Arras User's Manual

TR85-036

1985

John B. Smith

The University of North Carolina at Chapel Hill
Department of Computer Science
Sitterson Hall, 083A
Chapel Hill, NC 27599-3175

A TextLab Report

ARRAS
USER'S MANUAL

John B. Smith

Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina 27514

Starting ARRAS

Note for those *not* using TUCC:

> The ARRAS system is supplied with procedures to start the SCAN program to *Prepare New Texts* and to start the analysis and retrieval system, normally referred to simply as ARRAS. These procedures are described below as they are used at the Triangle Universities Computation Center. Your local computation center may have modified or replaced them to make ARRAS fit into your local environment. *If ARRAS or ARRASCAN does not begin operating as described here, notify your local computation center.*

FIRST TIME

Before you use ARRAS for the first time, you must prepare several files for it to use.

If you don't already have a **$LOGON.CLIST** file, or don't know what this means, you can have the ARRAS set-ups done automatically for you by typing the following command:

```
EXEC 'UNC.CS.F743U.ARRAS.ARRASONE.CLIST'
```

You do this only the first time; in subsequent sessions, proceed to the ARRASCAN and ARRAS steps.

If you already have a **$LOGON.CLIST** file, do the following:

- add a line to **$LOGON.CLIST** by typing
  ```
  QED $LOGON.CLIST
  INPUT
  MYPROC 'UNC.CS.F743U.ARRAS.COMM.CLIST'
  break
  SAVE
  ```

- create a new file, using **QED**, and then **SAVE** it, by typing:
  ```
  QED    $SPOOL    NEW
  type several blanks and a carraige-return
  break
  SAVE
  ```

ARRASCAN

To prepare a new text for ARRAS, use the following command:

ARRASCAN INFILE (filename) **OUTFILE** (filename)

INFILE designates the encoded text (See *Preparing Texts*).

OUTFILE designates the prepared form of the text to be used by ARRAS.

Example:

ARRASCAN INFILE (RAWPOE) **OUTFILE** (POE)

Note: ARRASCAN uses 3:00 minutes as the default time limit. You may increase or decrease that time by adding **TIME** (#) as a third parameter.

Example:

ARRASCAN INFILE (RAWPOE) **OUTFILE** (POE) **TIME** (5)

ARRAS

To start the ARRAS retrieval and analysis system, use the following command:

ARRAS filename

Filename is the name of the text you wish to consider. For the texts that you control, the name will be one designated as the **outfile** *filename* in an earlier **ARRASCAN** command, as shown above.

Example:

ARRAS POE

Note:

ARRAS will use a file in which to save the categories you define and from which to restore them with the name **$CATS.filename**, where **filename** is the name of the text being considered. You may specify an alternative file, for saving and/or restoring categories, by adding to the ARRAS command the parameter, **CATFILE (filename)**, where **filename** is the name of the category file you wish to use instead of the default file.

Example:

ARRAS POE CATFILE (DARKPOE)

Note:

You may use ARRAS with a SAMPLE text that we have prepared by giving the following command:

ARRAS SAMPLE

Contents

Acknowledgments

ARRAS goes way back. I can't acknowledge all those who have helped or contributed, but I wish to recall some of those who were most influential. I see ARRAS as the fifth evolution of a set of ideas and computer procedures that I began while a graduate student at UNC. That work was done under the direction of Sally Y. Sedelow and Weldon Thornton. I am grateful to them both for early guidance and encouragement.

While at Penn State, I profited from the suggestions and technical expertise of many individuals. George Borden and Ken Frandsen offered much needed perspectives as colleagues active in text analysis. Skip Knoble, Bill Verity, Dan Bernitt, Chet Smith, Tom Minsker, and other Computation Center Faculty provided technical assistance, always patiently and good-naturedly, beyond what anyone could reasonably ask for or expect to receive. I am particularly grateful to Russ Miller and Bill Verity for allowing me to include in the system their routine that links ARRAS to CMS. I also wish to thank the many students who used ARRAS and/or listened to and sometimes learned the internal mysteries of inverted file structures and modular system design and provided valuable feedback. In particular, I wish to thank Paul Schuepp for writing the initial version of a number of ARRAS's internal procedures.

ARRAS has been tested in three different locations. As the primary analytic system for the ARTFL Database at the University of Chicago, it has received its most thorough testing by them. I am grateful for the errors they have found and told us about and for their suggestions. I also wish to thank Randall Jones at Brigham-Young and George Logan and Dave Barnard at Queen's University in Canada for their contributions, as well. Since ARRAS is currently being tested at the Triangle Universities Computing Center, other colleagues, I am sure, will make valuable suggestions after this is written.

After coming to UNC, I profited from my work with Steve Weiss, in particular. He has been, and remains, data structures expert, colleague, and friend. I also wish to acknowledge the influence of Fred P. Brooks, Jr., who has taught me, through his example, the meaning of clean, clear, elegant thinking. Since most of ARRAS was fixed in design when I came to UNC, this influence will show more strongly, I hope, in future versions of the system.

I am grateful to Gang Yang for contributing an early draft of the *Tutorial* and to Gordon Ferguson for writing the *Reference Manual*. Gordon has also contributed significantly in late-stage modifications, testing, and final debugging. John Gauch is responsible for regularizing the internal documentation; I wish to thank him for that largely invisible contribution. I also wish to thank Vicki Baker and Rebecca Highsmith for proofreading the manuscript. And Leigh Pittman! She has done yeowoman service in formatting and correcting the many drafts of this document.

Finally, I wish to thank Ian and Catherine. They have listened patiently and encouraged much. They have also put-up with that glazed preoccupation that those who live with programmers know so well. Thank you.

Thank you all.

Chapel Hill
September, 1984

Introduction

Definition

ARRAS — **AR**chive **R**etrieval and **A**nalysis **S**ystem — is a computer system designed to provide fast access to long texts and flexible aids for analyzing them. It can recall a portion of a text, reveal subtle patterns that might be missed or only partially perceived while reading, and help you gain a sense of emphasis and proportion. However, ARRAS itself is not the "analyzer." It is you who decides what information should be retrieved and what the results mean. For example, ARRAS provides very general and powerful facilities for defining and manipulating *categories* (collections of words or locations, discussed later), but you must decide what is to be included in the categories.

ARRAS is designed to be used interactively: you ask ARRAS for information, and ARRAS supplies that information immediately — within a second for most requests. For example, if you want concordance information for a particular word, you ask for it, ARRAS supplies it, you consider it, and then make another request. This rhythm or pattern of use is different from "batch" concordance programs. These programs first construct the complete concordance requested before printing any of it—a process that often takes hours.

System Organization

To provide fast, interactive access to large texts, ARRAS must first prepare the text ahead of time. Most users will never see this process since they will use texts that have already been prepared. But when a specific text is not available, you must know how to encode and process new texts. Detailed rules and procedures are spelled out in the section entitled, *Preparing Texts for ARRAS*

Once a text is prepared, it is stored in a data base or in the computer's file system. To examine a given text, you must direct ARRAS's attention to that particular text. The exact procedure for doing this will differ from installation to installation; but, for now, you should realize that a text selection step is required.

We can now assemble the pieces—printed text, text preparation, data base, text selection, and ARRAS proper—into a *system*. Figure 1 shows this system and the relations among its parts.

Most of this manual is concerned with the analysis portion of ARRAS. In fact, unless a specific step is being discussed, such as preparing a new text, we will use the term, ARRAS, to refer to the analysis portion of the overall system.

$$\begin{tikzcd}
\boxed{\text{Physical Text}} \arrow[r] & \boxed{\text{Encoded Text}} \arrow[r] & \boxed{\text{Text Data Base}} \arrow[d] \\
& & \boxed{\text{Text Select}} \arrow[d] \\
& & \boxed{\text{ARRAS}} \arrow[d] \\
& & \boxed{\text{Terminal}}
\end{tikzcd}$$

Using This Manual

In designing this manual, we have assumed that each reader will go through three stages of development. Consequently, we have described ARRAS in three different ways.

When you first begin to use ARRAS, or any other system, the first thing you need to do is to get a "feel" for it — a general sense of what it does or doesn't do, a sense of the rhythm of use. You also need to build a mental image of the system: "where" a text is located, how ARRAS "views" it, etc. For example, ARRAS permits you to refer to any given word in a text by its *linear* position — the first word is numbered 1, the second, 2, out to 100,000, or whatever. This "view" of a text might be thought of as one long ticker-tape, running left to right, on which the entire text has been printed. But that tape can also be thought to have been "cut" into lines and those lines to have been "pasted" on pages. To lead you through this initial stage of familiarization, we have provided a *Tutorial*.

The *Tutorial* is time oriented. It presumes you know nothing about ARRAS and leads you, step-by-step, through a sample session. It does not attempt to explain ARRAS functions in detail; instead it tries to give you a feel for the system and to explain some of the assumptions that underlie its design. After working through the tutorial several times, you should begin to feel comfortable with ARRAS and begin to see how its capabilities can be used to explore substantive questions in the texts that interest you.

Next comes a grouping of ARRAS commands by *function*. After working through the tutorial and as you begin to use ARRAS, you may recall that ARRAS can do some particular thing, but you may forget the command that makes ARRAS do it. This section groups the command words by what they do—commands that permit you to establish and manipulate categories, commands that search for one thing or another, etc.

The final stage of learning comes after you are familiar with ARRAS. At this time, you may wish to review a command, check a particular detail, or see the full technical specifications of a command. The third section is a *Reference Manual*, arranged alphabetically. Each command is explained in detail and each is explained within a consistant format. First, you will see an example followed by a brief explanation. If you are coming back to a description, rather than reading it for the first time, this may be all the information you need. Next, comes a precise explanation of the syntax of the command and a list of other key words the command is used with. The final portion of the description gives the abbreviation of each key word and detailed notes that explain nuances of use. This organization is intended to provide fast, quick access as well as full technical detail.

One further word about instructions for using ARRAS. ARRAS has built into it a self-instruction or **help** system. Anytime you make a mistake with a command, ARRAS will offer help, unless you have told ARRAS not to. You can also type **help** as a command and get the same results. The details of the help system are explained below.

Tutorial

Introduction

This tutorial assumes that you have a basic understanding of your computer eviron-ment, including either the CMS or TSO systems. If you don't, you may need to ask someone to help you *logon* to the system and get **ARRAS** started. (Normally you just type the word, **arras**, followed by the file name for the text you wish to examine; how-ever, different procedures may be used in different installations. See the section entitled, *Starting ARRAS* at the beginning of this manual.)

The examples included in this manual are taken from James Joyces's *A Portrait of the Artist as a Young Man.* We will describe a dozen or so basic commands. You should first try an identical command and then try several analogous commands. View the experience as an adventure. Since you can't "break" ARRAS, don't be afraid to make mistakes.

After successfully starting **ARRAS**, the first thing you will see is a screen that an-nounces **ARRAS**, much like the title page of a book. The last line shows the word, *command.* **ARRAS** will then pause and wait for you to give it a command.

Before telling you how to make **ARRAS** start doing something, we want to tell you first how to **stop** it. For some commands, **ARRAS** can be a bit like the sorcerer's apprentice—think what would happen, for example, if you asked for a concordance for the word, *the*! It wouldn't really hurt anything if you did, but you can stop **ARRAS** at anytime by using the *break* key.

