MIT/LCS/TR-399

WALTER USER'S MANUAL (VERSION 3.0)

David K. Gifford
Robert G. Cole
David A. Segal

September 1987

This blank page was inserted to preserve pagination.

MIT/LCS/TR-399

Walter

User's Manual

(Version 1.0)

David K. Gifford, Robert G. Cote, and David A. Segal

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under contract number N00014-83-K-0125.

© 1987 Massachusetts Institute of Technology

Walter User's Manual

Version 1.0

Walter is a UNIX program that provides access to databases located at MIT via the DARPA Internet. The databases provided by Walter include the full-text of the New York Times for the past 90 days. A sophisticated full-text query language is provided, and Walter uses a query routing algorithm to direct requests to the proper database server at MIT. Walter was built as an experimental test of query routing and the remote pipe and procedure model for distributed communication.

Keywords: Walter, database server, user interface, remote pipe and procedure model, query routing, full-text, database

Table of Contents

Introduction 1
Section 1 — Walter's User Interface 4
1.1 — Starting Up 4
1.2 — How Walter Uses the Screen 4
1.3 — The Query Window 7
1.3.1 — Submitting Queries 7
1.3.2 — Query Routing 8
1.3.3 — Creating and Manipulating Your Query List 8
1.4 — The Summary Window 10
1.5 — The Article Window 12
1.6 — The Help Window 13
Section 2 — Databases and Services 15
2.1 — Fields 15
2.2 — Categories 16
2.3 — Subjects 16
Section 3 — The Query Language 20
3.1 — Query Grammar 20
3.2 — Dates 20
3.3 — Case and Punctuation 21
3.4 — Useful Queries 21
Appendix I — How to Become an Authorized User 22
Appendix II — Installing Walter 23
Appendix III — Summary of Commands and Keystrokes 24
III.I — Editing Keys 24
III.II — Display Keys 24
III.III — Function Keys 24
III.IV — Alphabetical Listing of Commands 25
Appendix IV — Your Responsibilities 26

List of Figures

Figure 1-1: The Screen Format 5
Figure 1-2: The Information Window 6
Figure 1-3: The Query Window 7
Figure 1-4: The Summary Window 10
Figure 1-5: The Article Window 12
Figure 1-6: The Help Window 13
Figure IV-1: Text of Agreement 27

Introduction

Welcome to *Walter* — one of the experimental computer systems that is part of the Boston Community Information System Project. *Walter* allows you to use a UNIX system (including DEC Vaxes, IBM RTs, and Sun Workstations) connected to the DARPA internet to access a 90-day database of the *New York Times*. Walter implements a simple, yet powerful, full-text query language that allows you to locate information of interest.

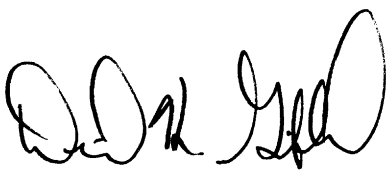
Walter runs on any UNIX system and implements a user interface to a collection of database servers at MIT. *Walter* serves as your intermediary to the database servers, sending your requests to an appropriate server, and displaying the responses generated by the servers. A *query routing* algorithm is automatically used to determine which database server at MIT should be used to process a query, and you will not be aware of which MIT server is selected.

This user's manual is designed to tell you everything you need to know to use the *Walter* system. We have prepared this manual for new and experienced UNIX users alike. Most importantly, we hope the manual will answer all your questions about using the program.

Walter was built as an experimental test of both a new communication model for distributed computer systems [Gifford86], and a query routing algorithm [Gifford85]. Because *Walter* is an experiment, we are very much interested in your thoughtful evaluation of the software. Not only do we want your first impressions of the system, *i.e.* your ideas on making the system easier to use during the first few hours, but we also want your later impressions, *i.e.* your suggestions and advice (on all aspects of the system) from the perspective of an experienced user. Please feel free to send your comments and suggestions to us via Internet mail to `"walter@db.lcs.mit.edu"`.

As a participant in this experiment, you have signed an agreement concerning your responsibilities. Note that you must abide by this agreement, in spirit and in letter.

We welcome you to this experiment to test *Walter*, and we look forward to working with you in the coming months.



Professor David K. Gifford
Massachusetts Institute of Technology
Cambridge, Massachusetts
July, 1987

[Gifford85] Gifford, D. K., et al., "An Architecture for Large Scale Information Systems", Proc. of the 10th Symposium on Operating System Principles, Orcas Island, Washington, December 1 - 4, 1985, pp. 161-170.

[Gifford86] Gifford, D. K., "Remote Pipes and Procedures for Efficient Distributed Communication", Technical Report TR-394, MIT Laboratory for Computer Science, Cambridge, MA, October 1986.

Organization of this Manual

This introduction is followed by three sections which describe, in turn:

- the capabilities of *Walter*, and how to use it;
- the information services provided by *Walter*; and
- a complete specification of *Walter's* query language.

There are four appendices. Appendix I is the most important one for new users: it describes how to become an authorized user of *Walter*. Appendix II instructs you how to install the software at your host site. Appendix III is a reference section that summarizes all valid *Walter* commands and keystrokes. And finally, Appendix IV briefly discusses your responsibilities as an authorized user of The Community Information System services.

How to Use this Manual

If you are not already authorized as a user of *Walter*, you should read Appendix I and follow the instructions for becoming authorized. You should also read Appendix IV in order to gain an understanding of the responsibilities of an authorized user.

If you are already authorized, then read through the sections in the order they are presented. First, read the section about *Walter's* user interface and familiarize yourself with its functions. After you feel comfortable with the program, read the section on Databases and Services and the section on the Query Language. The former describes the types of data that are available. The latter is a detailed description of the query language. Of course, the best way to learn how to use *Walter* is to experiment with it yourself. Good luck!

Section 1 — Walter's User Interface

Walter is a database querying system. It can access several databases containing up-to-date news articles, gathered from sources such as the *New York Times* news wire. When you submit a query to the program, it finds the articles in the database that *match* the query and displays a brief summary of each article. You can then peruse the list of summaries and instruct *Walter* to display articles that interest you. At any point, you can return to the list of summaries and look for other articles you want to read.

1.1 — Starting Up

When you first start *Walter*, it initializes the screen and attempts to establish a connection to one of its databases. Once the connection is established, the program checks whether you are an "authorized user"; if you are not authorized, the program will check to see if you are an authorized user of any of the other databases it can access; if not, it prints a message and terminates. For information about becoming authorized, see Appendix I.

While *Walter* is busy establishing connections and checking your authorization, it may display various messages near the bottom of the screen — in particular, if you are a new user, it will display a message saying that it cannot find your "query list file" (the query list file is described in section 1.3.3). If you are an authorized user, a one-line message telling you to type "?" for help should appear. At this point, *Walter* is ready to process your queries and/or commands.

