U S L / D B M S   N A S A / R E C O N

W O R K I N G   P A P E R   S E R I E S

Report Number

DBMS.NASA/RECON-11

The USL/DBMS NASA/RECON Working Paper Series contains a collection of reports representing results of activities being conducted by the Computer Science Department of the University of Southwestern Louisiana pursuant to the specifications of National Aeronautics and Space Administration Contract Number NASW-3846. The work on this contract is being performed jointly by the University of Southwestern Louisiana and Southern University.

For more information, contact:

Wayne D. Dominick

Editor
USL/DBMS NASA/RECON Working Paper Series
Computer Science Department
University of Southwestern Louisiana
P. O. Box 44330
Lafayette, Louisiana 70504
(318) 231-6308

AN OVERVIEW OF SELECTED INFORMATION STORAGE AND RETRIEVAL ISSUES IN COMPUTERIZED DOCUMENT PROCESSING

Valentine U. Ihebuzor

Computer Science Department,
University of Southwestern Louisiana,
Lafayette, Louisiana.

December 29, 1984

ABSTRACT

The rapid development of computerized information storage and retrieval techniques has introduced the possibility of extending the word processing concept to document processing. A major advantage of computerized document processing is the relief of the tedious task of manual editing and composition usually encountered by traditional publishers through the immense speed and storage capacity of computers. Furthermore, computerized document processing provides an author with centralized control which is a handicap of the traditional publishing operation. A survey of some computerized document processing techniques is presented with emphasis on related information storage and retrieval issues. String matching algorithms are considered central to document information storage and retrieval and are also discussed.

ACKNOWLEDGEMENT

The author wishes to acknowledge the assistance of Dr. Wayne D. Dominick of the University of Southwestern Louisiana for making this work possible, and Dr. Tim M. Merrett of McGill University for introducing the author to some interesting topics in computerized text processing.

TABLE OF CONTENTS

AN OVERVIEW OF SOME INFORMATION STORAGE AND RETRIEVAL ISSUES IN COMPUTERIZED DOCUMENT PROCESSING

1. INTRODUCTION

Word processing applications with the aid of computers are almost becoming commonplace. Most applications have however concentrated on the use of text editors and text formatters to manipulate text files in order to produce a desired output which is formatted according to the needs of the user. Thus word processing applications are essentially input/output techniques which take advantage of the immense power, speed and storage capacity of computers to relieve the tedious problem of manual editing and composition.

The rapid development of information storage and retrieval techniques has introduced the interesting possibility of extending the word processing concept to document processing. Until recently, document processing has entirely been a manual craft involving specialists like Editors, Proof readers, Graphic designers, Typographers and other publishing professionals. Several document preparation systems have recently been developed for use by writers of technical documents and by publishing houses. A few of these have been specialized to address problems of a specific kind while others have been general to provide answers to a wide range of document preparation problems. However, the goal of complete computerization of all document preparation problems is still to be met as some of the related semantic problems are not amenable to computerization.

In this report a discussion of some IS&R issues in document processing will be presented. The approach will be to survey some of the fairly recent material in the literature with a view to identify topics that relate to information storage and retrieval. Specifically, the topics to be discussed include:

* Functional and Design Requirements of a Document Preparation System
* Document Specification Methods
* Data Base Characteristics and Content
* Support Environment for a Document Preparation System,
* Data Structures for Record and Storage Management
* Algorithms for String Matching.

2. FUNCTIONAL AND DESIGN REQUIREMENTS

The main objective of this report is to present a description of some functional components of a document delivery system. The focus will be on a system that would provide comparable services to those provided by standard publishing houses. In order to carefully define the functional objectives of such a system, it is necessary to start by examining the functional attributes of a traditional publishing house. Figure 2.1 shows the information flow schematic in a traditional publishing operation. A major problem of the scheme is the removal from the author of control of the various stages in the production of the finished product. The intermediate stages in the diagram represent stages in which the author of the book is not involved. All the author does is to provide a rough type written manuscript to an editor (an expert in document design) who after making some changes submits the manuscript to a typesetter who produces typset galleys from the manuscript. Proof reading of the galleys against the original then follows. The next stage is that of pagination where the galleys are cut into page sizes and the addition of figures, footnotes as well as page numbers and perhaps some cross reference material takes place. Indexing then follows as the material is then sent to an indexer who produces an index from the available material. The index is the sent for typesetting, and later added at the end of the book, which is then printed [Reid, 80].

At each stage of the process, errors, possible enhancements, observed inconsistencies, cause the material to be recycled. The addition of new material (possibly as a result of current information on the subject matter) lead to numerous problems and great expense of changing page layouts, footnotes, cross references and indexes. Because the author is not involved in most of these production stages, there are inherent problems of misunderstanding and even errors. Sometimes communication problems cause unnecessary delays when the production stations exist in remote locations or even different states or countries. Typically, the time lapse between the completion of the author's manuscript and the publishing of the finished product is on the order of months or even years and potential inconsistencies may potentially lead to author frustration; also revision may become a continous necessity since the time lag causes the material to always need revision as new research developments overtake the current contents of the document during the lengthy production time.

