Converting bibliographic records into ISO 2709 format (Access Presentation)

Thesis · January 2002

CITATION
1

READS
2,623

1 author:

Satya Ranjan Sahu
CSIR-Institute of Minerals and Materials Technology
10 PUBLICATIONS **28** CITATIONS

SEE PROFILE

Converting bibliographic records into ISO 2709 format (Access Presentation)

A
Minor Project

Submitted during the training course on Information Technology Application to library and Information Services

By

Satya Ranjan Sahu

Under the guidance of

Francis Jayakanth

National Center for Science Information
Indian Institute of Science
Bangalore-560012
January-2002

CERTIFICATE

This is to certify that the Minor project work embodied in this project entitle **"Converting bibliographic records into ISO 2709 format"** has been carried out by "Satya Ranjan Sahu" under my guidance in partial fulfillment of the requirements for the **Training course on Information Technology Application to Library and Information Services**, National Center for Science Information, Indian Institute of Science, bangalore-560012.

(Francis Jayakanth)
NCSI, IISC
Bangalore-560012

Date: 11-01-2002
Place: Bangalore

Declaration

I hereby declare that this project report on, **"Converting bibliographic records into ISO2709 format"** as a partial fulfillment for the '**Training Program in Information Technology Application to Library and Information Services'** is the result of the work carried out by me under the guidance of Mr. Francis Jayakanth, Scientific Staff, National Centre for Science Information, Indian Institute of Science. I further declare that this project work has not been submitted anywhere else for any purpose.

Date: (Satya Ranjan Sahu)
Place: Bangalore

Acknowledgements

I acknowledge with deep sense of gratitude to my guide Mr. Francis Jayakanth, Scientific staff, National center for Science Information, Indian Institute of Science, Bangalore for guiding and helping me at every step through out the project to complete this.

I express my thanks to Dr. T. B. Rajashekar, Associate Chairman, National Center for Science Information, Indian institute of Science for giving me suggestions, encouragement as well as providing me the facility to carry out this project.

I am very grateful to all the staffs, project Assistants, project trainees and my classmates for their encouragement and cooperation during the project work.

(Satya Ranjan Sahu)

Table of Contents

Chapter -1

INTRODUCTION

Bibliographical information plays an important role to Research community particularly in the field of science and technology. But during the Bibliographical information exchange certain problems arise and more when the information interchange is on magnetic tape or CD-ROM.

Different International organizations such as UNESCO/PGI, UNISIST, ICSU-AB, IFLA, ISO have taken many steps towards the standardization of Bibliographic Exchange formats. The process of standardization follows a set of codes given by International Standard Organization (ISO).

The three major purposes of standardization are:

- To permit the exchange of bibliographic records between groups of libraries and abstracting and indexing Services.
- To permit a bibliographic agency to manipulate bibliographic records received from both libraries and abstracting and indexing services.
- To serve as the basis of a format for an agency's own bibliographic database by providing a list of useful data elements. To assist the development of individual systems.

Converting a text file into a standard format such as ISO2709 has many benefits. For example, the set of programs can be used for indexing and retrieval. The report describes the set of programs developed on C language to carry out conversion of sample records extracted from current content database to Standard text file.

Though C language is designed as a systems language for UNIX, it can be used for database applications and text processing. A host of function included in its library for input/output, string handling, and so on, make it a general-purpose language. C is a small language that supports simple data types and includes no in-built facilities for file operations such as input/output or file accessing. But any complex data type can be constructed from the available ones and different file organization and access mechanisms can be built using the functions included in the library. This general features allows a lot of flexibility in developing effective database applications. Another factor that made C as the first choice is the portability afforded by it. Since C is not tied to a particular machine, the program developed on one system can be transferred to other systems with few changes.

Chapter -2

File formats for Exchange of bibliographic data

Different file formats for exchange of bibliographic data came into existence when peoples and organizations were investigating the feasibility of producing catalogue data in machine-readable form. Hence Exchange formats were developed in parallel with the development of computers and other electronic storage devices to facilitate the transfer of bibliographic data between computer systems. Their use affects economies by reducing the duplication of effort implicit in different libraries acquiring and cataloguing the same material. Depending on the aim and objective of bibliographic data exchange different country and organization developed their own format. Some of the formats developed in International and National levels are as follows:

- MARC (**MA**chine-**R**eadable **C**ataloguing) - is a format standard for the storage and exchange of bibliographic records and related information in machine-readable form. All MARC Standards conform to: ISO 2709:1996 Information and documentation -- Format for Information Exchange.
- UKMARC - The UKMARC format is the standard developed, managed and promoted by the British Library and applied by the Library in its bibliographic products and services and by many UK libraries in their bibliographic processing operations.
- MARC 21 (Formerly USMARC and CANMARC) - The MARC 21 formats are standards for the representation and communication of bibliographic and related information in machine-readable form. The MARC 21 formats are maintained by the Library of Congress in consultation with various user communities. The British Library follows developments in MARC 21 and since 1995 has followed a policy of limiting divergence between MARC 21 and UKMARC.

