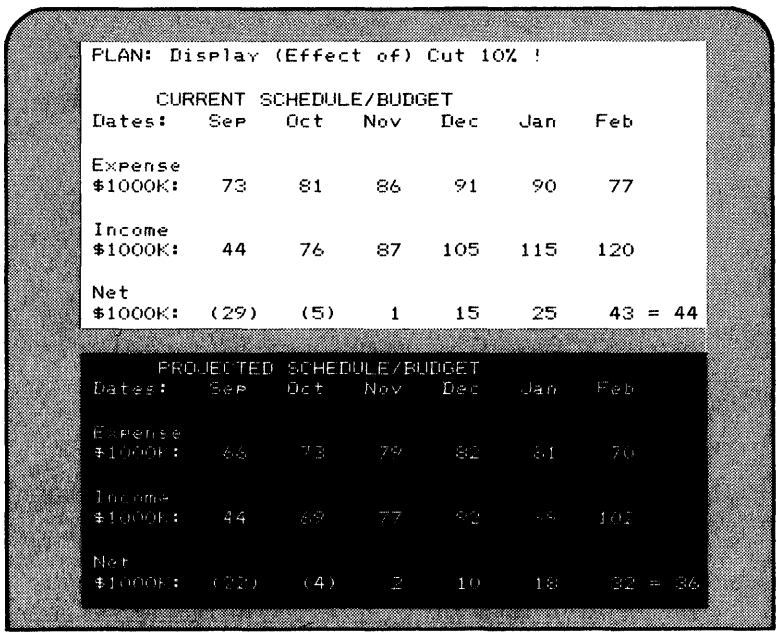


FIGURE 5: SPECIAL-PURPOSE SUBSYSTEM

Sending messages to Feedback

We also offer a way to get expert help personally through the system. Anyone using one of Tymshare's AUGMENT service computers may report bugs, questions, their difficulties with the system, and suggestions about improvements on the system to the Feedback mechanism by using our electronic mail capabilities to send a message to the name FEEDBACK. They may also ask Feedback about AUGMENT status, current plans, and the decision status of other users' suggestions. All messages are normally answered within one working day. If a problem cannot be fixed immediately, an acknowledgement is sent indicating that a specialist will handle it as soon as the necessary resources are available, and any further developments are reported back. All other inquiries that need to be studied, such as suggestions, are promptly acknowledged.

Operations Control Center

The experienced staff of the AUGMENT Operations Control Center manage Tymshare's AUGMENT service computers. At the Control Center, computer system experts constantly monitor all aspects of the system for correct performance. Users can address their needs and questions to the Control Center through Feedback.

SUMMARY

Most people who read this lesson will be learning to use AUGMENT in their work. Different people will have different expectations. The expectations may range from eager anticipation to a feeling that it is foolish to suggest changing a way of working which has been learned well to a new method which introduces new complications. We are keenly aware that once an organization begins using AUGMENT, the most important factor in its success or failure is the attitude of the users: how you feel about it.

This introduction has attempted to give you some familiarity and confidence before you begin learning and working in AUGMENT. We have pointed out that AUGMENT is the fruit of years of effort to make the power of computers available to ordinary people in a way flexible enough to accommodate a range of needs in an organization and to permit growth and learning for individual users. We have indicated that the work station and command language are designed to give you easy access to a large amount of information, with commands that are like English sentences and that keep the same form for various functions. We have tried to put across what it feels like to use AUGMENT in your work and to describe some of what you will encounter later on. Remember that behind this powerful system are a group of experienced people at Tymshare's Office Automation Division who want to help you grow in productivity and enjoyment of your work.

Price $3.00

BEGINNING USE OF AUGMENT

Copyright Tymshare Inc., May 1980
20705 Valley Green Drive
Cupertino, California 95014

This document was prepared and composed for printing with AUGMENT by the Office Automation Division of Tymshare, Inc.

AUGMENT Journal Number 48747
Tymshare Document Number 1823

This lesson discusses basic concepts and commands you need to write and edit in AUGMENT, 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 AUGMENT. You should already know how to log in and enter AUGMENT, as described in the lesson "Starting and Ending an AUGMENT Work Session".

CONTENTS

INTRODUCTION

AUGMENT 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 reading, writing, and editing documents to sophisticated methods for retrieving and printing information.

Why use AUGMENT?

Using AUGMENT to write and edit has many advantages over working with paper, pencils, and typewriters. With AUGMENT 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 learning AUGMENT by learning the basic writing and editing commands. Although this is only a small part of what AUGMENT 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 AUGMENT will automatically adjust the text so that the document will look as if it had been 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. When you are working in AUGMENT at a standard display terminal, you can use the mouse to point to what you are talking about. If, for example, you want to take out a particular comma on the screen, you can simply give a command and use the mouse to point to the comma you want removed. You can also use the mouse to point to words, pieces of text, paragraphs or headings, and sections of a document.

The Base subsystem

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

Working with files

In AUGMENT, you will always work with material in a "file", a work space on the computer much like a file folder in a filing cabinet. This lesson will teach you how to use Base subsystem commands to write and edit a simple file containing a business letter.

WINDOWS ON THE DISPLAY SCREEN

You start an AUGMENT session by logging in and entering AUGMENT. When you enter AUGMENT, you will see information displayed on your screen. The screen is actually divided into four areas called "windows", as illustrated in Figure 1. The information displayed in each window has a separate function.

status window: This window displays messages from AUGMENT 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 "Writing and Reading an Organized File".

command window: As you give a command, it is displayed in this window along with information to prompt you and help you figure out what to do next. This window also shows the name of the subsystem you are working in.

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

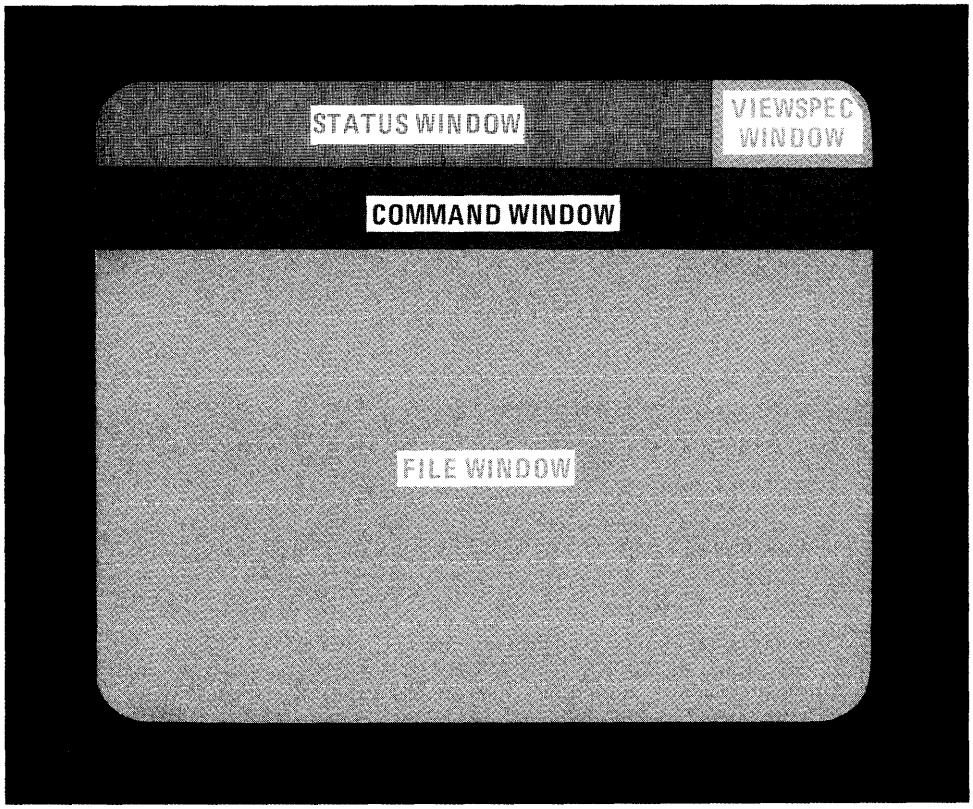


FIGURE 1: WINDOWS ON THE DISPLAY SCREEN

COMMANDS

This section gives you some important general information about AUGMENT commands. It describes features that you will try out later when you begin using the commands introduced in this lesson. While reading this section, it would be helpful for you to follow along in Figure 2, which illustrates much of what is discussed.

Simple English verbs and nouns

AUGMENT commands use simple English words and were carefully designed to make it easy for you to figure out what you can do next. They all have a similar form, like sentences telling someone to "Do this" or "Put that there". Every command you will learn in this lesson begins with a verb followed by a noun; the verb tells AUGMENT what action to take, and the noun tells it what to act on. For example, the command verb "Delete" will be followed by one of three nouns, Character, Text, or Word. Understanding the basic form of AUGMENT commands will make it much easier for you to learn new commands and new subsystems.

Command recognition

In addition to having a standard form, AUGMENT commands are entered in a standard way. To give AUGMENT a word in a command, you usually need to type only the first letter of the word; AUGMENT will recognize the word after the first letter and will display the entire word in the command window. We call a word that AUGMENT recognizes as part of a command a "command word". For example, the letter "d" stands for the command word "Delete", and when you type "d", you see "Delete" in the command window. As you learn AUGMENT, it will help you to pay careful attention to your command window, so you can be sure you are giving the right command.

Space before command words

NOTE: Where AUGMENT will recognize more than one command word beginning with the same letter, you may have to type a space and then one or more letters of the word you want to give. AUGMENT commands are designed, however, so that the most common words do not require you to type a space.

Prompts and Noise Words

Prompts

Before you give a command or after you have given part of some commands, you will see "C:" in the command window. "C:" is a "prompt". In general, a prompt is one or more uppercase letters, followed by a colon, that tell you what you can do next. "C" in a prompt stands for "command word" and means that AUGMENT is waiting for you to enter a command word. For example, after "BASE C:", you may type "d" for "Delete". AUGMENT will recognize the command word after one letter, display the entire word in the command window, and give you a new prompt, "C/OPT:", when it is ready for you to do something else.

/ in a prompt

A slash (/) between the letters in a prompt means that you have a choice. For example, the "C/OPT:" prompt you see after "Delete" means you can either enter another command word, thus responding to the "C" in the prompt, or type the special "option" character,

responding to the "OPT". (The option character is discussed in another lesson.) The meanings of the letters in AUGMENT prompts will be explained to you as you learn the commands.

Noise words

If you type "c" for "Character" in response to the "C/OPT:" prompt following "Delete", you see in your command window "Delete Character (at) M/A:". The "at" in parentheses is what we call a "noise word". Noise words provide additional information to help you understand a command. In this case, the "(at)" means you have to think about the location of the character you want to delete.

Marking with the Mouse

When to mark

To use the basic writing and editing commands in AUGMENT, you must learn how to point and "mark" with the mouse. Marking is a way of telling AUGMENT exactly what you want a command to act on. You can mark in an AUGMENT command whenever you see "M" in a prompt. For example, when you see the "M/A:" prompt after typing "dc" for "Delete Character", you can mark the exact character that you want deleted.

How to mark

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 move correspondingly on the screen. When it is appropriate to mark in a command, move the cursor under the character you want to mark and type <OK> by pressing the right mouse button. We show this process as <MARK> in examples. Remember that you must type <OK> after positioning the cursor; moving the cursor without typing <OK> has no effect.

! when you type <OK>

NOTE: Whenever you type <OK>, AUGMENT displays an exclamation point (!) in the command window to let you know it has received the <OK>.

The marked character is highlighted.

The character you mark will be highlighted. When you mark a character that is invisible, and thus cannot be highlighted, AUGMENT will instead show a special highlighted character. For example, when you mark a space, you see a highlighted tilde (∿). If you accidentally try to mark a position where there is no character, AUGMENT will mark a nearby character for you. It is a good practice to look at your screen when you mark so that you can be sure you have marked the right character.

Erasing a <MARK> with <BC>

If you mark the wrong character, you can type <BC> to "backspace a character" and erase the <MARK>, by pressing the left mouse button. You can then mark the correct character.

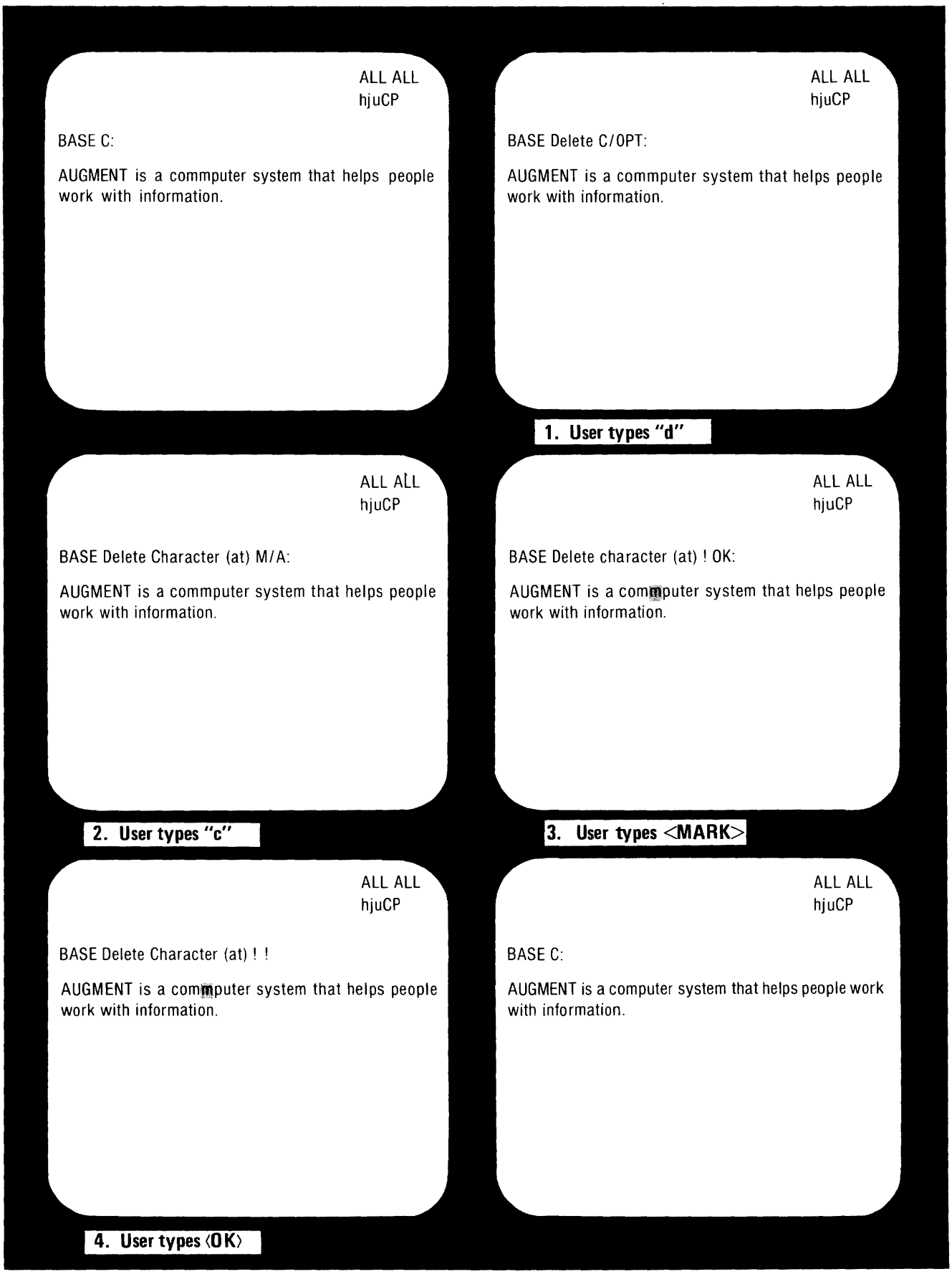


FIGURE 2: DELETING A CHARACTER

Confirming Commands with <OK>

In AUGMENT commands, you type <OK> not only to mark but also to complete other parts of a command and to indicate the end of the command. For example, when you type text in a command, you indicate that you are finished by typing <OK>. If there are no more steps in the command, this <OK> also tells AUGMENT that you are finished with the command. If you mark something as the last step, AUGMENT will prompt you to type a final <OK> with an "OK:" prompt, so you can erase the <MARK> if you wish before ending the command.

Two ways to type <OK>

You can type <OK> by pressing either the right mouse button, as recommended for marking, or the key for <OK> on the keyboard.

After the final <OK>, AUGMENT will carry out the command as entered, erase it from the command window, and prompt you with "C:" to begin another command.

Figure 2 illustrates the complete process of giving a Delete Character command, from the prompt for the first command word, through marking the character and confirming the command, to the actual deletion of the character.

Canceling a Command

<CD>

After you begin a command, you may change your mind or realize you have made a typing error and started a command you do not want. To cancel a command you have started, type the special "command delete" character, <CD>. When you type <CD>, the command you were giving will disappear and the command window will display only the subsystem name and the first prompt, "C:", telling you that AUGMENT is again ready for you to begin a command. For example, this is how you would cancel the Delete Character command:

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

Erasing the last step with <BC>

If you want to erase not the entire command but only the last step in the command, use <BC>.

MAKING A NEW WORK AREA: THE CREATE FILE COMMAND

To enter new information in AUGMENT and keep it separate from other information, you need to make a new file. The command you use to do this is the Create File command. In this command, you give the file a name that identifies the information you will store in it and that will be easy to remember when you want to look at the file in the future.

Because more than one command in the Base subsystem begins with "c", you have to type "<SP>cr" for AUGMENT to recognize "Create"; then type "f" for "File". You will see "Create File (to be named)" followed by the prompt "M/T/[A]:"; AUGMENT is waiting for the name of your new file. The "T" in this prompt means that you can give the name by typing it. Type the name and end with <OK>. AUGMENT will then carry out the command and create a file with the name you have indicated. For example, this is how you would create a file named LETTER:

You type:	Command window shows:
<SP>cr	BASE Create C:
f	BASE Create File (to be named) M/T/[A]:
letter<OK>	BASE Create File (to be named) letter!
	BASE C:

The origin statement

When you create a file, AUGMENT assumes you want to work in it and automatically displays it. At the top of the file window, you will see a file that is empty except for a heading. This heading is the origin statement of the file, and every file has one. It contains the name of the file and other information such as the identity of the person who created or last updated the file.

WRITING: THE INSERT STATEMENT COMMAND

AUGMENT statements

When you want to put information into a new AUGMENT file, you add it in the form of statements following the origin statement. The statement is the basic unit of information in AUGMENT and can be anything you feel is a logical unit. For example, if you were using AUGMENT to make a list, you might want to make each item in the list a separate statement. Or if you were writing a letter in AUGMENT, the various parts of the letter -- the salutation, each paragraph in the body, and the closing -- would all be different statements. To enter statements into a file, you can use the Insert Statement command.

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

> Dear Ms. Jones:
>
> I would like to take this opportunity to thank you for your presentation last week. The people present were impressed with your demonstration, and you certainly indeed presented us with some interesting alternatives. I am currently planning to evaluate several products. Our staff will later this week be meeting to discuss a purchase. I will be in touch with you Tuesday of next week to work out the involved details.
>
> Sincerely Yours
>
> Elinore Botoh

Entering the first statement

Each section of this letter should be entered as a separate statement. Begin the Insert Statement command by typing "is". In the command window, you will see "Insert Statement (to follow statement at) M/A:". You must then indicate which statement you want your new statement to follow; mark any character in the origin statement to show that your first new statement should follow it. Your next prompt will be "L/T/[A]:". When you see this prompt, respond to the "T" choice by typing the salutation, "Dear Ms. Jones:", and ending it with <OK>. Entering your first statement would look like this:

You type:	Command window shows:
i	BASE Insert C:
s	BASE Insert Statement (to follow statement at) M/A:
<MARK>	BASE Insert Statement (to follow statement at) ! L/T/[A]:
Dear Ms.	BASE Insert Statement (to follow statement at) ! Dear
Jones:<OK>	Ms. Jones:!
	BASE C:

When you type the <OK> at the end of the command, the statement you have been typing in the command window will be inserted in place in the file. After you enter the first statement, you will see it in your file window following the origin statement.

You don't need carriage returns.

For the second statement, type "is" for "Insert Statement" and mark one of the characters in the statement you just added, so the body of the letter will follow the salutation. When typing this long paragraph, you do not have to type a return to indicate the end of a line. Simply type the words, spacing, and punctuation that you want. As you type, AUGMENT will automatically adjust the text, starting a new line when it cannot fit any more text on the current line (because it has reached the right edge of the command window). When breaking the text into lines, AUGMENT breaks between words. The text is always adjusted in this way, both when you are typing it in the command window and later when it is shown in the file window.

The command window expands.

When you enter a lot of text, it may seem that what you are typing is writing over the top lines of your file. What is actually happening is that, in order to display the text you are typing, the command window is expanding into the file window. When you end the command and the text is inserted, the command window will return to its former size and your file will reappear.

Correcting with <BC> and <BW> while you type

Try to type the paragraph exactly as it is shown; later in this lesson you will learn how to correct it with editing commands. If you make mistakes while typing, you can use <BC> to erase the last character you typed and <BW>, the "backspace word" character, to erase the last word plus any spaces or punctuation that immediately follow it. 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 type:	Command window shows:
i	BASE Insert C:
s	BASE Insert Statement (to follow statement at) M/A:
<MARK>	BASE Insert Statement (to follow statement at) ! L/T/[A]:
I would...	BASE Insert Statement (to follow statement at) ! I would
details.<OK>	like to take this opportunity to thank you for your presentation last week. The people present were impressed with your demonstration, and you certainly indeed presented us with some interesting alternatives. I am currently planning to evaluate several products. Our staff will later this week be meeting to discuss a purchase. I will be in touch with you Tuesday of next week to work out the involved details.!
	BASE C:

Complete the letter by adding the last two statements (the closing and the name). For each one, use the Insert Statement command and mark 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 WITH DELETE, MOVE, REPLACE, AND INSERT

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

Delete

"Delete" enables you to remove information from a file. For example, if you deleted the second "r" in "perrfect", the word would be "perfect".

Move

"Move" allows you to reorder information in a file by taking it from one place and putting it in another place. For example, if you moved the word "that" in "a phrase makes that sense" to follow the word "phrase", you would have "a phrase that makes sense".

Replace

"Replace" lets you remove information and put other information in its place; that is, it combines Delete and Insert into one verb. The new information may be shorter or longer than what it replaces. For example, if you replaced the word "good" with the word "terrific" in the phrase "a good sentence", you would have "a terrific sentence".

After you edit with any of these verbs, AUGMENT adjusts the text to reflect the change. For example, when you delete information in a statement showing on your screen, AUGMENT shows the statement as if the information had never been there, possibly breaking the text into lines differently than before.

Once you know the command verbs, you need to know the nouns that go with them, to tell AUGMENT what you want to delete, move, replace, or insert. Three of the most important nouns are Character, Text, and Word.

Character

A "Character" is a single letter, number, punctuation mark, space, return character, or special control character. Characters you can see on your screen are called "visible" characters; those you cannot see are "invisible". Invisible characters can be marked and edited just like any other characters in AUGMENT statements. A space, for example, is an actual character that separates one word from another and can be deleted, moved, replaced, or inserted; it is not emptiness.

Text

"Text" is any series of characters within a statement. It may begin or end within a word and may include punctuation, spaces, and any other visible or invisible characters. Text can be only one character or it can be all the characters in a statement. You point to text by marking the beginning character and the ending character.

Word

A "Word" is a 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!) AUGMENT does not consider the surrounding characters as part of the word. You point to a word by marking any character within it. Whenever you delete, move, replace, or insert a word, you do not need to worry about the spacing around the word; AUGMENT knows words commonly have spaces around them and will provide and remove them as necessary.

Figure 3 illustrates Character, Text, and Word. To help you practice using Delete, Move, Replace, and Insert with these nouns, we have provided a list of corrections to be made to the LETTER file and have suggested a way of making each correction. For the first five corrections, we show the details of what you type and what you see on your screen. After making the first five corrections, you should be familiar enough with the form of the commands to finish the rest of the corrections without seeing the details.

Delete Character

1. Remove the comma following "your demonstration" by using the Delete Character command and marking the comma. AUGMENT will leave the space between "demonstration" and "and".

You type:	Command window shows:
d	BASE Delete C/OPT:
c	BASE Delete Character (at) M/A:
<MARK>	BASE Delete Character (at) ! OK:
<OK>	BASE Delete Character (at) ! !
	BASE C:

Delete Word

2. Remove the word "indeed" between "certainly" and "presented". Use the Delete Word command and mark any character in "indeed". Notice how the text is adjusted when the word is deleted.

You type:	Command window shows:
d	BASE Delete C/OPT:
w	BASE Delete Word (at) M/A:
<MARK>	BASE Delete Word (at) ! OK:
<OK>	BASE Delete Word (at) ! !
	BASE C:

Delete Text

3. Change the word "presented" to "present" by deleting the "ed". Use the Delete Text command. Mark the "e" as the first character of the text and the "d" as the last character.

You type:	Command window shows:
d	BASE Delete C/OPT:
t	BASE Delete Text (at) M/A:
<MARK>	BASE Delete Text (at) ! (through) M/A:
<MARK>	BASE Delete Text (at) ! (through) ! OK:
<OK>	BASE Delete Text (at) ! (through) ! !
	BASE C:

Move Character

4. Change "Botoh" to "Booth" by moving the "t" to follow the second "o". Use the Move Character command. Mark the "t" as the character to be moved and mark the "o" as the character it should follow.

You type:	Command window shows:
m	BASE Move C/OPT:
c	BASE Move Character (from) M/A/[T]:
<MARK>	BASE Move Character (from) ! (to follow character at) M/A:
<MARK>	BASE Move Character (from) ! (to follow character at) ! OK:
<OK>	BASE Move Character (from) ! (to follow character at) ! !
	BASE C:

Move Text

5. Change the phrase "Our staff will later this week be meeting" to "Our staff will be meeting later this week". Use the Move Text command. For the text that you want to move, mark the space before "later" as the first character of the text and the "k" in "week" as the last character of the text. For the character you want the text you are moving to follow, mark the "g" in "meeting".

CHARACTER

The Office Automation Division of Tymshare provides general-purpose, computer-based services for controlling, reshaping, and disseminating information. These services fill a variety of organizational needs, such as office automation, document production, and software engineering.

WORD

The Office Automation Division of Tymshare provides general-purpose, computer-based services for controlling, reshaping, and disseminating information. These services fill a variety of organizational needs, such as office automation, document production, and software engineering.

TEXT

The Office Automation Division of Tymshare provides general-purpose, computer-based services for controlling, reshaping, and disseminating information. These services fill a variety of organizational needs, such as office automation, document production, and software engineering.

STATEMENT

FIGURE 3: CHARACTER, WORD, TEXT, AND STATEMENT

You type:	Command window shows:
m	BASE Move C/OPT:
t	BASE Move Text (from) M/A/[T]:
<MARK>	BASE Move Text (from) ! (through) M/A:
<MARK>	BASE Move Text (from) ! (through) ! (to follow character at) M/A:
<MARK>	BASE Move Text (from) ! (through) ! (to follow character at) ! OK:
<OK>	BASE Move Text (from) ! (through) ! (to follow character at) ! !
	BASE C:

Move Word

6. Change the phrase "the involved details" to "the details involved". Use the Move Word command to move "involved" to follow "details" by marking any character in "involved" and then any character in "details".

Replace Character

7. Use the Replace Character command to replace the "Y" in "Yours" with "y" by marking the "Y" and then typing "y" followed by <OK> or marking any "y". Notice that if you mark a "y" to indicate what the character you want inserted, the "y" you mark will not be changed.

Replace Text

8. Change the phrase "planning to evaluate" to "evaluating". Use the Replace Text command. Mark the "p" in "planning" as the first character of the text you want to replace and then the "e" in "evaluate" as the last character of the text. Type the new text "evaluating" and then <OK>.

Replace Word

9. Use the Replace Word command to replace "present" with "attending" by marking any character in "present" and typing "attending" followed by <OK>.

Insert Character

10. Add a comma after "Sincerely yours" to make "Sincerely yours,". Use the Insert Character command and mark the "s" as the character the new character should follow; then type a comma followed by <OK> or mark a comma.

Insert Text

11. Add a new sentence "I hope this will be convenient." to follow the last sentence in the body of the letter. Use the Insert Text command and mark the period at the end of the last sentence. Then type "<SP> <SP>I hope this will be convenient." and end with <OK>. You need to type the two spaces so there will be two spaces between the old sentence and the new one.

Insert Word

12. Change the phrase "a purchase" to "a possible purchase". Use the Insert Word command and mark the "a" as the word you want the new word to follow; then type the new word "possible" and end with <OK>.

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

Remember that "Text" can be a single character. To replace one character with text showing on your screen, you can use the Replace Text command, mark the character as both the beginning and ending character of the text to be replaced, and then mark the text you want to replace it with.

Marking between words

Marking words in AUGMENT has an additional feature. When you work with words, you may also mark the space between two words; AUGMENT will consider the two words as one. Or you can mark a hyphenated word by marking the hyphen between the words comprising it. Marking between words is often very useful in editing, and can significantly reduce the number of commands you have to give. For example, you could revise the phrase "I will be in touch with you Tuesday of next week" to read "I will be in touch with you next week", by giving the Delete Word command and marking the space between "Tuesday" and "of".