A major disadvantage of the traditional set up is the absence of centralization. By using a computer, the processes which were hitherto handled manually can be automated and the management of the various components can then be achieved by means of a computer. Figure 2.2 illustrates this scheme. This scheme represents an improvement over the scheme of Figure 2.1 in the sense that the entire production process can be managed centrally and the production time frame can be greatly reduced by the enormous power and speed of a computer. However the responsibility for the final appearance of the document is still removed from the author because of the use of external "expert" software packages to handle document design, typographic design and document layout. These packages are designed externally(possibly at different places) and applied to the document sequentially via the computer in a stereotype fashion. Besides the added speed due to computerization, the scheme has obvious similarities with the traditional scheme in that once the author submits the manuscript, he/she has no more control over the appearance of the finished document. Some of the problems of the traditional setup will also arise.

In order to provide the author with the required level of control over the final appearance of the document, it is desirable to provide mechanisms which enable the author to control the document design process, the typography and the layout. This means that the author must be provided with the tools to control the parameters which control all the document design processes. The schematics of this approach are shown in

Figure 2.3 where the author is the source of all design specifications for the document. Most of the initiative is the author's. The idea is that a writer prepares a manuscript that is represented in a document preparation language. This manuscript is processed by the system into a finished document. Some measure of competence is required of the author regarding document preparation as well as the knowledge of the syntax of the document preparation language. In such a system, desired features for output control are parameterized and are available to the author. In some systems, e.g., SCRIBE [Reid, 80], the information system draws on a database of format specifications that have been produced by a graphic designer to produce a document that contains the author's text in the designer's format. In other cases, a user must associate explicit structural description with each document [Walker, 81].

In order to provide more power to the latter system, a support environment with an interactive work station is required. This corresponds to the document editor. According to Janet Walker [Walker, 81], ideally a document editor manipulates a data structure that is a canonical representation of a structured document. The actual representation of the document is parsed into the internal form as an import operation when the document is loaded into the document editor. Upon exit, the editor unparses the internal form back into the appropriate document preparation language. The use of these parsing techniques makes it possible to design document editors which maintain the structure of each language independently of the language in which the structure is represented. Such an editor allows for both structure editing and text editing operations; the structure editing operations work not on the textual representation but on the document structure itself. Adding a new document language consists of writing the parser and unparser for that language. Users can choose to view the text editor either as a text editor or a structure editor. Text editing commands work as if the internal representation were text and structure editing commands operate directly on the internal representation. Conceptually, the document editor is a multiprocess executive whose top level is a text editor. The text editor provides the command interface for the rest of the system and also edits the text. The text editor integrates a set of document editing tools into a unified environment and provides the command interface and help facility for them. Several editors with these capabilities have been designed and implemented. They include EMACS [Stallman, 81], ETUDE [Good, 81], ED3 [Stromfors, et al., 81] and PEN [Allen, et al., 81].

Only a few systems have been generalized enough to provide many answers to the problems of computerized document design. Notable examples include: TEX [Knuth, 78], SCRIBE [Reid, 80] and

ETUDE [Hammer, et al., 81]. JANUS [Chamberlain, et al., 81], a proposed system, is intended to provide support for authors of complex documents containing mixtures of text, line art and tone art by providing them with immediate feedback and direct electronic control over page layouts, using a special 2-display workstation. Knuth's TEX is designed for the production of documents containing large amounts of mathematical notation; it represents an expansion over EQN [Kernighan, et al., 75] which is a system for typesetting mathematics.

Several editors and formatters have also been developed for a variety of specific applications. PEN - a hierarchical document editor, is directed at documents having a significant amount of mathematical notation; it is an interactive formatter which provides a concise notation for specifying objects. ED3 provides a powerful tool for data editing in interactive systems through a combined ability for handling hierarchical structures and screen oriented text editing. EMACS an Extensible, Customizable, Self Documenting Display editor is a display editor which is implemented in an interpretive high level language. The main attribute of EMACS is its high extensibility which gives users the ability to add or define new editing commands or to modify the existing commands to suit their editing needs. The EMACS provides a suitable environment for document editor design by taking advantage of its extensibility.

Several text formatting systems (programs) which provide word processing capability have been the direct descendants of RUNOFF. These include ROFF, ROFF, TROFF, NROFF, and SCRIPT.

3. DOCUMENT STORAGE AND RETRIEVAL TECHNIQUES

3.1 Document Specification

3.1.1 The Markup Language

The technique for document specification is a scheme for marking (labelling) regions of text and locations in it. A document preparation system would have a facility for passing information to the system via declarations. In this regard, the writer identifies segments of the text in abstract terms and the system will retrieve the concrete details from the system database. This means that the designers of the document preparation system must identify the proper set of abstractions, name them appropriately and devise a simple syntax that would allow these abstractions to be represented in a file of text characters. Several schemes are available for document markup. The following shows three versions of the same scheme for this purpose [Reid, 80]:

<u>The desired document text</u>

Markup is for marking regions of text,
individual letters and words and also

specific points within the text.

