# Development and debugging MEX-Functions with Eclipse IDE and GDB.

This document describes the setup of Eclipse IDE for the development and debugging of mex-functions.

Demo project contains C source code of the exemplar mex-function,make script and MATLAB script which invokes the mex-function. These exhibit a complete workflow using the standard and available utilities and interfaces.

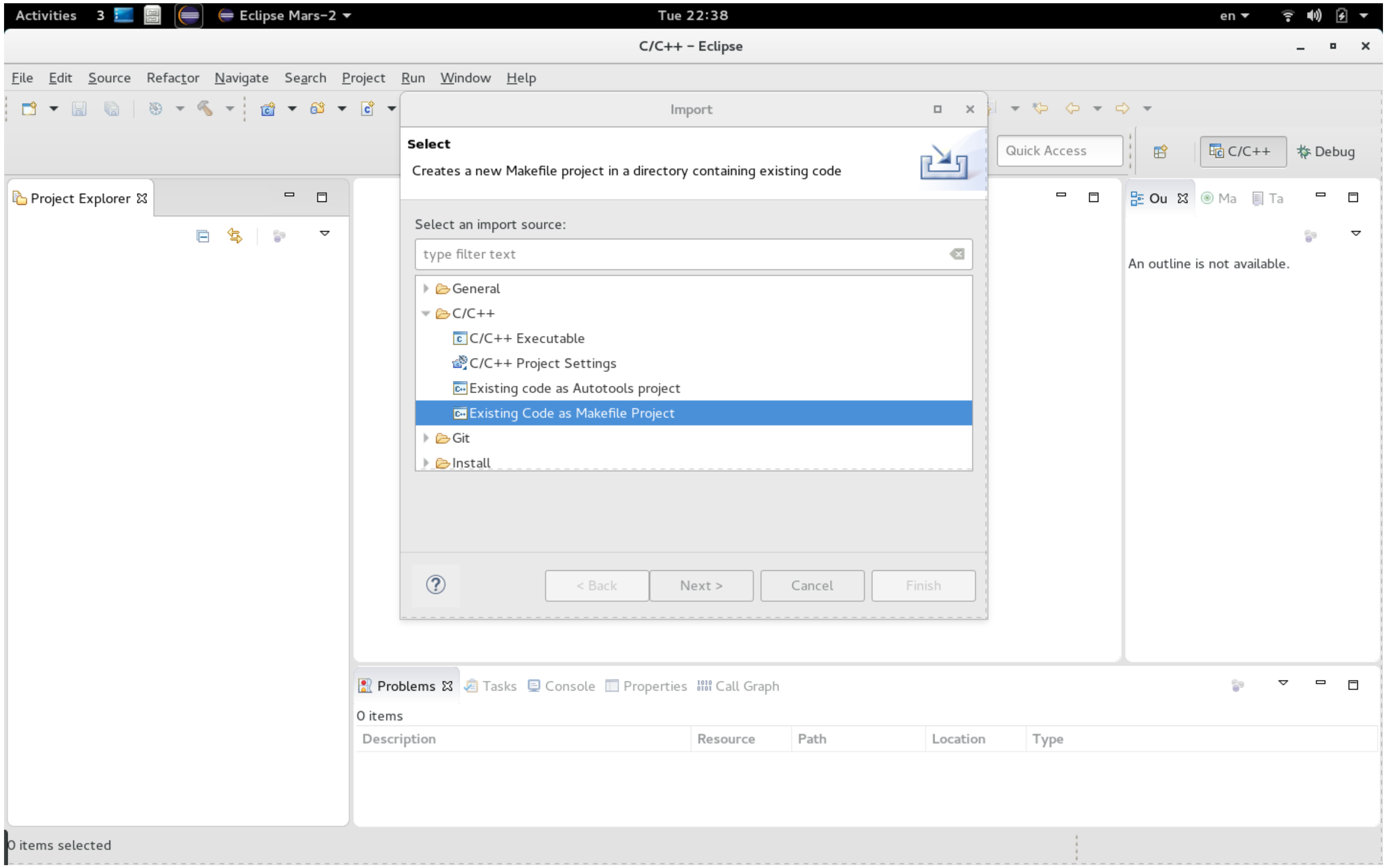
Prerequisites:
Matlab (R2018b in this demo)
Eclipse CDT (Mars-2 in this demo)
GCC (4.8.5 in this demo)
GDB (7.11 in this demo)
make (3.82 in this demo)

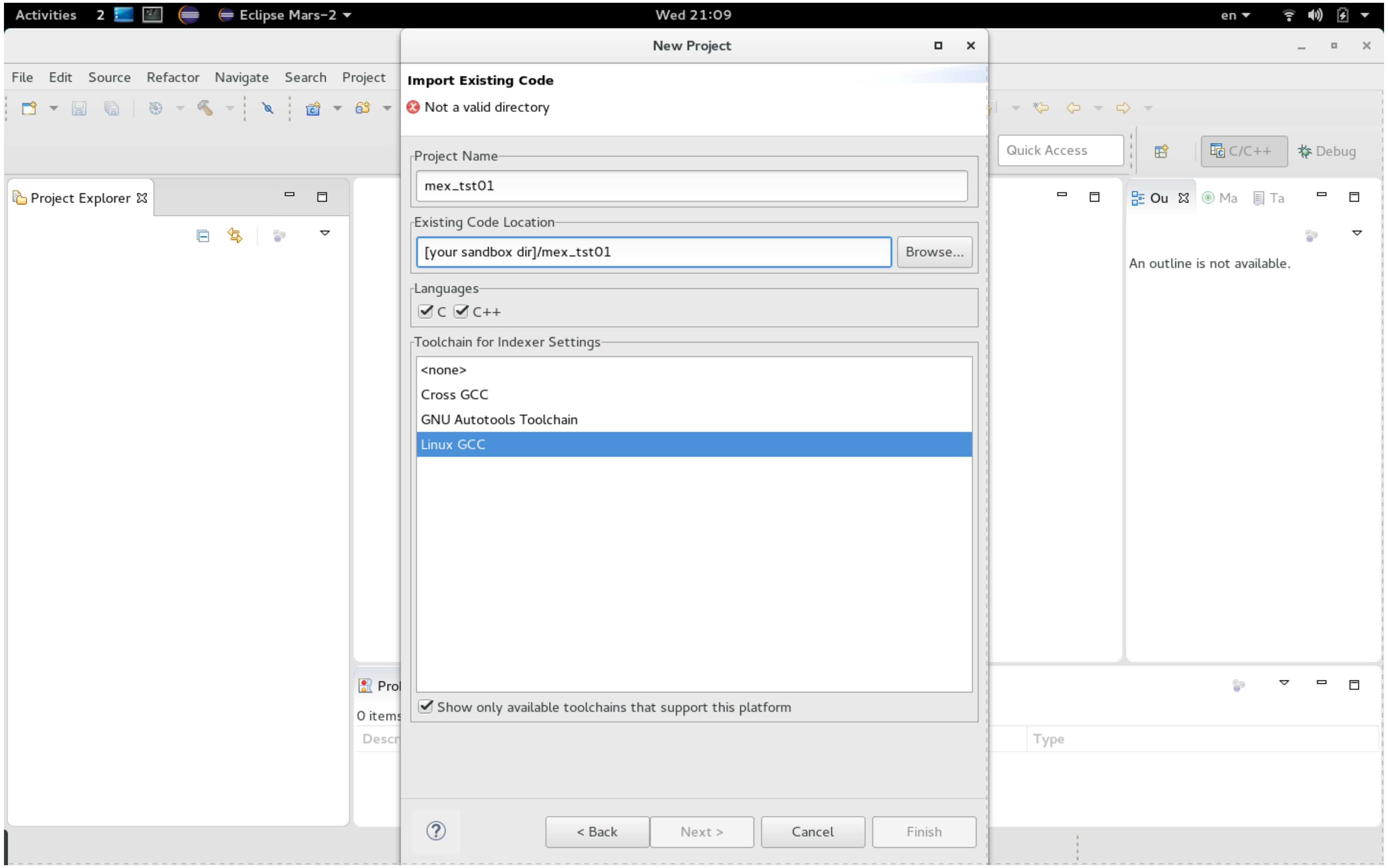
Environment variables:
$MATLAB points to Matlab home

```
export  CC=$(command -v gcc)
export GCC=$(command -v gcc)
export CXX=$(command -v g++)
```

Close MATLAB session
Close Eclipse
Unzip demo.