Why does this work? As an editing convenience, AUGMENT has been designed so that whenever you tell it you want to work with a word, it assumes that what you mark is a character in a word, without actually examining the character. This means that when you mark a space or hyphen between two words, AUGMENT assumes it is a character in the middle of a word and treats the two words as one. Similarly, you can move or insert a word to follow a punctuation mark immediately following a word, such as a comma, simply by marking the punctuation; AUGMENT assumes the punctuation is the last character in a word, places your new word after it, and inserts a space, as usual, before your new word.

There may be times when you want to move or insert a word to follow a punctuation mark that immediately precedes a word, such as a left parenthesis. When you mark the parenthesis, AUGMENT again thinks you have marked a character in a word (in this case, the first character) and inserts your new word after what it believes is the whole word. Thus your new word comes not only after the parenthesis but also after the word immediately following it.

Spaces at line breaks

NOTE: Line breaks do not affect what you can mark; spaces that fall at the ends of lines can be marked just as when they are within lines.

RETURN CHARACTERS IN STATEMENTS

Although you do not need return characters to end lines when you type text in AUGMENT, there may be places where you will want them. One example of such a place is in an address; you might want it all to be one statement, but you would not want to let AUGMENT decide where to break it into lines. Instead you might want a name on the first line, the street address on the second, and the city, state, and zip code on the last. You would achieve this, just as you would on a typewriter, by

typing a return at the end of each of the first two lines. Return characters, which we indicate as <RET>, force AUGMENT to begin a new line.

Thus, if you wanted to add an address at the top of the letter written as described in this lesson, you could use the Insert Statement command, mark the origin statement to indicate that the address should follow it, and then enter the address, typing <RET> each time you wanted a new line and ending the entire address with <OK>. For example:

You type:	Command window shows:
i	BASE Insert C:
s	BASE Insert Statement (to follow statement at) M/A:
<MARK>	BASE Insert Statement (to follow statement at) ! L/T/[A]:
Growers Inc.<RET>	BASE Insert Statement (to follow statement at) !
45 B Street<RET>	Growers Inc.
Rome NY 14703<OK>	45 B Street
	Rome NY 14703!
	BASE C:

When you are entering text and you type <RET>, the information showing in the top line of your file window will disappear; AUGMENT knows you want a new line and has simply expanded the command window into the file window in order to accommodate it.

Editing return characters

It is important to understand that return characters are like spaces in that although they are invisible they can be marked and edited just like any other characters. You can insert a return character with the Insert Character command, or delete a series of characters that includes a return character with the Delete Text command, and so on. If, in the address entered in the above example, you replaced the return character after the street address with a space, the last two lines of the address would become one line (with "Street" and "Rome" separated by a space). When you mark a return character, AUGMENT will show a highlighted left angle bracket (<).

UPDATING A FILE: THE UPDATE COMMAND

Consolidating changes

The changes you make to a file are automatically saved by AUGMENT; however, although you see the changes along with the file, they are actually saved separately from it. Only when you "update" your file are the changes completely integrated into it. Any time you have significantly changed a file or do not plan to work on it in the near future, you should update the file as follows:

You type:	Command window shows:
u	BASE Update (file) OK/C:
<OK>	BASE Update (file) ! OK:
<OK>	BASE Update (file) ! !
	BASE C:

This command makes a new version of the file that incorporates all the changes you have made since you created or last updated it. The information in the origin statement of the file changes to show that you are working with a new version. You could then do more editing, making a new set of changes, and could go back to the old version if you wished.

GETTING TO A FILE: THE JUMP (TO) LINK COMMAND

Suppose you leave AUGMENT after writing and editing a file as described in this lesson, and you later want to see the file again. Whenever you enter AUGMENT, your initial file is displayed; if you want to see a different file, you need to use a command to get to it. To reach any one of your files, use the Jump (to) Link command. Simply type "jl" and then type the name of the file, a comma, and <OK>. For example, this is how you could get to the file named LETTER after logging in and entering AUGMENT:

You type:	Command window shows:
j	BASE Jump (to) M/C:
l	BASE Jump (to) Link M/T/[A]:
letter,<OK>	BASE Jump (to) Link letter,!
	BASE C:

EXERCISES

1. Write the following answer to Ms. Booth's letter and keep it separate from the information in your other files:

> Monday I will be giving a demonstration to your marketing division. So we'll see at last what they think!
>
> <your name>

Now add the statement "Dear Ms. Booth:" before the first statement you added above.

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

(a) Change "Monday" to "Next Monday".

(b) Change "So we'll see at last what they think" to "So at last we'll see what they think", 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.

(d) After your name, add the name of your organization on a separate line without adding another statement.

SUMMARY

Several basic AUGMENT concepts were introduced in this lesson: using the verb–noun pattern to combine various command words; typing only enough characters in a command word for AUGMENT to recognize the word; understanding and responding to prompts; marking with the mouse; and ending commands or parts of commands with <OK>. You have also learned how to cancel all or part of a command with <CD> or <BC>. These features are consistent throughout all of AUGMENT. Learning these elements will make it easy for you to learn the rest of AUGMENT.

This lesson has also taught you some basic commands for writing and editing in AUGMENT. You should now know how to create a file and add statements to it, how to use <BC> and <BW> to make corrections while you are typing text, and how to edit a statement in a file such as by removing or adding characters, words, or text. You have also learned how to update a file you have worked on and how to get to it again the next time you enter AUGMENT. We recommend that you now read the lesson "Writing and Reading an Organized File".

LIST OF COMMANDS

Create File (to be named) TYPEIN

Insert Statement (to follow statement at) LOCATION TYPEIN

Insert Character (to follow character at) LOCATION CONTENT <OK>
Insert Text (to follow character at) LOCATION CONTENT <OK>
Insert Word (to follow word at) LOCATION CONTENT <OK>

Delete Character (at) LOCATION <OK>
Delete Text (at) LOCATION (through) LOCATION <OK>
Delete Word (at) LOCATION <OK>

Move Character (from) LOCATION (to follow character at) LOCATION <OK>
Move Text (from) LOCATION (through) LOCATION (to follow character at) LOCATION <OK>
Move Word (from) LOCATION (to follow word at) LOCATION <OK>

Replace Character (at) LOCATION (by) CONTENT <OK>
Replace Text (at) LOCATION (through) LOCATION (by) CONTENT <OK>
Replace Word (at) LOCATION (by) CONTENT <OK>

Update (file) <OK> <OK>

Jump (to) Link TYPEIN

Definitions:

LOCATION	Prompted by "M/A:" For M you may <MARK>.
CONTENT	Prompted by "M/T/[A]:" For M you may <MARK>. For T you may type a series of characters, ending with <OK> (if another <OK> follows, a second one is not needed).
TYPEIN	Type a series of characters, ending with <OK>.

VOCABULARY

The page numbers indicate where the vocabulary item is discussed in this lesson.

Base subsystem: A basic set of AUGMENT commands for reading, writing, editing, printing, and filing information. Page 1

<BC>: "BC" stands for "backspace character". Typing <BC> erases the last character you typed. You can also use <BC> to erase the last step in a command. Pages 4, 5, 7

<BW>: "BW" stands for "backspace word". Typing <BW> deletes the last word you typed (plus any spaces, punctuation, or other characters following the word). Page 7

<CD>: "CD" stands for "command delete". Typing <CD> cancels any command you have not finished (that is, before you have given the final <OK>). You may then begin a new command. Page 5

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. Page 9

command: An instruction you give to the computer to perform an action. When you give AUGMENT a command, AUGMENT will perform the action after you complete the command with the final <OK>. Page 3

command window: As you give AUGMENT a command, it is displayed in this window along with prompts and noise words. You also see the name of the subsystem you are working in. Page 2

command word: A word that AUGMENT knows is part of a command, for example, the verb and noun that usually begin a command. Page 3

Create File command: A Base subsystem command that makes a new file, giving it the name you specify. Page 5

Delete command: A Base subsystem command you can use to remove information, such as a character, a word, or some text. Page 8

exclamation point: Each time you type <OK> when giving an AUGMENT command, you will see an exclamation point (!) in the command window. The exclamation point simply lets you know that the <OK> was received by AUGMENT. Page 4

file: A work space on the computer, like a file folder in a filing cabinet, that you can fill with information. Page 1

file window: This window displays files or parts of files. Page 2

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. Pages 6, 8

invisible character: A character you cannot see on your screen, such as a space or a return character. Page 9

Jump (to) Link command: An AUGMENT command you can use to move from one file to another. Page 14

mark: To mark means to indicate a character on the screen by using the mouse to point to it and then typing <OK>. Page 4

<MARK>: This notation means that you are to mark a character on the screen. Page 4

Move command: A Base subsystem command to reorder information in a file; for example, you can move one character to follow another. Page 8

noise words: When you give a command word, AUGMENT may respond with a word or phrase in parentheses. These are called "noise words"; they help you understand the purpose of the command or what you need to do to complete it. Page 4

<OK>: Typing <OK> tells AUGMENT that you have finished giving a command or part of a command. When you type <OK>, you see an exclamation point (!). Pages 4, 5

origin statement: The first statement in every file. It contains information such as the name of the file and the identity of the person who created or last updated the file. Page 6

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 (:). Page 3

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. Page 8

<RET>: This represents a return character, the character that forces AUGMENT to begin a new line. Return characters can be marked and edited just like any other characters. Page 12

<SP>: This represents a space, that is, what you type with the space bar on the keyboard. In AUGMENT, a space is an actual character that separates one word from another and can be deleted, moved, replaced, or inserted; it is not emptiness. Page 9

statement: The basic unit of information in an AUGMENT file. A statement may be a single character, a word, a title, a heading, an address, a paragraph, or any logical unit. Every character in an AUGMENT file is in a statement. Page 6

status window: This window displays messages from AUGMENT or the Executive. Page 2

subsystem: AUGMENT is divided into subsystems, which are sets of commands related to particular activities. Page 1

text: Any series of characters within a statement. It may begin or end within a word and may include punctuation, spaces, and any other visible or invisible characters. Page 9

Update command: A Base subsystem command to consolidate recent changes into your file. You can make a new version incorporating the changes, do more editing, and return to the old version if you wish. Page 13

viewspec window: This small window displays characters that tell you what kind of view you have of the file being displayed. Page 2

visible character: A character you can see on your screen, such as a letter, number, or punctuation mark. Page 9

windows: When you use AUGMENT, the display screen is divided into four areas, called "windows". Page 2

word: A series of letters and/or numbers surrounded by spaces, punctuation marks, or any other characters that are not letters or numbers. Page 9

SOLUTIONS TO EXERCISES

1. Use the Create File command and give the file a name different from the names of your other files. Using the Insert Statement command and marking the origin statement of the new file, type in the statement that begins with "Monday", ending with <OK>. Then use the Insert Statement command again, marking the statement you just added and typing in the statement consisting of your name. To add a statement before the first statement, do exactly what you did to enter the first statement: Give the Insert Statement command, mark the origin statement, and type "Dear Ms. Booth:" followed by <OK>.

2. (a) You can do this with the Replace Character command, replacing the "M" with "Next M", or with the Replace Word command, replacing "Monday" with "Next Monday".

(b) Use the Move Word command. Type "mw", mark the space between "at" and "last" to indicate the word to be moved, and mark any character in "So" to indicate the word it should follow. To change it back again, type "mw", mark the space between "at" and "last" and then mark any character in "see". Note that if you mark the space between "we'll" and "see", you will be referring to "ll see", because the word ends at the punctuation mark.

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

(d) Give the Insert Text command and mark the last character of your name as the character the text should follow; then type a return character and the name of your organization followed by <OK>.

Price $3.00

STARTING AND ENDING AN AUGMENT WORK SESSION: FOR USERS OF ARPANET

Copyright Tymshare Inc., May 1980
20705 Valley Green Drive
Cupertino, California 95014

This document was prepared and composed for printing with AUGMENT by the Office Automation Division of Tymshare, Inc.

AUGMENT Journal Number 48744
Tymshare Document Number 1839

This lesson teaches you how to start a work session by identifying yourself to the computer (log in), how to enter AUGMENT, and how to end your work session (log out). It is written for users who reach the AUGMENT computer through ARPANET and who use an AUGMENT 1200 display terminal. It also introduces you to the equipment you will be using.

Illustrations of the normal, simple procedures for logging in, entering AUGMENT, and logging out appear on the inside covers. This lesson gives you helpful background information, elaborates on the procedures, and teaches you how to deal with unusual situations such as interruptions in service.

You should learn from your AUGMENT architect the telephone number you must dial or other procedure you must follow to make the initial connection to the computer, as well as the specific identifying numbers, names, or passwords you will have to provide.

CONTENTS

INTRODUCTION

To work in AUGMENT, you first establish a connection to the computer and "log in" by identifying yourself. You then enter AUGMENT and do your work. When you want to stop working, you "log out" and break your connection to the computer. With practice, the procedures of logging in and logging out will become almost automatic.

Using a host computer through a network

You will be working at a terminal that may be thousands of miles from the computer that supports AUGMENT. The distant computer is called a "host computer". A collection of small computers, called a "network", connects your terminal to the AUGMENT host computer. The computers in the network are dispersed geographically; users can connect to the nearest one to communicate with a host computer, wherever its location.

When you press a key on a manual typewriter, you activate a series of levers and springs that ends up with the key striking the ribbon and printing a character; when you press a key on an electric typewriter, an arrangement of electric wires, motors, and magnets performs the same function. When you press a key on your AUGMENT terminal, an arrangement of digital electronics in the terminal and the network similarly transmits a character and, in addition, brings you the response of the host computer.

Computer networks are designed to be as invisible and reliable as possible. Normally as you work you will be no more aware of them than you are of the levers or wires in a typewriter. When you begin your work, however, it is necessary to identify to the network the host computer you want. The network also affects other procedures you may occasionally need to use. For these reasons, we believe you will have more confidence in your work if you have some background information on the network.

What the network is

The network consists of a number of small computers, each of which is connected to several others. These small computers are called "nodes". When you type at your terminal, signals from your keystroke go to the node you are connected to, and from that node to another node, and so on through the network to the host computer of your choice. The host computer carries out work for you and sends back characters to your display by the same method. Along the way, the various nodes store your characters for safekeeping until the host computer has responded, check for transmission errors, and perform other communications services. Figure 1 shows the path of communication through the network between your terminal and the host computer.

ARPANET

This lesson is written for users who reach AUGMENT through ARPANET, a United States Department of Defense network. You need to learn from within your own organization which network serves you.

Overview of the steps to reach AUGMENT

Figure 2 briefly indicates the steps most users take to reach AUGMENT through ARPANET. After preparing your equipment, you can connect the terminal to the network through a telephone line. You then need to connect to the host computer by instructing the network to open the connection. Your last step before entering AUGMENT is to log into the host computer.

Operating system and the Executive

The first system you communicate with on the host computer is its "operating system", a set of programs that supervises the operation of other programs and performs general support functions. The operating system on the host computer you will be using is TENEX. The instructions that you give to the computer are called "commands"; you interact with the operating system through a set of commands that is collectively called "the Executive".

The Executive includes commands for logging into the host computer, entering AUGMENT, and logging out of the host computer. Along with these commands, you will learn two AUGMENT commands: one for logging out, and one for returning to the Executive in case you want to log out from there. This lesson will also teach you how to attach back to the host computer if your connection breaks due to network or terminal malfunction.

TIP and ANTS

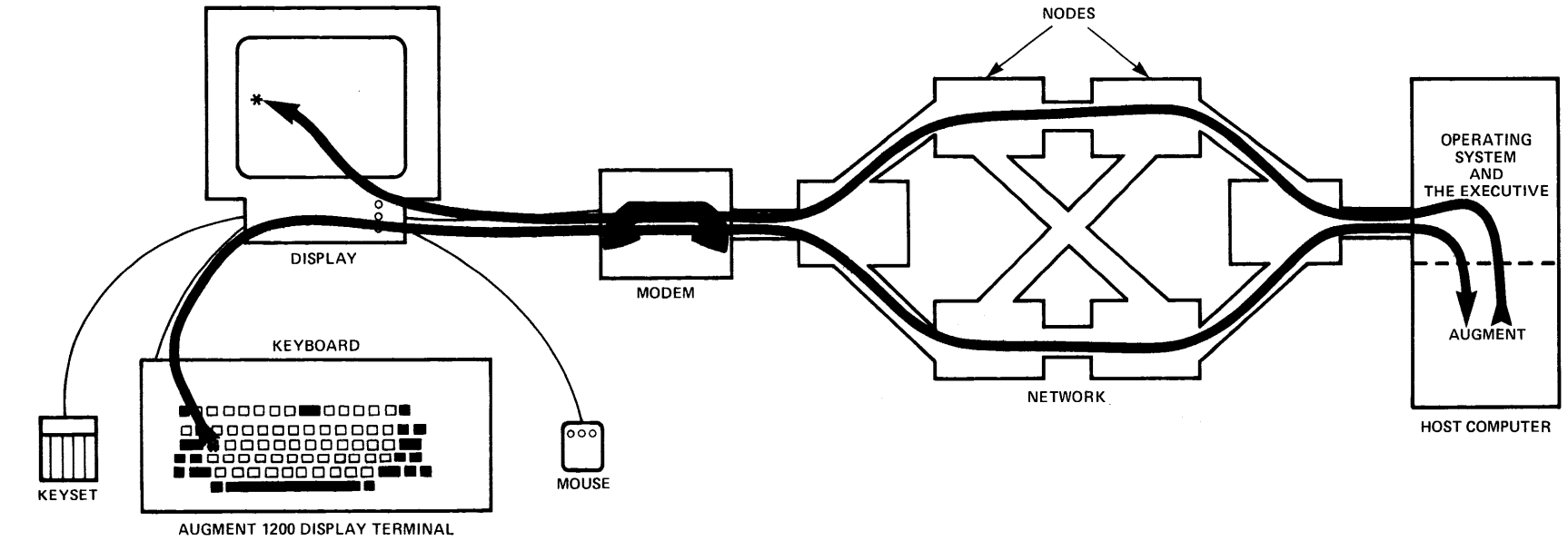
Most ARPANET users communicate with the network through a node called a "TIP", which stands for "Terminal Interface Processor". Some users, however, reach ARPANET through an "ANTS"; if you are one of these users, ask your AUGMENT architect for special instructions regarding connecting to and disconnecting from the network.

Multiplexers

At some sites, devices called "multiplexers" are employed to communicate between your terminal and the network. Multiplexers are important to logging in because they may make it easier or more difficult, depending on the equipment involved. Some multiplexers make logging in easier by automatically logging in the user; others require the user to log into the multiplexer as a separate, first step. Because of the differences, this lesson cannot instruct you in how to deal with multiplexers. Please ask your AUGMENT architect for the additional, special instructions you need.

Format of examples

This lesson includes examples of doing work outside of AUGMENT, that is, of interacting with the operating system. The format of these examples differs somewhat from that of AUGMENT examples (explained in the "Introduction to the AUGMENT Textbook Series"), because of differences in the way information is displayed and in the type of interaction that occurs between the user and the computer. The format is similar enough, however, that you should have no problem following the non-AUGMENT examples. They are shown in two columns. As in all examples, the first column indicates what you type and uses indenting to show continuation; you type a space only where one appears between words or is specially indicated by the notation "<SP>". The second column simply shows everything that appears on the display screen.



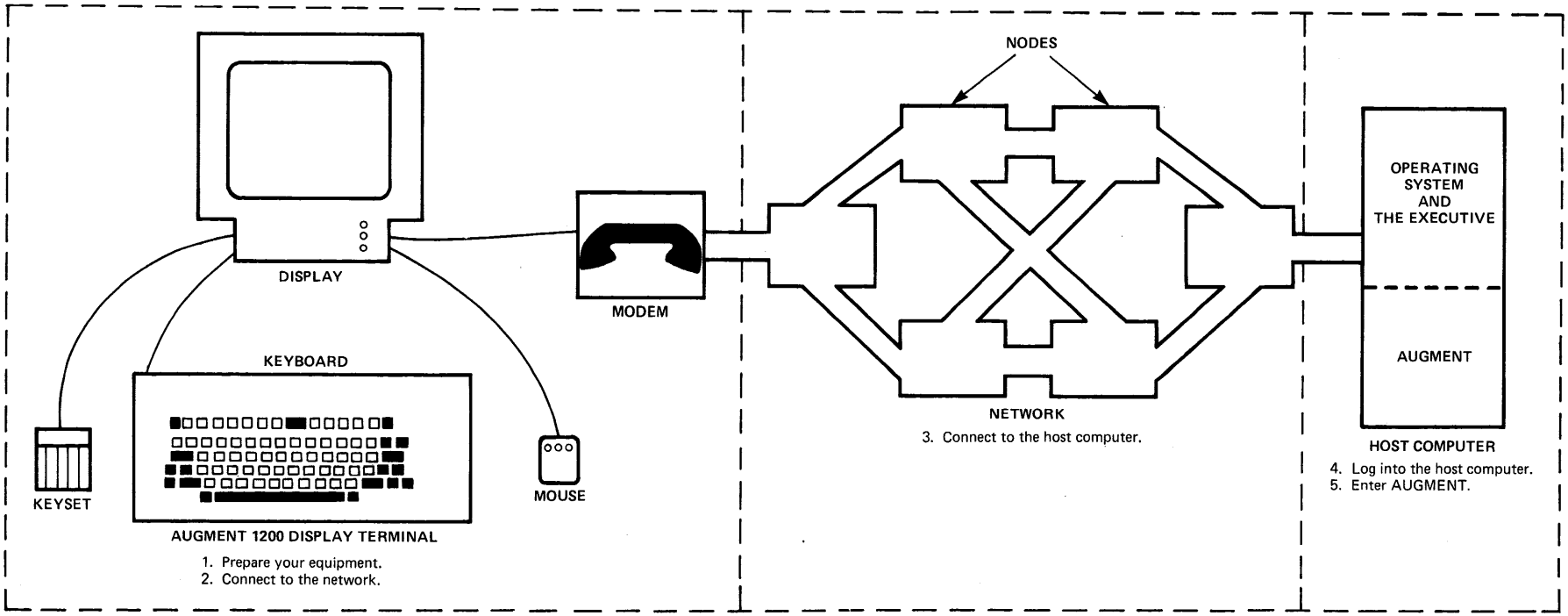


FIGURE 1(Upper): THE PATH OF COMMUNICATION THROUGH THE NETWORK
FIGURE 2(Lower): THE STEPS TO REACH AUGMENT

NOTE: Except where indicated otherwise, the numbers, names, and passwords shown in examples are hypothetical. The actual information that you will supply and receive depends on your situation.

PREPARING YOUR EQUIPMENT

When you are ready to use your AUGMENT work station for the first time, you should start by identifying the various pieces of equipment. Your AUGMENT 1200 display terminal consists of the following:

display: A television-like device that shows information stored in the computer and your interaction with it.

keyboard: An arrangement of keys as on a typewriter, plus keys that have special uses. You can type text and enter commands with the keyboard.

mouse: A hand-sized device, with three buttons on the top, that rolls freely over a flat surface and correspondingly moves the cursor (traveling mark) on the display screen.

keyset: A device with five piano-like keys for entering characters into AUGMENT at a display terminal; an alternative to the keyboard.

With the keyset and the buttons on the mouse, you can enter commands and characters quickly while moving the cursor with the mouse. The keyset is a convenience, to be learned as time permits.

Modem

Unless your connection uses a multiplexer instead of a telephone line or your terminal is wired directly to a TIP, your work station will also include a "modem", which is a device that translates information between computer devices and telephone lines.

This equipment should be set on a table large enough to hold your papers and at an appropriate height to permit comfortable use of the keyboard. Your work station may also include a printing device.

Switches on the display

When you are ready to begin working, look on the left side of your display for the power ON/OFF switch. After turning it on, you will hear the sound of a fan. If you have a modem, turn it on as well. On the lower right front of the display you will see several push-switches. Make sure that the switch marked "EIA" is pushed in. Generally, the other switches should remain out. An explanation of these switches and other information about your terminal appears in the "AUGMENT 1200 Display Terminal Users' Guide", available through the Office Automation Division at Tymshare.

Adjusting the screen

Look on the right side of your display for two knobs, one marked "B" (for "brightness") and the other marked "C" (for "contrast"). As with a television set, if the brightness or contrast is too low, you will see nothing. Adjust the screen so that you can clearly and comfortably see the cursor on it. Later, when information appears on the display, you may want to make further adjustments.

Checking the mouse

The next step is to check your mouse by moving it across the table to see that the cursor moves across the screen correspondingly.

Once you have completed these preparatory steps, you are ready to connect your terminal to the network.

CONNECTING TO THE NETWORK

To connect to the network, most users dial a special telephone number (which answers with a high-pitched sound) and then follow the instructions for the modem being used. Some ARPANET users have terminals that are wired directly to a TIP and therefore do not have to use the telephone. If you do not know the procedure for connecting your terminal to the network, ask your AUGMENT architect for assistance.

Upon connecting to the network, ARPANET users normally see some coded information about the connection, such as:

```
AMES TIP 112 #: 66
```

If you do not see something like this, you may have to type "e" or "E" to get the attention of the TIP.

You type:	Display shows:
e	AMES TIP 112 #: 66

CONNECTING TO THE HOST COMPUTER

Your next step is to connect to the host computer. You do this by giving a command to open the connection and specifying the number of the host computer. For example, to open the ARPANET connection to host computer number 43, you would do this:

You type:	Display shows:
@i<SP>25<RET>	@i 25
<CTRL-Y>o<SP>	o 43
43<RET>	Trying...
	Open
	Welcome to AUGMENT Service from Office-1
	@

If you do not know the number of the host computer you want to use, ask your AUGMENT architect. Note that "@i<SP>25<RET>" is a special instruction that you must give to be able to use AUGMENT in display mode.

Errors in opening the connection

If you see the word "BAD" when you are trying to open the ARPANET connection, it means that you made a typing error and should start over again from the "@" as shown above. If you see the word "CAN'T", or "Open" followed by a message from a host computer other than the one you wanted, you should first close the connection (as described later in this lesson) and then start over again from the "@".

Host not responding

Occasionally you may not succeed in connecting to the host computer because it is "down", that is, not operating. In this case, you will see a message such as "Host not responding" and should turn off your equipment and start over again. Try starting over again in two minutes; if the computer is still down, try again in about twenty minutes. If you still cannot connect to the host computer and would like more information, call the Office Automation Division at Tymshare.

NOTE: To prevent unauthorized access, ARPANET will break your connection to the network if you do not connect to the host computer within a certain amount of time. The exact amount of time depends on the network and the connection.

Operating system message

When you connect to the host computer, you see a message from its operating system. The information given in the message depends on the operating system; normally it includes the name of the host computer.

@ from the Executive

Following the message from the operating system, you see an at sign ("@"). This character indicates that you can give an Executive command. The host computer is waiting for you to identify yourself so that you can work in AUGMENT.

LOGGING INTO THE HOST COMPUTER

Your last step before entering AUGMENT is to log into the host computer. You must identify yourself to the computer by supplying a name called a "directory name" (sometimes called a "user name") and a password. Some users may also have to specify an account; in most cases, however, the computer will automatically charge the appropriate account according to the directory name. If you do not know the specific identifying information you must supply, ask your AUGMENT architect for assistance.

You log into the AUGMENT host computer by giving an Executive command that begins with "login" (or the abbreviation "log"). The following example shows how you could log in if your directory name were TWAIN and your password were "clemens".

You type:	Display shows:
log twain<RET>	@log twain
clemens<RET>	(password)
<RET>	(Account)
	Job 23 on TTY64 8-Feb-79 13:22
	Previous login: 8-Feb-79 08:10
	@

For security reasons, the password does not appear on the display screen when you type it.

Account

Notice that after entering the password, you see "(Account)". If, like most users, you do not specify the account, your reply is simply a return character; otherwise, type the account designation followed by a return character.

You can also use the following quicker method of logging into the host computer:

You type:	Display shows:
log twain<SP>	@log twain
clemens<SP>	Job 23 on TTY64 8-Feb-79 13:22
<RET>	Previous login: 8-Feb-79 08:10
	@

Users who specify the account may do so after the space following the password.

? after trying to log in

If you see only a question mark (?) and another "@" after trying to log into the host computer, you have not succeeded in logging in. You can then simply repeat the command.

Some host computers restrict the number of people who may log in. If you see a message telling you that you may not log in for this reason, contact your AUGMENT architect.

Job number

Upon logging into the host computer, you see an operating system message that tells you the job number and terminal number for this work session, the current date and time, and possibly other information. For example, the message in the example above tells you the job number is 23, the terminal number is 64, the date and time are 8-Feb-79 and 1:22 p.m., and you last logged in at 8:10 that morning. The exact content of the message depends on the operating system. Note that all messages reflect the time zone of the host computer.

Other messages may follow this operating system message. Their meanings will become clear as you become more familiar with the system. You may, for example, see the message "[You have new mail]"; another lesson will describe how to read the mail this message refers to.

Finally, after all messages, you again see the "@" indicating that you can give an Executive command.

ENTERING AUGMENT

After logging into the AUGMENT host computer, you are ready to enter AUGMENT.