Call The Software House 232 6730

and watch Vals Computers do the job.

The above segment can be marked up with control words as follows:

> Markup is for marking ITALIC regions END;ITALIC of text, individual letters and words and also specific points within the text.
> QUOTATION Call ITALIC The Software House END;ITALIC 232-6730 and watch BOLD Vals Computers END;BOLD do the job END;QUOTATION.

The amount of work required to markup the segment is reduced considerably by the use of escape-character notation; e.g.,

> Markup is for marking @i[regions] of text, individual letters and words and also specific points within the within the text.
> @begin(Quotation) Call @i[THe Software House] 232 6730 and watch @b‹Vals Computers› do the job @end(Quotation).

The latter example in the above illustration would appear to be less cumbersome and easier to manage. To use the escape-character type of notation, the markup alphabet would need to have very low probability of occurrence within the text. Characters such as "@" and "\" are suitable candidates for this choice.

A manuscript will therefore consist of a mixture of text and markup. The document preparation system should be able to tell them apart. In general, this added information called markup to a piece of text designated for processing serves two useful purposes [Goldfarb, 81]:

(i) It separates the logical elements of the document

(ii) It specifies the processing functions to be performed on those elements.

Three distinct steps are required in the document markup process:

(a) Analysis of information structure and other attributes of the document, including the identification of each meaningful separate element, paragraph, heading, ordered list, footnotes or some other element type.

(b) Determination from memory or style book, the processing instructions("controls") that will produce the desired format for a given type of element.

(c) The insertion of the chosen controls into the text.

Two classes of markup techniques have been identified [Goldfarb, 80]:

(1) Procedural markup which consists of instructions or control words. Examples of its usage can be found in SCRIPT and RUNOFF families of formatting languages.

(2) Descriptive markup which consists of tags which describe specific elements and types in the document.

Procedural markup has several disadvantages including:

(i) The loss of information about some attributes of the document. For example, if the centering control is used in RUNOFF to center both headings and figure captions, no indication of the specific attributes of each case is kept within the text so that future retrieval programs can distinguish the headers from the captions.

(ii) Another disadvantage is the fact that it lacks flexibility because of its procedural approach. A user who decides to change the style of a document (perhaps because of the use of another output device) will need major revisions of major elements of the document to reflect the change.

(iii) Markup with control words can be time consuming, error-prone, and will require a high degree of operator training particularly when complex typographic results are required.

The disadvantages of procedural markup are avoided by a markup schema - The Generalized Markup Language (GML) due to Goldfarb [Goldfarb, 80].

3.2 The Document Specification Language

The Document Specification Language provides an interface between the writer and the computer system. The language must therefore be able to provide most of the services provided by major publishing houses. Some of the characteristics which the specification language must possess to provide such services include [Chamberlain, et al., 81]:

(1) Power to describe such complex objects as footnotes, numbered lists, bibliographic material, tables of content, tabular and graphic materials, special characters, font variations, graphical display,

ligatures and other forms of typeset.

(2) Simplicity and Friendliness, so that writers need only have minimal training in order to use the system.

(3) Flexibility, so that writers can have a wide range of alternative formats to choose from.

(4) Symbol independence, so that usage of one symbol at one point will be independent of its usage at other points.

(5) Extensibility of the scope of language operators so that they can accept additional "directions" from authors during formatting.

In general, three classes of notation are needed in the document preparation language:

I. Region labels - a notation for attaching a label or attribute to indicate the authors intention regarding a region of text.

II. Markers - a notation for marking specific points within the text. Markers may include commands.

III. Declaration - a notation for passing values to the system to control certain details of its behaviour.

The first two of these classes are included in the text by means of the appropriate markup language. The requirements for declarations will be the subject of the next few sections.

3.2.1 Declarations

Declarations in a document specification language, serve to control the system by passing to it various parameters and values. Most declarations are restricted to the begining of the document manuscript but some are permitted to occur anywhere. In SCRIBE [Reid, 80], simple declarations include @Device(name), which instructs the SCRIBE compiler to format the document for the named device; @Make(what) which instructs the compiler to produce a document of a requested type or @Pageheading which tells the compiler what text to put in the running page heads. Certain declarations, such as @Define and @Form are intended primarily for use in document format definition [Reid, 80].

3.2.2 Document Type Definitions

Whenever a system produces a document from a manuscript, it does so under the control of the format which is specified in the document editor database. For example, the business letter document type would provide the usual form letter environment which provides environment for return address, letter headings, greeting and a signature. The Thesis (Dissertation) document type would provide environments for chapter headings, footnotes, title pages, bibliography, etc. The document type definition completely determines the final appearance of the document. The manuscript is expected to contain a document type declaration, else the system uses a prespecified default type.