- UNIMARC - The primary purpose of UNIMARC is to facilitate the international exchange of data in machine-readable form between national bibliographic agencies. UNIMARC may also be used as a model for the development of new machine-readable bibliographic formats. The interests of users of UNIMARC records are represented by the Permanent UNIMARC Committee (PUC) on behalf of IFLA Universal Bibliographic Control and International MARC (UBCIM), which has ultimate responsibility for UNIMARC. The British Library is an active member of the Permanent UNIMARC Committee.
- ONIX - ONIX International is the international standard for representing and communicating book industry product information in electronic form, incorporating the core content which has been specified in national initiatives such as Book Industry Communications (BIC) Basic and American Association of Publishers' ONIX Version 1. ONIX is developed and maintained by EDItEUR jointly with Book Industry Communication and the Book Industry Study Group. The British Library works with BIC and EDItEUR towards the realization of a continuum of bibliographic product information.
- **C**ommon **C**ommunication **F**ormat – The CCF has been prepared with the support of Unesco, within the framework f the General Information Programme, By the Ad hoc Group on the Establishment of a common communication Format.

The chief purpose of the format is to provide a detailed and structured method for recording a number of mandatory and optional data elements in a computer-readable bibliographic record for exchange purpose between two or more computer –based systems. The structure of format conforms to the international standard ISO 2709. The core record consists of a small number of mandatory data elements essential to bibliographic description, identified in a standard manner. The core record augments

additional optional data elements, identified in a standard manner. It also accommodates levels, relationships, and links between bibliographic entities.

Chapter - 3

ISO2709

(Format for Bibliographic Information Interchange on Magnetic Tape)

This International Standard specifies the requirements for a generalized exchange format that will hold records describing all forms of material capable of bibliographic description as well as related records such as authority records. It does not define the length or content of individual records and does not assign any meaning to tags, indicators or identifiers; these specifications being the functions of an implementation format.

This International Standard describes a generalized structure, a framework designed specifically for communications between data processing systems and not for use as a processing format within system. Although this International Standard is designed for magnetic tape, its structure may be used for other data carriers.

Principle and codings:

The standard ISO 2709 (standard AFNOR NF Z 47300, December 1987) makes it possible to present any structured bibliographic record a large variety of formats, in particular the MARC or UNIMARC or CCF formats.

A recording comprises in ISO the 2709 following parts:

- **The guide**, continuation of 24 numbered natures from 0 to 20
- **The repertory**, which comprises a variable succession of numerical natures
- **Bibliographical data** themselves

Chapter - 4

Structure of Communication format for bibliographic record

The general structure of a bibliographic record consists of four major parts:

- Record label
- Directory
- Data fields
- Record separator

When building a physical volume (magnetic tape or disk) of bibliographic records, consideration must be given to the structure of the volume.

Record label:

Each bibliographic record begins with a fixed-length label of 24 characters, the contents of which are as follows:

Character Position (s)	Contents
0 to 4	Record length. The length of the record includes the label, directory, data fields, and record separator. (Use of 5 characters for the record length permits records as long as long as 99,999 characters.)
5	Record status, using a code taken from the list of Record Status, using Codes.