Start Eclipse. Specify workspace directory. Workspace will contain the reference to the project. Open two perspectives: C/C++ and Debug. Set C/C++ as an active perspective. Import the sample project: File → Import → C/C++ → Existing Code as Makefile Project





New Project

**Import Existing Code**

Not a valid directory

Project Name: mex_tst01

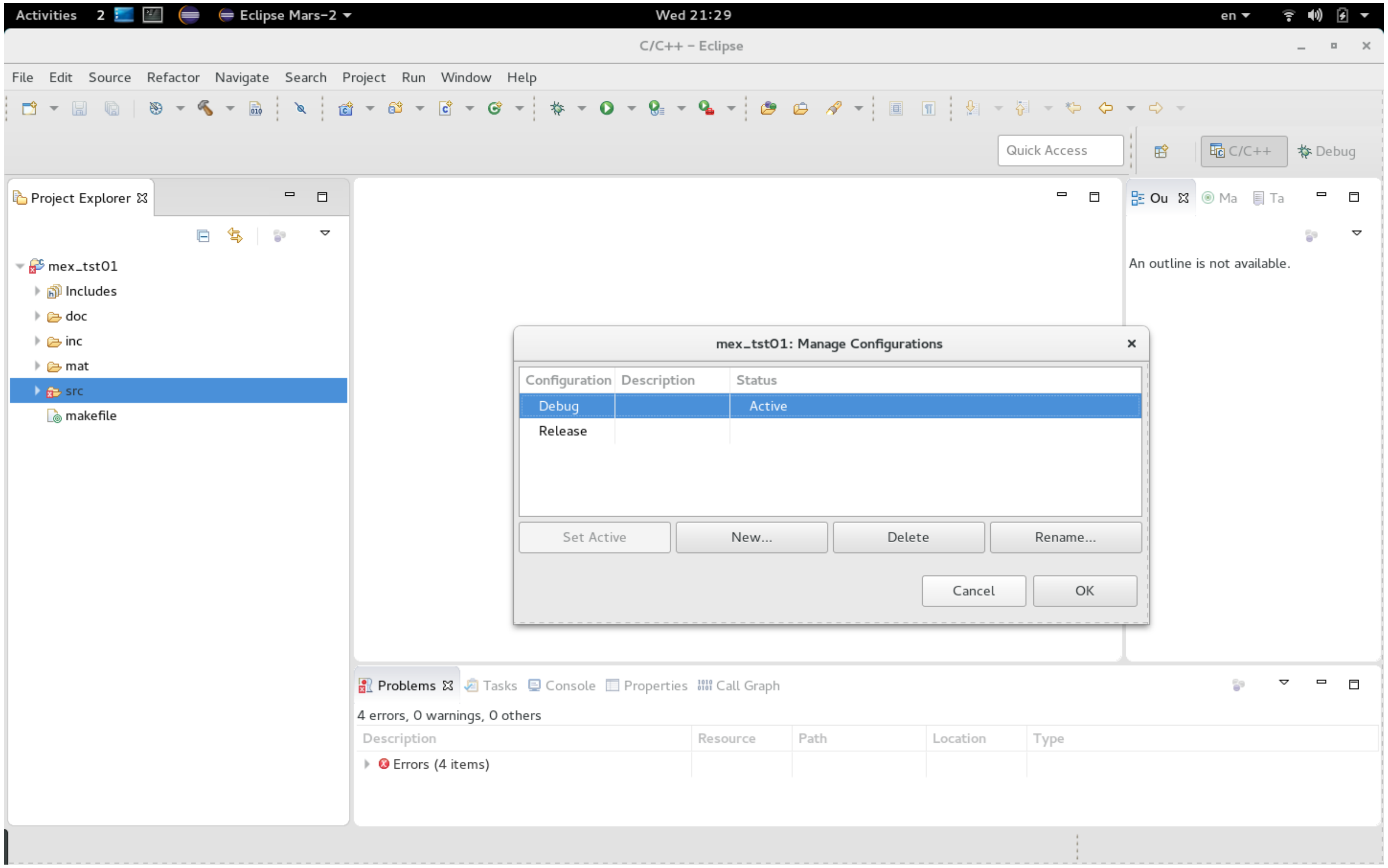
Existing Code Location: [your sandbox dir]/mex_tst01

Languages: ✓ C ✓ C++

Toolchain for Indexer Settings:
- <none>
- Cross GCC
- GNU Autotools Toolchain
- Linux GCC

✓ Show only available toolchains that support this platform

Create two build configurations: Debug and Release.
When the Debug configuration is active, the project is built with no optimization and the mex-file includes debug information.
Release configuration is for the “end product”. Compile optimization is enabled and no debug information is present in the mex-file.
This set of configurations is supported by makefile. Now we need Eclipse to know about them.
Project → Build Configurations → Manage



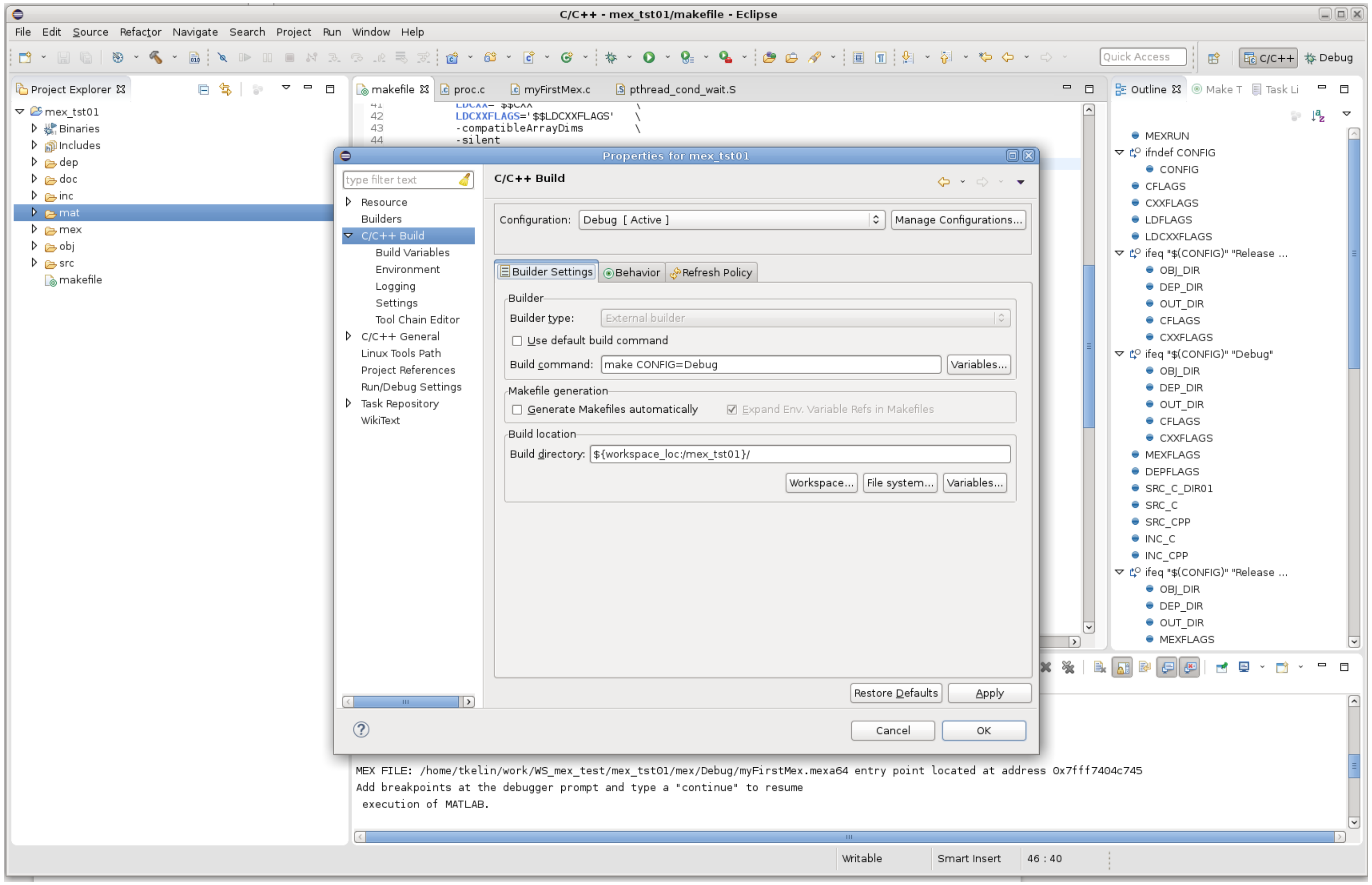
mex_tst01: Manage Configurations

| Configuration | Description | Status |
|---|---|---|
| Debug | | Active |
| Release | | |

4 errors, 0 warnings, 0 others

| Description | Resource | Path | Location | Type |
|---|---|---|---|---|
| Errors (4 items) | | | | |

## Edit Debug Build Configuration
## Project → Properties → C/C++ Build



```
LDCXXFLAGS='$$LDCXXFLAGS'   \
-compatibleArrayDims        \
-silent
```

**C/C++ Build**

Configuration: Debug [ Active ] — Manage Configurations...

Builder
- Builder type: External builder
- [ ] Use default build command
- Build command: make CONFIG=Debug — Variables...

Makefile generation
- [ ] Generate Makefiles automatically
- [✓] Expand Env. Variable Refs in Makefiles