3.3 Character Sets and Font Variations

The initial set of ASCII, BCD and EBCDIC characters provide an initial set of characters which are easily represented on the system. Special characters present some difficulty of representation. However some special characters are just ligatures of the initial set of characters, and for this set the system will just substitute the ligature graphic of the group of characters that appear in the manuscript. Other special characters are printed as special fixed macros whose definition encodes information about how to print the special character on the available printing devices. The naming scheme presupposes that the document preparation system designer knows all the special characters that will exist on the printing devices and gives them names in advance.

3.4 The Database

The system database contains most of the information needed by the system to produce documents. Two major types of information are stored:

(1) Device Information

Used for determining the printing device and

and for retrieving the device definition from the database. The device definition so retrieved contains the specifications for the physical properties of the device, including the specifications for retrieving fonts and format data pertinent to that device.

(2) Document Format Definition

After the system has processed the device data, it retrieves and processes the appropriate format definition from the database. The document format definition data is used for selecting fonts and environments for specified document types.

4. THE SUPPORT ENVIRONMENT

The development of a document calls for a rich environment which provides the writer with as many tools as possible to enable him manipulate both the structural and textual parts of the document as well as providing him with facilities for document management. The goal of this support environment is to provide an integrated set of tools that automate as much as possible the clerical work involved in preparing a document. Such an environment will exist in the form of a high level screen oriented work station with possibly many windows or screens for simultaneous viewing with both structure and text editing capabilities.

Some of such editors in the literature have been discussed in the earlier sections of this report (see Section 2). The implementation of the editor would consist of a top level executive and the set of tools that it invokes. There are three major functional group of tools [Walker, 81]:

(1) Writing Aids,

(2) Structure Editing Aids,

(3) Document Management Aids.

4.1 Writing Aids

These are tools which assist the writer with strictly English language aspects of document preparation. There are three main categories of such aids: Batch, Interactive Batch and Immediate aids. Batch aids provide output from the document after the whole document has been processed (e.g., MULTICS Wordlist). An interactive batch aid would provide both batch and interactive options for processing control (e.g., the optional use of the "continuous" option for display in the MADAM system). Immediate aids would operate in interpretive mode to provide immediate execution of processing commands as they are typed in.

4.2 Structure Editing Aids

These are tools for manipulating the structural elements of the document. The structural elements are regions corresponding to logical hierarchical components. A file tree can be used to represent a document by designating the document as the root, the chapters are the decendants of the root and are roots themselves of other lower level structures such as sections which in turn are roots of paragraphs and so on. The four main classes of commands for structure editing are Locators, Constructors, Mutators and Selectors.

4.2.1 Locators

These are commands for locating places in a document file having certain structural properties. Some Locator commands discussed by Janet Walker [Walker, 81] include the use of keywords DOWN, UP, NEXT, ANY and FIND in association with a specified structural element.

4.2.2 Constructors

The support environment must provide facility for the addition of new structural elements of a specified type. Constructors are used for this purpose. Typical constructor commands would employ the keywords CREATE, MAKE and LIST to create, make or list specified structural elements.

4.2.3 Mutators

Mutators provide facility for structural revision by changing the abstract status of a particular portion of the manuscript. Examples of command keywords include PUSH, DROP, RAISE, POP which are used respectively in conjunction with the specified structural element. For example, PUSH "section" makes a section into a subsection and the former subsections into subsubsections. DROP reduces the status of section to those of its subsections. POP and RAISE have opposite effects respectively.

4.2.4 Selectors

Selectors provide an avenue for visualizing the organizational structure of a growing document by displaying several levels of it. Selector command keywords include the verbs SHOW, SELECT, and DISPLAY.

4.3 Document Management Aids

The set of desirable document management aids include facilities for cross referencing, indexing and annotation.

4.3.1 Cross Referencing

In order to provide cross references which are always correct, cross reference candidates must be identified and tagged with the appropriate markup symbol. An entry will then be made in the cross reference table. In SCRIBE [Reid, 80] correct cross referencing is achieved by tagging cross reference candidates. For example @ref [idTable] informs the system that "I am the table having cross reference label idTable". Given the availability of a cross reference table as well as the appropriate markup for the target symbols, an interactive utility will be required for cross reference lookup. Example commands in such a utility include: FOLLOW XREF pointers and FIND all fingers(to find all cross references that point to particular object).

4.3.2 Indexing

Indexing is perhaps one of the most difficult aspects of document preparation to computerize. Robert Collison [Collison, 66], author of an outstanding work on manual indexing, lists twenty rules that a well prepared index must have. Most of Collison's rules require the indexer to have a clear understanding of the meaning of the text. The need to understand the meaning of words and phrases raises semantic problems which are difficult to computerize. The KWIC (Keyword In Context) method of automated indexing is an approach which was designed to overcome this problem. Because of these semantic difficulties, it would be desirable to have an interactive indexing facility which allows the writer to type in the basic phrase and get the system to generate permutations from which the user can make appropriate selections and possibly weed out poor candidates. Suitable commands include: FOLLOW index pointer, FIND all fingers(all indexes that point to this one), MAKE index entry, SHOW index symbols, MODIFY index entry and ANALYZE index (to determine suitability).

