PROJECT REPORT (VOLUME II)

NATURAL LANGUAGE PROCESSING SYSTEM USING PUNJABI

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF COMPUTER APPLICATIONS

BY

HIMNISH NARANG  JAGMEET KAUR
(8/90)         (9/90)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY
PATIALA - 147 001

(DEEMED TO BE A UNIVERSITY)

MAY 1993
#include<fcntl.h>
#include<time.h>
app_dbf(dbfNam,instr)
char *instr,*dbf_nam;
!
char ch[901],cd[31],cg,*tmpbuf;
int i,hdsiz,fa,tt,c;
fa=open(dbf_nam,O_RDWR);
tmpbuf=(char *)malloc(47);
memset(tmpbuf,32,47);
memcpy(tmpbuf,instr,strlen(instr));
puts(tmpbuf);
1seek(fa,0L,2);
write(fa,tmpbuf,47);
close(fa);
#include<stdio.h>
#include<string.h>
#include<fcntl1.h>
#include<memory.h>
#include<malloc.h>
#define null 0

A dictionary is maintained that contains the commonly used words of punjabi and their corresponding english words. This structure is used to fetch the information about various words from the dictionary. In head.w we get the Punjabi word, in head.wik the type of the word whether it is a function, a database word, or the value. head.ewr contains the corresponding english words of a punjabi word, and header.g contains the information whether the word is useful for the retrieval of the query or not. header.g contains the garbage.

typedef struct
{
    char w[30];
    char ty[2];
    char ewr[12];
    char use[2];
    char g;
} head;

The structure list is made to keep the information about the useful words of RAM. An array of structures is declared, which helps in keeping the information about the useful words. The different fields of the structure contain different information. In ewr field of the structure the corresponding english word of the punjabi word is stored. In order to make the system portable to any database the field offset is declared. We can retrieve the information about any database by filling in dictionary the corresponding punjabi words of the database fields and then calculating the offset, i.e. how far from a
address the information for that field will be retrieved. The field c_type
of the structure contains the information, whether the word is a database
function or a value of some database field. The field type of the structure
contains the information whether the word is of type character or numeric
if it is a database field. The field length of the structure contains the
length of the field, if the word corresponds to a database field.

struct list{
    char ewr[12];
    unsigned char offset;
    char c_type;
    char type;
    char length;
    char dep;
} list;

/*
This structure is used to read the header of the database file. In the
database file the header contains the information like type of the file,
time of last update, no_of_records, header_size, record size. By reading
this information in a structure, we can get the required information */

typedef struct
    char magic_no;
    char last_up[3];
    long no_of_records;
    short header_size;
    short record_size;
    char garbage[20];
} here;

/*
This structure is used to read the field description from the database
file. The field description in the database file contains the information
like field_name, field_type, field_length, field_decimal etc. The description
of each field takes 32 bytes and we can get useful information in this
structure */
typedef struct
{
    char field_name[11];
    char field_type;
    long garbage1;
    char field_length;
    char field_decimal;
    char garbage2[14];
}fld;

/*This field of type here is to read the header of the database file*/
/*Here name;*/

/* This variable of type fld is to read the description of each field of*/
/*database file*/
fld field;

/*This variable is used to read the description of each word of dictionary*/
head header;

/* this is the array of structures of type lis to store the information*/
/*regarding useful words of the query*/
struct lis new[10];
char t[3],len[3],ty[2],cty[2],x[3],ax[3];
char st[12],p,bx[3],cx[3],mn[13];
int z,fd,k,norec,fg;
char cf, ch[100],d[31],e[13],u[31],ind[30],value[30],s[31],*inn, ch1[30];
char cnt,t1[13],t2[13],c[131],jh[12];
int i=0,j=0,o=0,g,b,w,coun,c[10],cn;

main()
|
char ans='Y';
char str1[100], str2[30];
int choice = 1;

while (choice < 4)
{

/* The following message gets displayed on the screen */

printf("\n"");
printf("\n");
printf("\n");
printf("\n");
printf("1. xUfmB xNLgUBm xYbUK xHkJnASr xjt? *\n");
printf("2. xVwgUG xVngPkJk xHkJnASr xjt? *\n");
printf("3. DATABASE xSm xJKPdKbGm xSrEPkJk xHkJnASr xjt? *\n");
printf("4. XXkJb *\n");
printf("\n");
printf("\n");
printf("\n");
printf("\n");
printf("\n");