1.2 — How Walter Uses the Screen

As shown in Figure 1-1, *Walter* divides the screen of your terminal into five separate areas:

- The *header line* is the highlighted line at the top of the screen. On the left hand side of the header line, the window name is displayed — initially, the query window. On the right hand side of the header line, the portion of the window that is in view is displayed. For example, if you are viewing an article, the header line will indicate the range of lines you are viewing, as well as the total number of lines in the article.
- The *message line*, also highlighted, is the last line on the screen. This line informs you of the progress the program is making in executing a command, as well as informing you of any errors that may have occurred. Initially, the message line displays a message that begins with "*** Type ? for help..".
- The *query line* (2 lines) is located just above the message line. Initially, it contains the word "QUERY:". You can use the query line to type in queries that you want to submit to *Walter*. Hitting carriage return on this line processes the query. There are a number of editing keys that may be used to edit the query line. They are a subset of the Emacs line-editing keystrokes and are outlined in Appendix III.
- The *command line* is the highlighted line just above the query line. Initially, the command line is blank. If you want to issue a command, press Cntl-x (this places the cursor on the command line) and type the name of the command (see Appendix III - Summary of Commands) followed by a carriage return. When you type a space, *Walter* attempts to complete the command you have typed. For example, if you type Cntl-x "r" "space", *Walter* completes this to the command "read". All of the editing keys available on the query line can also be used on the command line.
- The *window display* occupies the remainder of the screen. It is used to display one of the following

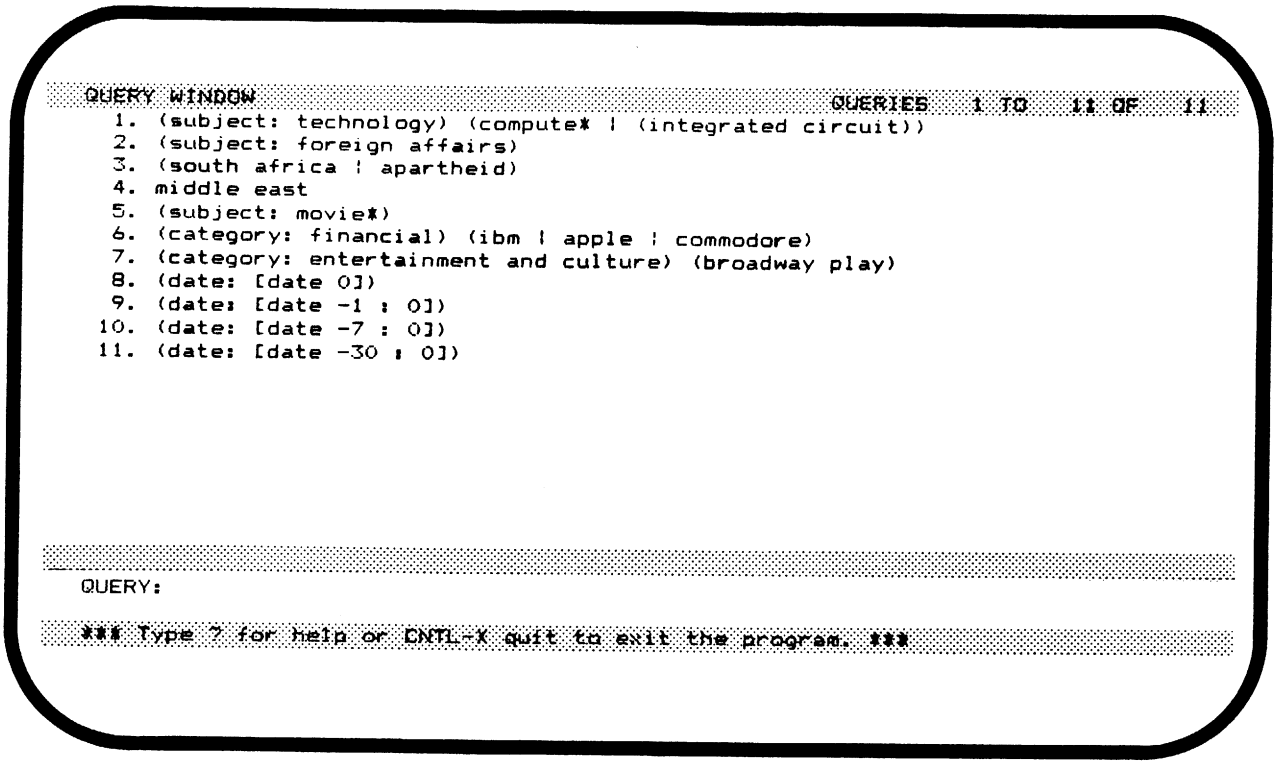


Figure 1-1: The Screen Format

four windows:

- the *query window* displays your personalized query list (if one exists)
- the *summary window* displays summaries of the articles that matched the last submitted query (this window is initially empty)
- the *article window* displays the text of the last article that was displayed in response to the "read" command (this window is also initially empty)
- the *help window* displays on-line documentation (this window is also initially empty)

When *Walter* starts up, initially it displays the query window.

When *Walter* needs to get additional information, it "pops up" an *information window* with a prompt for a response. Once you respond, by pressing an appropriate choice, this window goes away and the program restores the screen to its original state. The information window is shown in Figure 1-2. Note that the information window only covers part of the screen: in the figure, the last two lines of the query list are still visible below the dashed line which signifies the bottom of the information window.

Walter uses the information window primarily to ask the course of action it should take when it is unsure of how to process a given query or command. Occasionally, the program uses the information window to ask for confirmation when you have asked it to undo something that cannot be undone easily.

> For your protection, *Walter* always prompts for a confirmation before performing a potentially destructive action. This request is done either in the information window or on the command line.

You can move between the different windows by pressing the escape key and the first letter of the