4.4 Annotation

A mechanism is required for associating with any alleged fact, the predigree for the fact. The annotation problem is supported by a document type definition which includes an environment for annotation.

4.5 Partial Output

A large document will consist essentially of a set of files as illustrated in the document file tree. When the document is large, there is need to be able to produce partial outputs of each of these files since many of these files may be produced in different time frames at possibly remote locations. To provide the possibility of partial outputs, the system must include a facility to save the designated structural elements and their relative relations to each other so that cross references as well as indexes may be preserved.

4.6 Document Maintenance

The contents and size of a document is bound to change from time to time following revisions and the inclusion of new material. Things like cross references, bibliography, annotation change correspondingly. A document maintainance facility for updating (possibly changing) all the variables is required as an aid to document revision.

5. DATA STRUCTURES

This section addresses the question of data structures and algorithms required to represent text in order to permit the type of storage and retrieval techniques that have been discussed thus far. The record and storage structures which appear in this section have been taken from SCRIBE [Reid, 80]. The record and storage structure of SCRIBE is a means of dynamic storage allocation whereby memory is allocated by creating a record and deallocated by destroying the record.

5.1 Strings

Strings are variable length objects built on top of the record system. A string consists of two parts - a record token and a buffer record.

```
TYPE String_Token =
    RECORD
         Buffer : tring_Buffer;
         String_Size :  integer;
         Left_pointer : character_index;
         Right_pointer : character_index;
    END;
TYPE  String_Buffer = ARRAY[1..N]  of  character;
```

5.2 Association Lists

An association list, or pair list, is a list of pairs of typed values. Each cell of the list carries two values with an explicit record of the type of each. These pair lists are sometimes used as property lists - one list for each object with its contents being attribute value pairs, and sometimes as associations - one list for each attribute with its contents being object/value pairs.

```
TYPE list_pointer = st_cell;

TYPE list_cell =
   RECORD
        next_cell:list_pointer;
        value_1:any;
        type_1:type;
        value_2:any;
        type_2:type;
   END;
```

5.3 Manuscript Files

As a manuscript file is being processed, it is represented as a sequence of records, each corresponding to one line of the manuscript file:

```
TYPE Manuscript_line =
  RECORD
         Line_name:string;
         Text_of_line:string;
         Processing_cursor:integer;
  END;
```

5.4 Fonts

Font information is kept in order to know the following information:

(1) Sizes (widths and heights) of letters and symbols in order to know how many words to place on a line.

(2) Ligature combinations that are available in fonts.

(3) Codes to send to the printing device in order for it to print or draw the desired letter.

Font information for SCRIBE is kept in the database. Its structure during usage is as follows:

```
TYPE FONT =
    RECORD
        Font_name: string;
        Font_Size: vertical distance;
        Character widths: ARRAY[1..127] of horizontal_distance;
        Character_disp: ARRAY[1..127] of horizontal_distance;
        Character_constructions: ARRAY[1..127] of string;
        Ascii_translation: ARRAY[1..127] of integer;
        Draw_codes: ARRAY[1..127] of string;
        Ligatures: pairlist of(name:string,code:integer);
        Special_symbols:pairlist of(name:string,code:integer)
    END;
```

5.5 Environments

Environments are implemented as pair lists and used as property list. An environment is an ordered set of dynamic parameter names and changes to be made to these parameters. The environment structure is:

```
TYPE Environment = (not) Environment_pair;
TYPE Environment_pair =
     RECORD
          Next_Cell: (not) Environment_Pair;
          Parameter_Name:integer;
          Change_value:any;
```

```
                    Change_type:type;
               END;
```

5.6 Text Buffers

The manuscript text is assembled by the formatter into words, lines, boxes and pages. Each of these is kept in an appropriate record. Word records are assembled into line records, line records are assembled into box or page records and when assembly is complete, the assembled page is written into the device file.

