ภาคผนวก

PSUbase (PSU DataBase Management System)

Date-Written: SEP 1992
By : Mongkol Kuanhavet
: Head of Application development section.
: Computer Center, Prince of SONKLA Univ.,Hatyai,Songkla,Thailand.

Summary of commands in the PSUbase

Commands	Usage	Description
append	append {fn1 from fn2}	Add records to end of file.
average	average {fn field_name}	Compute the arithmetic mean and stdv.
copy	copy {fn1 to fn2}	Make a copy of database file.
create	create {fn}	Create a structrue for new file.
fcreate	fcreate {fn}	Create a structrue for new file from file.
delete	delete {fn} [arg]	Mark specified record(s) for deletion.
dir	dir {fn,*}	Display specified files.
struct	struct {fn,*}	Display the record structrue of files.
editdbf	editdbf {fn}	Display database fields for editing.
erase	erases {fn}	Remove database file.
list	list {fn} [arg]	Display specified records.
modify	modify {fn}	Modify the record structrue.
pack	pack {fn} [arg]	Permanently remove records marked.
recall	recall {fn} [arg]	Recall the Database File.
reindex	reindex {fn}	Rebuild the index file.
replace	replace {fn with str}	Change data in specified field.
rename	rename {fn1 to fn2}	Rename database file.
sortdbf	sortdbf {fn1 to fn2}	Sort the database file.
sum	sum {fn field_name}	Add the numeric field values.
zap	zap {fn,*}	Remove all records specified files.

Summary of functions in the library libinx.a and libdbf.a

Fucntions	Description
append();	Append the new record at the end of database.
beep();	Echo the beep sound.
bof();	Return the begin-of-file status, true or false.
clean();	Clean or clear the buffer record of database.
clear();	Clear the screen.
closedata();	Close all the database files.
comma(so,si);	Insert the comma into string so.
date();	Return the current system date.
delete();	Mark the current record for deletion.
endscr();	Return to the normal mode of screen.
eof();	Return the end-of-file status, true or false.
file(fn);	Check the file is exists, true or false.
flockdbf();	Lock the current file.
found();	Return the found status, true or false.
getchxy();	Get the one character in raw mode.

```
getfielddec(n);          Get the field decimal #n in the database file.
getfieldoffs(n);         Get the field offset #n in the database file.
getfieldkey(field,n);    Get the field status is the key in the database file.
getfieldlen(n);          Get the field lenght #n in the database file.
getfieldname(field,n);   Get the field name #n in the database file.
getfieldtype(field,n);   Get the field type #n in the database file.
getkeyinfo(rec,len,of); Get the key lenght and key offset in the database file.
gettfield();             Get the total number of the field in the database file.
gettrec();               Get the total records in the database file.
getyn();                 Get one character and check for 'y' or 'n'.
go(n);                   Go to the record number n in the database file.
gobuttom();              Go to the last record in the database file.
gotop();                 Go to the first record in the database file.
goyx(y,x);               Go to position y,x <row>,<col> axis on the screen.
initscr();               Set screen to the raw mode and non echo.
isdbf(fn);               Check file fn is the database file.
isdelete();              Check record is the deleted.
lthai(c,s,l1,l2,l3,l4); Transform the 1 line Thai char. to 4 lines.
ltrim(s);                Removed all heading blank of string.
mapchar(field,cvar);     Mapping the field name with the character variable.
mapreal(field,rvar);     Mapping the field name with the real variable.
defchar(field,cvar);     Define field name type char. for sayget().
defreal(field,rvar);     Define field name type real for sayget().
recall();                Unmark the current record for deletion.
recno();                 Return the current record number.
repl(s,n);               Repeat the number of times specified by n.
replace();               Replace all the data in field_name to database file.
replchar(field,cvar);    Replace the data in character variable to field_name.
replreal(field,rvar);    Replace the data in real variable to field_name.
say(y,x,s);              Print the string to the position x,y on screen.
sayget(sp);              Say/Get is a full-screen edit operation(see sam8.c).
seekdbf(cvar);           Access the records of database file with key.
select(n);               Select the work area for open the database file.
setbell(n);              Set BELL to OFF/ON, (n=0 or 1).
setcolor(sc);            Set color to sc option.(sc=W+,/W,..).
setconfirm(n);           Set confirm to OFF/ON, (n=0 or 1).
skip(n);                 Get next n records in the database file.
space(n);                Return the spaces specified by n.
stolower(s);             Conversion all characters in string to lower case.
stoupper(s);             Conversion all characters in string to upper case.
text(s);                 Display the lines of text to the screen.
time();                  Return the current system time.
trim(s);                 Removed all trailing blank of string.
unflockdbf();            UnLock the current file.
use(fn,mode);            Use the database file.
```

```
::::::::::::::
cst_db.h
::::::::::::::
/*****************************************************************************
*
* cst_db.h: This is a const value for program in PSUbase
*           Written by mongkol Kuanhavet, 18 Jun 1992
*           Computer center, Prince of Songkla univ.,Hatyai, Thailand.
*
*****************************************************************************
*/
#define  MAXFILE    64          /* Maximum value for open file PSUbase */
#define  MAXFIELD  256          /* Maximum value for field */
#define  MAX1025  1025          /* Maximum of buffer size */
#define  MAXPULL   256          /* Maximum value of Pull list */
#define  BSPACE   0x7F          /* Back space code */
#define  UP       0x05          /* Up arrow */
#define  DOWN     0x18          /* Down arrow */
#define  LEFT     0x13          /* Left arrow */
#define  RIGHT    0x04          /* Rigth arrow */
#define  RETURN   0x0D          /* Return or Enter */
#define  CTRL_A   0x01          /* Control A */
#define  CTRL_B   0x02          /* Control B */
#define  CTRL_C   0x03          /* Control C */
#define  CTRL_D   0x04          /* Control D */
#define  CTRL_E   0x05          /* Control E */
#define  SPACE    0x20          /* Space bar */
#define  BEEP     0x07          /* Beep sound */
#define  ESC      0x1B          /* Escape */
#define  CTRL_K   0x0b          /* Control K */
#define  CTRL_N   0x0E          /* Control N */
#define  CTRL_P   0x10          /* Control P */
#define  CTRL_I   0x09          /* Control I */
#define  CTRL_U   0x15          /* Control U */
#define  CTRL_X   0x18          /* Control X */
#define  CTRL_Y   0x19          /* Control Y */
#define  CTRL_F   0x06          /* Control F */
#define  CTRL_R   0x12          /* Control R */
#define  CTRL_G   0x07          /* Control G */
#define  CTRL_O   0x0f          /* Control O */
/* End-of-file */
::::::::::::::
ut_db1.c
::::::::::::::
/*****************************************************************************
*
* ut_db1.c: This is a utility file #1 for the PSUbase
*           Written by Mongkol kuanhavet, 18 Jun 92
*           Computer center, Prince of Songkla Univ., Hatyai, Thailand.
*
* Functions:
*     select()        Select the work area for I/O buffer file.
*     use()           Use or Open the file for read/write.
*     use2()          Use or Open the file for read/write specified by fn.stc
*     mapchar()       Mapping address of character varible to the field name.
*     mapreal()       Mapping address of real varible to the field name.
```

```
 57  *     mapall()        Mapping address of character varible all field name.
 58  *     mapseq()        Mapping sequence of field name in file.
 59  *     getfieldname()  Get the field name.
 60  *     getfieldtype()  Get the field type.
 61  *     getfieldkey()   Get the key field.
 62  *     getfieldlen()   Get the field length.
 63  *     getfieldoffs()  Get the key offset.
 64  *     getdelcol()     Get the position column for deleted.
 65  *     gettfield()     Get total number of fields.
 66  *     gettrec()       Get total number of records.
 67  *     getkeyinfo()    Get information of key.
 68  *     skip()          Skip the currect record, backward or forward.
 69  *     seekdbf()       Seek or random access by key.
 70  *     gotop()         Go to the first records.
 71  *     gobutton()      Go to the last records.
 72  *     get_rec()       Get the data from buffer area.
 73  *     go()            Go to the record number.
 74  *     getfield()      Get the data from buffer to variable.
 75  *     replchar()      Replace the character data from variable to field.
 76  *     replreal()      Replace the real data from variable to field.
 77  *     replreal0()     Replace the real data from variable to field insert zero.
 78  *     replace()       Replace the data in field to file.
 79  *     delete()        Mark delete record.
 80  *     recall()        Unmark deleted record.
 81  *     isdelete()      Check for mark delete record.
 82  *     append()        Append the record to file.
 83  *     found()         Check for found after seekdbf.
 84  *     eof()           Check for end of file.
 85  *     bof()           Check for begin of file.
 86  *     recno()         Ark the record number of file.
 87  *     flockdbf()      Lock the file for sharing.
 88  *     unflockdbf()    Unlock the file for sharing.
 89  *     closedata()     Close all the PSUbase file.
 90  *     isdbf()         Check for the PSUbase file.
 91  *     file()          Ark the file is exist.
 92  *     clean()         Clear the screen.
 93  *     freemem()       Release the working memory.
 94  ********************************************************************************
 95  */
 96  #include <stdio.h>
 97  #include <ctype.h>
 98  #include <math.h>
 99  #include <malloc.h>
100  #include "cst_db.h"
101
102  double atof();
103  static FILE   *_fpinx      [MAXFILE][2];           /* file pointer */
104  static char    _statfp     [MAXFILE];              /* status of fp. */
105  static char    _fieldname  [MAXFILE][MAXFIELD][11]; /* array for field name */
106  static char    _fieldtype  [MAXFILE][MAXFIELD];    /* array for field type */
107  static char    _fieldkey   [MAXFILE][MAXFIELD];    /* array for field key */
108  static int     _fieldlen   [MAXFILE][MAXFIELD];    /* array for field len. */
109  static char    _fielddec   [MAXFILE][MAXFIELD];    /* array for field decimal*/
110  static int     _fieldoffset[MAXFILE][MAXFIELD];    /* array for field offset */
111  static char    _fieldstat  [MAXFILE][MAXFIELD];    /* array for field status */
112  static char  **_fieldchar  [MAXFILE][MAXFIELD];    /* array for field char */
```

```
113  static char   *_fieldtmp   [MAXFILE][MAXFIELD];     /* array for field buff */
114  static double *_fieldreal  [MAXFILE][MAXFIELD];     /* array for field real */
115  static int     _fieldseq   [MAXFILE][MAXFIELD];     /* array for field seq. */
116  static int     _delcol     [MAXFILE];               /* delete column */
117  static char   *_srec       [MAXFILE];               /* buffer for data record */
118  static int     _sumfield   [MAXFILE];               /* sum of field */
119  static int     _eofstat    [MAXFILE];               /* eof status */
120  static int     _foundstat  [MAXFILE];               /* found status */
121  static int     _currec     [MAXFILE];               /* current record */
122  static int     _reclen     [MAXFILE];               /* record lenght */
123  static int     _keylen     [MAXFILE];               /* key lenght */
124  static int     _keyoffs    [MAXFILE];               /* key offset */
125  static int     _trec       [MAXFILE];               /* total record */
126  static long    _lret       [MAXFILE];               /* record number */
127  static int     _se;                                 /* current working area */
128  static char    _init;                               /* initial status */
129
130  /*****************************
131  * select()
132  *****************************
133  */
134  select(n)
135  int n;
136  {
137    if (n<1 || n>=MAXFILE) {          /* if out of rang */
138      fprintf(stderr,"** Error: select(n);  n>0 and n<%d\n",MAXFILE);
139      exit(-1);
140      _se=0;
141    } else
142      _se=n;
143  }
144
145  /*****************************
146  * use()
147  *****************************
148  */
149  use(fn,mode)
150  char *fn,*mode;
151  {
152    FILE *fpstc;                          /* File pointer for fn.stc */
153    char s[200],fnstc[80],shead[513];
154    long ihead[1025];                     /* buffer for header of index file */
155    int  i,ioffset;
156
157    if (_se<1 || _se>=MAXFILE)
158      error("** Error: use(fn,mode); never select area, select(n)"," ");
159
160    trim(fn);
161    if (strlen(fn)==NULL) {           /* select(" ","r") for close file */
162      if (_statfp[_se]==1) {          /* if open, close it */
163        fcloseinx(_fpinx[_se]);
164        _statfp[_se]=0;               /* reset status to 0 */
165        freemem();                    /* free memory */
166      }
167      return;
168    }
```

```
169     if (_statfp[_se]==1) {          /* if open, close and reopen */
170       fcloseinx(_fpinx[_se]);
171       freemem();
172     }
173     if (fopeninx(fn,mode,_fpinx[_se])==NULL)      /* open index file */
174       error("** Error: use(fn,mode); fn not found:",fn);
175
176     _read_header(_fpinx[_se],ihead,shead);        /* get header of index file */
177     _srec[_se]=(char *)malloc((int)ihead[2] +2);  /* allocate memory for I/O */
178
179     strcpy(fnstc,fn);
180     strcat(fnstc,".stc");
181     if ((fpstc=fopen(fnstc,"r"))==NULL)           /* open the file fn.stc */
182       error("** Error: use(fn,mode); fn not found:",fnstc);
183
184     i=0;
185     ioffset=1;
186     while (fgets(s,200,fpstc)!=NULL.&& i<MAXFIELD) {  /* get field information*/
187       if (s[0]=='#'||s[1]=='#')                   /* end of information */
188         break;
189       if (s[0]=='/'&&s[1]=='*')                   /* if comment, skip it */
190         continue;
191       substr(s,1,10,_fieldname[_se][i]);          /* get field name */
192       trim(_fieldname[_se][i]);
193       _fieldtype  [_se][i]=tolower(s[11-1]);      /* get field type */
194       _fieldkey   [_se][i]=tolower(s[17-1]);      /* get field key */
195       _fieldlen   [_se][i]=stoi(s,12,3);          /* get field lenght */
196       _fielddec   [_se][i]=stoi(s,15,2);          /* get field decimal */
197       _fieldoffset[_se][i]=ioffset;               /* get field offset */
198       ioffset=ioffset +_fieldlen[_se][i];         /* count field offset */
199       i++;
200     }
201     fclose(fpstc);
202     _sumfield[_se]=i;                             /* store total number field */
203     _statfp  [_se]=1;                             /* set fp status to 1 */
204
205     if (i==MAXFIELD) {                            /* if more than MAXFILED */
206       fprintf(stderr,"** Error: maximum field grater than %d\n",MAXFIELD);
207       exit(-1);
208     }
209     if ((ioffset+1)!=(int)ihead[2]) {             /* header is unmatch */
210       fprintf(stderr,"** Error: %s is not a database rec.len=%4ld total=%4d\n",
211         fn,ihead[2],ioffset+1);
212       exit(-1);
213     }
214     _delcol [_se]=ioffset -1;                     /* delete position */
215     _currec [_se]=1;                              /* set currect record to 1L */
216     _reclen [_se]=(int)ihead[2];                  /* store record lenght */
217     _trec   [_se]=(int)ihead[11];                 /* store total record */
218     _keylen [_se]=(int)ihead[15];                 /* store key lenght */
219     _keyoffs[_se]=(int)ihead[16];                 /* store key offset */
220     _lret   [_se]=1L;                             /* set currect record to 1L */
221   }
222
223   /*************************************************************
224   * use2(): This function is as sam as use, but fnstc2 is adder
```

```
225  ***************************************************************
226  */
227  use2(fn,mode,fnstc2)
228  char *fn,*mode,*fnstc2;
229  {
230    FILE *fpstc;
231    char s[200],fnstc[80],shead[513];
232    long ihead[1025];
233    int  i,ioffset;
234
235    if (_se<1 || _se>=MAXFILE)
236      error("** Error: use(fn,mode); never select area, select(n)"," ");
237
238    trim(fn);
239    if (strlen(fn)==NULL) {    /* select(" ","r") for close file */
240      if (_statfp[_se]==1) {
241        fcloseinx(_fpinx[_se]);
242        _statfp[_se]=0;
243        freemem();
244      }
245      return;
246    }
247    if (_statfp[_se]==1) {
248      fcloseinx(_fpinx[_se]);
249      freemem();
250    }
251    if (fopeninx(fn,mode,_fpinx[_se])==NULL)
252      error("** Error: use(fn,mode); fn not found:",fn);
253
254    _read_header(_fpinx[_se],ihead,shead);
255    _srec[_se]=(char *)malloc((int)ihead[2] +2);
256
257    strcpy(fnstc,fn);
258    strcat(fnstc,".stc");
259    if (!file(fnstc))                /* if not found fnstc, open fnstc2 */
260      strcpy(fnstc,fnstc2);          /* This statement is adding from use */
261    if ((fpstc=fopen(fnstc,"r"))==NULL)
262      error("** Error: use(fn,mode); fn not found:",fnstc);
263
264    i=0;
265    ioffset=1;
266    while (fgets(s,200,fpstc)!=NULL && i<MAXFIELD) {
267      if (s[0]=='#'||s[1]=='#')
268        break;
269      if (s[0]=='/'&&s[1]=='*')
270        continue;
271      substr(s,1,10,_fieldname[_se][i]);
272      trim(_fieldname[_se][i]);
273      _fieldtype  [_se][i]=tolower(s[11-1]);
274      _fieldkey   [_se][i]=tolower(s[17-1]);
275      _fieldlen   [_se][i]=stoi(s,12,3);
276      _fielddec   [_se][i]=stoi(s,15,2);
277      _fieldoffset[_se][i]=ioffset;
278      ioffset=ioffset +_fieldlen[_se][i];
279      i++;
280    }
```

```
281     fclose(fpstc);
282     _sumfield[_se]=i;
283     _statfp  [_se]=1;
284     if (i==MAXFIELD) {
285       fprintf(stderr,"** Error: maximum field grater than %d\n",MAXFIELD);
286       exit(-1);
287     }
288     if ((ioffset+1)!=(int)ihead[2]) {
289       fprintf(stderr,"** Error: %s is not a database rec.len=%4ld total=%4d\n",
290         fn,ihead[2],ioffset+1);
291       exit(-1);
292     }
293     _delcol [_se]=ioffset -1;
294     _currec [_se]=1;
295     _reclen [_se]=(int)ihead[2];
296     _trec   [_se]=(int)ihead[11];
297     _keylen [_se]=(int)ihead[15];
298     _keyoffs[_se]=(int)ihead[16];
299     _lret   [_se]=1L;
300   }
301
302   /*****************************
303   * mapchar()