When you press the *break* key, the system will prompt

```
ATTN:
```

and wait for your response. You have three choices:

C/R (CARRIAGE RETURN)	—CONTINUE THE PROCESSING.
q	—QUIT THE PROCESSING AND AWAIT THE NEXT COMMAND.
stop	—TERMINATE THE SESSION.

So, for example, if you are stuck in the middle of a long concordance request that you don't wish to continue, you can press the *break* key and when you see the ATTN prompt, type *q*. This will *quit* the processing and return you to the command prompt.

General ARRAS Command Format

The ARRAS command format roughly resembles normal English syntax. Each command can be a complete English sentence with a subject, verb, object, etc, but a command can be abbreviated to include only the key words necessary for the system to recognize and carry out the intended action. In fact, most key words can be further abbreviated to three or four letters. For example, to display every occurrence of the word, *fire*, you could type the command

```
PLEASE display A concordance FOR THE WORD: FIRE.
```

But you can shorten this to

```
display concordance: FIRE.
```

or even

```
disp conc: FIRE.
```

In the first example, ARRAS does not understand all of the words. Instead, it picks out only the key words, **display** and **concordance**, and ignores the rest.

The general ARRAS command format is

```
COMMAND [SUB-COMMAND] : PARAMETERS.
```

where

COMMAND	—	the verb specifying the action. e.g., **display**.
SUB-COMMAND	—	the object of the verb, e.g., **concordance**.
:	—	the punctuation mark that separates the command from the parameters.
PARAMETERS	—	the specific word or number the command applies to. e.g., FIRE.
.	—	the punctuation mark that ends the command.

Some commands can have more than one clause; clauses are separated by a semicolon. Usually these additional clauses are optional. For example

```
display concordance: FIRE;

context: -5 TO 5 WORDS.
```

The **concordance** command, unless told otherwise, displays each full sentence in which the parameter word appears; the **context** clause changes this default so that ARRAS will display more or less context.

Note that every command ends with a period. If you type a carriage return after a line without a period, ARRAS assumes that you wish to continue the command on the next line and prompts:

```
CONTINUE:
```

Carriage return after a blank continuation line is interpreted as the end of the command and ARRAS will go off and execute the command.

Basic Commands

help

ARRAS is anxious to help you. You can get help in two ways. The first way is to type **help** as a command (be sure to put a period after it).

```
        help.
```

The system will display a list of all commands for which **help** descriptions are available followed by a prompt for the specific command you are interested in. You can type the command you want to know about and the system will display the format of the command followed by examples. Figure 2 shows a dialogue with **help** in which a request is made for information about the **concordance** command.

```
INTRODUCTION      GENERAL          BREAK            CATEGORY
CHANGE            CONCORDANCE      CONFIGURATION    CONTEXT
CONVERT           CMS              DEFINE           DICTIONARY
DISPLAY           DISTRIBUTION     EXPAND           HELP
INDEX             MODIFY           SAVE             STOP
SET               TEXT

TERM FOR WHICH YOU WISH HELP?  concordance
    FORMAT:
        [sub-command] ... concordance P1, P2, ... PN;
                                 CONTEXT : -N TO +M UNITS.
    EXAMPLES:
            DISPLAY CONCORDANCE   :fire.
            DISPLAY CONCORDANCE   :firecat;
                    CONTEXT :-1 to +1 SENTENCES.
    CONTINUE?: no
```

(Figure 2)

After displaying the format and the examples, ARRAS will ask you if you want to continue the description. If you respond with **yes**, ARRAS will display a narrative explanation and again ask you if you want more information. A second **yes** will produce a list of all available options for that particular command. A **no** or a carriage return in response to the CONTINUE prompt returns you to the command prompt.

The second way in which ARRAS will offer help is when a command is used incorrectly. When ARRAS finds an error in your command, it will prompt:

```
ERROR IN LAST COMMAND

DO YOU NEED HELP?:
```

Type **yes** to get help or **no** if you don't want help. If you type **yes**, the help system will be invoked as if you had typed **help** as a command, just discussed above.

If you decline ARRAS's offer of help after a command error—by typing **no** to the prompt—ARRAS will ask you if you wish to modify the command statement. If you type **yes**, the system will display the erroneous command followed by the prompt

```
CHANGE:
```

Type the incorrect part of your command; then the system will prompt

```
TO:
```

Type a correction to replace the incorrect portion you just typed. Since there may be more than one error in a command, ARRAS will prompt for additional changes. When you have finished making all the changes you need to, type **no** to the prompt for additional changes. ARRAS will then attempt to execute the command as modified. For example, suppose you typed the erroneous command

desplay dectionary: FIRE.

Figure 3 shows a dialogue in which the two misspellings are corrected.

```
ERROR IN LAST COMMAND

DO YOU WANT HELP?: no

DO YOU WANT TO EDIT?: yes

DESPLEY DICTIONARY: FIRE.

CHANGE: desplay

TO: display

MORE CHANGES?: yes

CHANGE: dectionary

TO: dictionary

MORE CHANGES?: no
```

(FIGURE 3)

Display

The most frequently used ARRAS command is **display**. **Display** displays the information requested on the terminal screen. There are several important sub-commands that are used with **display**. Let's start with the simplest one.

Dictionary. ARRAS keeps an alphabetic list, or **dictionary**, of every word (without duplications) that appears in the text. If you want to examine the vocabulary of the text, type

```
display dictionary.
```

The system will display all the words in the dictionary along with their frequencies.

More often, you will want to see whether a given word occurs or to check the frequency of occurrence for a given word. To do this, add the word as a parameter. For example,

```
disp dict: FIRE.
```

The system will display

```
56 FIRE
```

where 56 is the number of times the word *fire* occurs in the text.

A third form of the command lets you to see all the words and their frequencies for a specified alphabetic range. For example,

```
disp dict: FIRE  -  FIREZ.
```

will display all words that begin with the characters, *fire*. It produces the following:

```
DICTIONARY FREQUENCY AND WORD:
56 FIRE           1 FIREARMS        1 FIRECONSUMED
 2 FIRED          1 FIREEATER       3 FIRELIGHT
 7 FIREPLACE      5 FIRES           1 FIRESHOVEL
```

(FIGURE 4)

Concordance. If you would next like to see the contexts for a given word in the text, use the **concordance** sub-command. ARRAS will display each full sentence in which the word occurs. For example, after you use **display dictionary** to find out that the word *fire* occurs 56 times in the text, you may want to know in what contexts *fire* occurs. To do this, type

```
disp conc: FIRE.
```

Figure 5 shows a partial list of the results:

```
CONCORDANCE FOR THE WORD, FIRE
 FIRE          IT WOULD BE NICE TO LIE ON THE HEARTHRUG BEFORE THE
               FIRE, LEANING HIS HEAD UPON HIS HANDS, AND THINK ON THOSE
               SENTENCES.
               1369: P. 4
 FIRE          MOTHER WAS SITTING AT THE FIRE WITH DANTE WAITING FOR
               BRIGID TO BRING IN THE TEA.
               1369: P. 4
 FIRE          IT WOULD BE LOVELY TO SLEEP FOR ONE NIGHT IN THAT COT-
               TAGE BEFORE THE FIRE OF SMOKING TURF, IN THE DARK LIT
               BY THE FIRE, IN THE WARM DARK, BEATHING THE SMELL OF
               THE PEASANTS, AIR AND RAIN AND TURF AND CORDUROY.
               4485: P. 12
 FIRE          IT WOULD BE LOVELY TO SLEEP FOR ONE NIGHT IN THAT COT-
               TAGE BEFORE THE FIRE OF SMOKING TURF, IN THE DARK LIT
               BY THE FIRE, IN THE WARM DARK, BREATHING THE SMELL OF
               THE PEASANTS, AIR AND RAIN AND TURF AND CORDUROY.
               4496: P. 12
```

(Figure 5)

On the left, ARRAS displays the word for which the concordance is requested; on the right, ARRAS displays the contexts in which *fire* occurs. After each sentence are the linear number and the page number for that occurrence.

Sometimes you may want to see more or less context than the default context of one full sentence. To specify a different context, use **display concordance** together with the **context** sub-command. The **context** sub-command allows you to specify the number of preceding (negative) and the number of succeeding (positive) units (*words*, *sentences*, *paragraphs*, *chapters*, *volumes*, *lines*, or *pages*). Here are some examples:

```
disp conc: FIRE;

context: -5 TO +7 WORDS.
```

displays every occurrence of the word, *fire*, together with the five words preceding and the seven words succeeding it.

```
disp conc: FIRE;

context: -0 TO 0 PARAGRAPH.
```

will display every occurrence of the word *fire* together with the paragraph in which it occurs (i.e. no other preceding or succeeding paragraphs).

Question: what do you think a context of -1 TO +1 PARAGRAPHS will produce?

Text. While **display concordance** will display the context for each occurrence of a given word, sometimes you may be interested in only one instance of the word. ARRAS provides the sub-command, **text**, to do this. **Display text** takes the linear number of a word and a specified context, and displays the desired portion of the text. For example, after displaying a small context for every occurrence of the word *enflaming* with the **concordance** option, you may wish to see a larger context for the instance numbered 92546. To do this, type

```
disp text: 92546, -2 TO +3 SENTENCES.
```

Figure 6 shows the system display

```
TEXT FOR LINEAR WORD 92546 (ENFLAMING): 92531 TO 92566
      THE IMAGES HE HAD SUMMONED GAVE HIM NO PLEASURE.
      THEY WERE SECRET AND ENFLAMING BUT HER IMAGE WAS
      NOT ENTANGLED BY THEM. THAT WAS NOT THE WAY TO
      THINK OF HER. IT WAS NOT EVEN THE WAY IN WHICH
      HE THOUGHT OF HER. COULD HIS MIND THEN NOT TRUST
      ITSELF?
```

(FIGURE 6)

Distribution. After examining individual occurrences of a word, you can gain a sense of proportion and emphasis for that word over the text by using the sub-command **distribution**. It produces a bar graph: the horizontal axis represents the text extending left to right from the beginning to the end, divided into 50 segments; the vertical axis indicates the number of occurrences for the word, within each text segment. The graph in Figure 7 shows the distribution for the word, *fire*.

```
FREQ. 20    |
            |
            |
            |                      X
            |                      X
            |                      X
            |                      X
            |                      X
            |                      X
            |                      X
FREQ. 10    |                      X
            |                      X
            |                      X
            |                      X            X
            |                      X            X
            |                      X            X
            |    X                 X            X
            |  X X      XX         X     X      X
            |X XXX      XX         XXX  XX      X
            |X XXXX  X  XX      XXXXXX XXX      XX        XX
            +++++++++++++++++++++++++++++++++++++++++++++++++++
             2%                       50%                      100%
                          DISTRIBUTION: FIRE
```

(Figure 7)

Advanced Commands

Categories

So far, ARRAS commands have operated on single words in the text. However, you may wish to use categories or groups of words in some instances instead of single words. For example, in a study involving themes or concepts, groups of synonyms may be a more natural basis for the study than individual words. One of ARRAS' most powerful features is its ability to handle such categories.

A category is a set. It can be a collection of words, text locations, or other categories. Normally, different types of data are not mixed in the same category. To maintain this distinction, ARRAS recognizes two major kinds of categories, *dictionary categories* and *linear categories.*

Dictionary Categories. A dictionary category is a set of words from the text. For example, you can categorize the words ***enflaming, fire, fireconsumed, fires, flame, flamed, flames, flaming, heat, heated, hot, hotly***, and name it FIRECAT.