A word buffer holds one word:

```
          TYPE Word_buffer =
               RECORD
                    Text:string;
                    Bounding_width:Horizontal_distance;
                    Bounding_height:Vertical_distance;
                    Left_spacing:Horizontal_distance;
                    Right_spacing:Horizontal_distance;
                    top_spacing:Vertical_distance;
                    Bottom_spacing:vertical_distance;
                    Footnote_box:Text_box;
               END;
```

A line buffer holds one line:

```
TYPE Line_buffer =  RECORD
                         Text:string;
                         Next_line: <> (Line_buffer OR Box_Buffer);
                         Parent_box: <> Box_buffer;
                         X_origin:Horizontal_distance;
                         Y_origin:Vertical_distance;
                         Bounding_width:Horizontal_distance;
                         Bounding_height:Vertical_distance;
                         Left_spacing:Horizontal_distance;
                         Right_spacing:Horizontal_distance;
                         Top_spacing:vertical_distance;
                         Bottom_spacing:vertical_distance;
                         Footnote_box:Text_box;
               END;
```

The text of the line is the concatenation of the text strings in the line. A box bufffer is similar to a line buffer but instead of a text field, it has a child box field which points to a list of line records that contain the actual text.

```
    TYPE box_buffer = RECORD
                   Child_box: (not) (Line_buffer OR Box_buffer)
                   Parent_box:(not) Box_buffer;
```

```
            Next_line:(Line_buffer OR Box_buffer)
            X_origin:Horizontal_distance;
            Y_origin:Vertical_distance;
            Bounding_width:Horizontal_distance;
            Bounding_height:Vertical_distsnce;
            Left_spacing:Horizontal_distance;
            Right_spacing:Horizontal_distance;
            Top_spacing:Vertical_distance;
            Bottom_spacing:Vertical_distance;
            Footnote_box:Text_box
        END;
```

5.7 Other Support Structures

In addition to the above structures, a document preparation system will use a symbol table where all commands, environments, user defined names and file names are kept. The system will also maintain several dictionaries where certain data items are kept for easy lookup.

6. STRING MATCHING ALGORITHMS

In many information retrieval, text editing and document processing applications, it is necessary to quickly locate some or all occurrences of user specified patterns of words and phrases in the text. The problem of pattern matching is central to all IS&R issues as the efficiency of a retrieval technique will depend largely on the speed with which objects are located in storage. In this section, a brief discussion of a few efficient algorithms will be presented. The focus will be the introduction of the technique and an identification of related publications.

Simple programs for searching text typically require a worst case running time of O(m*n) where m is the length of the pattern and n is the length of the string(i.e., the text). However, Knuth [Knuth, et al., 77] showed that this can be reduced to O(n). Later, Boyer and Moore [Boyer, et al., 77] published a practical and simpler algorithm that also has a linear worst case running time. Other algorithms have been published using different techniques such as text signatures [Harrisson, 71], character frequency in text [Horspool, 80], and Huffman encoding with tries (PATRICIA) [Morrison, 68]. Most of these linear time algorithms rely on the fact that, in the average case, only a small fraction of the n characters of the text are actually inspected.

6.1 Scan for First Character(SFC) [Horspool, 80].

Some computers have a single instruction that can be used to search memory for the first occurrence of a designated character. If such a search instruction were available, it would seem very reasonable to employ it in locating a substring within a larger string. This is idea behind the SFC algorithm which follows.

```
Algorithm SFC
/* STRING is text to be searched, STRINGLEN its length*/
/* PAT is substring to be found, PATLEN its length*/
begin
   if patlen > stringlen then return 0;
   ch := pat[1];
   i:=0;
   repeat
     SCAN string[i+1 .. stringlen - patlen + 1] to find
       first occurrence of ch;
     if ch was not found then return 0;
     i:=position where ch was found;
   until string[i .. i+patlen-1] = pat;
   return i
end;
```

If the algorithm returns "0", then PAT does not occur in STRING, otherwise the result is the first occurrence.

6.2 Scan for Lowest Frequency Character [Horspool, 80]

A walk through of algorithm SFC with a pattern containing the letter E (high frequency English letter) on any reasonably sized text string will lead to a lot of mismatches as the pattern is moved across the text. Consider the word EXTRA, the letter X in the pattern occurs infrequently in English words. So if the search instruction were used to locate successive occurrences of X instead of E, it would be possible to skip through hundreds of characters at the same time. The idea behind Algorithm SFLC is to pick the character in PAT which occurs least frequently in STRING.

```
Algorithm SLFC
 begin
   if patlen > stringlen then return 0;
   find j such that pat[j] is the character in pat with
      the lowest frequency in English text;
   ch:=pat[j];
   i:=j-1;
   repeat
     SCAN string[i+1 .. stringlen - patlen + j]
```

```
            to find first occurrence of ch;
          if ch was not found then return 0;
          i:=position where ch was found;
        until string[i-j+1 .. patlen-j]=pat;
        return i-j+1
      end;
```

The character frequency information required by this algorithm is provided in the form of an alphabet table sorted in order of expected frequency of occurrence.

6.3 The KNUTH-MORRIS-PRATT Algorithm [Knuth, et al., 77].

The idea behind the KMP algorithm is to construct a failure transition table from the pattern which indicates how far to slide the pattern across the text string when a mismatch occurs at each position of the pattern. The pattern matching procedure constructs a deterministic finite state automaton having a next state transition for each alphabet in the pattern. These values are kept in a table called the NEXT[j] table (j is the index of the pattern alphabets).

```
Algorithm KMP
  begin
    place pattern at the left;
    while pattern not fully matched
         and text not exhausted do
```

```
        begin
          while pattern character differs from
             current text character
               do shift pattern appropriately;
          advance to next character of text;
         end;
    end;
```

The detailed coded algorithm for this procedure can be found in [Knuth, et al., 77].