304   *****************************
305   */
306   mapchar(strfield,ptrfield)
307   char  *strfield;                    /* pointer of field name */
308   char **ptrfield;                    /* pointer of pointer of field name */
309   {
310     int i,k,len,map=0;
311     char fen1[11],fen2[11];
312
313     if (_statfp[_se]==0)              /* if current area is not select, use */
314       error("** Error: mapchar() must be call function 'use' first"," ");
315
316     for (i=0;i<_sumfield[_se];i++) {                  /* search field name */
317       for (k=0;k<10 && _fieldname[_se][i][k]>' ';k++)
318         fen1[k]=tolower(_fieldname[_se][i][k]);       /* field name in file */
319       fen1[k]=NULL;
320       for (k=0;k<10 && strfield[k]>' ';k++)
321         fen2[k]=tolower(strfield[k]);                 /* field name parameter */
322       fen2[k]=NULL;
323       if (strcmp(fen1,fen2)==0) {                     /* is mapping */
324         if (_fieldtype[_se][i]=='c') {                /* check type is char? */
325           _fieldchar[_se][i]=ptrfield;
326           _fieldstat[_se][i]=1;
327           _fieldtmp [_se][i]=(char *)malloc(_fieldlen[_se][i] +1);
328           map=1;
329           break;
330         }
331       }
332     }
333     if (map==0)                  /* is not mapping, say error, stop */
334       error("** Error: field name not found, or type mismatch: ",strfield);
335   }
336
```

```
337  /*****************************
338  * mapreal()
339  *****************************
340  */
341  mapreal(strfield,ptrfield)
342  char   *strfield;
343  double *ptrfield;
344  {
345    int i,k,len,map=0;
346    char fen1[11],fen2[11];
347
348    if (_statfp[_se]==0)
349      error("** Error: mapreal() must be call function 'use' first"," ");
350
351    for (i=0;i<_sumfield[_se];i++) {
352      for (k=0;k<10 && _fieldname[_se][i][k]>' ';k++)
353        fen1[k]=tolower(_fieldname[_se][i][k]);
354      fen1[k]=NULL;
355      for (k=0;k<10 && strfield[k]>' ';k++)
356        fen2[k]=tolower(strfield[k]);
357      fen2[k]=NULL;
358      if (strcmp(fen1,fen2)==0) {
359        if (_fieldtype[_se][i]=='n') {       /* type must be real */
360          _fieldreal[_se][i]=ptrfield;
361          _fieldstat[_se][i]=1;
362          _fieldtmp [_se][i]=(char *)malloc(_fieldlen[_se][i] +1);
363          map=1;
364          break;
365        }
366      }
367    }
368    if (map==0)
369      error("** Error: field name not found, or type mismatch: ",strfield);
370  }
371
372  /*****************************
373  * mapall()
374  *****************************
375  */
376  mapall(ic,ptrfield)
377  int  ic;
378  char **ptrfield;
379  {
380    if (_statfp[_se]==0)
381      error("** Error: mapall() must be call function 'use' first"," ");
382
383    _fieldtype[_se][ic]='c';              /* set field type to char */
384    _fieldchar[_se][ic]=ptrfield;
385    _fieldstat[_se][ic]=1;
386    _fieldtmp [_se][ic]=(char *)malloc(_fieldlen[_se][ic] +1);
387  }
388
389  /*****************************
390  * mapseq()
391  *****************************
392  */
```

```
393  mapseq(strfield)
394  char  *strfield;
395  {
396    int i,k;
397    char fen1[11],fen2[11];
398
399    for (i=0;i<_sumfield[_se];i++) {
400      for (k=0;k<10 && _fieldname[_se][i][k]>' ';k++)
401        fen1[k]=tolower(_fieldname[_se][i][k]);
402      fen1[k]=NULL;
403      for (k=0;k<10 && strfield[k]>' ';k++)
404        fen2[k]=tolower(strfield[k]);
405      fen2[k]=NULL;
406      if (strcmp(fen1,fen2)==0)     /* if mapping field, return seq position */
407        return(i);
408    }
409    return(-1);                     /* for not found */
410  }
411
412  /*****************************
413  * getfieldname():
414  *    For get the field name
415  *****************************
416  */
417  getfieldname(tmp1,ic)
418  char *tmp1;
419  int  ic;
420  {
421    strcpy(tmp1,_fieldname[_se][ic]);    /* store field name to tmp buffer */
422  }
423
424  /*****************************
425  * getfieldtype()
426  *    For get the field type
427  *****************************
428  */
429  getfieldtype(tmp1,ic)
430  char *tmp1;
431  int  ic;
432  {
433    *tmp1=_fieldtype[_se][ic];
434  }
435
436  /*****************************
437  * getfieldkey()
438  *    For get the field key
439  *****************************
440  */
441  getfieldkey(tmp1,ic)
442  char *tmp1;
443  int  ic;
444  {
445    *tmp1=_fieldkey[_se][ic];
446  }
447
448  /*****************************
```

```
449  * getfieldlen()
450  *    For get the field length
451  ******************************
452  */
453  getfieldlen(ic)
454  int  ic;
455  {
456    return(_fieldlen[_se][ic]);
457  }
458
459  /******************************
460  * getfieldoffs()
461  *    For get the field offset
462  ******************************
463  */
464  getfieldoffs(ic)
465  int  ic;
466  {
467    return(_fieldoffset[_se][ic]);
468  }
469
470  /************************************
471  * getdelcol()
472  *    For get the colume that deleted
473  ************************************
474  */
475  getdelcol()
476  {
477    return(_delcol[_se]);
478  }
479
480  /**************************************
481  * gettfield()
482  *    For get the total number of field
483  **************************************
484  */
485  gettfield()
486  {
487    return(_sumfield[_se]);
488  }
489
490  /**************************************
491  * gettrec()
492  *    For get the total number of record
493  **************************************
494  */
495  gettrec()
496  {
497    return(_trec[_se]);
498  }
499
500  /********************************
501  * getkeyinfo()
502  *    For get the key information
503  ********************************
504  */
```

```
505  getkeyinfo(reclen,keylen,keyoffs)
506  int *reclen,*keylen,*keyoffs;
507  {
508    *reclen =_reclen [_se];
509    *keylen =_keylen [_se];
510    *keyoffs=_keyoffs[_se];
511  }
512
513  /*****************************
514  * skip()
515  *    For skip the data record
516  *****************************
517  */
518  skip(n)
519  int n;
520  {
521    char s[MAX1025];
522    long p_dat;
523
524    if (_statfp[_se]==0) {          /* current area not select */
525      fprintf(stderr,"** Error: file no %ld not use\n",_statfp[_se]);
526      exit(-1);
527    }
528    if (n!=1) {                     /* n not equal to 1 */
529      _lret[_se]=_lret[_se] +n;     /* compute new current record */
530      if (_lret[_se]<0L)            /* if less than first rec, set to 1 */
531        _lret[_se]=1L;
532      _currec[_se]=(int)_lret[_se];   /* new current record */
533      p_dat=(_lret[_se] -1L)* (long)_reclen[_se];  /* pointer of record */
534      rewind(_fpinx[_se][0]);
535      fseek(_fpinx[_se][0],p_dat,0);              /* seek data record */
536    }
537    if (fgets(_srec[_se],MAX1025,_fpinx[_se][0])!=NULL) {  /* get data record */
538      _currec[_se]=_currec[_se] +1;               /* store data to work area */
539      _lret[_se]=_lret[_se] +1;
540      _eofstat[_se]=0;
541    } else {                                      /* if eof() */
542      _eofstat[_se]=1;                            /* set status to 1 */
543      clean();                      /* clean the buffer work area */
544    }
545    getfield();                     /* store data in work area to field */
546  }
547
548  /*****************************
549  * seekdbf()
550  *    For seek the data record
551  *****************************
552  */
553  seekdbf(sk)
554  char *sk;
555  {
556    char s[MAX1025];
557
558    if (_statfp[_se]==0) {
559      fprintf(stderr,"** Error: file no %ld not use\n",_statfp[_se]);
560      exit(-1);
```

```
561     }
562     if ((_lret[_se]=freadinx(_srec[_se],sk,_fpinx[_se]))>0L) { /* if found */
563       _eofstat[_se]=0;                    /* set eof status to 0 */
564       _foundstat[_se]=1;                  /* set found status to 1 */
565       _currec[_se]=(int)_lret[_se];       /* new current record */
566     } else {                             /* if not found */
567       _eofstat[_se]=1;
568       _foundstat[_se]=0;
569       clean();
570     }
571     getfield();                    /* store data in work area to field */
572   }
573
574   /*****************************
575   * gotop()
576   *    For goto the top of file
577   *****************************
578   */
579   gotop()
580   {
581     go(1);
582   }
583
584   /*********************************
585   * gobuttom()
586   *    For goto the buttom of file
587   *********************************
588   */
589   gobuttom()
590   {
591     go(_trec[_se]);
592   }
593
594   /*****************************
595   * get_rec()
596   *    For goto get data record
597   *****************************
598   */
599   get_rec(shead,nr)
600   char *shead;
601   int nr;
602   {
603     go(nr);
604     strcpy(shead,_srec[_se]);
605   }
606
607   /*****************************
608   * go()
609   *    For goto the data record
610   *****************************
611   */
612   go(n)
613   int n;
614   {
615     long p_dat;
616
```

```
617     if (_statfp[_se]==0) {
618       fprintf(stderr,"** Error: file no %ld not use\n",_statfp[_se]);
619       exit(-1);
620     }
621     if (n<=0)
622       n=1;
623     if (n>_trec[_se])
624       n= _trec[_se];
625     if (n<=0)
626       n=1;
627     _currec[_se]=n;
628     _lret[_se]=(long)n;
629     p_dat=(_lret[_se] -1L)* (long)_reclen[_se];
630     rewind(_fpinx[_se][0]);
631     fseek(_fpinx[_se][0],p_dat,0);
632     if (fgets(_srec[_se],MAX1025,_fpinx[_se][0])!=NULL) {
633       _eofstat[_se]=0;
634     } else {
635       _eofstat[_se]=1;
636       clean();
637     }
638     getfield();
639   }
640
641   /*****************************
642   * getfield()
643   *    For get the field
644   *****************************
645   */
646   getfield()
647   {
648     int i;
649
650     for (i=0;i<_sumfield[_se];i++) {
651       if (_fieldstat[_se][i]==1) {
652         substr(_srec[_se],_fieldoffset[_se][i],_fieldlen[_se][i],
653           _fieldtmp[_se][i]);
654         if (_fieldtype[_se][i]=='c')     /* character type */
655           *_fieldchar [_se][i]= _fieldtmp[_se][i];   /* store data to variable*/
656         if (_fieldtype[_se][i]=='n')
657           *_fieldreal [_se][i]=atof(_fieldtmp[_se][i]);
658       }
659     }
660   }
661
662   /*********************************************
663   * replchar()
664   *    For replace character data to buffer
665   *********************************************
666   */
667   replchar(strfield,sx)
668   char *strfield;
669   char *sx;
670   {
671     int  i,k,j,ios;
672     char fen1[11],fen2[11];
```

```
673
674     for (i=0;i<_sumfield[_se];i++) {
675       for (k=0;k<10 && _fieldname[_se][i][k]>' ';k++)
676         fen1[k]=tolower(_fieldname[_se][i][k]);
677       fen1[k]=NULL;
678       for (k=0;k<10 && strfield[k]>' ';k++)
679         fen2[k]=tolower(strfield[k]);
680       fen2[k]=NULL;
681       if (strcmp(fen1,fen2)==0) {
682         if (_fieldstat[_se][i]==1) {
683           ios=_fieldoffset[_se][i] -1;
684           for (k=0;k<_fieldlen[_se][i] && (unsigned char)sx[k]>NULL;k++,ios++)
685             _srec[_se][ios]=sx[k];
686           for (j=k;j<_fieldlen[_se][i];j++,ios++)
687             _srec[_se][ios]=' ';
688           strcpy(_fieldtmp[_se][i],sx);
689           *_fieldchar [_se][i]= _fieldtmp[_se][i];
690         }
691         break;
692       }
693     }
694   }
695
696   /**********************************
697   * repreal0()
698   *    For replace real data to buffer
699   **********************************
700   */
701   replreal0(strfield,fx)
702   char *strfield;
703   double fx;
704   {
705     int  i,k,j,ios;
706     char fen1[11],fen2[11],stmp[50];
707
708     for (i=0;i<_sumfield[_se];i++) {
709       for (k=0;k<10 && _fieldname[_se][i][k]>' ';k++)
710         fen1[k]=tolower(_fieldname[_se][i][k]);
711       fen1[k]=NULL;
712       for (k=0;k<10 && strfield[k]>' ';k++)
713         fen2[k]=tolower(strfield[k]);
714       fen2[k]=NULL;
715       if (strcmp(fen1,fen2)==0) {
716         if (_fieldstat[_se][i]==1) {
717           sprintf(stmp,"%030.10f",fx);
718           if (_fielddec[_se][i]>0)
719             j=20-(_fieldlen[_se][i] - _fielddec[_se][i]);
720           else
721             j=20-(_fieldlen[_se][i] - _fielddec[_se][i]) -1;
722           ios=_fieldoffset[_se][i] -1;
723           for (k=0;k<_fieldlen[_se][i];k++,ios++,j++)
724             _srec[_se][ios]=stmp[j];
725           *_fieldreal [_se][i]=fx;
726         }
727         break;
728       }
```

```
729     }
730   }
731
732   /*****************************
733   * repreal()
734   *    For replace real data to buffer
735   *****************************
736   */
737   replreal(strfield,fx)
738   char *strfield;
739   double fx;
740   {
741     int  i,k,j,ios;
742     char fen1[11],fen2[11],stmp[50];
743
744     for (i=0;i<_sumfield[_se];i++) {
745       for (k=0;k<10 && _fieldname[_se][i][k]>' ';k++)
746         fen1[k]=tolower(_fieldname[_se][i][k]);
747       fen1[k]=NULL;
748       for (k=0;k<10 && strfield[k]>' ';k++)
749         fen2[k]=tolower(strfield[k]);
750       fen2[k]=NULL;
751       if (strcmp(fen1,fen2)==0) {
752         if (_fieldstat[_se][i]==1) {
753           sprintf(stmp,"%30.10f",fx);
754           if (_fielddec[_se][i]>0)
755             j=20-(_fieldlen[_se][i] - _fielddec[_se][i]);
756           else
757             j=20-(_fieldlen[_se][i] - _fielddec[_se][i]) -1;
758           ios=_fieldoffset[_se][i] -1;
759           for (k=0;k<_fieldlen[_se][i];k++,ios++,j++)
760             _srec[_se][ios]=stmp[j];
761           *_fieldreal [_se][i]=fx;
762         }
763         break;
764       }
765     }
766   }
767
768   /*****************************
769   * replace()
770   *    For replace buffer to file
771   *****************************
772   */
773   replace()
774   {
775     if (_statfp[_se]==0) {
776       fprintf(stderr,"** Error: replace() file no %ld not use\n",_statfp[_se]);
777       exit(-1);
778     }
779     if (_lret[_se]>0L) {
780       frewriteinx(_srec[_se],_lret[_se],_fpinx[_se]);
781   /*
782       go(_currec[_se]);
783   */
784     }
```

```
785  }
786
787  /*****************************
788  * delete()
789  *    For delete the data record
790  *****************************
791  */
792  delete()
793  {
794    if (_statfp[_se]==0) {
795      fprintf(stderr,"** Error: delete() file no %ld not use\n",_statfp[_se]);
796      exit(-1);
797    }
798    if (_lret[_se]>0L) {
799      _srec[_se][_delcol[_se]]='D';
800      frewriteinx(_srec[_se],_lret[_se],_fpinx[_se]);
801    }
802  }
803
804  /*****************************
805  * recall()
806  *    For recall the data record
807  *****************************
808  */
809  recall()
810  {
811    if (_statfp[_se]==0) {
812      fprintf(stderr,"** Error: recall() file no %ld not use\n",_statfp[_se]);
813      exit(-1);
814    }
815    if (isdelete()) {
816      _srec[_se][_delcol[_se]]=' ';
817      frewriteinx(_srec[_se],_lret[_se],_fpinx[_se]);
818    }
819  }
820
821  /*****************************
822  * isdelete()
823  *    Check if delete?
824  *****************************
825  */
826  isdelete()
827  {
828    if (_srec[_se][_delcol[_se]]=='D')
829      return(1);
830    else
831      return(0);
832  }
833
834  /*****************************
835  * append()
836  *    For append the data file
837  *****************************
838  */
839  append()
840  {
```

```
841     char s[MAX1025];
842
843     if (_statfp[_se]==0) {
844       fprintf(stderr,"** Error: append() file no %ld not use\n",_statfp[_se]);
845       exit(-1);
846     }
847     if (freadinx(s,_srec[_se],_fpinx[_se])==0L) {
848       fwriteinx(_srec[_se],_fpinx[_se],"a");
849       strcpy(s,_srec[_se]);
850       seekdbf(s);
851     }
852   }
853
854   /*******************************
855   * found()
856   *    Check if found?
857   *******************************
858   */
859   found()
860   {
861     return(_foundstat[_se]);
862   }
863
864   /*******************************
865   * eof()
866   *    Check if end-of-file?
867   *******************************
868   */
869   eof()
870   {
871     return(_eofstat[_se]);
872   }
873
874   /*******************************
875   * bof()
876   *    Check if begin-of-file?
877   *******************************
878   */
879   bof()
880   {
881     if (_currec[_se]==1)
882       return(1);
883     else
884       return(0);
885   }
886
887   /*******************************
888   * recno()
889   *    Ark the record number
890   *******************************
891   */
892   recno()
893   {
894     return(_currec[_se]);
895   }
896
```

```
897  /*****************************
898  * flockdbf()
899  *    For lock the data file
900  *****************************
901  */
902  flockdbf()
903  {
904    flockinx(_fpinx[_se][0]);
905  }
906
907  /*****************************
908  * unflockdbf()
909  *    For unlock the data file
910  *****************************
911  */
912  unlockdbf()
913  {
914    funlockinx(_fpinx[_se][0]);
915  }
916
917  /*****************************
918  * closedata()
919  *    For close all data file
920  *****************************
921  */
922  closedata()
923  {
924    int i;
925
926    for (i=0;i<MAXFILE;i++) {
927      if (_statfp[i]==1) {
928        fcloseinx(_fpinx[i]);
929        _statfp[i]=0;
930      }
931    }
932  }
933
934  /*****************************
935  * isdbf()
936  *   Check if PSUbase file
937  *****************************
938  */
939  isdbf(fn)
940  char *fn;
941  {
942    char fntmp[100];
943
944    if (!file(fn))
945      return(0);
946    sprintf(fntmp,"%s.inx",fn);
947    if (!file(fntmp))
948      return(0);
949    sprintf(fntmp,"%s.stc",fn);
950    if (!file(fntmp))
951      return(0);
952
```

```
  return(1);
}

/*****************************
* file()
*   Check if found a file
*****************************
*/
file(fn)
char *fn;
{
  FILE *fps;

  if ((fps=fopen(fn,"r"))==NULL)
    return(0);
  else
    fclose(fps);
  return(1);
}

/*****************************
* clean()
*   Clean a buffer record
*****************************
*/
clean()
{
  int i;

  for (i=0;i<_reclen[_se];i++)
    _srec[_se][i]=' ';
}

/*****************************
* freemem()
*   Free the allocate memory
*****************************
*/
freemem()
{
  int i;

  free(_srec[_se]);
  for (i=0;i<_sumfield[_se];i++) {
    if ((int)_fieldtmp[_se][i]>0)
      free(_fieldtmp [_se][i]);
  }
}
:::::::::::::::
ut_db2.c
:::::::::::::::
/*****************************************************************************
*
* ut_db2.c: This is a utility file #2 for the PSUbase
*           Written by Mongkol kuanhavet, 18 Jun 92
*           Computer center, Prince of Songkla Univ., Hatyai, Thailand.