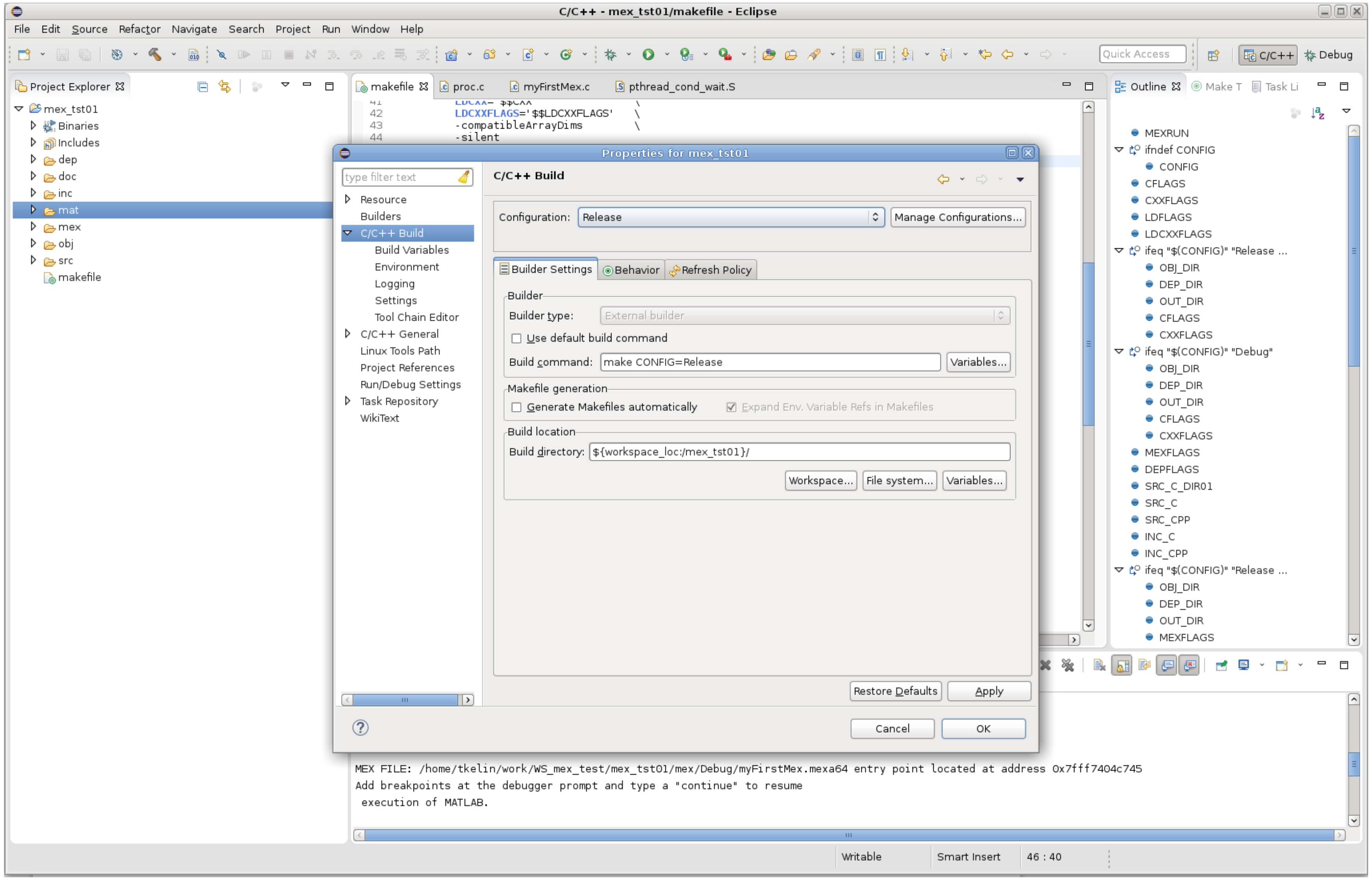
Build location
- Build directory: ${workspace_loc:/mex_tst01}/ — Workspace... File system... Variables...

Outline:

```
MEXRUN
ifndef CONFIG
    CONFIG
CFLAGS
CXXFLAGS
LDFLAGS
LDCXXFLAGS
ifeq "$(CONFIG)" "Release ...
    OBJ_DIR
    DEP_DIR
    OUT_DIR
    CFLAGS
    CXXFLAGS
ifeq "$(CONFIG)" "Debug"
    OBJ_DIR
    DEP_DIR
    OUT_DIR
    CFLAGS
    CXXFLAGS
MEXFLAGS
DEPFLAGS
SRC_C_DIR01
SRC_C
SRC_CPP
INC_C
INC_CPP
ifeq "$(CONFIG)" "Release ...
    OBJ_DIR
    DEP_DIR
    OUT_DIR
    MEXFLAGS
```

```
MEX FILE: /home/tkelin/work/WS_mex_test/mex_tst01/mex/Debug/myFirstMex.mexa64 entry point located at address 0x7fff7404c745
Add breakpoints at the debugger prompt and type a "continue" to resume
 execution of MATLAB.
```

## Edit Release Build Configuration



```
        LDCXX= $$CXX                \
        LDCXXFLAGS='$$LDCXXFLAGS'   \
        -compatibleArrayDims        \
        -silent
```

Configuration: Release

Builder

Builder type: External builder

☐ Use default build command

Build command: make CONFIG=Release

Makefile generation

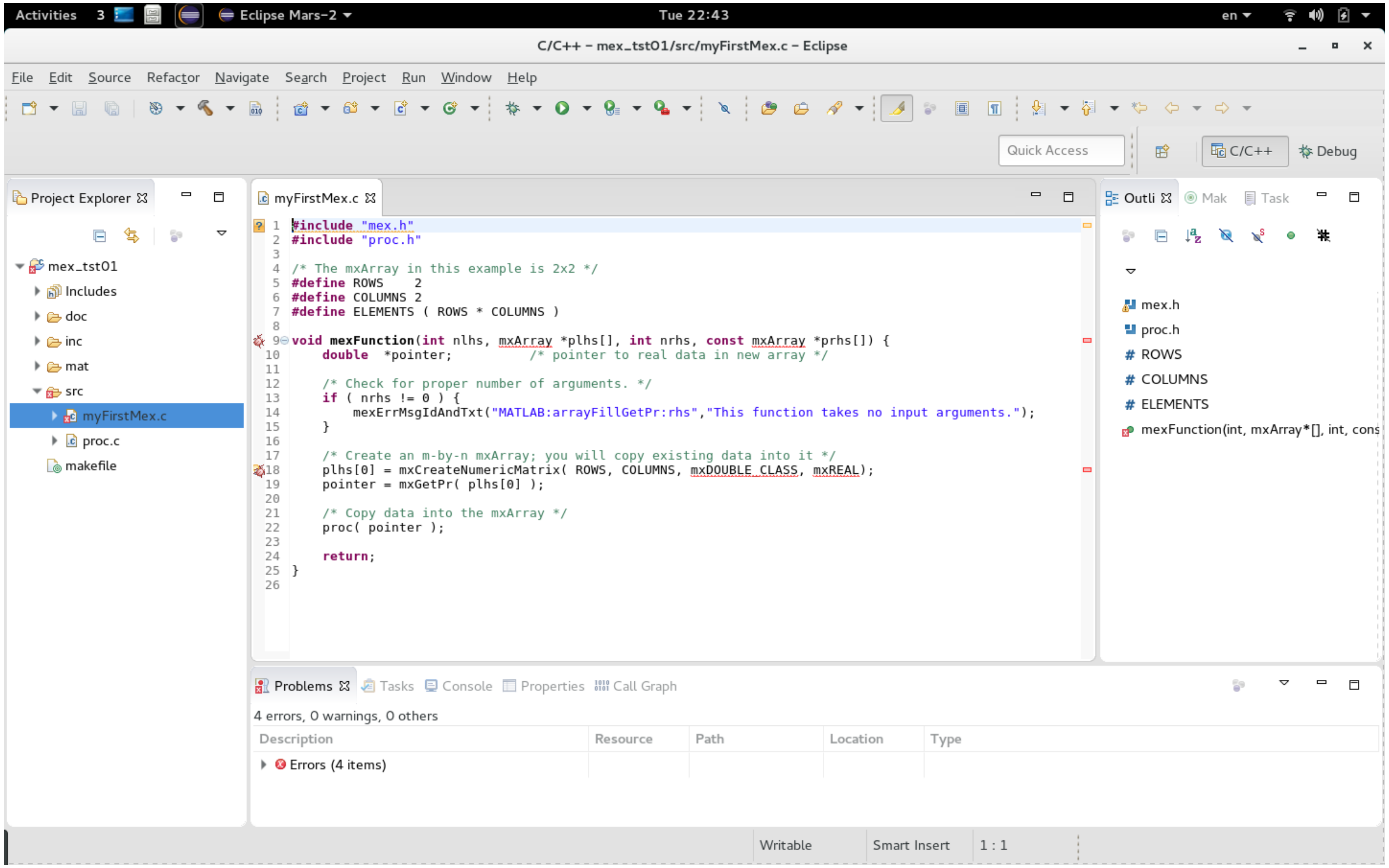
☐ Generate Makefiles automatically  ☑ Expand Env. Variable Refs in Makefiles

Build location

Build directory: ${workspace_loc:/mex_tst01}/

```
MEX FILE: /home/tkelin/work/WS_mex_test/mex_tst01/mex/Debug/myFirstMex.mexa64 entry point located at address 0x7fff7404c745
Add breakpoints at the debugger prompt and type a "continue" to resume
 execution of MATLAB.
```

Project Explorer shows the directory tree. It is the same as the project tree.
Open any source file, there are red underlines for unknown macros and data types.