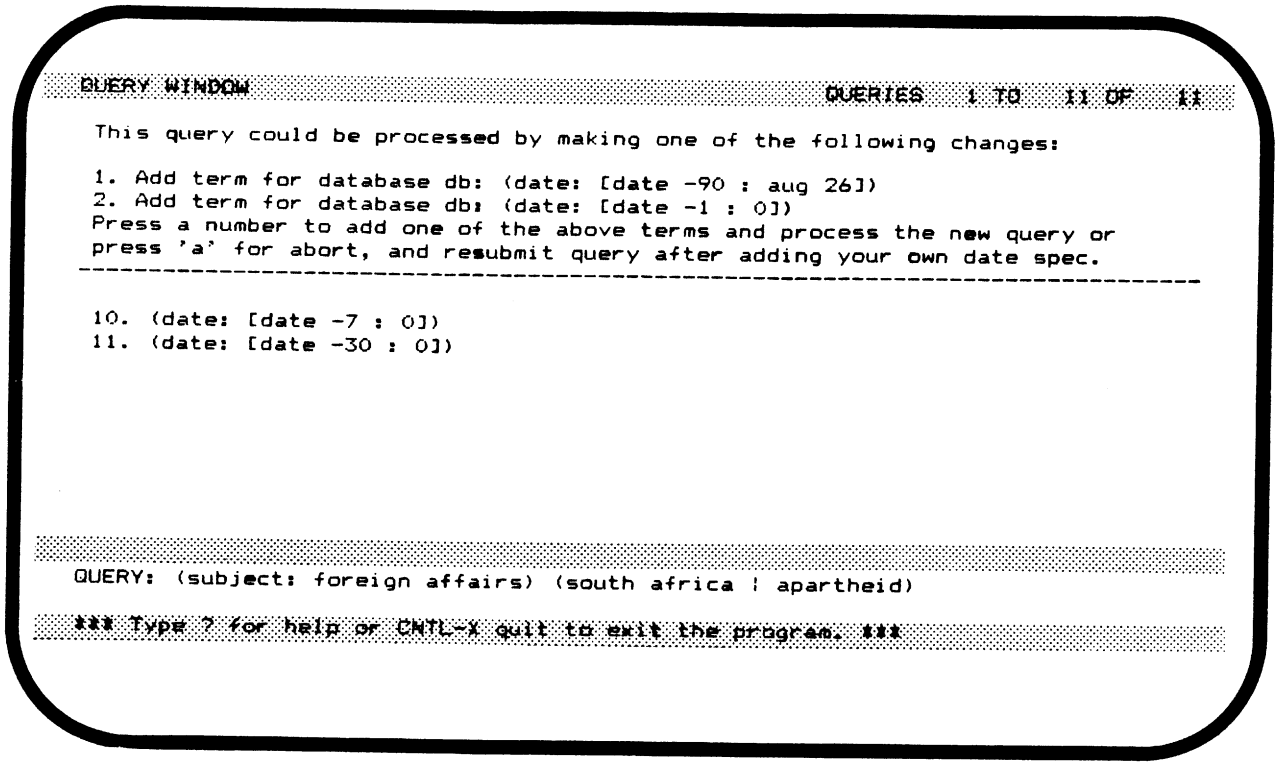


Figure 1-2: The Information Window

window name: **<Esc>-q** displays the query window, **<Esc>-s** displays the summary window, **<Esc>-a** displays the article window and **<Esc>-h** displays the help window. The name of the window being displayed is always indicated on the header line.

Whenever there is more than one screenful of text, you can scroll forward and backward through the text a screenful at a time by pressing **<Ctrl>-v** and **<Esc>-v** , respectively. You can scroll forward and backward one line (or one summary) at a time by pressing **<Ctrl>-n** and **<Ctrl>-p**, respectively. You can also go directly to the top of a window by pressing **<Esc>-<**, or the bottom of a window by pressing **<Esc>->**.

1.3 — The Query Window

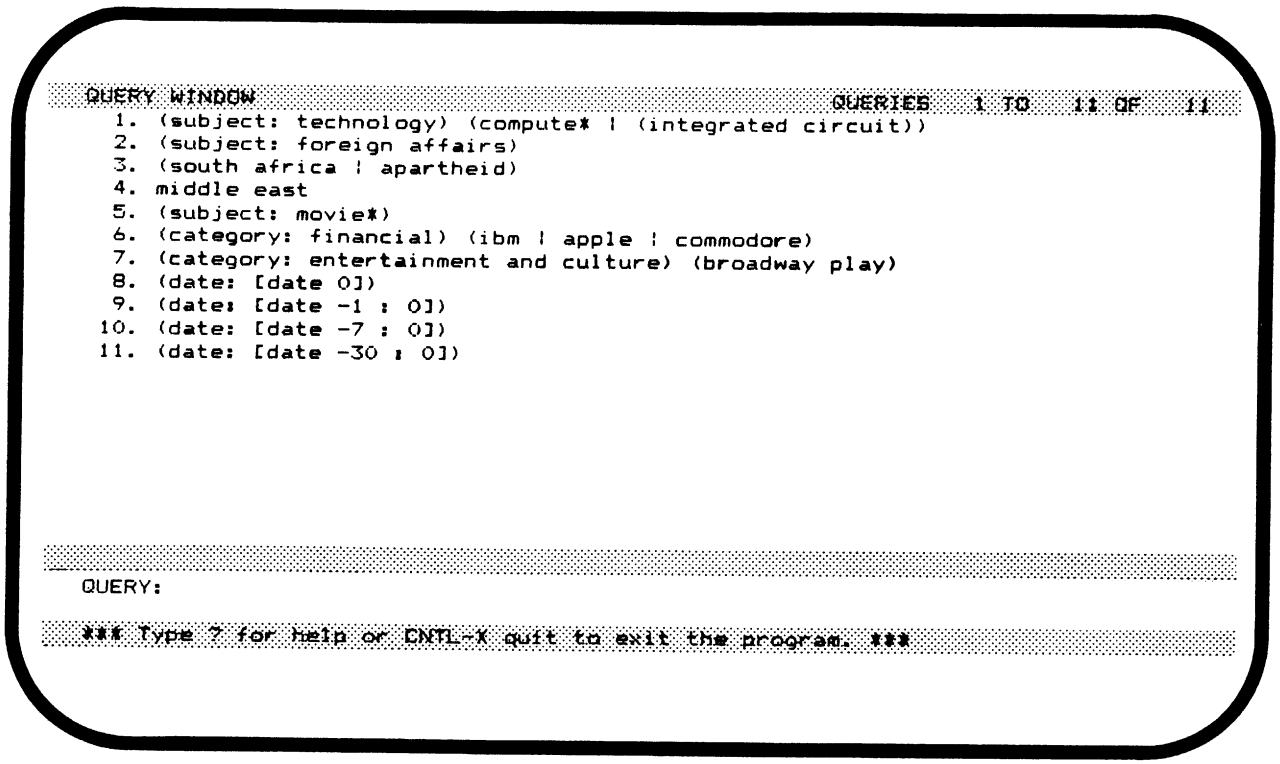


Figure 1-3: The Query Window

The query window (see Figure 1-3) displays your personalized query list, if you have one. Your query list makes it easy to submit your favorite queries with a minimum of typing. Before we discuss the creation and manipulation of the query list, let's turn our attention to queries and how to submit them to the database.

1.3.1 — Submitting Queries

A query is an expression consisting of words, parentheses (which are used for grouping) and *connectives*. (see the section entitled *The Query Language* for full description of the query language). A typical query might be:

```
reagan budget & (not (defense | military))
```

This query, which can be read as "reagan, budget *and not* defense *or* military", matches all articles that contain the words "reagan" and "budget" and neither the word "defense" nor the word "military". There are three connectives: logical conjunction (which can be written "&" or "and"), disjunction (which can be written "|" or "or") and negation (which can be written "~" or "not"). A space between two words is interpreted as a conjunction, or logical **and**, in this case, the space between the words "reagan" and "budget".