```

```
1009  *
1010  * Functions:
1011  *     initscr()       Initial a screen to raw mode, nonecho
1012  *     endscr()        Return to original screen mode
1013  *     eject()         Eject or print Control_L
1014  *     clear()         Clear a screen
1015  *     beep()          Say the beep sound
1016  *     getnc()         Get n number of character
1017  *     getyn()         Get a character 'y' or 'n'
1018  *     text()          Print a text
1019  *     say()           Say one line message
1020  *     goyx()          Goto y-x axis
1021  *     stolower()      Change all string to lower
1022  *     arespace()      Check if space
1023  *     filetype()      Check a file type
1024  *     space()         Say a space
1025  *     date()          Print cuurent date
1026  *     time()          Print current time
1027  ********************************************************************************
1028  */
1029  #include <stdio.h>
1030  #include <ctype.h>
1031  #include <signal.h>
1032  #include <string.h>
1033  #include <malloc.h>
1034  #include "cst_db.h"
1035
1036  double atof();                    /* define double for atof() */
1037
1038  initscr()
1039  {
1040    system("stty raw -echo");
1041  }
1042
1043  endscr()
1044  {
1045    system("stty -raw echo pass8");
1046    fflush(stdout);
1047  }
1048
1049  eject()
1050  {
1051    printf("%c\n",12);
1052  }
1053
1054  clear()
1055  {
1056    printf("2J");
1057    goyx(0,0);
1058  }
1059
1060  beep()
1061  {
1062    putc(BEEP,stdout);
1063  }
1064
```

```
1065  getnc(nc,cc,sb)
1066  int *nc;
1067  char *cc,*sb;
1068  {
1069    int  j,k=0,len;
1070    char sc[30];
1071
1072    initscr();
1073    len=strlen(sb);
1074    while (1) {
1075      sc[k]=getc(stdin);
1076      for (j=0; j<len; j++) {
1077        if (sc[k]==sb[j]) {
1078          *cc=sc[k];
1079          sc[k]=NULL;
1080          *nc=(k==0)?1:atoi(sc);
1081          endscr();
1082          return;
1083        }
1084      }
1085      if (isdigit(sc[k])&&k<29)
1086        k++;
1087      else if (sc[k]==BSPACE)
1088        k=(k>0)?k-1:k;
1089      else
1090        beep();
1091    }
1092  }
1093
1094  getyn()
1095  {
1096    int  ic;
1097
1098    initscr();
1099    while (1) {
1100      ic=getc(stdin);
1101      putc(ic,stdout);
1102      fflush(stdout);
1103      if (ic=='Y'||ic=='y'||ic=='N'||ic=='n') {
1104        endscr();
1105        return(ic);
1106      }
1107      beep();
1108    }
1109  }
1110
1111  text(s)
1112  char *s;
1113  {
1114    printf("%s",s);
1115  }
1116
1117  say(y,x,s)
1118  char  y,x,*s;
1119  {
1120    printf("%d;%dH%s",y,x,s);
```

```
1121    fflush(stdout);
1122  }
1123
1124  goyx(y,x)
1125  char  y,x;
1126  {
1127    printf("%d;%dH",y,x);
1128  }
1129
1130  stolower(s)
1131  char *s;
1132  {
1133    int i;
1134
1135    for (i=0;s[i]!=NULL;i++)
1136      s[i]=tolower(s[i]);
1137  }
1138
1139  arespace(s)
1140  char *s;
1141  {
1142    int i;
1143
1144    for (i=0;s[i]!=NULL;i++)
1145      if (s[i]>' ')
1146        return(0);
1147    return(1);
1148  }
1149
1150  filetype(s)
1151  char *s;
1152  {
1153    FILE *fp;
1154    char buf[512],comm[100],fn[100];
1155
1156    if (!file(s))
1157      return(0);
1158    sprintf(fn,"/tmp/lstype#%06d",s,getpid());
1159    sprintf(comm,"ls -l %s > %s",s,fn);
1160    system(comm);
1161    fp=fopen(fn,"r");
1162    fgets(buf,512,fp);
1163    fclose(fp);
1164    sprintf(comm,"rm %s",fn);
1165    system(comm);
1166    return(buf[0]);
1167  }
1168
1169  char *space(n)
1170  int n;
1171  {
1172    int i;
1173    char tmp[1024];
1174    static char *_pspace[1024];
1175
1176    if (n<1||n>=1024)
```

```
1177        return((char *)NULL);
1178
1179      if (_pspace[n]==0) {
1180        _pspace[n]=(char *)malloc(n+1);
1181        if ((int)_pspace[n]>0) {
1182          for(i=0;i<n;i++)
1183            tmp[i]=' ';
1184          tmp[i]=NULL;
1185          strcpy(_pspace[n],tmp);
1186          return((char *)_pspace[n]);
1187        } else {
1188          return((char *)NULL);
1189        }
1190      } else {
1191        return((char *)_pspace[n]);
1192      }
1193    }
1194
1195    char *repl(s,n)
1196    char *s;
1197    int n;
1198    {
1199      int i,len=0,size=0;
1200      char tmp[4096];
1201      static int _maxrepl;
1202      static char *_prepl[1024];
1203
1204      if (_maxrepl>=1024) {
1205        error("** Error: from repl(s,n) maxrepl>1024"," ");
1206      }
1207      tmp[0]=NULL;
1208      len=strlen(s);
1209      for(i=0;i<n&&len<4096;i++) {
1210        strcat(tmp,s);
1211        size=size+len;
1212      }
1213      for (i=0;i<_maxrepl;i++) {
1214        if (strcmp(tmp,_prepl[i])==0)
1215          return((char *)_prepl[i]);
1216      }
1217      if (i<1024) {
1218        _prepl[i]=(char *)malloc(size+1);
1219        strcpy(_prepl[i],tmp);
1220        _maxrepl++;
1221        return((char *)_prepl[i]);
1222      }
1223    }
1224
1225    char *date()
1226    {
1227      char s[7];
1228      static char *_pdate;
1229
1230      if (_pdate==0) {
1231        _pdate=(char *)malloc(9);
1232        get_date_time(s);
```

```
1233        sprintf(_pdate,"%02d/%02d/%02d",s[0],s[1],s[2]);
1234      }
1235      return((char *)_pdate);
1236    }
1237
1238    char *time()
1239    {
1240      char s[7];
1241      static char *_ptime;
1242
1243      if (_ptime==0)
1244        _ptime=(char *)malloc(9);
1245      get_date_time(s);
1246      sprintf(_ptime,"%02d:%02d:%02d",s[3],s[4],s[5]);
1247      return((char *)_ptime);
1248    }
1249    ::::::::::::::
1250    ut_db3.c
1251    ::::::::::::::
1252    #include <stdio.h>
1253    #include <ctype.h>
1254    #include <math.h>
1255    #include <malloc.h>
1256    #include "cst_db.h"
1257
1258    double atof();                    /* define double for atof() */
1259    static int     _isay;
1260    static char    _deftype[MAXPULL];
1261    static char   *_defname[MAXPULL];
1262    static char   *_defmem[MAXPULL];
1263    static char  **_adsc[MAXPULL];
1264    static double *_adsr[MAXPULL];
1265    static int     _nf;
1266    static int     _ay[MAXPULL];
1267    static int     _ax[MAXPULL];
1268    static int     _varlen[MAXPULL];
1269    static char    _tokg[4096];
1270    static char    _vartype[MAXPULL];
1271    static char    _vardec[MAXPULL];
1272    static char    _sayname[MAXPULL][50];
1273    static char    _varname[MAXPULL][20];
1274    static char    _vardata[MAXPULL][80];
1275    static char    _varpic[MAXPULL][80];
1276    static char    _varrang1[MAXPULL][20];
1277    static char    _varrang2[MAXPULL][20];
1278    static double  _cstrang1[MAXPULL];
1279    static double  _cstrang2[MAXPULL];
1280    static char    _varvalid[MAXPULL][80];
1281    /* var. for set status
1282    */
1283    static char    _bell;
1284    static char    _inson;
1285    static char    _confirm;
1286    static char    _noecho;
1287
1288    defchar(sx,ax)
```

```
1289  char *sx;
1290  char **ax;
1291  {
1292    int i;
1293
1294    for (i=0;i<_isay;i++) {
1295      if (strcmp(_defname[i],sx)==0)
1296        return(0);
1297    }
1298    if (_isay>=0 && _isay<MAXPULL) {
1299      _defname[_isay]=(char *)malloc(20);
1300      _defmem[_isay]=(char *)malloc(80);
1301      _defmem[_isay][0]=NULL;
1302      strcpy(_defname[_isay],sx);
1303      _adsc[_isay]=ax;
1304      _deftype[_isay]='c';
1305      _isay++;
1306    }
1307  }
1308
1309  defreal(sx,ax)
1310  char  *sx;
1311  double *ax;
1312  {
1313    int i;
1314
1315    for (i=0;i<_isay;i++) {
1316      if (strcmp(_defname[i],sx)==0)
1317        return(0);
1318    }
1319    if (_isay>=0 && _isay<MAXPULL) {
1320      _defname[_isay]=(char *)malloc(20);
1321      _defmem[_isay]=(char *)malloc(80);
1322      _defmem[_isay][0]=NULL;
1323      strcpy(_defname[_isay],sx);
1324      _adsr[_isay]=ax;
1325      _deftype[_isay]='n';
1326      _isay++;
1327    }
1328  }
1329
1330  genfield(s1,sp,nf)
1331  char *s1,*sp;
1332  int nf;
1333  {
1334    int i,k,j,cf=0;
1335
1336    for (i=0;sp[i]!=NULL;i++) {
1337      if (sp[i]=='@')
1338        cf++;
1339      if (cf>nf)
1340        break;
1341    }
1342    if (sp[i]==NULL)
1343      return(0);
1344
```

```
1345      for (k=i+1,j=0;sp[k]!=NULL;k++,j++) {
1346        if (sp[k]=='@')
1347          break;
1348        if (j>=511) {
1349          sl[j]=NULL;
1350          error("** Error: tokgen>512 ",sl);
1351        }
1352        sl[j]=sp[k];
1353      }
1354      sl[j]=NULL;
1355
1356      return(1);
1357    }
1358
1359    genyx(yy,xx,sl)
1360    int *yy,*xx;
1361    char *sl;
1362    {
1363      int i,k,j;
1364      char s2[200];
1365
1366      for (i=0;sl[i]!=NULL;i++) {
1367        if (sl[i]==',')
1368          break;
1369        s2[i]=sl[i];
1370      }
1371      s2[i]=NULL;
1372      *yy=atoi(s2);
1373
1374      for (k=i+1;sl[k]!=NULL;k++) {
1375        if (sl[k]!=' ')
1376          break;
1377      }
1378      for (j=0,i=k;sl[i]!=NULL;i++,j++) {
1379        if (sl[i]==' ')
1380          break;
1381        s2[j]=sl[i];
1382      }
1383      s2[j]=NULL;
1384      *xx=atoi(s2);
1385    }
1386
1387    gensh(sh,sl)
1388    char *sh,*sl;
1389    {
1390      int i,k,j;
1391
1392      sh[0]=NULL;
1393      for (i=1;sl[i]!=NULL;i++) {
1394        if (sl[i-1]==' '&&sl[i]=='s'&&sl[i+1]=='a'&&sl[i+2]=='y'&&sl[i+3]==' ')
1395          break;
1396      }
1397      if (sl[i]==NULL)
1398        return;
1399      for (k=i+3;sl[k]!=NULL;k++) {
1400        if (sl[k]=='\'')
```

```
1401          break;
1402      }
1403      for (j=0,i=k+1;j<=49 && sl[i]!=NULL;i++,j++) {
1404        if (sl[i]=='\'')
1405          break;
1406        sh[j]=sl[i];
1407      }
1408      sh[j]=NULL;
1409    }
1410
1411    genvar(sx,sl)
1412    char *sx,*sl;
1413    {
1414      int i,k,j;
1415
1416      sx[0]=NULL;
1417      for (i=1;sl[i]!=NULL;i++) {
1418        if (sl[i-1]==' '&&sl[i]=='g'&&sl[i+1]=='e'&&sl[i+2]=='t'&&sl[i+3]==' ')
1419          break;
1420      }
1421      if (sl[i]==NULL)
1422        return;
1423      for (k=i+3;sl[k]!=NULL;k++) {
1424        if (sl[k]!=' ')
1425          break;
1426      }
1427      if (sl[k]==NULL)
1428        return;
1429      for (j=0,i=k;j<=19 && sl[i]!=NULL;i++,j++) {
1430        if (sl[i]==' ')
1431          break;
1432        sx[j]=sl[i];
1433      }
1434      sx[j]=NULL;
1435    }
1436
1437    genrang(sx1,sx2,cx1,cx2,sl)
1438    char *sx1,*sx2;
1439    char *sl;
1440    double *cx1,*cx2;
1441    {
1442      int i,k,j;
1443
1444      sx1[0]=sx2[0]=NULL;
1445      *cx1=0;
1446      *cx2=0;
1447      for (i=1;sl[i]!=NULL;i++) {
1448        if (sl[i-1]==' '&&sl[i]=='r'&&sl[i+1]=='a'&&sl[i+2]=='n'
1449          &&sl[i+3]=='g'&&(sl[i+4]==' '||(sl[i+4]=='e'&&sl[i+5]==' '))) {
1450          if (sl[i+4]=='e')
1451            sl[i+4]=' ';
1452          break;
1453        }
1454      }
1455      if (sl[i]==NULL)
1456        return;
```

```
1457    for (k=i+4;sl[k]!=NULL;k++) {
1458      if (sl[k]!=' ')
1459        break;
1460    }
1461    if (sl[k]==NULL)
1462      return;
1463    for (j=0,i=k;j<=19 && sl[i]!=NULL;i++,j++) {
1464      if (sl[i]==',')
1465        break;
1466      sx1[j]=sl[i];
1467    }
1468    sx1[j]=NULL;
1469    for (k=i+1;sl[k]!=NULL;k++) {
1470      if (sl[k]!=' ')
1471        break;
1472    }
1473    for (j=0,i=k;j<=19 && sl[i]!=NULL;i++,j++) {
1474      if (sl[i]==' ')
1475        break;
1476      sx2[j]=sl[i];
1477    }
1478    sx2[j]=NULL;
1479    *cx1=atof(sx1);
1480    *cx2=atof(sx2);
1481  }
1482
1483  genvalid(sx1,sl)
1484  char *sx1;
1485  char *sl;
1486  {
1487    int i,k,j;
1488
1489    sx1[0]=NULL;
1490    for (i=1;sl[i]!=NULL;i++) {
1491      if (sl[i-1]==' '&&sl[i]=='v'&&sl[i+1]=='a'&&sl[i+2]=='l'
1492        &&sl[i+3]=='i'&&sl[i+4]=='d'&&sl[i+5]==' ')
1493        break;
1494    }
1495    if (sl[i]==NULL)
1496      return;
1497    for (k=i+5;sl[k]!=NULL;k++) {
1498      if (sl[k]=='\'')
1499        break;
1500    }
1501    if (sl[k]==NULL)
1502      return;
1503    for (j=0,i=k+1;j<=79 && sl[i]!=NULL;i++,j++) {
1504      if (sl[i]=='\'')
1505        break;
1506      sx1[j]=sl[i];
1507    }
1508    sx1[j]=NULL;
1509  }
1510
1511  genlen(sl,dec,spic,sl)
1512  int *sl;
```

```
1513  char *sl,*spic,*dec;
1514  {
1515    int i,k,j;
1516    char cdec=0,idec=0;
1517
1518    spic[0]=NULL;
1519    *sl=0;
1520    *dec=0;
1521    for (i=1;sl[i]!=NULL;i++) {
1522      if (sl[i-1]==' '&&sl[i]=='p'&&sl[i+1]=='i'&&sl[i+2]=='c'&&
1523          sl[i+3]=='t'&&sl[i+4]==' ')
1524        break;
1525    }
1526    if (sl[i]==NULL)
1527      return;
1528    for (k=i+4;sl[k]!=NULL;k++) {
1529      if (sl[k]=='\'')
1530        break;
1531    }
1532    if (sl[k]==NULL)
1533      return;
1534    for (j=0,i=k+1;sl[i]!=NULL;i++,j++) {
1535      if (sl[i]=='\'' || j>=79)
1536        break;
1537      if (cdec==1 && idec<=9)
1538        idec++;
1539      if (sl[i]=='.')
1540        cdec=1;
1541      spic[j]=sl[i];
1542    }
1543    spic[j]=NULL;
1544    *sl=j;
1545    *dec=idec;
1546  }
1547
1548  sayget(sp)
1549  char *sp;
1550  {
1551    int  i,ret;
1552    char sl[512],chg;
1553
1554    if (strcmp(_tokg,sp)!=0) {
1555      if (strlen(sp)>=4096)
1556        error("** Error: Tokgen is long (max 4096)",sp);
1557      _nf=0;
1558      while (1) {
1559        if (genfield(sl,sp,_nf)==0)
1560          break;
1561        genyx(&_ay[_nf],&_ax[_nf],sl);
1562        gensh(_sayname[_nf],sl);
1563        genvar(_varname[_nf],sl);
1564        genlen(&_varlen[_nf],&_vardec[_nf],_varpic[_nf],sl);
1565       genrang(_varrang1[_nf],_varrang2[_nf],&_cstrang1[_nf],&_cstrang2[_nf],sl);
1566        genvalid(_varvalid[_nf],sl);
1567        if (_vardec[_nf]>0)
1568          _vartype[_nf]='F';
```

```
1569          else if (_varpic[_nf][0]=='!')