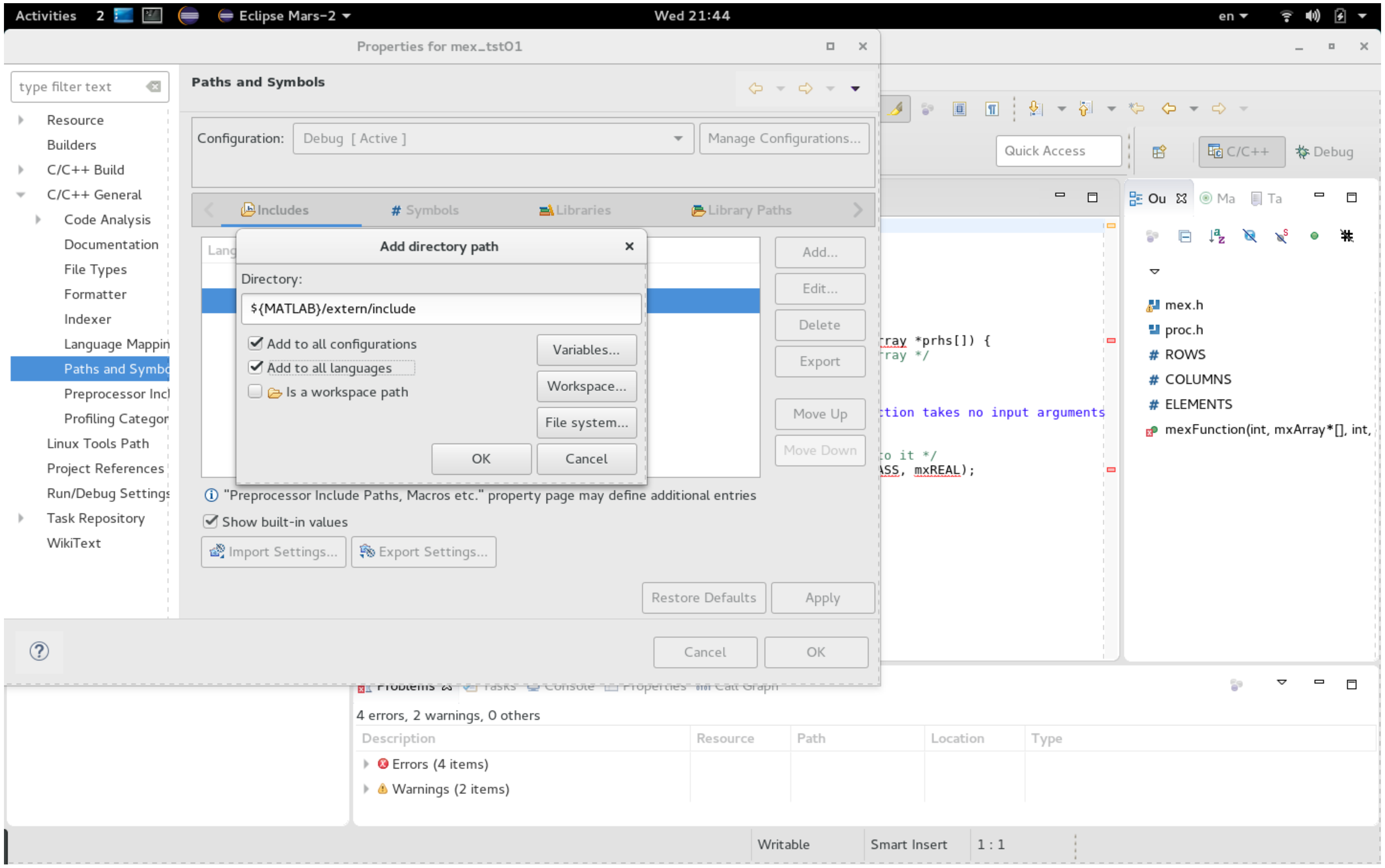


```c
#include "mex.h"
#include "proc.h"

/* The mxArray in this example is 2x2 */
#define ROWS    2
#define COLUMNS 2
#define ELEMENTS ( ROWS * COLUMNS )

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]) {
    double  *pointer;          /* pointer to real data in new array */

    /* Check for proper number of arguments. */
    if ( nrhs != 0 ) {
        mexErrMsgIdAndTxt("MATLAB:arrayFillGetPr:rhs","This function takes no input arguments.");
    }

    /* Create an m-by-n mxArray; you will copy existing data into it */
    plhs[0] = mxCreateNumericMatrix( ROWS, COLUMNS, mxDOUBLE_CLASS, mxREAL);
    pointer = mxGetPr( plhs[0] );

    /* Copy data into the mxArray */
    proc( pointer );

    return;
}
```

4 errors, 0 warnings, 0 others

Errors (4 items)

```
Edit the project properties.
Project → Properties → C/C++ General → Paths and Symbols → GNU C
```



Configuration: Debug [ Active ]

Directory:
${MATLAB}/extern/include

- [✓] Add to all configurations
- [✓] Add to all languages
- [ ] Is a workspace path

"Preprocessor Include Paths, Macros etc." property page may define additional entries

- [✓] Show built-in values

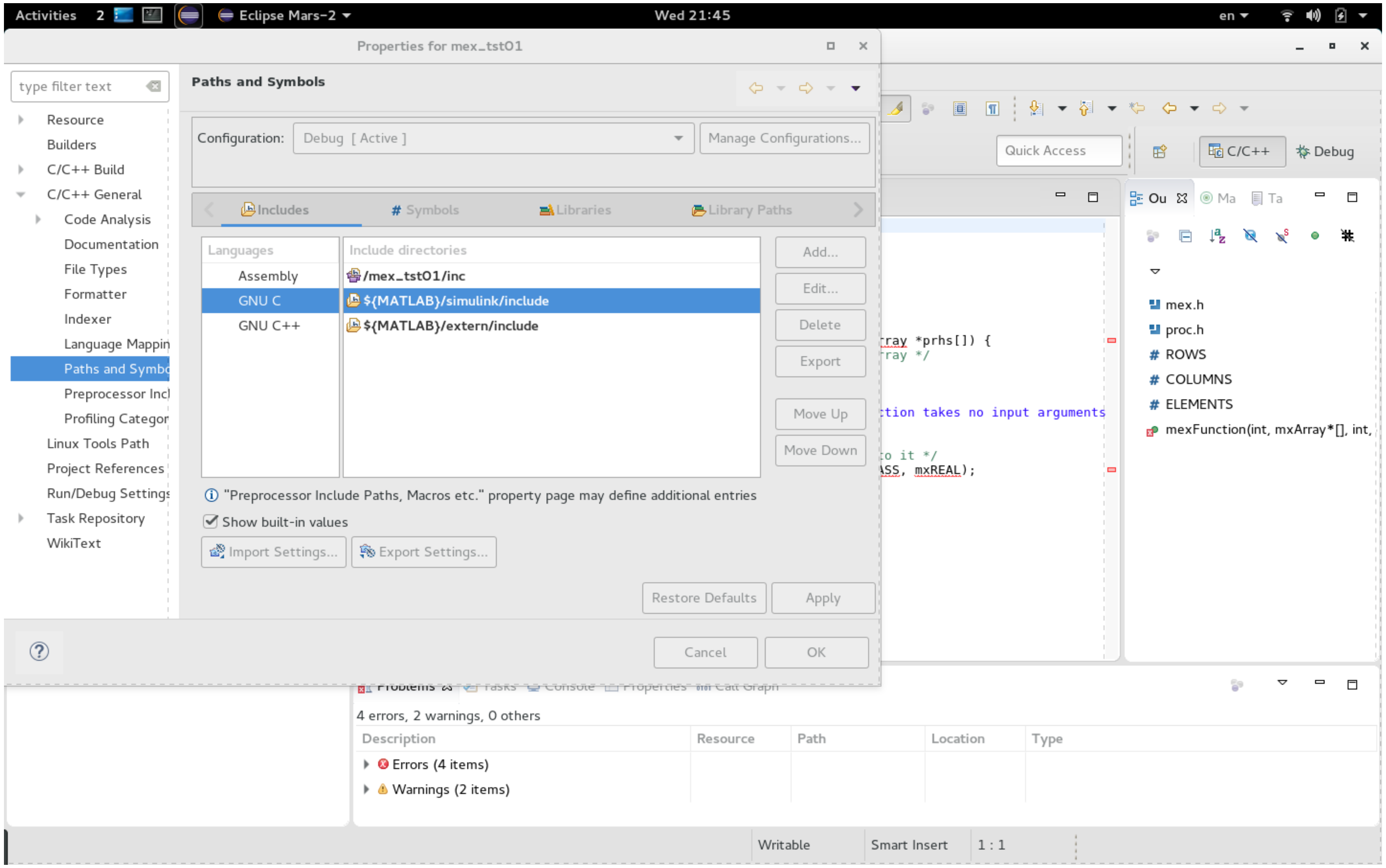
```
ray *prhs[]) {
ray */
tion takes no input arguments
o it */
SS, mxREAL);
```

4 errors, 2 warnings, 0 others

- Errors (4 items)
- Warnings (2 items)

8

Add MATLAB directories which contain include files corresponding to mex development. We also specify local inc folder with .h-files.



Properties for mex_tst01

Paths and Symbols

Configuration: Debug [ Active ]

Includes | Symbols | Libraries | Library Paths

| Languages | Include directories |
|---|---|
| Assembly | /mex_tst01/inc |
| GNU C | ${MATLAB}/simulink/include |
| GNU C++ | ${MATLAB}/extern/include |