6	‘b’ (blank) This character position is not used.
7	Bibliographic level of the target item, using a code taken from the list of Bibliographic Level Codes.
8	‘b’ (blank) This character position is not used.
9	‘b’ (blank) This character position is not used.
10	‘2’ The indicator length. This shows the number of indicator characters in each field.
11	‘2’ The subfield identifier length. This shows the number of characters used as the subfield identifier.
12 to 16	Base address of data. The location within the record at which the first datafield begins, relative to the first character in the record, which is designated character position ‘0’ (zero).
17 to 19	‘b’ (blank) These character positions are not used.
20	‘4’ The length of ‘Length of Datafield’ in the directory. (Use of 4 characters permits datafields as long as 9,999 characters).
21	‘5’ The length of ‘Starting Character Position’ in the directory.
22	‘2’ The length of implementation-defined section of each entry in the directory. Of the two characters, one is used for the Segment identifier, the other for the occurrence Identifier.
23	‘b’ (blank) this character position is not used

Directory:

The directory is a table containing a variable number of fourteen-character entries; the table is terminated by a field separator character. Each director entry corresponds to an occurrence of datafield separator character. Each directory entry corresponds to an occurrence of a datafield in the record, and is divided into five parts:

- Tag
- Length of datafield
- Starting character position
- Segment identifier
- Occurrence identifier

Tag: A three character code identifying the datafield which corresponds to the directory entry.

Length of Datafield A four-digit number showing how many characters are occupied the datafield, including indicators and datafield separator but excluding the record separator code if the datafield is the last field in the record.

Staring Character Position A five-digit number giving the position of the first character of the datafield relative to the base address of data, i.e. the first character of the first of the datafield.

Segment Identifier A single character (chosen from 0-9 and/or A-Z) which designates the datafield as being a member of particular segment.

Occurrence Identifier A single character (chosen from 0-9 and A-Z) which differentiates multiple occurrences of the datafields that carry the same tag within the same record segment.

A single directory entry is organized as follows:

TAG	LENGTH OF DATAFIELD	STARTING CHARACTER POSITION	SEGMENT IDENTIFIER	OCCURRENCE IDENTIFIER
3 character	4 character	5 character	1 character	1 character

Here is an example of a directory entry:

300003300028910

Datafields:

A datafield consists of:

- Indicators
- One or more subfields each of which is preceded by a subfield identifier
- A datafield separator

Indicators Two bytes reserved for use as defined for each data field. These may supply further information about the contents of the datafield, or about the action required in certain data manipulation process.

Subfields A subfield consists of a subfield identifier followed by a data string, which is terminated by either another subfield identifier or a field separator. A subfield identifier consists of subfield identifier flag followed by one other character.

Datafield Separator The datafield separator constitutes the final character of every datafield.

INDICATORS 2 characters	SUBFIELD INDENTIFIER 2 characters	SUBFIELD Variable	FIELD SEPARATOR 1 character

A datafield, which has two subfields, will be organized as follows:

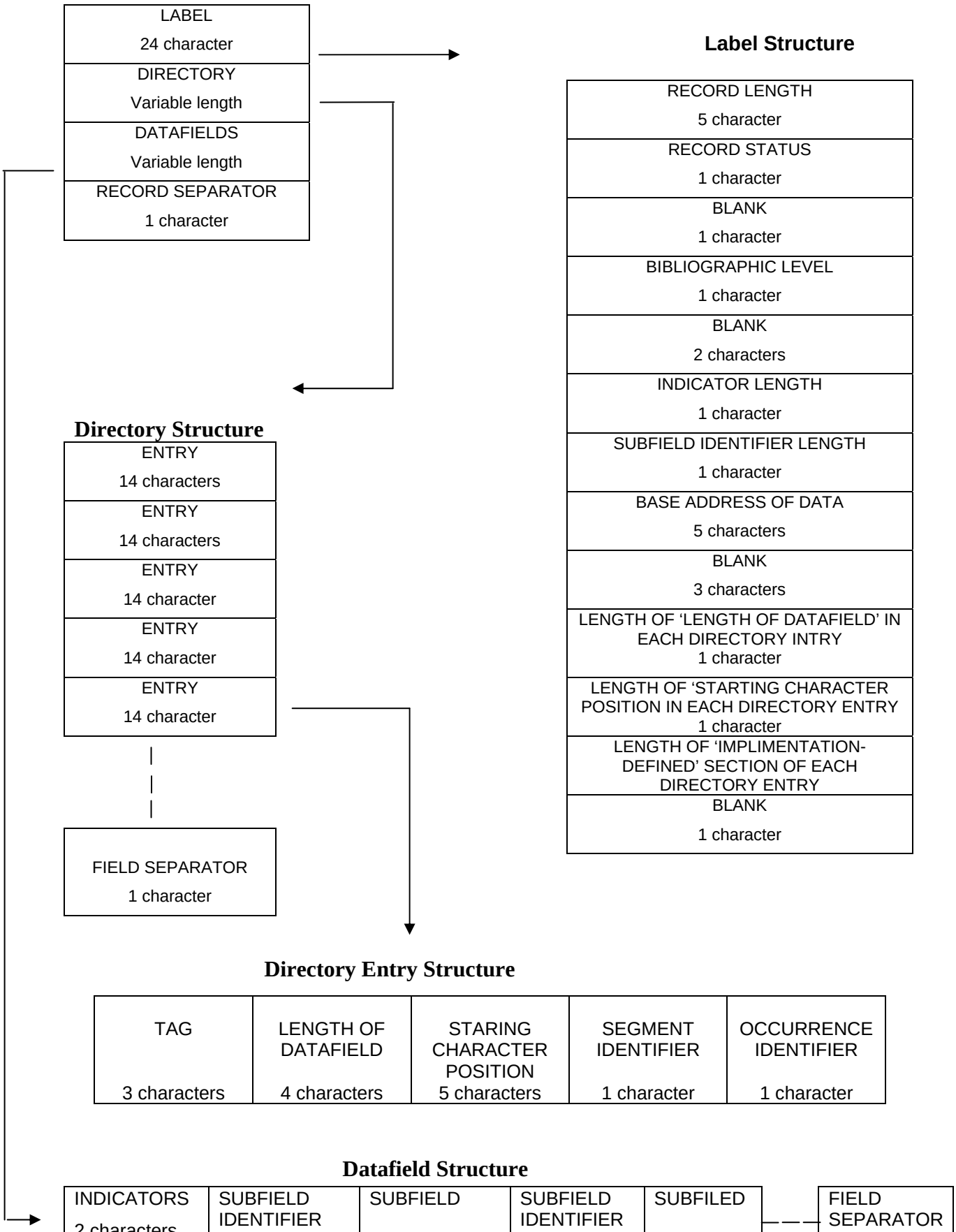
INDICATORS 2 characters	FIRST SUBFIELD INDENTIFIER 2 characters	FIRST SUBFIELD Variable	SECOND SUBFIELD IDENTIFIER 2 characters	SECOND SUBFIELD IDENTIFIER Variable	FIELD SEPARATOR 1 character

11^aStephenson^bBM^c1975-

Record separator:

The record separator is the final character of the record. It follows the field separator of the final datafield of the record.

DIAGRAMMATIC REPRESENTATION OF THE RECORD STRUCTURE



Chapter - 5

Advantages of ISO2709:

Standards are documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose. Hence ISO2709 (Format for Bibliographic Information Interchange on Magnetic Tape) has many advantages. Some of the important advantages are given below:

- It provides a small number of mandatory data elements, which are recognized by all sectors of the information community as essential in order to identify an item.
- It gives mandatory data elements that are sufficiently flexible to accommodate varying descriptive practices.
- It also provides a number of optional elements, which may be useful to describe an item according to practices of the agency, which creates the record.
- It provides a mechanism for linking records and segments of records without imposing on the originating agency any uniform practice regarding the treatment of related groups of records or data elements.

Chapter - 6

STATEMENT OF THE PROBLEM

Objective:

Aim of the project is to “Convert Bibliographic Records into ISO2709 format”. ISO 2709 is a format for information exchange of bibliographic data in machine-readable form between national bibliographic agencies.

Need of the project:

NCSI subscribes to various bibliographic databases like BIOSIS, INSPEC, MATHSCI etc. These databases are accessible on the campus network. Apart from providing access to these databases, many value added services can be provided using these databases. In order to provide these value added services, data can be extracted from the databases and build our own indexing and retrieval system. In order to build the indexing and retrieval system, we can convert the data extracted from the bibliographic databases and invest them into ISO file format so that same set of programs can be used for indexing and retrieval.

Limitations:

The program works only with current content bibliographic data.

Chapter - 7

IMPLEMENTATION AND METHODOLOGY

Conversion of text file into ISO2709 format is tedious and time consuming. Therefore a utility is used to convert a standardized text file into ISO2709 format. The standardization of the text file is carried out using “C” programming Language. There are some steps involved in doing the standardization of the text file, they are as follows.

1. Tags must begin in the same character positions (columns) in each record/line of the input file.

2. The program assumes that all tags are of three digits. Thus single or two-digit tags should be expanded to three digits, e.g., 001, 020, etc.

3. The data for all tags should also begin from the same character position (column) in each record/line of the input file. For instance, tags could always be in columns 1-3, and data for each tag could begin in column 5.