To submit a query, simply type it on the query line and press carriage return. To abort a query, press **<Ctrl>-g**. If you submit a new query while *Walter* is still processing an earlier query, the earlier query is aborted and the new query will be processed.

1.3.2 — Query Routing

An interesting feature of *Walter's* querying facility is its ability to route a query to a database where it can best be answered. This feature is called *query routing*. In order to route queries, the program includes a list of database content descriptions. When you submit a query, it compares the query against the descriptions of each of the databases and determines where (if anywhere) your query can be answered. If the query cannot be answered anywhere, a modification to the query may be suggested so that it will be able to be processed by one of the available databases. In particular, if you submit a query without a date specification, *Walter* suggests several ways of confining the query to specific dates.

Walter offers to extend queries by listing a menu of options via a pop-up information window (See Section 1.2). You can choose one of the options by typing the appropriate number and your query will be modified accordingly. The modified query will be processed automatically.

If you decide you want to change the query yourself instead of choosing an option provided in the information window, you can type "a" for abort. You may now edit the query and submit it again. If you submit a query that cannot be answered anywhere (such as `reagan & ~reagan`), the query router will generate a warning message.

1.3.3 — Creating and Manipulating Your Query List

Walter allows you to put together and save a list containing your favorite queries. If you are a new user, your query list is initially empty. To add a query to the list, type the query on the query line and use the *insert* command. This command adds the query from the query line to the query list.

If you want to process a query that appears on the query list, or if you want to use such a query as part of a larger query, you can use the *fetch* command to fetch the query from the list and place it on the query line. *Fetch* takes as its argument a text string and interprets any numbers in the string as references to queries in the query list. *Fetch* replaces each number by the corresponding query, and places the result on the query line. For example, if the argument is

```
4 & (2 | congress)
```

fetch forms the query that will match all articles matching the fourth query and also either matching the second query or containing the word `"congress."` This query is then placed on the query line at the cursor position. For example, if the query list is as shown in Figure 1-3, the following string would be put on the query line:

```
middle east & ((subject: foreign affairs) | congress)
```

You may also delete queries from the query list, undelete them (if you change your mind) and expunge the deleted queries (when you are sure you do not want them anymore). You can delete one or more queries by specifying the numbers of the queries, separated by commas, as arguments to the *delete* command. Deleted queries are marked by an asterisk. You can undelete one or more queries in the same way with the *undelete* command. The deleted queries are not removed from the query list until you execute the *expunge* command or exit the program.

Walter keeps your query list in a file called .querylist. When the program is started, it looks for the .querylist file in your home directory and in your working directory (in that order). Likewise, when you exit the program, it will save the .querylist file in the directory where it found it, or in your home directory if no file was found when the program was started.

Note: Changes to the query list, such as insertions and deletions, are not permanent unless you exit from *Walter* via the *quit* command and confirm the saving of changes. If you exit by typing Cntl-c or Cntl-s, any changes you made will be lost.

Note: *Fetch* does not check whether the text string it produces is a legal query. Also, *fetch* blindly interprets all numbers in its argument as references into the query list. So, the following argument to *fetch*

```
(date: [date -5 ])
```

would result in the text string

```
(date: [date -(subject: movie*) ])
```

which is a meaningless query.

1.4 — The Summary Window

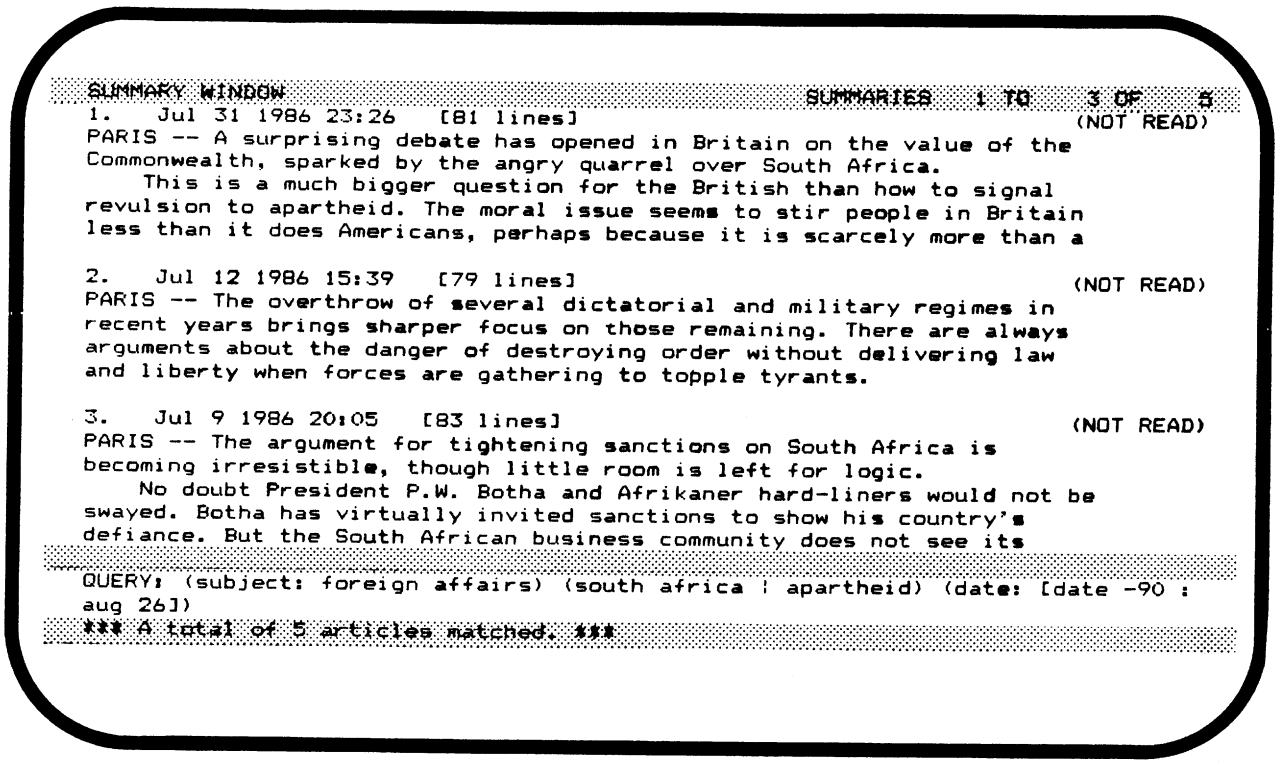


Figure 1-4: The Summary Window

The summary window (see Figure 1-4) displays summaries of all the articles that matched a query. A summary is composed of five parts:

- the *summary number*, in the left-most column, is the index of the summary in the list of summaries;
- the *date*, to the right of the summary number, is the date and time associated with the article;
- the *number of lines*, to the right of the date, is the number of lines of text in the article corresponding to the summary;
- the *read flag*, to the right of the number of lines, indicates whether or not the article has been displayed in the current session;
- the *summary text* is the first few lines of text of the corresponding article.