1570            _vartype[_nf]='U';
1571          else if (_varpic[_nf][0]=='9')
1572            _vartype[_nf]='N';
1573          else
1574            _vartype[_nf]='A';
1575          if (_varname[_nf][0]!=NULL) {
1576            if (chkvar(_varname[_nf])==0)
1577              error("** Error: not mapping var ",_varname[_nf]);
1578          }
1579          _nf++;
1580        }
1581        strcpy(_tokg,sp);
1582        chg=1;
1583      } else {
1584        chg=0;
1585      }
1586      for (i=0;i<_nf;i++) {
1587        if (_sayname[i][0]!=NULL && chg==1) {
1588          say(_ay[i],_ax[i],_sayname[i]);
1589        }
1590        if (_varname[i][0]!=NULL) {
1591          getads(_vardata[i],_varlen[i],_vardec[i],_varname[i]);
1592          say(_ay[i],_ax[i]+strlen(_sayname[i]),_vardata[i]);
1593        }
1594      }
1595      i=0;
1596      while (i<_nf && _varname[i][0]==NULL)
1597        i++;
1598      while (i>=0 && i<_nf) {
1599        ret=getstrdb3(_vardata[i],_ay[i],_ax[i]+strlen(_sayname[i]),
1600          _varlen[i],_vartype[i],_vardec[i],_varpic[i]);
1601        storeads(_vardata[i],_varname[i]);
1602        if (_vartype[i]=='F' && _varrang1[i][0]!=NULL) {
1603          if (chkrang(_vardata[i],i)==0) {
1604            beep();
1605            continue;
1606          }
1607        }
1608        if (_vartype[i]!='F' && _varvalid[i][0]!=NULL) {
1609          if (chkvalid(_vardata[i],i)==0) {
1610            beep();
1611            continue;
1612          }
1613        }
1614        if (ret==UP) {
1615          i--;
1616          while (i>=0 && _varname[i][0]==NULL)
1617            i--;
1618        } else {
1619          i++;
1620          while (i<_nf && _varname[i][0]==NULL)
1621            i++;
1622        }
1623        if (ret==CTRL_K||ret==CTRL_C||ret==CTRL_P)
1624          break;
```

```
1625      }
1626      return(ret);
1627    }
1628
1629    chkvar(sx)
1630    char *sx;
1631    {
1632      int i;
1633
1634      for (i=0;i<_isay;i++) {
1635        if (strcmp(_defname[i],sx)==0)
1636          return(1);
1637      }
1638      return(0);
1639    }
1640
1641    storeads(sd,sx)
1642    char *sd,*sx;
1643    {
1644      int i;
1645
1646      for (i=0;i<_isay;i++) {
1647        if (strcmp(_defname[i],sx)==0) {
1648          strcpy(_defmem[i],sd);
1649          if (_deftype[i]=='c')
1650            *_adsc[i]=_defmem[i];
1651          else
1652            *_adsr[i]=atof(_defmem[i]);
1653          break;
1654        }
1655      }
1656    }
1657
1658    chkrang(sd,ix)
1659    char *sd;
1660    int  ix;
1661    {
1662      int i;
1663      double dx;
1664
1665      if (!isdigit(_varrang1[ix][0])) {
1666        for (i=0;i<_isay;i++) {
1667          if (strcmp(_defname[i],_varrang1[ix])==0) {
1668            _cstrang1[ix]= *_adsr[i];
1669            break;
1670          }
1671        }
1672        if (i== _isay)
1673          error("** not define ",_varrang1[ix]);
1674      }
1675      if (!isdigit(_varrang2[ix][0])) {
1676        for (i=0;i<_isay;i++) {
1677          if (strcmp(_defname[i],_varrang2[ix])==0) {
1678            _cstrang2[ix]= *_adsr[i];
1679            break;
1680          }
```

```
1681        }
1682        if (i== _isay)
1683          error("** not define ",_varrang1[ix]);
1684      }
1685      dx=atof(sd);
1686      if (dx>=_cstrang1[ix] & dx<=_cstrang2[ix])
1687        return(1);
1688      else
1689        return(0);
1690    }
1691
1692    chkvalid(sd,ix)
1693    char *sd;
1694    int  ix;
1695    {
1696      int i,len;
1697      char s[80];
1698
1699      len=strlen(_varvalid[ix]);
1700      for (i=1;i<=len;i=i+_varlen[ix]) {
1701        substr(_varvalid[ix],i,_varlen[ix],s);
1702        if (strcmp(sd,s)==0)
1703          return(1);
1704      }
1705      return(0);
1706    }
1707
1708    getads(sd,len,dec,sx)
1709    char *sd,*sx,dec;
1710    int len;
1711    {
1712      int i,k,j;
1713      char stmp[80],*pc;
1714      double rx;
1715
1716      for (i=0;i<_isay;i++) {
1717        if (strcmp(_defname[i],sx)==0) {
1718          if (_deftype[i]=='c') {
1719            pc= *_adsc[i];
1720            strcpy(sd,pc);
1721            for (j=strlen(sd);j<len;j++)
1722              sd[j]=' ';
1723            sd[len]=NULL;
1724          } else {
1725            rx= *_adsr[i];
1726            sprintf(stmp,"%30.10f",(double)rx);
1727            j=20-(len - dec);
1728            j=(dec>0)?j:j-1;
1729            for (k=0;k<len;k++,j++)
1730              sd[k]=stmp[j];
1731            sd[k]=NULL;
1732          }
1733          break;
1734        }
1735      }
1736    }
```

```
1737
1738  getstrdb3(t,y,x,n,ty,dec,spic)
1739  char t[],x,y,ty,dec,*spic;
1740  char n;
1741  {
1742    int i,j,c,c1,c2,c3,x1,count,len,px,py;
1743    char thai,change,fntmp[50];
1744    char s[128], sp[128],endkey[128],comm[100];
1745
1746    i=c1=c2=c3=thai=count=change=0;
1747    x1=x;
1748                              /* dispose ctrl characters and count */
1749                              /* CTRL_A & CTRL_N in buffer t */
1750    strcpy(s,t);
1751
1752    while (1) {
1753      printf("%d;%dH",y,x);
1754
1755      c = getchmenu();
1756
1757      switch(c) {
1758        case 0x0d:  /* carriage return */
1759          break;
1760
1761        case 0x09:  /* insert switch mode */
1762          _inson = !_inson;
1763          if (_inson)
1764            printf("24;76H7mINS0m");
1765          else
1766            printf("24;76H   ");
1767          break;
1768
1769        case 0x08:  /* backspace */
1770        case 0x7f:  /* backspace */
1771          if (x > x1) {
1772            i--, x--;
1773            shift_d3_lf(s,i,1,n);
1774            if (_noecho!=1)
1775              printf("%d;%dH%s",y,x1,s);
1776          }
1777          change=1;
1778          break;
1779
1780        case 0x85:  /* shift left with arrow <== */
1781        case 0x86:  /* shift left with arrow <== */
1782          if (x > x1) {
1783            i--, x--;
1784          }
1785          break;
1786
1787        case 0x84:  /* shift right with arrow ==> */
1788          if (x < x1+n-count) {
1789            i++, x++;
1790          }
1791          break;
1792
```

```
1793        case 0x19:  /* CTRL_Y Delete field */
1794          for (i=0;i<n;i++)
1795            s[i]=' ';
1796          s[i]=NULL;
1797          i=c1=c2=c3=thai=count=change=0;
1798          x=x1;
1799          if (_noecho!=1)
1800            printf("%d;%dH%s",y,x1,s);
1801          break;
1802
1803        case 0x14:  /* CTRL_T Delete word right */
1804          for (j=i;j<n;j++) {
1805            if (s[j]==1 || s[j]==14)
1806              count--;
1807            s[j]=' ';
1808          }
1809          s[j]=NULL;
1810          if (_noecho!=1)
1811            printf("%d;%dH%s",y,x1,s);
1812          break;
1813
1814        default:  /* any characters between 0x20 to 0x7e */
1815          if (!(c==0x0d || c==0x82 || c==0x83 || c==CTRL_K || c==CTRL_P ||
1816                c==CTRL_R || c==CTRL_C || c==CTRL_E || c==CTRL_X)) {
1817            if (spic[i]=='!')
1818              c=toupper(c);
1819            if (spic[i]=='9' || spic[i]=='.') {
1820              if (!(isdigit(c)||c=='.'||c=='-'||c=='+')) {
1821                beep();
1822                continue;
1823              }
1824            }
1825          }
1826          if (i < n && (unsigned char)c > 0x1f && c!=0x82 && c!=0x83) {
1827            if (_inson) {
1828              count -= shift_d3_rt(s,i,1,n);
1829              s[i] = c;
1830              if (_noecho!=1)
1831                printf("%d;%dH%s",y,x1,s);
1832            } else {
1833              if (_noecho!=1)
1834                putc(c,stdout);
1835              s[i] = c;
1836            }
1837            i++, x++;
1838            change=1;
1839          }
1840          if (i>=n && _bell!=0)
1841            beep();
1842          if (i>=n && _confirm==0)
1843            c=0x0d;
1844          break;
1845      }
1846      if (c==0x0d || c==0x82 || c==0x83 || c==CTRL_K || c==CTRL_P ||
1847          c==CTRL_R || c==CTRL_C || c==CTRL_E || c==CTRL_X) {
1848        switch(c) {
```

```
1849            case 0x0d: c=RETURN; break;
1850            case 0x82: c=UP; break;
1851            case 0x83: c=DOWN; break;
1852            case 0x19: c=RETURN; break;
1853          }
1854          break;
1855        }
1856        c1 = c;
1857      }
1858      strcpy(t,s);
1859      if (ty=='N'||ty=='M'||ty=='S'||ty=='B') { /* check numeric */
1860        if (change) {
1861          t[i]=NULL;
1862          rigthadj2(n,s,t,ty,dec);
1863          strcpy(t,s);
1864        }
1865      } else if (ty=='F') { /* check numeric */
1866        if (change) {
1867          t[i]=NULL;
1868          realtostr(n,s,t,ty,dec);
1869          strcpy(t,s);
1870        }
1871      } else if (ty=='U' || ty=='L') { /* check upper, lower case */
1872        if (change) {
1873          upperlow2(s,t,ty);
1874          strcpy(t,s);
1875        }
1876      }
1877      fflush(stdout);
1878      return(c);
1879    }
1880
1881    getchdb3()   /* read character in raw mode */
1882    {
1883      int  c;
1884
1885      c = getc(stdin);
1886      if (c == 0x1b || c == 0x9b) {
1887        c = getc(stdin);
1888        if (c == 0x5b || c == 0x3a) {
1889          c = getc(stdin);
1890          switch(c) {
1891            case 0x41: c = 0x82; break; /* Eng up */
1892            case 0x42: c = 0x83; break; /* Eng down */
1893            case 0x43: c = 0x84; break; /* Eng right */
1894            case 0x44: c = 0x85; break; /* Eng left */
1895            case 0x29: c = 0x84; break; /* Tha right */
1896            case 0x2f: c = 0x86; break; /* Tha left */
1897            default:   c = 0x83; break; /* Other is down */
1898          }
1899        }
1900      }
1901      return(c);
1902    }
1903
1904    shift_d3_lf(s,i,n1,n2)
```

```
1905  char *s;      /* array */
1906  int i,n1,n2;  /* i position, n1 # shift, n2 total string of s */
1907  {
1908    int j;
1909
1910    for (j=i; j<n2-n1; j++)
1911      s[j] = s[j+n1];
1912
1913    for (j=n1; j>0; j--)
1914      s[n2-j] = 0x20;
1915  }
1916
1917  shift_d3_rt(s,i,n1,n2)
1918  char *s;      /* array */
1919  int i,n1,n2;  /* i position, n1 # shift, n2 total string of s */
1920  {
1921    int j, count=0;
1922
1923    for (j=n2-1; j>i+n1-1; j--)
1924      s[j] = s[j-n1];
1925
1926    return(count);
1927  }
1928
1929  replic2(sp,c,count)
1930  char *sp,c;
1931  int count;
1932  {
1933    int i=0;
1934
1935    while(i < count)
1936       sp[i++] = c;
1937
1938    sp[i] = NULL;
1939  }
1940
1941  upperlow2(so,si,ty)
1942  char  ty, so[], si[];
1943  {
1944    int i,len;
1945
1946    len = strlen(si);
1947    for (i=0; i<=len; i++) {
1948      if (ty=='U')
1949        so[i] = toupper(si[i]);
1950      else
1951        so[i] = tolower(si[i]);
1952    }
1953  }
1954
1955  rigthadj2(len,so,si,ty,dec)
1956  char  ty, so[], si[],dec;
1957  int  len;
1958  {
1959    char tmp[31],tmp2[31];
1960    int i,k,n1,n2,n3;
```

```
1961
1962    if (ty=='S') {
1963      nl=strlen(si);
1964      for (i=nl-1;si[i]==' '&&i>0; i--)
1965        ;
1966      si[i+1]=NULL;
1967      sprintf(tmp,"%30s",si);
1968    } else if (ty=='N')
1969      sprintf(tmp,"%010d",atoi(si));
1970    else if (ty=='M')
1971      sprintf(tmp,"%10d",atoi(si));
1972    else if (ty=='B'||ty=='F') {
1973      for (i=0;i<len;i++)
1974        if (si[i]==NULL || si[i]=='.')
1975          break;
1976      if (si[i]=='.') {
1977        for (k=0;k<5;k++) {
1978          if (si[i+k+1]==NULL)
1979            break;
1980          else if (!(si[i+k+1]>='0'&&si[i+k+1]<='9'))
1981            tmp2[k]='0';
1982          else
1983            tmp2[k]=si[i+k+1];
1984        }
1985        while (k<5)
1986          tmp2[k++]='0';
1987        tmp2[k]=NULL;
1988        n2=atoi(tmp2);
1989      } else
1990        n2=0;
1991      si[i]=NULL;
1992      nl=atoi(si);
1993      if (n2>0)
1994        sprintf(tmp,"%10d.%5s",nl,tmp2);
1995      else
1996        sprintf(tmp,"%10d.00000",nl);
1997      tmp[11+dec]=NULL;
1998    }
1999    nl=strlen(tmp);
2000    for (i=0;i<len;i++)
2001      so[i]=tmp[i+nl-len];
2002    so[i]=NULL;
2003    if (ty=='B') {
2004      for (i=0;i<len;i++)
2005        if (so[i]==' ')
2006          so[i]='0';
2007        else
2008          break;
2009      if (so[i]=='-') {
2010        so[0]='-';
2011        so[i]='0';
2012      }
2013    }
2014  }
2015
2016  realtostr(len,so,si,ty,dec)
```

```
2017  char  ty, so[], si[],dec;
2018  int  len;
2019  {
2020    int j,k;
2021    char stmp[50];
2022    double dx;
2023
2024    dx=atof(si);
2025    sprintf(stmp,"%30.10f",dx);
2026    j=20-(len - dec);
2027    j=(dec>0)?j:j-1;
2028    for (k=0;k<len;k++,j++)
2029      so[k]=stmp[j];
2030  }
2031
2032  setconfirm(n)
2033  char n;
2034  {
2035    _confirm=n;
2036  }
2037
2038  setbell(n)
2039  char n;
2040  {
2041    _bell=n;
2042  }
2043
2044  setcolor(sc)
2045  char *sc;
2046  {
2047    stolower(sc);
2048    if (strcmp(sc,"w")==0)
2049      printf("1m");
2050    else if (strcmp(sc,"w+")==0)
2051      printf("4m");
2052    else if (strcmp(sc,"w*")==0)
2053      printf("5m");
2054    else if (strcmp(sc,"/w")==0)
2055      printf("7m");
2056    else if (strcmp(sc,"w+/n,x")==0)
2057      _noecho=1;
2058    else {
2059      printf("0m");
2060      _noecho=0;
2061    }
2062  }
2063  ::::::::::::::
2064  ut_menu.c
2065  ::::::::::::::
2066  /****
2067  * ut_menu.c : This is a utility for support the menu bar and sayget
2068  *   menu_bar()
2069  *   menu_to()
2070  *   menu_key()
2071  *   saygetmenu()
2072  *   getvar()
```

```
2073  **/
2074
2075  #include <stdio.h>
2076  #include <ctype.h>
2077  #include <math.h>
2078  #include <malloc.h>
2079  #include <sys/types.h>
2080  #include "cst_db.h"
2081
2082  double   atof();
2083  static int     _nf;
2084  static int     _ay[MAXPULL];
2085  static int     _ax[MAXPULL];
2086  static int     _varlen[MAXPULL];
2087  static char    _tokg[4096];
2088  static char    _vartype[MAXPULL];
2089  static char    _vardec[MAXPULL];
2090  static char    _sayname[MAXPULL][80];
2091  static char    _varname[MAXPULL][20];
2092  static char    _vardata[MAXPULL][80];
2093  static char    _varpic[MAXPULL][80];
2094  static char    _varrang1[MAXPULL][20];
2095  static char    _varrang2[MAXPULL][20];
2096  static double  _cstrang1[MAXPULL];
2097  static double  _cstrang2[MAXPULL];
2098  static char    _varvalid[MAXPULL][80];
2099  static char    _varecho[MAXPULL];
2100  static char    _funckey[5];
2101  /* var. for set status
2102  */
2103  static char    _inson;
2104
2105  mgenfield(s1,sp,nf)
2106  char *s1,*sp;
2107  int nf;
2108  {
2109    int i,k,j,cf=0;
2110
2111    s1[0]=NULL;
2112    for (i=0;sp[i]!=NULL;i++) {
2113      if (sp[i]=='@')
2114        cf++;
2115      if (cf>nf)
2116        break;
2117    }
2118    if (sp[i]==NULL)
2119      return(0);
2120
2121    for (k=i+1,j=0;sp[k]!=NULL;k++,j++) {
2122      if (sp[k]=='@')
2123        break;
2124      if (j>=511) {
2125        s1[j]=NULL;
2126        error("** Error: tokgen>512 ",s1);
2127      }
2128      s1[j]=sp[k];
```

```
2129      }
2130      sl[j]=NULL;
2131
2132      return(1);
2133    }
2134
2135    mgenyx(yy,xx,sl)
2136    int *yy,*xx;
2137    char *sl;
2138    {
2139      int i,k,j;
2140      char s2[200];
2141
2142      for (i=0;sl[i]!=NULL;i++) {
2143        if (sl[i]==',')
2144          break;
2145        s2[i]=sl[i];
2146      }
2147      s2[i]=NULL;
2148      *yy=atoi(s2);
2149
2150      for (k=i+1;sl[k]!=NULL;k++) {
2151        if (sl[k]!=' ')
2152          break;
2153      }
2154      for (j=0,i=k;sl[i]!=NULL;i++,j++) {
2155        if (sl[i]==' ')
2156          break;
2157        s2[j]=sl[i];
2158      }
2159      s2[j]=NULL;
2160      *xx=atoi(s2);
2161    }
2162
2163    mgensh(sh,sl)
2164    char *sh,*sl;
2165    {
2166      int i,k,j;
2167
2168      sh[0]=NULL;
2169      for (i=1;sl[i]!=NULL;i++) {
2170        if (sl[i-1]==' '&&sl[i]=='s'&&sl[i+1]=='a'&&sl[i+2]=='y'&&sl[i+3]==' ')
2171          break;
2172      }
2173      if (sl[i]==NULL)
2174        return;
2175      for (k=i+3;sl[k]!=NULL;k++) {
2176        if (sl[k]=='\'')
2177          break;
2178      }
2179      for (j=0,i=k+1;j<=79 && sl[i]!=NULL;i++,j++) {
2180        if (sl[i]=='\'')
2181          break;
2182        sh[j]=sl[i];
2183      }
2184      sh[j]=NULL;
```

```
2185  }
2186
2187  mgenvar(sx,s1)
2188  char *sx,*s1;
2189  {
2190    int i,k,j;
2191
2192    sx[0]=NULL;
2193    for (i=1;s1[i]!=NULL;i++) {
2194      if (s1[i-1]==' '&&s1[i]=='g'&&s1[i+1]=='e'&&s1[i+2]=='t'&&s1[i+3]==' ')
2195        break;
2196    }
2197    if (s1[i]==NULL)
2198      return;
2199    for (k=i+3;s1[k]!=NULL;k++) {
2200      if (s1[k]!=' ')
2201        break;
2202    }
2203    if (s1[k]==NULL)
2204      return;
2205    for (j=0,i=k;j<=19 && s1[i]!=NULL;i++,j++) {
2206      if (s1[i]==' ')
2207        break;
2208      sx[j]=s1[i];
2209    }
2210    sx[j]=NULL;
2211  }
2212
2213  mgenrang(sx1,sx2,cx1,cx2,s1)
2214  char *sx1,*sx2;
2215  char *s1;
2216  double *cx1,*cx2;
2217  {
2218    int i,k,j;
2219
2220    sx1[0]=sx2[0]=NULL;
2221    *cx1=0;
2222    *cx2=0;
2223    for (i=1;s1[i]!=NULL;i++) {
2224      if (s1[i-1]==' '&&s1[i]=='r'&&s1[i+1]=='a'&&s1[i+2]=='n'
2225        &&s1[i+3]=='g'&&(s1[i+4]==' '||(s1[i+4]=='e'&&s1[i+5]==' '))) {
2226        if (s1[i+4]=='e')
2227          s1[i+4]=' ';
2228        break;
2229      }
2230    }
2231    if (s1[i]==NULL)
2232      return;
2233    for (k=i+4;s1[k]!=NULL;k++) {
2234      if (s1[k]!=' ')
2235        break;
2236    }
2237    if (s1[k]==NULL)
2238      return;
2239    for (j=0,i=k;j<=19 && s1[i]!=NULL;i++,j++) {
2240      if (s1[i]==',')
```

```
2241          break;
2242        sx1[j]=s1[i];
2243      }
2244      sx1[j]=NULL;
2245      for (k=i+1;s1[k]!=NULL;k++) {
2246        if (s1[k]!=' ')
2247          break;
2248      }
2249      for (j=0,i=k;j<=19 && s1[i]!=NULL;i++,j++) {
2250        if (s1[i]==' ')
2251          break;
2252        sx2[j]=s1[i];
2253      }
2254      sx2[j]=NULL;
2255      *cx1=atof(sx1);
2256      *cx2=atof(sx2);
2257    }
2258
2259    genvalid(sx1,s1)
2260    char *sx1;
2261    char *s1;
2262    {
2263      int i,k,j;
2264
2265      sx1[0]=NULL;
2266      for (i=1;s1[i]!=NULL;i++) {
2267        if (s1[i-1]==' '&&s1[i]=='v'&&s1[i+1]=='a'&&s1[i+2]=='l'
2268          &&s1[i+3]=='i'&&s1[i+4]=='d'&&s1[i+5]==' ')
2269          break;
2270      }
2271      if (s1[i]==NULL)
2272        return;
2273      for (k=i+5;s1[k]!=NULL;k++) {
2274        if (s1[k]=='\'')
2275          break;
2276      }
2277      if (s1[k]==NULL)
2278        return;
2279      for (j=0,i=k+1;j<=79 && s1[i]!=NULL;i++,j++) {
2280        if (s1[i]=='\'')
2281          break;
2282        sx1[j]=s1[i];
2283      }
2284      sx1[j]=NULL;
2285    }
2286
2287    genvalue(sx1,s1,len)
2288    char *sx1;
2289    char *s1;
2290    int len;
2291    {
2292      int i,k,j;
2293
2294      for (i=0;i<len;i++)
2295        sx1[i]=' ';
2296      sx1[i]=NULL;
```

```
2297
2298      for (i=1;s1[i]!=NULL;i++) {
2299        if (s1[i-1]==' '&&s1[i]=='v'&&s1[i+1]=='a'&&s1[i+2]=='l'
2300          &&s1[i+3]=='u'&&s1[i+4]=='e'&&s1[i+5]==' ')
2301          break;
2302      }
2303      if (s1[i]==NULL)
2304        return;
2305      for (k=i+5;s1[k]!=NULL;k++) {
2306        if (s1[k]=='\'')
2307          break;
2308      }
2309      if (s1[k]==NULL)
2310        return;
2311      for (j=0,i=k+1;j<=79 && j<len && s1[i]!=NULL;i++,j++) {
2312        if (s1[i]=='\'')
2313          break;
2314        sx1[j]=s1[i];
2315      }
2316      for (i=j;i<len;i++)
2317        sx1[i]=' ';
2318      sx1[i]=NULL;
2319    }
2320
2321    gen_echo(echo,s1)
2322    char *echo;
2323    char *s1;
2324    {
2325      int i,k,j;
2326
2327      *echo=' ';
2328
2329      for (i=1;s1[i]!=NULL;i++) {
2330        if (s1[i-1]==' '&&s1[i]=='e'&&s1[i+1]=='c'&&s1[i+2]=='h'
2331          &&s1[i+3]=='o'&&s1[i+4]==' ')
2332          break;
2333      }
2334      if (s1[i]==NULL)
2335        return;
2336      for (k=i+5;s1[k]!=NULL;k++) {
2337        if (s1[k]=='\'')
2338          break;
2339      }
2340      if (s1[k]==NULL)
2341        return;
2342
2343      *echo=toupper(s1[k+1]);
2344    }
2345
2346    /* modi: 7/1/37 use in ut_db3.c
2347    mgenlen(s1,dec,spic,s1)
2348    int *s1;
2349    char *s1,*spic,*dec;
2350    {
2351      int i,k,j;
2352      char cdec=0,idec=0;
```

```
2353
2354      spic[0]=NULL;
2355      *sl=0;
2356      *dec=0;
2357      for (i=1;sl[i]!=NULL;i++) {
2358        if (sl[i-1]==' '&&sl[i]=='p'&&sl[i+1]=='i'&&sl[i+2]=='c'&&
2359            sl[i+3]=='t'&&sl[i+4]==' ')
2360          break;
2361      }
2362      if (sl[i]==NULL)
2363        return;
2364      for (k=i+4;sl[k]!=NULL;k++) {
2365        if (sl[k]=='\'')
2366          break;
2367      }
2368      if (sl[k]==NULL)
2369        return;
2370      for (j=0,i=k+1;sl[i]!=NULL;i++,j++) {
2371        if (sl[i]=='\'' || j>=79)
2372          break;
2373        if (cdec==1 && idec<=9)
2374          idec++;
2375        if (sl[i]=='.')