4. Data for a tag could be continued in more than one line. However, continuation lines for data begin also from the same column position. Thus for instance if data begins in column 5 of each line, all continuation lines of data will also begin in column 5.

5. Each logical record of the input file ends with a ## line in the first two positions.

The next process is the running of the utility using the standardized text file. Conversion steps of text file into ISO2709 format using the utility (TXT2ISO.EXE) are as follows:

1. After running the EXE file, it will prompt to enter the column position where the tag begins in records of the input file.

2. Next it will prompt to enter the column position where data fields begin in records of the input file.

3. Finally it will ask to enter the name of the input ASCII text file.

5. The program then begins converting records of the input file into a file called ISO.MST, which is the ISO file that can later be imported into a pre-defined CDS/ISIS database using the same tagging structure as given in the input file.

Chapter - 8

OBSERVATION AND CONCLUSION

The program works only with current content data. The program can be generalized so that it is able to convert records from different databases into the file format as required by the “TXT2ISO.EXE” program. Numbers of researchers come to NCSI to avail different information services. They are provided with bibliographic data in text format. Program can be modified so that these records given to users can be converted to ISO2709 format so that they can import these records into a database management system like CDS/ISIS.

REFERENCES

1. ISO 2709: Documentation format for bibliographic information interchange for magnetic tape, 2nd ed. Switzerland. Publisher- International organisation for standardization 1981- 5p.27cm
2. Simmons, Peter; Hopkinson, Alan: CCF: The common communication format. 2nd ed
3. Simmons, Peter: Microcomputer software for ISO 2709 record conversion. Microcomputers for Information Management, 6 (3) Sept 89, 197-205. illus. 4 refs, 1991
4. Mishra, K. S: Bibliographic databases and exchange formats. DESIDOC Bulletin of Information Technology, 17 (5) Sep 1997, p.17-22. refs,1991
5. http://www.iso.ch/iso/en/aboutiso/introduction/index.html
6. http://www.bib.wau.nl/isis/software.html#1
7. http://www.bl.uk/services/bibliographic/exchange.html
8. http://www.acctbief.org/avenir/ccf.htm

APPENDICES

Appendix – I

Record 1 of 7942
Authors MJ Smekens, PH vanTienderen
Title Genetic variation and plasticity of Plantago coronopus under saline conditions
Full source Acta Oecologica - International Journal of Ecology, 2001, Vol 22, Iss 4, pp 187-200
Author keywords salt adaptation; Plantago coronopus; phenotypic plasticity
KeyWords Plus PHENOTYPIC PLASTICITY; REACTION NORMS; SALT TOLERANCE; LIFE-HISTORY; EVOLUTION; LIMITS; GROWTH; POPULATIONS; ALLOCATION; SELECTION
TGA/Book No. 488NK
Discipline Environment / Ecology
Document type Article
Language English
Address van Tienderen PH, Netherlands Inst Ecol, Ctr Terr Ecol, POB 40, NL-6666 ZG Heteren, NETHERLANDS
ISBN/ISSN 1146-609X
Publisher Gauthier-Villars/Editions Elsevier, 23 Rue Linois, 75015 Paris, France
Abstract Phenotypic plasticity may allow organisms to cope with variation in the environmental conditions they encounter in their natural habitats. Salt adaptation appears to be an excellent example of such a plastic response. Many plant species accumulate organic solutes in response to saline conditions. Comparative and molecular studies suggest that this is an adaptation to osmotic stress. However, evidence relating the physiological responses to fitness parameters is rare and requires assessing the potential costs and benefits of plasticity. We studied the response of thirty families derived from plants collected in three populations of Plantago coronopus in a greenhouse experiment under saline and non-saline conditions. We indeed found a positive selection gradient for the sorbitol percentage under saline conditions: plant families with a higher proportion of sorbitol produced more spikes. No effects of sorbitol on fitness parameters were found under non-saline conditions. Populations also differed genetically in leaf number, spike number, sorbitol concentration and percentages of different soluble sugars. Salt treatment led to a reduction of vegetative biomass and spike production but increased leaf dry matter percentage and leaf thickness. Both under saline and non-saline conditions there was a negative trade-off between vegetative growth and reproduction. Families with a high plasticity in leaf thickness had a lower total spike length under non-saline conditions. This would imply that natural selection under predominantly non-saline conditions would lead to a decrease in the ability to change leaf morphology in response to exposure to salt. All other tests revealed no indication for any costs of plasticity to saline conditions. (C) 2001 Editions scientifiques et medicales Elsevier SAS.