When you submit a query, *Walter* forwards the query to the database, and receives the requested information. Initially, messages indicating the number of articles that match the query are displayed. As soon as the first summary is received, the program displays the summary window and displays the summaries as they arrive. At this point, the right hand side of the header line shows how many summaries have been received.

If you want to read the article corresponding to a given summary, you can ask *Walter* to retrieve an article from the database with the *read* command, which takes a summary number as its argument. You may issue this command as soon as the summary in question appears in the summary window. If summaries are still being received, the program will suspend receiving summaries, retrieve the article, and display it in the article window. After the full article is received, the remaining summaries will be

received in the background.

1.5 — The Article Window

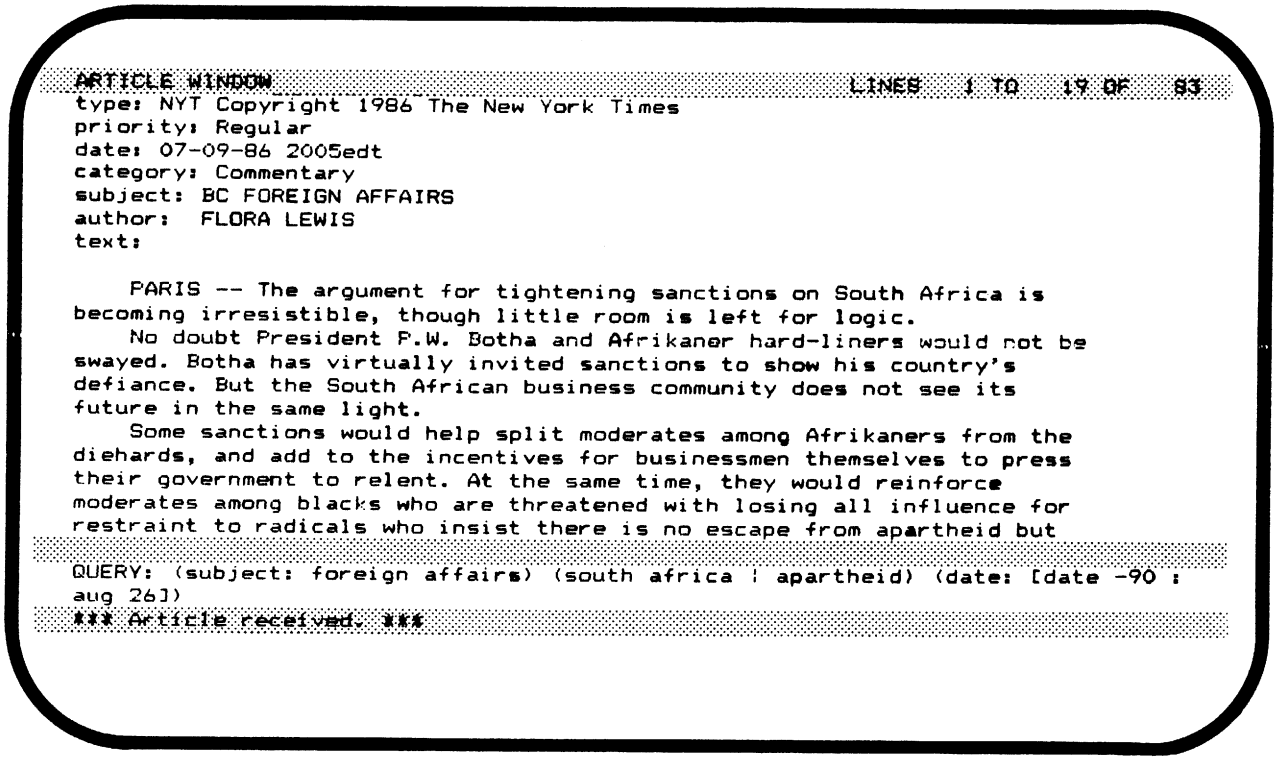


Figure 1-5: The Article Window

The article window (see Figure 1-5) displays an article after the article has been retrieved using the *read* command. An article consists of a number of fields, as described in Section 2.

When you submit a read command, the database forwards the requested article. As soon as *Walter* receives the first line of the article, the article window displays that line, and the incoming article is displayed as it arrives. You can save an article by means of the *save* and *append* commands. The *save* command prompts for a file name and places the article into the given file, overwriting whatever already exists in the file. The *append* command prompts for a file name also, but it appends the text of the article to the end of the given file.

1.6 — The Help Window

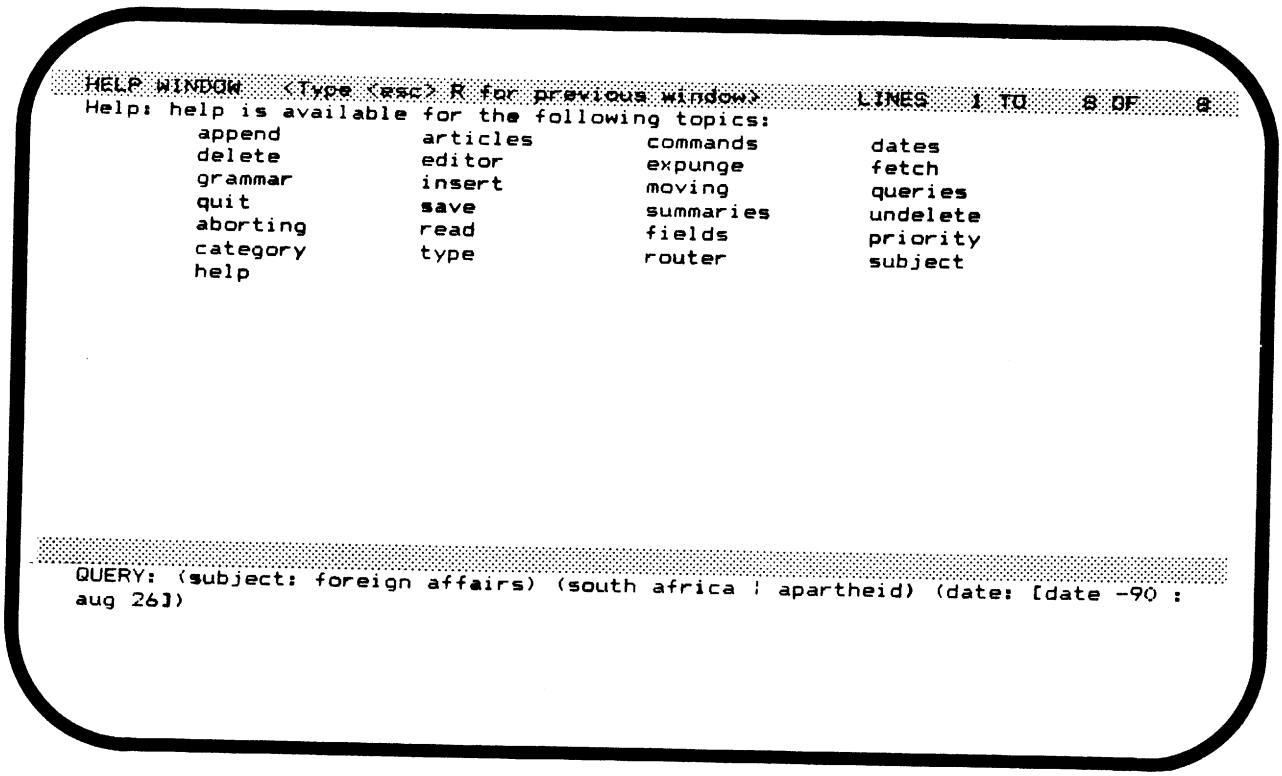


Figure 1-6: The Help Window

The help window (see Figure 1-6) displays help text in response to the *help* command or the *quick help* command.