2376          cdec=1;
2377        spic[j]=toupper(sl[i]);
2378      }
2379      spic[j]=NULL;
2380      *sl=j;
2381      *dec=idec;
2382    }
2383    */
2384
2385    menu_bar(op1,op2,op3,sp)
2386    char *op1,*op2,*op3;
2387    char *sp;
2388    {
2389      int  i,k,ret,cno;
2390      char sl[512],fn[30],tty[20],sbar[90],who[20];
2391      FILE *fp;
2392
2393      if (strcmp(_tokg,sp)!=0) {
2394        if (strlen(sp)>=4096)
2395          error("** Error: Tokgen is long (max 4096)",sp);
2396        _nf=0;
2397        while (1) {
2398          if (genfield(sl,sp,_nf)==0)
2399            break;
2400          genyx(&_ay[_nf],&_ax[_nf],sl);
2401          gensh(_sayname[_nf],sl);
2402          if (_sayname[_nf][0]==NULL||_ay[_nf]<0||_ay[_nf]>25||
2403              _ax[_nf]<0||_ax[_nf]>80) {
2404            endscr();
2405            error("** Error: Text error, ",sl);
2406          }
2407          _nf++;
2408        }
```

```
2409        strcpy(_tokg,sp);
2410      }
2411      if (_nf==0) {
2412        endscr();
2413        error("** Error: Text error, ",sp);
2414      }
2415      for (i=0;i<_nf;i++) {
2416        say(_ay[i],_ax[i],_sayname[i]);
2417      }
2418      i=0;
2419      while(1) {
2420        sprintf(sbar,"%s%s%s",op1,_sayname[i],op2);
2421        say(_ay[i],_ax[i],sbar);
2422        while(1) {
2423          goyx(_ay[i],_ax[i]);
2424          _funckey[0]=NULL;
2425          ret = toupper(getchmenu());
2426          if (strcmp(_funckey,"ESCM")==0) {   /* Press F12 EXIT */
2427            ret=RETURN;
2428            break;
2429          }
2430          if (ret==CTRL_K||ret==RETURN||ret==CTRL_P||
2431             (_nf>1 && ret>=0x82 && ret<=0x85)) {
2432            break;            /* CTRL_K=F12= Exit, RETURN = Select */
2433          }
2434          for (k=0;k<_nf;k++)
2435            if (toupper(_sayname[k][0])==ret) {
2436              sprintf(sbar,"%s%s%s",op2,_sayname[i],op1);
2437              say(_ay[i],_ax[i],sbar);
2438              i=k;
2439              sprintf(sbar,"%s%s%s",op1,_sayname[i],op2);
2440              say(_ay[i],_ax[i],sbar);
2441              ret=RETURN;
2442              break;
2443            }
2444          if (ret==RETURN)
2445            break;
2446          beep();
2447        }
2448        cno=i+1;
2449        if (ret==CTRL_K) {
2450          cno=i=0;
2451          break;
2452        }
2453        if (ret==RETURN||ret==CTRL_P)
2454          break;
2455        sprintf(sbar,"%s%s%s",op2,_sayname[i],op1);
2456        say(_ay[i],_ax[i],sbar);
2457        if (ret==0x82) {   /* UP */
2458          i--;
2459          if (i<0) {
2460            i=_nf-1;
2461          }
2462        } else if (ret==0x83) {   /* DOWN */
2463          i++;
2464          if (i>=_nf) {
```

```
2465            i=0;
2466          }
2467        } else if (ret==0x84) {   /* RIGHT */
2468          for (k=i+1;k<_nf;k++) {
2469            if (_ay[i]==_ay[k])
2470              break;
2471          }
2472          if (_ay[i]==_ay[k])
2473            i=k;
2474          else {
2475            for (k=0;k<i;k++) {
2476              if (_ay[i]==_ay[k])
2477                break;
2478            }
2479            i=k;
2480          }
2481        } else if (ret==0x85) {   /* LEFT */
2482          for (k=i-1;k>0;k--) {
2483            if (_ay[i]==_ay[k])
2484              break;
2485          }
2486          if (_ay[i]==_ay[k] && i!=k)
2487            i=k;
2488          else {
2489            for (k=_nf-1;k>i;k--) {
2490              if (_ay[i]==_ay[k])
2491                break;
2492            }
2493            if (_ay[i]==_ay[k])
2494              i=k;
2495          }
2496        }
2497      }
2498      sprintf(sbar,"%s%s",op3,"Γ╕╪╞═╥íñ╚♦.....");
2499      printf("%d;%dH%s",24,01,sbar);
2500      goyx(23,01);
2501      fflush(stdout);
2502      get_tty(tty);
2503      whoami(who);
2504      sprintf(fn,"/tmp/me-%s-%2s",who,tty);
2505      fp=fopen(fn,"w");
2506      fprintf(fp,"%02d %s\n",cno,_funckey);
2507      fclose(fp);
2508    }
2509
2510    saygetmenu(op1,op2,op3,sp)
2511    char *op1,*op2,*op3;
2512    char *sp;
2513    {
2514      int  i,k,ret;
2515      char s1[512],chg,fn[30],tty[20],sbar[90],tmp[90],who[20];
2516      FILE *fp;
2517
2518      if (strcmp(_tokg,sp)!=0) {
2519        if (strlen(sp)>=4096)
2520          error("** Error: Tokgen is long (max 4096)",sp);
```

```
2521        _nf=0;
2522        while (1) {
2523          if (genfield(sl,sp,_nf)==0)
2524            break;
2525          genyx(&_ay[_nf],&_ax[_nf],sl);
2526          gensh(_sayname[_nf],sl);
2527          genvar(_varname[_nf],sl);
2528          genlen(&_varlen[_nf],&_vardec[_nf],_varpic[_nf],sl);
2529          genrang(_varrang1[_nf],_varrang2[_nf],&_cstrang1[_nf],
2530           &_cstrang2[_nf],sl);
2531          genvalid(_varvalid[_nf],sl);
2532          genvalue(_vardata[_nf],sl,_varlen[_nf]);        /* modi 27/07/93 */
2533          gen_echo(&_varecho[_nf],sl);                     /* modi 27/07/93 */
2534
2535          if (_varpic[_nf][0]=='U')
2536            _vartype[_nf]='U';
2537          else if (_varpic[_nf][0]=='L')
2538            _vartype[_nf]='L';
2539          else if (_varpic[_nf][0]=='9')
2540            _vartype[_nf]='N';
2541          else {
2542            _vartype[_nf]='A';
2543          }
2544          _nf++;
2545        }
2546        strcpy(_tokg,sp);
2547        chg=1;
2548      } else {
2549        chg=0;
2550      }
2551      for (i=0;i<_nf;i++) {
2552        if (_sayname[i][0]!=NULL && chg==1) {
2553          sprintf(sbar,"%s%s%s",op1,_sayname[i],op2);
2554          say(_ay[i],_ax[i],sbar);
2555        }
2556        if (_varname[i][0]!=NULL) {
2557          if (_varecho[i]=='N') {
2558            for (k=0;_vardata[i][k]!=NULL && k<80; k++)
2559              tmp[k]=' ';
2560            tmp[k]=NULL;
2561            sprintf(sbar,"%s%s%s",op2,tmp,op1);
2562          } else
2563            sprintf(sbar,"%s%s%s",op2,_vardata[i],op1);
2564          say(_ay[i],_ax[i]+strlen(_sayname[i]),sbar);
2565        }
2566      }
2567      i=0;
2568      ret=!UP;
2569      while (1) {
2570        if (_varname[i][0]!=NULL) {
2571          say(_ay[i],_ax[i]+strlen(_sayname[i]),op2);
2572          ret=getstrmenu(_vardata[i],_ay[i],_ax[i]+strlen(_sayname[i]),
2573            _varlen[i],_vartype[i],_vardec[i],_varpic[i],_varecho[i]);
2574
2575          if (_vartype[i]=='N' && _varrang1[i][0]!=NULL) {
2576            if (chkrang(_vardata[i],i)==0) {
```

```
            beep();
            continue;
          }
        }
        if (_vartype[i]!='N' && _varvalid[i][0]!=NULL) {
          if (chkvalid(_vardata[i],i)==0) {
            beep();
            continue;
          }
        }
      }
      if (ret==CTRL_K||ret==CTRL_P)
        break;

      if (ret==UP) {
        i--;
        while (i>=0 && _varname[i][0]==NULL)
          i--;
        if (i<0) {
          i=_nf-1;
          break;
        }
      } else {
        i++;
        while (i<_nf && _varname[i][0]==NULL)
          i++;
        if (i>=_nf) {
          i=0;
          break;
        }
      }
    }
    sprintf(sbar,"%s%s",op3,"Γ╕├┤├═╤íñ┘♦.....");
    printf("%d;%dH%s",24,01,sbar);
    goyx(23,01);
    fflush(stdout);
    get_tty(tty);
    whoami(who);
    sprintf(fn,"/tmp/sa-%s-%2s",who,tty);
    fp=fopen(fn,"w");
    for (i=0;i<_nf;i++) {
      if (_varname[i][0]!=NULL) {
        fprintf(fp,"%-15s %s\n",_varname[i],_vardata[i]);
      }
    }
    if (ret==CTRL_K)
      fprintf(fp,"%-15s %s\n","return","F12");

    fclose(fp);

    return(ret);
  }

  mchkrang(sd,ix)
  char *sd;
  int  ix;
```

```
2633  {
2634    int i;
2635    char s[90];
2636    double dx;
2637
2638    strcpy(s,sd);
2639    trim(s);
2640    dx=atof(s);
2641    if (dx>=_cstrang1[ix] && dx<=_cstrang2[ix])
2642      return(1);
2643    else
2644      return(0);
2645  }
2646
2647  mchkvalid(sd,ix)
2648  char *sd;
2649  int  ix;
2650  {
2651    int i,len;
2652    char s[80];
2653
2654    len=strlen(_varvalid[ix]);
2655    for (i=1;i<=len;i=i+_varlen[ix]) {
2656      substr(_varvalid[ix],i,_varlen[ix],s);
2657      if (strcmp(sd,s)==0)
2658        return(1);
2659    }
2660    return(0);
2661  }
2662
2663  getstrmenu(t,y,x,n,ty,dec,spic,echo)
2664  char t[],x,y,ty,dec,*spic,echo;
2665  char n;
2666  {
2667    int i=0,j,c,len;
2668    char s[128],xb;
2669
2670    strcpy(s,t);
2671
2672    xb=x;
2673    while (1) {
2674      goyx(y,x);
2675      c = getchmenu();
2676
2677      if (ty=='U')
2678        c=toupper(c);
2679      else if (ty=='L')
2680        c=tolower(c);
2681
2682      switch(c) {
2683        case 0x0d:  /* carriage return */
2684          break;
2685
2686        case 0x09:  /* insert switch mode */
2687          _inson = !_inson;
2688          break;
```

```
2689
2690        case 0x08:  /* backspace */
2691          if (i>0) {
2692            i--;
2693            x--;
2694            shift_me_lf(s,i);
2695            say_yx(y,x,s,i,echo);
2696          }
2697          break;
2698
2699        case 0x7F:  /* DELETE */
2700          if (i<n) {
2701            shift_me_lf(s,i);
2702            say_yx(y,x,s,i,echo);
2703          }
2704          break;
2705
2706        case 0x85:  /* shift left with arrow <== */
2707          if (i>0) {
2708            i--;
2709            x--;
2710          }
2711          break;
2712
2713        case 0x84:  /* shift right with arrow ==> */
2714          if (i<n) {
2715            i++;
2716            x++;
2717          }
2718          break;
2719
2720        case 0x19:  /* CTRL_Y Delete field */
2721          for (j=0;j<n;j++)
2722            s[j]=' ';
2723          i=0;
2724          x=xb;
2725          say_yx(y,xb,s,0,echo);
2726          break;
2727
2728        case 0x14:  /* CTRL_T Delete word right */
2729          for (j=i;j<n;j++)
2730            s[j]=' ';
2731          say_yx(y,x,s,i,echo);
2732          break;
2733
2734        default:  /* any characters between 0x20 to 0xfe */
2735          if (!(c==0x0d || c==0x82 || c==0x83 || c==CTRL_K || c==CTRL_P ||
2736                c==CTRL_R || c==CTRL_C || c==CTRL_E || c==CTRL_X)) {
2737            if (spic[i]=='9' || spic[i]=='.') {
2738              if (!(isdigit(c)||c=='.'||c=='-'||c=='+')) {
2739                beep();
2740                continue;
2741              }
2742            }
2743          }
2744          if (i<n && (unsigned char)c>0x1f && c!=0x82 && c!=0x83) {
```

```
2745            if (_inson) {
2746              shift_me_rt(s,i);
2747              s[i] = c;
2748              say_yx(y,x,s,i,echo);
2749            } else {
2750              s[i] = c;
2751              if (echo!='N') {
2752                putc(c,stdout);
2753                fflush(stdout);
2754              }
2755            }
2756            i++;
2757            x++;
2758          }
2759          break;
2760      }
2761      if (c==0x0d || c==0x82 || c==0x83 || c==CTRL_K || c==CTRL_P ||
2762          c==CTRL_R || c==CTRL_C || c==CTRL_E || c==CTRL_X) {
2763        switch(c) {
2764          case 0x0D: c=RETURN; break;
2765          case 0x82: c=UP; break;
2766          case 0x83: c=DOWN; break;
2767          case 0x19: c=RETURN; break;
2768        }
2769        break;
2770      }
2771    }
2772    strcpy(t,s);
2773
2774    return(c);
2775  }
2776
2777  say_yx(y,x,s,ix,echo)
2778  char y,x,*s,echo;
2779  int ix;
2780  {
2781    int j,k;
2782    char s2[128];
2783
2784    if (echo=='N')
2785      return;
2786
2787    for (k=0,j=ix;s[j]!=NULL && j<125;k++,j++)
2788      s2[k]=s[j];
2789    s2[k]=NULL;
2790
2791    say(y,x,s2);
2792  }
2793
2794  getchmenu()   /* read character in raw mode */
2795  {
2796    int  c;
2797
2798    c = getc(stdin);
2799    if (c == ESC) {
2800      c = getc(stdin);
```

```
2801        if (c == '[') {
2802          c = getc(stdin);
2803          sprintf(_funckey,"ESC%c",c);
2804          switch(c) {
2805            case 'A' : c = 0x82;   break; /* up */
2806            case 'B' : c = 0x83;   break; /* down */
2807            case 'C' : c = 0x84;   break; /* right */
2808            case 'D' : c = 0x85;   break; /* left */
2809            case 'G' : c = CTRL_P; break; /* PageDown */
2810            case 'X' : c = CTRL_K; break; /* F12 */
2811            case 'L' : c = 0x09;   break; /* Insert */
2812            default:   c = 0x83;   break; /* Other is down */
2813          }
2814        }
2815      }
2816      return(c);
2817    }
2818
2819    shift_me_lf(s,ix)
2820    char *s;
2821    int ix;
2822    {
2823      int j;
2824
2825      for (j=ix; s[j+1]!=NULL && j<125; j++)
2826        s[j] = s[j+1];
2827      s[j]=' ';
2828    }
2829
2830    shift_me_rt(s,ix)
2831    char *s;
2832    int ix;
2833    {
2834      int j,len;
2835
2836      len=strlen(s);
2837
2838      for (j=len; j>ix && j>0; j--)
2839        s[j]=s[j-1];
2840      s[len]=NULL;
2841    }
2842    ::::::::::::::
2843    ut_tty.c
2844    ::::::::::::::
2845    /***
2846    * ut_tty.c: This is a utility group tty, include functions
2847    ***/
2848
2849    #include <stdio.h>
2850    #include <ctype.h>
2851    #include <signal.h>
2852
2853    get_tty(tty)
2854    char *tty;
2855    {
2856      char fn[30],comm[40],s[82],s2[82];
```

```
2857    int  i,j;
2858    FILE *fp;
2859
2860    sprintf(fn,"/tmp/sayget%08d",getpid());
2861    sprintf(comm,"tty > %s",fn);
2862    system(comm);
2863    fp=fopen(fn,"r");
2864    fgets(s,80,fp);
2865    for (i=0,j=0;i<80 && s[i]!=NULL;i++) {
2866      if ((s[i]>='0'&&s[i]<='9')||
2867          (s[i]>='A'&&s[i]<='Z')||
2868          (s[i]>='a'&&s[i]<='z'))
2869        s2[j++]=s[i];
2870    }
2871    s2[j]=NULL;
2872    if (j<2)
2873      strcpy(s2,"xyz");
2874    tty[0]=s2[j-2];
2875    tty[1]=s2[j-1];
2876    tty[2]=NULL;
2877    sprintf(comm,"rm %s",fn);
2878    system(comm);
2879  }
2880
2881  whoami(who)
2882  char *who;
2883  {
2884    char fn[30],comm[40],s[82];
2885    FILE *fp;
2886
2887    sprintf(fn,"/tmp/who%08d",getpid());
2888    sprintf(comm,"whoami > %s",fn);
2889    system(comm);
2890    fp=fopen(fn,"r");
2891    fgets(s,80,fp);
2892    strcpy(who,s);
2893    trim(who);
2894    sprintf(comm,"rm %s",fn);
2895    system(comm);
2896  }
2897  ::::::::::::::
2898  append.c
2899  ::::::::::::::
2900  /* append.c: This is a program for append the the Database file
2901  *           Written by Mongkol kuanhavet, 18 Jun 92
2902  *           Computer center, Prince of Songkla Univ.(PSU), Hathai, Thailand.