To define a dictionary category, you use the command, **define**, followed by the sub-command **category** (default is dictionary category), followed by the words that are to be included in the category. In order to refer to the category later, you must give it a name by using the **name** sub-command. As an example, let's define the category mentioned above.

```
define category:   ENFLAMING, FIRE, FIRECONSUMED, FIRES,
                   FLAME, FLAMED, FLAMES, FLAMING, HEAT,
                   HEATED, HOT, HOTLY;
           name:   FIRECAT.
```

Linear Categories. A linear category is a set of linear numbers in the text. Each number refers to a specific instance of a word. (These numbers appear directly after the context displayed by the **concordance** command along with the page number for that instance). Thus, you can use the linear category to distinguish among different uses of a word. For example, after looking at the concordance for *fire*, you could separate the noun *fire* from the verb *fire* by defining two different linear categories.

To define a linear category, add the sub-command, **linear**, to the **define category** command. For example, to define the linear category ***nounfire*** containing the locations 1281,1367,4485,4496 where the word *fire* is used as a noun, type

```
def linear cat: 1281,1369,4485,4496;
```

name: NOUNFIRE.

Categories of Categories Actually, there is a third kind of category — one in which the members are not words or linear numbers but, rather, other categories. Consider the following example.

Suppose that in addition to *firecat*, you had defined a category, *watercat*, that consists of the words — *water, watery, wet, wetness, damp, dampness* — and similar categories, *earthcat*, and *aircat*. You may use these categories individually; but you may also wish to collect them, for some considerations, into a category, *elementscat*, composed of members *firecat, watercat, earthcat*, and *aircat*. The command to do so is

define cat: FIRECAT, WATERCAT, EARTHCAT, AIRCAT;

name: ELEMENTSCAT.

Reference to *elementscat* now refers to the contents of all four categories.

If you think of simple dictionary and linear categories as "level 1" categories, you may wish to think of a category whose members are level 1 categories as a "level 2" category. However, the basic notion of category is recursive; you can build level 3 categories whose members are level 2 categories, etc. Thus, you may view a system of categories as a hierachy.

This capability is quite general and quite flexible, so you may need to practice using it and to give it some thought in order to relate it to the problems and questions that interest you.

Using Categories

As we mentioned at the beginning, categories can be used in ARRAS commands in most of the places where a text word is used. You can **display** a category, examine **concordances** of the words or locations contained in it, and display a cumulative **distribution** for it.

Display Categories

First, to display the contents of a category, use the command **display category** followed by the category name. The system will show the name of the category (CATEGORY), the numbers of distinct words (TYPES), their total, cumulative frequency (TOKENS), the kind of category (DICTIONARY, LINEAR, etc.), followed by the actual members. For example,

display cat: FIRECAT.

produces the display shown in Figure 8.

```
CURRENT CATEGORIES ARE DEFINED AS FOLLOWS:
CATEGORY     MEMBERS       TYPES        TOKENS       KIND
FIRECAT                    12           125          DICTIONARY
             ENFLAMING
             FIRE
             FIRECONSUMED
             FIRES
             FLAME
             FLAMED
             FLAMES
             FLAMING
             HEAT
             HEATED
             HOT
             HOTLY
```

(FIGURE 8)

If the category is a higher level category (i.e. a category of categories), ARRAS will put an asterisk "*" before the category member to indicate that this element is a category. Figure 9 shows an example of a display of a higher level category.

```
CURRENT CATEGORIES ARE DEFINED AS FOLLOWS:
CATEGORY   MEMBERS    TYPES      TOKENS     KIND
FIRECAT               2          1          DICTIONARY
           *WATERCAT
           *FIRECAT
```

(FIGURE 9)

Finally,

display category.

will display all categories in the order in which they were defined. If you wish to see only summary information, not the list of members, add the sub-command **names:**

display category names.

This will display summary information for each category in the order in which they were defined.

Concordance of Categories

Next, you can display a concordance for the words or locations of a category. For a dictionary category, this is equivalent to displaying a concordance for each word of the category; for a linear category, this will produce a concordance for each text location contained in the category. For example,

display conc: FIRECAT.

```
CONCORDANCE FOR THE LINEAR CATEGORY, FIRECAT WITHOUT VALUES
 ENFLAMING  THEY WERE SECRET AND ENFLAMING BUT HER IMAGE WAS NOT
            ENTANGLED BY THEM.
            92546: P. 232
 FIRE       IT WOULD BE NICE TO LIE ON THE HEARTHRUG BEFORE THE
            FIRE, LEANING HIS HEAD UPON HIS HANDS, AND THINK ON THOSE
            SENTENCES.
            1281: P. 4
 FIRE       MOTHER WAS SITTING AT THE FIRE WITH DANTE WAITING FOR
            BRIGID TO BRING IN THE TEA.
            1369: P. 4
 FIRE       IT WOULD BE LOVELY TO SLEEP FOR ONE NIGHT IN THAT COT-
            TAGE BEFORE THE FIRE OF SMOKING TURF, IN THE DARK LIT
            BY THE FIRE, IN THE WARM DARK, BREATHING THE SMELL OF
            THE PEASANTS, AIR AND RAIN AND TURF AND CORDUROY.
            4485: P. 12
```

(Figure 10)

will show the contexts for all words contained in *firecat*. (See Figure 10)

Similarly,

display conc: NOUNFIRE.

will display a concordance for those specific occurences of *fire* that were nouns.

Distribution of Categories

You can also display a distribution for a category in which each word or location in the category will be accumulated to form a single distribution of the entire category. For example,

display dist: FIRECAT.

produces

```
FREQ. 20      |                      *
              |                      *
              |                      *
              |                      *
              |                      *
              |                      *
              |                      *
              |                      *
              |                      *
              |                      *
FREQ. 10      |                      *
              |                      *
              |  X                   *
              |  X                   *            X
              | X X                  *            X
              | X X X                *XX          X
              | X X X   X  X         *XX      X   X
              |XXXXX    X  XX        *XX  XXX X   X  X      X
              |XXXXXX   X  XX       X*XX XXXX XX  XX XX XX  X
              |XXXXXXX  X  XX  X  XXXX*XX XXXX XXX XXXXX XX XXX
              ++++++++++++++++++++++++++++++++++++++++++++++++-+
               2%                     50%                      100%
                           DISTRIBUTION: FIRECAT
```

Save/Retrieve/Modify/Convert Categories.

There are still several other commands related to categories. For example, once a group of categories is defined, you can **save** them at the end of a session and retrieve them at the beginning of a subsequent session. To save your categories, type

save categories.

This command will save a copy of all your current categories. In the later session, type

define category: $input.

to retrieve the saved categories. At that point, all the categories from the previous session will be re-established.

(TSO and CMS handle categories differently. After you are comfortable with the

Tutorial, you should read the technical discussion for DEFINE in the ARRAS *Reference Manual*, below, before making extensive use of ARRAS's category storage facilities.)

You can also modify the contents of a category — add members, delete members, change the name of the category and convert a dictionary category to a linear category.

Configurations

Configuration is a particularly powerful function that operates on words or categories. If you have little background in math or logic, it will seem a complicated function at first. But its flexibility and usefulness will repay your hard work.

Configuration searches for a contextual pattern of words or categories and produces a linear category of the locations where the pattern appears. For example, you can search for all places in the text where *both* fire and water occur in the same sentence.

ARRAS uses one full sentence as the default context in which to search, although this may be changed in two ways, as we will explain a moment. The **configuration** command always applies to two or more words or categories that are related to one another by an operator. ARRAS recognizes three operators: **and**, **or**, and **not**. We will explain each operator separately, show an example for each, and then discuss how to combine them to form more complex expressions. We will discuss them in terms of words; but whenever a word is mentioned, a category name could be used. More about this later.

And — denoted by &

And takes two words and constructs a linear category that contains only those locations where *both* words occur in the same sentence. Take *fire* and *water* as an example,

configuration: FIRE & WATER;

name: FFWW.

This command constructs a linear category, *ffww*, consisting of the locations where both *fire* and *water* occur in the same sentence. For an **and** expression, ARRAS will select the left word of the pair as the "target" so that the linear number identifying a sentence where both appear will always be that of the target word. Thus, the result of an **and** configuration will be a subset of the locations for the left word.

And not — denoted by &¯ or in some installations, & ^

And not takes two words and constructs the linear category that contains those locations where the first word occurs but where the second word does **not** occur within the same sentence. The command

```
configuration: FIRE &~ WATER;

name: FF.
```

produces the linear category, *ff*, which contains the locations where within the same sentence *fire* occurs but *water* does **not** occur. Again, the results will be a subset of the left word of the expression.

Or — denoted by |

Or takes two words or categories and constructs a linear category of the locations where *either* of the two words occurs. For example,

```
configuration: FIRE | WATER; name: FW.
```

constructs the linear category, *fw*, which contains the locations where either *fire* or *water* occurs. If *fire* and *water* are categories, the *fw* will contain all the locations at which any member of either *fire* or *water* occurs. In fact, **or** is logically equivalent to defining a category consisting of the two words or categories.

Contexts and Combinations

Having learned each individual configuration operation, we can now combine them to form more complex expressions.

Each of the above expressions was evaluated within a context of one full sentence. This is the default. You may change the default context by adding a **context** clause to the **configuration** command. The unit of the context can be defined in terms of *words, sentences, paragraphs, chapters, volumes, lines,* and *pages*. For example,

```
configuration: FIRECAT & WATERCAT;

context: -50 TO +50 WORDS;

name: FWCAT.
```

In this example, ARRAS will go to each occurrence of each word in *firecat*, see if there is any word from the *watercat* within 50 words, to the left or the right, and if so, add the location of that particular *firecat* word to the linear category, *fwcat*, being formed.

```
configuration: FIRECAT & (WATERCAT | COLDCAT);

name: MIXCAT.
```

ARRAS will look for the contexts where a *firecat* word occurs near a word from either the *watercat* or the *coldcat*. Note the use of parentheses: they may be used freely to indicate logical combinations that should be used as elements relative to the other word or category elements in the expression.

You may also change the context used to evaluate only part of the total search expression. This is done by placing a context specification, enclosed in square braces, immediately after an **and** operator (**context** has no meaning with a logical **or**). The format of the "local" context specification can be inferred from the following example,

```
conf: (FIRECAT & [-5 TO +5 WORDS] RELIGIONCAT)

    & WATERCAT;

name: FRWCAT.
```

This expression tells ARRAS to go to each word location in *firecat* and see if any word from the *religioncat* occurs within five words of that location; if so, then see if any word from the *watercat* occurs in the same sentence (the default "global" context for the expression). One further note: context specifications, local or global , can define contexts that lie entirely to the "left" of a word (-10 to -5) or entirely to the right (+1 to +10).

Finally, ARRAS will let you use linear categories in search expressions as well as words or dictionary categories. This feature becomes particularly powerful, and abstract, when you consider that the results of a **configuration** is a *linear category*. Thus, you may search, first, for patterns defined in terms of words. After doing a number of such **configurations**, you can then combine the linear categories produced by this first series of searches into second level configurations: patterns defined in terms of elements which are, themselves, patterns. Such patterns might represent sentence-level patterns. Then, you could look for paragraph-level patterns, defined in terms of sentence-level patterns, ... The process can be repeated, and raised hierarchically, as high as you wish to go.

