

## MEX-filer

Kombinera Matlab med C/C++

Beräkningsvetenskap DV, 5.0 hp

### MEX-fil

mex1.c

#### Filinnehåll (mex1.c)

```
#include "mex.h"

void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] ) {
    ...
}
```

#### Kompilera

```
>> mex mex1.c
```

#### Kör

```
>> mex1
>>
```

2

### MEX-fil

mex2.c

```
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[] )

nlhs  antal utdata (number of left-hand-sides)
plhs  utdata (pointer to left-hand-sides)
nrhs  antal inparametrar (number of right-hand-sides)
prhs  inparametrar (pointer to right-hand-sides)
```

3

### Kontroll av inparametrar

mex3.c

#### All data i Matlab har typen mxArray

- ▶ Skalärer, vektorer, matriser, textsträngar, structs, etc

#### Funktioner för att ta reda på datatyp och storlek

```
bool mxIsDouble(const mxArray *pm) // kolla datatyp
size_t mxGetM(const mxArray *pm) // antal rader
size_t mxGetN(const mxArray *pm) // antal kolonner
size_t mxGetNumberOfElements(array) // antal element
```

#### Skriv ut meddelanden

```
mexPrintf(const char *message, ...)
mexErrMsgTxt(const char *message) // avbryter körningen
```

4

### Läsa data från mxArray

mex4.c

Läsa från mxArray

```
double *mxGetPr(const mxArray *p) // pekare till data
double mxGetScalar(const mxArray *pm) // hämta skalär
```

#### Data lagras i Column Major-format

$$A = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

lagras i minnet som

[ 1 2 3 4 5 6 7 8 9 ]

### Dokumentation

5

#### Inbyggd hjälp i Matlab

- ▶ User's Guide → C/C++ and Fortran API Reference.
- ▶ Advanced Software Development → External Programming Language Interfaces → Application Programming Interfaces to MATLAB

#### Slå upp en mx- eller mex-funktion i Matlab

```
>> doc mexFree
```

6

## Skapa mxArray

mex5.c

```
Skapa mxArray
mxArray *mxCreateDoubleMatrix( mwSize m, mwSize n,
                                mxComplexity ComplexFlag )
mxArray *mxCreateDoubleScalar( double value )
```

### Exempel: Skapa matris med 3 rader, 4 kolumner

```
int nRows = 3;
int nCols = 4;
mxArray *a = mxCreateDoubleMatrix(nRows, nCols, mxREAL);

double *data = mxGetPr(a);
for (c = 0; c < nCols; c++)
    for (r = 0; r < nRows; r++)
        data[c*nRows+r] = r;
```

## Minneshantering

### Frigöra mxArray

```
void mxDestroyArray ( mxArray * pm ) // frigör minne
```

### Regler

- ▶ Matlab frigör minne automatiskt
- ▶ Det rekommenderas att man gör det själv
- ▶ Utdata ska inte frigöras

## Minneshantering

mex6.c

### Regler

- ▶ Använd inte malloc() eller calloc()
- ▶ Använd istället
  - void \*mxMalloc(mwSize n)
  - void \*mxCalloc(mwSize n, mwSize size)
- ▶ Frigör minnet med
  - void mxFree(void \*ptr)(Behövs egentligen inte, men rekommenderas)

## Anropa Matlab från en MEX-fil

mex7.c  
mex8.c

### Kör Matlab-kommando från en MEX-fil

```
int mexEvalString(const char *command);
```

### Alternativ (med mer kontroll)

```
int mexCallMATLAB(int nlhs, mxArray *plhs[],
                   int nrhs, mxArray *prhs[],
                   const char *functionName);
```

## Anropa Matlab från en MEX-fil

mex9wrap.m  
mex10.c  
mex11.c

### Åtkomst till Matlab-variabler

```
// Kopiera en variabel från Matlab
mxArray *mexGetVariable(const char *workspace,
                        const char *varname);

// Hämta pekare till variabel i Matlab
const mxArray *mexGetVariablePtr(const char *workspace,
                                 const char *varname);

// Skapa/ändra variabel i Matlab
int mexPutVariable(const char *workspace,
                   const char *varname,
                   const mxArray *pm);
```

### Där workspace är någon av

- ▶ base
- ▶ caller
- ▶ global

## Bevara data mellan anrop

mex12.c  
mex13.c

- ▶ Statiska variabler ligger kvar mellan anrop
- ▶ Matlab-arrayer frigörs när MEX-filen har kört klart
- ▶ Statiska Matlab-arrayer:
  - // Säg åt Matlab att inte frigöra en variabel
void mexMakeArrayPersistent( mxArray \*pm );
- ▶ Men alla variabler måste frigöras någon gång
  - // Säg åt Matlab att anropa exitfunction vid rensning
int mexAtExit( exitfunction );

### Rensa ut MEX-filer

```
>> clear mex
```

11

12

## Vanliga misstag

### ► Fel funktionsnamn

```
#include "mex.h"

void mexFunction (int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[]) {
    plhs[0] = createDoubleScalar(123);
}
```

Ger kompileringsfelet

```
mexerr1.c: In function 'mexFunction':
mexerr1.c:6: warning: assignment makes pointer from integer without a cast
mexerr1.o: In function 'mexFunction':
mexerr1.c:(.text+0x17): undefined reference to 'createDoubleScalar'
```

- Glömma att kompilera MEX-filen
- Indexera arrayer fel
- Försöka använda mxArray som int / double / double \* / etc

## Textsträngar

### Ta emot textsträng

```
int mxGetString(const mxArray *pm,
                char *str, mwSize strlen);
```

- Returnerar 0 när allt gick bra
- strlen är buffertstorleken, behöver plats för nollterminering
- Om strängen inte får plats trunkeras den och returvärdet är 1

### Skapa textsträng

```
mxArray *mxCreateString(const char *str);
```

mex14.c

## Komplexa data

mex15.c

### Kolla om data är komplex

```
bool mxIsComplex(const mxArray *pm);
```

### Hämta pekare till komplex del

```
double *mxGetPi(const mxArray *pm);
```

- Både mxIsDouble och mxIsComplex returnerar true
- Om mxIsComplex är false returnerar mxGetPi null

15