2903  *
2904  *     usage: append fn1 from fn2 [sdf]
2905  *
2906  *              fn1=database file, fn2=database file or text file
2907  */
2908  #include <stdio.h>
2909  #include <ctype.h>
2910  #include <math.h>
2911  #include "cst_db.h"
2912
```

```
2913  FILE   *fpinx1[2],*fpinx2[2],*fpstc1,*fpstc2;
2914  char   buf1[1025],buf2[1025],fnstc1[100],fnstc2[100],fnabort[100],comm[120];
2915  long   lret,freadinx();
2916  char   A_fieldname[MAXFIELD][11];
2917  char   B_fieldname[MAXFIELD][11];
2918  char   A_fieldtype[MAXFIELD];
2919  char   B_fieldtype[MAXFIELD];
2920  char   *A_datachar[MAXFIELD];
2921  double  A_datareal[MAXFIELD];
2922  char   *B_datachar[MAXFIELD];
2923  double  B_datareal[MAXFIELD];
2924  int    A_tfield=0;
2925  int    B_tfield=0;
2926  char   **av;
2927
2928  main(ac,argv)
2929  int  ac;
2930  char **argv;
2931  {
2932    if (!(ac==4 || ac==5) || strcmp(argv[2],"from")!=0 ||
2933        (ac==5 && strcmp(argv[4],"sdf")!=0)) {
2934      fprintf(stderr,"usage: append fn1 from fn2 [sdf]\n");
2935      exit(0);
2936    }
2937    av=argv;
2938
2939    if (ac==5)
2940      appendsdf(ac);                          /* Append from standard file. */
2941    else {
2942      if (isdbf(av[1]) && isdbf(av[3]))
2943        appenddbf(ac);                        /* Append from Database file. */
2944      else {
2945        fprintf(stderr,"%s is not a database file\n",av[3]);
2946        exit(-1);
2947      }
2948    }
2949  }
2950
2951  appendsdf(ac)
2952  int  ac;
2953  {
2954    strcpy(fnstc1,av[1]);
2955    strcat(fnstc1,".stc");
2956    if (!file(fnstc1))                        /* Check file is exists? */
2957      error ("can't open file: ",fnstc1);
2958
2959    if (fopeninx(av[1],"r+",fpinx1)==NULL)    /* Open Index file for update*/
2960      error ("can't open file: ",av[1]);
2961
2962    if ((fpinx2[0]=fopen(av[3],"r"))==NULL)   /* Open standard file for raed*/
2963      error ("can't open file: ",av[3]);
2964
2965    while (fgets(buf1,1025,fpinx2[0])!=NULL) {
2966      lret = freadinx(buf2,buf1,fpinx1);      /* Read Database file by key */
2967      if (lret==0L)                           /* Check key not found */
2968        fwriteinx(buf1,fpinx1,"a");           /* Append Database file */
```

```
2969        else
2970          frewriteinx(buf1,lret,fpinx1);        /* Update Database file */
2971      }
2972      fcloseinx(fpinx1);
2973      fclose(fpinx2[0]);
2974    }
2975
2976    appenddbf(ac)
2977    int  ac;
2978    {
2979      int  i,k,j,delon,kx;
2980      char skey[256];
2981
2982      select(1);
2983      use(av[1],"r+");                          /* Open or use master file */
2984      A_tfield=gettfield();                     /* get total no. of field */
2985      for (i=0;i<A_tfield;i++) {
2986        getfieldname(A_fieldname[i],i);         /* get field name */
2987        stolower(A_fieldname[i]);
2988        getfieldtype(&A_fieldtype[i],i);        /* get field type */
2989        if (A_fieldtype[i]=='c')
2990          mapchar(A_fieldname[i],&A_datachar[i]); /* mapping char. type field */
2991        else
2992          mapreal(A_fieldname[i],&A_datareal[i]); /* mapping real type field */
2993      }
2994      gotop();                                  /* store 1st record to mem. var*/
2995
2996      select(2);
2997      use(av[3],"r");                           /* Open the transcation file */
2998      B_tfield=gettfield();
2999      for (i=0;i<B_tfield;i++) {                /* mappig the field, the same*/
3000        getfieldname(B_fieldname[i],i);
3001        stolower(B_fieldname[i]);
3002        getfieldtype(&B_fieldtype[i],i);
3003        if (B_fieldtype[i]=='c')
3004          mapchar(B_fieldname[i],&B_datachar[i]);
3005        else
3006          mapreal(B_fieldname[i],&B_datareal[i]);
3007      }
3008      gotop();
3009
3010      select(1);
3011      clean();
3012      select(2);
3013      while (!eof()) {                          /* get transaction until eof */
3014        delon=(isdelete())?1:0;
3015        select(1);
3016        for (i=0;i<A_tfield;i++) {              /* check for mapping any field */
3017          for (k=0;k<B_tfield;k++) {
3018            if (strcmp(A_fieldname[i],B_fieldname[k])==0) {
3019                if (A_fieldtype[i]=='c')    /* store master field from transac.*/
3020                  replchar(A_fieldname[i],B_datachar[k]);
3021                else
3022                  replreal(A_fieldname[i],B_datareal[k]);
3023                break;
3024            }
```

```
3025          }
3026        }
3027        append();                              /* append the master file */
3028        if (delon==1)
3029          delete();              /* is delete on, delete it record */
3030        else
3031          recall();
3032        select(2);
3033        skip(1);
3034      }
3035      closedata();
3036
3037      sprintf(fnabort,"%s.@@@",av[1]);    /* mark file for interrrupt */
3038      if (file(fnabort)) {
3039        sprintf(comm,"rm %s",fnabort);    /* remove it when process is completed*/
3040        system(comm);
3041      }
3042    }
3043    ::::::::::::::
3044    copy.c
3045    ::::::::::::::
3046    /* copy.c: This is a program for copy the the Database file
3047    *          Written by Mongkol kuanhavet, 18 Jun 92
3048    *          Computer center, Prince of Songkla Univ., Hathai, Thailand.
3049    *
3050    *   usage: copy fn1 to fn2 [stru]
3051    */
3052    #include <stdio.h>
3053
3054    main(ac,av)
3055    int  ac;
3056    char **av;
3057    {
3058      FILE *fpinx1[2],*fpstc1;
3059      char  fnstc1[100],fnstc2[100];
3060      char  fninx1[100],fninx2[100],comm[256],shead[513],stru=0,av3[100];
3061      long  ihead[1025],lret,freadinx();
3062      int   ftype;
3063
3064      if (!(ac==4 || ac==5) || strcmp(av[2],"to")!=0 ||
3065          (ac==5 && strcmp(av[4],"stru")!=0)) {
3066        fprintf(stderr,"usage: copy fn1 to fn2 [stru]\n");
3067        exit(0);
3068      }
3069      if (!isdbf(av[1]))
3070        error ("not database file: ",av[1]);
3071
3072      if (ac==5)
3073        stru=1;
3074
3075      ftype=filetype(av[3]);
3076      if (ftype=='-' || ftype==NULL)
3077        sprintf(av3,"%s",av[3]);
3078      else
3079        sprintf(av3,"%s/%s",av[3],av[1]);
3080
```

```
3081    sprintf(fnstc1,"%s.stc",av[1]);
3082    sprintf(fnstc2,"%s.stc",av3);
3083    sprintf(fninx1,"%s.inx",av[1]);
3084    sprintf(fninx2,"%s.inx",av3);
3085
3086    if (stru==1) {
3087      if (fopeninx(av[1],"r",fpinx1)==NULL)
3088        error ("file not found: ",av[1]);
3089      _read_header(fpinx1,ihead,shead);
3090      fcloseinx(fpinx1);
3091      fcreatinx(av3,(int)ihead[2],(int)ihead[15],(int)ihead[16],fpinx1);
3092      fcloseinx(fpinx1);
3093    } else {
3094      sprintf(comm,"cp %s %s",av[1],av3);
3095      system(comm);
3096      sprintf(comm,"cp %s %s",fninx1,fninx2);
3097      system(comm);
3098    }
3099    sprintf(comm,"cp %s %s",fnstc1,fnstc2);
3100    system(comm);
3101  }
3102  ::::::::::::::
3103  create.c
3104  ::::::::::::::
3105  /* create.c: This is a program for create the Database file
3106  *           Written by Mongkol kuanhavet, 18 Jun 92
3107  *           Computer center, Prince of Songkla Univ., Hathai, Thailand.
3108  *
3109  *    create:  create fn
3110  */
3111  #include <stdio.h>
3112  #include <ctype.h>
3113  #include "cst_db.h"
3114
3115  FILE *fpinx[2],*fpstc,*fpdol;
3116  long ihead[1025];
3117  char shead[513],s[200],name[11],type[12],fnstc[128],key[3],c,comm[80];
3118  char av1[80],av2[80],av3[80],fndol[100],bfield[MAXFIELD][11];
3119  char bname[8][11],blen[8],btype[8];
3120  int  i,n,len,dec,tlen,klen,koffset,flagkey=1,flagerr=0,tedit=0;
3121
3122  main(ac,av)
3123  int  ac;
3124  char **av;
3125  {
3126    if (ac<2||ac>4) {
3127      fprintf(stderr,"usage: create fn [-h]\n");
3128      exit(0);
3129    }
3130    strcpy(av1,av[1]);
3131    if (file(av1)) {
3132      fprintf(stderr,"File '%s' is exists\n",av1);
3133      exit(0);
3134    }
3135    if (ac==4) {
3136      strcpy(av2,av[2]);
```

```
3137      strcpy(av3,av[3]);
3138    } else {
3139      strcpy(av2," ");
3140      strcpy(av3," ");
3141    }
3142    if (strcmp(av1,"-h")==0 && strcmp(av2,"-h")!=0) {
3143      strcpy(comm,av2);
3144      strcpy(av2,av1);
3145      strcpy(av1,comm);
3146    }
3147    strcpy(fnstc,av1);
3148    strcat(fnstc,".stc");
3149    strcpy(fndol,av1);
3150    strcat(fndol,".dol");
3151
3152    if (strcmp(av2,"-r")!=0) {
3153      if (file(fnstc)) {
3154        fprintf(stderr,"** warning: file %s is exists.. Overwrite(Y/N):",fnstc);
3155        c=getyn();
3156        fprintf(stderr,"\n");
3157        if (tolower(c)!='y')
3158          exit(0);
3159      }
3160      if ((fpstc=fopen(fnstc,"w"))==NULL)
3161        error ("can't create file: ",fnstc);
3162
3163      if (strcmp(av2,"-h")==0) {
3164        fprintf(fpstc,"/*---------------------------------------------\n");
3165        fprintf(fpstc,"/* This is a format file for create database file\n");
3166        fprintf(fpstc,"/*   Type '/' and '*' at column 1 for comment\n");
3167        fprintf(fpstc,"/*   column    Description\n");
3168        fprintf(fpstc,"/* ---------------------------------------------\n");
3169        fprintf(fpstc,"/*     1-10    Field_name\n");
3170        fprintf(fpstc,"/*    11-11    Field_type (c=char,n=real)\n");
3171        fprintf(fpstc,"/*    12-14    Field_width\n");
3172        fprintf(fpstc,"/*    15-16    Decimal\n");
3173        fprintf(fpstc,"/*    17-17    Key field (k)\n");
3174        fprintf(fpstc,"/*    21-78    Note\n");
3175        fprintf(fpstc,"/*345678901234567890123456789012345678901234567890123456789012345678\n");
3176        fprintf(fpstc,"Field_1   c  7  k   Field #1 is a key field\n");
3177        fprintf(fpstc,"Field_2   c 30      Field #2 is character field\n");
3178        fprintf(fpstc,"Field_3   n  9 2    Field #3 is a numeric field\n");
3179        fprintf(fpstc,"Field_n   c  1      Field #n is a other field\n");
3180      } else {
3181        fprintf(fpstc,"/*345678901234567\n");
3182        fprintf(fpstc,"Field_1   c  7  k\n");
3183        fprintf(fpstc,"Field_2   c 30   \n");
3184        fprintf(fpstc,"Field_3   n  9 2 \n");
3185        fprintf(fpstc,"Field_n   c  1   \n");
3186      }
3187      fclose(fpstc);
3188    }
3189    while (1) {
3190      sprintf(comm,"vi %s",fnstc);
3191      system(comm);
3192      flagkey=1;
```

```
3193        flagerr=0;
3194        creatdbf();
3195        if (flagerr==0) {
3196          if (strcmp(av2,"-r")==0)
3197            mkdol(av3);
3198          else {
3199            mkdol(av1);
3200            fprintf(stderr,"Input data records now(Y/N)?");
3201            c=getyn();
3202            c=tolower(c);
3203            if (c=='y') {
3204              sprintf(comm,"editdbf %s",av1);
3205              system(comm);
3206            }
3207          }
3208          exit(0);
3209        }
3210        fprintf(stderr,"** Error: Edit your data AGAIN(Y/N)?");
3211        c=getyn();
3212        c=tolower(c);
3213        if (c=='n') {
3214          sprintf(comm,"rm %s",fnstc);
3215          system(comm);
3216          exit(0);
3217        }
3218      }
3219    }
3220
3221    creatdbf()
3222    {
3223      int i,nf,fno=0;
3224
3225      if ((fpstc=fopen(fnstc,"r"))!=NULL) {
3226        tlen=0;
3227        klen=0;
3228        koffset=9999;
3229        tedit=0;
3230        while (fgets(s,200,fpstc)!=NULL) {
3231          if (s[0]=='/'&&s[1]=='*')
3232            continue;
3233          substr(s,1,10,name);
3234          substr(s,11,1,type);
3235          len=stoi(s,12,3);
3236          dec=stoi(s,15,2);
3237          substr(s,17,1,key);
3238          type[0]=tolower(type[0]);
3239          key[0]=tolower(key[0]);
3240          chkfield(fno);
3241          if (key[0]=='k') {
3242            klen=klen+len;
3243            if (koffset>tlen)
3244              koffset=tlen;
3245          }
3246          nf=len/68;
3247          nf=((nf*68)<len)?nf+1:nf;
3248          tedit=tedit+nf;
```

```
3249          tlen=tlen+len;
3250          fno++;
3251          if (fno>MAXFIELD) {
3252            fprintf(stderr,"** error: Total field grater than %d\n",MAXFIELD);
3253            flagerr=1;
3254          }
3255        }
3256        if (klen==0) {
3257          fprintf(stderr,"** error: key field not found in:%s\n",fnstc);
3258          flagerr=1;
3259        }
3260        if (flagerr==0) {
3261          fcreatinx(av1,tlen+2,klen,koffset,fpinx);
3262        }
3263      }
3264    }
3265
3266    chkfield(fno)
3267    int fno;
3268    {
3269      int i,k;
3270
3271      for (i=0;i<fno;i++) {
3272        if (strcmp(name,bfield[i])==0) {
3273          fprintf(stderr,"** Error: Field %s is duplicate\n",name);
3274          flagerr=1;
3275        }
3276      }
3277      strcpy(bfield[fno],name);
3278      if (flagkey==1) {
3279        if (key[0]!='k')
3280          flagkey=0;
3281      } else {
3282        if (key[0]=='k') {
3283          fprintf(stderr,"** Error: Key must be start at the first field\n");
3284          flagerr=1;
3285        }
3286      }
3287      if (key[0]=='k' && type[0]!='c') {
3288        fprintf(stderr,"** Error: Type for Key must be character\n");
3289        flagerr=1;
3290      }
3291      if (!(isalpha(name[0]))) {
3292        fprintf(stderr,"** Error: Invalid field name=%s\n",name);
3293        flagerr=1;
3294        return;
3295      }
3296      for (i=1;i<10&&name[i]!=' ';i++) {
3297        if (!(isalpha(name[i])||isdigit(name[i])||name[i]=='_')) {
3298          fprintf(stderr,"** Error: Invalid field name=%s\n",name);
3299          flagerr=1;
3300          return;
3301        }
3302      }
3303      for (k=i;k<10;k++) {
3304        if (name[k]!=' ') {
```

```
3305        fprintf(stderr,"** Error: Invalid field name=%s\n",name);
3306        flagerr=1;
3307        return;
3308      }
3309    }
3310    if (!(type[0]=='c'||type[0]=='n')) {
3311      fprintf(stderr,"** Error: %s: Type must be 'c' or 'n'\n",name);
3312      flagerr=1;
3313    }
3314    if (type[0]=='n' && (len>30||dec>9)) {
3315      fprintf(stderr,"** Error: %s Numeric field width >30 or decimal>9\n",name);
3316      flagerr=1;
3317    }
3318    if (len<1) {
3319      fprintf(stderr,"** Error: %s Field width must be grater than 0\n",name);
3320      flagerr=1;
3321    }
3322  }
3323
3324  mkdol(fn)
3325  char *fn;
3326  {
3327    int  i,ix,len68,y=0,pk=1,b8=1,tpage,cpage=1,nf,py1=1;
3328    char cd,nextp;
3329
3330    if ((fpdol=fopen(fndol,"w"))==NULL)
3331      error ("can't create file: ",fndol);
3332
3333    tpage=tedit/22;
3334    tpage=(tedit>(tpage*22))?tpage+1:tpage;
3335    nextp=(tpage>1)?'Y':' ';
3336
3337    for (i=0;i<72;i++)
3338      fprintf(fpdol,"#");
3339    fprintf(fpdol," \n");
3340    fprintf(fpdol,"%40s01 01 01 001 001 [ A\n"," ");
3341    fn[34]=NULL;
3342
3343    tlen=0;
3344    klen=0;
3345    koffset=9999;
3346    rewind(fpstc);
3347
3348    while (fgets(s,200,fpstc)!=NULL) {
3349      if (s[0]=='/' && s[1]=='*')
3350        continue;
3351
3352      if (y==0 || y>22) {
3353        for (i=0;i<72;i++)
3354          fprintf(fpdol,"-");
3355        fprintf(fpdol," \n");
3356        if (tpage>1 && tpage<10)
3357          fprintf(fpdol,"Page: %1d/%1d %30s60 00            7 \n",
3358            cpage,tpage," ");
3359        else if (tpage>1 && tpage<20)
3360          fprintf(fpdol,"Page: %2d/%2d %30s60 00            7 \n",
```

```
3361            cpage,tpage," ");
3362          y=1;
3363          cpage=cpage+1;
3364        }
3365        substr(s,1,10,name);
3366        substr(s,11,1,type);
3367        len=stoi(s,12,3);
3368        dec=stoi(s,15,2);
3369        substr(s,17,1,key);
3370        cd=' ';
3371        if (type[0]=='n'||type[0]=='N') {
3372          if (dec>0) {
3373            cd='0'+dec;
3374            type[0]='F';
3375          }
3376        } else if (type[0]=='c')
3377          type[0]='A';
3378        else if (type[0]=='C')
3379          type[0]='U';
3380
3381        if (key[0]=='k') {
3382          strcpy(bname[pk],name);
3383          blen[pk]=len;
3384          btype[pk]=type[0];
3385          fprintf(fpdol,"%-40s01 %02d 11 %03d %03d [ %c D %ld 1 P %c\n",name,y,
3386            tlen+1,tlen+len,type[0],pk,nextp);
3387          y++;
3388          pk++;
3389          tlen=tlen+len;
3390        } else {
3391          nf=len/68;
3392          nf=((nf*68)<len)?nf+1:nf;
3393          for (ix=1;ix<=nf;ix++) {
3394            if (ix<nf)
3395              len68=68;
3396            else
3397              len68=len -((ix-1)*68);
3398            if (ix==1 || py1==1) {
3399              if (b8==1) {
3400                fprintf(fpdol,"%-40s01 %02d 11 %03d %03d%c[ %c   %ld\n",name,y,
3401                  tlen+1,tlen+len68,cd,type[0],8);
3402                b8=0;
3403              } else {
3404                fprintf(fpdol,"%-40s01 %02d 11 %03d %03d%c[ %c  \n",name,y,
3405                  tlen+1,tlen+len68,cd,type[0]);
3406              }
3407              py1=0;
3408            } else {
3409              fprintf(fpdol,"%-38sN 01 %02d 11 %03d %03d%c[ %c  \n",name,y,
3410                tlen+1,tlen+len68,cd,type[0]);
3411            }
3412            tlen=tlen+len68;
3413            y++;
3414            if (y>22) {
3415              klen=0;
3416              for (i=0;i<72;i++)
```

```
3417              fprintf(fpdol,"-");
3418            fprintf(fpdol," \n");
3419            if (tpage>1 && tpage<10)
3420              fprintf(fpdol,"Page: %1d/%1d %30s60 00            7 \n",
3421                cpage,tpage," ");
3422            else if (tpage>1 && tpage<20)
3423              fprintf(fpdol,"Page: %2d/%2d %30s60 00            7 \n",
3424                cpage,tpage," ");
3425            y=1;
3426            klen=0;
3427            for (i=1;i<pk;i++) {
3428              fprintf(fpdol,"%-40s01 %02d 11 %03d %03d < %c D %1d 1 P %c\n",
3429                bname[i],y,klen+1,klen+blen[i],btype[i],i,nextp);
3430              y++;
3431              klen=klen+blen[i];
3432            }
3433            cpage=cpage+1;
3434            py1=1;
3435          }
3436        }
3437      }
3438    }
3439    for (i=0;i<72;i++)
3440      fprintf(fpdol,"*");
3441    fprintf(fpdol," \n");
3442    fclose(fpstc);
3443    fclose(fpdol);
3444    sprintf(comm,"cat %s >> %s",fndol,fnstc);
3445    system(comm);
3446    sprintf(comm,"rm %s",fndol);
3447    system(comm);
3448  }
3449  ::::::::::::::
3450  delete.c
3451  ::::::::::::::
3452  /* delete.c: This is a program for delete record of the database
3453  *           Written by Mongkol kuanhavet, 18 Jun 92
3454  *           Computer center, Prince of Songkla Univ.(PSU), Hatyai, Thailand.