To a great extent, an analysis of a text becomes the development of an evolving hierarchy of categories and/or patterns.

Functional Groups

Introduction

Learning commands by functional groups marks a second stage in learning to use ARRAS. In the *Tutorial*, commands are presented in a time-oriented sequence to enable you to gain a sense of the "rhythm of use" for ARRAS; in the *Reference Manual*, commands are presented in alphabetical order to enable you to get the information you need quickly and easily. In this section commands are discussed in terms of three groupings according to function:

- **retrieval and analysis**
- **category manipulation**
- **environment control.**

The strategy of presentation is from basic commands (e.g., displaying a concordance) to more subtle commands (e.g., turning off the automatic HELP facility). Emphasis is placed on the inter-relations among commands. We have attempted in several instances to explain the rationale behind ARRAS's design. With an intuitive understanding of this design philosophy, you will often be able to anticipate how functions work and you will be better able to use them in concert to achieve your larger intellectual objectives.

List of ARRAS Commands by Function

Retrieval and Analysis

CONFIGURATION
CONCORDANCE
CONTEXT
DICTIONARY
DISPLAY
DISTRIBUTION
INDEX
TEXT

Category Manipulation

ADD
CATEGORY
CONVERT
DEFINE
DELETE
EXPAND
FILE
LINEAR
MODIFY
NAME
SAVE

Environment Control

BREAK KEY
CHANGE
CMS
EDIT
HELP
POETRY
SCREEN
SET
SPOOL
STOP

Retrieval and Analysis

Retrieval and Analysis is what ARRAS is all about—hence its name **AR**chive **R**etrieval and **A**nalysis **S**ystem. ARRAS's most basic objective is to provide fast, flexible, access to long texts. In fact, you should see no difference in ARRAS's response regardless of the length of the text you are exploring (except, of course, relative to the number of tokens involved in the particluar command—it takes longer to do a concordance for a word that occurs one hundred times than for one that occurs ten times, whatever the overall length of the text). To achieve this objective, ARRAS requires that the text be prepared ahead of time, as explained in the section entitled *Preparing Texts for ARRAS*.

Once a text has been prepared, you can perform six basic retrieval and analysis operations:

- **DISPLAY DICTIONARY**
- **DISPLAY CONCORDANCE**
- **DISPLAY INDEX**
- **DISPLAY TEXT**
- **DISPLAY DISTRIBUTION**
- **CONFIGURATION**

Notice that most retrieval and analysis commands are used with the key word, **DISPLAY**. When a particular retrieval is requested (e.g., a retrieval from the **DICTIONARY** for the text being considered), the results are displayed *immediately*. The one major exception is **CONFIGURATION**, which produces a category of locations where the search pattern occurs. That category, however, can be **DISPLAY**ed using most of the other commands in this group. (We will explain this in more detail, below.)

DISPLAY

The **DISPLAY** command is used as a verb with one of the following direct objects: **DICTIONARY**, **CONCORDANCE**, **INDEX**, **TEXT**, and **DISTRIBUTION**. It is *not* used with **CONFIGURATION**. If you wish to obtain a *printed* copy of what is **DISPLAY**ed, see the **SET SPOOL** command under the *Environment Control* group or in the *Reference Manual*.

DICTIONARY

DISPLAY DICTIONARY shows the words that occur in the text. The results are a *type* list (each unique configuration of characters appears once in the dictionary) with each word type accompanied by its frequency (the number of *tokens*, or repetitions) in the text. The command can be used to see whether a given word occurs or not, to see the words that occur within a given alphabetic range (ab – ac produces a list of all words beginning with the letters ab or ac), or to see the entire vocabulary for the text. This command is particularly useful for developing categories, such as themes or concept groups.

CONCORDANCE

DISPLAY CONCORDANCE shows each occurrence (token) of a word (type) in its textual context. The command may also be used with **CATEGORIES**. The default context is one full sentence; however, you may request more or less context using an (optional) **CONTEXT** clause. Accompanying each sentence (or other context) are the word-token's page number and linear number (sequential position in the complete text).

The linear number is particularly useful in conjunction with the **DISPLAY TEXT** command, which permits you to display contextual information for a single occurrence or position in the text (i.e., a single linear number). Thus, you may wish to use a rather narrow context on the **CONCORDANCE** command for speed (say, -5 to +5 words) and then use the **DISPLAY TEXT** command to extend the context for only those places that interest you.

INDEX

DISPLAY INDEX shows the numeric context (linear number and page number) for each occurrence of a word. The information displayed is that produced at the end of a **DISPLAY CONCORDANCE** command, but without the textual context.

It is useful for seeing quickly and precisely where words occur. You may then examine any given context by using the numeric **INDEX** information as parameters for a **DISPLAY TEXT** command. It is also useful for seeing the precise locations of words for which graphic distributions are displayed.

TEXT

DISPLAY TEXT shows the text focused at a specific linear number but including whatever context to the left and/or right you indicate. When used with *only* a linear number, ARRAS begins there and displays the text (potentially) to the end. (You can, of course, **BREAK** or interrupt the display.) With no linear numbers given, it displays (potentially) the entire text.

DISPLAY TEXT is particularly useful in conjunction with **DISPLAY INDEX** and **DISPLAY CONCORDANCE** commands to move around the text quickly and easily.

DISTRIBUTION

DISPLAY DISTRIBUTION produces a bar graph showing where over a text a word or category occurs. It shows fifty columns of x's or *'s. Each column represents 2% of the text or 4–5 pages in the average novel. X's indicate the number of times the word or category occurs in each text interval. *'s indicate that the word or category occurs more than twenty times for that particular interval. To determine the precise number of times a word or category occurs in this last case, use the **DISPLAY INDEX** command.

DISPLAY DISTRIBUTION is helpful in getting an overall sense of the author's use of a word or category. You can see at a glance relative density. Words or categories that are uniformly distributed over a text are often part of the "background" while words that cluster strongly in places are often part of the "foreground." Such places can, in turn, be examined more closely, using the **CONCORDANCE** and **TEXT** commands, to see *precisely* what is happening there.

Category Manipulation

ARRAS's category-handling capability is one of its most important features. You can use categories to identify syntactic classes of words, thematic or conceptual clusters, positions in the text where certain intrinsic or extrinsic features occur, and many other applications. In general, you may use a category name in a command wherever you can use a text word.

A category is a set—a set of word types, a set of text locations (linear numbers), or a set of other categories. A category consisting of words is called a **DICTIONARY** category; a category consisting of linear numbers is called a **LINEAR** category. A category composed of other categories has no special name associated with it. A category is identified by a **NAME** that you assign it. You refer to the category by referring to its name; consequently, you should give it a name that is *not* a word in the text.

Categories are processed recursively. Suppose you ask for a concordance of a dictionary category: ARRAS constructs and displays the concordance for the first word; when it is finished with all occurrences for that word, it does the same for the second word; etc. If the concordance is for a category of categories, ARRAS performs the above sequence for the first *category*, then the second category, etc.

In general, categories may be thought to form a hierarchy. Categories consisting of words or linear numbers may be referred to as Level 1 categories. Categories whose members are Level 1 categories may be referred to as Level 2 categories. Categories of Level 2 categories would be Level 3, etc. To a great extent, many studies when viewed functionally will be concerned largely with defining and managing an evolving hierarchy of categories.

ARRAS category functions can be divided into three basic groups:

- **functions to define categories**
- **functions to modify categories**
- **functions to store and retrieve categories.**

Functions to Define Categories

DEFINE CATEGORY is the basic command to establish a new category. **DICTIONARY** is the default type of category and all parameters will be interpreted as words. A word can be *fire*, or *water*, but it could also be *1984* (e.g., a date or the title of Orwell's novel). **LINEAR** must be added as a keyword to the command if the category is to consist of a set of text locations. Thus, *1984* as a parameter for a *linear category* means the word after the 1983rd word in the text, not a date or title mentioned somewhere in the text.

Since categories are identified by **NAME**, you should add a **NAME** clause to the **DEFINE CATEGORY** command. If you don't, ARRAS will ask for one.

Several other options may be used with the **DEFINE CATEGORY** command; however, since they relate to retrieving and re-establishing **SAVE**d categories, we will discuss these functions below.

Functions to Modify Categories

MODIFY CATEGORY is the basic command for altering a category. With it you may **ADD** new members to an existing category, **DELETE** members from an existing category, or change the **NAME** of an existing category.

You may also perform several of these operations in the same command. That is, you may *both* **ADD** and **DELETE** items from a category in one command composed of several clauses. When you **ADD** and/or **DELETE** items without using the **NAME** option, the contents of the category referred to by the category name are changed. If you add a **NAME** clause, the old category will be deleted and a new, modified category will be created that can be referred to by the new name given in the **NAME** clause.

A **MODIFY CATEGORY** command that contains *only* a **DELETE** clause with *no* individual words or linear numbers will delete the entire category. Similarly, a **MODIFY CATEGORY** command with only a **NAME** clause will rename the category.

CONVERT permits you to change a **DICTIONARY** category to a **LINEAR** category. For most commands, such a change will be transparent. The exception is for **CONCORDANCE** or **INDEX** commands. For a **DICTIONARY** category, ARRAS constructs a concordance for the first word, listing each sentence or whatever context is requested, from the beginning of the text to the end; then it goes back and does the same for the second word, etc., until all words and contexts have been shown. By **CONVERT**ing the category to **LINEAR** form, you may obtain a single, inter-leaved concordance in which the first occurrence of whatever word in the category appears first in the text also appears first in the concordance, then the second word, etc., until the end of the text.

EXPAND permits you to change Level 2, or higher, categories (categories containing other categories as members) into Level 1 categories. ARRAS does this by rounding up all the individual words or linear numbers in the categories and creating a new category consisting of them. Any duplicate words that may have been included in the constituent categories will be eliminated in the new category. Again, except for **DISPLAY CONCORDANCE**, **DISPLAY INDEX**, and **DISPLAY CATEGORY** commands, the level of the category should be transparent.

Functions to Store and Retrieve Categories

Categories can be **SAVE**d and restored within a session or between sessions. The options and capabilities under CMS are more extensive than those under TSO. However, since the TSO options are, essentially, a subset of the CMS options, we will describe that common group of functions first and then describe the additional features available under CMS.

Categories can be **SAVE**d using the **SAVE** command much as you save a text file when using a text editor. The primary difference is that under TSO you have access to only a single external file into which you may **SAVE** categories. You may save all categories (**SAVE CATEGORIES**) or you may save only selected categories which you list as parameters. When you save categories, however, the new group replaces whatever group was previously saved in the file.

Categories may be restored using the **DEFINE CATEGORY** command by specifying **$INPUT** as the only parameter. ARRAS will ask you if you wish to clear the categories currently in effect; if you respond **yes**, the categories previously **SAVE**d will become active and any new categories established previously in the session will be lost. If you wish to add to the group of categories currently in effect those previously **SAVE**d, a **no** to the clear prompt will do this.

Under TSO the name of the category file will be **$CATS**, unless changed by your local installation. Under CMS, the default category file name will be **$CATS DATA A**, unless changed by your local installation.

Under CMS, however, you may dynamically change the name of the category file. Thus, you may divide your categories into logical groups saved in different files or into groups that apply to different texts. In effect, you can name the file into which categories are saved and from which they are restored in a manner analogous to the way text editors use file names.