Appendix – II

300 ^fMJ^lSmekens%^fPH^lvanTienderen
200 Genetic variation and plasticity of Plantago coronopus under saline conditions
020 Acta Oecologica - International Journal of Ecology
440 2001
490 ^aVol 22^bIss 4^cpp 187-200
620 salt adaptation; Plantago coronopus; phenotypic plasticity
625 PHENOTYPIC PLASTICITY; REACTION NORMS; SALT TOLERANCE; LIFE-HISTORY; EVOLUTION; LIMITS; GROWTH; POPULATIONS; ALLOCATION; SELECTION
011 488NK
610 Environment / Ecology
060 Article
040 English
330 van Tienderen PH, Netherlands Inst Ecol, Ctr Terr Ecol, POB 40, NL-6666 ZG Heteren, NETHERLANDS
101 1146-609X
400 Gauthier-Villars/Editions Elsevier, 23 Rue Linois, 75015 Paris, France
600 Phenotypic plasticity may allow organisms to cope with variation in the environmental conditions they encounter in their natural habitats. Salt adaptation appears to be an excellent example of such a plastic response. Many plant species accumulate organic solutes in response to saline conditions. Comparative and molecular studies suggest that this is an adaptation to osmotic stress. However, evidence relating the physiological responses to fitness parameters is rare and requires assessing the potential costs and benefits of plasticity. We studied the response of thirty families derived from plants collected in three populations of Plantago coronopus in a greenhouse experiment under saline and non-saline conditions. We indeed found a positive selection gradient for the sorbitol percentage under saline conditions: plant families with a higher proportion of sorbitol produced more spikes. No effects of sorbitol on fitness parameters were found under non-saline conditions. Populations also differed genetically in leaf number, spike number sorbitol concentration and percentages of different soluble sugars. Salt treatment led to a reduction of vegetative biomass and spike production but increased leaf dry matter percentage and leaf thickness. Both under saline and non-saline conditions there was a negative trade-off between vegetative growth and reproduction. Families with a high plasticity in leaf thickness had a lower total spike length under non-saline conditions. This would imply that natural selection under predominantly non-saline conditions would lead to a decrease in the ability to change leaf morphology in response to exposure to salt. All other tests revealed no indication for any costs of plasticity to saline conditions. (C) 2001 Editions scientifiques et medicales Elsevier SAS.