3455  *           Under the VAX 11/785, ULTRIX Operating system v3.11
3456  *
3457  * compile: cc delete.c /staff/mongkol/bin/psurdbms /staff/mongkol/bin/libinx
3458  *
3459  * run:     a.out fn [scope]
3460  */
3461  #include <stdio.h>
3462  #include <math.h>
3463  #include "cst_db.h"
3464
3465  main(ac,av)
3466  int ac;
3467  char **av;
3468  {
3469    int  irec,brec=1,erec=9999999;
3470
3471    if (ac<2) {
3472      fprintf(stderr,"usage: delete fn [from_record to_record]\n");
```

```
3473      fprintf(stderr,"exam : delete name 5 20\n");
3474      exit(1);
3475    }
3476    select(1);                          /* Seleect work area #1 */
3477    use(av[1],"r+");                    /* Use or open the database file */
3478    gotop();                            /* Get the 1st record to memory var.*/
3479
3480    if (ac>2) {                         /* Check the scope for delete */
3481      brec=atoi(av[2]);
3482      if (ac>3)
3483        erec=atoi(av[3]);
3484      else
3485        erec=brec;
3486    }
3487    while (!eof()) {                    /* Get record until eof */
3488      irec=recno();
3489      if (irec>erec)                    /* If out of scope then exit */
3490        break;
3491      if (irec>=brec)                   /* Is a deleted record? */
3492        delete();                       /* Mark record for deletion */
3493      skip(1);                          /* Get next record */
3494    }
3495    closedata();
3496  }
3497  ::::::::::::::
3498  dir.c
3499  ::::::::::::::
3500  /* dir.c: This is a program for list the contents of the Database directory
3501  *         Written by Mongkol kuanhavet, 18 Jun 92
3502  *         Computer center, Prince of Songkla Univ., Hathai, Thailand.
3503  */
3504  #include <stdio.h>
3505
3506  main(ac,av)
3507  int  ac;
3508  char **av;
3509  {
3510    int   i,trec,reclen,keylen,keyoffs,size,tfiles=0,tsize=0;
3511
3512    if (ac<2) {
3513      fprintf(stderr,"usage: dir {fn,*}\n");
3514      exit(0);
3515    }
3516    printf("%-31s%-10s%-10s%-18s\n","Database Files"," # Records",
3517      "  Rec len."," Key len.   Sizes");
3518
3519    for (i=1;i<ac;i++) {
3520      if (!isdbf(av[i]))
3521        continue;
3522      select(1);
3523      use(av[i],"r");
3524      trec=gettrec();
3525      getkeyinfo(&reclen,&keylen,&keyoffs);
3526      size=trec*reclen;
3527      printf("%-30s%10d%10d%9d%10d\n",av[i],trec,reclen,keylen,size);
3528      tfiles++;
```

```
3529      tsize=tsize+size;
3530    }
3531    printf("\n    ** Total %2ld bytes  in %ld files.\n",tsize,tfiles);
3532    closedata();
3533  }
3534  ::::::::::::::
3535  erase.c
3536  ::::::::::::::
3537  /* erase.c: This is a program for erase the the Database file
3538  *          Written by Mongkol kuanhavet, 18 Jun 92
3539  *          Computer center, Prince of Songkla Univ., Hathai, Thailand.
3540  *
3541  *   usage: erase fn1
3542  */
3543  #include <stdio.h>
3544
3545  main(ac,av)
3546  int  ac;
3547  char **av;
3548  {
3549    FILE *fpinx1[2],*fpstc1;
3550    char  fnstc1[100],fnstc2[100];
3551    char  fninx1[100],fninx2[100],comm[256],shead[513],stru=0,av3[100];
3552    long  ihead[1025],lret,freadinx();
3553    int   ftype;
3554
3555    if (ac!=2) {
3556      fprintf(stderr,"usage: erase fn\n");
3557      exit(0);
3558    }
3559    sprintf(fnstc1,"%s.stc",av[1]);
3560    sprintf(fninx1,"%s.inx",av[1]);
3561
3562    if (!file(av[1])) {
3563      fprintf(stderr,"file not found: %s\n",av[1]);
3564      exit(0);
3565    }
3566    if (!file(fnstc1))
3567      fnstc1[0]=NULL;
3568    if (!file(fninx1))
3569      fninx1[0]=NULL;
3570
3571    sprintf(comm,"rm %s %s %s",av[1],fnstc1,fninx1);
3572    system(comm);
3573  }
3574  ::::::::::::::
3575  getvar.c
3576  ::::::::::::::
3577  #include <stdio.h>
3578  #include <stdlib.h>
3579
3580  main(ac,av)
3581  char ac,**av;
3582  {
3583    char s[257],tty[10],key[16],data[257],fxx[16],fn[30],who[20];
3584    int len;
```

```
3585    FILE *fp;
3586
3587    if (ac!=2)
3588      error("usage: getvar field_name"," ");
3589
3590    strcpy(key,av[1]);
3591    len=strlen(key);
3592    get_tty(tty);
3593    whoami(who);
3594    sprintf(fn,"/tmp/sa-%s-%2s",who,tty);
3595    fp=fopen(fn,"r");
3596    while(fgets(s,256,fp)!=NULL) {
3597      substr(s,1,15,fxx);
3598      trim(fxx);
3599      if (strcmp(key,fxx)==0) {
3600        substr(s,17,256,data);
3601        printf("%-s",data);
3602        exit(0);
3603      }
3604    }
3605    printf(" ");
3606  }
3607  ::::::::::::::
3608  menu.c
3609  ::::::::::::::
3610  /* menu.c:  This is a program for demo. the menu utility
3611  */
3612
3613  #include <stdio.h>
3614
3615  main(ac,av)
3616  char ac,**av;
3617  {
3618    char choi[5];
3619
3620    initscr();
3621
3622    while (toupper(choi[0])!='Q') {
3623      dspmain(av[1]);
3624      getchoi(22,32,1,choi);
3625    }
3626    endscr();
3627  }
3628
3629  dspmain(av1)
3630  char *av1;
3631  {
3632    clear();
3633    text("\r\n\
3634      \r\n\
3635      \r\n\
3636      \r\n\
3637      \r\n\
3638                          1.  MARKETING DEPARTMENT MENU\r\n\
3639                          2.  SHIPPING  DEPARTMENT MENU\r\n\
3640                          3.  FINANCIAL DEPARTMENT MENU\r\n\
```

```
3641                          4.  DOCUMENT  DEPARTMENT MENU\r\n\
3642                          5.  BACKUP DATA\r\n\
3643                          6.  SYSTEM UPDATE MENU\r\n\
3644                          Q.  QUIT\r\n\
3645     ");
3646     say(01,04,"ABC CO.,LTD.");
3647     say(03,28,"M A I N    M E N U");
3648     say(22,24,"SELECT:");
3649     if (av1[0]=='b' || av1[0]=='m')
3650       box("01-01-24-79-mc");
3651   }
3652   ::::::::::::::
3653   menu_bar.c
3654   ::::::::::::::
3655   /***
3656   * menu_bar.c: This is a program for handle the Menu bar
3657   *             Written by Mongkol Kuanhavet, Jul 1993
3658   */
3659   #include <stdio.h>
3660   #include <stdlib.h>
3661
3662   main(ac,av)
3663   int ac;
3664   char **av;
3665   {
3666     int  i;
3667     char sav[4096];
3668
3669     if (!(ac==2||ac>=5)) {
3670       fprintf(stderr,"usage:menu_bar [color1 color2 color3] text1 text2 ..\n");
3671       fprintf(stderr,"      color1 is a bar_color\n");
3672       fprintf(stderr,"      color2 is a backgroup color\n");
3673       fprintf(stderr,"      color3 is a message color(wait..)\n");
3674       fprintf(stderr,"exam:menu_bar \"$b_red\" \"$b_blue\" \"$b_green\" \"\\\n");
3675       fprintf(stderr,"       @ 10,20 say '1. Entry data'\\\n");
3676       fprintf(stderr,"       @ 12,20 say '2. Print report'\\\n");
3677       fprintf(stderr,"     \"\n");
3678       exit(1);
3679     }
3680     initscr();
3681     if (ac>=5) {
3682       strcpy(sav,av[4]);
3683       for (i=5;i<ac;i++)
3684         strcat(sav,av[i]);
3685       menu_bar(av[1],av[2],av[3],sav);
3686     } else
3687       menu_bar("7m","0m","0m",av[1]);
3688     endscr();
3689   }
3690   ::::::::::::::
3691   menu_key.c
3692   ::::::::::::::
3693   /***
3694   * monu_key.c : This is a program for get the Function key
3695   *              Written by Mongkol Kuanhavet, Jul 1993
3696   */
```

```
3697  #include <stdio.h>
3698  #include <stdlib.h>
3699
3700  main(ac,av)
3701  char ac,**av;
3702  {
3703    char s[256],tty[10],fn[30],who[20],funckey[5];
3704    FILE *fp;
3705
3706    get_tty(tty);
3707    whoami(who);
3708    sprintf(fn,"/tmp/me-%s-%2s",who,tty);
3709    if ((fp=fopen(fn,"r"))!=NULL) {
3710      fgets(s,256,fp);
3711      substr(s,4,4,funckey);
3712      printf("%s",funckey);
3713    } else
3714      printf(" ");
3715  }
3716  ::::::::::::::
3717  menu_to.c
3718  ::::::::::::::
3719  /***
3720  * monu_to.c : This is a program for get the menu choice no. for select
3721  *             Written by Mongkol Kuanhavet, Jul 1993
3722  */
3723  #include <stdio.h>
3724  #include <stdlib.h>
3725
3726  main(ac,av)
3727  char ac,**av;
3728  {
3729    char s[256],tty[10],fn[30],who[20];
3730    FILE *fp;
3731
3732    get_tty(tty);
3733    whoami(who);
3734    sprintf(fn,"/tmp/me-%s-%2s",who,tty);
3735    if ((fp=fopen(fn,"r"))!=NULL) {
3736      fgets(s,256,fp);
3737      printf("%1d",stoi(s,1,2));
3738    } else
3739      printf("00");
3740  }
3741  ::::::::::::::
3742  modify.c
3743  ::::::::::::::
3744  /* modify.c: This is a program for modify the the Database file
3745  *           Written by Mongkol kuanhavet, 18 Jun 92
3746  *           Computer center, Prince of Songkla Univ., Hathai, Thailand.
3747  *
3748  *     usage: modify fn
3749  */
3750  #include <stdio.h>
3751  #include <ctype.h>
3752  #include <signal.h>
```

```
3753  #include "cst_db.h"
3754
3755  main(ac,av)
3756  int  ac;
3757  char **av;
3758  {
3759    FILE *fp,*fpi,*fpo;
3760    char sdate[7],comm[200],c,fnabort[100],s[256];
3761    char fndbf1[100],fninx1[100],fnstc1[100];
3762    char fndbf2[100],fninx2[100],fnstc2[100];
3763
3764    if (ac!=2) {
3765      fprintf(stderr,"usage: modify fn\n");
3766      exit(0);
3767    }
3768    if (!isdbf(av[1]))
3769      error (av[1]," is not a database file");
3770
3771    sprintf(fndbf1,"%s"    ,av[1]);
3772    sprintf(fnstc1,"%s.stc",fndbf1);
3773    sprintf(fninx1,"%s.inx",fndbf1);
3774
3775    sprintf(fndbf2,"%s#modi#%06d",av[1],getpid());
3776    sprintf(fnstc2,"%s.stc",fndbf2);
3777    sprintf(fninx2,"%s.inx",fndbf2);
3778
3779    if ((fpi=fopen(fnstc1,"r"))==NULL)
3780      error ("can't open file",fnstc1);
3781    if ((fpo=fopen(fnstc2,"w"))==NULL)
3782      error ("can't create file",fnstc2);
3783    while (fgets(s,256,fpi)!=NULL) {
3784      if (s[0]=='#'||s[1]=='#')
3785        break;
3786      fprintf(fpo,"%s",s);
3787    }
3788    fclose(fpi);
3789    fclose(fpo);
3790
3791    sprintf(comm,"create %s -r %s",fndbf2,fndbf1);
3792    system(comm);
3793
3794    if (!isdbf(fndbf2))
3795      error ("** Error: Modify field name... Modify is abort: ",fndbf1);
3796
3797    fprintf(stderr,"File %s will be modify... Are you sure(Y/N)?",av[1]);
3798    c=getyn();
3799    fprintf(stderr,"\n");
3800    if (tolower(c)!='y') {
3801      sprintf(comm,"rm %s %s %s",fndbf2,fninx2,fnstc2);
3802      system(comm);
3803      exit(0);
3804    }
3805    fprintf(stderr,"waiting...");
3806
3807    sprintf(fnabort,"%s.@@@",fndbf2);
3808    fp=fopen(fnabort,"w");              /* flag for append */
```

```
3809    fclose(fp);
3810
3811    sprintf(comm,"append %s from %s",fndbf2,fndbf1);
3812    system(comm);
3813
3814    if (!file(fnabort)) {        /* fnabort removed when append complete */
3815      sprintf(comm,"mv %s %s",fndbf2,fndbf1);
3816      system(comm);
3817      sprintf(comm,"mv %s %s",fninx2,fninx1);
3818      system(comm);
3819      sprintf(comm,"mv %s %s",fnstc2,fnstc1);
3820      system(comm);
3821    } else {
3822      sprintf(comm,"rm %s %s %s %s",fndbf2,fninx2,fnstc2,fnabort);
3823      system(comm);
3824      error ("** Interrupt: Modify is abort: ",fndbf1);
3825    }
3826    fprintf(stderr,"\n");
3827  }
3828  ::::::::::::::
3829  pack.c
3830  ::::::::::::::
3831  /* pack.c: This is a program for modify the the Database file
3832  *          Written by Mongkol kuanhavet, 18 Jun 92
3833  *          Computer center, Prince of Songkla Univ., Hathai, Thailand.