} /* End of while loop */

/* Open text file*/
fd = open("des.txt", O_RDONLY);

/* Open database file */
fg = open("villdata.dbf", O_RDONLY);
fflush(stdin);

/* The user is prompted to enter his choice*/

printf("enter choice");
scanf("%d", &choice);

/*case starts to suggest different procedures for user's choice*/
switch (choice)
{
    case 1:

    /* This function appends new words in the dictionary*/
    append();
}
break;

case 2:

/* This function displays the information in punjabi */
    printf();
/* total processing from accepting the query to display the answer*/
do{
  
    i=0;
    memset(str1,0,100);
    fflush(stdin);

    printf("xCKVPk xVwbqU xdlEt");

    /* Accept user's query*/
    gets(str1);

    /* Get one word from the query */

    while(str1[i]!=0)
    {
      memset(str2,0,20);
      j=0;
      do
      {
        str2[j]=str1[i];
        if ((str2[j]== -45) || (str2[j] == -46))
          str2[j] = -47;
          i=i+1;
      j=j+1;
    }while((str1[i]==32) && (str1[i]!=0));
    i++;
    memset(st,0,30);
    memcpy(st,str2,j);
    st[j]=0;
/* For each word in the query, retrieve information from the dictionary */

/* Call function pr() */

pr();

/* If the word is useful, append information in array */

if(strcmp(t,bx)==0)
 |
  memcpy(new[gl.ewr,header.ewr,12];
  new[gl.offset]=cnt;
  new[gl.type]=cty[0];
  new[gl.c_type]=ty[0];
  new[gl.length]=len[0];
  g++;
  }

/* if the word is value, store dev = 'i' for the previous element of the array */

if(strcmp(s,cx)==0)
{  
  o=g-2;  
  new[o].dep='i';
  }

/* After retrieving and storing the information about the useful words of the query, and storing them in the array, retrieve the information from array to reply the query. For this purpose it call function count */

  count();

/* The user is asked whether he wants to ask more queries */

  printf("xjtb xWwbgU xVnxIPk xHkJnASr xjt");
  scanf("%c", &ans);
  printf("\n");

/* go to the first element of the array */

  z=0;
  q=0;
close(fd);
fd=open("des.txt",O_RDONLY);
close(fg);
fg=open("villedata.dbf",O_RDONLY);
while(ans=='y'){
  break;
}
case 3:
  printf();
  break;

  case 4:
  exit(0);
  }
  
/***************************************************************************/
/* This function is called in main. It searches each word of */
/* the query in the dictionary. It gets information about each */
/* word from the dictionary. */
/***************************************************************************/
pr()
{
  int a,i;

  /* seek dictionary file from the beginning */
  lseek(fd,0L,0);
  /* Initialize all the variables */
  memset(ty,0,3);
  memset(ch,0,30);
  memset(s,0,3);
  memset(ety,0,3);

  /* Scan the dictionary file sequentially to get the information of */
  /* the word */
do
|
read(fd,&header,sizeof(header));
memcpy(ch,header.w,30);
ch[30]=0;
while(memcmp(ch,st,strlen(st)) && (header.w[0]!=EOF));
/*store the info. about the usefulness of the word */
memcpy(t,header.use,2);
t[2]=0;
memset(s,0,3);
memset(x,0,3);
memset(ax,0,3);
memset(bx,0,3);
strcpy(bx,"us");
/* retrieve and store the information, if the word is useful*/
if(!strcmp(t,bx))
|
memset(s,header.ty,2);
s[2]=0;
strcpy(x,"dw");
strcpy(ax,"fn");
strcpy(cx,"va");
/*********************************************************************************/

/* Information to be retrieved, and later to store in the array
of structures, if the variable corresponds to a database field */
if (strcmp(s,x)==0)
|
/*variable to count the offset */
cnt=0;
/* seek database file from the origin */
lseek(fg,0L,0);