6.4 The BOYER and MOORE algorithm (BM) [Boyer, et al., 77].

The BM algorithm is similar in concept to the KMP algorithm but differs in that the algorithm matches the pattern in a right to left order while the pattern is being moved to the right. They are similar in concept in that there is a preprocessing stage in which the pattern is processed to obtain a failure transition table which indicates how far right to slide the pattern in case of a mismatch. In the case of Boyer and Moore, two tables delta1(char) and delta2(string) are used to decide how far right to slide the pattern in the text. Delta1(char) provides the opportunity to jump through whole pattern lengths on the string while delta2(i) determines how far right to slide based on the "rightmost plausible reoccurrence of the pattern in the text". The Algorithm follows:

```
Algorithm BM
  /Assume the availability of Delta1 and Delta2 table*/
   last_ch := pat[patlen];
         i:=patlen;
   While i <= stringlen do
     begin
        ch := string[i] ;
        if ch = last_ch then
         begin
           j=patlen - 1;
           repeat
            if j=0 then return i;
            j:=j-1;
            i:=i-1;
           until string[i]<>pat[j];
           i := i + max(delta1[ch], delta2[j]);
          end
        else
          i := i+delta1[ch];
       end;
     return 0;
end.
```

6.5 Text Signatures [Harrison, 71].

Malcolm Harrison has observed that superimposed coding can speed up text searching. If it is desired to locate all occurrences of a pattern in a string of text, assuming that the text is divided into individual lines c1,c2,...,c50 of fifty character each (an example). Harrison suggests encoding each of the 49 pairs c1c2, c2c3, ... , c49c50 by hashing them into a number between 0 and 127, say; then the "signature" of the line c1c2...c50 is the string of 128 bits b0b1...b127 where b(i) = 1 if and only if h(c(j)c(j+1)) = i for some j.

The pattern matching algorithm then proceeds by comparing the signature of the text against the signature of the pattern using fast binary operators such as AND.

6.6 Practical Algorithm for Retrieval of Information Coded in Alphanumeric (PATRICIA) [Morrison, 68].

PATRICIA is particularly suited for dealing with extremely long, variable-length keys such as titles or phrases stored within a large bulk file. Its basic idea is to build a binary trie but to avoid one-way branching by including in each node the number of bits to skip over before making the next test.

SUMMARY

In this report, we have outlined some of the fairly recent issues that relate to document processing. Many problems with semantic requirements still continue to defy computerization; however, the development of artificial intelligence and knowledge-based systems continue to promise well for the future as more efficient techniques for storage and retrieval continue to be discovered through research. Much material relating to document processing in particular and text processing in general has been omitted because of the limited scope of this report. However, we believe that the material presented herein will provide adequate incentives for further research for the interested reader.

REFERENCES

[Allen, et al. 81]. Allen T., Nix R. and Perlis A., "PEN: A Hierarchical Document Editor," ACM SIGPLAN / SIGOA Symposium on Text Manipulation, Jun. 81, pp. 74-81.

[Boyer, et al., 77]. Boyer, R. S. and Moore, J. S., "A Fast String Searching Algorithm," CACM, 20:10, Oct. 77. pp. 762-772.

[Chamberlain, et al., 81]. Chamberlain D. D., King J. C., Slutz D. R., Todd S. J. and Wade B. W., "JANUS: AN INTERACTIVE SYSTEM FOR DOCUMENT COMPOSITION," ACM SIGPLAN/SIGOA Symposium on Text Manipulation, Jun. 81, pp. 82-91.

[Collison, 59]. Collison, R. L., Indexes and Indexing: Guide to Indexing Books and other Material, John de Graff Inc., New York, 1959.

[Goldfarb, 81]. Goldfarb, C. F., "A Generalized Approach to Document Markup," Conference Record, ACM SIGPLAN/SIGOA Symposium on Text Manipulation, Portland, Oregon. Jun. 1981, pp. 68-73.

[Hammer, et al., 81]. Hammer M., Ilson R., Anderson T., Good M., Niamir B., Rosentein L., and Schoichet S., "The Implementation of Etude, An Integrated and Interactive Document Production System," Conference Record, ACM SIGPLAN/SIGOA Symposium on Text Manipulation. Portland, Oregon. Jun., 1981 pp. 137 - 146.

[Harrison, 71]. Harrison, M. C., "Implementation of the Substring Test by Hashing," CACM, 14:12, Dec. 1971. pp. 777-779.

[Horspool, 80]. Horspool, N., "Practical Fast Searching in Strings," Software Practice and Experience, 1980, pp. 501-506.

[Kernighan, et al., 75]. Kernighan B. W. and Cherry L. L., " A System for Typesetting Mathematics," CACM, 18, Mar. 75. pp. 151-156.