You type:	Display shows:
augment<RET>	@augment

Ident and sharing directories

Sometimes several people use the same directory name when they log in. In this case, each individual user must give further identification upon entering AUGMENT, by supplying his or her "ident". An "ident" is a short series of characters that identifies an individual to AUGMENT. After the step shown above, the screen will go blank and then display a request for the ident. For example, if you were sharing a directory and your ident were MTN, the second step of entering AUGMENT would be as follows:

You type:	Display shows:
mtn<RET>	Ident = MTN

Command window

When you enter AUGMENT, the screen will go blank briefly and then show you information in a special way. In AUGMENT, the screen is divided into areas called "windows"; the information displayed in each window has a separate function and can change independently. When you give a command in AUGMENT, one of these windows, the "command window", will show the command you are giving and the feedback you get from AUGMENT, such as prompts telling you what you can do next. Most users see "BASE C:" in the command window upon entering AUGMENT.

Initial file

In another window, just below the command window, you will see the beginning of your "initial file". A "file" is a work space on the computer, like a folder in a filing cabinet, that you can fill with information. Your initial file is a special file, with the same name as your ident, that automatically appears when you enter AUGMENT. As you learn more about AUGMENT, you will learn more about the windows on your screen and about working with files.

ENDING YOUR WORK SESSION: LOGGING OUT

Logout Job command

When you want to end your AUGMENT work session, you can give the Logout Job command in AUGMENT.

You type:	Command window shows:
	BASE C:
l	BASE Logout C:
j	BASE Logout Job OK/T/[M]:
<OK>	BASE Logout Job !

Your screen will then go blank except for the word "Bye".

Quit command

You may instead want to return to the Executive before logging out (for example, if you are familiar with Executive commands and want to do some work there). To return to the Executive, give the Quit command in AUGMENT, as follows:

You type:	Command window shows:
	BASE C:
q	BASE Quit OK:
<OK>	BASE Quit !

Executive Logout command

You will then see the Executive "@" again, possibly preceded by a message from the operating system, and can give any Executive command. When you want to log out, you can give a Logout command, by typing "logout" (or the abbreviation "logo") followed by a return character.

You type:	Display shows:
logout<RET>	@logout Logout Job 23, User TWAIN, Acct ARCOVH, TTY 64, at 2/08/79 1430 Used 0:8:7 in 1:8:12

The message you see when you log out this way depends on the operating system.

Autologout

NOTE: At any time after logging into the host computer, if you do not do any work in a certain amount of time, you will be automatically logged out. The exact amount of time preceding "autologout" depends on the host computer; it is usually about twenty minutes.

Disconnecting from the host computer

After logging out of a host computer that you reached through ARPANET, you should explicitly break or "close" the connection to the host computer, as follows:

You type:	Display shows:
<CTRL-Y>c<RET>	c Closed

If you do not do this, ARPANET will eventually close the connection.

Disconnecting from the network

Finally, you may disconnect from ARPANET by turning off your equipment. If you do not do so, the network will break the connection after a certain amount of time.

REVIEW OF LOGGING IN AND LOGGING OUT

The following examples will help you review much of what you have learned so far in this lesson.

Reaching AUGMENT through ARPANET:

You type:	Display shows:
	AMES TIP 112 #: 66
@i<SP>25<RET>	@i 25
<CTRL-Y>o<SP>	o 43
43<RET>	Trying...
	Open
	Welcome to AUGMENT Service from Office-1
log twain<RET>	@log twain
clemens<RET>	(password)
<RET>	(Account)
	Job 23 on TTY64 8-Feb-79 13:22
	Previous login: 8-Feb-79 08:10
augment<RET>	@augment

Logging out:

You type:	Command window shows:
	BASE C:
l	BASE Logout C:
j	BASE Logout Job OK/T/[M]:
<OK>	BASE Logout Job !

Closing the ARPANET connection:

You type:	Display shows:
<CTRL-Y>c<RET>	c
	Closed

CHECKING THE STATUS OF YOUR JOB: <CTRL-T>

At any time while you are working in AUGMENT, you can get information about the status of your job by typing <CTRL-T>. The information appears in your status window without disturbing what you are doing, even if you are in the middle of a command. You will see "Waiting" if AUGMENT is waiting for you to do something, and "Running" if you are waiting for it.

ATTACHING BACK TO A DETACHED JOB

Signs of a detached job

Network or terminal malfunction may break the connection between your terminal and the AUGMENT host computer, leaving you with what is called a "detached job". You might notice, for example, that nothing happens when you give a command, or that you get no response when you type <CTRL-T>. In either of these cases, you may have a detached job. If you have a detached job, you can resume from where you left off by attaching back to the job.

Locating your job

When you think you have a detached job, you should start over again, connecting to the host computer in the usual way; however, instead of logging into the host computer, your next step should be to give the Where command, which will tell you what jobs are currently logged in under your directory name and whether they are detached. You would do the following, for example, if your directory name were TWAIN:

You type:	Display shows:
where twain<RET>	@where twain Detached, Job 23, AUGMEN @

As shown in this example, "Detached" indicates that a job is detached; for a job that is still connected to a terminal, the command would instead show the terminal number.

One detached job

If you see that there is one detached job under your directory name, you can attach back to the job by giving a command that begins with "attach" (or the abbreviation "att"). This command resembles the command you give to log into the host computer. For example:

You type:	Display shows:
att twain<RET>	@att twain
clemens<RET>	(password)
<RET>	(Tenex Job #)

<LOCAL-RESET>

If you were working in AUGMENT when your job was detached, you should then type <LOCAL-RESET>. You will return directly to your job and can resume from where you left off. If the information on your screen appears garbled, type <LOCAL-RESET> again.

More than one detached job

There may be more than one detached job under your directory name; this can happen, for example, if other people use the same directory name as you when they log in. To attach back to one of the jobs, you must specify the number of that job. For this reason, it would be wise for a user who shares a directory to copy down the job number when first logging into the AUGMENT host computer. This is how such a user might attach back if the job number were 23:

You type:	Display shows:
att twain<RET>	@att twain
clemens<RET>	(password)
23<RET>	(Tenex Job #) 23

You can also use a quicker method of attaching, analogous to the quicker method of logging in described earlier in this lesson.

Job is not detached

If you see that your job is not detached (for example, if you see "TTY64, Job 23, AUGMEN"), then you will need to type another return character at the end of the process of attaching, as follows:

You type:	Display shows:
att twain<RET>	@att twain
clemens<RET>	(password)
23<RET>	(Tenex Job #) 23
<RET>	[attached to TTY64]

Not logged in

If you learn that you are not logged in anywhere, it means that your job was terminated rather than detached. In this case, you can start a new job by logging into the host computer again as usual.

If you log into the host computer while there is still a detached job under your directory name, you will see a message telling you that you have a detached job. To terminate the new job and attach back to the detached job, simply log out and start over again, taking the steps described in the preceding paragraphs.

NOTE: A job that no one has attached to after a certain amount of time will be automatically logged out. The exact amount of time depends on the host computer; it is usually about an hour.

Review of attaching

The following example reviews the procedure for attaching back to a detached job.

You type:	Display shows:
	AMES TIP 112 #: 66
@i<SP>25<RET>	@i 25
<CTRL-Y>o<SP>	o 43
43<RET>	Trying...
	Open
	Welcome to AUGMENT Service from Office-1
where twain	@where twain
<RET>	Detached, Job 23, AUGMENT
att twain<RET>	@att twain
clemens<RET>	(password)
<RET>	(Tenex Job #)
<LOCAL-RESET>	

SUMMARY

Logging in and logging out enable you to reach the computer and then AUGMENT, and to leave AUGMENT and the computer. This lesson has taught you the background and the vocabulary that you need to understand logging in and logging out. You have learned that the computer that has AUGMENT on it is called the "host computer", and that you reach it through a "network" of other computers. You now know how to check your work station and get your terminal turned on and ready to go, how to connect to the network and then the host computer, and how to log into the host computer and enter AUGMENT. Once you have completed your work, you can end your session by following one of the simple logout procedures presented in this lesson.

It is possible for your job to be detached from the host computer due to network or terminal malfunction. You have learned how to recognize and verify that this has happened, and how to attach yourself back and continue to do your work.

If you are now ready to start learning how to use AUGMENT, we recommend that you read the lesson "Beginning Use of AUGMENT".

LIST OF COMMANDS

The following AUGMENT commands were discussed in this lesson.

Logout Job <OK>

Quit <OK>

VOCABULARY

The page numbers indicate where the vocabulary item is discussed in this lesson.

ARPANET: A United States Department of Defense computer network. Page 1

attach: You can reestablish the connection between your terminal and a detached job by "attaching" to the job. Page 11

command: An instruction you give to the computer to perform an action.

command window: The area on your display screen where you see the command you are giving AUGMENT. Page 8

CTRL: This stands for "control". When followed by a dash and a letter and enclosed in angle brackets (<>), it represents the control character you type by holding down the CTRL key while typing the letter in either lowercase or uppercase.

<CTRL-T>: The character you type to ask the host computer to give you information about the status of your job. Page 10

cursor: The mark that moves on your display screen as you move the mouse. Page 4

detached job: A job that remains on the host computer but is no longer connected to a terminal (possibly due to network or terminal malfunction). Page 11

directory name: The name you type when you log into an AUGMENT host computer. Page 6

display: A television-like device that shows information stored in the computer and your interaction with it. Page 4

display terminal: A terminal that consists of a display, a keyboard, a mouse, and a keyset. Page 4

Executive: A set of commands that lets you interact with the operating system that supports AUGMENT. Page 2

file: A work space on the computer, like a file folder in a filing cabinet, that you can fill with information. Page 8

host computer: A computer you reach through a network to perform your work (for example, the computer that has AUGMENT on it). Page 1

ident: A short series of characters that identifies an individual to AUGMENT. Your initial file uses your ident as its name. Page 8

initial file: The file, with the same name as your ident, that automatically appears when you enter AUGMENT. Page 8

job: Your work session on a host computer.

keyboard: An arrangement of keys as on a typewriter, plus keys that have special uses. You can type text and enter commands with the keyboard. Page 4

keyset: A device with five piano-like keys for entering characters into AUGMENT at a display terminal; an alternative to the keyboard. Page 4

log in: To log in means to start a work session on a computer by supplying the necessary identifying information. Page 6

log out: To log out means to end your work session on a host computer. Page 8

Logout command: A command you can give in the Executive to log out of the host computer. Page 9

Logout Job command: A command you can give in AUGMENT to log out of the host computer. Page 8

modem: A device that translates information between computer devices and telephone lines. Page 4

mouse: A hand-sized device, with three buttons on the top, that rolls freely over a flat surface and correspondingly moves the cursor (traveling mark) on the display screen. Page 4

network: A number of interconnected, geographically dispersed computers that systematically allow users to reach one or more host computers. Page 1

node: One of the network's small computers through which information travels from your terminal to a host computer. Page 1

<OK>: This notation means that you are to press either the key for <OK> on the keyboard or the right mouse button, to tell AUGMENT that you have finished giving a command or part of a command. When you type <OK>, you see an exclamation point (!).

operating system: A set of computer programs permanently stored in the host computer to supervise the operation of other programs and perform general support functions. TENEX is an example of an operating system. Page 2

password: A series of characters that you type along with the name you supply when you log into an AUGMENT host computer. This is a private word and will not appear on the display screen when typed. Page 6

Quit command: The AUGMENT command you use to return to the Executive. Page 9

<RET>: This notation represents a return character, the character you type to indicate that you are finished giving a command or part of a command in the Executive.

<SP>: This notation represents a space, that is, what you type with the space bar on the keyboard. We use this notation where it may not be clear that you type a space.

terminal: A device at which a computer user may enter, retrieve, view, print, or manipulate information in the computer.

TIP: Most ARPANET users communicate with the network through a node called a "TIP", which stands for "Terminal Interface Processor". Page 2

Where command: A command you can give in the Executive to learn what jobs are currently logged in under your directory name and whether they are detached. Page 11

work station: Your AUGMENT work area, including the display terminal, modem (optional), and table. It may also include a printing device. Page 4

Price $3.00

WRITING AND READING AN ORGANIZED FILE

Copyright Tymshare Inc., May 1980
20705 Valley Green Drive
Cupertino, California 95014

This document was prepared and composed for printing with AUGMENT by the Office Automation Division of Tymshare, Inc.

AUGMENT Journal Number 48749
Tymshare Document Number 1822

This lesson teaches you how to use the Base subsystem to write and read an organized file in AUGMENT. You should already know how to write simple AUGMENT files, as described in the lesson "Beginning Use of AUGMENT".

CONTENTS

INTRODUCTION

Writing in AUGMENT

Writing in AUGMENT means adding text to a file, either by typing at your keyboard or by copying text already stored online. When you add to an AUGMENT file, you can organize the paragraphs and headings into an outline form that reflects the relationships between the statements. For example, you can enter a document in which the first chapter title will be at position 1, the first subheading in that chapter at position 1a, the first paragraph under that subheading at position 1a1, and so on. This outline form is called "hierarchical structure". This lesson will teach you about hierarchical structure and tell you how to write a file that is organized in this way. You will see how hierarchical structure helps you organize both your thoughts and your writing.

Reading in AUGMENT

You will also learn how hierarchical structure makes it possible for you to look at an online document from many different points of view. Reading in AUGMENT is much more flexible than reading text printed in a book. When reading a book you are limited by the fact that the content of each page is fixed; you have only two choices, to read sections straight through, from beginning to end, or to scan, picking out sentences here and there. In AUGMENT you can read the paragraphs on your screen, one after the other, as if they were on the pages of a book, but you can also look at a document in many other useful ways. Since you control how you see a file on the screen, you can take advantage of the structure of AUGMENT files by looking at only headings, for example, or only headings and the first line of every paragraph; the other text is still in the file, but you look at only what you need to see.

ORGANIZED FILES: HIERARCHICAL STRUCTURE

Hierarchies show relationships.

Human beings tend to classify the things they see around them. One way of classifying things is in hierarchies. (Another way would be in groups of similar objects, for example, all chairs, or all land masses.) Classifying in hierarchies is an important kind of classification because hierarchies allow you not only to talk about groups of things but also to show the relationships between things. For example, it is common to think of the world itself in terms of a hierarchy, as follows: The world is divided into continents, the continents are divided into countries, countries are divided into states, states into counties, and so on.

Because hierarchical arrangement is such an important tool, AUGMENT files are structured so that you can organize statements into an outline form that shows the relationships between the statements. Here is an example showing how you would express the hierarchical organization of the world with AUGMENT structure.

```
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
```

Each item in this outline is a separate statement. Note that although each statement shown here consists of one or two words, a statement in a hierarchy can have any content: a character, a number, a word, a line, a sentence, or a paragraph. What is important in setting up the structure is the relationships between the statements. AUGMENT shows these relationships with indenting. Each statement that is indented three character positions from the preceding statement is subordinate to that statement.

Levels

The hierarchy in the above outline 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. We could fill in this outline by including more information at level 3 (for example, adding the rest of the countries under each continent), or we could add more levels (by adding states under countries, counties under states, and so on, until we reached streets or even individual buildings).

Up and down

To describe the relationships between statements at different levels, we use the words "up" and "down". Statements at level 1 (also called "first-level statements") are one level "up" from statements at level 2 ("second-level statements"). Second-level statements are one level "down" from first-level statements and one level up from third-level statements. Likewise, third-level statements are one level down from second-level statements, two levels down from first-level statements, one level up from fourth-level statements, and so on.

Substatement and upstatement

Two more AUGMENT terms used to describe the relationships between statements are "substatement" and "upstatement". A statement's "substatements" are the statements that are subordinate to it and one level down from it. In the preceding example, Africa has the substatements Ethiopia, Kenya, and Zambia, and Antarctica has no substatements. "Upstatement" is the reverse of substatement; a statement's upstatement is the statement it is subordinate to and one level down from. Thus the upstatement for Ethiopia, Kenya, or Zambia is Africa, and Antartica's upstatement is World.

Substructure

We also refer to the "substructure" of any AUGMENT statement that has substatements. A statement's substructure is *all* the statements subordinate to it, regardless of how many levels they are down from it. In other words, the substructure of a statement is all of its substatements, plus all of their substatements, plus their substatements and so on. For example, the substructure of the World statement is the entire rest of the outline, including all the continents and the countries under each continent. The substructure of the continent Australia consists of the countries Australia and New Zealand. Not every statement has substructure. Since Antartica has no countries under it, it has no substructure, and since none of the individual countries have statements under them, none of them have substructure.

This lesson will show you how to write a structured AUGMENT file containing a table of contents, a familiar example of hierarchical structure. A sample table of contents appears on the following page. (The page is detachable for easy reference while you read the lesson.) When you know how to insert this table of contents, you will know everything necessary to write any structured file you wish.

NOTE: We have chosen an example that consists of only headings, for brevity; of course, the files you write will normally include information under headings.

Starting the file

To create a file named BENEFITS that is to contain this table of contents, you would give the Create File command and type "benefits" as the name of the file. When you do this, AUGMENT automatically makes and displays an origin statement containing information about the file. You would then be ready to insert the table of contents.

INSERTING STATEMENTS AT DIFFERENT LEVELS

To insert statements at different levels, as you would need to do to enter the sample table of contents, you can use the Insert Statement command. After creating the BENEFITS file that is to contain the table of contents, you could begin by inserting the first statement to follow the origin statement of the file. You would do this by giving the Insert Statement command and marking the origin statement as the statement your new statement should follow.

You type:	Command window shows:
i	BASE Insert C:
s	BASE Insert Statement (to follow statement at) M/A:
<MARK>	BASE Insert Statement (to follow statement at) ! L/T/[A]:
Contents<OK>	BASE Insert Statement (to follow statement at) ! Contents!
	BASE C:

The "L/T/[A]:" prompt

As shown in this example, when you insert a statement you see the prompt "L/T/[A]:" and can respond to the "T" in this prompt by typing your new statement and ending with <OK>. However, you have another choice. Before entering the new statement, you can indicate that you want to put it on a different level than the previous one. The "L" in "L/T/[A]:" means that you can specify the level of the new statement relative to the level of the statement it is to follow; we call this "adjusting the level" of the new statement. This ability to put statements at different levels as you enter them allows you to arrange the statements in your file into a hierarchy as you type.

How to adjust the level

In response to the "L" in "L/T/[A]:", you can specify that your new statement should be down from, up from, or at the same level as the statement it is to follow. To indicate that your new statement should be down a level (and therefore a substatement of the statement it is to follow), type "d<OK>". To put your new statement up a level, type "u<OK>"; it will be inserted at the next highest level (that is, the same level as the upstatement of the statement it is to follow). If you want the statement to be up more than one level, type another "u" for each additional level. For example, "uuu<OK>" will put your new statement three levels higher than the one you insert it to follow. If you want your new statement to be at the same level, you can type <OK> in response to the "L" or, as you have already learned, you may respond to the "T" by simply typing the new statement.

Contents

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

Inserting to follow the origin statement

There is one place where you do not need to indicate a level change at the "L/T/[A]:" prompt, even though the statement you insert will be at a different level than the one it follows: You do not need to specify a level change when you are inserting to follow the origin statement. The origin statement is the upstatement of all the first-level statements you add to a file; in other words, the origin statement is at level 0 and all the other statements in the file are the substructure of the origin statement. AUGMENT knows this and thus automatically puts at level 1 any statement you insert to follow the origin statement. When inserting the first statement as shown in the previous example, you took advantage of this when you responded to the "L/T/[A]:" prompt by simply typing your new statement. Although you did not tell AUGMENT to change the level of the new statement, AUGMENT made it a first-level statement, one level lower than the origin statement.

Example of inserting down a level

Inserting to follow the origin statement is the only place AUGMENT will automatically adjust the level of your new statement. When entering the second statement of the table of contents, for example, you will have to adjust the level yourself. To do this, give the Insert Statement command again and mark the first statement you inserted, "Contents", as the statement your new statement is to follow. At the "L/T/[A]:" prompt, type "d<OK>" to indicate that your new statement should be down a level from "Contents". You will see a new prompt, "M/T/[A]:". AUGMENT is waiting for your new statement; type "Part I. Holidays" and end with <OK>. The process of entering the second statement looks like this:

You type:	Command window shows:
i	BASE Insert C:
s	BASE Insert Statement (to follow statement at) M/A:
<MARK>	BASE Insert Statement (to follow statement at) ! L/T/[A]:
d<OK>	BASE Insert Statement (to follow statement at) ! d! M/T/[A]:
Part I.	BASE Insert Statement (to follow statement at) ! d! Part I.
Holidays<OK>	Holidays!
	BASE C:

After the final <OK>, AUGMENT will display the statement at the place and level you have indicated. The new statement will follow the previous one immediately, with no blank line between them. (Later in this lesson you will learn how to see the file with blank lines between the statements.)

You are now ready to enter Sections I and II of Part I. Because these are logically subsections of Part I, you would want to take advantage of AUGMENT's hierarchical structure and make them substatements of Part I. You could mark Part I as the statement you want your new statement to follow and insert Section I down a level from it, as follows:

You type:	Command window shows:
i	BASE Insert C:
s	BASE Insert Statement (to follow statement at) M/A:
<MARK>	BASE Insert Statement (to follow statement at) ! L/T/[A]:
d<OK>	BASE Insert Statement (to follow statement at) ! d! M/T/[A]:
Section I.	BASE Insert Statement (to follow statement at) ! d! Section I.
...<OK>	Legal days off. List of legal holidays, including national, local, and company–specific days.!
	BASE C:

Example of inserting at the same level

You could then mark Section I as the statement your next new statement should follow and insert Section II. Since Section II is to be at the same level as Section I, you do not have to adjust the level. Thus at the "L/T/[A]:" prompt you could just respond to the "T" by typing your new statement and ending it with <OK>.

You type:	Command window shows:
i	BASE Insert C:
s	BASE Insert Statement (to follow statement at) M/A:
<MARK>	BASE Insert Statement (to follow statement at) ! L/T/[A]:
Section II.	BASE Insert Statement (to follow statement at) ! Section II.
...<OK>	Absence from work. Rules governing permissible days off with and without pay.!
	BASE C:

Or you could respond to the "L" by indicating "same level" with <OK>, and then go on to type Section II at the "M/T/[A]:" prompt.

You type:	Command window shows:
i	BASE Insert C:
s	BASE Insert Statement (to follow statement at) M/A:
<MARK>	BASE Insert Statement (to follow statement at) ! L/T/[A]:
<OK>	BASE Insert Statement (to follow statement at) ! ! M/T/[A]:
Section II.	BASE Insert Statement (to follow statement at) ! ! Section II.
...<OK>	Absence from work. Rules governing permissible days off with and without pay.!
	BASE C:

<SP> when adjusting levels

If you find it more convenient or easier to type, you can use <SP> instead of <OK> when you adjust the level of your new statement. For example, "u<SP>" would put your new statement up one level and <SP> would put it at the same level.

Statements beginning with <SP>, d<SP>, u<SP>, and so on

You have learned that if you want to insert a statement at the same level as the statement it is to follow, you can respond to the "L" in the "L/T/[A]:" prompt by typing <OK> or <SP>, or you can respond to the "T" in this prompt by typing the statement. However, if your new statement begins with a space or one or more d's or u's followed by a space, or if it consists entirely of one or more d's or u's, you cannot just type the statement. In this case, you must first respond to the "L" in the prompt by typing <OK> or <SP>; only then can you type the

statement. If you do not do this, AUGMENT will take the first character or characters you type as adjusting the level of your statement and the rest of what you type as the statement itself.

Example of inserting up a level

Once you have finished entering the Sections under Part I, you can begin entering Part II. Since the Part II statement is logically on the same level as the Part I statement, one level higher than the Sections under Part I, you can insert it to follow Section II of Part I up a level.

```
You type:       Command window shows:

i               BASE Insert C:
s               BASE Insert Statement (to follow statement at) M/A:
<MARK>          BASE Insert Statement (to follow statement at) ! L/T/[A]:
u<OK>           BASE Insert Statement (to follow statement at) ! u! M/T/[A]:
Part II.        BASE Insert Statement (to follow statement at) ! u! Part II.
 ...<OK>         Recreational Activities!
                BASE C:
```

At this point you could continue giving Insert Statement commands, marking the statement your new statement should follow, adjusting the level when necessary, typing the statement, and ending with <OK>, until you have entered the entire table of contents. But a faster way to do this would be to use "insert mode", as described below.

INSERTING A SERIES OF STATEMENTS USING INSERT MODE

Whenever you want to insert a series of statements where each new statement directly follows the previous one, you can, of course, give a separate Insert Statement command for each statement; however, it saves time to use "insert mode". In insert mode you do not have to give the Insert Statement command over and over; instead you just type your statements one after another, adjusting the level when necessary, and the statements are inserted into your file in the order you type them.

Entering and leaving insert mode

To get into insert mode, you type <INS>. The following paragraphs tell where you can do this. To leave insert mode, you type <CD>, as you would to cancel a command.

<INS> at "BASE C:" after Insert Statement

You can type <INS> at "BASE C:" after you have given an Insert Statement command; AUGMENT will assume that you want to begin inserting right after the statement you just entered. When you type <INS>, you see an "L/T/[A]:" prompt. This is the same prompt as in the Insert Statement command, and you should respond to it in the same way: Adjust the level of your new statement if necessary, type the statement, and end with <OK>. AUGMENT will insert the statement into your file and prompt you again with "L/T/[A]:". You can then enter your next statement to follow the one just inserted, adjusting the level if necessary, follow this with the next statement, and so on, until you have entered all the statements. When you are finished, type <CD> at the "L/T/[A]:" prompt.

Continuing with the example of inserting the BENEFITS table of contents, suppose you have just entered the Part II statement with the Insert Statement command and you want to use insert mode to enter the remaining statements.

You type:	Command window shows:
	BASE C:
<INS>	L/T/[A]:
d<OK>	d! M/T/[A]:
Section I.	d! Section I. Noon time. Location for classes and
...<OK>	dressing rooms and regulations concerning their use.!
	L/T/[A]:
Section II.	Section II. Scheduled activities. List of available
...<OK>	classes and location for each calendar year.!
	L/T/[A]:
Section III.	Section III. Company–sponsored classes. Off–site
...<OK>	classes and registration information.!
	L/T/[A]:
u<OK>	u! M/T/[A]:
Part III.	u! Part III. Education!
...<OK>	L/T/[A]:

Notice that just as when you use the Insert Statement command, you can respond to the "L/T/[A]:" prompt either by adjusting the level of your new statement (as when entering the Section I and Part III statements above) or, if the new statement is at the same level as the one it follows, by typing the statement (as when entering Sections II and III).

Inserting outside the file window

You could continue in this way to enter the entire table of contents; however, you would not see all of it in your file window. At some point the file window would be too small to display all the statements in your file. Later in this lesson you will learn how to see the rest of your file; right now it is important to realize that as long as you type each statement and end it with <OK>, it will be added to your file even though you cannot see it being added. Of course, the command window will always show the statement you are currently typing.

When entering the "Notes" statement, be sure to type "uu<OK>" after the "L/T/[A]:" prompt, since this statement is two levels up from the statement it follows (Section III of Part III). Remember that to leave insert mode you must type <CD> after the "L/T/[A]:" prompt.

<INS> at any "BASE C:"

While it is perhaps safest to enter insert mode after giving an Insert Statement command, you can, in fact, type <INS> any time you are prompted with "BASE C:". When you type <INS>, AUGMENT will assume you want to begin inserting after your current statement, that is, the last statement you inserted, edited, or jumped to. If you are inserting into a newly created file, or just beginning to work, the origin statement will be your current statement and the first statement you insert will follow the origin statement. Because it may be difficult at times to determine just what your current statement is, it is a good

practice to begin inserting with the Insert Statement command. After the Insert Statement command, when you know where you are, you can safely enter insert mode.

<INS> at the end of Insert Statement

You can also type <INS> in place of the final <OK> in an Insert Statement command. Because you are replacing <OK> with <INS>, you will not see the exclamation point (!) that lets you know AUGMENT has received an <OK>. AUGMENT will carry out the Insert Statement command as usual and then give you an "L/T/[A]:" prompt just as if you had typed <INS> at "BASE C:". Type <OK> at the end of each statement you insert, and type <CD> after the "L/T/[A]:" prompt when you are done.

INSERTING INDIVIDUAL STATEMENTS ANYWHERE

After entering a series of statements into a file, you may find that you want to add statements to what you entered. To add statements anywhere in a file, use the Insert Statement command, mark the statement you want the new statement or statements to follow, and proceed as usual. If you are adding one statement, end the Insert Statement command with <OK>; if you are adding a series of statements, you may want to enter insert mode by typing <INS> either in place of the final <OK> or at "BASE C:" after the Insert Statement command. For example, to add an "Introduction" before Part I in the BENEFITS table of contents, you could use the Insert Statement command, mark the Contents statement, and add the Introduction statement to follow it, down one level.

The new statement follows the substructure.

When you insert a statement to follow another at the same level and the statement it is to follow has substructure, AUGMENT will put the new statement *after* the substructure. If, for example, you added a Part IV statement to the table of contents and indicated that it should follow Part III at the same level, AUGMENT would put your new Part IV at the same level as Part III, but after Section III of Part III. (You will be able to check this later in this lesson, after you learn about reading and moving around in a file.)

More than one way to add a statement

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 Section III of Part 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. In fact, the same thing would happen if you inserted a statement to follow Section I or II of Part III up one level.