/* read the information in the header of the database file */
read(fg,&name,sizeof(name));
/* Scan the fields of the database file sequentially to get 
the information like field length, field offset if the record is 
 fetched at a address*/

do
 |
j=read(fg,&field,sizeof(field));
/* calculate cnt by summing up the field_lengths*/
cnt=cnt+field.field_length;

/* size offset is declared of type character, hence more than 
128 value can not be stored in it. This following step is to 
store the value above 128*/

if(cnt<0)
 |
c[2]++;
cnt=1-cnt-127;
}

/* store the field name of each field in this variable*/
memcpy(e,field.field_name,12);
e[12]=0;

/* Repeat the name until the field name corresponding to punjabi word 
found */

}while(strncmp(header.ewr,e,strlen(e)));

/* When the fiel is found, store length and type of the field*/
memset(len,0,2);
len[0]=field.field_length;
memset(ty,0,3);
ty[0]=field.field_type;
cty[0]='d';

***************************************************************************/

/* if the word is a function, store f to be appended, in the type 
field of the structure*/
***************************************************************************/

if(strncmp(s,ax)==0)
 |
c[0]="f";
}
/** if the word is a function, store v to be appended, in the type field of the structure */
if(strcmp(s, cx) == 0)
{
    cty[0] = 'v';
}
if(!strcmp(t, bx))
{ 
z++; 
c[z] = 0; 
}

/* This function is to retrieve the information from the database */
/* It reads the information stored in the array and then uses this information to reply the user queries */
count() 
{
    int i, a; 
    cn = 0; 
    coun = 0; 

    /* Scan the array to get an element with type='f', so as to know which function is to be performed */
    for(b=0; b<=g; b++)
    {
        if (new[b].type == 'f')
            break;
    }
    memset(mn, 0, 13);
    memcpy(mn, new[b].ewr, 12);

    /* read the header and field description of the database file */
    lseek(fg, OL, 0);
read(fg, &name, sizeof(name));
  w=0;
  while(w<((name.header_size -32)/32)-1)
  {
    read(fg, &field, sizeof(field));
    w++;
  }
  read(fg, &field, sizeof(field)+1);

  /* Read the array for type='v' so that we can know for what value
     of a field, we have to scan the database */

  for(b=0; b<=g; b++)
  {
    if((new[b].type)=='v')
      break;
  }

  /* store the value in a variable */
  memset(ind, 0, 30);
  strcpy(ind, new[b].ewr);

  /* Read the array for dep='i' so that we can know we have to retrieve a record
     with that value for which field */
  for(b=0; b<=g; b++)
  {
    if((new[b].type)=='d') & (new[b].dep=='i')
      break;
  }
  if((g-b)>0)
    do
      

/* scan the database sequentially to get a record with the value ind of the field */
   do
      { 
         inn = malloc(name.record_size+1);
         memset(inn,0,name.record_size+1);
         read(fg,inn,name.record_size);
         memset(ch1,0,30);
         memcpy(ch1,inn[((c[b]*128)+new[b].offset)-new[b].length+1],new[b].length);
         while((strcmp(ind,ch1,strlen(ind))!=0) && (inn[0]!=0));
   }/* scan the whole array to find we have to get the value of which field */

   for(b=0;b<g;b++)
   { 
      if((new[b].type=='d') && (new[b].dep==0))
         break;
   }/* inset the value at the offset of that field in a variable */

   memset(ch1,0,30);
   memcpy(ch1,inn[((c[b]*128)+new[b].offset)-new[b].length+1],new[b].length);

   /* Action to perform if the function is list*/
   if(memcmp(mn"list",4))=0)
   
      if (new[b].c_type =='C')
      {
         /* read the dictionary database to read the punjabi counterpart of the word*/
         close(fd);
         fd=open("des.txt",O_RDONLY);
         lseek(fd,0L,0);
do
{
  read(fd,&header,sizeof(header));
  memset(ci,0,13);
  memcpy(ci,header.ewr,12);
  ci[12]=0;
  while(strncmp(ci,ch1,strlen(ci)))
  {
    memset(ch,0,31);
    memcpy(ch,header.w,30);
    ch[31]=0;
    puts(ch);
  }
  else
  {
    puts(ch1);
  }
}