Appendix – III

```
# include <stdio.h>
 # include <stdlib.h>
  int get_field(char line[]);
        FILE *fp,*fpt;
 main()
  {
        char str[2056];
        char ch;
        char strsub1[]="Authors\t";
        char replace1[]="300 ";
        char strsub2[]="Title\t";
        char replace2[]="200 ";
        char strsub3[]="Full source\t";
        char replace3[]="020 ";
        char strsub4[]="Author keywords\t";
        char replace4[]="620 ";
        char strsub5[]="KeyWords Plus\t";
        char replace5[]="625 ";
        char strsub6[]="TGA/Book No.\t";
        char replace6[]="011 ";
        char strsub7[]="Discipline\t";
        char replace7[]="610 ";
        char strsub8[]="Document type\t";
        char replace8[]="060 ";
        char strsub9[]="Language\t";
        char replace9[]="040 ";
        char strsub10[]="Address\t";
        char replace10[]="330 ";
        char strsub11[]="ISBN/ISSN\t";
        char replace11[]="101 ";
        char strsub12[]="Publisher\t";
        char replace12[]="400 ";
        char strsub13[]="Abstract\t";
        char replace13[]="600 ";
        char field[15];
        int i=0,count=0,j=0,l=0,space_count,comma_count;
        fp=fopen("ag100.txt","r");
        fpt=fopen("new.txt","w");
        fgets(str,2056,fp);
          str[strlen(str)-2]='\0';

        while(!feof(fp))
        {
```

```
switch(get_field(str) )
     {
       case 0:
               break;
     case 1:
               space_count=0;
                if(strstr(str,strsub1))
        {
                fputs(replace1,fpt);
                  fputs("^f",fpt);
                for(j=8;str[j]!='\0';j++)
                {
                      if(str[j]==',')
                      {
                          ch='%';
                                fputc(ch,fpt);
                                j++;
                      }
                      if(str[j]==' ')
                                {
                                  space_count++;
                                 if(space_count==1)
                                 {
                                     j++;
                                     fprintf(fpt,"^l");
                                 }
                                    else if(space_count==2)
                                    {
                                       j++;
                                       fprintf(fpt,"^f");
                                    }
                                      else if(space_count==3)
                                      {
                                         j++;
                                         fprintf(fpt,"^l");
                                      }
                                    else
                       {
                                      j++;
                                      fprintf(fpt,"^");
                       }
                          }
                        fputc(str[j],fpt);
                }
                fputc('\n',fpt);
         }
```

```
    count++;
      break;
case 2:
      if(strstr(str,strsub2))
        fprintf(fpt,"%s",replace2);
        split_rec(str,6);
      fputc('\n',fpt);
      break;
case 3:
          comma_count=0;
      if(strstr(str,strsub3))
      {
         fputs(replace3,fpt);
         for(j=12;str[j]!='\0';j++)
         {
                  if(str[j]==',')
                  {
                           comma_count++;
                           if(comma_count==1)
                           {
                                    j+=2;
                                    fprintf(fpt,"\n440 ");
                           }
                           else if(comma_count==2)
                           {
                                    j+=2;
                                    fprintf(fpt,"\n490 ^a");
                           }
                           else if(comma_count==3)
                           {
                                    j+=2;
                                    fprintf(fpt,"^b");
                           }
                           else
                           {
                                  j+=2;
                                    fprintf(fpt,"^c");
                           }
                  }
             fputc(str[j],fpt);
              }
      }

      fputc('\n',fpt);
      break;
```

```
case 4:

            if(strstr(str,strsub4))
              fprintf(fpt,"%s",replace4);
              split_rec(str,16);
            fputc('\n',fpt);
            break;
      case 5:

            if(strstr(str,strsub5))
              fprintf(fpt,"%s",replace5);
              split_rec(str,14);
            fputc('\n',fpt);
            break;
      case 6:

            if(strstr(str,strsub6))
           {
              fputs(replace6,fpt);
              for(j=13;str[j]!='\0';j++)
              {
                        ch=str[j];
                 fputc(ch,fpt);
           }
              }

           fputc('\n',fpt);
           break;
      case 7:

            if(strstr(str,strsub7))
           {
              fputs(replace7,fpt);
              for(j=11;str[j]!='\0';j++)
              {
                        ch=str[j];
                 fputc(ch,fpt);
           }
              }

           fputc('\n',fpt);
           break;

   case 8:
```

```
        if(strstr(str,strsub8))
           {

         fputs(replace8,fpt);
               for(j=14;str[j]!='\0';j++)
               {
                        ch=str[j];
                  fputc(ch,fpt);
           }
               }

           fputc('\n',fpt);
           break;
      case 9:

            if(strstr(str,strsub9))
           {
               fputs(replace9,fpt);
               for(j=9;str[j]!='\0';j++)
               {
                        ch=str[j];
                  fputc(ch,fpt);
           }
               }

           fputc('\n',fpt);
           break;
      case 10:

            if(strstr(str,strsub10))
               fprintf(fpt,"%s",replace10);
               split_rec(str,8);
           fputc('\n',fpt);
           break;


case 11:

            if(strstr(str,strsub11))
           {
              fputs(replace11,fpt);
               for(j=10;str[j]!='\0';j++)
               {
                        ch=str[j];
                  fputc(ch,fpt);
```

```
          }
               }

            fputc('\n',fpt);
            break;
       case 12:

             if(strstr(str,strsub12))
               fprintf(fpt,"%s",replace12);
               split_rec(str,10);
            fputc('\n',fpt);
            break;
       case 13:

             if(strstr(str,strsub13))
               fprintf(fpt,"%s",replace13);
               split_rec(str,9);
            fputc('\n',fpt);
            break;

       case -1: //printf("13%s",str);  // to handle failure to match a field //
            fprintf(fpt,"##\n");

       } // end of switch //
       fgets(str,2056,fp);
         str[strlen(str)-2]='\0';
   }
   fclose(fp);
   fclose(fpt);
}




/*=====================================*/
int get_field(char line[])
{
int num=0;
char field[15]="";
  char allfields[15][16]={"Record",
"Authors",
"Title",
"Full source",
"Author keywords",
"KeyWords Plus",
```

```
"TGA/Book No.",
"Discipline",
"Document type",
"Language",
"Address",
"ISBN/ISSN",
"Publisher",
"Abstract"
};


        for(num=0;num<14;num++)
        {
                if( strncmp(line,allfields[num],strlen(allfields[num]))==0 )
                {
                        return num;
                }
        }
//printf("NOT FOUND!\n");
return -1;

        return(-1); // to indicate failure to match any of above fields //
}


split_rec(field_value,start_posn)
  char field_value[];
  int start_posn;
{
  int m,i,j,k,len,printed;
  char fld_val[86];

  i=k=m=len=printed=0;
  j=0;
  fld_val[0]='\0';
  len=strlen(field_value);
  for (i=start_posn;i<=len;i++)
  {
    fld_val[j]=field_value[i];
    j++;
    printed=0;
    if(j>75)
    {
      fld_val[j]='\0';
      j=0;
```

```
  if(m==0)
     {
       fprintf(fpt,"%s\n",fld_val);
          printed=1;
       m=1;
     }
     else
     {
       fprintf(fpt,"    ");
       fprintf(fpt,"%s\n",fld_val);
          printed=1;
     }
     //printf("\n%s:",fld_val);
     fld_val[0]='\0';
     /*printf("press enter to continue: ");
     getc(stdin);*/
    }
  }

  if(!printed)
  {
     if(m==0)
     { fprintf(fpt,"%s",fld_val); }
     else
     { fprintf(fpt,"    %s",fld_val); }
  }
}
```