In general we can say that when you insert a statement, AUGMENT puts your new statement in the next available place at the level you specify, without disturbing the substructure of the preceding statement at that level. Thus inserting a statement to follow a third-level statement up one level puts the new statement in the next available place at level 2, inserting it up two levels puts it in the next available place at level 1,

and inserting it up three levels puts it immediately after the origin statement.

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 what you added. To read statements that are not displayed on your screen, you can use one of the many forms of the Jump command. This command is called "Jump" because it lets you jump from one statement to another without reading through all the statements between them. When you jump to a certain statement, AUGMENT displays that statement at the top of the file window and shows as much of what follows as will fit on the screen. AUGMENT offers many ways of jumping around in a file and helping you find a particular statement. In this lesson you will learn several basic kinds of Jump commands; later you will learn others.

The "M/C:" prompt

After you type "j" for "Jump", your command window will show "Jump (to)" followed by the prompt "M/C:". AUGMENT is asking you to specify where you want to jump. You can respond to the "M" in this prompt by marking the statement you want to jump to or to the "C" in this prompt by giving a command word describing where you want to jump.

Jump (to) <MARK>

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

You type:	Command window shows:
j	BASE Jump (to) M/C:
<MARK>	BASE Jump (to) ! V:
<OK>	BASE Jump (to) ! !
	BASE C:

Notice that no matter what character you mark in a statement, the Jump command always moves the *beginning* of that statement to the top of the file window.

Jump (to) Origin

After reading the entire table of contents, you could return to the beginning of the file by using the Jump (to) Origin command. When you see the "M/C:" prompt after typing "j" for "Jump", respond to the "C" by typing "o" for the command word "Origin", indicating the origin statement of the file. Then mark any character in the file and end with an <OK> after the "V:" prompt. AUGMENT will display the origin statement of your file at the top of the file window.

You type:	Command window shows:
j	BASE Jump (to) M/C:
o	BASE Jump (to) Origin (of file) M/A:
<MARK>	BASE Jump (to) Origin (of file) ! V:
<OK>	BASE Jump (to) Origin (of file) ! !
	BASE C:

Jump is universal.

NOTE: The Jump command verb is "universal" within AUGMENT, that is, it is available in every subsystem of AUGMENT (except subsystems that have been designed for special, limited applications).

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

You give a reference point.

Four important command words you can use with the verb "Jump" to get to a statement not displayed in the file window are Next, Back, Successor, and Predecessor. Like "substatement" and "upstatement", these terms describe structural relationships between statements in an AUGMENT file. They are illustrated in Figure 1 using our original example, the World outline. When you use these command words, you must indicate a statement to serve as a reference point so that AUGMENT will know which statement the one you want is next from, back from, and so on.

Next and back

The "next" statement from the statement you indicate is the statement immediately following it, and the statement "back" from the statement you indicate is the statement immediately preceding it, regardless of level. For example, to jump to the statement following the last statement in your file window, you can give the Jump (to) Next command and mark this last 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) M/C:
<SP>n	BASE Jump (to) Next (from) M/A:
<MARK>	BASE Jump (to) Next (from) ! V:
<OK>	BASE Jump (to) Next (from) ! !
	BASE C:

If you then give the Jump (to) Back command and mark the statement at the top of your file window, the immediately preceding statement will appear at the top of the window.

You type:	Command window shows:
j	BASE Jump (to) M/C:
b	BASE Jump (to) Back (from) M/A:
<MARK>	BASE Jump (to) Back (from) ! V:
<OK>	BASE Jump (to) Back (from) ! !
	BASE C:

Every statement except the last statement in a file has a "next" statement, and every statement except the origin statement has a statement that is "back" from it.

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. If a statement is the last or only substatement under a particular upstatement, it has no successor. The "predecessor" of a statement is the preceding statement that is at the same level and has the same upstatement. Just as not every statement has a successor, not every statement has a predecessor. If a statement is the first or the only substatement under a particular upstatement, it has no predecessor.

Jumping to the successors and predecessors of statements will let you quickly view the various sections of a document that are at the same level. If, for example, the Notes statement of the BENEFITS table of contents were showing on your screen, and you gave the Jump (to) Predecessor command and marked that statement, you would see "Contents" at the top of the file window.

You type:	Command window shows:
j	BASE Jump (to) M/C:
p	BASE Jump (to) Predecessor (of) M/A:
<MARK>	BASE Jump (to) Predecessor (of) ! V:
<OK>	BASE Jump (to) Predecessor (of) ! !
	BASE C:

If you then jumped to the successor of "Contents", you would see "Notes" at the top of the file window.

If you try to jump to Next, Back, Successor, or Predecessor when there is none relative to the statement you indicate, the statement you indicate will appear at the top of the file window. For example, since "Addendum" is the last first-level statement in the BENEFITS file, you could not jump to its successor; if you tried to, AUGMENT would display "Addendum" itself at the top of the file window.

CHANGING VIEWS: VIEWSPECS

Your view changes, not your file.

AUGMENT enables you to control not only what part of your file you see but also how you see it. Just as you can jump around in a file using the various Jump commands to look at only the information you need, you can also control the way information is displayed on your screen. You can, for example, display your file with blank lines between the statements. When you view your file with blank lines, it does not mean AUGMENT has put spaces into your file; AUGMENT merely displays your file in a different way, leaving its contents just as you entered them.

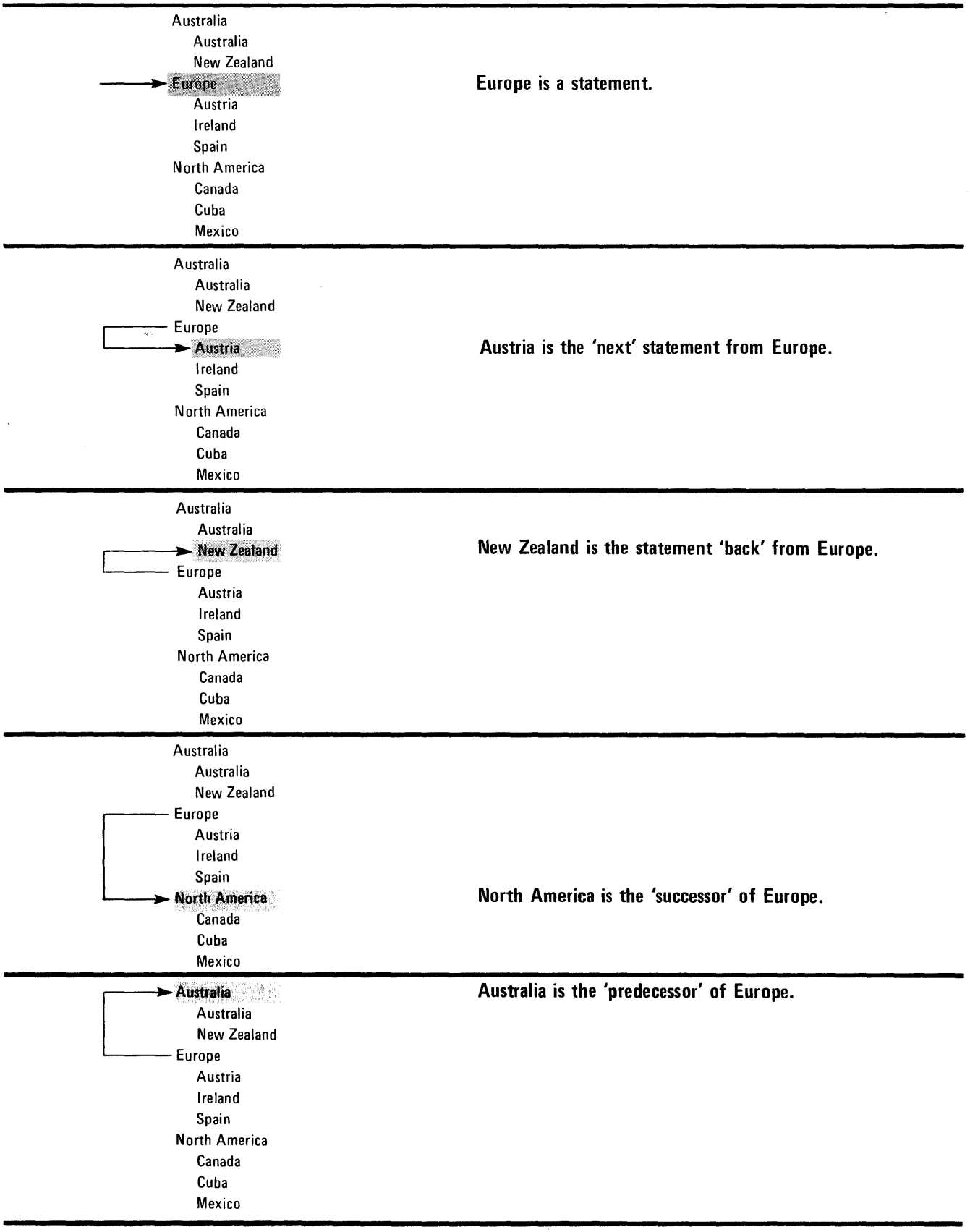


FIGURE 1: NEXT, BACK, SUCCESSOR, AND PREDECESSOR

Set Viewspecs

The way you tell AUGMENT how you want to see the information in your file window is by setting "viewspecs". A "viewspec" is a single letter that specifies the kind of view you want of your file. Some of the viewspecs that are currently controlling your view are displayed in the viewspec window (the upper right corner of your screen). When you enter AUGMENT, 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, which begins as follows:

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

The "V:" prompt 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.

Viewspecs are an important tool in working with your files. By using viewspecs effectively you can make working and reading, as well as searching your files, much easier. The rest of this lesson will introduce you to some basic viewspecs.

VIEWING BLANK LINES BETWEEN STATEMENTS

Viewspecs y and z

Viewspecs y and z allow you to control whether AUGMENT displays blank lines between statements.

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

Thus, to see your file with blank lines between the statements, you would set viewspec y.

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

Many people find reading files much easier with viewspec y. You can change your view in other ways, by setting other viewspecs, while keeping the blank lines between statements on. To see the file again without blank lines, just set viewspec z as you set viewspec y above.

VIEWING STATEMENT NUMBERS

Statement numbers

Viewspecs enable you to display helpful information about the statements in your file. For example, you can set a viewspec to 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 statement number of the origin statement is always 0. The first statement after the origin statement is statement 1, the first substatement after statement 1 is statement 1A, the successor of statement 1 is statement 2, and so on. By viewing your file with statement numbers, you can easily determine the level and place of any statement.

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

Viewspec m

To see the statement numbers for your file, set viewspec m.

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

For example, the statement numbers for the original BENEFITS table of contents would appear as follows:

1 Contents
  1A Part I...
    1A1 Section I...
    1A2 Section II...
  1B Part II...
    1B1 Section I...
    1B3 Section II...
    1B3 Section III...
  1C Part III...
    1C1 Section I...
    1C2 Section II...
    1C3 Section III...
2 Notes
3 Addendum

Statement numbers change.

Since statement numbers indicate exactly where statements lie within the hierarchical structure, they will change if you change the structure of your file, such as by adding statements. If, for example, you inserted an "Introduction" to follow "Contents" down a level, the Introduction would become statement 1A, Part I would become statement 1B, Section I of Part I would become 1B1, and all the remaining statement numbers would similarly change.

Statement numbers are not text.

NOTE: Like the blank lines between statements that you can get with viewspec y, the statements numbers you get with viewspec m are not part of the text of your file. They are simply a convenience provided by AUGMENT. You do not enter them and you cannot mark them the way you can mark text within a statement.

Viewspec n

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

LEVEL AND LINE CLIPPING VIEWSPECS

To get an overall picture of a book, a reader often looks at a list of the chapter titles in the table of contents. In AUGMENT, you can easily display only the headings in a document by using viewspecs to show only the statements at level 1. You can also show only the statements at levels 1 and 2, or only those at levels 1 through 3, and so on. This is called "level clipping". In a similar way you can control the number of lines displayed for each statement. This is called "line clipping". Since line clipping viewspecs let you fit more statements on the screen, they are useful in scanning the overall structure of a file. You will find level and line clipping viewspecs very helpful in both writing and reading.

The first line of the viewspec window

As you learn to use viewspecs to change the way you see your file, it will help you to look at the viewspec window. The top line of this window tells you how many levels and lines are being displayed. The number on the left indicates the number of levels displayed and the number on the right indicates the number of lines displayed. When you enter AUGMENT, the viewspec window shows "ALL ALL", meaning that you will see statements at all levels and all the lines of every statement.

Level Clipping Viewspecs

You can use level clipping viewspecs to display statements at the first level of your file, the first two levels, the first three levels, or any number of levels up to 63. When you first try using level clipping viewspecs, you may find it helpful to have viewspec m set so you can see your file with statement numbers; the statement numbers will enable you to easily determine the level of each statement. These are the level clipping viewspecs:

Viewspec:	Means:
a	Show one level less
b	Show one level more
c	Show all levels
d	Show first level only

Viewspec d

For a view of your file that shows only first-level statements, you would use the Set Viewspecs command with viewspec d to "show first level only". If you set viewspec d and have not set any line clipping

viewspecs, you will see "1 ALL" in the viewspec window, meaning that one level and all lines are being displayed.

Viewspec b

After setting viewspec d, you can set viewspec b, which means "show one level more"; you will then see statements at levels 1 and 2 in the file window and "2 ALL" in the viewspec window. Figure 2 shows how the World outline would look with viewspec d and viewspec b set. You could obtain the same type of view of your BENEFITS file by doing the following:

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

If you looked at the BENEFITS file after setting viewspecs d and b, you would see the statements at levels 1 and 2, and you would not see the Section statements (which are at level 3). To display three levels of statements, you could add another viewspec b to "show one level more". Similarly, to display four levels, you would add still another viewspec b, and so on. Each b will add a level to what you had before.

Viewspec a

Viewspec a means "show one level less". If statements at three levels were being displayed and you wanted to see only the statements at levels 1 and 2, you could set viewspec a.

More or less than what?

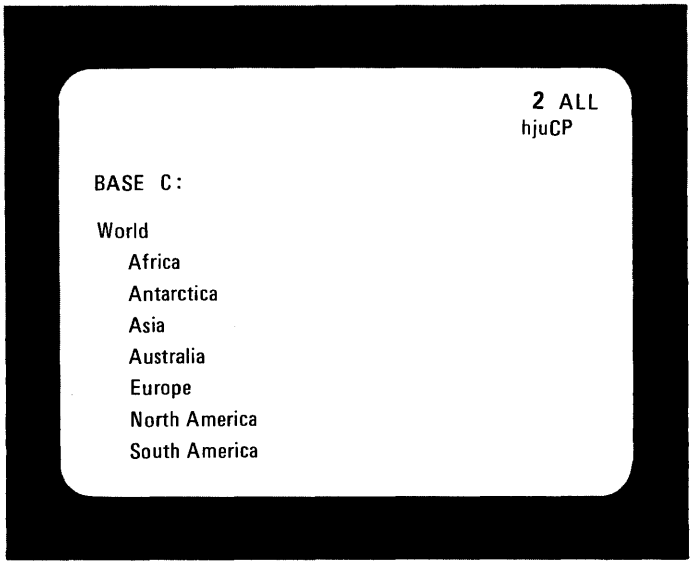
It is important to understand that level clipping viewspecs display a number of levels more or less than the number in effect from previous viewspecs, not necessarily a number of levels more or less than what you see on your screen. For example, ALL levels is 63 levels. Thus, if you set viewspec a when you have ALL levels showing, you will have 62 levels; this will change your view only if the part of the file you are looking at contains a statement at level 63.

When the top statement is at a clipped level

NOTE: Whenever you clip levels and the statement at the top of the file window is at a lower level than what you want to see, AUGMENT will continue to show that statement and any statements immediately following it at the same level; depending on the structure of your file, other statements at clipped levels may also appear before the first statement that is at the level you limited your view to. For example, if you set viewspec d to "show first level only" when there are two second-level statements at the top of your file window, you will see these second-level statements and then only first-level statements after them.

Viewspec c

To return to a view showing statements at all levels in your file, you can use viewspec c. This viewspec is in effect when you enter AUGMENT. If, after looking at your BENEFITS file with the levels clipped, you again wanted to "show all levels", you could set viewspec c as follows:



This is the outline as it is displayed with viewspecs d and b.

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

This is the entire outline. The shaded statements are the ones displayed above.

FIGURE 2: WORLD OUTLINE WITH VIEWSPECS d AND b

```
You type:       Command window shows:

<SP>se          BASE  Set C:
v               BASE  Set Viewspecs V:
c<OK>           BASE  Set Viewspecs c!
                BASE C:
```

Line Clipping Viewspecs

You can use line clipping viewspecs to see the first line of the statements displayed, the first and second lines, the first three lines, and so on, up to all lines of each statement displayed. These are the line clipping viewspecs:

Viewspec:	Means:
q	Show one line less
r	Show one line more
s	Show all lines
t	Show first lines only

Comparison with level clipping viewspecs

As you can see from this table, line clipping viewspecs work much like level clipping viewspecs, although of course they control different things. Level clipping viewspecs control which statements you see, but do not affect the number of lines you see of each statement. With line clipping viewspecs, on the other hand, you can control how many lines you see of each statement, but you cannot control which statements are displayed. The actual statements displayed will be those at the levels you have selected with your level clipping viewspecs.

Viewspec t

To see the only the first line of each statement displayed, use viewspec t.

```
You type:       Command window shows:

<SP>se          BASE  Set C:
v               BASE  Set Viewspecs V:
t<OK>           BASE  Set Viewspecs t!
                BASE C:
```

If you set viewspec t when statements at all levels are being displayed, your viewspec window will show "ALL 1", meaning "all levels and one line". This view will allow you to fit more statements on the screen and is useful when you want an overview of the organization of a file or need to see the exact position and context of individual statements or parts of a document.

Viewspec r

After setting viewspec t, you can set viewspec r to "show one line more"; you will then see the first two lines of every statement displayed. Or you could see three lines by using "r" twice after setting viewspec t, and so on, adding an "r" for each additional line you wanted to see. For example, after looking at only the first lines, you might want to see the same statements with three lines showing, and you could do this:

```
You type:       Command window shows:

<SP>se          BASE  Set C:
v               BASE  Set Viewspecs V:
rr<OK>          BASE  Set Viewspecs rr!
                BASE C:
```

Viewspec q

To "show one line less", use viewspec q. For example, if you have three lines showing and you set viewspec q, the last line will disappear and you will see only the first two lines of each statement displayed.

Like level clipping viewspecs that show one level more or one level less, line clipping viewspecs that show one line more or one line less add to or subtract from the number in effect from previous viewspecs, not necessarily from what you see on your screen. The maximum number (shown as "ALL" in the viewspec window) is 63.

Viewspec s

When you want to return to a full view of every statement displayed, you can use viewspec s to "show all lines". This viewspec is in effect when you enter AUGMENT.

Combining Level and Line Clipping Viewspecs

1 level and 1 line

Level and line clipping viewspecs are very handy when used separately, but they can be even more effective when used together. Suppose, for example, you are working with a large file, and you decide to set viewspec d so you can scan the statements at level 1. If you show every line, you may not be able to see all the first-level statements at once, because they may not fit on the screen. To see as many of the statements as possible, you could add a line clipping viewspec; in addition to viewspec d, you could set viewspec t. Your viewspec window would then show "1 1", meaning "one level and one line". You would see only the first lines of the first-level statements in your file, so more of your file would fit on the screen.

```
You type:       Command window shows:

<SP>se          BASE  Set C:
v               BASE  Set Viewspecs V:
dt<OK>          BASE  Set Viewspecs dt!
                BASE C:
```

2 levels and 1 line

If, after looking at your file with viewspecs d and t, you wanted to add another level to your view, you would set viewspec b. You would then see the first lines of the statements at levels 1 and 2 in your file and your viewspec window would show "2 1", for "two levels and one line". This type of view is useful when you are writing or reading a document and you want to see only the major headings (at level 1) and the subheadings under them (at level 2). For example, this is how the original BENEFITS table of contents would look with this view:

Contents
  Part I. Holidays
  Part II. Recreational Activities
  Part III. Education
Notes
Addendum

ALL levels and 1 line

Once you have seen your file with viewspecs d, b, and t, you might want to display the first lines of all the statements in the file so that you can scan its structure. You would do this by setting viewspec c to "show all levels"; AUGMENT would continue to show only one line of each statement displayed, and you would see "ALL 1" in the viewspec window. Continuing from the previous example, this is what the table of contents would look like if you then set viewspec c:

Contents
  Part I. Holidays
    Section I. Legal days off. List of legal holidays, including
    Section II. Absence from work. Rules governing permissible
  Part II. Recreational Activities
    Section I. Noon time. Location for classes and dressing rooms
    Section II. Scheduled activities. List of available classes and
    Section III. Company-sponsored classes. Off-site classes and
  Part III. Education
    Section I. Qualification. Information regarding hours and
    Section II. Costs. Regulations regarding reimbursement for
    Section III. Advanced degrees. Applications, requirements, and
Notes
Addendum

NOTE: The line length shown here may differ from what you see on your terminal.

ALL levels and 2 lines

If you then wanted to add a second line to the statements being displayed, you would set viewspec r. AUGMENT would show one more line of each statement, and you would see "ALL 2" in the viewspec window. An additional viewspec r would change the above view to this:

Contents
  Part I. Holidays
    Section I. Legal days off. List of legal holidays, including national, local, and company-specific days.
    Section II. Absence from work. Rules governing permissible days off with and without pay.
  Part II. Recreational Activities
    Section I. Noon time. Location for classes and dressing rooms and regulations concerning their use.
    Section II. Scheduled activities. List of available classes and location for each calendar year.
    Section III. Company-sponsored classes. Off-site classes and registration information.
  Part III. Education

Section I. Qualification. Information regarding hours and requirements for work-related classes.
Section II. Costs. Regulations regarding reimbursement for registration and tuition fees.
Section III. Advanced degrees. Applications, requirements, and program development.
Notes
Addendum

Since no statement consists of more than two lines, this view shows you the full table of contents. (As when you first entered the statements into the file, you would not be able to see all of them at once on your screen.) Note that if you then inserted a statement of more than two lines into this file, only the first two lines of it would appear in the file window.

Once you had this view, you could take away the third-level statements using viewspec a, or subtract a line using viewspec q, and so on, continuing to combine the level and line clipping viewspecs as desired. Because the combination of level and line clipping is so helpful and so frequently used, AUGMENT offers the following pair of viewspecs:

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

Viewspec x

Viewspec x combines viewspecs d and t. Setting viewspec x will display the first lines of the first-level statements in your file and will cause your viewspec window to show "1 1". Viewspec x not only combines level and line clipping, but can itself be conveniently used in combination with other level and line clipping viewspecs. For example, to show the first lines of the statements at levels 1 and 2 in your file, you could set viewspecs x and b, as follows:

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

This will have the same effect as setting viewspecs d, b, and t. Likewise, if you want to see the first two lines of the statements at level 1, you can set viewspecs x and r, which would have the same effect as setting viewspecs d, t, and r.

Viewspec w

Just as viewspec x is a combination of viewspecs d and t, viewspec w is a combination of viewspec c, which shows all levels, and viewspec s, which shows all lines. Setting viewspec w is an easy way of counteracting any level and line clipping you may have done and restoring the "all levels and all lines" view you had when you entered AUGMENT. When you set viewspec w, you will again see "ALL ALL" in your viewspec window.

CHANGING VIEWS AS YOU READ: JUMPING WITH VIEWSPECS

When you give any of the Jump commands you have learned in this lesson (for example, if you type "j" for "Jump" and then mark the statement you want to jump to), AUGMENT prompts you with "V:". Just as when you see this prompt in the Set Viewspecs command, you can respond by typing any viewspecs followed by <OK>. When AUGMENT displays the statement you have jumped to, it will observe the viewspecs you specified. Thus you can change viewspecs while you jump, all in one command.

Searching by jumping and changing views

Changing viewspecs while you jump is a good way to look for something in a structured file. You could first show only one line of the statements at level 1, then 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 that shows all lines and all levels so that you could read the entire paragraph and whatever follows it.

Searching BENEFITS

Suppose, for example, that you were unfamiliar with the BENEFITS file and needed to locate the section dealing with advanced degrees. You could begin by viewing the file with viewspec x. You would then see that the three main parts of the file are Contents, Notes, and Addendum, and you would want to look further under Contents. You could then jump to "Contents" and at the same time set viewspec b to see another level of statements, that is, the "Part" statements. After noting that Part III deals with education, you could then jump to that statement, setting viewspec b again to add yet another level. The "Section" statements would be displayed under Part III and you would see that Section III of Part III was the section you wanted. Finally you could jump there, setting viewspec w to show all lines and all levels; AUGMENT would display the entire statement, and you would be able to read or modify it as you wished.

RESTORING YOUR INITIAL VIEWSPECS

Reset Viewspecs

Whenever you have changed any of your viewspecs and want to go back to the view you had when you entered AUGMENT, you could, of course, use the Set Viewspecs command and specify individually all the viewspecs necessary to get back to this view; however, it is probably more convenient to use the Reset Viewspecs command.

You type:	Command window shows:
<SP>res	BASE Reset C:
v	BASE Reset Viewspecs (to defaults) OK:
<OK>	BASE Reset Viewspecs (to defaults) !
	BASE C:

REVIEW OF VIEWSPECS

You have now learned many of the most frequently used and convenient of the viewspecs AUGMENT offers. The following table is intended as a review of the viewspecs presented in this lesson and as a quick reference table. To set these viewspecs, use the Set Viewspecs command; to return to the viewspecs in effect when you entered AUGMENT, use the Reset Viewspecs command. For a complete explanation of these commands and of the individual viewspecs, see the sections introducing them.

Viewspec:	Means:
y	Blank lines between statements on
z	Blank lines between statements off
m	Statement numbers on
n	Statement numbers off
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
w	Show all levels and all lines
x	Show one level and one line only

Viewspecs modify or counteract each other.

Whenever you set viewspecs, they either modify or counteract the viewspecs already in effect. For example, viewspec n counteracts viewspec m and vice versa; viewspec b modifies the view you get with viewspec d (by adding a level); and you can use viewspecs b and r to modify the effect of viewspec x (by adding a level and a line).

If you use a single Set Viewspecs command to set several viewspecs, the result will be the same as if you had set each viewspec with a separate command. If the viewspecs you set modify each other, AUGMENT will interpret the viewspecs in the order that you type them, modifying each viewspec by what follows it. For example, if you type "xbr" and then, before ending with <OK>, you realize that you do not actually want to set viewspec b, you can type "a" at that point; viewspec a will take away the additional level added by viewspec b, and you will see two lines of only statements at level 1. Likewise, 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 last viewspec you type will take effect. So if you set viewspecs "dc", viewspec c will take effect and viewspec d will be ignored.

EXERCISES

1. What is the upstatement of the Section II of Part III in the BENEFITS table of contents? What is the upstatement of "Addendum"?

2. What is the statement "next" from Section I of Part III? What is the one "back" from it?

3. What is the successor of the Part I statement? What is the predecessor of Section I of Part I? What would happen if you gave the Jump (to) Predecessor commmand and marked Section I of Part I?

4. If you inserted a statement to follow Section II of Part II, one level up from Section II, where would you expect to see the statement displayed? Why?

5. Name three ways that you could add a statement before the "Notes" statement and on the same level as "Notes".

6. What is the statement number of "Contents"? What is the statement number of its upstatement? If you added a statement following "Contents" and one level down from it, what would be the statement number of this new statement? If you added a statement before "Contents", what would be the statement number of "Contents" and the statement you added below "Contents"?

7. How many levels of statements and lines of each statement must you see to read through all the text in a file? How many levels and lines give you an overview of as many statements as will fit on the screen at once?

8. What command would you give to see only one line of the first-level statements in your file? What would you then do to see two more lines of each first-level statement and, after that, to see one more level of statements? After doing all these operations, what would you see in the first line of your viewspec window?

SUGGESTED PROJECT

To gain more experience in writing and reading structured files, make a new file and type in the management structure of your department or organization, that is, enter the hierarchy of managers and non-managers. Use level clipping viewspecs to see only personnel at the top level of the hierarchy, then only those at the top two levels, and so on.

SUMMARY

The ability to arrange the information in your AUGMENT 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 AUGMENT, you are encouraged to plan the overall structure of the document, and this often means that a document written in AUGMENT is more carefully organized than one written by hand or on a typewriter. Further, you can use level and line clipping viewspecs to see the organization of your document at a glance.

This lesson has taught you the commands you need to write a hierarchically structured file. You have learned how to enter statements in insert mode and how to adjust their levels at the "L/T/[A]:" prompt. In addition, you have learned some of the basic viewspecs that make working with and reading structured files much easier.

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 intermediate lesson on editing will teach you how to do this. You will also find that a document with a consistent structure is much easier to format for printing.