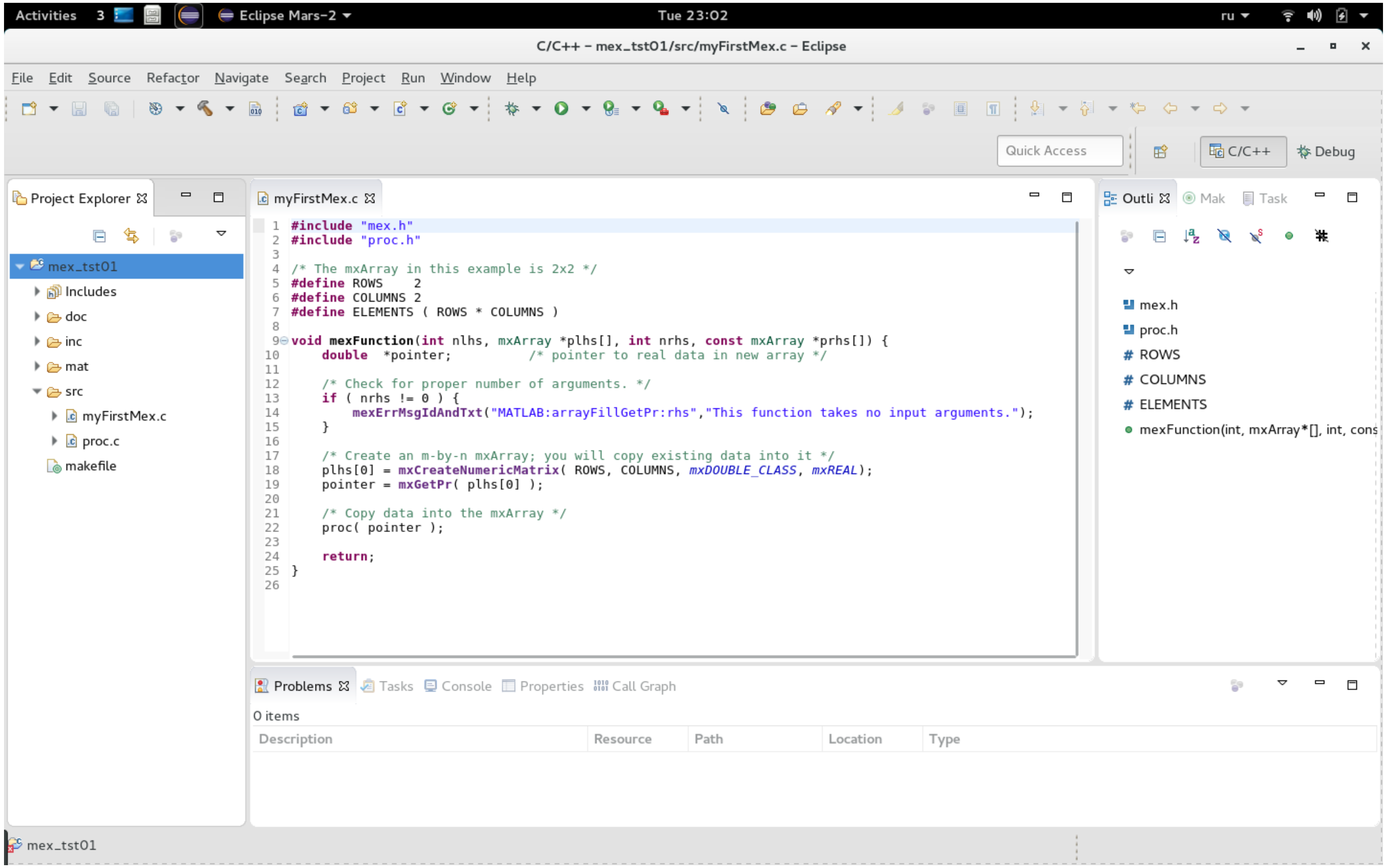
"Preprocessor Include Paths, Macros etc." property page may define additional entries

[✓] Show built-in values

4 errors, 2 warnings, 0 others

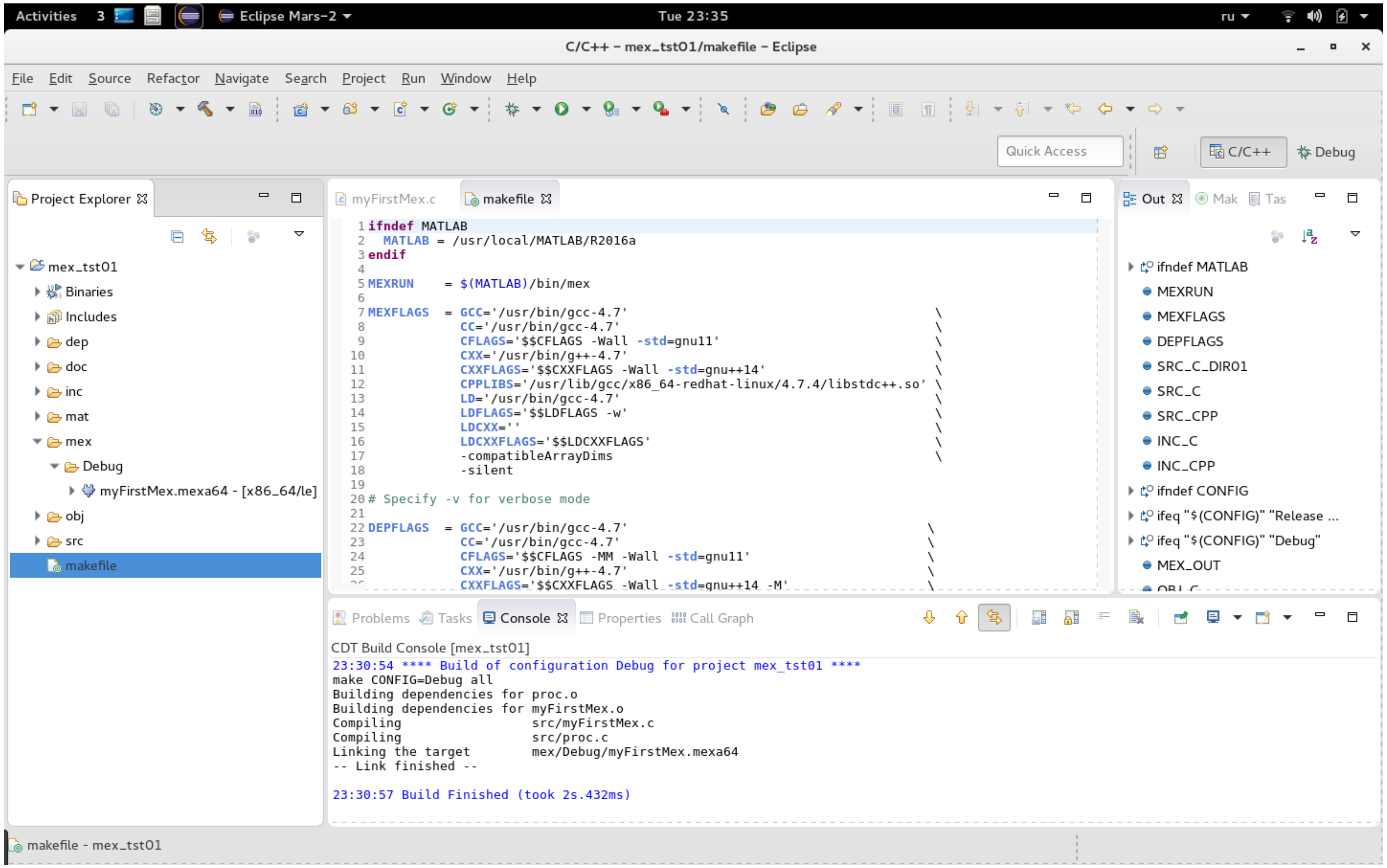
| Description | Resource | Path | Location | Type |
|---|---|---|---|---|
| Errors (4 items) | | | | |
| Warnings (2 items) | | | | |

Project → C/C++ Index → Rebuild. The project is now indexed with no missing specifications and all the red underlines disappeared.



```c
#include "mex.h"
#include "proc.h"

/* The mxArray in this example is 2x2 */
#define ROWS    2
#define COLUMNS 2
#define ELEMENTS ( ROWS * COLUMNS )

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]) {
    double  *pointer;          /* pointer to real data in new array */

    /* Check for proper number of arguments. */
    if ( nrhs != 0 ) {
        mexErrMsgIdAndTxt("MATLAB:arrayFillGetPr:rhs","This function takes no input arguments.");
    }

    /* Create an m-by-n mxArray; you will copy existing data into it */
    plhs[0] = mxCreateNumericMatrix( ROWS, COLUMNS, mxDOUBLE_CLASS, mxREAL);
    pointer = mxGetPr( plhs[0] );

    /* Copy data into the mxArray */
    proc( pointer );

    return;
}
```

Now we are going to try building the project in the Debug configuration:
Project → Build Configurations → Set Active → Debug
Project → Build Project
In the Console output window we see no build errors. New build specific folders were created in the Project Explorer. Mex-function executable was also created. We will be running and debugging it.



```
ifndef MATLAB
  MATLAB = /usr/local/MATLAB/R2016a
endif

MEXRUN    = $(MATLAB)/bin/mex

MEXFLAGS  = GCC='/usr/bin/gcc-4.7'                                        \
            CC='/usr/bin/gcc-4.7'                                         \
            CFLAGS='$$CFLAGS -Wall -std=gnu11'                            \
            CXX='/usr/bin/g++-4.7'                                        \
            CXXFLAGS='$$CXXFLAGS -Wall -std=gnu++14'                      \
            CPPLIBS='/usr/lib/gcc/x86_64-redhat-linux/4.7.4/libstdc++.so' \
            LD='/usr/bin/gcc-4.7'                                         \
            LDFLAGS='$$LDFLAGS -w'                                        \
            LDCXX=''                                                      \
            LDCXXFLAGS='$$LDCXXFLAGS'                                     \
            -compatibleArrayDims                                          \
            -silent

# Specify -v for verbose mode

DEPFLAGS  = GCC='/usr/bin/gcc-4.7'                                       \
            CC='/usr/bin/gcc-4.7'                                        \
            CFLAGS='$$CFLAGS -MM -Wall -std=gnu11'                       \
            CXX='/usr/bin/g++-4.7'                                       \
            CXXFLAGS='$$CXXFLAGS -Wall -std=gnu++14 -M'                  \
```