Under CMS, a file may be named in both the **SAVE** and the **DEFINE** commands by adding an optional **FILE** clause. One word of caution, however. If you use the **FILE** option, that file name will become the default category file for all subsequent **SAVE**'s and restores, unless you add a **FILE** clause to subsequent commands. But, of course, adding such a clause will change the default to *that* file name! To restore the default to its original value, use the **FILE** name, **$CATS DATA A**.

One final word about categories. In some cases you may wish to use categories that are larger than or independent of a given text; for example, applying a common set of themes to a number of different texts for purposes of comparison. Since ARRAS checks to see that words **DEFINE**d as a category do, in fact, occur in the text under consideration, such general categories cannot be established directly. They can, however, be established indirectly. The external format for ARRAS categories is quite simple. Go into the CMS or TSO environments, edit the category file, and note the format. Then using that editor, you can extend any category or add analogous new categories. When ARRAS later attempts to restore these modified or additional categories, it will do so for whatever words, in fact, do occur in the particular text under consideration. Words that don't occur in the text will be displayed on the screen but will cause no problem.

Environment Control

The last functional group of commands permits you to alter, in one way or another, the general working environment ARRAS provides. It is a rather disparate collection which we will discuss in terms of four subgroups:

- **stopping ARRAS**
- **setting default conditions**
- **changing texts**
- **entering and leaving CMS**

Stopping ARRAS

You may stop ARRAS in two ways: temporarily and "permanently."

The **BREAK** or **ATTENTION** key on your terminal will interrupt ARRAS temporarily. On many systems, ARRAS will display information, such as a concordance listing, at a rate of about four lines a second. This is faster than most people can read. Consequently, you can use the **BREAK** key to interrupt the display and later press the **carriage-return** to resume. Typing a **q**, for **QUIT**, discontinues the listing and returns you to the **COMMAND** prompt while a response of **STOP** discontinues the entire ARRAS session and returns you to TSO or CMS.

STOP typed as a command has the same effect as **STOP** typed after the **BREAK** interrupt—it discontinues the entire ARRAS session and returns you to TSO or CMS. There you can do all the things those general systems support, including starting another ARRAS session.

Before you **STOP** ARRAS, you should be sure you are ready to do so. The most important consideration is to be sure that you have **SAVE**d your categories if you have changed them or added to them during the session (assuming, of course, you wish to keep all such changes or additions).

Setting Default Conditions

ARRAS is designed to be a powerful tool for the experienced user while an easy tool to learn for the novice. Part of this flexibility comes from building into ARRAS a number of default values and conditions, such as the full sentence default context for **CONCORDANCE** or the automatic prompt offering **HELP** when a command syntax error is detected. However, these built-in assumptions may not always meet your needs and may become annoying as you become more familiar with ARRAS. You can change some of the basic ones with the **SET** command.

SET can be used in several ways. First, **SET**, works as a toggle: **SET** it once and it will be set to the opposite of what it is now (*on* goes to *off*); **SET** it again, and the option reverts to the original default. Consider the **HELP** option.

SET HELP turns *on* or *off* the automatic **HELP** prompt that occurs after ARRAS detects an error in a command. You can explicitly say *on* or *off*, as the parameter, but without the option specified, the **SET HELP** command switches the automatic **HELP** facility to the opposite *on/off* position, as just explained.

The default for *help* is *on*. The reason is that when you initiate ARRAS anything you type other than the **BREAK** key but especially the **carriage-return** will take you (eventually) to the prompt for **HELP?** A response of *yes*, *ok*, or **carriage-return** produces a list of **HELP** discussions. You can then read an *Introduction* that gives an overview of ARRAS, a description of ARRAS's command language, or descriptions of major command terms. Thus, you don't have to have separate printed instructions (e.g., this manual) to begin using ARRAS; it is self-instructional for the new user or the user at a remote site.

SET EDIT turns *on* or *off* the second set of prompts after ARRAS detects an error in a command. When **EDIT** mode is *on*, ARRAS asks you if you wish to *edit* the last command. If you respond affirmatively, ARRAS displays the incorrect command and prompts for **CHANGE:**. You can then type a unique sequence of characters that is part of the erroneous command and that includes the error. When ARRAS responds **TO:**, you can type a sequence of characters to replace the sequence you just indicated and which includes a correction for the error. Since a command could have several errors, ARRAS will ask if you wish to make more than one editorial change.

The **EDIT** facility is useful when you make a mistake in a long command, but may be more trouble than it's worth after you become thoroughly familiar with ARRAS's command language.

ARRAS offers two commands that affect the display: **SET SCREEN** and **SET POETRY**.

SET SCREEN lets you display information either continuously (the default) or in blocks suitable to fill your screen. If you toggle the **SET SCREEN**, or explicitly set it to *on*, ARRAS will display twenty-four lines of text and then **PAUSE**. A **carriage-return** continues the listing to produce the next screen-sized block. When in page or block mode, you can toggle back to continuous display or add the explicit **OFF** to the command.

Finally, should you wish to display information in blocks other than 24 line units, you may add that number as a parameter. However, if you indicate fewer than twenty-four lines, this will distort the bar graph produced by **DISPLAY DISTRIBUTION**.

SET POETRY instructs ARRAS to display text in a line by line format identical to that of the text as originally prepared for ARRAS. Normally, ARRAS does not maintain original line format since it presumes such boundaries are "accidents" of printing; this presumption, of course, is not true for poetry and may not be true for other specialized texts. You can toggle **POETRY** (original line preservation) mode or you can add an explicit *on* or *off* parameter.

When you are preparing texts for which you know you will use ARRAS's **POETRY** mode, be sure to review ARRAS's encoding rules that pertain to line representation (see the section entitled, *Preparing Texts for ARRAS*).

SET SPOOL provides a means to print the information displayed on the screen or to otherwise manipulate it within TSO or CMS. It does this by making a copy of the information as it is displayed and putting it in either a TSO file, called **$SPOOL**, or a CMS file, called **$SPOOL DATA A**, unless changed by your installation. After you leave ARRAS (by **STOP**ping it or, in CMS, using the CMS option, explained below), you can review that file, add to it or add it to another file, print it out, or do anything you can normally do to files in your TSO or CMS environment.

SET SPOOL also works as a toggle or it can be explicitly turned *on* or *off*. It uses the IBM MOD option which means that if you turn it on for a while, turn it off, and later turn it back on, the second batch of information will be added to the bottom of the preceding batch. Note that this is *different* form the way **SAVE CATEGORY** works—there, a second **SAVE** *replaces* a previously **SAVE**d group of categories instead of adding-on.

Changing Texts

When you initiate ARRAS in TSO, you will normally tell it which text you wish to consider. (This procedure may differ in some installations.) ARRAS will address that text throughout the session. To explore another text under TSO, you must **STOP** ARRAS and restart it with a different text. Thus, you have no facility to change texts dynamically under TSO.

In CMS, however, you can re-direct ARRAS's attention to another text while ARRAS is operating. You do this with the **CHANGE TEXT** command and by adding the filename of the new text you wish to address through the **FILE** option. ARRAS will then inform CMS of the change and ask that system to make the change. This operation usually takes several seconds. However, it provides you with great flexibility to move around in different texts, make comparative studies, etc.

Since categories usually apply to a specific text, when you *change texts* ARRAS will ask you if you wish to clear current categories. If you don't, you are likely to get erroneous results.

Entering and Leaving CMS

When ARRAS is run under CMS, you have options that are not available under TSO. One of the most important and versatile capabilities is to leave ARRAS, enter CMS, and later return to ARRAS as if you had never left it; that is, with all categories intact.

To leave ARRAS, simply give ARRAS the command, **CMS**; to return to ARRAS from CMS, give CMS the command, **RETURN**.

While you are in CMS—technically, the *CMS Subset*—you can do a number of different things. For example, you can call the CMS editor—**XEDIT** or another system editor—and do word processing on a file. You could edit a file of notes, adding observations resulting from your recent analysis using ARRAS. You could also edit the file for, say, a journal article describing the study you are doing with ARRAS's assistance. If you had told ARRAS to make a copy of a distribution, concordance, or other information (with the **SET SPOOL** command), you could copy that information directly into the text of the article you are working on.

If your installation is linked to a communication network, such as BITNET, you can send a copy of the paper you are working on or a copy of the ARRAS data you are considering to a colleague across the nation as easily as to a colleague on your own campus. You could also review your files or other system files to find the name of another text you wish to look at (using the **CHANGE TEXT** command.) And a number of other operations supported by CMS.

By combining ARRAS's analytic capabilities with word processing, text formatting, network communication, and other general system features offered by CMS, you will find yourself in a new environment. The computer can become not just a tool that does different things, but the *place* where you do a significant amount of your work. The things you can do there will increase as new features are added to ARRAS and to CMS and as you learn to combine them in new and different ways. The potential is limited largely by our imaginations.

Reference Manual

Introduction

The *Reference Manual* provides complete technical descriptions for all current ARRAS command words. It is designed so that you may quickly get in, get the information you need, and get back to work. While providing detailed information, it presumes that you have worked through the *Tutorial* and the *Functional Groups* sections.

The organization is alphabetic by command word. Each description is composed of three parts: examples and a brief explanation, the syntax of the command and a list of the other command words used with it, and, finally, its abbreviation and additional notes.

The *Reference Manual* also uses a consistent layout for each description and consistent type fonts to signal specific kinds of information. Both of these conventions are described below in more detail.

Format of the Detailed Command Section

Each command description is divided into three sections; the following template shows this format.

COMMAND WORD

The *command word* heads the section.

EXAMPLE:

Examples start each section.

EXPLANATION:

A full explanation of what the command does or a cross-reference is provided.

SYNTAX:

The *Syntax* for the word is a 'fill in the blanks' recipe for constructing proper commands using the word.

USE WITH:

A list of all the command words that can be used with the command being discussed. This repeats information in the usage sections, but it saves looking around.

ABBREVIATION:

The minimum abbreviation ARRAS will recognize. In most cases the first three letters are all ARRAS cares about; some commands, however, require four letters when the first three are ambiguous.

NOTES:

- For more complex commands, additional technical details are provided in notes such as this one.

How to Read Command Syntax

A syntax description is a pattern to be used in forming commands. The pattern consists of three kinds of entries:

- **BOLD CAPITAL**'s indicate command words that are to be copied exactly as they appear in the template (except that they may be typed in upper or lower case and usually may be abbreviated).
- Punctuation and other characters, which are neither numbers or letters are to be copied *exactly* (there is one exception: any colon (:) may be replaced by an equals sign (=)).
- *Lower-case-phrases* (single words or words grouped by dashes) can be replaced according to the following table.

PHRASE	**MAY BE REPLACED BY**
AND/OR	*The Boolean operators & (which stands for 'and'), | (which stands for 'or').*
NOT	*The 'not' symbol which is usually ~, but is typed ∧ on some systems.*
category-name	*Any previously defined dictionary category.*
category-name-list	*A list of category names separated by commas or blanks.*
known-word	*Either a word that appears in the text or a previously defined category-name.*
known-word-list	*A list of known words or category-names separated by commas or blanks.*
new-name	*A string of letters that is new to ARRAS. It should not be a word that occurs in the text or a previously defined category name.*
number	*Any number, usually a small value. A positive integer.*
text-location	*A number that refers to the location of a word in the text.*
text-location-list	*A list of numbers that refer to locations in the text.*
units	*In most ARRAS files, the units of text are the following* **VOLUME** **CHAPTER** **PARAGRAPH** **SENTENCE** **WORD** **PAGE** **LINE**

ADD

EXAMPLE:

MODIFY CATEGORY :FWCAT; **ADD** : FLAMES.

