Matlab与C混合编程API之mxCreateStructMatrix

zcl.space

目录

1 引言

在matlab中创建和操作结构体非常便捷，mathwork公司把这种便捷延伸到了桥梁函数中。matlab为桥梁函数定义了创建结构体的API: `mxCreateStructMatrix` 。

2 调用语法

`mxCreateStructMatrix` 的调用语法如下

```
#include "matrix.h"
mxArray *mxCreateStructMatrix(mwSize m, mwSize n,
     int nfields, const char **fieldnames);
```

输入参数表如下：

表 1: `mxCreateStructMatrix` 输入参数对照表

参数名	描述
m	结构体矩阵的行数
n	结构体矩阵的列数
nfields	结构体矩阵中每个结构体域的个数
fieldnames	结构体矩阵中每个结构体的域名

`mxCreateStructMatrix` 的返回值是一个指向 `mxArray` 的指针。我们还是通过一个例子来说明 `mxCreateStructMatrix` 的使用。

3 一个例子

这个例子实现了电话本功能，是目前为止用到的最复杂的例子。

```
#include "mex.h"
#include "string.h"

#define MAXCHARS 80   /* max length of string contained in
```

```
                                         each field */

/*  the gateway routine.  */
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )
{
     /* pointers to field names */
    const char **fnames;
    const mwSize *dims;
    mxArray    *tmp, *fout;
    char       *pdata=NULL;
    int        ifield, nfields;
    mxClassID  *classIDflags;
    mwIndex    jstruct;
    mwSize     NStructElems;
    mwSize     ndim;

    /* check proper input and output */
    if(nrhs!=1)
        mexErrMsgIdAndTxt(
                "MATLAB:phonebook:invalidNumInputs",
                "One␣input␣required.");
    else if(nlhs > 1)
        mexErrMsgIdAndTxt(
                "MATLAB:phonebook:maxlhs",
                "Too␣many␣output␣arguments.");
    else if(!mxIsStruct(prhs[0]))
        mexErrMsgIdAndTxt(
                "MATLAB:phonebook:inputNotStruct",
                "Input␣must␣be␣a␣structure.");
    /* get input arguments */
    nfields = mxGetNumberOfFields(prhs[0]);
    NStructElems = mxGetNumberOfElements(prhs[0]);
    /* allocate memory  for storing classIDflags */
    classIDflags = mxCalloc(nfields, sizeof(mxClassID));

    /* check empty field, proper data type,
     * and data type consistency;
     * and get classID for each field. */
    for(ifield=0; ifield<nfields; ifield++) {
        for(jstruct = 0; jstruct < NStructElems; jstruct++) {
            tmp = mxGetFieldByNumber(prhs[0], jstruct, ifield);
            if(tmp == NULL) {
                mexPrintf("%s%d\t%s%d\n", "FIELD:␣",
                           ifield+1, "STRUCT␣INDEX␣:", jstruct+1);
                mexErrMsgIdAndTxt( "MATLAB:phonebook:fieldEmpty",
                        "Above␣field␣is␣empty!");
            }
            if(jstruct==0) {
                if( (!mxIsChar(tmp) && !mxIsNumeric(tmp))
                     || mxIsSparse(tmp)) {
                    mexPrintf("%s%d\t%s%d\n", "FIELD:␣", ifield+1,
                              "STRUCT␣INDEX␣:", jstruct+1);
                    mexErrMsgIdAndTxt( "MATLAB:phonebook:invalidField",
                            "Above␣field␣must␣have␣either␣string␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣numeric␣non-sparse␣data.");
                }
                classIDflags[ifield]=mxGetClassID(tmp);
            } else {
                if (mxGetClassID(tmp) != classIDflags[ifield]) {
                    mexPrintf("%s%d\t%s%d\n", "FIELD:␣", ifield+1,
                    "STRUCT␣INDEX␣:", jstruct+1);
                    mexErrMsgIdAndTxt( "MATLAB:phonebook:invalidFieldType",
                            "Inconsistent␣data␣type␣in␣above␣field!");
                } else if(!mxIsChar(tmp) &&
                        ((mxIsComplex(tmp) || mxGetNumberOfElements(tmp)!=1))){
                    mexPrintf("%s%d\t%s%d\n", "FIELD:␣", ifield+1,
                    "STRUCT␣INDEX␣:", jstruct+1);
                    mexErrMsgIdAndTxt( "MATLAB:phonebook:fieldNotRealScalar",
                                    "Numeric␣data␣in␣above␣field␣must␣be␣scalar
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣and␣noncomplex!");
                    }
                }
            }
        }

    /* allocate memory  for storing pointers */
    fnames = mxCalloc(nfields, sizeof(*fnames));
    /* get field name pointers */
    for (ifield=0; ifield< nfields; ifield++){
        fnames[ifield] = mxGetFieldNameByNumber(prhs[0],ifield);
    }
    /* create a 1x1 struct matrix for output  */
    plhs[0] = mxCreateStructMatrix(1, 1, nfields, fnames);
    mxFree((void *)fnames);
    ndim = mxGetNumberOfDimensions(prhs[0]);
    dims = mxGetDimensions(prhs[0]);
    for(ifield=0; ifield<nfields; ifield++) {
        /* create cell/numeric array */
        if(classIDflags[ifield] == mxCHAR_CLASS) {
            fout = mxCreateCellArray(ndim, dims);
        }else {
            fout = mxCreateNumericArray(ndim, dims,
                       classIDflags[ifield], mxREAL);
            pdata = mxGetData(fout);
        }
        /* copy data from input structure array */
        for (jstruct=0; jstruct<NStructElems; jstruct++) {
            tmp = mxGetFieldByNumber(prhs[0],jstruct,ifield);
            if( mxIsChar(tmp)) {
                mxSetCell(fout, jstruct, mxDuplicateArray(tmp));
            }else {
                mwSize     sizebuf;
                sizebuf = mxGetElementSize(tmp);
                memcpy(pdata, mxGetData(tmp), sizebuf);
                pdata += sizebuf;
            }
        }
        /* set each field in output structure */
        mxSetFieldByNumber(plhs[0], 0, ifield, fout);
    }
    mxFree(classIDflags);
    return;
}
```

代码中使用 `mxGetNumOfFields` 获得输入的结构体矩阵中每个结构体域的个数，使用 `mxGetNumberOfElements` 来获得输入结构体矩阵中结构体的个数。

```
plhs[0] = mxCreateStructMatrix(1, 1, nfields, fnames);
```

创建一个结构体保存输出。

```
mxSetFieldByNumber(plhs[0], 0, ifield, fout);
```

为输出结构体的每个域赋值。

假设我们的输入是

```
a.a =1;
a.b =4;
a.c = 'hello world'
```

调用 phonebook

```
n=phonebook(a);
```

则 n 的值为：

```
n.a =1;
n.b =4;
n.c ='hello world'
```

4 尾声

mxCreateStructMatrix 是matlab中创建结构体的API，方便了matlab和C之间结构体类型数据的传递。