LIST OF COMMANDS

Insert Statement (to follow statement at) LOCATION TYPEIN
Insert Statement (to follow statement at) LOCATION LEVADJ CONTENT <OK>

<INS> TYPEIN TYPEIN ... <CD>
<INS> LEVADJ CONTENT LEVADJ CONTENT ... <CD>
<INS> TYPEIN LEVADJ CONTENT ... <CD>
<INS> LEVADJ CONTENT TYPEIN ... <CD>

Jump (to) <MARK> VIEWSPECS
Jump (to) Origin (of file) LOCATION VIEWSPECS
Jump (to) Next (from) LOCATION VIEWSPECS
Jump (to) Back (from) LOCATION VIEWSPECS
Jump (to) Successor (of) LOCATION VIEWSPECS
Jump (to) Predecessor (of) LOCATION VIEWSPECS

Set Viewspecs VIEWSPECS

Reset Viewspecs (to defaults) <OK>

Definitions:

LOCATION	Prompted by "M/A:" For M you may <MARK>.
TYPEIN	Type a series of characters, ending with <OK>. In the Insert Statement command, you may end with <INS> and proceed as shown for <INS> above.
LEVADJ	Type any number of level-adjustment characters (u for up, d for down), ending with <OK> or <SP>, or just type <OK> or <SP> for same level.
CONTENT	Prompted by "M/T/[A]:" For M you may <MARK>. For T you may type a series of characters, ending with <OK> (if another <OK> follows, a second one is not needed). In the Insert Statement command, you may type <INS> in place of the final <OK> and proceed as shown for <INS> above.
VIEWSPECS	Type any viewspec characters, ending with <OK>, or just <OK> for no view change.

VOCABULARY

back: In the context of the Jump command, the statement "back" from a statement you indicate is the statement immediately preceding it, regardless of level. Page 11

current statement: The last statement you inserted, edited, or jumped to in a file. Page 8

down: A term used to describe the relationships between statements at different levels: Statements at level 2 are "down" one level from statements at level 1, statements at level 3 are down one level from statements at level 2, and so on. Page 3

hierarchical structure: The structure of AUGMENT files; an outline form that shows the relationships between the statements. Page 1

<INS>: "INS" stands for "insert". Typing <INS> puts you in insert mode. Page 7

insert mode: You enter this mode when you type <INS>, either at "BASE C:" or in place of the final <OK> at the end of the Insert Statement command. In insert mode, you can continually add statements, each one following the last, until you type <CD>. Page 7

Insert Statement command: A Base subsystem command that lets you add statements to a file. Pages 4, 9

Jump command: An AUGMENT command to move from one point in a file to another, or from one file to another. The statement you jump to is displayed at the top of the file window. Page 10

level: A number that indicates how far down a statement is in the hierarchical structure of a file. The greater the number, the further down the statement is in the hierarchy. Pages 3, 5

level clipping: Using viewspecs to show statements at a limited number of levels. Page 15

line clipping: Using viewspecs to show a limited number of lines of each statement displayed. Page 17

mark: To mark means to indicate a character on the screen by pointing to it and then typing <OK>.

next: In the context of the Jump command, the "next" statement from a statement you indicate is the statement immediately following it, regardless of level. Page 11

predecessor: The predecessor of a statement is the preceding statement that is at the same level and has the same upstatement. Page 12

Reset Viewspecs command: A Base subsystem command that restores your viewspecs to what they were when you entered AUGMENT. Page 21

Set Viewspecs command: A Base subsystem command that lets you change how you view the statements in your file by specifying viewspecs. Page 13

statement number: A series of alternating numbers and letters that indicates the exact position of a statement within the hierarchical structure of a file. Page 14

structure: The arrangement of statements in a file. AUGMENT files have a hierarchical structure. Page 1

substatement: The substatements of a particular statement are all the statements that are subordinate to it and one level down from it. Page 3

substructure: A statement's "substructure" is all the statements subordinate to it, regardless of how many levels they are down from it. Page 3

successor: The successor of a statement is the next statement that is at the same level and has the same upstatement. Page 12

universal: A "universal" AUGMENT command is one that is available in every subsystem of AUGMENT (except subsystems that have been designed for special, limited applications). Page 11

up: A term used to describe the relationships between statements at different levels: Statements at level 1 are "up" one level from statements at level 2, statements at level 2 are up one level from statements at level 3, and so on. Page 3

upstatement: The upstatement of a statement is the statement it is subordinate to and one level down from. Page 3

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. Page 12

SOLUTIONS TO EXERCISES

1. The upstatement of the Section II statement is Part III. Since "Addendum" is a first-level statement, its upstatement is the origin statement.

2. Section II of Part III is "next" from Section I of Part III. The Part III statement is "back" from it.

3. The successor of the Part I statement is the Part II statement. Section I of Part I does not have a predecessor. If you tried to jump to its predecessor, Section I itself would appear at the top of the file window.

4. You would expect to see the statement after all the substructure of Part II and up one level, that is, right before the Part III statement and on the same level as Part III. The new statement would appear here because this is the next available place AUGMENT could put a second-level statement.

5. You could insert the new statement to follow "Contents" at the same level, to follow the Part III statement up one level, or to follow Section III of Part III up two levels. In fact, the statement would fall in the right place if you inserted it to follow the Part I or Part II statements up one level, or Section I or II of Part III up two levels; in all cases, the new statement would fall after the substructure of "Contents".

6. The statement number of "Contents" is 1; its upstatement is the origin statement, which always has the statement number 0. If you added a statement following "Contents" and one level down from it, its statement number would be 1A. To add a statement before "Contents", you would insert it to follow the origin statement; the new statement would then have the statement number 1, "Contents" would have the statement number 2, and the statement you added below "Contents" would have the statement number 2A.

7. To read all the text in a file, you must see all levels and all lines, as when you first enter AUGMENT. For an overview of as many statements as will fit on the screen at once, you would want to see all levels and one line.

8. To see only the first line of each first-level statement in your file, you could give the Set Viewspecs command and either combine viewspecs d, for "show first level only", and t, for "show first lines only", or use viewspec x for "show one line and one level only". To see two additional lines for each first-level statement, you would set viewspecs "rr". To add second-level statements to your view, you would set viewspec b. Your viewspec window would then show "2 3", meaning that you were seeing statements at two levels and three lines of each of these statements.

Price $3.00

WORKING WITH TABLES: A BEGINNING LESSON

Copyright Tymshare, Inc., May 1981
20705 Valley Green Drive
Cupertino, California 95014

This document was prepared and composed for printing with AUGMENT by the Office Automation Division of Tymshare, Inc.

AUGMENT Journal Number 75631
Tymshare Document Number 218(6/81)0.5M6156

This lesson describes how to use Table, an AUGMENT subsystem that helps you enter information into a file in rows and columns and easily perform editing or arithmetic operations on that information. You will learn the basic commands for creating and editing tables and for totaling rows and columns. Before reading this lesson, you should be familiar with the editing and viewing concepts and commands of AUGMENT up to the intermediate level.

CONTENTS

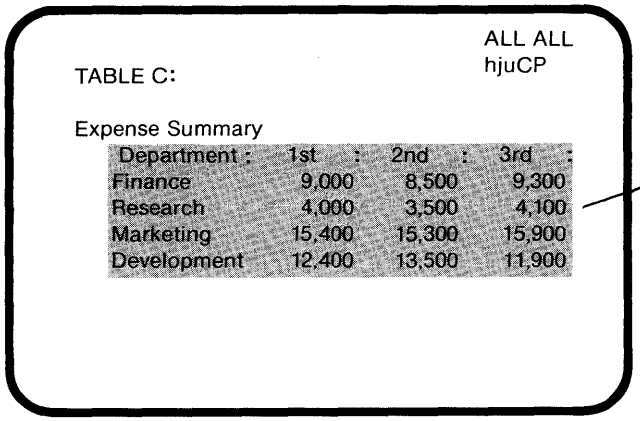
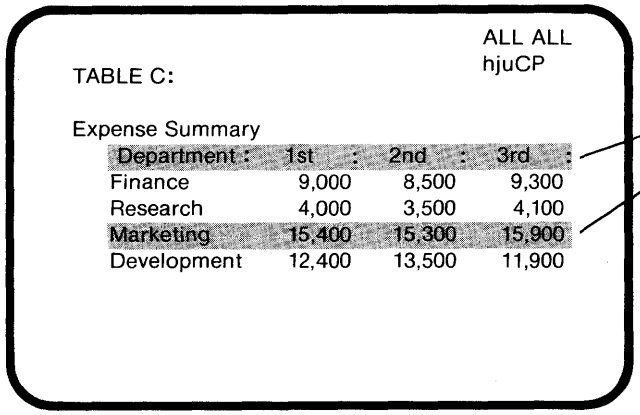
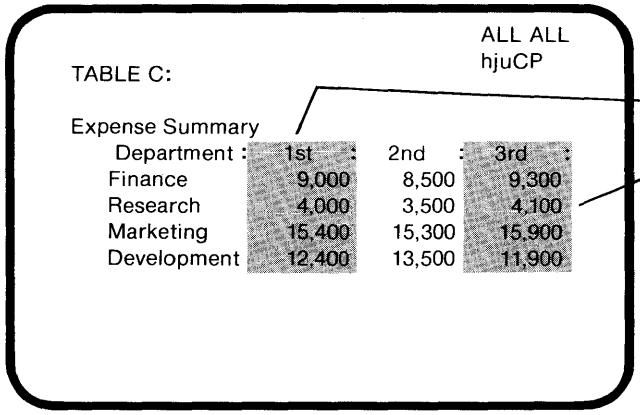


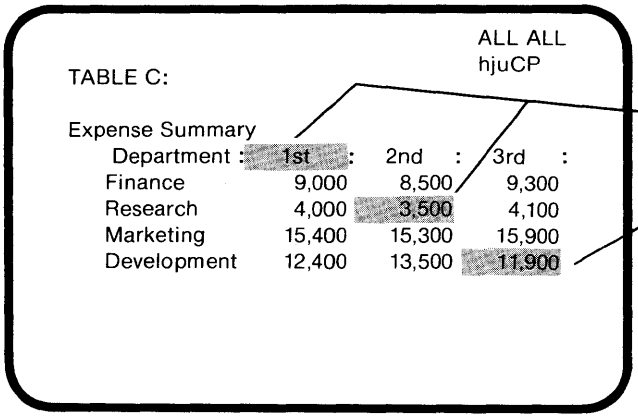
TABLE
A table is a plex of statements in which information is arranged into horizontal rows and vertical columns. To mark a table, mark any character in the table plex.



ROW
A row is a statement. Successive rows form the table plex. To mark a row, mark any character in the row. The top row of the table is called the "table header". Colons in the table header indicate where column boundaries will be when rows are added.



COLUMN
A column is a vertical section of a table plex. To mark a column, mark any character between column boundaries. Many commands that operate on columns do not affect entries in the table header.



ENTRY
An entry includes all characters between column boundaries, including spaces. To mark an entry, mark any character between column boundaries.

FIGURE 1: A TABLE IN AUGMENT

INTRODUCTION

Descriptive and illustrative tables are an important part of many technical, educational, and financial documents. AUGMENT offers a subsystem, called the Table subsystem, which is designed to make easy the task of creating and editing information in a table format. In Table, textual and numeric entries are quickly and automatically arranged into horizontal rows and vertical columns, eliminating the tedious process of arranging information in this way with Base subsystem commands. Table is an invaluable tool for creating clean–looking, well–formatted tables for printing or online viewing.

Rows and columns

In the Table subsystem, you can create a table consisting of any number of rows and columns. Each row of a table is a single statement and all rows are at the same level, thus forming a plex. Each column is a vertical section of the plex of rows. Figure 1 illustrates what Table recognizes as a table, row, column, and table entry, and summarizes other basic concepts you will learn as you read this lesson.

Table commands

With Table commands, you can arrange information into a table format without using tabs or typing a lot of spaces, and you can easily perform editing or arithmetic operations on the table entries. Just as you can use command nouns such as Word and Statement to refer to parts of a file when you are editing with Base subsystem commands, you can use the nouns Entry, Row, and Column to refer to parts of a table in Table commands.

Applications of Table

Textual applications for Table might include tables of personnel data or tables of comparison among product suppliers. Common numeric applications include tables of expenditure and projections of expenditure in project management. Commands that total rows or columns make it easy to analyze changes to financial data.

This lesson discusses the Table commands you can use to create a table and perform editing and totaling operations on the table contents. The Table subsystem also includes all the "universal" commands, such as Jump and Help, as well as the Update and Delete Modifications commands, which work as they do in the Base subsystem.

ENTERING AND LEAVING THE TABLE SUBSYSTEM

Goto

You enter Table by using the Goto command. For example, you would go to Table from the Base subsystem as follows:

You type:	Command window shows:
g	BASE Goto (subsystem) C/OPT:
<OPT>	BASE Goto (subsystem) (subsystem name) M/T/[A]:
table<OK>	BASE Goto (subsystem) (subsystem name) table!
	TABLE C:

Programs buffer full

When Table is ready for a command, you will see "TABLE C:" in the command window. If AUGMENT cannot give you access to Table because of insufficient room in the working space allocated to you, the message "Programs Buffer Full" will appear in your status window. In this case, you must do the following to obtain space for Table:

You type:	Command window shows:
e	BASE Execute (command in) C/OPT:
p	BASE Execute (command in) Programs
	PROGR C:
d	PROGR Delete C:
a	PROGR Delete All (programs in buffer) OK:
<OK>	PROGR Delete All (programs in buffer) !
	BASE C:

You can then give the Goto command again to enter Table.

Returning to Base: Quit

To reenter the Base subsystem after working in Table, use the Quit command.

You type:	Command window shows:
q	TABLE Quit OK:
<OK>	TABLE Quit !
	BASE C:

Returning to Table

To return to Table from Base, simply type "g" for "Goto" and "t" for the command word "Table". End with <OK>. Typing <OPT> followed by "table<OK>" is necessary only the first time you enter Table during a work session.

Execute

You can also use the Execute command in either the Base or the Table subsystem to give a single command in the other subsystem. For example, you could give the Base Delete Branch command from Table as follows:

You type:	Command window shows:
e	TABLE Execute (command in) C/OPT:
b	TABLE Execute (command in) Base
	BASE C:
d	BASE Delete OPT/C:
b	BASE Delete Branch (at) M/A:
<MARK>	BASE Delete Branch (at) ! (really?) OK:
<OK>	BASE Delete Branch (at) ! (really?) !
	TABLE C:

After the command is executed, you are automatically returned to the subsystem from which you gave the Execute command.

Check your command window.

It is common to move frequently between the Table and Base subsystems when working with a file that contains tables. It is a good practice to check your command window when you give a command to make sure you are in the Base subsystem when you give a Base command and in the Table subsystem when you give a Table command. Note that although Table and Base have some command words in common, you do not necessarily type the same character or characters to get the same command word.

LINES IN TABLES

So that you will know what kinds of tables you can work with in the Table subsystem, there are a few important things you should know about lines in tables.

Overflow lines

You may want to avoid making a table row so long that it is automatically continued onto a second line. As usual when a statement is too long to fit on one line, AUGMENT will break between words and continue with the rest of the row (the "overflow") on a new line. This will affect the appearance of the table in an undesirable way, by showing the overflow text beneath the first column in every row. Note that since your printing device may allow longer lines than your display screen, you may be able to print the table satisfactorily. The overflow lines do not interfere with your use of Table commands; they only affect your view of the table while you are working with it.

Displaying longer lines

To avoid overflow, you can increase the amount of horizontal space available for displaying a table by setting viewspec B to suppress level indenting. The standard AUGMENT display terminal also offers a "wide screen" option that increases the display width from 80 to 160 characters, which is useful for viewing tables that are otherwise too wide to fit on your display screen. To learn how to use the wide screen option, give the Jump (to) Locator command and look under the heading ONLINE DOCUMENTATION FOR AUGMENT.

Return characters

A return character in a table row, like a return character in any statement, will cause AUGMENT to start a new line. In general, however, you should plan the body of your table to consist of single–line statements only, since Table offers limited handling of information in lines beyond the first. When return characters are present, many Table commands will work only on the first line of the row. Later in this lesson, some ways of setting up multiple–line entries are suggested.

STARTING A TABLE

There are three steps in starting a table:

1. "Size" the table by determining the number and widths of columns you want it to have.

2. In the Base subsystem, insert a statement under which the table will go.

3. Go to the Table subsystem and begin building the table with the Insert Header command.

Sizing a table

Sizing the table depends on how it will be used, and involves deciding how many columns it will contain and how wide each column will be. In general, consider the width of the device on which you are editing the table and, if you are going to print the table, the width of the printing device you will use. Be sure to allow for one character position between each pair of columns.

When determining the widths of columns containing numeric data, remember that decimal points, commas, and minus signs take extra characters. Using a format with no commas, or showing large numbers in large units such as thousands, may help if the table becomes too wide. Note that when an arithmetic operation is performed on a numeric entry, AUGMENT imposes a restriction of twelve digits on the result. When sizing a table that will contain textual entries, plan carefully the amount of space you will need; abbreviations are useful at times.

The statement above the table

Once you have sized your table, you can use the Insert Statement command in Base to insert the statement under which the table will go. This statement can be the title of the table or a paragraph that introduces or describes it. For example, you could insert a statement such as the following to serve as a title for a table:

```
Expense Summary
```

Insert Header

After inserting such a statement, you can go to the Table subsystem and begin building the table with the Insert Header command. This command inserts the first statement in the table plex, called the "table header". This statement is essential in structuring the table because it tells Table the number and widths of columns. When giving the Insert Header command, you must indicate the following:

The statement below which you want the table to be inserted. The table will be a plex one level down from this statement.

The number of columns in the table.

The width of each column. Type the number of characters wide you want each column to be, separating the numbers with semicolons.

For example:

You type:	Command window shows:
i	TABLE Insert C:
h	TABLE Insert Header (below) M/A:
<MARK>	TABLE Insert Header (below) ! (with number of columns) M/T/[A]:
4<OK>	TABLE Insert Header (below) ! (with number of columns) 4! (of widths) M/T/[A]:
12;8;8;8<OK>	TABLE Insert Header (below) ! (with number of columns) 4! (of widths) 12;8;8;8!
	TABLE C:

This command tells Table to insert below the marked statement a table header for a table with four columns. The first column will be twelve characters wide, and the remaining three will each be eight characters wide. You can instead type just "12;8<OK>", and Table will assume the last width typed, 8, for the remaining columns.

If the marked statement were the title shown previously, Table would respond to the above Insert Header command by inserting the table header a level below the title, as follows:

```
Expense Summary
                :        :        :        :
```

Column delimiters and boundaries

In this table header, there are twelve spaces before the first colon and eight spaces before each of the remaining colons; the statement ends at the last colon. The colons in the table header are called "column delimiters". When rows are added to the table, spaces will appear in each row where column delimiters appear in the table header. These column delimiters and spaces serve to separate one column from the next, and thus form the "column boundaries" for the table.

Blank entries

The table header itself is a row, though one that Table sometimes treats differently because of its special function. The blank areas between column delimiters in the table header are not empty; they are filled with spaces and should be thought of as blank entries. There are no empty entries in a table, even though they may appear blank. In the table header, spaces (or any other characters that are not column delimiters) indicate the positions in which the table entries will appear in each row. Note that a table header should not contain <TAB> characters.

Other column delimiters

Table also recognizes the characters vertical bar (|) and exclamation point (!) as column delimiters. Furthermore, the right boundary of the last column may be indicated by a return character or simply by the end of the header.

NOTE: On some terminals and printers, vertical bar is broken in the middle (¦).

ENTERING COLUMN HEADINGS

Replace Row

Once the table header has been inserted, you can replace its blank entries with textual or numeric column headings by using the Replace Row command. When using Replace Row to enter column headings, you mark the table header and type the column headings separated by semicolons. These headings are the entries that will replace the blank entries created by the Insert Header command. The Replace Row command also asks for the "justification", or alignment of the entries relative to the column boundaries. There are many types of justification, as described later in this lesson. The example below illustrates Center justification, which centers entries between column boundaries. Continuing from the previous example:

You type:	Command window shows:
rr	TABLE Replace Row (at) M/A:
<MARK>	TABLE Replace Row (at) ! (by) M/T/[A]:
Department;1st;	TABLE Replace Row (at) ! (by) Department;1st;
2nd;3rd<OK>	2nd;3rd! (justified) OK/C:
c<OK>	TABLE Replace Row (at) ! (by) Department;1st;
	2nd;3rd! (justified) Center !
	TABLE C:

Notice that in this example and the remaining examples in this lesson, some steps of typing the command are combined for brevity. After the command shown above, the table would look like this:

```
Expense Summary
    Department :  1st   :  2nd   :  3rd   :
```

Later, after rows of information are entered into the table, Replace Row can be used to replace any row.

ENTERING ROWS OF INFORMATION

Insert Row

After entering the column headings in the table header, you are ready to enter information into the body of the table. To do this a row at a time, you can use the Insert Row command. In this command, you must indicate the following:

The statement (table header or other row) that you want the new row to follow.

The justification of the entries in the row. You may type "l" for Left justification, "r" for Right justification, or "c" for Center justification. Typing <OK> will give you a "default" justification, normally Left. Later in this lesson, you will learn about other types of justification and how to change this default justification.

Semicolon format

The entries in the row. You can type them, separated by semicolons or <TAB> characters (called "semicolon format" for short), or you can mark an existing row, from which the entries will be copied.

The following example shows how you could enter the first row into the Expense Summary table.

You type:	Command window shows:
ir	TABLE Insert Row (after) M/A:
<MARK>	TABLE Insert Row (after) ! (with) M/T/[A]:
Finance;9,000;	TABLE Insert Row (after) ! (with) Finance;9,000;
8,500;9,300<OK>	8,500;9,300! (justified) OK/C:
r<OK>	TABLE Insert Row (after) ! (with) Finance;9,000;
	8,500;9,300! (justified) Right !
	TABLE C:

You would mark the table header. The table with its first row inserted would look like this:

```
Expense Summary
    Department :  1st    :  2nd    :  3rd    :
        Finance    9,000     8,500     9,300
```

Leaving entries blank

If you supply fewer entries than the number of columns, Table enters spaces into the remaining columns. Often it is useful to have a blank entry to leave space for results of Table arithmetic operations. You can leave middle columns blank by typing successive semicolons or <TAB> characters. For example, to leave the third entry blank in the example above, you would type "Finance;9,000;;9,300". If you supply more entries than the number of columns, the additional ones are ignored.

Truncation

If you type an entry that is longer than the width of the column in which it must fit, Table truncates the entry by cutting off the textual or numeric information that will not fit. When this happens, a message informing you of the truncation is displayed in your status window. Whether truncation occurs on the left or right depends on the type of justification, as explained later in this lesson.

Copying from existing rows

With Insert Row you can also add rows to a table by marking an existing row, either in the table where you are working or in a different table. From the table header for the row you mark, Table will determine what the individual entries are and then insert them as though you had typed them in semicolon format.

Repeat mode

You can give the Insert Row command repeatedly by typing <RC> in place of the final <OK>. This puts you in repeat mode and automatically starts the next Insert Row command for you. To leave repeat mode, type <CD>.

Create Row

Another Table command that lets you enter a row is Create Row. With Create Row, you can mark a statement that contains the entries in semicolon format, and those entries will be inserted into the table as a new row. You would use the Base Insert Statement command to insert the statement containing the entries and then, in Table, give the Create Row command. For example, you could mark this as the statement containing the entries:

Finance;9,000;8,500;9,300

Create Group (of rows)

Thus, you can first use Base subsystem commands to create and edit statements containing rows of entries in semicolon format, and then use Create Row in Table to enter each one into the table. To simplify this procedure, Table offers the Create Group (of rows) command, in which you can mark a plex of statements containing table entries in semicolon format. The entries in each statement of the plex are entered as a row in the table, as though with successive Create Row commands.

Using Create Group (of rows) is the fastest and easiest way to build a large table. The steps for using this command are:

1. From Table, go to Base and use the Insert Statement command to create a plex of statements, all at the same level, containing the entries in semicolon format. A convenient way of isolating this from other information in your file is to create a branch consisting of any statement, such as "entries", with the plex inserted a level below it. Remember that typing <INS> to enter insert mode lets you continually add statements, each one following the last, until you type <CD>.

2. Go to Table and use Create Group (of rows) to enter the new rows into the table. You will be asked to mark the statement the new rows are to follow and then the plex containing the entries. Finally, you will be asked for the justification.

3. Give a Delete command in the Base subsystem to delete the structure containing the entries in semicolon format.

Example of Create Group (of rows)

For example, to add three rows to the Expense Summary table, you could first use the Insert Statement command in Base to create the following:

```
entries
    Research;4,000;3,500;4,100
    Marketing;15,400;15,300;15,900
    Development;12,400;13,500
```

Note that the last row contains only three entries. This will leave a blank entry in the fourth column, to be replaced later. To insert the three rows into the Expense Summary table, you would go to Table and use the Create Group (of rows) command.

You type:	Command window shows:
<SP>crg	TABLE Create Group (of rows after) M/A:
<MARK>	TABLE Create Group (of rows after) ! (from plex at) M/A:
<MARK>	TABLE Create Group (of rows after) ! (from plex at) ! (justified) OK/C:
r<OK>	TABLE Create Group (of rows after) ! (from plex at) ! (justified) Right !
	TABLE C:

You would first mark the previously inserted table row and then mark the plex below the "entries" statement. The new rows would be entered, and the table would look like this:

```
Expense Summary
    Department :  1st    :  2nd    :  3rd    :
        Finance    9,000     8,500     9,300
       Research    4,000     3,500     4,100
      Marketing   15,400    15,300    15,900
    Development   12,400    13,500
```

You could then use Execute (command in) Base and give a Delete Branch command to remove the "entries" branch.

Inserting substatements

Other statements, such as comments or even subtables, can be entered into the table plex, but these must be entered one level down from the row they follow so they will not be considered part of the table contents. If you add statements to the table at the same level as the rows, Table will consider them to be part of the table regardless of their content.

ENTERING COLUMNS OF INFORMATION

You may have information that is more easily entered into a table by columns instead of by rows. The combined use of the commands Insert Blank Group (of rows) and Replace Column allows you to enter information a column at a time. Insert Blank Group (of rows) inserts a group of blank rows; the blank entries in these rows are then replaced with Replace Column.

Why a group of blank rows?

A group of blank rows must be present for you to enter columns of information because of the basic difference between columns and rows in Table: A row is a statement, but a column is not, so a column cannot be entered in a table the way a row can. The length of a column is determined by the number of rows already present in a table, so before columns of information can be entered into a table, an appropriate number of rows must already exist.

Insert Blank Group (of rows)

One way to create an appropriate number of rows is with the Insert Blank Group (of rows) command. This command inserts rows of blank entries after the row that you indicate. For example, suppose you wanted to enter information into the Expense Summary table column by column instead of row by row. After inserting the table header and entering the column headings, you could proceed as follows, marking the table header:

You type:	Command window shows:
ibg	TABLE Insert Blank Group (of rows after) M/A:
<MARK>	TABLE Insert Blank Group (of rows after) ! (number of rows) M/T/[A]:
4<OK>	TABLE Insert Blank Group (of rows after) ! (number of rows) 4!
	TABLE C:

Replace Column

You could then replace the blank entries with textual or numeric information by using the Replace Column command. With Replace Column, you can replace all the entries in a column with entries you type in semicolon format or with entries in another column that you mark. The following example shows how you could enter the first column of the Expense Summary table.

You type:	Command window shows:
rc	TABLE Replace Column (at) M/A:
<MARK>	TABLE Replace Column (at) ! (by) M/T/[A]:
Finance;Research; Marketing; Development<OK>	TABLE Replace Column (at) ! (by) Finance;Research; Marketing;Development! (justified) OK/C:
r<OK>	TABLE Replace Column (at) ! (by) Finance;Research; Marketing;Development! (justified) Right !
	TABLE C:

MARKING IN TABLE

Before learning any more about specific Table commands, you should review the following information about marking entries, rows, columns, and entire tables in the Table subsystem.

Marking an entry

A table entry consists of all the characters between column boundaries, including spaces. To mark an entry, you can mark any of these characters. You can mark entries in the table header or in a subsequent row. If a row contains return characters, you can mark entries in any part of the row, not just in the first line.

Marking a row

You can mark a row by marking any character in the row. Remember that if the statement contains a return character, many Table commands will work only on the entries in the first line of the row, even if you mark a character in a subsequent line.

Marking a column

To mark a column, mark any character in the column between the column boundaries in either the table header or in a subsequent row. Caution is required, however, if any rows in the table contain return characters, because unpredictable results will occur if you mark a character in any line beyond the first.

NOTE: Except where indicated otherwise, commands that operate on rows affect only the first line of the row, and commands that operate on columns do not affect entries in the table header. Table commands that operate on columns never affect entries in lines beyond the first.

If you mark a column boundary, Table will proceed as though you marked the character to the left of the boundary.

You can mark in any table.

The entry, row, or column you mark may be in any table; a single Table command may refer to parts of different tables, such as a column in one and an entry in another.

Marking a table

To mark a table, you can simply mark any character in the table plex.

Other choices

After an "M/A:" prompt, you can type an address instead of marking. This alternative is most useful when you are indicating a row or a table, that is, when you do not have to refer to a specific character position. Similarly, when you can type something after an "M/T/[A]:" prompt, you have the alternative of marking. Usually you can mark text consisting of what you would otherwise type; sometimes, Table may expect you to mark a number or a word. To find out exactly what your choices are after any prompt, type a question mark (?).

ALIGNING TABLE ENTRIES: JUSTIFICATION

In Table commands, you can select the "justification" of entries, that is, their alignment relative to the column boundaries defined in the table header. When you see "(justified) OK/C:" in your command window, you can select Left, Right, Center, or any of the other available justification types, many of which are described later.

