سرعت بخشیدن به اجرای الگوریتم‌ها در متلب

مقدمه

متلب محیط کامل و جامعی را برای برنامه‌نویسی در اختیار قرار می‌دهد که باعث سرعت بخشیدن به برنامه‌نویسی می‌شود اما ذات اسکریپتی بودن این زبان باعث می‌شود تا در مقایسه با زبان‌هایی چون C، C++ و Fortran سرعت اجرای پایین‌تری داشته باشد. یکی از راهکارها برای افزایش سرعت اجرای برنامه‌ها در متلب یافتن گلوگاه‌های محاسباتی الگوریتم، بازنویسی آن با زبان‌های C، C++ و Fortran و متصل کردن قسمت بازنویسی شده به برنامه موجود در متلب است. رابطی که امکان اتصال برنامه‌ها و داده‌های خارجی را به متلب می‌دهد MEX نام دارد که مخفف MATLAB executable است و زبان‌های C، C++و Fortranرا می‌پذیرد.

معرفی فایل‌های

فایل‌های MEX زیرروال‌هایی هستند که از کدهایی به زبان‌های C، C++ و Fortran ساخته شده‌اند اما با تغییرات مختصری که در ساختار آن‌ها ایجاد شده است امکان استفاده از آن‌ها در متلب نیز فراهم شده است. با بازنویسی قسمت‌هایی از برنامه که گلوگاه محاسباتی هستند، در قالب یک فایل MEX بهبود قابل توجه‌ای در کارایی برنامه ایجاد می‌شود.

فایل‌های MEX براساس این‌که در چه سیستمی کامپایل شده باشند، پسوندهای متفاوتی پیدا می‌کنند که این پسوندها به همراه نوع سیستم عامل در جدول زیر آمده است.

سیستم عامل	پسوند فایل
گنو/لینوکس	mex.glx
مک او اس	mexmac
ویندوز	dll

متلب بشکل پیش فرض شامل یک کامپایلر با نام Lcc برای کامپایل کدهای C و C++ است و بشکل خودکار از این کامپایلر برای ایجاد فایل‌های MEX استفاده می‌کند. در صورتی که نیاز به کامپایل برنامه‌های Fortran دارید، باید یک کامپایلر Fortran بر روی سیستم خود نصب کنید. البته در صورتی که چندین کامپایلر برای یک زبان بر روی سیستم خود نصب کرده‌اید با استفاده از دستور زیر کامپایلر دلخواه برای استفاده را انتخاب کنید.

```
Mex -setup
```

ساختار یک فایل MEX

هر فایل MEX باید شامل دو قسمت زیر باشد.

1. . فایل سرآیند mex.h : با افزودن این فایل سرآیند در ابتدای برنامه تمام نیازمندی‌ها برای کامپایلرهای مختلف فراهم می‌شود.

```
#include <mex.h>
```

2. تابع دروازه ورود : این تابع اصلی‌ترین قسمت در هر فایل MEX است که با استفاده از آن ارتباط با دیگر توابع متلب برقرار می‌شود و شکل کلی آن به صورت زیر است.

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

}
```

اجزای تابع دروازه ورود

1. *nlhs*: تعداد *mxArray* های مورد انتظار؛
2. *plhs*: آرایه‌ای از اشاره‌گرها به خروجی‌های مورد انتظار؛
3. *nrhs*: تعداد آرگومان‌های ورودی؛
4. *prhs*: آرایه‌ای از اشاره‌گرها به داده ورودی. این داده‌ها فقط خواندی هستند.

پس از ایجاد فایل MEX، یک تابع هم‌نام با فایل MEX ایجاد می‌شود که همانند توابع دیگر متلب دارای ورودی و خروجی است. به عنوان مثال اگر نام فایل MEX ایجاد شده hello.c باشد، تابعی با نام hello به متلب افزوده می‌شود که دارای ورودی و خروجی‌هایی به صورت زیر است.

```
[C, D]= hello(A,B)
```

در این‌جا A و B ورودی‌های تابع و C و D خروجی‌های تابع هستند که تعداد ورودی‌ها و خرجی‌ها برابر با ۲ است.

مثالی از بازنویسی کد برنامه به زبان C++ و ایجاد فایل MEX آن

برنامه ۱ به زبان C++ نوشته شده است و دو عدد j , i را از ورودی دریافت می‌کند و نتیجه ضرب آن‌ها را در متغییر z ذخیره و سپس به خروجی می‌فرستد.