/*action to perform if the function is count */
if(strncmp(nn,t2,strlen(t2))==0)
{
  if(new[b].c_type=='N')
  {
    cn=cn+atoi(ch1);
    printf("%d",cn);
  }
/*if the value stored in field is (y/n);*/
if ( ((new[b].c_type=='C') && (ch1[0]=='Y'))
{
  coun=coun+1;
}
for(b=0;b<=g;b++)
{
  if((new[b].type=='d') && (new[b].dep=='i'))
    break;
  printf("\n%\n",inn[0]);
/*loop untill end of database file */
}while(inn[0]!=0);
printf("%d",coun);
printf("%d",cn);
/* This function reads the fields names from the database file, takes their punjabi counterparts from the dictionary file and then displays the whole list*/

printf();

int a;
char rs[12];
fflush(stdin);
fd = open("des.txt",O_RDONLY);
fg = open("villdata.dbf",O_RDONLY);
seek(fg,0L,0);
read(fg,&name,sizeof(name));
i=0;
/* Read field names sequentially from the database file */
do
{read(fg,&field,sizeof(field));
memset(jh,0,12);
memcpy(jh,field.field_name,11);
seek(fd,0L,0);
/*read their punjabi counterparts from the dictionary file*/
do
{read(fd,&header,sizeof(header));
memset(rs,0,12);
memcpy(rs,header.ewr,11);
while(strcmp(jh,rs,strlen(jh))!=0);
memset(jh,0,12);
memcpy(jh,header.w,11);
puts(jh);
i++;
}while(i<=(name.header_size -32)/32);


append()
{
    int fp,i;
    char ci[31], cp[12], ck[47], cr[30], cl[12], an='y', ab;
    printf("In Order To Make The System Portable To Any System The User");
    printf("\n");
    printf("Is Requested To Append In The Dictionary Only The Field Names");
    printf("\n");
    printf("And Their Corresponding Punjabi Words or The Values That These");
    printf("\n");
    printf("Fields Can Take");
    printf("\n");
    printf("\n");

do
{
    fflush(stdin);
    printf("Enter value or field name (v/f) ");
    scanf("%c", &ab);

    /* action if the user wants to append a field */
    if(ab=='f')
    {
        fflush(stdin);
        printf("Enter field_name ");
        gets(cp);
        printf("\n");
        printf("Enter Punjabi word ");
        gets(ci);
        memset(cr, 0, 30);
        for(i=0; i<(30-strlen(ci)-1); i++)
            cr[i]=32;
        cr[i]=0;
        memset(cl, 0, 12);
        for(i=0; i<(12-strlen(cp); i++)
            cl[i]=32;
cl[i]=0;
strcat(ck,ci);
strcat(ck,cr);
strcat(ck,"dw");
strcat(ck,cp);
strcat(ck,cl);
strcat(ck,"us");
strcat(ck,"\n");
printf("%d",strlen(ck));

/* This procedure is written in db1 file, which was compiled along with the program */

app_dbf("des.txt",ck);

/* action if the user wants to append a value */
if(ab!='v')
{
    fflush(stdin);
    printf("enter value");
    gets(cp);
    printf("\n");
    printf("enter punjabi word");
    gets(ci);
    memset(cr,0,30);
    for(i=0;i<(30-strlen(ci)-1);i++)
    {
        cr[i]=32;
    }
    cr[i]=0;
    memset(cl,0,12);
    for(i=0;i<12-strlen(cp);i++)
    cl[i]=32;
    cl[i]=0;
    strcat(ck,ci);
    strcat(ck,cr);
    strcat(ck,"va");
    strcat(ck,cp);
    strcat(ck,cl);
    strcat(ck,"us");
    strcat(ck,"\n");
    app_dbf("des.txt",ck);
}
printf("enter more");
scanf("%c", &an);
while (an == 'y');