EXPLANATION:

- A full description of this command is given in the sections for the **MODIFY**
- This command is a subcommand of **MODIFY**
- The **ADD** sub-command adds words, categories, or linear numbers to an existing category.

SYNTAX:

MODIFY CATEGORY: *category-name* ; **ADD** : *known-word* .

(for linear categories only:)

MODIFY CATEGORY: *category-name* ; **ADD** : *text-location* .

(for linear categories only:)

WHERE:

category-name	*Any previously defined dictionary category.*
known-word	*Either a word that appears in the text or a previously defined category-name.*
text-location	*A number that refers to the location of a word in the text.*

USE WITH:

CATEGORY

MODIFY

DELETE

NAME

ABBREVIATION:

ADD

BREAK KEY

EXPLANATION:

The *BREAK KEY* interrupts ARRAS in the middle of whatever command it is executing at the time. There are three ways to resume ARRAS after it has been interrupted.

- Entering **STOP** will cancel the *entire* ARRAS session. This is exactly like entering **STOP** as a direct command.

- Entering Q will immediately terminate the command that was interrupted. This is the best way to escape from a long display (an inadvertent listing of the entire text, for example). ARRAS will return to normal command mode and await your next instruction .

- **Carriage-return** (the return key) will resume the display approximately where it was interrupted. It is likely, however, that several lines of output may be lost by the system when this is done. These are lines that ARRAS has, in fact, output but which were lost by the system during the interruption.

NOTES:

- On some systems the *BREAK KEY* sometimes takes a moment to get the computer's attention. If after several seconds the system has not stopped, press *BREAK KEY* again.

CATEGORY

EXAMPLES:

DEFINE CATEGORY : FIRE, WATER; **NAME** : FWCAT.

DISPLAY CATEGORY : FWCAT.

DEFINE LINEAR CATEGORY : 1023, 1444; **NAME** : NUMBCAT.

EXPLANATION:

- Full descriptions for this command are in the sections for:
  DEFINE *and* **DISPLAY**

- The **CATEGORY** command is used to define and display categories.

SYNTAX:

DEFINE CATEGORY : *known-word-list* ; **NAME** : *new-name* .

DEFINE LINEAR CATEGORY : *text-location-list*; **NAME** : *new-name* .

DISPLAY CATEGORY : *category-name* .

WHERE:

category-name	*Any previously defined dictionary category.*
known-word-list	*A list of known words or category-names separated by commas or blanks.*
new-name	*A string of letters that is new to ARRAS. It should not be a word that occurs in the text or a previously defined category name.*
text-location-list	*A list of numbers that refer to locations in the text.*

ABBREVIATION:

CAT

CHANGE

EXAMPLE:

CHANGE TEXT ; **FILE**: *file-name.*

EXPLANATION:

The **CHANGE** command is used *only* under CMS (see the **CMS** command). With it, the text under study can be changed. The file-name is the regular CMS file name that refers to the text to be addressed. The user or the local installation is responsible for any linking to non-user disk or for transfering texts from off-line to on-line storage.

SYNTAX:

CHANGE TEXT ; **FILE**: *file-name.*

USE WITH:

TEXT

ABBREVIATION:

CHA

CMS

EXAMPLE:

CMS .

EXPLANATION:

The **CMS** command allows you to suspend ARRAS while you work within the CMS subset. ARRAS will wait, quietly and patiently, for your **RETURN**. CMS is a part of your IBM operating system; before using this command, verify that your computer uses the CMS (not the TSO) system.

SYNTAX:

CMS

NOTES:

- **CMS** may *not* be used under the TSO system. In this case, ARRAS will simply inform you that 'CMS is not available at this installation'.
- Type **RETURN** to return from CMS to ARRAS and to continue your analysis from where you left off, i.e., with all categories intact.

CONCORDANCE

EXAMPLES:

DISPLAY CONCORDANCE : FWCAT.

DISPLAY CONCORDANCE : FWCAT; -2 TO +3 SENTENCES.

EXPLANATION:

The **CONCORDANCE** command displays every occurence of a word or category along with its surrounding context. You may specify the text width to be displayed. (ARRAS will show the single sentence surrounding each occurence of the word if you do not specify otherwise.) To specify the width, see the **CONTEXT** sub-command, which is described in its own section.

SYNTAX:

DISPLAY CONCORDANCE : *known-word* .

DISPLAY CONCORDANCE : *known-word* ;
CONTEXT -number **TO** +*number* units.

WHERE:

known-word	*Either a word that appears in the text or a previously defined category-name.*
number	*Any number, usually a small value. A positive integer.*
units	*In most ARRAS files, the units of text are the following* **VOLUME** **CHAPTER** **PARAGRAPH** **SENTENCE** **WORD** **PAGE** **LINE**

USE WITH:

CONTEXT

ABBREVIATION:

CONC

NOTES:

- Useful information on this command is also given with **CONTEXT** .

CONFIGURATION

EXAMPLES:

CONFIGURATION : FWCAT & EACAT; **NAME** : NEWCAT.

CONFIGURATION : FWCAT |EACAT;
CONTEXT : -3 **TO** +1 **PARAGRAPHS**; **NAME** : NEWCAT.

CONFIGURATION : FWCAT & EACAT;
CONTEXT : -4 **TO** +4 **WORDS**; **NAME** : NEWCAT.

CONFIGURATION : FWCAT & [-1 **TO** +1 **WORDS**] *NOT* EACAT
NAME : NEWCAT.

CONFIGURATION : (FWCAT |EACAT)
& (HOTCAT |COLDCAT); **CONTEXT** : -3 **TO** +3 **WORDS**;
NAME : NEWCAT.

EXPLANATION:

- The **CONFIGURATION** command creates new categories by combining old categories and words. The basic command has two distinct formats.

  - **Configurations using &** select locations from the first category based on their proximity to words or locations in the second category. If such a pattern can be found within a given context, the location is included in a new category. If the 'not' parameter, ~, is used, then the location is chosen only if **no** word from the second category appears within the context.

  - **Configurations using** | combine two categories exactly as though the new category had been created using **DEFINE** .

- *Nested* configurations are simply a shorthand way of combining intermediate configurations to form a "pattern of patterns." In particular, a nested configuration can be rewritten as a series of several basic configuration commands. Anything that can be accomplished using nested configuration commands can be done equally effectively in another way. For example,

  CONFIG: (A & B) & (C & D); **NAME** : E.

  is equivalent to:

  CONFIG: A & B ; **NAME** : A1.
  CONFIG: C & D ; **NAME** : B1.
  CONFIG: A1 & B1 ;**NAME** : E.

SYNTAX:

CONFIGURATION : *known-word AND/OR known-word* .

CONFIGURATION : *known-word AND/OR known-word* ;
CONTEXT : - *number* **TO** + numbr *units* .

CONFIGURATION : *known-word* **AND** [-numbr **TO** +numbr *units*] *known-word* .

CONFIGURATION : *known-word* **AND** *NOT known-word* .

CONFIGURATION : *known-word* **AND** *NOT known-word* ;
CONTEXT : - *number* **TO** + numbr *units* .

CONFIGURATION : *known-word* **AND** [-numbr **TO** +numbr *units*] *NOT known-word* .

CONFIGURATION : *nested-configuration AND/OR nested-configuration* .

CONFIGURATION : *nested-configuration AND/OR nested-configuration* ;
CONTEXT : - *number* **TO** + numbr *units* .

CONFIGURATION : *nested-configuration* AND *NOT nested-configuration* .

CONFIGURATION : *nested-configuration* AND *NOT nested-configuration* ;
CONTEXT : - *number* **TO** + numbr *units* .

CONFIGURATION : *nested-configuration*
AND [-numbr **TO** +numbr *units*] *NOT nested-configuration* .

WHERE:

known-word	*Either a word that appears in the text or a previously defined category-name.*
number	*Any number, usually a small value. A positive integer.*

units	*In most ARRAS files, the units of text are the following* **VOLUME** **CHAPTER** **PARAGRAPH** **SENTENCE** **WORD** **PAGE** **LINE**
nested-configuration	*A nested configuration.*
AND/OR	*The Boolean operators & (which stands for 'and'), | (which stands for 'or').*
NOT	*The 'not' symbol which is usually ~, but is typed ∧ on some systems.*

USE WITH:

CONTEXT .

ABBREVIATION:

CONF

CONTEXT

EXAMPLES:

```
DISPLAY CONCORDANCE : FWCAT;
    CONTEXT : -3 TO +2 WORDS.

DISPLAY CONCORDANCE : FWCAT;
    CONTEXT : +1 TO +10 WORDS.

CONFIGURATION : FWCAT & EACAT;
    CONTEXT : -2 TO +2 SENTENCES.
```

EXPLANATION:

CONTEXT is a sub-command that specifies a width of text; it must be used with some other primary command. ARRAS first goes to the exact location requested by the main command, then examines the optionally specified context relative to that positon as indicated by the **CONTEXT** subcommand. So, for example, -2 TO +3 **SENTENCES** will instruct ARRAS to go to the sentence where the word occurs, go back two sentences before it, and then go forward three sentences after it. Context can always be omitted, in which case ARRAS assumes that the single sentence containing the word is to be considered. Note, also, that the two values of context may both have the same sign; e.g., +1 **TO** +10 **WORDS** tells ARRAS to look at the context beginning with the first word after the target word and concluding ten words forward from that position. Hence the target word will not appear in the specified context.

SYNTAX:

DISPLAY CONCORDANCE : *known-word* ;
CONTEXT : *-number* **TO** *+number units* .

CONFIGURATION : *known-word AND/OR known-word* ;
CONTEXT : *-number* **TO** *+number units* .

WHERE:

known-word	*Either a word that appears in the text or a previously defined category-name.*
number	*Any number, usually a small value. A positive integer.*
units	*In most ARRAS files, the units of text are the following* **VOLUME** **CHAPTER** **PARAGRAPH** **SENTENCE** **WORD** **PAGE** **LINE**

USE WITH:

DISPLAY

PRINT

CONCORDANCE

CONFIGURATION

ABBREVIATION:

CONT

CONVERT

EXAMPLE:

CONVERT : FWCAT; **NAME** : FWLINCAT.

EXPLANATION:

The **CONVERT** command changes a categeory from dictionary to linear form. In the example, FWLINAT is the resulting category. The members of this new category will be in increasing textual order. The difference between the original dictionary category and the resulting linear category produced by **CONVERT**ing that dictionary category can be seen in the concordance output produced by each. A concordance for a dictionary category will display the context for all occurences of the first category word in the order in which they appear in the text, followed by all occurences for the second word, etc. A concordance for a **CONVERT**ed category will display the contexts for all the individual words inter-leaved into a single sequence from beginning to end of the text.

SYNTAX:

CONVERT : *category-name* ; **NAME** : *new-name* .