Text default justification

If you type <OK> in response to "(justified) OK/C:", the current "text default justification" is used. This is initially Left, but you can change it to any other type of justification with the Set Text (default justification) command.

Numeric default justification

Later you will learn about Table commands that perform arithmetic operations on table entries and do not ask you for the justification. These commands use the current "numeric default justification", which is initially Right and can be changed with the Set Numeric (default justification) command.

Justify

You can change the justification of any entry, row, column, or table with the Justify command. For example, Right justification was used when the Expense Summary table was built because it was appropriate for most of the entries, but suppose you wanted to change the entries in the first column to Left justification.

You type:	Command window shows:
<SP>jc	TABLE Justify Column (at) M/A:
<MARK>	TABLE Justify Column (at) ! (justified) OK/C:
<OK>	TABLE Justify Column (at) ! (justified) !
	TABLE C:

You could mark any entry in the first column. Since the text default justification is Left, typing "<OK>" after "(justified) OK/C:" suffices. If, however, you had previously changed the default justification, you would have to type "l" to specify Left justification. In either case, the table would now look like this:

```
Expense Summary
    Department :  1st    :  2nd    :  3rd    :
   Finance          9,000     8,500     9,300
   Research         4,000     3,500     4,100
   Marketing       15,400    15,300    15,900
   Development     12,400    13,500
```

Notice that Justify Column did not affect the entry in the table header. The Justify Table command also excludes entries in the table header.

Justification types

The following paragraphs describe several common types of justification. Except where indicated otherwise, Table discards any spaces at the beginning or end of an entry before justifying it.

Left

Left justification aligns the entry with the left column boundary, adding spaces on the right if the entry is shorter than the width of the column. If the entry is longer than the column width, it is truncated on the right.

Right

Right justification aligns the entry with the right column boundary, adding spaces on the left if necessary. If the entry is longer than the width of the column, it is truncated on the left.

Center

Center justification centers the entry between the left and right column boundaries, adding spaces before and/or after the entry if necessary. If the entry is longer than the width of the column, it is truncated on the right. If an entry cannot be exactly centered, the extra space is added on the right.

Same

Same justification aligns the entry with the left column boundary, as does Left justification, but without discarding spaces at the beginning of the entry. This type of justification is especially useful when you mark existing entries and want to keep the same number of leading spaces, or when you are typing something and you want it indented exactly the number of spaces you type.

Numeric entry

The effect of the following types of justification depends on whether the entry is numeric. A numeric entry is one that contains digits (0 through 9) and may also contain, in the appropriate places, a plus sign (+), a minus sign (−), a dollar sign ($), a decimal point (.), and commas (,). For example, −16,120.50 is a numeric entry, but 150−35 and 14E5 are not.

Mixed

Mixed justification uses the text or numeric default justification, depending on the entry: If the entry is numeric, the numeric default justification is used; otherwise, the text default justification is used. For example, this justification would have been convenient to use when the Expense Summary table was built.

Integer

Integer justification rounds the entry to an integer and right justifies the result. If the entry contains commas, the commas are removed. An entry that contains no integer digits is replaced entirely by spaces, and an entry that is not numeric is simply right justified.

Comma

Comma justification adds commas to the integer part of the entry, one after every three digits starting from the rightmost digit of the integer part, and right justifies the result. For example, the entry

1536.82 becomes 1,536.82. If the entry is not numeric, it is right justified.

Icomma

Icomma justification rounds the entry to an integer (as does Integer justification), adds commas (as does Comma justification), and right justifies the result. For example, the entry 1536.82 becomes 1,537 right justified.

Places, One, and Two

Places justification rounds or expands the entry to a number of decimal places that you specify and right justifies the result. If the entry contains commas, the commas are removed. One and Two are shorter ways of specifying one and two decimal places. For example, One justification changes the entries 7.899 and 5 to 7.9 and 5.0, right justified. If the entry is not numeric, it is right justified.

Point

Point justification aligns the decimal point in the entry with the decimal point in the first entry in the column (in the row that immediately follows the table header), rounding the decimal part if necessary. If there is no decimal point in the entry determining the alignment, Right justification occurs instead. If there is no decimal point in the entry being justified, it is aligned as though it ended in a decimal point.

EDITING INDIVIDUAL ENTRIES

With Table commands, you can change individual table entries by replacing one entry by another, replacing a character within an entry, clearing an entry, or transposing two entries. Remember that you fill in a blank entry by replacing it by another entry, and you mark an entry by marking any character between column boundaries in the table header or in a subsequent row.

Replacing an entry

To replace one entry by another, you can use the Replace command verb with Entry, Wentry, or Item. You must always indicate the entry to be replaced, the entry to replace it by, and the justification.

Replace Entry

The Replace Entry command lets you replace an entry you mark by another entry that you type or mark. For example, you could fill in the blank entry in the Expense Summary table as follows:

You type:	Command window shows:
re	TABLE Replace Entry (at) M/A:
<MARK>	TABLE Replace Entry (at) ! (by) M/T/[A]:
11,900<OK>	TABLE Replace Entry (at) ! (by) 11,900! (justified) OK/C:
r<OK>	TABLE Replace Entry (at) ! (by) 11,900! (justified) Right !
	TABLE C:

The marked character could be any of the spaces in the blank entry; the entry would be replaced by 11,900, right justified.

Replace Wentry

So that you can easily replace entries by information that is not in a table, the Replace Wentry command allows you to replace a table entry by any word you indicate. ("Wentry" is short for "word entry".) The word may be, but not need be, a table entry.

Replace Item

In certain applications the Replace Item command may be useful. In this command, you type the "coordinates" of the entry to be replaced instead of marking it; then you type or mark the replacement entry. You indicate coordinates as follows:

row,column

row,column,line

The first value is the row number and the second is the column number. Row number 0 refers to the table header; column number 0 is not allowed. The optional third value is the line number. If no line number is given, line 1 is assumed. For example, to specify that the entry to be replaced is in the first line of the second row after the table header, and in the first column, you would type "2,1", ending as usual with <OK>.

Before typing the coordinates, you must tell Table which table they refer to. When you see "(at table) M/A:" in the command window, reply by indicating any character in the table plex. For example, your reply could be to type the statement number or SID of any row in the table.

Replace Character

Besides replacing entire table entries, you can replace a character within an entry by any text with the Replace Character command in Table. You first mark the character to be replaced and then type or mark the replacement text. A character can be replaced either by a single character or by any text that fits within the column boundaries. When a character is replaced by more than one other character, spaces are removed from within the column boundaries to accommodate the replacement, and the replaced entry is justified according to the text default justification. The alignment of all other entries in the row is maintained. Note that Replace Character in Table is not restricted to use within an entry; it can be used to replace any character in the table by any other character.

Clear Entry

As a simple way of replacing a table entry by an entry consisting entirely of spaces, Table also offers the Clear Entry command.

Transpose Entry

Finally, when entries in a table need rearranging, you can use the Transpose Entry command. This command transposes any two table entries that you mark. If the widths of the entries are the same, the content and justification of the entries are not changed; however, if the widths are different, the transposition may change the justification or cause truncation to occur.

EDITING ROWS AND COLUMNS

Just as editing commands in Table can be used on individual entries, they can also be used on entire rows, columns, or tables. You have already learned about Replace Row and Replace Column, which replace all the entries in a row or column by entries you type in semicolon format or by a row or column you mark. Other editing commands that affect entire rows or columns are discussed in this section.

Clear Row, Column, or Table

The Clear Row and Clear Column commands replace the entries in a specified row or column by entries consisting of spaces. Likewise, the Clear Table command replaces the entries in all rows and columns of the table by spaces (except for the entries in the table header).

Delete Row or Column

The Delete Row and Delete Column commands remove the indicated row or column. Delete Row is like Delete Statement in Base; it deletes the entire row from the table plex. Delete Column deletes the entire column, including the entry in the table header, as well as the right column boundary.

Move Row or Column

The Move Row command moves an entire row (including any extra lines) to follow another row that you mark. The Move Column command moves an entire column, along with the entry in the table header and the right column boundary, to follow another column that you mark.

Copy Column

The Copy Column command copies a column to follow another one that you mark, creating a new column containing the same width, content, and justification as the copied column.

Transpose Column

The Transpose Column command transposes two columns that you mark, including the entries in the table header. The content and justification of the entries in the columns are unchanged.

Replace And (change) Row or Column

The Replace And (change) Row or Column command changes a row into a column or a column into a row. It is similar to Replace Row or Column except that it gets the replacement entries from a column you mark if you are changing a row, or from a row if you are changing a column.

Replace Table

You can also use the Replace command verb to replace an entire table. The Replace Table command replaces all the entries in a table by the entries in another table that you indicate (excluding the entries in the table headers). If you want every column of a table to have the same entries, you can use Replace Table and type the entries in semicolon format.

Rows or columns of different lengths

In the Replace Table command and the other editing commands that may operate on rows or columns in different tables, the rows or columns need not have the same length. Extra entries are simply ignored, and blank entries are inserted to fill out longer rows or columns. For example, if you copy a column from a table with four rows to follow a column in a table with six rows, the last two entries in the new column will be blank.

ADDING NEW COLUMNS

You may want to edit a table by adding new information to it. Inserting new rows of information is straightforward; you can use any of the commands for building a table a row at a time, such as Insert Row or Create Group (of rows). Adding a new column, however, is a two-step process, much like the process of initially entering information into a table a column at a time. First you use Insert Blank Column to make space for the new entries, and then you use Replace Column to replace the entries in the blank column.

Insert Blank Column

The Insert Blank Column command inserts a column of blank entries to follow the column you mark. For example, you could add a blank column having a width of eight characters to the Expense Summary table as follows:

You type:	Command window shows:
ibc	TABLE Insert Blank Column (to follow) OPT/M/[A]:
<MARK>	TABLE Insert Blank Column (to follow) ! (with width) M/T/[A]:
8<OK>	TABLE Insert Blank Column (to follow) ! (with width) 8!
	TABLE C:

Replacing the blank entries

If you marked any character in the last column, or even its right boundary, the blank column would become the new last column. You could then use Replace Column to replace the blank entries in that column. Remember that Replace Column does not affect entries in the table header, so you would have to use Replace Entry to replace the blank entry in the table header. If you replaced the column with the entries "9,500;4,250;16,000;13,100", right justified, and replaced the entry in the table header with the centered heading "4th", the table would look like this:

```
Expense Summary
    Department :  1st    :  2nd    :  3rd    :  4th    :
   Finance          9,000     8,500     9,300     9,500
   Research         4,000     3,500     4,100     4,250
   Marketing       15,400    15,300    15,900    16,000
   Development     12,400    13,500    11,900    13,100
```

You can, of course, replace only individual entries in the blank column, or leave all of them blank if you wish. The important thing to remember is that, in any case, adding a new column means first inserting a blank column.

Inserting a column in front of a table

By typing <OPT> after "(to follow) OPT/M/[A]:" in the Insert Blank Column command, you can insert the blank column before the first column in the table. When you type <OPT>, this will appear in your command window:

(in front of first column in table at) M/A:

Reply by marking any character in the table plex and, when prompted, typing the width of the column.

CHANGING COLUMN BOUNDARIES

After setting up a table, you may find that you need to widen or narrow a column; the Insert Field and Delete Field commands will help you do this. You may also want to change the characters that form the column boundary. Remember that the column boundary is initially made up of the column delimiter in the table header and the space under it in each successive row of the table. To change this boundary so that it consists of a single character that you specify, you can use the Replace Boundary command. Refer to Figure 2 for illustrations of the parts of a table that are discussed in this section.

What a field is

A "field" is a series of characters from one character position through another in every row of a table. By inserting a field of spaces, you can increase the width of a column. By deleting a field, you can decrease the width of a column, remove information from certain positions in every entry in a column, or even remove several columns. A field may extend from any character position in a row through any other, regardless of column boundaries or table contents.

Insert Field

You can make a column wider with the Insert Field command. This command inserts a field of spaces after the character position you indicate, adding as many spaces in each row as the number you supply for the field width. You can indicate the position by marking it, either in the table header or in a subsequent row. The following example shows how you could widen the first column of the Expense Summary table by one character position.

You type:	Command window shows:
if	TABLE Insert Field (of spaces after) M/A:
<MARK>	TABLE Insert Field (of spaces after) ! (of width) M/T/[A]:
1<OK>	TABLE Insert Field (of spaces after) ! (of width) 1!
	TABLE C:

It would be best in this case to mark the space in the last character position of any entry in the first column. Marking the first space after any of the shorter entries in that column would not work, because the inserted field would then split the longer entries. Note that to widen any of the remaining columns, you could add a field of spaces at the beginning of the column by marking the column's left boundary.

Delete Field

Columns can be made narrower with the Delete Field command. This command deletes a field of characters between and including any two positions that you indicate, regardless of column boundaries. Thus, an individual column can be made narrower, by the removal of spaces or other text within every row, or entire columns can be deleted. To decrease the width of a column by one character, you can mark the same position twice. In general, however, the two positions you mark do not have to be in the same row.

NOTE: The Insert Field and Delete Field commands take into account which line the positions you mark are in; if you mark positions in the first line of a row, only first lines will be affected.

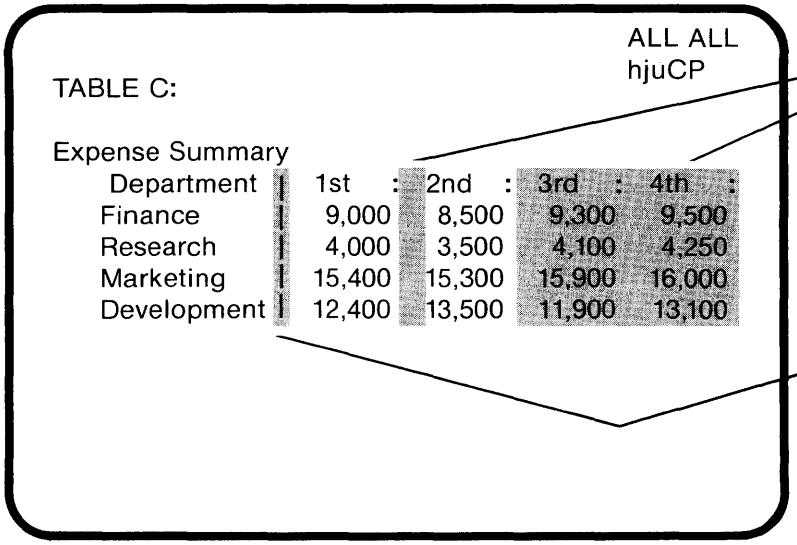


FIGURE 2: FIELD AND BOUNDARY

Replace Boundary

With the Replace Boundary command, you can replace the characters comprising a column boundary with a character that you specify. You can replace either the entire boundary, including the column delimiter in the table header, or only the spaces under it in successive rows, omitting the column delimiter. If you include the column delimiter, remember that only colon (:), vertical bar (|), and exclamation point (!) are valid delimiters.

Including the table header

For example, you could replace the entire right boundary of the first column of the Expense Summary table with vertical bars as follows:

You type:	Command window shows:
rb	TABLE Replace Boundary (at) M/A:
<MARK>	TABLE Replace Boundary (at) ! (with the character) M/T/[A]:
| <OK>	TABLE Replace Boundary (at) ! (with the character) | ! OK/C:
i<OK>	TABLE Replace Boundary (at) ! (with the character) | ! Include (header row) !
	TABLE C:

You could mark either the column delimiter in the table header or the space under it in any subsequent row. The following shows the effect of this command on the appearance of the table, as well as the effect of the Insert Field example given previously.

```
Expense Summary
    Department  |  1st    :  2nd    :  3rd    :  4th    :
   Finance      |   9,000     8,500     9,300     9,500
   Research     |   4,000     3,500     4,100     4,250
   Marketing    |  15,400    15,300    15,900    16,000
   Development  |  12,400    13,500    11,900    13,100
```

Omitting the table header

Typing "<OK>" in response to the "OK/C:" prompt at the end of the Replace Boundary command would replace the boundary in all rows except the table header.

You should be sure, when using any of the commands for changing column boundaries, that you mark the exact character position you intend, or the command will not work as expected.

TOTALING ROWS AND COLUMNS

Table subsystem commands can be used to add up numeric entries in rows and columns. You can have the total for a row or column entered in your table, inserted anywhere as text, or just displayed in your status window. When you want to include totals in your table, you of course need to have space for them. You can make space with the Insert Blank Row or Insert Blank Column command. To add up the entries, you can use the Total Row or Total Column command, or one of the other Total commands described below. Whenever Table enters the result of a Total command in your table, it uses the numeric default justification, which is initially Right and can be changed with the Set Numeric (default justification) command.

Insert Blank Row

The Insert Blank Row command inserts a row of blank entries after a row that you indicate. For example, you could use Insert Blank Row on the Expense Summary table, marking the last row, to make space for column totals at the bottom of the table.

You type:	Command window shows:
ibr	TABLE Insert Blank Row (after) M/A:
<MARK><OK>	TABLE Insert Blank Row (after) ! !
	TABLE C:

Similarly, the Insert Blank Column command could be used to provide space for row totals.

Total Row or Column

To add up the numeric entries in a specified row or column, you can give the Total Row or Total Column command. The following example shows how you could use this command to total any column in the Expense summary table and enter the result in the blank row you inserted at the bottom of the table.

You type:	Command window shows:
tc	TABLE Total Column (of numbers at) M/A:
<MARK>	TABLE Total Column (of numbers at) ! (enter result as) C:
e	TABLE Total Column (of numbers at) ! (enter result as) Entry (at) M/A:
<MARK><OK>	TABLE Total Column (of numbers at) ! (enter result as) Entry (at) ! ! TABLE C:

Total as an entry

You would mark the column to be totaled and, as the place to enter the result, the blank entry at the bottom of that column. In general, the entry you mark can be any entry in the table; if it already contains a number and is in the row or column you are totaling, the number will not be included in the result. This means that when you are retotaling, you can mark the old total and it will not be added in.

Messages in the status window

The total, the number of entries added, and their average are displayed in your status window. If the row or column contains no numeric entries, Table does nothing more than display a message to this effect.

Total as a visible or only in the status window

Instead of typing "e" for "Entry" after "(enter result as) C:", you can type "v" for "Visible" to indicate any visible that you want the total to follow, or "s" for "Status (window)" to have the total displayed in your status window only.

Total All Rows or Columns

To total all rows or columns and put the results in the table, you can use the single command Total All Rows or Total All Columns. In the Total All Rows command, you indicate the table containing the rows to be totaled and the column where the results will go. For each row, Table adds up the numeric entries and puts the result in that row in the specified column, as though you were giving a Total Row command and replacing that entry by the result. As when you use Total Row, any entries in the column you indicate are not included in the total. Similarly, the Total All Columns command adds up the numeric entries in each column and puts the results in a specified row, excluding the entries in that row from the total. For example, a single Total All Columns command could be used to total the last four columns of the Expense Summary table; the first column would not be totaled because it contains no numeric entries.

You type:	Command window shows:
tac	TABLE Total All Columns (in table at) M/A:
<MARK>	TABLE Total All Columns (in table at) ! (put results in row at) M/A:
<MARK><OK>	TABLE Total All Columns (in table at) ! (put results in row at) ! ! TABLE C:

If you marked the Expense Summary table and then its last row, the totals would be placed in that row, using the numeric default justification. To justify the totals so they would appear the same as the other numbers in the table, you could give the Justify Row command, marking the totals row and specifying Comma justification. The table would then look like this:

```
Expense Summary
    Department  |  1st    :  2nd    :  3rd    :  4th    :
   Finance      |   9,000     8,500     9,300     9,500
   Research     |   4,000     3,500     4,100     4,250
   Marketing    |  15,400    15,300    15,900    16,000
   Development  |  12,400    13,500    11,900    13,100
                   40,800    40,800    41,200    42,850
```

Total Table

You can, in fact, total all rows and all columns in a table with a single command, Total Table. Row totals are inserted in the column you specify, as with Total All Rows, and column totals are inserted in the row you specify, as with Total All Columns. Remember that entries in columns or rows that receive totals are excluded from the totals.

ADDING LINES

You can enhance the appearance of a table by using Table commands to add a line of spaces, equal signs, or other characters to a row. Table inserts a return character followed by as many occurrences of the appropriate character as necessary to match the length of the row (or the length of the first line of the row, if it already contains extra lines). You tell Table where to insert the line by marking a character in the row. The line is inserted at the end of the row (or, if there are already extra lines, at the end of the line you mark). Figure 3 illustrates the type of lines you will learn how to add in this section, along with another use of extra lines that you will learn later.

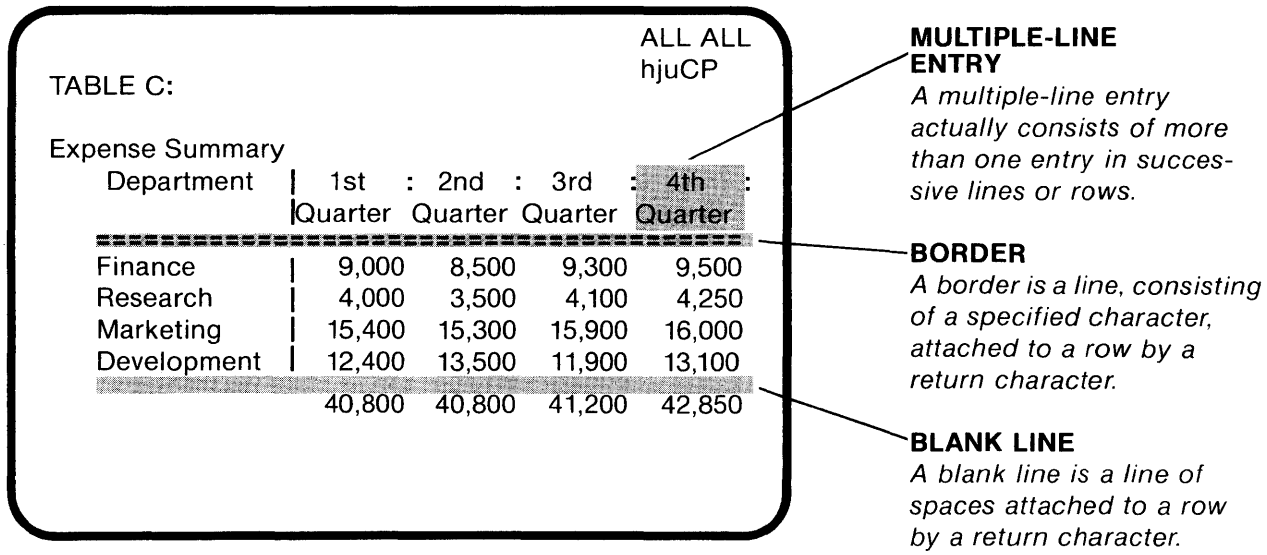
Insert Blank Line

To add a line of spaces to a row, use the Insert Blank Line command. You could, for example, set off the totals row of the Expense Summary table by adding a blank line to the row preceding it (the "Development" row), using the following command and marking any character in that row.

You type:	Command window shows:
ibl	TABLE Insert Blank Line (to follow) M/A:
<MARK><OK>	TABLE Insert Blank Line (to follow) ! !
	TABLE C:

A blank line is not a row.

It is important to remember that a blank line is not a separate row, but rather part of the row you indicate. Although you can mark individual entries in lines beyond the first, you cannot refer to any line other than the first as a row. For example, you could not put the results of a Total All Columns command in the second line of a row.



Most Table commands do not affect entries in lines beyond the first.

FIGURE 3: MULTIPLE-LINE ENTRY, BORDER, AND BLANK LINE

Insert Border

Besides adding a line of spaces to a row, you can add a line consisting of a specified character, such as equal sign (=), with the Insert Border command. For example, you could insert a border of equal signs at the end of the table header in the Expense Summary table as follows:

You type:	Command window shows:
i<SP>b	TABLE Insert Border (of characters) C:
e	TABLE Insert Border (of characters) Equals (to follow) M/A:
<MARK><OK>	TABLE Insert Border (of characters) Equals (to follow) ! !
	TABLE C:

In this case you would mark any character in the table header, and the border would be inserted as a line underneath the header. Besides Equals, you can specify any of the following command words for border characters: Asterisks (*), Backarrows (←), Colons (:), Dashes (−), Exclamations (!), Periods (.), and Uparrows (↑). Note that on most terminals and printers, back arrow appears as an underline (—) and up arrow appears as a circumflex (^).

After the Insert Blank Line and Insert Border commands shown in the examples above, the Expense Summary table would look like this:

```
Expense Summary
    Department  |  1st    :  2nd    :  3rd    :  4th    :
    ============================================================
    Finance     |   9,000     8,500     9,300     9,500
    Research    |   4,000     3,500     4,100     4,250
    Marketing   |  15,400    15,300    15,900    16,000
    Development |  12,400    13,500    11,900    13,100

                   40,800    40,800    41,200    42,850
```

CREATING MULTIPLE-LINE ENTRIES

Many Table commands do not affect second lines.

You can arrange your table so that it appears to contain "multiple-line entries", which in fact would consist of a number of entries in successive lines or rows. Such entries are handled in only a limited way in the Table subsystem. When a row contains more than one line, information in lines beyond the first is often not treated as part of the row. Nevertheless, there may be some applications in which multiple-line entries are useful. You may, for example, want the column headings in the table header to be more than one line long.

Insert Blank Line and Replace Entry

There is no single Table command for creating multiple-line entries, but there are methods that use combinations of commands to achieve the desired effect. One method is to use Insert Blank Line and then replace the blank entries in the line with Replace Entry. (You cannot use Replace Row to replace the blank entries because that command works on only the first line of a row.) For example, you could use this method to create multiple-line column headings in the table header of the Expense Summary table. Suppose you wanted to make the table look like this:

```
Expense Summary
    Department  |  1st    :  2nd    :  3rd    :  4th    :
                |Quarter     Quarter     Quarter     Quarter
    ============================================================
    Finance     |   9,000     8,500     9,300     9,500
    Research    |   4,000     3,500     4,100     4,250
    Marketing   |  15,400    15,300    15,900    16,000
    Development |  12,400    13,500    11,900    13,100

                   40,800    40,800    41,200    42,850
```

You could first insert a blank line at the end of the table header and then replace each of the last four entries in that line with "Quarter", specifying Center justification. There would then be a space under the first vertical bar in the right boundary of the first column; you could replace this space with the Replace Character command.

You arrange the text.

Because Table does not understand the relationship between the first and second lines of a row, there is no automatic carry–over of long text from the first line to the second line, so you must arrange the text in the blank line (or lines) yourself.

Using multiple rows

Another method is to insert a row containing entries that form the second part of the multiple–line entries. If you do this, remember that these "logical" rows containing the multiple–line entries are actually composed of two "physical" rows. Note, for example, that if you wanted to print the table with blank lines between the logical rows, you would have to take into account that you would not want blank lines between the physical rows comprising the multiple–line entries.

Refer to Figure 3 for a quick review of multiple–line entries and extra lines in tables.

FIXING UP TABLES

Careless editing of a table with Base subsystem commands may cause undesirable results when you try to work with the table again in the Table subsystem. So when editing a table that you plan to work with again in Table, you should consider the effects your changes may have on what Table sees as the table format and contents.

Entry beyond row

If there is something wrong with your table, some Table commands will not work properly or at all. You may see an error message, such as "Entry beyond row", in your status window. There are a number of things you should check for if you are having trouble. Be sure you have not deleted the table header or replaced any of the column delimiters in it with an invalid delimiter character. Check that the table header is in fact the first statement of the table plex, and that all statements in the top level of the plex are actually meant to be rows of the table.

Fillout Row or Table

If after making these checks, you still don't see what is wrong with the table, it may be that one of the rows is not as long as it should be according to the table header. This is often difficult to detect because you cannot see the spaces at the end of a row. To help you detect and solve this problem, Table offers the commands Fillout Row and Fillout Table. Fillout Row fills out a specified row with spaces if it is shorter than it should be according to the table header; Fillout Table fills out all rows that are too short.

You type:	Command window shows:
ft	TABLE Fillout Table (at) M/A:
<MARK><OK>	TABLE Fillout Table (at) ! !
	TABLE C:

PRINTING TABLES

Output Processor directives

Tables included in a printed document often require a different format than the rest of the document. When you give a Print command to print an AUGMENT file and you do not specify Quickformat, a program called the "Output Processor" makes it ready for printing. The Output Processor uses an initial printing format, but you can modify this format by using Output Processor "directives".

Blank lines

The initial printing format does not include blank lines between statements. A common practice is to put the directive ".Ybs=1;" at the end of the origin statement so that all statements will be separated by blank lines. If you do this but do not want blank lines between the rows of a table in your file, you can use the following directives:

.Ybs=0; Inserting this directive at the end of table header prevents blank lines from appearing between rows.

.Ybs=1; This directive, inserted at the end of the last row of a table, resumes placing blank lines between statements in the rest of the document.

Keeping the table together

A very useful directive for controlling the output of a table is ".Grab=NUMBER;", where NUMBER is the total number of lines from the statement above the table to the last line of the table, including any blank lines. Placing this directive at the end of the statement above the table prevents the table from being broken between the end of one page and the start of another. For example, the Expense Summary table as it appears in the last example above would have the directive ".Grab=10;" at the end of the "Expense Summary" statement.