```
1   #include <iostream>
2   int mul (int a, int b){
3   int c=a*b;
4   return c;
5   }
6
7   int main(){
```

```
8   int i=0;
9   int j=0;
10  std::cin>>i;
11  std::cin>>j;
12  int z=mul(i,j);
13  std::cout<<"z="<<z<<std::endl;
14  return 0;
15  }
```

قدم اول در تبدیل برنامه ۱ به فایل MEX، افزودن فایل سرآیند mex.h به ابتدای برنامه است. پس از آن تابع main را حذف و تابع دروازه ورودی را جایگزین آن می‌کنیم. نکته پایانی بررسی تعداد آرگومان‌های ورودی و خروجی به همراه دریافت مقادیرشان است. پس از انجام این کار، برنامه نهایی بشکل برنامه ۲ خواهد بود.

```
1   # include <mex.h>
2   # include <iostream>
3   int zarb (int a, int b){
4   int c = a*b;
5   return c;
6   }
7
8   void mexFunction (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
9
10  {
11  if (nrhs !=2) mexErrMsgTxt ("Two input argument required.");
12  else if (nlhs>1) mexErrMsgTxt ("Too many output arguments!");
13  int i=mxGetScalar (prhs[0]);
14  int j=mxGetScalar (prhs[1]);
15  int z=zarb (i,j);
16  std::cout<<"z="<<z<<std::endl ;
17  }
```

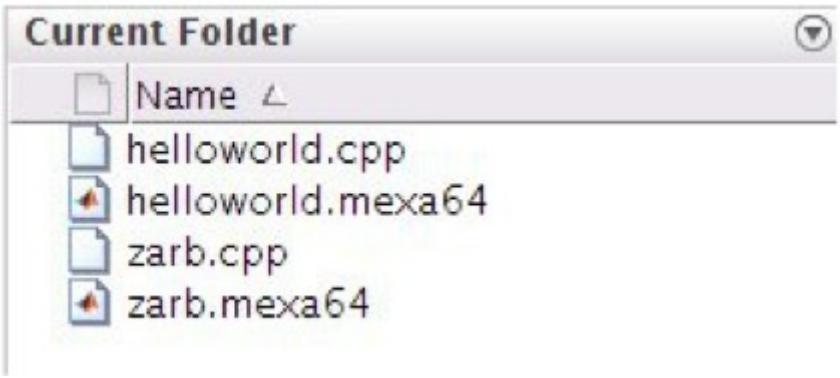
برای اجرای تابع ضرب نیاز به دو آرگومان ورودی است، به همین دلیل در خط ۱۱ بررسی می‌شود که دقیقا ۲ آرگومان ورودی وجود داشته باشد. در صورتی که تعداد آرگومان‌های ورودی متفاوت باشد پیام خطای مناسب چاپ می‌شود. در خط ۱۲ همین مساله برای تعداد آرگومان‌های خروجی بررسی می‌شود. خطوط ۱۳ و ۱۴ مربوط به دریافت مقادیر دو آرگومان ورودی هستند که با استفاده از تابع mxGetScalar انجام می‌شود. خط ۱۵ مقادیر دریافت شده را برای انجام عمل ضری به تابع zarb ارسال و نتیجه آن را در متغییر z ذخیره می‌کند. در پایان نیز در خط ۱۶ با نمایش نتیجه، کار به پایان می‌رسد.

پردازش فایل MEX

فایل ایجاد شده در مرحله قبل را اکنون باید پردازش کرد تا به عنوان یک تایع جدید در متلب شناخته شود. برای این کار در قسمت Current Folder به محلی که فایل MEX ذخیره شده است بروید و در پنجره Command Window متلب دستور mex را مانند شکل ۱ وارد کنید.

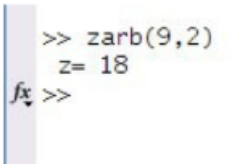
شکل 1: اجرای تابع zarb

اگـر تمـامی مـوارد بـه درسـتی انجـام شـده باشد. هیچ خطـایی نمـایش داده نمی‌شـود و فـایلی بـا نـام zarb.mexa64 به محتویات Current Folder افزوده می‌شود.



شکل 2: نمایش دایرکتوری جاری در متلب

اکنون تابعی با نام zarb به توابع موجود در متلب اضافه شده است که می‌توانید آن را بشکل جداگانه یا در درون برنامه‌ای دیگر استفاده کنید. این عمل در شکل ۳ آمده است.



شکل 3: اجرای برنامه نوشته شده به زبان C++ در متلب

در این مثال دو عدد ۹ و ۲ به عنوان آرگومان‌های ورودی به تابع داده شده‌اند و تابع نتیجه را که برابر با عـدد ۱۸ است به خروجی فرستاده است.

برای دریافت کد برنامه اصلی و برنامه تغییر یافته برای ساخت فایل mex به مخزن https://github.com/snima/MEX بر روی گیت‌هاب سر بزنید.