WHERE:

category-name	*Any previously defined dictionary category.*
new-name	*A string of letters that is new to ARRAS. It should not be a word that occurs in the text or a previously defined category name.*

ABBREVIATION:

CONV

NOTES:

- You must supply a name to **CONVERT** ; if you fail to do so, ARRAS will prompt for one. A blank name is an error.
- Only one category may be converted at a time.

DEFINE

EXAMPLES:

DEFINE CATEGORY : FIRE , WATER ; **NAME** : FWCAT .

DEFINE LINEAR CATEGORY : 1000, 1099; **NAME** : NUMBCAT .

DEFINE CATEGORY : $INPUT.

EXPLANATION:

DEFINE is used to create a new category and give it a name. It can be used in three ways.

- **DEFINE** builds a category consisting of words which occur in the text or previously defined categories. While you may mix words and categories to form a new category, *in general* it is preferable to think of your categories as forming a hierarchy: level one categories contain just words or text locations, level two categories contain just level one categories, level three categories contain level two categories, and so on, to as high a level of abstraction as you find useful.

- **DEFINE LINEAR** builds a category consisting of a set of text locations.

- **DEFINE CATEGORY** : $INPUT. retrieves categories which were **SAVE**d during a previous **ARRAS** session. (See the **SAVE** command.) *Only* under CMS may a file name be specified.

SYNTAX:

DEFINE CATEGORY : *known-word-list* ; **NAME** : *new-name* .

DEFINE LINEAR CATEGORY : *text-location-list*; **NAME** : *new-name* .

DEFINE CATEGORY : $INPUT

DEFINE CATEGORY : $INPUT; FILE: *file-name* .

WHERE:

file-name	*The name of a system file in which a set of categories was previously* **SAVE***d (See the SAVE Command). This option can only be specified on CMS systems; there is no choice of file name under TSO.*
known-word-list	*A list of known words or category-names separated by commas or blanks.*
new-name	*A string of letters that is new to ARRAS. It should not be a word that occurs in the text or a previously defined category name.*
text-location-list	*A list of numbers that refer to locations in the text.*

USE WITH:

CATEGORY

LINEAR

NAME

ABBREVIATION:

DEF

DELETE

EXAMPLE:

MODIFY CATEGORY : FWCAT ; **DELETE** : FIRE .

MODIFY CATEGORY : FWCAT ; **DELETE** .

EXPLANATION:

- A full description of this command is given in the sections for the **MODIFY** command.
- **DELETE** is used to remove an item from a category or to delete the entire category.

SYNTAX:

```
MODIFY CATEGORY category-name ; DELETE .
                              ; DELETE : known-word .
                              ; DELETE : known-word ; NAME : new-name .
                              ; DELETE : known-word ; ADD : known-word .
                              ; DELETE : known-word ; ADD : known-word ;
                                         NAME : new-name .
```

WHERE:

category-name	*Any previously defined dictionary category.*
known-word	*Either a word that appears in the text or a previously defined category-name.*
new-name	*A string of letters that is new to ARRAS. It should not be a word that occurs in the text or a previously defined category name.*

USE WITH:

ADD

CATEGORY

MODIFY

NAME

ABBREVIATION:

DEL

NOTES:

- When **NAME** is specified the **DELETE** command creates a new category and leaves the original unchanged. When no name is specified the original is permanently altered.

DICTIONARY

EXAMPLE:

```
DISPLAY DICTIONARY : FI - FZ.

DISPLAY DICTIONARY : FIRE.
```

EXPLANATION:

The **DICTIONARY** command shows the words within a specified alphabetical interval that appear in the text along with their frequency. If no alphabetical range is specified, the *entire* dictionary or vocabulary will be displayed! If a single word is given, ARRAS will report whether or not that word appears in the text and, if so, its frequency.

SYNTAX:

DISPLAY DICTIONARY .

DISPLAY DICTIONARY : *known-word* .

DISPLAY DICTIONARY : *string1 - string2* .

WHERE:

word	*Any group of letters you wish.*
string1 - string2	*Two strings of characters such that string1 is before string2 in alphabetical sequence.*

USE WITH:

DISPLAY

ABBREVIATION:

DIC

NOTES:

- Placing spaces around the dash is **critical** when using **DICTIONARY** . The construct FI-FZ is interpreted as a single hyphenated word rather than a range.

DISPLAY

EXAMPLES:

```
DISPLAY CATEGORY : FWCAT.
        CONCORDANCE : FWCAT.
        DICTIONARY : FIRE.
        DICTIONARY : FIR - FIZ.
        DISTRIBUTION : FWCAT.
        INDEX : FIRE.
        TEXT : 15023, - 3 TO + 5 SENTENCES.
```

EXPLANATION:

- **DISPLAY** indicates that output is to be shown on the screen.

- A full description of this command is given in the sections for the commands listed above.

SYNTAX:

DISPLAY **CATEGORY** : *category-name* .
CONCORDANCE : *known-word* .
DICTIONARY : *known-word* .
DICTIONARY : *word* - *word* .
DISTRIBUTION : *known-word* .
INDEX : *known-word* .
TEXT :*number* , - *number* TO + *number units* .

WHERE:

category-name	*Any previously defined dictionary category.*
known-word	*Either a word that appears in the text or a previously defined category-name.*
number	*Any number, usually a small value. A positive integer.*
word	*Any group of letters you wish.*

USE WITH:

CATEGORY

CONCORDANCE

DICTIONARY

DISTRIBUTION

INDEX

TEXT

ABBREVIATION:

DISP

DISTRIBUTION

EXAMPLE:

DISPLAY DISTRIBUTION : FWCAT.

EXPLANATION:

DISTRIBUTION produces a bar graph showing the frequency distribution over the text for the word or category specified. The text is broken into fifty sections of equal size; the graph shows how often the word or category occurs in each section. If a word or category occurs more than twenty times in a section, a column of asterisks (*) is printed.

SYNTAX:

DISPLAY DISTRIBUTION : *known-word* .

WHERE:

known-word	*Either a word that appears in the text or a previously defined category-name.*

USE WITH:

DISPLAY

ABBREVIATION:

DIST

EDIT

EXAMPLE:

SET EDIT : OFF.

EXPLANATION:

The **EDIT** command controls the automatic prompting that ARRAS does when a command error is found. Initially, ARRAS always asks if you wish to edit erroneous commands. Setting the edit switch OFF will stop this. ARRAS may still inquire if you wish help. The **SET HELP** command can be used to prevent this prompt, as well.

SYNTAX:

SET EDIT : *ON/OFF* .

USE WITH:

SET

ABBREVIATION:

EDI

NOTES:

- The **EDIT** switch is initially ON.
- The default, if you simply type **SET EDIT** with no parameters, is to set automatic prompting OFF.

EXPAND

EXAMPLE:

EXPAND : FWCAT, HCAT.

EXPLANATION:

The **EXPAND** command only affects categories which have other categories as members. Each member category is 'expanded' into the words (or numbers) which comprise it. The effect is to compress the hierarchy of the category. The result is that the original category no longer has categories as members; it only has words or linear numbers as members. This command is more a conceptual convenience than an extension of ARRAS's funcional capacity. Its results can only be seen by using the **DISPLAY CATEGORY** command.

SYNTAX:

EXPAND : *category-name-list* .

WHERE:

category-name-list	*A list of category names separated by commas or blanks.*

ABBREVIATION:

EXP

HELP

EXAMPLE:

HELP .

HELP : CONTEXT.

HELP : CONFIGURATION; DESCRIPTION.

EXPLANATION:

The **HELP** command starts an internal self-instruction system. It offers three modes of access.

- By typing the command, **HELP**, or by responding YES to the prompt when ARRAS detects a command error, the system will list current command words for which explanations are availible. To select one, simply type that word, when prompted.
- Second, you may request help for a specific command word by typing it as the parameter to the **HELP** command.
- Third, you may request a specific part of the help instructions for a given word by typing FORMAT, DESCRIPTION, or OPTIONS as a subcommand to **HELP**. This requires that a command word parameter also be specified.

To turn *off* the automatic prompt for help when ARRAS finds an incorrect command, see the **SET** command description.

SYNTAX:

HELP .

HELP :*command-word* .

HELP :*command-word* ;DESCRIPTION.

HELP :*command-word* ;FORMAT.

HELP :*command-word* ;OPTIONS.

WHERE:

command-word .... *Any word in ARRAS's command vocabulary.*

ABBREVIATION:

HEL

INDEX

EXAMPLE:

DISPLAY INDEX : FIRE.

EXPLANATION:

INDEX produces a list of locations in the text (linear positions and page numbers). For a specified word, **INDEX** produces a list of locations for every occurrence of the word; for a category, a list of locations for each occurrence of each word in the category.

SYNTAX:

DISPLAY INDEX : *known-word-list* .

WHERE:

known-word-list	*A list of known words or category-names separated by commas or blanks.*

ABBREVIATION:

IND

NOTES:

- The index will not necessarily be in ascending order if it is done for a category or list of words. To get an ordered index, **CONVERT** the category to linear form and then **INDEX** the result.

LINEAR

EXAMPLE:

DEFINE LINEAR CATEGORY : 1000, 1400, 1935; **NAME** : LINCAT.

EXPLANATION:

- A full description of this command is given in the sections for the **DEFINE** command.
- This command is a subcommand of **DEFINE** .
- **This command permits defining a category as a sequence of text positions, as opposed to a list of words. Thus, its members are tokens rather than types.**

SYNTAX:

DEFINE LINEAR CATEGORY : *text-location-list*; **NAME** : *new-name* .

WHERE:

text-location-list	*A list of numbers that refer to locations in the text.*
new-name	*A string of letters that is new to ARRAS. It should not be a word that occurs in the text or a previously defined category name.*

USE WITH:

DEFINE

NAME

ABBREVIATION:

LIN

MODIFY

EXAMPLES:

```
MODIFY CATEGORY : FWCAT; DELETE .
                       ; ADD : BLAZE.
                       ; DELETE : WATER.
                       ; DELETE : WATER; NAME : DRYCAT.
                       ; NAME : NEWFWCAT.
```

EXPLANATION:

The **MODIFY** command is used to change existing categories. Four types of changes can be specified.

- The command can delete the category altogether.
- The command can add new words, categories, or text-location numbers to the original category.
- The command can remove particular words, categories, or text-location numbers from the original.
- The command can rename the category.

SYNTAX:

```
MODIFY CATEGORY : known-word ; DELETE .
                             ; ADD : known-word .
                             ; ADD : text-location .
                             ; DELETE : known-word .
                             ; DELETE : known-word ; NAME : new-name .
```

WHERE:

category-name	*Any previously defined dictionary category.*
known-word	*Either a word that appears in the text or a previously defined category-name.*
text-location	*A number that refers to the location of a word in the text.*
new-name	*A string of letters that is new to ARRAS. It should not be a word that occurs in the text or a previously defined category name.*

USE WITH:

ADD

CATEGORY

DELETE

NAME

ABBREVIATION:

MOD

NOTES:

- Text location numbers can only be added to or deleted from a linear category.
- When an entire category is deleted, ARRAS will prompt for confirmation that you really meant to do so.
- Several **ADD** 's and/or **DELETE** 's can be done in a single command.

NAME

EXAMPLES:

DEFINE CATEGORY : FIRE, WATER; **NAME** : FWCAT.

CONVERT : FWCAT; **NAME** : FWLIN.