Quick help is displayed whenever you press a '?'. It gives a brief overview of *Walter*, including how to submit queries, how to read articles and what commands are available. The purpose of *quick help* is to quickly remind you of the available features, rather than reading a detailed description of these features. You can ask for detailed help on a particular topic by executing the help command and specifying a topic name. If you type `help` as the topic name, *Walter* displays a list of the topic names for which help is available. If you type * as the topic name, *Walter* displays *all* of the help text available. If you do not specify a topic, *quick help* is displayed. To return to the window you were in before you invoked the *help* command, press **<Esc>-r**.

Section 2 — Databases and Services

Currently, *Walter* has access to the text of the *New York Times* for the past three months. We anticipate that other database services will be added in the future.

The *New York Times* service includes the complete text of the articles that appear in that newspaper. In general, the *New York Times* provides highly polished and readable news stories, corresponding to the printed version of the *New York Times*. Developing news stories are usually updated once a day.

2.1 — Fields

All articles are in the same standard format: an article consists of several *fields*, one of which, the TEXT field, contains the main text of the article. Section 3 describes how to take advantage of the information in the various fields when you form queries. The fields of an article are defined below.

TYPE The TYPE field of an article identifies the source of the article. Currently, the only source is the *New York Times* news wire, `"nyt"`. Other sources, such as the *Associated Press* news wire, `"ap"`, may be supported in the future.

DATE The DATE field indicates the date the news wire transmitted the article.

CATEGORY The CATEGORY field identifies the general subject area of an article. The categories used by the *New York Times* editors for their new articles are described in the next subsection.

AUTHOR The AUTHOR field identifies the author of an article. This field is not always present. Occasionally, the author of a news article is identified by a by-line in the TEXT field of the article.

PRIORITY The PRIORITY field indicates the priority of the article. The PRIORITY field contains one of the following words or phrases:

`flash`	the highest priority; it is seldom used
`bulletin`	the priority level of prime news; it is also used for corrections, updates, and occasional retractions of previous stories
`urgent`	the priority level of important news
`regular`	the priority level of routine news
`deferred`	the lowest priority level for news
`Weekday advance`	material for a future weekday
`Sunday advance`	material for a future Sunday
`reruns`	an article that was already published at least one other time

SUBJECT The SUBJECT field contains a subject identifier that is assigned by the originating information source. Certain subject identifiers are used consistently; some of these are listed in Section 2.3. Some news articles are actually listings of upcoming news articles, and they give the subject identifier for each upcoming article (for example, "REAGAN-SPEECH"). If you are interested in such an article, you can submit the query `(subject: "reagan speech")` to *Walter* and, if the article has been received, the program will retrieve it.

TITLE The TITLE field indicates the title of the article. This field is not always present.

TEXT The TEXT field contains the text of the article itself.

2.2 — Categories

As will be explained in *Section 3 — The Query Language*, a query of the form `(category: <something>)` matches articles whose CATEGORY field contains *<something>*. The CATEGORY field is very useful, because the *New York Times* editors put every news article in an appropriate category. The following is a complete list of the categories used.

`Advisories`	Listings of upcoming articles
`Commentary`	Editorials and columns
`Domestic`	Domestic news, except for news from Washington D.C.
`Entertainment and Culture`	Articles on entertainment, culture, and the arts, including reviews of movies, plays, books, and television programs
`Financial`	Business news, financial indicators, and some stock market information
`International News`	News about foreign events, including articles originating at the United Nations headquarters
`Lifestyle`	Social news
`Presidential Election`	Election coverage (when applicable)
`Sports`	Sports features and scores
`Standing`	Feature articles that are non-urgent
`Travel`	Travel information and human interest stories involving travel
`Unknown`	All other categories of articles
`Washington News`	News concerning national politics, the United States government, actions of the President, and congressional decisions

2.3 — Subjects

Our information sources provide a variety of information, such as news summaries, on a regular basis. Information of this kind is identified using the SUBJECT field; for example, the query `(subject: newssummary)` matches the *New York Times* news summary.

The following subject identifiers are used consistently by our information sources:

`aboutcars`	Articles on automobiles
`advertising column`	Regular article on advertising
`advisory`	Listings of upcoming articles or available services
`anderson column`	Dave Anderson's sports commentary
`baker column`	Russell Baker's commentary
`bank*`	Assorted articles on banking, including regular features
`baseball`	Assorted articles on baseball
`berkow column`	Sports commentary
`bestsellers`	Hardcover bestseller list
`biz*`	Business articles, including business week, health, law and people
`book review`	Reviews of new books
`booktalk`	Column about books
`boxing`	Columns about boxing
`brf`	Briefing on Washington news and political events

`briefing`	Briefing on Washington news and political events
`budget`	Listings of articles for upcoming publication
`campaign*`	Articles on the presidential campaign
`careers`	Features on interesting careers
`claiborne`	Cooking with Craig Claiborne
`colleges`	Stories on colleges, such as sports
`column`	Columns from the *New York Times*
`commodities`	Articles on the commodities market
`comput*`	Various columns about computers, computing, etc.
`consumer notes`	Consumer advocacy column
`consumerrates`	Consumer credit rates
`credit`	Reports on commercial credit markets
`credit rates`	Present commercial credit rates
`databank`	National economic indicators
`econ`	Various economic articles `editl`
	Editorials from the *New York Times*
`fashion*`	Articles about fashion
`finbriefs`	Collections of short news items about business and finance
`findigest`	A daily digest of top financial news from the *New York Times*. This digest is identical to the business digest on the front page of the *New York Times* business section.
`followups`	Follow up on the news
`foreign affairs`	Foreign affairs column
`frontpage`	A daily description of the layout of the front page of the *New York Times*
`glass column`	Andrew Glass' commentary
`homevideo`	Recent home videotape releases
`investing`	Assorted articles on investments
`kisselgoff dance`	A weekly column on dance
`lewis column`	Anthony Lewis' column `market`
	A daily report on the stock market
`marketplace`	Wall street news
`matchups`	Pre-game analysis of upcoming sporting events
`mortgage`	Articles about mortgages
`movie*`	Movie reviews and features about movies, including columns by Maslin and Canby
`movie notes`	Movie column
`movie review`	Movie reviews `music`
	Various columns on different types of music
`nba`	Assorted articles on the NBA
`newssummary`	A capsule summary of top news stories from the *New York Times*. The *New York Times* news summary comes out once a day around midnight, and is identical to the news summary that appears on the second page of the first section of the printed paper. Use the query `(type: nyt) (subject: newssummary)` for this summary.