[Knuth, et al., 77]. Knuth, D. E., Morris, J. H. and Pratt, V. R., "Fast Pattern Matching in Strings," SIAM J. Comp., 6:2, Jun. 77, pp. 323-350.

[Knuth, 79]. Knuth, D. E. TEX and METAFONT: New Directions in Typesetting, American Math. Society and Digital Press, 1979.

[Morrison, 68]. Morrison, D. "Practical Algorithm To Retrieve Information Coded in Alphanumeric," JACM, 15, 1968, pp. 514-534.

[Reid, 80]. Reid, B.K. SCRIBE: A Document Specification Language and its Compiler, Ph.D Dissertation, Dept. of Computer Science, Carnegie-Mellon University, Oct., 1980. Tech. Rep. CMU-CS-81-100.

[Stallman, 81]. Stallman, R.M. "EMACS: The Extensible Customizable, Self-Documenting Display Editor," Conference Record, ACM SIGPLAN/SIGOA Symposium on Text Manipulation. PortLand, Oregon. Jun., 1981. pp. 147-156.

[Stromfors, et al., 81]. Stromfors O. and Jones J. L., "The Implementation and Experiences of a Structure Oriented Text Editor," Conference Record, ACM SIGPLAN/SIGOA Symposium on Text Manipulation, Jun. 81, pp. 22-27.

[Walker, 81]. Walker, J.H. "The Document Editor: A Support Environment for Preparing Technical Documents," Conference Record, ACM SIGPLAN/SIGOA Symposium on Text Manipulation, Portland, Oregon. Jun. 81, pp. 44-50.

APPENDIX

```
                          -------------
---------------------    | Finished    |
|    The Author     |    | Document    |
---------:-----------     -------------
         V                      |
 --------------------      ------------
 |     Editor       |      | Indexer  |
 --------:-----------      ------------
         V                      |
 --------------------      -------------
 |   Typesetter     |--> | Page Makeup|
 --------------------      -------------
```

Fig. 2.1 Information Flow Schematic in a
Traditional Publishing Operation.

```
          ______________
          | The        |
          | Author     |
          ------|------
     __________V__________
     |                   |  <-- Document Design
     |     THE           |  <-- Typo Design
     |                   |
     |   COMPUTER        |
     |                   |  <-- Layout Design
     |                   |  <-- Indexes
     |                   |
     |                   |
     ---------:-----------
              V
     --------------------
     | Finished Document |
     --------------------
```

Fig. 2.2 Information Flow in an Automated
Publishing Operation.

```
                      ------------
   Typo Design->| The         | <--Document Design
 Layout Design->| Author      | <--Indexing
                      ------------
                           | CONTROL
                           V
                   ------------------
                   | The Computer     |
                   |                  |
                   ------------------
                           |
                           V
                   ------------------
                   | Finished         |
                   | Document         |
                   ------------------
```

Fig. 2.3 Information Flow in a Computerized Publishing Operation.

4.12

1. Report No. IN-82

2. Government Accession No. 183557 ~~[illegible]~~

3. Recipient's Catalog No.

4. Title and Subtitle
2 11.
USL/NGT-19-010-900: AN OVERVIEW OF SELECTED INFORMATION STORAGE AND RETRIEVAL ISSUES IN COMPUTERIZED DOCUMENT PROCESSING

5. Report Date DATE OVERRIDE
December 29, 1984

6. Performing Organization Code

7. Author(s)
VALENTINE U. IHEBUZŌR

8. Performing Organization Report No.

10. Work Unit No.

9. Performing Organization Name and Address
University of Southwestern Louisiana
The Center for Advanced Computer Studies
P.O. Box 44330
Lafayette, LA 70504-4330

11. Contract or Grant No.
NGT-19-010-900

12. Sponsoring Agency Name and Address

13. Type of Report and Period Covered
FINAL; 07/01/85 - 12/31/87

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The rapid development of computerized information storage and retrieval techniques has introduced the possibility of extending the word processing concept to document processing. A major advantage of computerized document processing is the relief of the tedious task of manual editing and composition usually encountered by traditional publishers through the immense speed and storage capacity of computers. Furthermore, computerized document processing provides an author with centralized control, the lack of which is a handicap of the traditional publishing operation. A survey of some computerized document processing techniques is presented with emphasis on related information storage and retrieval issues. String matching algorithms are considered central to document information storage and retrieval and are also discussed.

This report represents one of the 72 attachment reports to the University of Southwestern Louisiana's Final Report on NASA Grant NGT-19-010-900. Accordingly, appropriate care should be taken in using this report out of the context of the full Final Report.

17. Key Words (Suggested by Author(s))
Computerized Document Processing Techniques, String Matching Algorithms, Information Storage and Retrieval Systems

18. Distribution Statement

19. Security Classif. (of this report)
Unclassified

20. Security Classif. (of this page)
Unclassified

21. No. of Pages
49

22. Price*