Spaces in place of column delimiters

You can also use a directive that will cause the column delimiters in the table header to be replaced by spaces in the printed output. For example, the directive ".Code[':]=' ;" at the end of the table header would cause a space to be printed in place of every colon in the header. Thus, you would not see the colons in the printed table, but they would be there when you wanted to work with the table again in the Table subsystem. To resume the printing of colons as colons, you would use the directive ".Code[':]=':;" in a later statement.

Printing longer lines

An important use of Output Processor directives when printing wide tables is to adjust the margins in effect for your printed output. Note that the maximum length of a printed line is a separate measurement from the maximum for a displayed line. The directive ".Rm=80;" in the origin statement would extend the right margin for a printed line from the initial 72 characters to 80 characters.

Reducing or suppressing indenting

You can also use directives to indent wide tables in your printed output less than they would be indented by the usual automatic level indenting. For example, you can use the Pxi and Pxishow directives to indent your table only slightly under the statement preceding it, or you can use ".Ilev=0;" to align the table with the left margin (suppressing indenting entirely). In either case, you must insert the directive at the

end of the statement above the table and place directives in the last row of the table to restore normal indenting if desired.

Inserting directives

To insert directives into a statement, you can give either the Insert Text command or the Insert Directive command in Base. If you use Insert Text, be sure to type the period and semicolon and to capitalize the first letter, exactly as shown in the examples above. It is easier to use Insert Directive because directive delimiters are inserted without your typing them, and the first letter is capitalized automatically. Be sure not to insert extra spaces around your directives; the Output Processor does not print any of the characters between and including the period and the semicolon, but it does print all other characters. Also be sure to insert the directives at the very end of the statements in the table plex, so that they do not interfere with table manipulation.

The Output Processor Users' Guide gives more information about the directives discussed here and other directives that may be useful in printing tables. The advanced lesson on working with tables will discuss the printing of tables in greater detail.

EXERCISES

1. After creating the branch

```
entries
    55;67;95.3
    34;55;76.1
    77;53;86.2
```

and a statement consisting of the word "Budget", with what five commands could you build the following table and delete the "entries" branch? (Note that the table is indented three spaces under the "Budget" statement.)

```
Budget
    FY79 : FY80 :  FY81  :
       55     67     95.3
       34     55     76.1
       77     53     86.2
```

2. How could you add a column, as shown below, to the table created in Exercise 1?

```
Budget
            : FY79 : FY80 :  FY81  :
   Travel        55     67     95.3
   Supplies      34     55     76.1
   Misc.         77     53     86.2
```

3. If the "entries" branch in Exercise 1 had contained entries for all four of the columns shown in the table in Exercise 2, could you have created the latter table with only five commands? Explain.

4. How could you remove the decimal part of each numeric entry in the last column of the table shown in Exercise 2 and, as shown below, make that column the same width as the second and third columns?

```
Budget
            : FY79 : FY80 : FY81 :
   Travel        55     67     95
   Supplies      34     55     76
   Misc.         77     53     86
```

5. How could you have created the Expense Summary table used in the examples in this lesson without typing commas in the numbers? If you later changed your mind and decided you didn't want commas in the numbers, how could you most easily remove them?

6. With what three Table commands could you total all columns and rows in the table in Example 4 and store the results in the table as shown below?

```
Budget
            : FY79 : FY80 : FY81 :      :
   Travel        55     67     95    217
   Supplies      34     55     76    165
   Misc.         77     53     86    216
                166    175    257    598
```

7. What Table commands could you use to make the table in Exercise 6 look like the table shown below, without adding any more statements to the table plex?

```
Budget
            : FY79 : FY80 : FY81 : Item :
                                    totals
   ---------------------------------------
   Travel        55     67     95    217
   Supplies      34     55     76    165
   Misc.         77     53     86    216
   ---------------------------------------
                166    175    257    598
```

8. In the table below, could you use Total Row or Total Column to add up the numeric entries in any row or column? Why or why not? If not, what changes would you make so that you could add them up?

```
Budget
          |   1    :   2    :   3    :
   -------------------------------
   1979   |  1,300    2,500    3,400
   1980   |  1,250    1,375    1,400
   1981   |  1,500    1,950    1,400
```

SUMMARY

The Table subsystem makes it easy for you to arrange information in an AUGMENT file into a table format. A table is a one–level plex in which the first statement is the table header, which defines the column boundaries and may contain column headings, and the subsequent statements are the rows containing the information. You start a table by inserting a statement under which the table will go, then entering the Table subsystem and telling Table where and how to insert the table header. You then give commands that enter the information into columns defined by column delimiters in the table header. The steps commonly taken to create a table are summarized on the facing page.

With Table commands, you can easily change the content or alignment of table entries, change the column boundaries, insert blank lines and borders, and manipulate table entries in other ways. Various types of editing and totaling operations can be performed on individual entries or on all the entries in a particular row, column, or table. You can work with your table in other subsystems and return to Table when you again want to take advantage of its many useful commands.

The advanced lesson on working with tables will discuss additional, useful Table commands. For complete information about any Table command, use the universal Help command in the Table subsystem.

1. Size the table by deciding on the number of columns and their widths.

   Consider the width of the device on which you will be displaying or printing the table.

2. Use the Insert Statement command in Base to insert a statement under which the table will go.

   This statement can serve as a title or introduction for the table.

3. Go to the Table subsystem and begin the table by giving the Insert Header command.

   This command inserts the top row of the table, the "table header".

4. Give the Replace Row command to enter column headings.

   Type the entries in semicolon format (separated by semicolons or <TAB> characters).

5. To enter information by rows, use the Create Group command.

   First use Insert Statement in Base to create a plex of statements containing the entries in semicolon format. After giving the Create Group (of rows) command, use Delete in Base to delete the structure containing the entries.

Or, to enter information by columns, give the commands Insert Blank Group (of rows) and Replace Column.

   Use Replace Column repeatedly to replace each column.

Leave the Table subsystem with the Quit command.

SUMMARY OF COMMON STEPS FOR CREATING A TABLE

LIST OF COMMANDS

Goto (subsystem) <OPT> (subsystem name) table<OK>
Goto (subsystem) Table <OK>
Execute (command in) Programs Delete All (programs in buffer) <OK>
Execute (command in) Base BASECOMMAND
Quit <OK>

Insert Header (below) LOCATION (with number of columns) CONTENT (of widths) CONTENT

Replace Row (at) LOCATION (by) CONTENT (justified) OK/JUST

Insert Row (after) LOCATION (with) CONTENT (justified) OK/JUST
Create Row (after) LOCATION (from) CONTENT (justified) OK/JUST
Create Group (of rows after) LOCATION (from plex at) LOCATION (justified) OK/JUST

Insert Blank Group (of rows after) LOCATION (number of rows) CONTENT
Replace Column (at) LOCATION (by) CONTENT (justified) OK/JUST

Set Numeric/Text (default justification to) (justified) OK/JUST
Justify Entry/Row/Column/Table (at) LOCATION (justified) OK/JUST

Replace Entry/Wentry (at) LOCATION (by) CONTENT (justified) OK/JUST
Replace Item (at table) LOCATION (Row,Col,Line) CONTENT (by) CONTENT (justified) OK/JUST
Replace Character (at) LOCATION (within the entry) (by) CONTENT
Clear Entry/Row/Column/Table (at) LOCATION
Transpose Entry/Column (at) LOCATION (and at) LOCATION
Delete Row/Column (at) LOCATION
Move Row/Column (from) LOCATION (to follow column at) LOCATION
Copy Column (from) LOCATION (to follow column at) LOCATION
Replace And (change) Row (at) LOCATION (from Column at) LOCATION (justified) OK/JUST
Replace And (change) Column (at) LOCATION (from Row at) LOCATION (justified) OK/JUST
Replace Table (at) LOCATION (by) CONTENT (justified) OK/JUST

Insert Blank Column (to follow) LOCATION (with width) CONTENT
Insert Blank Column (to follow) <OPT> (in front of first column in table at) LOCATION (with width) CONTENT

Insert Field (of spaces after) LOCATION (of width) CONTENT
Delete Field (from) LOCATION (through) LOCATION
Replace Boundary (at) LOCATION (with the character) CONTENT Include (header row) <OK>
Replace Boundary (at) LOCATION (with the character) CONTENT

Insert Blank Row (after) LOCATION
Total Row/Column (of numbers at) LOCATION (enter result as) RESULT
Total All Rows (in table at) LOCATION (put results in column at) LOCATION
Total All Columns (in table at) LOCATION (put results in row at) LOCATION
Total Table (at) LOCATION (putting row totals in column at) LOCATION (putting column totals in row at) LOCATION

Insert Blank Line (to follow) LOCATION
Insert Border (of characters) CHARTYPE (to follow) LOCATION

Fillout Row/Table (at) LOCATION

Definitions:

BASECOMMAND	Give any Base subsystem command.
LOCATION	Prompted by "M/A:" For M you may <MARK>. (If at the end of a command, <MARK> must be followed by <OK>.) For A you may type an address, ending with <OK>.
CONTENT	Prompted by "M/T/[A]:" For M you may <MARK>. (If at the end of a command, <MARK> must be followed by <OK>.) For T you may type a series of characters, ending with <OK>.
JUST	Select one of the following command words and then type <OK>: Left, Right, Center, Same, Mixed, Integer, Comma, Icomma, Places, One, Two, and Point.
RESULT	Select one of the following alternatives: Entry (at) LOCATION Visible (after visible at) LOCATION Status (window) <OK>
CHARTYPE	Select one of the command words Asterisks, Backarrows, Colons, Dashes, Equals, Exclamations, Periods, and Uparrows.

VOCABULARY

The page numbers indicate where the vocabulary item is discussed in this lesson.

blank: A blank entry is one that consists entirely of spaces. A blank row or column consists entirely of blank entries. Page 5

branch: A statement plus all of its substructure.

Clear command: In the Table subsystem, a command that replaces an individual entry or each entry in a row, column, or table by spaces. Pages 15, 16

column: The vertical unit of a table. Most Table commands that work on columns exclude entries in the table header, and all exclude entries in lines beyond the first. Pages 1, 11

column boundary: The character between columns in every row of a table. Pages 5, 18

column delimiters: In a table header, the characters indicating the column boundaries. These may be any of the characters colon (:), vertical bar (|), and exclamation point (!). Page 5

column headings: You can put column headings for your table in the table header. Pages 6, 24

coordinates: The row number, column number, and optionally the line number of an entry in a table; one way of specifying an entry to be replaced. Page 15

Copy Column command: A Table subsystem command that copies a column to follow another column that you indicate. Page 16

Create Group (of rows) command: A Table subsystem command that enters rows of information into a table from a plex of statements containing the entries in semicolon format. Page 8

Create Row command: A Table subsystem command that enters a row of information into a table from a statement containing the entries in semicolon format. Page 8

Delete Column command: In the Table subsystem, a command that removes a column that you indicate. Page 16

Delete Field command: A Table subsystem command that deletes a field between and including two character positions that you indicate. Page 18

Delete Row command: In the Table subsystem, a command that removes a row that you indicate. Page 16

directives: Instructions imbedded in the text of a file that modify the format when you use the Output Processor to print the file. Page 26

entry: In a row of a table, all the characters between (but not including) the column boundaries defined by a pair of column delimiters in the table header. It also refers to text before it is entered into a table (as when you type an entry). Page 11

Execute command: An AUGMENT command that allows you to enter another subsystem only to give a single command in that subsystem. Page 2

field: In the Table subsystem, a series of characters from one character position through another in every row of a table. Page 18

Fillout command: A Table subsystem command that fills out a specified row, or each row of a specified table, with spaces if the row is too short. Page 25

Goto command: The AUGMENT command you use to enter the Table subsystem (or any other subsystem) to gain access to its commands. Page 2

header: See table header.

Insert Blank Column command: A Table subsystem command that inserts a column of blank entries following a column that you mark. Page 17

Insert Blank Group (of rows) command: A Table subsystem command that inserts a specified number of blank rows following a row that you indicate. Page 10

Insert Blank Line command: A Table subsystem command that adds a line of spaces to a row that you indicate. Page 22

Insert Blank Row command: A Table subsystem command that inserts a row of blank entries following a row that you indicate. Page 20

Insert Border command: A Table subsystem command that adds a line of equal signs (=) or other characters to a row that you indicate. Page 23

Insert Field command: A Table subsystem command that inserts a field of spaces after a character position that you mark. Page 18

Insert Header command: A Table subsystem command that inserts a table header below a statement that you indicate. Page 4

Insert Row command: A Table subsystem command that enters a row of information into a table. You can type the entries in semicolon format or mark an existing row. Page 7

justification: The alignment of a table entry relative to the column boundaries defined in the table header. Page 12

Justify command: A Table subsystem command that justifies an individual table entry or each entry in a specified row, column, or table. Page 12

Move command: A Table subsystem command that moves a row or column to follow another one that you indicate. Page 16

numeric default justification: The justification used by Table commands that perform arithmetic operations without asking for the justification. Page 12

numeric entry: An entry that contains digits (0 through 9) and may also contain, in the appropriate places, a plus sign (+), a minus sign (−), a dollar sign ($), a decimal point (.), and commas (,). Page 13

Output Processor: When you give a Print command to print an AUGMENT file and you do not specify Quickformat, the Output Processor makes the file ready for printing. Page 26

plex: A branch plus all the other branches having the same upstatement. "One−level plex" means a statement plus all the other statements having the same upstatement.

Quit command: The AUGMENT command you use to leave the Table subsystem and return to the subsystem you were last working in. Page 2

repeat mode: When you type <RC> in place of the final <OK> at the end of a command, you enter repeat mode, causing the command to repeat until you type <CD>. Page 8

Replace And (change) command: A Table subsystem command that changes a row into a column or a column into a row. Page 16

Replace Boundary command: A Table subsystem command that replaces the characters comprising a column boundary with a character that you specify. Page 19

Replace Character command: In the Table subsystem, a command that replaces a character within a table entry by text that you type or mark. Page 15

Replace Column command: A Table subsystem command that replaces a column by entries you type in semicolon format or by a column you mark. Page 10

Replace Entry command: A Table subsystem command that replaces a table entry you mark by another entry that you type or mark. Page 14

Replace Item command: A Table subsystem command that replaces a table entry whose coordinates you type by another entry that you type or mark. Page 15

Replace Row command: A Table subsystem command that replaces a row by entries you type in semicolon format or by a row you mark. Page 6

Replace Table command: A Table subsystem command you can use to replace all the entries in a table by entries in another table that you indicate. Page 16

Replace Wentry command: A Table subsystem command that replaces a table entry you mark by an entry you type or by a word you mark. Page 15

row: The horizontal unit of a table; a statement in the table plex. Pages 1, 11

semicolon format: In a Table command, separating entries with semicolons or <TAB> characters. Page 7

Set Numeric (default justification) command: A Table subsystem command that changes the numeric default justification. Page 12

Set Text (default justification) command: A Table subsystem command that changes the text default justification. Page 12

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

table: A plex of statements at one level where information is arranged into horizontal rows and vertical columns. Page 1

table header: The top row of a table. It contains the column delimiters that define column boundaries for Table commands. Page 4

table plex: The plex formed by the statements comprising the rows of a table, excluding any substatements inserted beneath a row. Pages 1, 5

Table subsystem: A subsystem that helps you enter information into an AUGMENT file in a table format and perform editing or arithmetic operations on the table contents. Page 1

text default justification: The justification used by Table when you type <OK> in response to "(justified) OK/C:". Page 12

Total command: In the Table subsystem, a command that adds up the numeric entries in a single row or column, in all rows or columns, or in an entire table. Page 20

Transpose command: In the Table subsystem, a command that transposes two entries or columns. Pages 15, 16

visible: A series of printing characters, bounded by "invisible" characters such as spaces and return characters.

SOLUTIONS TO EXERCISES

1. After creating the "entries" branch and the "Budget" statement, you could build the table and then delete the "entries" branch with the following five commands:

   a. The Goto command, typing "<OPT>table<OK>" or, if you have already entered Table during this AUGMENT session, just "t" followed by <OK>.

   b. The Insert Header command, marking the "Budget" statement and specifying three columns and the widths "6;6;8".

   c. The Replace Row command, marking the table header and specifying the entries "FY79;FY80;FY81" with Center justification.

   d. The Create Group (of rows) command, marking first the table header and then any character in the plex within the "entries" branch, and specifying Right justification.

   e. The Execute command, specifying the Base subsystem and, at the "BASE C:" prompt, Delete Branch to delete the "entries" branch.

2. You could add the column by first using the Insert Blank Column command, typing <OPT> after "(to follow) OPT/M/[A]:", marking any character in the table, and specifying the width 9; this would insert a blank column in front of the table. You would then use the Replace Column command, marking the blank column and specifying the entries "Travel;Supplies;Misc." with Left justification (or <OK> for the text default justification, if it has not been changed from Left).

3. Yes, you could have used only five commands to create the table shown in Exercise 2 from an "entries" branch containing entries for all four columns. You would follow the same basic steps as for the table in Exercise 1, allowing for the extra column, except that you would specify Mixed justification in the Create Group (of rows) command. Unless you have changed either of the default justifications, the Mixed justification type would left justify text entries and right justify numeric entries.

4. You could remove the decimal part and narrow the last column by using the Delete Field command and marking the two-character field extending from the decimal point to the last position in the column. Or you could first remove the decimal part by using Justify Column, specifying Integer justification, and then narrow the column by using Delete Field and marking the two-character field of spaces at the beginning of the column. In either case, you must then use the Justify Entry command to justify the "FY81" column heading, specifying Center justification.

5. You could have specified Comma justification instead of Right justification when you entered the rows (or columns) containing numeric information. The commas would have been inserted into each numeric entry for you, and all entries would have been right justified. If you later decided you didn't want the commas after all, you could remove them by giving the Justify Table command and specifying Integer justification; however, since this would right justify the entries in the first column, you would then have to use Justify Column to restore Left justification to that column.

6. You would first use Insert Blank Row to insert a row of blank entries after the last row in the table, then Insert Blank Column to insert a blank column with a width of six characters after the last column in the table. You could then use the single command Total Table to add up all the columns and rows in the table, marking the blank column as the place to put row totals and the blank row as the place to put column totals.

7. You could use the following commands: Insert Blank Line, to insert a blank line below the table header; Replace Entry, specifying Center justification, to replace the blank entries in the table header by "Item" and "totals"; and Insert Border, specifying dashes, to insert a border first after of the last line of the table header and then after the "Misc." row.

8. Using Total Row on any row in the table provides an undesirable result because Table considers the entries "1979", "1980", and "1981" to be numeric entries, so they would be included in the row totals. To avoid this, you could replace the first column with nonnumeric entries, such as "FY79", "FY80", and "FY81". Total Column, however, could be used to add up the entries in any column of this table because, like many Table commands that operate on columns, it does not include the entry in the table header.

Price $3.00

GETTING HELP

Copyright Tymshare, Inc., November 1980
20705 Valley Green Drive
Cupertino, California 95014

This document was prepared and composed for printing with AUGMENT by the Office Automation Division of Tymshare, Inc.

AUGMENT Journal Number 75262
Tymshare Document Number **258**(11/80)0.5m7010

This lesson teaches you how to learn about AUGMENT by asking questions of the system and how to use special AUGMENT facilities to request assistance and services from Tymshare's Office Automation Division. To understand this lesson, you should know how to log in and enter AUGMENT and should have some working experience with elementary AUGMENT commands, such as those described in the lesson "Beginning Use of AUGMENT".

CONTENTS

INTRODUCTION

AUGMENT was specially designed to be easy to learn as you work. AUGMENT commands use simple English words and have a basic, consistent form. Almost all commands begin with a verb followed by a noun. AUGMENT commands also contain prompts and noise words to give you general information about what you can do next. However, there will be times when you need more information. For example, when you see a "C:" prompt, you know you should type a command word -- but what command word? Perhaps you know there is a Force command, but you are not sure what it does, or you think there is a command to help you with capitalization, but you don't know what it is or exactly how it works. In cases like these, you can find the answers to your questions by asking AUGMENT itself.

Online aids

This lesson will teach you about AUGMENT's online aids by showing how you would use them to learn a new editing command. When you finish this lesson, you should be able to follow the procedures taught here to learn any new commands that seem interesting or useful. You also can use these methods to investigate other aspects of AUGMENT, to learn about AUGMENT terms and procedures, or to learn about whole new subsystems and capabilities.

Online documentation

What if you want to read documents about AUGMENT? How do you find out what is available, where it is, and how to get it? AUGMENT provides a complete guide to all available documentation and reference literature. In this lesson you will learn to use this guide to locate the documentation you need.

Help from people

Of course, there will inevitably be a time when you need some kind of help you cannot get from AUGMENT, and what you really need is help from a person. The Feedback service, described in this lesson, provides an easy, effective method for getting help or advice. You can also use Feedback to register a complaint, a suggestion, or a compliment.

LEARNING WHAT YOU CAN DO NEXT: QUESTION MARK

How ? can help you

The simplest way to learn about AUGMENT while you work is by using question mark (?). Without interrupting the command you are giving, you can type a question mark to see a list of everything you can do next. This list will help you remember commands you have forgotten, work your way through unfamiliar commands, discover new ways to use familiar command verbs, and even learn whole new commands.

When and how to use ?

You can type a question mark after any prompt in an AUGMENT command. For example, typing a question mark after any "C:" prompt will tell you all the command words you can use at that point. As AUGMENT lists your alternatives, your command window will expand into the file window and you will be temporarily unable to see the information at the top of the file window. The information in your file, however, will not be affected. At the end of the list, AUGMENT will again display the prompt you questioned and wait for you to study the alternatives, make a choice, and give the next part of the command. When you do, your command window will return to its previous size, AUGMENT will display the command (including the part you have just given), and you can continue as if you had not used question mark.

Suppose you are working in the Base subsystem and you notice an entire statement entered in lowercase that should be all uppercase. If you were reluctant to retype the whole statement, you might decide to see whether AUGMENT has a command that could help. This would be a good time to use question mark.

? at "BASE C:"

You could begin by typing a question mark at "BASE C:". This will show you a list of all the command words that begin Base subsystem commands, as shown below. Note that since commands and features are added or changed in AUGMENT from time to time, the list of alternatives which actually appears on your screen may differ slightly from what you see in the examples.

You type:	Command window shows:
	BASE C:
?	BASE C: ?

```
Current alternatives are
   APPEND      BREAK      <>CHECK     <>CLEAR     <>COMMENT
<>CONNECT      COPY       <>CREATE      DELETE    <>ENLARGE
   EXECUTE  <>EXPUNGE       FORCE    <>FREEZE       GOTO
   HELP        INSERT       JUMP       LOGOUT       MOVE
<>POINT        PRINT      <>PROCESS     QUIT      <>RENAME
   REPLACE  <>RESET      <>REVERSE   <>SET          SHOW
<>SORT      <>START      <>STOP      <>THAW         TRANSPOSE
<>TRIM      <>TYPE       <>UNDELETE    UPDATE
   .            /            ;           \           ↑

C:
```

<> before a command word

Notice that "<>" precedes some command words. This means that to give that command word, you must type a space and then the first character or characters before AUGMENT will recognize the word.

Selecting an alternative

After reading this list, you might recall that "Force" is the command that changes capitalization. If you decided to try this command, you would begin it by typing "f" at the "C:" prompt following the list. Your file window would then return to the view you had before typing the question mark, and your command window would display the beginning of the Force command.

You type:	Command window shows:
	C:
f	BASE Force (case of) OPT/C:

? at an "OPT/C:" prompt

Here again you might want to know your alternatives. You could type a question mark again to see a list of the responses you could give to the new prompt, "OPT/C:". Among them you would discover the word "STATEMENT". Since this is what you want to edit, you would then type "s" as the next step in the command. The entire process would look like this:

You type:	Command window shows:
	BASE Force (case of) OPT/C:
?	BASE Force (case of) ?
	Current alternatives are
	BRANCH CHARACTER DIRECTIVE GROUP INVISIBLE
	LINK NUMBER <>PHRASE PLEX STATEMENT
	TEXT VISIBLE WORD <OPT>
	OPT/C:
s	BASE Force (case of) Statement (at) M/A:

<OPT> as an alternative

Notice that in the example above, question mark not only listed everything you can "force" with the Force command but also indicated that you can type <OPT>, as you can tell from the "OPT/C:" prompt. "OPT" in a prompt means that at this point in the command you have not only the standard choices listed when you use question mark, but also some additional options. To ask for these options, type <OPT>. You will then see another prompt and can respond accordingly, or you can once again use question mark to list your alternatives.

? at prompts not involving command words

At some steps in commands, AUGMENT does not expect a command word but instead waits for you to mark something on the screen, to give the location of something, or to type some text. In these cases, rather than a prompt containing a "C", you will see some other prompt. Question mark here will again show you what AUGMENT expects by listing your choices. If you queried the next step in the Force command, for example, you would see the following:

You type:	Command window shows:
	BASE Force (case of) Statement (at) M/A:
?	BASE Force (case of) Statement (at) M/A: ? Current alternatives are Mark the STATEMENT Type the address of the STATEMENT M/A:

Removing the list

If, after listing your alternatives with question mark, you need to remove the list before continuing (as when you need to mark something covered by the list), type any character that is not allowed as a response to the prompt you questioned. AUGMENT will then return you to where you were before you used question mark. To remove the list after a prompt that you can respond to by typing an address or some text, type <BC>, because AUGMENT will accept any other character as the first character of the address or text. Typing <BC> will not only get rid of the list but also backspace over the previous step in the command; that is, it will erase the command word you gave before you got the prompt you questioned. Thus you will have to give that command word over again.

<LIT> for typing ? as text

There may be times when you need to type text beginning with a question mark (if, for example, you are using the Insert Character command to insert "?" at the end of a question). But how do you type a question mark right after a prompt without causing AUGMENT to list your current alternatives? To type text beginning with a "?", you must first type <LIT>. <LIT> tells AUGMENT that the following character is literally text, not an instruction to do something. Of course, typing a question mark after other characters is not a problem, because the question mark does not immediately follow a prompt.

You have now learned how you can use question mark, along with the information you get from noise words and prompts, to go through an unfamiliar command step by step. You could continue this process until you either figured out how to use the Force command or realized it was not the command you needed and typed <CD> to cancel it. If, however, you felt a need for more complete information about the Force command or about how capitalization works in AUGMENT, you might want to take advantage of another AUGMENT online aid, Help.

GETTING COMPLETE INFORMATION ABOUT AUGMENT: HELP

What you can learn from Help

Help is designed to provide the most detailed and up-to-date information available on AUGMENT. Help will give you the definition of any AUGMENT command or term, descriptions of AUGMENT procedures, or advice on how to accomplish a particular task. You can use Help to answer specific questions you may have, or just to browse through a general subject. It will help you not only to find the particular information you need but to explore all of AUGMENT.

Why use Help?

Many people find that working with Help is awkward at first, because they are used to turning to a familiar reference manual or a person when they need assistance. Using Help is different from thumbing through a reference manual 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 having to leave the terminal. Furthermore, Help is always there; if you are using AUGMENT, you can be sure of getting Help. You will find that the ability to use Help will make you a more self-sufficient AUGMENT user. You will be able to answer your own questions and take on new tasks with more confidence.

What Help is

Help is based on information arranged in large files that are much like any other AUGMENT files. Every subsystem that is supported in AUGMENT has a Help file which contains the information you need to use that subsystem and references to other topics you may find interesting. In addition to the files associated with particular subsystems, there are also general files that deal with commands available in all subsystems and information applicable to all of AUGMENT. The Help files represent a great deal of knowledge about AUGMENT, but to take advantage of them you do not need to know the names of these files or even how they are organized. Instead you can use the AUGMENT Help command, or simply type <HELP> after any prompt in an AUGMENT command. The following sections discuss getting information about AUGMENT by using Help -- how to find the information you need and how to read the information you find.

Looking Up Information in Help

The Help command

To look up information in Help, you can use the Help command, a command available in every AUGMENT subsystem. To begin the Help command, type "h" for "Help", at "BASE C:" in the Base subsystem or, if you are in some other subsystem, when you are prompted for a command in that subsystem.

You type:	Command window shows:
h	BASE Help (type a term and then <OK>, or just <OK>) OK/T:

Type <OK> to learn about the subsystem.

When you see the "OK/T:" prompt, if you don't have a specific problem you need help with, simply type <OK>. Help will then give you a general discussion of the subsystem you are using and follow it with a list of topics for further reading.

Typing terms for Help to look up

To look up a specific term or command, type it after the "OK/T:" prompt and end it with <OK>. Help terms must begin with a letter, can be uppercase or lowercase, and may be one word or a number of words separated by spaces or dashes (-). They may contain numbers but not punctuation and not any other symbols except dash. If you want to use Help to look up something that normally includes angle brackets (<>) or parentheses, simply omit these characters when typing

the term. Thus, if you want help with the term "<OK>", type "OK". If you want information about a command, do not include the noise words; for example, type "Force Statement", not "Force (case of) Statement".

Searching index

While Help looks for the information you requested, your screen, except for your viewspec window, will go temporarily blank. If your term cannot be found in the Help file for the subsystem you are using, Help checks other files. If this is necessary, you will see the message "searching index" in your status window.