`obit*`	Obituaries
`on language`	Column by William Safire on the English language
`outdoors`	Column on outdoor activities
`patents`	Patents of special interest that have been recently issued
`paperbacks`	Paperback bestseller list
`personal health`	Column on personal health `prospects`
	Financial prospects for particular business sectors
`questionbox`	Sports questions and answers
`quindlen column`	Anna Quindlen's regular column
`racing`	Horse racing news
`reston column`	James Reston's column
`rosenthal`	Andrew M. Rosenthal's commentaries
`russell art`	John Russell's art column
`safire column`	Column by William Safire that is not "On Language"
`science q a`	Science questions and answers
`science watch`	Short summaries of recent scientific news
`scouting`	Stories on sports players
`silk column`	Financial commentary by Leonard Silk
`60 minute gourmet`	What to do for dinner in one hour
`ski`	Column on skiing
`sports column`	Various sports columns from SportsMonday
`tabletalk`	Food column
`tax column`	A column on tax advice
`theater`	Articles about and reviews of the theater
`trade`	Articles about US Trade
`travel notes`	Vacation ideas
`travel q a`	Travel questions and answers
`tv review`	Reviews of upcoming television programs
`tv weekend`	Weekend television
`vecsey column`	Sports column
`video`	Column on home videotapes
`wicker column`	Tom Wicker's column
`wine*`	Articles on wine, including Wine Talk

Section 3 — The Query Language

The queries you submit to *Walter* are written in a simple language called the *query language*. Each query matches a certain set of articles. You can describe what articles you want to see in terms of their subject, author, contents, date of publication and so forth. In this section, the query language is explained.

3.1 — Query Grammar

The simplest possible query is a single word. Imagine, for example, that you are interested in all articles that contain the word "peace". The corresponding query is the word `peace`. If you are interested in all articles that contain the words "war" and "peace", the corresponding query could be `war and peace`, `war & peace`, or simply `war peace`. Thus, you can combine two simple queries into a more specific query by means of `and`, `&`, or concatenation.

Next, suppose that you are interested in all articles about computers, computing, computation, and so on. To request all these articles, use the query `comput*`: it matches all articles that contain one or more words beginning with the letters "comput". These queries are called *stem queries*, because they match any word that begins with a specified *stem*. A stem query must be at least two characters long, not including the asterisk.

Now, imagine that you are interested in articles by a certain author, say John Smith. You could use the query `smith`, but it matches every article that *contains* the word "smith", regardless of the context in which it appears. To rule out all these extra articles, use the query `(author: smith)`: it matches all articles that have an AUTHOR field containing the word "smith". Queries of this kind are called *field-specific queries*; the possible fields (besides AUTHOR) are listed in Section 2.1. Other examples of field-specific queries include `(subject: newssummary)` and `(category: washington news)`.

Above, we showed how to combine two queries into a more specific query by means of `and`, `&`, or concatenation. This combining rule is very useful in conjunction with field-specific queries. For example, the query `mets (category: sports)` matches all articles that contain the word "mets" and have a CATEGORY field containing the word "sports".

3.2 — Dates

The database contains old news as well as recent news, so it is important to specify the date or dates of the articles that interest you. In fact, *Walter* not only deals with dates, but also with *ranges* of dates, such as `"oct 15 : nov 15"`. *Walter* recognizes three kinds of dates:

- *Specific* dates of the form [date <month><day><year>] as in the date [date Jan 8 1986]. Note that if the year is omitted, it defaults to the current year. If the day is omitted, it defaults to the first day of the given month. The month may not be omitted.
- *Relative* dates of the form [date -n] as in the date [date -7], where n is some number. This form effectively means "n days ago"; for example, [date -7] means "7 days ago".
- *Date ranges* of the form [date <date>:<date>] as in [date oct 15 : nov 15], matches any date between the two dates given, inclusive. The given dates in the range can be in either specific or relative form. For example, the date range [date jul 5 1986 : 0] would match any date between July 5, 1986 and the current date.

Dates are usually used in connection with the "date" field, which contains the date of the article. For

example, the following query matches all articles containing the word `mets` that appeared on the *New York Times* news wire the day before:

```
mets & (type: nyt) & (date: [date -1])
```

You can also combine dates to find, for instance, all articles that mention sushi and were published either today or exactly one or two weeks ago:

```
sushi & (type: nyt) & (date: [date 0] | [date -7] | [date -14])
```

If no field specifier is present, as in the query `war & peace`, then *every* field is searched for the given keywords. So, for instance, the query

```
(subject: movie*) streep
```

matches all articles that have the stem `movie*` in their subject fields and the word `streep` in any field of the article.

3.3 — Case and Punctuation

The following convention applies to all queries: uppercase and lowercase letters are considered the same, and queries may not contain punctuation characters. To compensate for this, all punctuation characters in *articles* are treated as spaces, except for apostrophes, which are ignored. Thus, the query `rogers` matches the following words: "rogers", "Rogers", "Roger's", "Rogers'", and so forth.

3.4 — Useful Queries

In order to be processed, a query must precisely specify the information desired. For example, the query `reagan` is non-sensical, because *Walter* can not produce every article that was ever published that includes the word `reagan`. Instead of rejecting queries that are not well defined, *Water* automatically narrows such queries to match the databases on hand. If you submit a query that does not include a date specification, *Walter* pops up an information window and offers you some date terms to choose in order to appropriately define the query. Since a date term is required, it is often handy to keep a few commonly used date queries in your query list, as is illustrated in Figure 1-3 (queries 8 through 11). In this way, you can append a date query to your query with the fetch command — this expedites query processing and gives you more control over what your query will retrieve.

We suggest that you keep a selection of four or five general date queries in your query list, for example:

- (date: [date 0]) all articles written today
- (date: [date -1 : 0]) all articles written in the past two days
- (date: [date -7 : 0]) all articles written in the past week
- (date: [date -30 : 0]) all articles written in the past month

Appendix I — How to Become an Authorized User

In order for you to use *Walter*, you need to be authorized as a user of one or more of the databases that the program has access to. To become an authorized user, contact the Programming Systems Research Group by sending computer mail to `walter-request@cls.lcs.mit.edu`. You will need to sign a legal agreement (see Appendix IV) before we can make you an authorized user.