3834  *
3835  *     usage: pack fn
3836  */
3837  #include <stdio.h>
3838  #include <ctype.h>
3839  #include <signal.h>
3840  #include "cst_db.h"
3841
3842  main(ac,av)
3843  int  ac;
3844  char **av;
3845  {
3846    FILE *fpdbf1,*fpdbf2[2];
3847    char sdate[7],comm[200],s[1025];
3848    char fndbf1[100],fndbf2[100];
3849    int  delcol;
3850
3851    if (ac!=2) {
3852      fprintf(stderr,"usage: pack fn\n");
3853      exit(0);
3854    }
3855    select(1);
3856    use(av[1],"r");
3857    delcol=getdelcol();
3858    closedata();
3859
3860    strcpy(fndbf1,av[1]);
3861    if ((fpdbf1=fopen(fndbf1,"r"))==NULL)
3862      error("can't open: ",fndbf1);
3863
3864    sprintf(fndbf2,"%s#pack#%06d",av[1],getpid());
```

```
3865    sprintf(comm,"copy %s to %s stru",fndbf1,fndbf2);
3866    system(comm);
3867
3868    if (fopeninx(fndbf2,"r+",fpdbf2)==NULL)
3869      error("can't open: ",fndbf2);
3870
3871    while (fgets(s,1025,fpdbf1)!=NULL) {
3872      if (s[delcol]==' ')
3873        fwriteinx(s,fpdbf2,"m");
3874    }
3875    sprintf(comm,"mv %s %s",fndbf2,fndbf1);
3876    system(comm);
3877    sprintf(comm,"mv %s.inx %s.inx",fndbf2,fndbf1);
3878    system(comm);
3879    sprintf(comm,"rm %s.stc",fndbf2);
3880    system(comm);
3881  }
3882  ::::::::::::::
3883  recall.c
3884  ::::::::::::::
3885  /* recall.c: This is a program for recall record of the database
3886  *           Written by Mongkol kuanhavet, 18 Jun 92
3887  *           Computer center, Prince of Songkla Univ.(PSU), Hatyai, Thailand.
3888  *           Under the VAX 11/785, ULTRIX Operating system v3.11
3889  *
3890  * compile: cc recall.c /staff/mongkol/bin/psurdbms /staff/mongkol/bin/libinx
3891  *
3892  * run:     a.out fn [scope]
3893  */
3894  #include <stdio.h>
3895  #include <math.h>
3896  #include "cst_db.h"
3897
3898  main(ac,av)
3899  int ac;
3900  char **av;
3901  {
3902    int  irec,brec=1,erec=9999999;
3903
3904    if (ac<2) {
3905      fprintf(stderr,"usage: recall fn [from_record to_record]\n");
3906      fprintf(stderr,"exam : recall name 5 20\n");
3907      exit(1);
3908    }
3909    select(1);                          /* Seleect work area #1 */
3910    use(av[1],"r+");                    /* Use or open the database file */
3911    gotop();                            /* Get the 1st record to memory var.*/
3912
3913    if (ac>2) {                         /* Check the scope for recall */
3914      brec=atoi(av[2]);
3915      if (ac>3)
3916        erec=atoi(av[3]);
3917      else
3918        erec=brec;
3919    }
3920    while (!eof()) {                    /* Get record until eof */
```

```
3921      irec=recno();
3922      if (irec>erec)                    /* If out of scope then exit */
3923        break;
3924      if (irec>=brec && isdelete())     /* Is a deleted record? */
3925        recall();                       /* Recall or unmark deleted flag */
3926      skip(1);                          /* Get next record */
3927    }
3928    closedata();                        /* Close database file */
3929  }
3930  ::::::::::::::
3931  reindex.c
3932  ::::::::::::::
3933  /* reindex.c: This is a program for reindex the Database file
3934  *            Written by Mongkol kuanhavet, 18 Jun 92
3935  *            Computer center, Prince of Songkla Univ., Hathai, Thailand.
3936  */
3937  #include <stdio.h>
3938
3939  main(ac,av)
3940  int  ac;
3941  char **av;
3942  {
3943    FILE *fpdbf1,*fpdbf2[2],*fpdbf2s;
3944    char sdate[7],comm[200],s[1025];
3945    char fndbf1[100],fndbf2[100],fndbf2s[100];
3946    int  reclen,delcol,keylen,keyoffs,trec,ntrec;
3947
3948    if (ac!=2) {
3949      fprintf(stderr,"usage: reindex fn\n");
3950      exit(0);
3951    }
3952    if (!isdbf(av[1])) {
3953      fprintf(stderr,"File '%s' Is not a database file\n",av[1]);
3954      exit(0);
3955    }
3956    strcpy(fndbf1,av[1]);
3957    select(1);
3958    use(fndbf1,"r");
3959    delcol=getdelcol();
3960    getkeyinfo(&reclen,&keylen,&keyoffs);
3961    trec=gettrec();
3962    closedata();
3963
3964    sprintf(fndbf2,"%s#rein#%06d",fndbf1,getpid());
3965
3966    sprintf(fndbf2s,"%s.s",fndbf2);
3967    sprintf(comm,"sort +0.%01d -0.%01d -o %s %s",keyoffs,keyoffs+keylen,
3968      fndbf2s,fndbf1);
3969    system(comm);
3970    if (!file(fndbf2s)) {
3971      fprintf(stderr,"** Interrupt: reindex is abort: %s\n",fndbf1);
3972      exit(-1);
3973    }
3974    sprintf(comm,"copy %s to %s stru",fndbf1,fndbf2);
3975    system(comm);
3976
```

```
3977    if ((fpdbf2s=fopen(fndbf2s,"r"))==NULL)
3978      error("can't open: ",fndbf2s);
3979    if (fopeninx(fndbf2,"r+",fpdbf2)==NULL)
3980      error("can't open: ",fndbf2);
3981
3982    ntrec=0;
3983    while (fgets(s,1025,fpdbf2s)!=NULL) {
3984      ntrec++;
3985      if (s[delcol]==' ')
3986        fwriteinx(s,fpdbf2,"m");
3987    }
3988    fclose(fpdbf2s);
3989    fcloseinx(fpdbf2);
3990    system("stty -raw");
3991    if (trec==ntrec) {
3992      sprintf(comm,"mv %s %s",fndbf2,fndbf1);
3993      system(comm);
3994      sprintf(comm,"mv %s.inx %s.inx",fndbf2,fndbf1);
3995      system(comm);
3996      sprintf(comm,"rm %s.stc %s.s",fndbf2,fndbf2);
3997      system(comm);
3998    } else {
3999      fprintf(stderr,"** Interrupt or data error: reindex abort: %s\n",fndbf1);
4000      sprintf(comm,"rm %s %s.inx %s.stc %s.s",fndbf2,fndbf2,fndbf2,fndbf2);
4001      system(comm);
4002    }
4003    endscr();
4004  }
4005  ::::::::::::::
4006  rename.c
4007  ::::::::::::::
4008  /* rename.c: This is a program for rename the the Database file
4009  *          Written by Mongkol kuanhavet, 18 Jun 92
4010  *          Computer center, Prince of Songkla Univ., Hathai, Thailand.
4011  *
4012  *   usage: rename fn1 to fn2
4013  */
4014  #include <stdio.h>
4015
4016  main(ac,av)
4017  int  ac;
4018  char **av;
4019  {
4020    char  comm[256];
4021
4022    if (ac!=4 || strcmp(av[2],"to")!=0) {
4023      fprintf(stderr,"usage: rename fn1 to fn2\n");
4024      exit(0);
4025    }
4026    if (!isdbf(av[1])) {
4027      fprintf(stderr,"File '%s' is not database file\n",av[1]);
4028      exit(0);
4029    }
4030    sprintf(comm,"mv %s.stc %s.stc",av[1],av[3]);
4031    system(comm);
4032    sprintf(comm,"mv %s %s",av[1],av[3]);
```

```
4033    system(comm);
4034    sprintf(comm,"mv %s.inx %s.inx",av[1],av[3]);
4035    system(comm);
4036  }
4037  ::::::::::::::
4038  replace.c
4039  ::::::::::::::
4040  /* replace.c: This is a program for replace field the database
4041  *            Written by Mongkol kuanhavet, 18 Jun 92
4042  */
4043  #include <stdio.h>
4044  #include <math.h>
4045  #include "cst_db.h"
4046
4047  main(ac,av)
4048  int ac;
4049  char **av;
4050  {
4051    int  i,j;
4052    char *sx[MAXFIELD];
4053    double fx[MAXFIELD];
4054
4055    if (ac<5) {
4056      fprintf(stderr,"usage: replace fn {(field_name data type)}\n");
4057      fprintf(stderr,"exam : replace name fac_code 200 c salary 3500 n\n");
4058      exit(1);
4059    }
4060    select(1);
4061    use(av[1],"r+");
4062
4063    for (i=2,j=0;i<ac;i+=3,j++) {
4064      if (av[i+2][0]=='c')
4065        mapchar(av[i],&sx[j]);
4066      else if (av[i+2][0]=='n')
4067        mapreal(av[i],&fx[j]);
4068      else
4069        error("** Error: Field name or type: ",av[i+2]);
4070    }
4071    gotop();
4072
4073    while (!eof()) {
4074      for (i=2;i<ac;i+=3) {
4075        if (av[i+2][0]=='c')
4076          replchar(av[i],av[i+1]);         /* replace data field for char.*/
4077        else
4078          replreal(av[i],atof(av[i+1]));   /* replace data field for Num.  */
4079      }
4080      replace();                           /* replace data record */
4081      skip(1);
4082    }
4083    closedata();
4084  }
4085  ::::::::::::::
4086  sam1.c
4087  ::::::::::::::
4088  /****
```

```
4089  * saml.c: This is a example program #1 for calling the database routine
4090  *         Written by Mongkol kuanhavet, 18 Jun 92
4091  *         Computer center, Prince of Songkla Univ.(PSU), Hatyai, Thailand.
4092  *         Under the VAX 11/785, ULTRIX Operating system v3.11
4093  *
4094  * compile: cc saml.c /staff/mongkol/bin/psurdbms /staff/mongkol/bin/libinx
4095  *
4096  * run:     a.out /staff/mongkol/bin/emp /staff/mongkol/bin/dept
4097  */
4098  #include <stdio.h>
4099
4100  /****
4101  * Define the constant name for refer to the field name in the database file.
4102  */
4103  #define  EMPNO     "empno"
4104  #define  NAME      "name"
4105  #define  SNAME     "sname"
4106  #define  SALARY    "salary"
4107  #define  DEPTNO    "deptno"
4108  #define  DEPTNAME  "deptname"
4109
4110  /****
4111  /* Declare the variable name for store the data from a field in database file.
4112  *  The first character (A_ and B_) are show the useing work area 1 and 2.
4113  */
4114  char  *A_empno;
4115  char  *A_deptno;
4116  char  *A_name;
4117  char  *A_sname;
4118  double A_salary;
4119
4120  char  *B_deptno;
4121  char  *B_deptname;
4122
4123  /****
4124  *  Map_A() is the function for mapping the field name with variable name
4125  *          in the work area 1.
4126  */
4127  map_A()
4128  {
4129    mapchar(EMPNO,   &A_empno);
4130    mapchar(NAME,    &A_name);
4131    mapchar(SNAME,   &A_sname);
4132    mapchar(DEPTNO,  &A_deptno);
4133    mapreal(SALARY,  &A_salary);
4134  }
4135
4136  /****
4137  *  Map_B() is the function for mapping the field name with variable name
4138  *          in the work area 2.
4139  */
4140  map_B()
4141  {
4142    mapchar(DEPTNO,  &B_deptno);
4143    mapchar(DEPTNAME,&B_deptname);
4144  }
```

```
4145
4146  /****
4147  * This is a main program sam1.c
4148  */
4149  main(ac,av)
4150  int ac;
4151  char **av;
4152  {
4153    char pdeptno[10];
4154
4155    if (ac!=3)
4156      error("usage: sam1 emp dept"," ");
4157
4158    select(1);                          /* select the work area #1 before use*/
4159    use(av[1],"r");                     /* open the database file 'emp' */
4160    map_A();
4161    gotop();                            /* store data from 1st record to var. */
4162
4163    select(2);                          /* select the work area #2 before use*/
4164    use(av[2],"r");                     /* open the database file 'dept' */
4165    map_B();
4166    gotop();
4167
4168    select(1);
4169    while (!eof()) {                    /* get database file #1 until eof */
4170      if (strcmp(pdeptno,A_deptno)!=0){ /* the deptno is change ? */
4171        select(2);
4172        seekdbf(A_deptno);              /* seek the dept file with deptno*/
4173        if (found())                    /* if found print the dept name */
4174          printf("\nDept:%s %s\n",B_deptno,B_deptname);
4175        else
4176          printf("\nDept:%s Not found in dept\n",A_deptno);
4177        strcpy(pdeptno,A_deptno);
4178      }                                 /* print the detail from file emp */
4179      printf("    %s %s %s %7.0f\n",A_empno,A_name,A_sname,A_salary);
4180      select(1);                        /* select work area #1 */
4181      skip(1);                          /* skip one record of file emp */
4182    }
4183    closedata();                        /* close all the database file */
4184  }
4185  ::::::::::::::
4186  sam8.c
4187  ::::::::::::::
4188  #include <stdio.h>
4189  #include <ctype.h>
4190  #include <signal.h>
4191  #include <math.h>
4192  #include "cst_db.h"
4193
4194  double atof();
4195  char   *passwd;
4196  char   *mempno;
4197  char   *mname;
4198  char   *msname;
4199  char   *mdeptno;
4200  char   *mdeptname;
```

```
4201  double  msalary;
4202  char   *mstatus;
4203
4204  char   *A_empno;
4205  char   *A_name;
4206  char   *A_deptno;
4207  char   *A_sname;
4208  double  A_salary;
4209
4210  char   *B_deptno;
4211  char   *B_deptname;
4212
4213  define_var()
4214  {
4215    defchar("passwd"   ,&passwd);
4216    defchar("mempno"   ,&mempno);
4217    defchar("mname"    ,&mname);
4218    defchar("msname"   ,&msname);
4219    defchar("mdeptno"  ,&mdeptno);
4220    defchar("mdeptname",&mdeptname);
4221    defreal("msalary"  ,&msalary);
4222    defchar("mstatus"  ,&mstatus);
4223  }
4224
4225  map_A()
4226  {
4227    mapchar("empno" ,&A_empno);
4228    mapchar("name"  ,&A_name);
4229    mapchar("sname" ,&A_sname);
4230    mapchar("deptno",&A_deptno);
4231    mapreal("salary",&A_salary);
4232  }
4233
4234  map_B()
4235  {
4236    mapchar("deptno"  ,&B_deptno);
4237    mapchar("deptname",&B_deptname);
4238  }
4239
4240  main(ac,av)
4241  int ac;
4242  char **av;
4243  {
4244    char tmp[30];
4245    int  ret;
4246
4247    if (ac!=3)
4248      error("usage: sam8 emp dept"," ");
4249    if (!isdbf(av[1]))
4250      error(av[1]," is not database file");
4251    if (!isdbf(av[2]))
4252      error(av[2]," is not database file");
4253
4254    select(1);
4255    use(av[1],"r+");
4256    map_A();
```

```
4257    gotop();
4258
4259    select(2);
4260    use(av[2],"r");
4261    map_B();
4262    gotop();
4263
4264    define_var();
4265    initscr();
4266    keypasswd();
4267
4268    dspmenu();
4269    while (1) {
4270      dspspace();
4271      ret=sayget(" \
4272        @ 03,10 get mempno  pict '9999999' \
4273      ");
4274      if (ret==CTRL_K||ret==CTRL_C)
4275        break;
4276      select(1);
4277      seekdbf(mempno);
4278      if (found()) {
4279        setcolor("/W");
4280        say(03,20,"OLD REC");
4281        setcolor("");
4282        mname=A_name;
4283        msname=A_sname;
4284        mdeptno=A_deptno;
4285        msalary=A_salary;
4286      } else {
4287        setcolor("/W");
4288        say(03,20,"NEW REC");
4289        setcolor("");
4290        mname=" ";
4291        msname=" ";
4292        mdeptno=" ";
4293        msalary=0;
4294      }
4295      select(2);
4296      seekdbf(mdeptno);
4297      if (found())
4298        say(9,16,B_deptname);
4299      else
4300        say(9,16,"*** Dept not found");
4301
4302      setcolor("W+");
4303      ret=sayget(" \
4304        @ 05,10 get mname   pict '!!!!!!!!!!!!' \
4305        @ 07,10 get msname  pict 'ccccccccccccccc' \
4306        @ 09,10 get mdeptno pict '999' \
4307        @ 11,10 get msalary pict '9999999.99' range 500,90000 \
4308        @ 13,10 get mstatus pict '!' valid 'YN' \
4309      ");
4310      setcolor("");
4311      if (ret==CTRL_K||ret==CTRL_C)
4312        break;
```

```
4313      else if (ret==CTRL_E)
4314        continue;
4315
4316      select(1);
4317      seekdbf(mempno);
4318      if (!found()) {
4319        replchar("empno",mempno);
4320        append();
4321      }
4322      replchar("name",mname);
4323      replchar("sname",msname);
4324      replchar("deptno",mdeptno);
4325      replreal("salary",msalary);
4326      replace();
4327    }
4328    goyx(23,0);
4329    endscr();
4330    closedata();
4331  }
4332
4333  dspmenu()
4334  {
4335    clear();
4336    text("\
4337                              \r\n\
4338                              \r\n\
4339    EmpNo [       ]           \r\n\
4340                              \r\n\
4341    Name  [           ]       \r\n\
4342                              \r\n\
4343    Sname [              ]    \r\n\
4344                              \r\n\
4345    Dept  [   ]               \r\n\
4346                              \r\n\
4347    Salary[          ]        \r\n\
4348                              \r\n\
4349    Status[ ]                 \r\n\
4350    ");
4351  }
4352
4353  dspspace()
4354  {
4355    char sp[100];
4356    int i;
4357
4358    for(i=0;i<40;i++)
4359      sp[i]=' ';
4360    sp[i]=NULL;
4361    say( 3,20,"       ");
4362    say( 3,10,"       ");
4363    say( 5,10,"            ");
4364    say( 7,10,"              ");
4365    say( 9,10,"   ");
4366    say( 9,16,sp);
4367    say(11,10,"         ");
4368    say(13,10," ");
```

```
4369  }
4370
4371  keypasswd()
4372  {
4373    clear();
4374    setconfirm(1);
4375    setbell(1);
4376    setcolor("W+");
4377    say(10,30,"Passwd?:");
4378    setcolor("");
4379    setcolor("/W");
4380    setcolor("W+/N,X");
4381    sayget("\
4382      @ 10,38 get passwd pict '!!!!!!!!!!' \
4383    ");
4384    setcolor("");
4385    setconfirm(0);
4386    setbell(0);
4387  }
4388  ::::::::::::::
4389  say.c
4390  ::::::::::::::
4391  #include <stdio.h>
4392  #include <stdlib.h>
4393
4394  main(ac,av)
4395  int ac;
4396  char **av;
4397  {
4398    int  n,i;
4399    char s1[3],s2[256];
4400
4401    if (ac!=4) {
4402      fprintf(stderr,"usage:say y x mesg\n");
4403      fprintf(stderr,"exam :say 10 40 \"Main Menu\"\n");
4404      exit(0);
4405    }
4406    if (av[3][1]==',') {
4407      sprintf(s1,"%c",av[3][0]);
4408      n=stoi(av[3],3,2);
4409      for(i=0;i<n;i++)
4410        s2[i]=' ';
4411      s2[i]=NULL;
4412      say(atoi(av[1]),atoi(av[2]),s2);
4413    } else
4414      say(atoi(av[1]),atoi(av[2]),av[3]);
4415  }
```