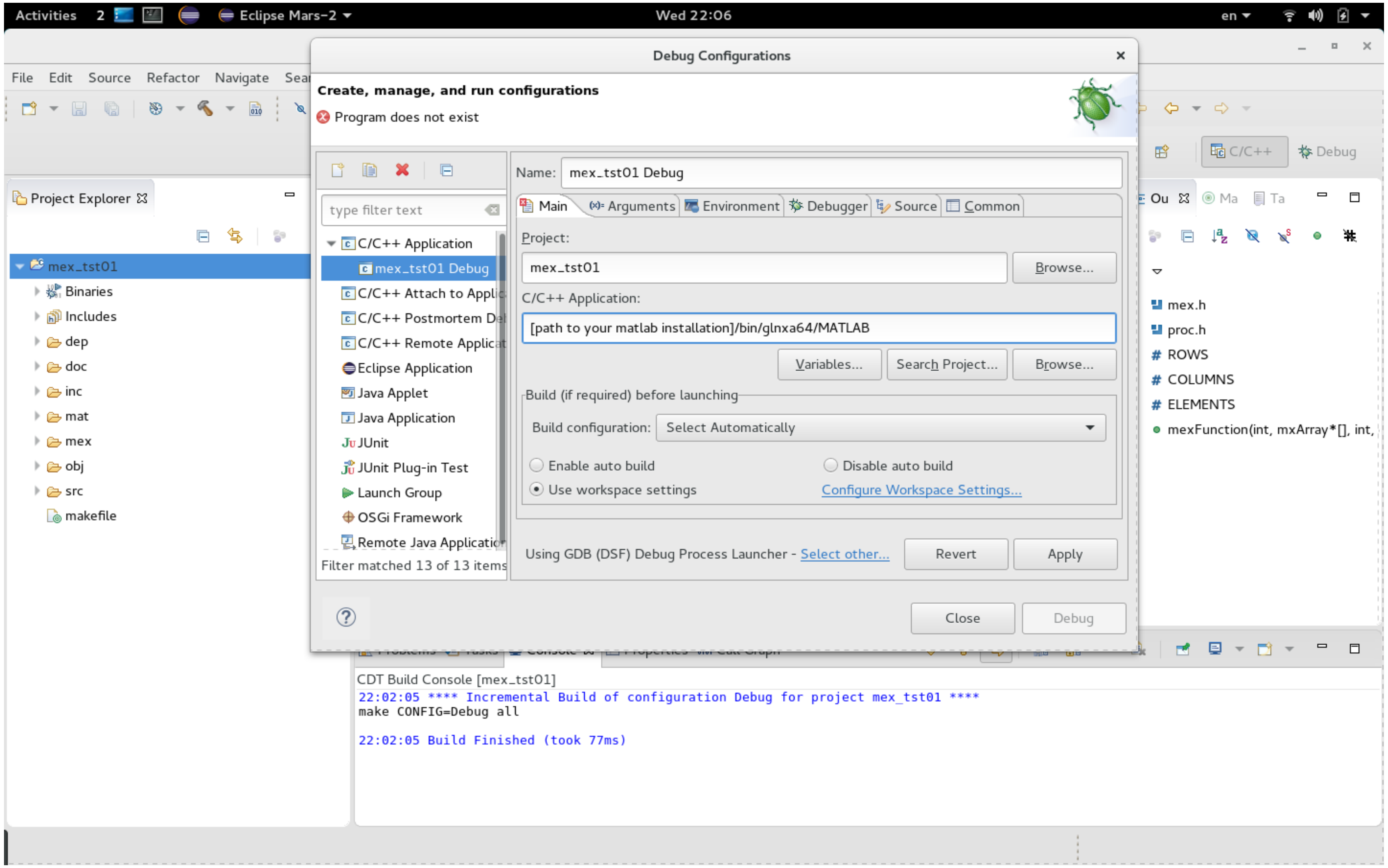
```
CDT Build Console [mex_tst01]
23:30:54 **** Build of configuration Debug for project mex_tst01 ****
make CONFIG=Debug all
Building dependencies for proc.o
Building dependencies for myFirstMex.o
Compiling                  src/myFirstMex.c
Compiling                  src/proc.c
Linking the target         mex/Debug/myFirstMex.mexa64
-- Link finished --

23:30:57 Build Finished (took 2s.432ms)
```

At this stage we need to set up Eclipse IDE so it calls GDB for debugging.

When debug process starts the sequence of the calls is the following:
- Eclipse starts GDB
- GDB runs MATLAB session with JVM disabled (-nojvm option)
- MATLAB runs script (mat/mex_dbg01.m) with the calls to the mex-function which we are going to debug.
- As soon as the execution of the script comes to the point when the mex-file is loaded, MATLAB stops the execution and passes the control to GDB presumably via the JIT Compilation Interface. At this stage we have full source-level visibility of the mex-function execution process and its internals for debugging purposes.

Passing the control to GDB can be enabled and disabled from within MATLAB script and we can specify which function call or calls are required to be debugged. In the sample script the mex-function is called 3 times but the debugger is enabled only for the second call.

Run→ Debug Configurations → C/C++ Application→ Double Click → Main
As “C/C++ Application” we specify the full path to MATLAB binary executable.



Create, manage, and run configurations
Program does not exist

Filter matched 13 of 13 items

Name: mex_tst01 Debug

Project:
mex_tst01

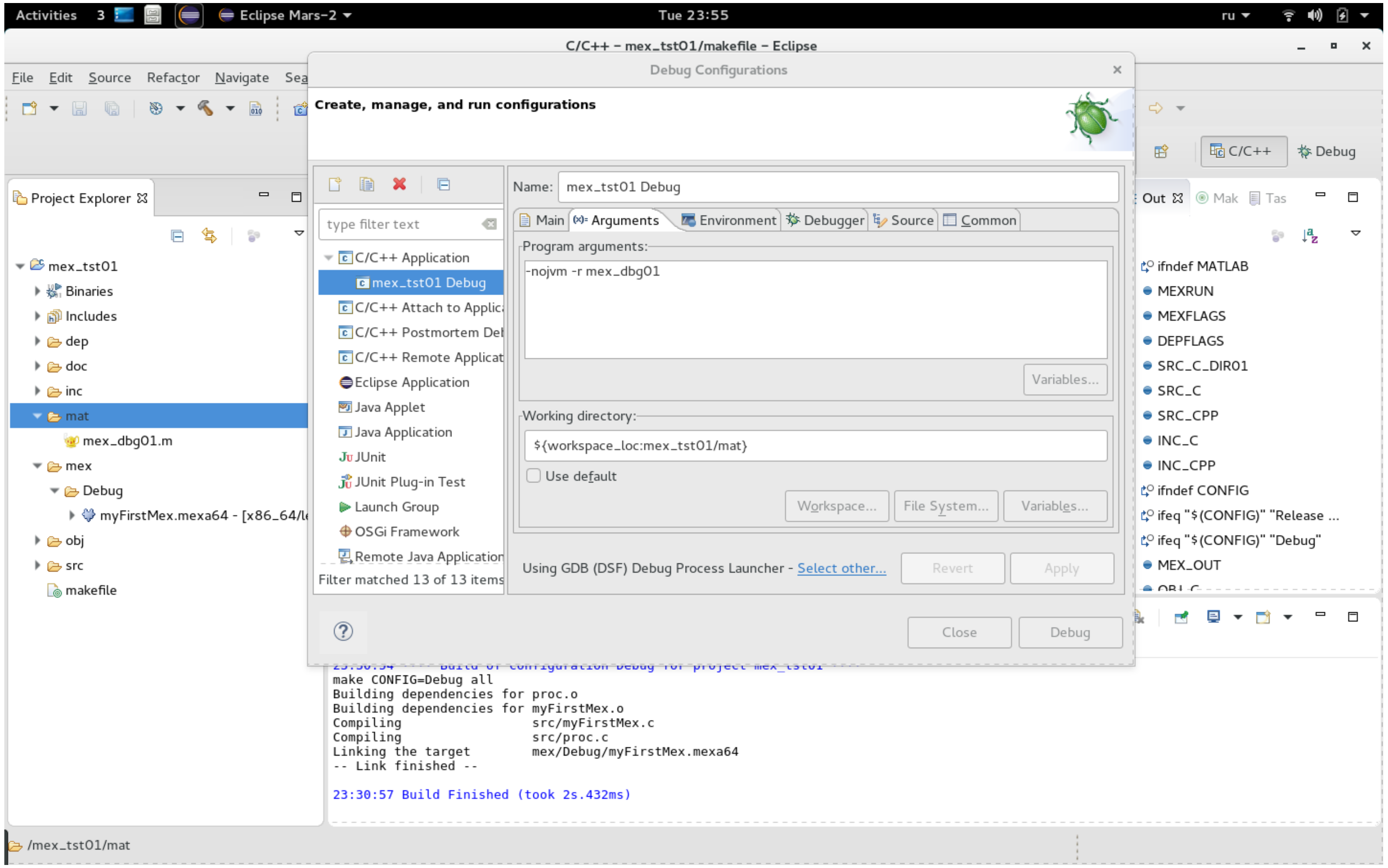
C/C++ Application:
[path to your matlab installation]/bin/glnxa64/MATLAB

Build (if required) before launching
Build configuration: Select Automatically
Enable auto build
Disable auto build
Use workspace settings [✓]
Configure Workspace Settings...

Using GDB (DSF) Debug Process Launcher - Select other...

CDT Build Console [mex_tst01]
```
22:02:05 **** Incremental Build of configuration Debug for project mex_tst01 ****
make CONFIG=Debug all

22:02:05 Build Finished (took 77ms)
```

Specify command line options for MATLAB executable:
-nojvm – start without Java
-r mex_dbg01 – run script mex_dbg01.m at MATLAB start. This script adds path to the mex-file to MATLAB path list and calls our mex function.
We also specify the directory where the script mex_dbg01.m is located.