Help recreates your display in a special way.

When Help finds the information you requested, your command window will show the prompt "C/T:" and your file window will display the Help information. Help will temporarily alter your viewspecs, so do not be disturbed by changes in your viewspec window. Once you have entered Help, you may remain there and ask for further information for as long as you wish. Figure 1 shows how your screen looks while you are in Help.

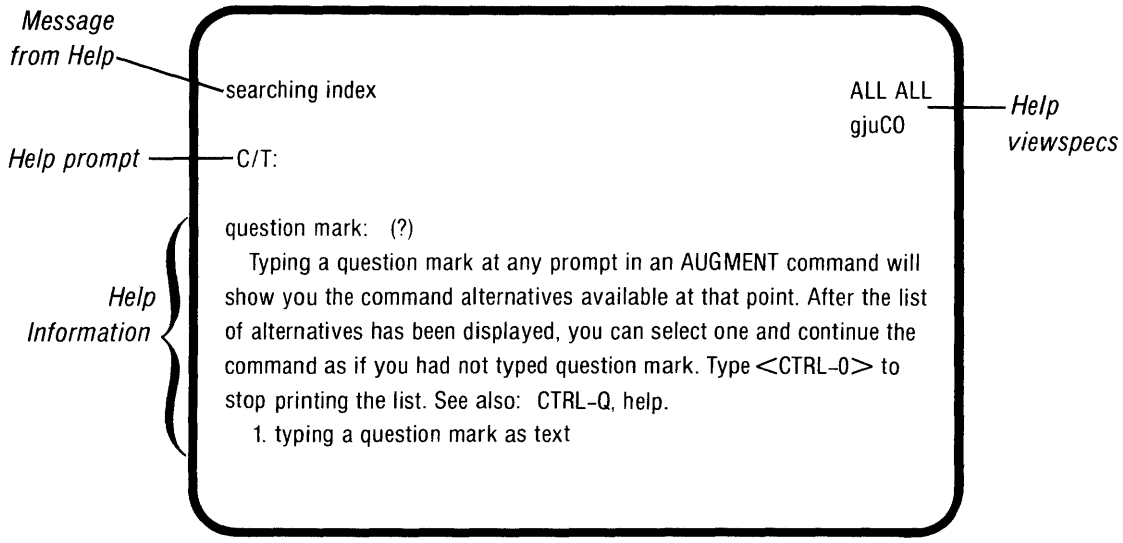


FIGURE 1: THE SCREEN ONCE YOU HAVE ENTERED HELP

Leaving Help

To return to your work in progress after using Help, simply cancel the Help command by typing the "command delete" character, <CD>. When you leave Help, the view that was displayed in your file window when you began the Help command will be restored, and your viewspecs will also be the same as before you started.

The following example shows how you could use Help to begin learning about capitalization in AUGMENT by investigating the term "uppercase". Note that since the information in Help changes to reflect changes in AUGMENT, what you read in this and later examples may not be exactly what you see when you use Help.

You type:	Command window shows:
h	BASE Help (type a term and then <OK>, or just <OK>) OK/T:
uppercase<OK>	BASE Help (type a term and then <OK>, or just <OK>) uppercase!

Your display would then be recreated to show the prompt "C/T:" and the Help information as follows:

```
 C/T:
uppercase

case
  The word "case" is used in AUGMENT to describe whether letters are
capitalized (uppercase) or not capitalized (lowercase). AUGMENT commands will
allow you to control and change case in online text and in printing.
    1. to change the case of a STRING or STRUCTURE: the Force command
    2. CASEMODE: to set the case for the Force command
    3. to change the default CASEMODE
    4. to reset CASEMODE to the default
    5. to control case in printing
```

Help may not define precisely what you expect.

As shown in this example, the term that Help defines may not always be exactly the same one that you typed and, in fact, the information Help offers may not be precisely what you expected. When you ask about a general term, such as "uppercase", Help begins by searching to see if the term you typed is identical to a Help term. If it is, Help then displays the information about that term. If your term is not identical to a Help term, Help displays information about a similar or equivalent term. In the example, Help determined that the term "uppercase" was covered by the definition of "case".

When Help can't find a term

If the term you type is not like any term defined in Help, or you make a typing error and Help cannot recognize the term, Help will display the term in your status window followed by a question mark. Below this will be the "C/T:" prompt. You can then retype the term if you made a typing error or, if you typed correctly but Help does not know the term, you can try a similar term or a different one altogether.

When Help can't find a two-word term

NOTE: Because some terms that are normally typed as two words can also be spelled as one word, or as two words separated by a dash, Help occasionally has trouble with two-word terms. If you are looking up a two-word term and Help fails to find it or gives you a definition applying only to the first word of the term, try typing the term as one word or as a hyphenated word. For instance, try typing "online" or "on-line" instead of "on line".

Making suggestions and reporting problems

AUGMENT's Help must serve many different kinds of users from varying backgrounds; some know a lot about computers, whereas many are using them for the first time. Thus some users may look up different terms or want different information than others. We have tried to ensure that Help will be useful to everyone, but we may not have always

succeeded. If you have suggestions or problems, please send them to Feedback as described later in this lesson.

Reading and Using Help Descriptions

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 the basic information necessary to understand the subject or command you have asked about and indicate related subjects and commands for you to look up.

Menu

Every Help description that has related topics will have a list below it, called a "menu". For example, the explanation of the term "case" on page 7 has a menu with five items. Each menu item consists of a single line that refers you to a detailed discussion of the topic it mentions. To read about a subject presented in a menu item, type its number followed by <OK>. If you accidentally type a number not on the menu, the number followed by a question mark will be displayed in your status window and you will again see the Help prompt. Enter the correct number. If, after reading about case, you wanted to read more about the Force command, you could ask for menu item 1 like this:

You type:	Command window shows:
	C/T:
1<OK>	1!
	C/T:

In your file window you would see:

```
to change the case of a STRING or STRUCTURE:  the Force command

Force (case of)
  The Base Force command allows you to control the capitalization of text in
the situations listed below.  For example, you can change a STRING, such as a
word or a phrase, from all lowercase to only the first letter of each word in
uppercase, or you can change a STRUCTURE, such as an entire statement, from all
uppercase to all lowercase.  See Set Force, Reset Force.
    1. STRING:  Force (case of) [CASEMODE (for)] STRING (at) LOCATION CASEMODE
    2. STRUCTURE:  Force (case of) [CASEMODE (for)] STRUCTURE (at) LOCATION
```

NOTE: Because the line length shown here may differ from what you see on your terminal, the menu items may not look exactly the same.

Long menus

Sometimes, when a menu is particularly long, you will see only part of the menu and the message "(do you want to see the rest of the menu?) Y/N:". If you wish to read the rest of the menu, type "y" or <OK> for "yes"; otherwise, type "n" for "no". As you read the menu, it is a good practice to note the numbers of any menu items you will want to read. You can ask to see any item on a menu, regardless of whether that item is displayed, so you can look over the whole menu and then go through the topics that interest you. If, after you have finished

reading the menu, you want to redisplay it from the beginning, type <OK> at "C/T:".

The general form of a Help description

Most descriptions in Help have a form like the description of Force in the previous example. At the top you will see the term or menu item to be explained. Below this will usually be a paragraph describing the term, and then a menu, which may contain several kinds of items. Some menu items may refer you to related topics, some may point to explanations of the different meanings of a term, and, if you are reading a general description of a command, some may lead you to a precise definition of a way of using the command. The example shows this last kind of menu item. The menu guides you to definitions of using the Force command on two types of entities: STRING and STRUCTURE.

Command syntax

The precise command definitions always include the "syntax" for the command they describe; that is, they show the sequence of steps in the command so you will know exactly how to use it. Do not be surprised if the command syntax shown in the menu item appears to be incomplete. Because menu items are restricted to one line, there may be times when the entire command syntax cannot be shown in the menu. Once you ask for a menu item, however, you will see the complete syntax.

Continuing from the previous example, if you were studying the Force command and wanted to find out more about using Force on a STRING, you would take menu item 1 as follows:

You type:	Command window shows:
	C/T:
1<OK>	1!
	C/T:

Command definition

Your file window would then display the command definition for the Force STRING command. It would include the complete command syntax, a paragraph describing exactly how to use the command, and menu items referring you to various related topics. This definition is abbreviated here:

> STRING: Force (case of) [CASEMODE (for)] STRING (at) LOCATION CASEMODE OK
> The Base "Force (case of) STRING" command changes the capitalization of letters in a STRING to the case you specify. ... If you want to change the case of statements, see the Force STRUCTURE command. See repeat command, OPT.
>
> 1. CASEMODE = Upper, Lower, First (letter upper), or Sentence (case)
> 2. STRING = Character, Text, Word, Visible, Invisible, Number, Phrase,

Uppercase words represent choices.

The uppercase words shown in command syntax represent what you can do at different steps in a command. Sometimes they represent special characters. For example, OK in command syntax means that you type <OK>. Often an uppercase word represents choices you have at that point in the command; it is like a "variable" for which you may select

any of the choices listed for it. For example, when you see the word STRING, it means that at this point in the command you should select the command word you want from among Character, Text, Word, and so on. Although this notation may seem a bit confusing at first, it is valuable. Without it, Help descriptions would be too long and repetitive. If you are studying a command in Help and want to learn what an uppercase word represents, you might find an explanation in the description itself or a menu item that mentions it. In the example above, for instance, to further investigate the choices for STRING, you would look at menu item 2.

[ ] indicate options.

You will notice that some steps shown in command syntax are enclosed in square brackets. These are optional steps that you may skip completely or may choose by first typing <OPT>.

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 learn more about any of these terms, simply type it at the "C/T:" prompt, ending with <OK>; do not type the comma following a term. In the previous example, the Force STRING description suggested that you see OPT. The following is what would happen if you decided to do this.

You type:	Command window shows:
	C/T:
opt<OK>	Opt!
	C/T:

Your file window would then show a description of OPT, as follows:

```
OPT stands for

OPTION
  Many AUGMENT commands have optional parts or optional ways of responding
to prompts that are available only if you ask for them. AUGMENT will show the
presence of these options either with "OPT" or with a letter enclosed in square
brackets ([]), such as [A]. To ask for these options, type <OPT> (OPTION key
or <CTRL-U>). You can then type "?" to see your alternatives. See also:
question mark.
    1. options vs alternatives
```

You can type any term.

The user of Help is not limited to looking up what Help suggests in menu items or "See" and "See also". Many of the terms in Help descriptions are defined elsewhere in Help. If you do not understand a term in a description or are interested in a concept suggested there, try looking it up by typing it at the "C/T:" prompt. From the previous example, for instance, you could proceed in one of several directions: You could follow the menu item; you could look at the topic suggested in "See also"; you could type another term from the description, such

as "prompt" or "CTRL-U"; or, if nothing in this definition caught your fancy, you could type any other AUGMENT term or command.

NOTE: If you are interested in a term covered by a current menu item, you should investigate it by giving the menu item number. If you take the menu item and want more information, you can then try typing the term itself at the "C/T:" prompt.

Getting more information about a term

If after typing a term and reaching an explanation of it, you want more information about the same term, try typing it again at the "C/T:" prompt. Help will look elsewhere for more information and will display it, if there is any. Often it will find a more general discussion of the term.

Stopping a Help search

If you type a term for Help to look up and then decide you are not interested in the term, you can stop the search by typing <CTRL-O>. After the search stops, you will again see the Help prompt, "C/T:".

Getting Around in Help with < and ↑

In addition to typing terms or menu item numbers for Help to look up, you can also use Help without looking up specific topics. You can respond to the "C" in "C/T:" with left angle bracket (<) or up arrow (↑). With left angle bracket you can return to previous views of Help information, and with up arrow you can move from specific definitions up to more general topics in the Help files. As you begin to use Help to explore AUGMENT, you will find many uses for "<" and "↑".

< for going back

You can use left angle bracket (<) to find and redisplay a previous Help description. When you type "<", AUGMENT begins to retrace your steps through Help, and displays in your status window the first few words of the description you saw before your current one. To look at that description again, type "y" or <OK> for "yes"; otherwise, type "n" for "no". If you type "n", AUGMENT will back up another step and display the first few words of the description two before your current one. If you want that description, type "y" or <OK>; otherwise, reject it with "n". This process can continue, with AUGMENT proposing descriptions and you rejecting them, until you reach and accept the description you want. If you reject all the previous descriptions, you will receive the message "Cannot back up farther" and will again see the prompt "C/T:".

Moving on from <

Once you find and accept the description you want, AUGMENT will clear your status window and display the description and another "C/T:" prompt. At this point you can type a term for Help to look up, give a menu item number, or type "<" or "↑". Of course, if you do type "<" here, the first screenful of information you will have a chance to review will be the one at which you gave your previous "<".

Example of using <

The following example shows how you would use "<" to go back from the information in the previous example to the description two steps before it. Notice that you first reject the description headed "STRING: Force (case of)" which you saw on page 9 and then, continuing back,

accept the description beginning "to change the case", shown on page 8.

You type:	Command window shows:	Status window shows:
	C/T:	
<	< (go back) Y/N:	STRING: Force (cas...
n	< (go back) no	
	< (go back) no Y/N:	to change the case...
y	< (go back) no yes	
	C/T:	

You would again see this in your file window:

to change the case of a STRING or STRUCTURE: the Force command

Force (case of)

The Base Force command allows you to change the capitalization of text in the situations listed below. For example, you can change a STRING, such as a word or a phrase, from all lowercase to only the first letter of each word in uppercase, or you can change a STRUCTURE, such as an entire statement, from all uppercase to all lowercase. See Set Force, Reset Force.

1. STRING: Force (case of) [CASEMODE (for)] STRING (at) LOCATION CASEMODE
2. STRUCTURE: Force (case of) [CASEMODE (for)] STRUCTURE (at) LOCATION

As you learn to use Help, you will find that "<" is very useful for studying. You can compare definitions, look again at old descriptions to check and make sure you understand them, or just find out how you got where you are.

↑ for going up

You can use "↑" to go up one level in the structure of the Help files. This will allow you to find the context of Help descriptions, to discover new and related areas, or just to explore the Help files. Of course, typing "↑" may bring you somewhere you have been before. If you arrived at the current Help description by passing from a general discussion to a specific topic, "↑" may just return you to your starting point.

Using ↑

You type "↑" at the "C/T:" prompt. Help will then look for the information requested and display it. If there is no higher-level information in the Help file for the subsystem you are using, Help will display general information about AUGMENT subsystems. If you again go up from there, Help will give you an overview of AUGMENT. If you again try to go up, Help will tell you it cannot go up farther. The example below shows how you could go up from the explanation of the Force command you reached in the previous example.

You type:	Command window shows:
	C/T:
↑	↑ (go up)
	C/T:

You would see in your file window the general discussion of the AUGMENT term "case" that you first saw when you asked for help on "uppercase".

When you mistype < or ↑

NOTE: If you make a mistake and type the wrong character when you want to type "<" or "↑", you can get back to the Help prompt and start over by typing <BC> twice. The first <BC> will erase the character and the second will get you back to the prompt, where you can type the character you wanted.

Reaching Help From Within a Command: <HELP>

There may be times when you need specific information about a command you are in the process of giving but do not want to retype command words into Help or try to find your way through Help to the particular information that you need. Instead of using the Help command in this situation, you can simply type the special character <HELP>. On some terminals there is a HELP key; if your terminal does not have such a key, see the list of special characters for your type of terminal to find the equivalent character.

When to type <HELP>

You can type <HELP> after any prompt in a command. When you type it, you will see "(searching Help information)" in your command window, and your file window will clear to display the information Help has on the command as far as you have given it.

Typing <HELP> puts you in Help.

Typing <HELP> puts you in Help, just as if you had given the Help command. You can interact with Help as previously described as long as you wish. To leave, simply type <CD>, and you will be back where you were before you gave the command you needed help with.

The following example shows how you would use <HELP> if you were in the middle of the Force Statement command and wanted to find out more about the command.

You type:	Command window shows:
	Force (case of) Statement (at) M/A:
<HELP>	Force (case of) Statement (at) (searching Help information) C/T:

The file window would show a detailed explanation of the Force Statement command, abbreviated below. Note that you would get the same information if you used the Help command and typed the term "Force Statement".

Force (case of) Statement is a special case of Force
STRUCTURE: Force (case of) [CASEMODE (for)] STRUCTURE (at) LOCATION
CASEMODE OK
The Base "Force (case of) STRUCTURE" command changes the capitalization of letters in a STRUCTURE to the case you specify. ... If you want to change the case of parts of statements, see the Force STRING command. See repeat command, OPT.
1. CASEMODE = Upper, Lower, First (letter upper), or Sentence (case)
2. STRUCTURE = Statement, Branch, Plex, or Group

FINDING DOCUMENTATION ONLINE: LOCATOR

LOCATOR, a guide to AUGMENT documentation

In addition to Help information, there are documents written about AUGMENT. Stored online for AUGMENT users to read are copies of most of AUGMENT's printed documentation and other documents that have not been printed. A file named LOCATOR serves as a guide to this online documentation and also to printed documents, brochures, and reference literature. You can use LOCATOR to discover what documentation is available, to find online documents you want to read, and to learn how to order printed documentation from Tymshare.

Jump (to) Locator

You can reach LOCATOR with the Jump (to) Locator command, which is available in all subsystems.

You type:	Command window shows:
j	BASE Jump (to) M/C:
<SP>l	BASE Jump (to) Locator OK:
<OK>	BASE Jump (to) Locator !
	BASE C:

Jump (to) Link <MARK>

When you arrive at LOCATOR, you will see headings covering the major categories of documentation listed in LOCATOR. Preceding these headings will be headings leading to instructions on how to use LOCATOR. The first time you use LOCATOR, you will want to read these instructions. To do this, use the Jump (to) Link command and mark the instruction statement.

You type:	Command window shows:
j	BASE Jump (to) M/C:
l	BASE Jump (to) Link M/T/[A]:
<MARK>	BASE Jump (to) Link ! OK:
<OK>	BASE Jump (to) Link ! !
	BASE C:

At this point you will be led through a complete explanation of how to use LOCATOR to find the documentation you need. Rather than duplicate this explanation, the following paragraphs will summarize what you would learn and illustrate the general method presented there. If you are at a terminal and want to read the explanation in LOCATOR, simply give the commands described above.

Getting around in LOCATOR

LOCATOR has an outline structure that organizes the names and descriptions of available AUGMENT documentation under appropriate headings. To find a particular document, you use the Jump (to) Link command and mark statements, making your way though the outline until finally you arrive at the document you want. The statements you mark may be headings or descriptive paragraphs. Here, for example, are the first few steps you would take after reaching LOCATOR if you decided to look for online copies of the AUGMENT Textbook lessons:

1. Since you want to read online, you would Jump (to) Link and mark the heading "ONLINE DOCUMENTATION FOR AUGMENT".

2. Here you would see some general information about the online documentation; to see specifically what is available, you would Jump (to) Link and mark this.

3. You would now see a list of names or subcategories of different online documents. Included would be the heading "Lessons in the AUGMENT Textbook Series"; Jump (to) Link and mark this statement.

The documents are not stored in LOCATOR.

You could continue in this way, using the Jump (to) Link command and marking the statement describing what you want, until you reached the name and then the description of the Textbook lesson that interested you. After using Jump (to) Link and marking this description, you would be in a different file, the file containing the actual lesson. You would no longer be in LOCATOR.

You see a "table of contents" view.

When you arrive in the file containing the documentation, you see a clipped view that shows only the origin statement and all the major headings in the file. To see the whole file (that is, every line of every statement), you can use the Set Viewspecs command and set viewspec w. You can remain in this file as long as you wish, using different viewspecs and the various Jump commands to read as much or as little as you want.

Returning to LOCATOR

To return to LOCATOR and look for other documents, simply give the Jump (to) Locator command again. You will be taken back to LOCATOR, where you will once more see the different categories of documentation and can explore further.

COMMUNICATING WITH PEOPLE: FEEDBACK

There may be times when you have a problem or need some kind of special service that you simply cannot take care of yourself. To fill this need, the organization that supports AUGMENT, the Office Automation Division (OAD), maintains a service called "Feedback". You can send a message to Feedback at any time to ask a question, request assistance, or offer suggestions, complaints, or compliments. Feedback accepts all messages and returns an answer within 24 hours during the work week.

What Feedback does

When you send Feedback a request for services or assistance, an OAD staff member performs the service and informs you of the outcome, or explains any delay. If you send Feedback a report of something that appears to be wrong with the system or a suggestion for an improvement or change, Feedback will forward your message to the appropriate person and report back to you on what is being done.

Sending and receiving Feedback mail

To send a message to Feedback, use AUGMENT's electronic mail capabilities. The Feedback service has its own special user name and "ident", FEEDBACK. Use this name when you send the message. To find out how to send a message to Feedback, use Help and ask about Feedback or about sending mail. To learn what documents are available on sending and reading mail, you could use LOCATOR as described in this lesson. If for some reason you cannot log in, then you can reach the Feedback service by calling the Office Automation Division at Tymshare.

EXERCISES

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

2. What does "OPT" mean in a prompt? How could you find out what you can do after typing <OPT> in a particular command?

3. What is special about changing a sentence into a question by replacing the period with a question mark?

4. Suppose you gave the command word "Force", used question mark to find out what you could "force", and saw the word "VISIBLE" as one of your choices. How could you then find out what a "visible" is?

5. If you asked Help about "Force" and then wanted general information about editing commands like Force, how could you find it?

6. If you are in Help and you notice an unfamiliar uppercase word in the command syntax, what does this represent in general, and how can you find out exactly what it means?

7. What would you do if you were using Help, had gone through several Help descriptions, and then wanted to reread the first description you saw?

8. If you were using Help to learn about the Force Visible command and then wanted to try the command, what would you do?

9. If in the middle of giving the Force Visible command you became worried about what was going to happen, what could you do?

10. What would you do if you wanted to read this lesson online?

SUMMARY

There are several ways to get information or assistance when you are using AUGMENT. At any point in a command, you may type a question mark; AUGMENT will list your alternatives and you can then choose among them or delete the command. For more complete information about a command you are in the middle of giving, you can type <HELP> to read the Help information about that command. You can also use the Help command to ask Help about commands, terms, or procedures and to find all sorts of information about AUGMENT. To find out about printed and online documentation and to read what is available online, you can use LOCATOR, the guide to AUGMENT documentation. If you have questions, problems, or suggestions, or need a special service, you may send a message to Feedback.

AUGMENT is a large system that includes many powerful tools and techniques. We encourage you to use question mark, the Help information, the documents pointed to in LOCATOR, and the Feedback service to learn about AUGMENT as you work and to become a self–sufficient and creative user. Even when you know enough to get your work done, you can always learn new commands, expand your knowledge of old ones, and discover more effective and more interesting methods of working.

LIST OF COMMANDS

?

Help (type a term and then <OK>, or just <OK>) OK/TERM ("HELPINFO") <CD>
Help (type a term and then <OK>, or just <OK>) OK/TERM ("HELPINFO") TERM/MENU ("HELPINFO") ... <CD>
Help (type a term and then <OK>, or just <OK>) OK/TERM ("HELPINFO") TERM/MENU ("HELPINFO") < (go back) ("FLASHBACK") ANSWER ... <CD>
Help (type a term and then <OK>, or just <OK>) OK/TERM ("HELPINFO") ↑ (go up) ("HELPINFO") ... <CD>

<HELP> (searching Help information) ("HELPINFO") <CD>
<HELP> (searching Help information) ("HELPINFO") TERM/MENU ("HELPINFO") ... <CD>
<HELP> (searching Help information) ("HELPINFO") TERM/MENU ("HELPINFO") < (go back) ("FLASHBACK") ANSWER ... <CD>
<HELP> (searching Help information) ("HELPINFO") ↑ (go up) ("HELPINFO") ... <CD>

Jump (to) Locator <OK>
Jump (to) Link <MARK> <OK>

Definitions:

TERM	Type a word or several words separated by spaces or dashes, ending with <OK>.
MENU	Type a menu item number, ending with <OK>.
ANSWER	Type "y" or <OK> for "yes", or "n" for "no".
("HELPINFO")	This stands for the Help information you will see in your file window.
("FLASHBACK")	This stands for the first few words of the old Help information, displayed in your status window.

VOCABULARY

The page numbers indicate where the vocabulary item is discussed in this lesson.

<CTRL-O>: Type <CTRL-O> to stop Help from searching for a term you have asked about. Page 11

electronic mail: This term refers to using AUGMENT to send and receive messages. Page 16

Feedback: A service provided by the Office Automation Division of Tymshare, Inc., that enables users to request assistance and register complaints, suggestions, or compliments. Page 15

FEEDBACK: The name and "ident" used to address Feedback through AUGMENT electronic mail. Page 16

Help: A vast source of information about AUGMENT commands, terms, and procedures, available through the Help command and <HELP>. Page 5

<HELP>: You can type <HELP> after any prompt in an AUGMENT command to get a Help description of the command as far as you have given it. Page 13

Help command: An AUGMENT command to get information about commands, terms, and procedures. Page 5

Jump (to) Link command: An AUGMENT command you can use to move between files and to move around within files. You use this command to get around in LOCATOR and to move from LOCATOR to other files for online reading. Page 14

Jump (to) Locator command: The AUGMENT command used to reach LOCATOR, the guide to available AUGMENT documentation. Page 14

left angle bracket: While using the Help command, you can type a left angle bracket (<) to review previous descriptions displayed by Help. Page 11

<LIT>: Typing <LIT> tells AUGMENT that the following character should be taken literally as text, not as an instruction to do something. Page 4

LOCATOR: A file pointing to AUGMENT documentation, which you reach with the Jump (to) Locator command. LOCATOR will tell you what documentation is available, guide you to copies for online reading, and tell you how to order printed copies. Page 14

menu item: A subtopic, to guide you to related information, listed under a Help description. Page 8

OAD: The Office Automation Division of Tymshare, Inc. OAD supports AUGMENT.

<OPT>: You type <OPT> to request an optional step in a command, in response to either "OPT" in a prompt or any part of a prompt that is enclosed in square brackets. Page 3

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 list what you can do next. Page 2

syntax: The word "syntax" means "how things are put together". Thus the syntax of a command shows the sequence of steps in the command. Page 9

term: A term is one or more words, separated by spaces or dashes and ending with <OK>, that you type for Help to look up. Page 5

up arrow: While using the Help command, you can type up arrow (↑) to go "up" in the structure of Help information. This is useful for a general overview of a particular subject, or to see the context of something you are studying. Page 12

variable: A variable is used to represent a number of alternatives. Often uppercase words in Help are like variables in that they represent the choices you have at a step in a command. Page 9

SOLUTIONS TO EXERCISES

1. There are many ways to find out what you can delete. The simplest and briefest information is available through question mark (?). To use question mark, you would begin a Delete command by typing "d" for "Delete" at "BASE C:" and then type "?". You would see a list of all the things that can be deleted. For more complete information, you could type <HELP> instead of "?". You would then be given the Help definition of the Delete command, which includes information about what you can delete. You could also use the Help command to find out about deleting and what can be deleted. Simply give the Help command by typing "h" for "Help" and type "delete" as the term you want to look up. If you try both typing <HELP> and using the Help command, you will find that they give you the same information.

2. "OPT" as part of a prompt means that at this point in the command you can type <OPT> to request additional choices in the command. After you type <OPT>, you will get a new prompt; at this point you can learn what your additional alternatives are by typing "?".

3. To replace any character with "?", you can give the Replace Character command and specify a question mark as your new character. But to do this you would have to type the question mark at a prompt, which normally lists your alternatives at that point in the command. To have the question mark taken as text rather than as a request for information, you must precede it with <LIT>.

4. After typing "f" for "Force" and "?" to find out what you can "force", you can learn what a "visible" is by typing <CD> to terminate the Force command, "h" to begin the Help command, and "visible" as the term for Help to look up.

5. After asking Help about "Force", you could get more general information about editing commands like Force by typing "↑" at the "C/T:" prompt. This would take you to higher-level information in the Help files and provide you with a context for what you have just read. You could continue to do this until you could not go any higher in the files.

6. Uppercase words in Help's command syntax and command definitions represent choices you have at that point in the command. To continue the command you can select any one of the choices. If you want to find out what a particular uppercase word stands for and it is not explained in the current Help description, take the menu item that mentions it; if there is no menu item that mentions it, type it at the "C/T:" prompt.

7. When you are using Help, you can use "<" to go backward through Help information you have already seen and find a view that you want to see again. When you type "<", you will see the first few words of the previous Help description in your status window. If this is what you want, type "y" or <OK>; otherwise, type "n". Continue back until you reach the information you want.

8. Since Help itself is a command, when you are using Help and want to give another command, you must first type <CD> to cancel the Help command. After doing this and arriving once again at "BASE C:", you can then type "f" for "Force", followed by "v" for "Visible", to give the Force Visible command.

9. If you were in the middle of giving the Force Visible command and you became worried about what was going to happen, you could type <HELP> after any prompt to get a definition of the command as far as you had given it. Or you could cancel the Force Visible command and give the Help command, typing "force visible" as the term for Help to look up. Remember that if you wanted to give the Force Visible command again, you would first have to leave Help by typing <CD>.

10. To find and read this lesson online, you would use LOCATOR. You can reach LOCATOR by giving the Jump (to) Locator command; it will instruct you in how to proceed from there.