Appendix II — Installing Walter

If you have a VAX, Sun workstation, or IBM PC-RT running UNIX (tm) and you wish to receive *Walter* for installation on your machine, contact the Programming Systems Research Group by sending electronic mail to:

`walter-request@cls.lcs.mit.edu`

The entire *Walter* system consists of an executable file and three text files. To install the system, put the executable file in a public directory that users on your machine usually have in their path (e.g. /usr/local/bin or some such directory). The three text files are walter.help, walter.quick_help and walter.auth. All three of these files need to be in the directory /usr/lib.

The executable file is approximately 450 kilobytes large and the three text files are collectively 35 kilobytes in size.

Appendix III — Summary of Commands and Keystrokes

III.I — Editing Keys

Ctrl-a	move cursor to the beginning of the line
Ctrl-b	move the cursor backward by one character
Ctrl-d	delete the character under the cursor
Ctrl-e	move the cursor to the end of the line
Ctrl-f	move the cursor forward by one character
Ctrl-k	kill the line from the current cursor position until the end
Ctrl-w	kill the entire line
Esc-b	move the cursor backward by one word
Esc-f	move the cursor forward by one word
Esc-delete	delete the word before the cursor
delete	delete the character before the cursor

III.II — Display Keys

Ctrl-l	redraw the entire screen
Ctrl-n	scroll forward by one line or summary
Ctrl-p	scroll backward by one line or summary
Ctrl-v	scroll forward by one screenful
Esc-a	display the article window
Esc-h	display the help window
Esc-q	display the query window
Esc-r	return from the help window to the previous window
Esc-s	display the summary window
Esc-v	scroll backward by one screenful
Esc-<	scroll to the top of the window
Esc->	scroll to the bottom of the window

III.III — Function Keys

Ctrl-c	exit the program without saving any changes
Ctrl-g	abort the current query or command
Ctrl-x	put the cursor on the command line
Ctrl-z	stop *Walter*
?	display the quick help message
<space>	on the command line, invoke command completion
<return>	execute command or process query

III.IV — Alphabetical Listing of Commands

APPEND	append the current article to a file (see section 1.4)
DELETE	delete one or more queries from the query list (see section 1.2.3)
EXPUNGE	expunge the deleted queries from the query list (see section 1.2.3)
FETCH	fetch queries from the query list (see section 1.2.3)
HELP	display help for the given topic (see section 1.5)
INSERT	insert a query into the query list (see section 1.2.3)
QUIT	quit the program, prompting for saving of changes (see section 1.2.3)
READ	read the article corresponding to a given summary (see section 1.3)
SAVE	save the current article in a file (see section 1.4)
UNDELETE	undelete one or more queries in the query list (see section 1.2.3)

Appendix IV — Your Responsibilities

Your Responsibilities Regarding Use of the Information Received

To conduct this research project in the distribution of news and information, MIT has entered into understandings and legal agreements with the Associated Press, the New York Times Company and Mead Data Central, Inc. As an authorized user in this project, you have agreed

1. not to make more than one copy — in any form — of any information received;
2. not to retain copies — in any form — of more than an "insubstantial part" of the information received;
3. not to transfer any of the information received — in any form — to third parties, whether or not for profit; and
4. not to retain any of the information received from the *New York Times* — in any form — for more than 90 days.

Because we believe *Walter* is an extremely useful and interesting program to use, we feel that the responsibilities outlined above are reasonable. If you do not act in keeping with the letter *and* the spirit of the agreement, we will be forced to discontinue your status as an authorized user of *Walter*.

Full Name (printed): _____

Legal Agreement with Mead Data Central, Inc.

The above designated "Recipient" agrees that it will not retain any copies of data from the New York Times in any form, whether print, machine readable or otherwise for a period of more than 90 days, and that any such data retained will be 1) limited to single copies, 2) never consist of more than an insubstantial part of the entire database made available to MIT, and 3) will not be transferred, whether or not for profit to any third party. It is agreed that all copies of such data will be returned or destroyed earlier than the 90-day maximum retention period upon written request from MIT, The New York Times Company, or Mead Data Central, Inc.

Agreed

_____ _____

Signature of Recipient Date

Understanding and Agreement with MIT

Whether or not for profit, I understand and agree not to transfer in printed or electronic form any information received via the Boston CommInS project to any third party.

I understand and agree that I must complete and return a questionnaire provided for that purpose to the Boston CommInS Project once a month or else have to return the receiver and software to the project.

I understand and agree that the software and hardware will remain the property of MIT and that I must return all the materials used in this test of the Boston CommInS system either (i) at the end of the experiment in approximately 12 months or (ii) upon written notice from the project -- whichever comes first.

I understand and agree that MIT has made no expressed nor implied guarantees to provide this information service without interuption throughout the duration of the experimental test.

Agreed

_____ _____

Signature of Recipient Date

Figure IV-1: Text of Agreement

This blank page was inserted to preserve pagination.

CS-TR Scanning Project
Document Control Form

Date : 3/23/95

Report # LCS-TR-399

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 31(38-IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:
☒ DOD Form ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☒ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:

IMAGE MAP (1-2) UNNUMBERED TITLE & USER MANUAL PAGES
(3-4) PAGES #'ED i, ii
(5-31) PAGES #'ED 1-27
(32-35) SCANCONTROL, COVER, PRINTERS NOTES, DOD
(36-38) TRGTS

Scanning Agent Signoff:
Date Received: 3/23/95 Date Scanned: 3/30/95 Date Returned: 3/30/95

Scanning Agent Signature: Michael W. Cook

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MIT/LCS/TR-399	5. MONITORING ORGANIZATION REPORT NUMBER(S) N00014-83-K-0125
6a. NAME OF PERFORMING ORGANIZATION MIT Laboratory for Computer Science 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research/Department of Navy
6c. ADDRESS (City, State, and ZIP Code) 545 Technology Square Cambridge, MA 02139	7b. ADDRESS (City, State, and ZIP Code) Information Systems Program Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA/DOD 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Walter User's Manual (Version 1.0)

12. PERSONAL AUTHOR(S)
Gifford, David K., Cote, Robert G., and Segal, David A.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	1987 September	27

16. SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Walter, database server, user interface, remote pipe and procedure model, query routing, full-text, database

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Walter is a UNIX program that provides access to databases located at MIT via the DARPA Internet. The databases provided by Walter include the full-text of the New York Times for the past 90 days. A sophisticated full-text query language is provided, and Walter uses a query routing algorithm to direct requests to the proper database server at MIT. Walter was built as an experimental test of query routing and the remote pipe and procedure model for distributed communication.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Judy Little, Publications Coordinator	22b. TELEPHONE (Include Area Code) (617) 253-5894 22c. OFFICE SYMBOL

*U.S. Government Printing Office: 1985—607-047

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 3/30/1995

M.I.T. Libraries
Document Services