EXPLANATION:

- Full descriptions for this command are in the sections for:
  DEFINE **CONVERT** **MODIFY** **CONFIGURATION**
- This subcommand associates a name with a category.

SYNTAX:

DEFINE CATEGORY : *known-word-list* ; **NAME** : *new-name* .

CONVERT : *category-name* ; **NAME** : *new-name* .

MODIFY CATEGORY : *category-name* ; **NAME** : *new-name* .

CONFIGURATION : *configuration-expression* ; **NAME** : *new-name*

WHERE:

category-name	*Any previously defined dictionary category.*
configuration-expression	*A Boolean combination of words or categories.*
known-word	*Either a word that appears in the text or a previously defined category-name.*
known-word-list	*A list of known words or category-names separated by commas or blanks.*
new-name	*A string of letters that is new to ARRAS. It should not be a word that occurs in the text or a previously defined category name.*

USE WITH:

CATEGORY

DEFINE

CONFIGURATION

CONVERT

MODIFY

ABBREVIATION:

NAM

POETRY

EXAMPLE:

SET POETRY: ON.

EXPLANATION:

- This command is a subcommand of **SET** .
- The **POETRY** command instructs ARRAS to display the text line for line as it was in the original printed version (rather, the version scanned by ARRAS). ARRAS normally displays text (e.g., in a CONCORDANCE or TEXT command) by filling up the display line with as many words as will fit. This option permits poetry or other texts for which the original line format is important to be displayed with line integrity preserved.

SYNTAX:

SET POETRY : *ON/OFF* .

USE WITH:

SET .

ABBREVIATION:

POE

SAVE

EXAMPLE:

SAVE .

SAVE : FIRECAT, WATERCAT, EARTHCAT, AIRCAT.

SAVE : FIRECAT, WATERCAT, EARTHCAT, AIRCAT;

FILE : ELEMENTS.

EXPLANATION:

The **SAVE** command saves categories in a disk file. They may be reinstated during a future session through the $INPUT option of the **DEFINE CATEGORY** command. The **FILE** option is available only under CMS.

SYNTAX:

SAVE : *known-word-list* .

SAVE : *known-word-list* ; **FILE** :5 *file-name* .

WHERE:

file-name	*The name of a system file in which a set of categories was previously* **SAVE***d (See the SAVE Command). This option can only be specified on CMS systems; there is no choice of file name under TSO.*
known-word-list	*A list of known words or category-names separated by commas or blanks.*

ABBREVIATION:

SAV

SCREEN

EXAMPLE:

SET SCREEN : 24.

SET SCREEN : ON.

SET SCREEN .

EXPLANATION:

The **SCREEN** command allows you to tell ARRAS how many lines to put on your display terminal at one time. ARRAS pauses each time it fills your screen to give you a chance to read before going on. To continue the ARRAS display after a pause, type carriage return. By setting **SCREEN OFF**, you disable this feature causing ARRAS to display continuously whatever information is requested.

SYNTAX:

```
SET SCREEN : number .
           : ON.
           : OFF.
```

NOTES:

- A default of 24 lines is assumed if you simply **SET SCREEN** ON.

SET

EXAMPLES:

```
SET EDIT : ON.

SET SPOOL : OFF.

SET HELP : ON.
SET SCREEN  : OFF.
            : 24.
```

EXPLANATION:

- Full descriptions for this command are in the sections for:
  EDIT **HELP** **POETRY** **SCREEN** **SPOOL**

- The **SET** command changes the particular environment specified by the sub-command.

SYNTAX:

SET EDIT : *ON/OFF*

SET SPOOL : *ON/OFF*

SET HELP : *ON/OFF*

```
SET SCREEN  : ON/OFF
            : number
```

USE WITH:

EDIT

HELP

POETRY

SCREEN

SPOOL

SPOOL

EXAMPLE:

SET SPOOL : ON.

EXPLANATION:

- This command is a subcommand of **SET** .
- The **SPOOL** command instructs ARRAS to store a copy of the information it displays on the screen in a disk file. You will later be able to examine and, if you wish, print the file after the ARRAS session is completed. Any use of this file will be through your local operating system (CMS or TSO), however, and independent of ARRAS .

SYNTAX:

SET SPOOL : *ON/OFF* .

USE WITH:

SET

ABBREVIATION:

SPO

NOTES:

- The file containing your output is named $SPOOL if you use TSO.
  If you use CMS the file is named $SPOOL DATA A.
- In general, *all* spooled output will be kept until explicitly discarded. The local operating system must be used to discard accumulated output.

STOP

EXAMPLE:

STOP .

EXPLANATION:

The **STOP** command terminates an ARRAS session. Any categories which have not been SAVEd (see SAVE in the setion that follows) will be lost.

ABBREVIATION:

STO

TEXT

EXAMPLE:

```
DISPLAY TEXT .

DISPLAY TEXT : 15301.

DISPLAY TEXT : 15301, - 3 TO + 4 PARAGRAPHS.
```

EXPLANATION:

The **TEXT** command displays text. It allows you to select a particular linear location in the text and the width of the context to be shown surrounding it. Used with no parameters, it displays the entire text. Used with only a linear number, it begins displaying the text from that position and continues to the end of the text.

SYNTAX:

DISPLAY TEXT .

DISPLAY TEXT : *text-location* .

DISPLAY TEXT : *text-location* , *-number* **TO** *+number units* .

WHERE:

text-location	*A number that refers to the location of a word in the text.*
number	*Any number, usually a small value. A positive integer.*
units	*In most ARRAS files, the units of text are the following* **VOLUME** **CHAPTER** **PARAGRAPH** **SENTENCE** **WORD** **PAGE** **LINE**

USE WITH:

DISPLAY

ABBREVIATION:

TEXT

Preparing Texts

Introduction

Preparing texts for ARRAS is a three-step procedure. First, the text must be transformed from a physical form (e.g., a printed book) to another form (e.g., ASCII or EBCDIC). Second, certain features, such as the beginnings of paragraphs and pages, must be marked in a way ARRAS can recognize. Most of these markings can be inserted automatically by an optical scanner or by a typist transcribing a text. Third, the text must be processed by a utility computer program, supplied with the ARRAS system, to turn it into a form the computer can "read" quickly. We will not address the first step, actual data entry, since options available will differ from site to site, depending upon equipment and policy. Instead, we will concentrate on the technical aspects of **encoding** and **processing new texts.**

One note of advice: a text worth encoding should be the best available edition. While the point is obvious, we have seen instances where literary works have been encoded using student editions rather than the standard, scholarly edition also available. We would urge the perspective that texts, once encoded, can be used by others for many different purposes. Whenever possible, the standard or best available edition should be used when that version will meet the needs of the individual preparing it.

Encoding

Encoding is the process of transforming information from physical to electronic form. Currently, the two primary options are typing on a terminal and scanning by optical reader, such as the Kurzweil Data Entry Machine. We have designed ARRAS's encoding conventions to correspond closely to output produced by an optical scanner. However, after a text is scanned, it may require some clean up and modification using an online editor, such as WYLBUR or XEDIT, or a microcomputer.

The primary intent of ARRAS's encoding conventions is to mark linguistic segments (paragraphs, chapters, etc) and physical segments (lines, pages, etc.). Readers (often unconsciously), use blank space as a major key in recognizing these units, but blank space is very difficult to control on output from a scanner and on typed copy—hence, ARRAS's explicit marks.

ARRAS Style Sheet

	Symbol	*Rules/Meaning*
Form	\xxxx	A backslash(\) followed by one or more characters will be interpreted as a format mark. Example: \para marks a new paragraph.
Segments	\para	Place on a line by itself before each paragraph including the first paragraph of the text and the first paragraph of a chapter or volume.
	\chap [title]	Place on a line, by itself or with title, before each chapter, including the first chapter of the text and the first chapter of a multivolume work. Example: \chap or \chap Introduction
	\vol [title]	Place on a line, by itself or with title, before each volume including the first volume of a multivolume text Example: \vol or \vol Part I
	\page [-#]	Place on a line, by itself or with a page number, before each page of a text including the first page of the text. With a number, \page sets the page number to that value. A hyphen must precede the number and no spaces are allowed in the sequence. Without a number, \page increments the page number. Example: \page–3 first page of text numbered 3. \page page number incremented by 1.
	\+	Since the text line by line as it appears in the printed form is normally typed as a command, mark logical lines too long to fit on one physical line with a \+ at the end of the line. Example: this line is too long to fit on one \+ physical line.
	\line	Forces a line break when otherwise ambiguous (the opposite of \+).

Abbreviations		Distinguishing between abbreviations and ends of sentences is difficult for ARRAS and other language analysis programs. The strategy adopted for ARRAS is to establish conventions that will work in *most* cases without manual editing but to mark manually those cases where the convention fails. Compare the following:
	xxx. a	As the right-most character of a word, followed by one or more spaces, followed by a lower-case character, period marks an abbreviation. Example: etc. is a common abbreviation
	xxxx. A	As the right-most character of a word, followed by one or more spaces, followed by an upper-case character, period marks end of sentence. Example: end of sentence. Begin next sentence.
	xxx. \ns A	Where the above rules would be misinterpreted, put a no-sentence code (\ns) after the period and a space, and before a space and the following upper-case characters. Exception: The \ns is not required after the following abbreviations: Mr., Mrs., Ms., Dr. Example: Mrs. Jones
	xxx. \sent	Forces a sentence break where otherwise ambiguous; for example, a sentence ending with an abbreviation. To avoid possible misinterpretation, put the last \sent after the abbreviation to ensure sentence break.
Quotations	“xxxx.” ‘xxxx.’ \’x x\’	Quotations may be marked with either single or double quote marks. You must use the single left quote mark, instead of the apostrophe for left single quotes. Punctuate conventionally *except* when an apostrophe occurs within a quote marked by single quote marks. In such cases, mark the leading or trailing apostrophe with a preceeding \. (i.e., \’). “He said, ‘\’79 was a great year for Jones\’ Burgundy!’ ”
Accents	ASCII > 127	Code accented characters with single ASCII codes (key values) > 127 as defined for the IBM-PC computer. ARRAS 1.0 does not provide special handling for accented characters, but this facility is anticipated for future releases. Such codes will preserve that information in texts prepared in the meantime and can be adapted to conventions used in future releases of ARRAS.

Processing New Texts

Once the text is *encoded*, it must be processed by a program called ARRASCAN. Briefly, ARRASCAN transforms a text that is in familiar line by line format that *you* can read into a decomposed form (inverted file) that the *computer* can read. More accurately—a form the computer can search and analyze more quickly and efficiently. While this processing is an extra step, it need be done only once and it repays ample dividends in speed and flexibility.

While a text can be processed and the inverted file form placed in a public library or database so all users may access it, we will describe the process you would use to create a personal text saved in and used from your own personal TSO file or CMS disk. To place a processed text in the public library or files, check with your local Computer Center.

The first step in processing a new text is to place it in a TSO or CMS file so that each text line is on a separate 80 character file line. This will insure that ARRAS can recreate the text in original line by line form, if you request it to do so (see the **SET POETRY** option in the *Reference Manual*).

The second (and last) step is to invoke the ARRASCAN program and tell it the name of the file that contains the unprocessed text and the name it is to use for the processed text. Pay attention to the mnemonics of this name since you will have to tell ARRAS this name when you wish to analyze the text.

For the specific forms of the command used in your environment, see *Starting ARRAS* at the front of this manual.

That's it.