Name: mex_tst01 Debug

Program arguments:
-nojvm -r mex_dbg01

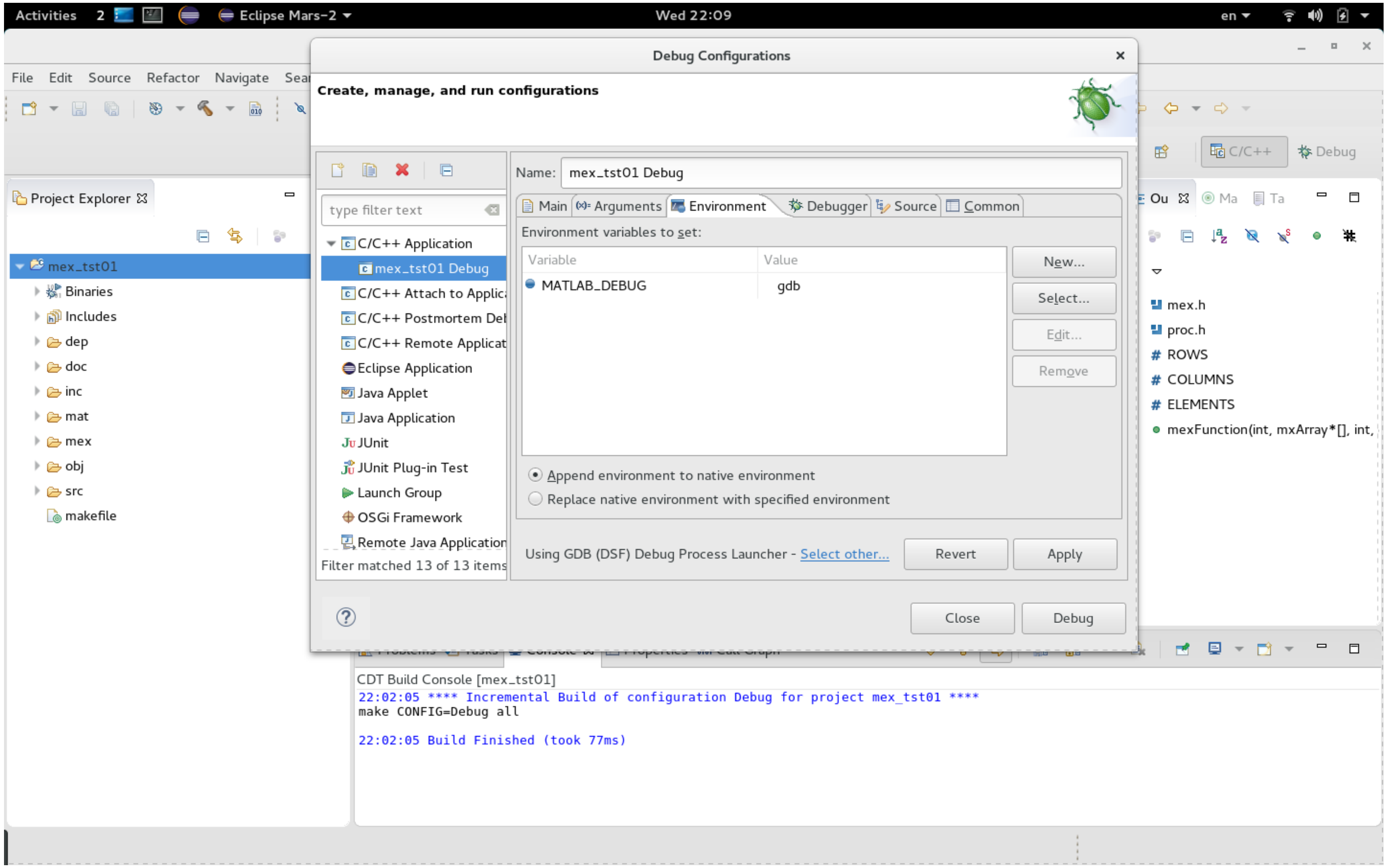
Working directory:
${workspace_loc:mex_tst01/mat}

[ ] Use default

Using GDB (DSF) Debug Process Launcher - Select other...

```
make CONFIG=Debug all
Building dependencies for proc.o
Building dependencies for myFirstMex.o
Compiling                 src/myFirstMex.c
Compiling                 src/proc.c
Linking the target        mex/Debug/myFirstMex.mexa64
-- Link finished --

23:30:57 Build Finished (took 2s.432ms)
```

Set the environment variable which tells MATLAB that it is executed under gdb, and that it needs to pass the control to the debugger in response to the corresponding commands in the script.


Debug Configurations

Create, manage, and run configurations

Name: mex_tst01 Debug

Environment variables to set:

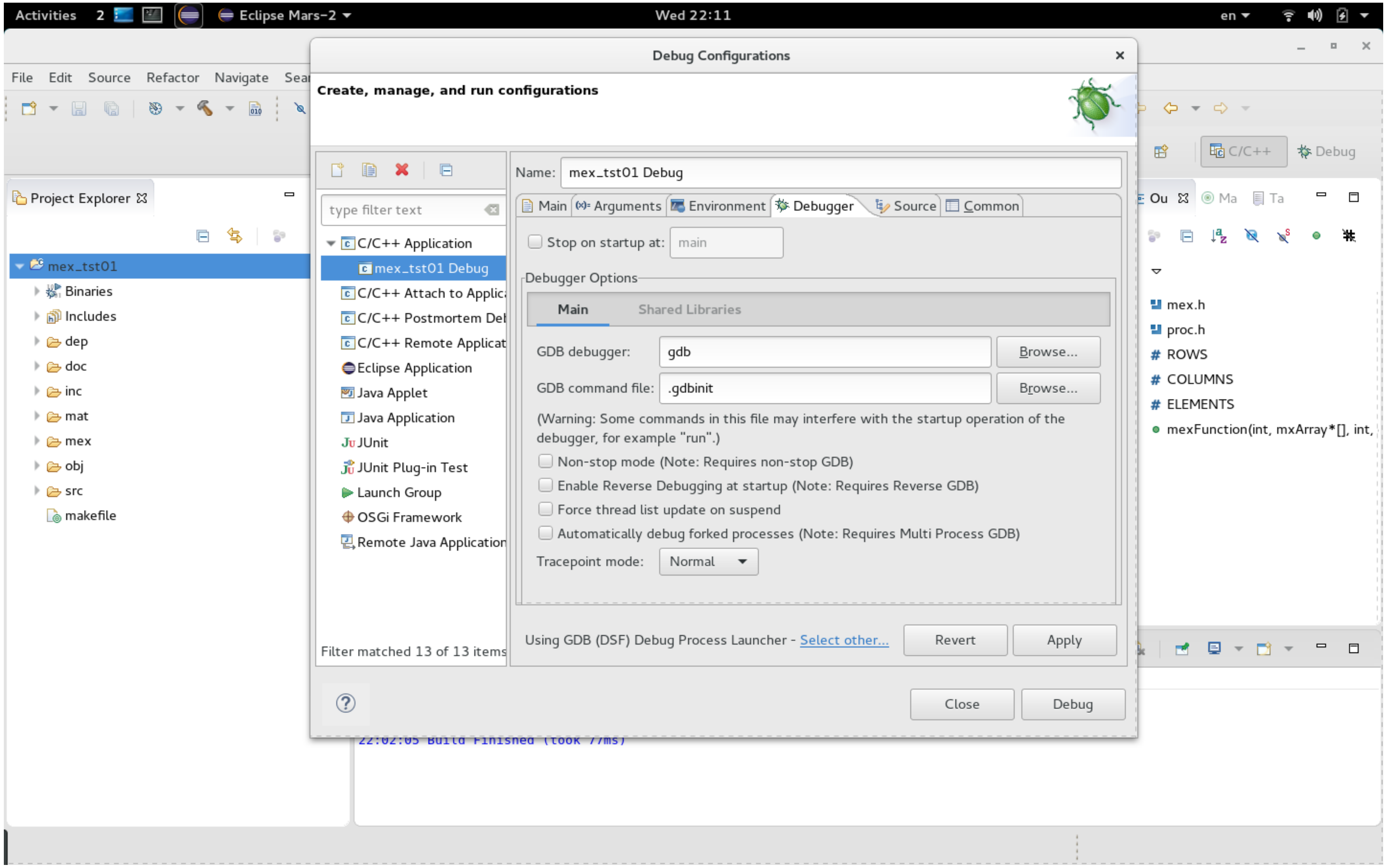
| Variable | Value |
|---|---|
| MATLAB_DEBUG | gdb |

(●) Append environment to native environment
( ) Replace native environment with specified environment

Using GDB (DSF) Debug Process Launcher - Select other...

```
CDT Build Console [mex_tst01]
22:02:05 **** Incremental Build of configuration Debug for project mex_tst01 ****
make CONFIG=Debug all

22:02:05 Build Finished (took 77ms)
```

We don't set the breakpoint on main(). We are not interested in MATLAB's main().



Debug Configurations

**Create, manage, and run configurations**

type filter text

- C/C++ Application
  - mex_tst01 Debug
- C/C++ Attach to Applic
- C/C++ Postmortem Del
- C/C++ Remote Applicat
- Eclipse Application
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- Launch Group
- OSGi Framework
- Remote Java Application

Filter matched 13 of 13 items

Name: mex_tst01 Debug

Main | Arguments | Environment | Debugger | Source | Common

☐ Stop on startup at: main

Debugger Options

Main | Shared Libraries

GDB debugger: gdb [Browse...]

GDB command file: .gdbinit [Browse...]

