```
   Help
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2008+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else
/**********************************************************
    *****************/
/*                                errrhandl.c
                     */
/**********************************************************
    *****************/
/*
                     */
/* ERRor HANDLing routines
                     */
/*
                     */
/* Copyright (C) 1992-1995 Tomas Skalicky. All rights res
    erved.              */
/*
                     */
/**********************************************************
    *****************/
/*
                     */
/*        ANY USE OF THIS CODE CONSTITUTES ACCEPTANCE OF TH
    E TERMS          */
/*              OF THE COPYRIGHT NOTICE (SEE FILE copyrght.h
    )               */
/*
                     */
/**********************************************************
    *****************/

#include <stdlib.h>
#include <string.h>

#include "laspack/errhandl.h"
#include "laspack/copyrght.h"

/* LASPack error status, procedure and objects where an
```

```
    error has ariced */
static LASErrIdType LASErrId = LASOK;
static char *LASProcName = NULL;
static char *LASObject1Name = NULL;
static char *LASObject2Name = NULL;
static char *LASObject3Name = NULL;

void LASError(LASErrIdType ErrId, char *ProcName, char *Ob
    ject1Name,
        char *Object2Name, char *Object3Name)
/* Set error status to ErrId, ... */
{
    LASErrId = ErrId;

    /* release current values of error variables */
    if (LASProcName != NULL)
        free(LASProcName);
    if (LASObject1Name != NULL)
        free(LASObject1Name);
    if (LASObject2Name != NULL)
        free(LASObject2Name);
    if (LASObject3Name != NULL)
        free(LASObject2Name);
    LASProcName = NULL;
    LASObject1Name = NULL;
    LASObject2Name = NULL;
    LASObject3Name = NULL;

    LASProcName = (char *)malloc((strlen(ProcName) + 1) *
    sizeof(char));
    if (LASProcName != NULL) {
        strcpy(LASProcName, ProcName);

  if (Object1Name != NULL && strlen(Object1Name) > 0) {
          LASObject1Name = (char *)malloc((strlen(Object1
    Name) + 1) * sizeof(char));
          if (LASObject1Name != NULL)
              strcpy(LASObject1Name, Object1Name);
  }
  if (Object2Name != NULL && strlen(Object2Name) > 0) {
          LASObject2Name = (char *)malloc((strlen(Object2
    Name) + 1) * sizeof(char));
            if (LASObject2Name != NULL)
                strcpy(LASObject2Name, Object2Name);
  }
  if (Object3Name != NULL && strlen(Object3Name) > 0) {
            LASObject3Name = (char *)malloc((strlen(Object3
    Name) + 1) * sizeof(char));
            if (LASObject3Name != NULL)
                strcpy(LASObject3Name, Object3Name);
  }
    } else {
        strcpy(LASProcName, "(procedure unknown)");
    }

}

void LASBreak(void)
/* user break */
{
    LASErrId = LASUserBreak;
}

LASErrIdType LASResult(void)
/* get result of linear algebra operations */
{
    return(LASErrId);
}

void WriteLASErrDescr(FILE *File)
/* write a short description of the reason caused break of
    LASPack */
{
    int NoPrintedObj;

    if (LASErrId != LASOK) {
  if (LASProcName != NULL) {
            fprintf(File, "in %s", LASProcName);
      NoPrintedObj = 0;
      if (LASObject1Name != NULL || LASObject2Name != NULL
    || LASObject3Name != NULL)
                fprintf(File, " for ");
      if (LASObject1Name != NULL) {
          if (NoPrintedObj > 0)
                    fprintf(File, ", ");
                fprintf(File, "%s", LASObject1Name);
    NoPrintedObj++;
      }
      if (LASObject2Name != NULL) {
          if (NoPrintedObj > 0)
                    fprintf(File, ", ");
                fprintf(File, "%s", LASObject2Name);
    NoPrintedObj++;
      }
      if (LASObject3Name != NULL) {
          if (NoPrintedObj > 0)
                    fprintf(File, ", ");
                fprintf(File, "%s", LASObject3Name);
    NoPrintedObj++;
      }
            fprintf(File, ":");
        }
        fprintf(File, "{n");
    }

    switch (LASErrId) {
        case LASOK:
            break;
        case LASMemAllocErr:
      fprintf(File, "Not enough memory is available.{n");
            break;
        case LASLValErr:
      fprintf(File, "L-value parameter is expected.{n");
            break;
        case LASDimErr:
      fprintf(File, "Objects have incompatible dimensions.
    {n");
            break;
        case LASRangeErr:
      fprintf(File, "Indices are out of range.{n");
            break;
        case LASSymStorErr:
      fprintf(File, "Some elements are stored in lower tri
        angular part");
          fprintf(File, " of a symmetric matrix.{n");
                break;
            case LASMatrCombErr:
          fprintf(File, "Matrices can not be combined.{n");
                break;
            case LASMulInvErr:
          fprintf(File, "Inverse multiplication can not be
        carried out.{n");
                break;
            case LASElNotSortedErr:
          fprintf(File, "Matrix elements are not sorted.{n");
                break;
            case LASZeroInDiagErr:
          fprintf(File, "Zero elements in matrix diagonal are
        not allowed.{n");
                break;
            case LASZeroPivotErr:
          fprintf(File, "Factorization produces zero pivot ele
        ments.{n");
                break;
            case LASILUStructErr:
          fprintf(File, "Matrix has structure which is not all
        owed for ILU factorization.{n");
                break;
            case LASBreakdownErr:
          fprintf(File, "Iterative solver fails.{n");
                break;
            case LASUserBreak:
          fprintf(File, "Termination by an user break.{n");
                break;
        }
    }

    #endif //PremiaCurrentVersion
```

References