Appendix – IV

SAMPLE RECORD:

(ISO2709)

02628000000000205000450030000330000020000790003302000510011244000050016349000280016862000590019662501310025501100060038661000220039206000080041404000080042233000960043010100100052640000710053660018150060 7#^fMJ^lSmekens%^fPH^lvanTienderen#Genetic variation and plasticity of Plantago coronopus under saline conditions#Acta Oecologica - International Journal of Ecology#2001#^aVol 22^bIss 4^cpp 187-200#salt adaptation; Plantago coronopus; phenotypic plasticity#PHENOTYPIC PLASTICITY; REACTION NORMS; SALT TOLERANCE; LIFE-HISTORY; EVOLUTION; LIMITS; GROWTH; POPULATIONS; ALLOCATION; SELECTION#488NK#Environment / Ecology#Article#English#van Tienderen PH, Netherlands Inst Ecol, Ctr Terr Ecol, POB 40, NL-6666 ZG Heteren, NETHERLANDS#1146-609X#Gauthier-Villars/Editions Elsevier, 23 RueLinois, 75015 Paris, France#Phenotypic plasticity may allow organisms to cope with variation in the environmental conditions they encounter in their natural habitats. Salt adaptation appears to be an excellent example of such a plastic response. Many plant species accumulate organic solutes in response to saline conditions. Comparative and molecular studies suggest that this is an adaptation to osmotic stress. However, evidence relating the physiological responses to fitness parameters is rare and requires assessing the potential costs and benefits of plasticity. We studied the response of thirty families derived from plants collected in three populations of Plantago coronopus in a greenhouse experiment under saline and non-saline conditions. We indeed found a positive selection gradient for the sorbitol percentage under saline conditions: plant families with a higher proportion of sorbitol produced more spikes. No effects of sorbitol on fitness parameters were found under non-saline conditions. Populations also differed genetically in leaf number, spike number, sorbitol concentration and percentages of different soluble sugars. Salt treatment led to a reduction of vegetative biomass and spike production but increased leaf dry matter percentage and leaf thickness. Both under saline and non-saline conditions there was a negative trade-off between vegetative growth and reproduction. Families with a high plasticity in leaf thickness had a lower total spike length under non-saline conditions. This would imply that natural selection under predominantly non-saline conditions would lead to a decrease in the ability to change leaf morphology in response to exposure to salt. All other tests revealed no indication for any costs of plasticity to saline conditions. (C) 2001 Editions scientifiques et medicales Elsevier SAS.##

Appendix – V

Addresses of Freely Available Software to convert the Bibliographic records into ISO2709:

http://www.bib.wau.nl/isis/software.html

Fangorn:

Hugo Besemer,

Bredeweg 17,6668 AR Randwijk,

the Netherlands, e-mail BESEMER@JKA.WAU.NL .

TXT2ISO:

L.J Haravu (Manager),

Library & Documentation Services

ICRISAT, Patancheru P.O.

Andhra Pradesh 502 324, India