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

☐ Non-stop mode (Note: Requires non-stop GDB)
☐ Enable Reverse Debugging at startup (Note: Requires Reverse GDB)
☐ Force thread list update on suspend
☐ Automatically debug forked processes (Note: Requires Multi Process GDB)

Tracepoint mode: Normal

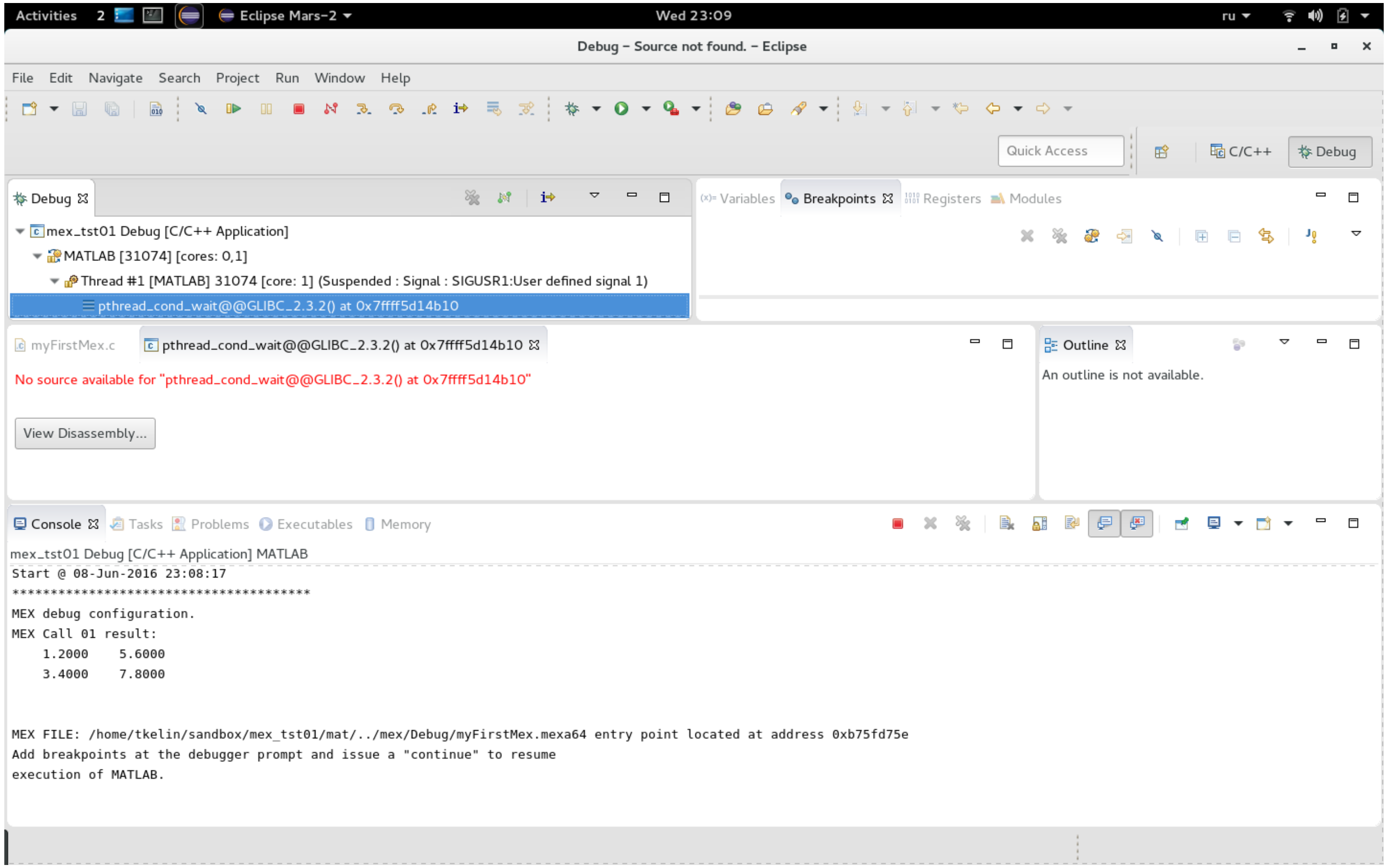
Using GDB (DSF) Debug Process Launcher - Select other...

Revert | Apply

Close | Debug

```
22:02:05 Build Finished (took 77ms)
```

Start MATLAB from under gdb: Run → Debug. Wait until gdb stops and then switch the active perspective to Debug. In the Console output window we can see the result of the first call to the mex-function for which the control wasn't passed to gdb. The execution stops when MATLAB loads mex function second time and is about to call it.



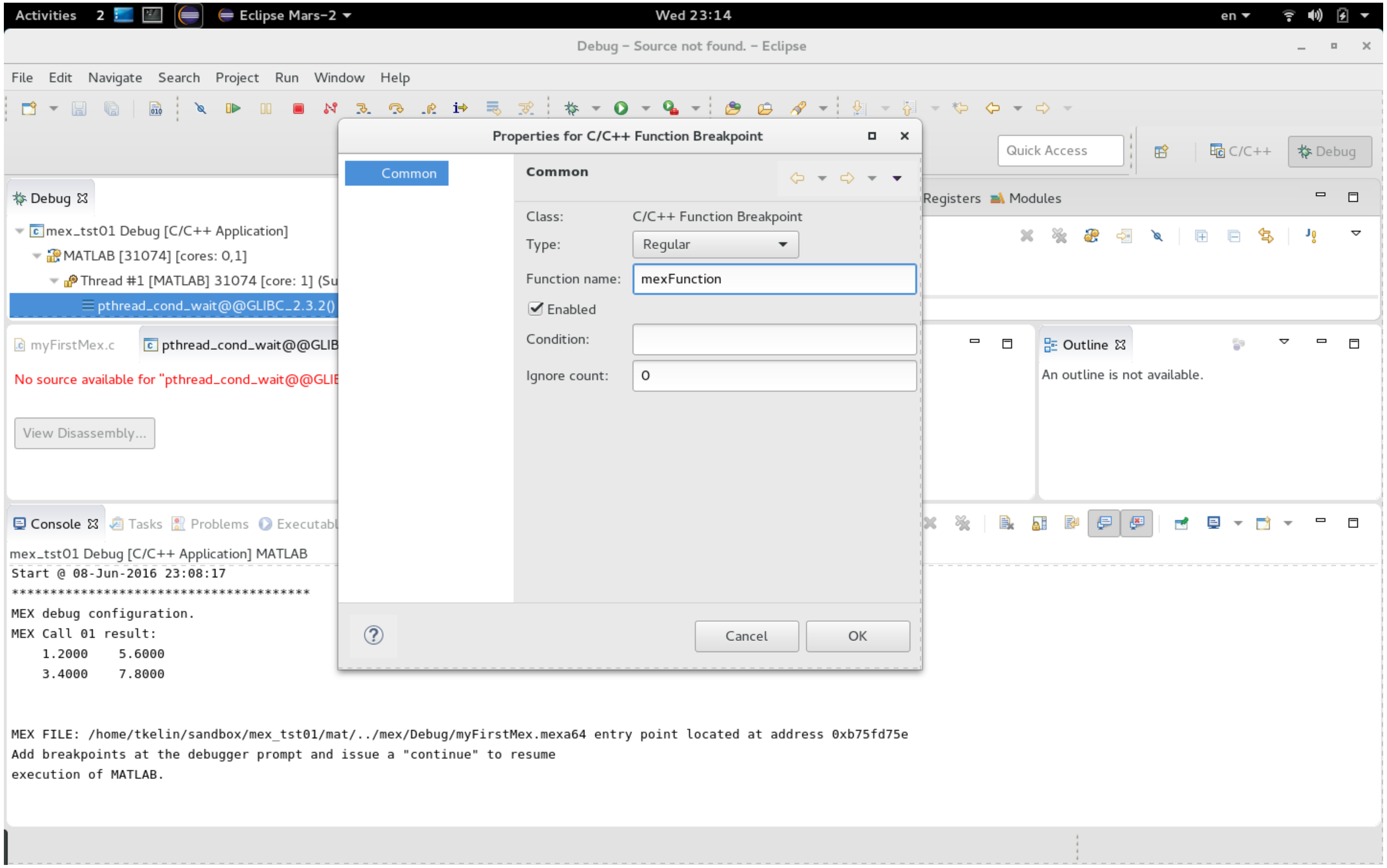
mex_tst01 Debug [C/C++ Application]
MATLAB [31074] [cores: 0,1]
Thread #1 [MATLAB] 31074 [core: 1] (Suspended : Signal : SIGUSR1:User defined signal 1)
pthread_cond_wait@@GLIBC_2.3.2() at 0x7ffff5d14b10

No source available for "pthread_cond_wait@@GLIBC_2.3.2() at 0x7ffff5d14b10"

mex_tst01 Debug [C/C++ Application] MATLAB

```
Start @ 08-Jun-2016 23:08:17
***************************************
MEX debug configuration.
MEX Call 01 result:
    1.2000    5.6000
    3.4000    7.8000


MEX FILE: /home/tkelin/sandbox/mex_tst01/mat/../mex/Debug/myFirstMex.mexa64 entry point located at address 0xb75fd75e
Add breakpoints at the debugger prompt and issue a "continue" to resume
execution of MATLAB.
```

Set breakpoint to our mex-function mexFunction(). Continue execution: Run → Resume.


Activities
Eclipse Mars-2
Wed 23:14
Debug – Source not found. – Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
C/C++
Debug
Debug
mex_tst01 Debug [C/C++ Application]
MATLAB [31074] [cores: 0,1]
Thread #1 [MATLAB] 31074 [core: 1] (Su
pthread_cond_wait@@GLIBC_2.3.2()
myFirstMex.c
pthread_cond_wait@@GLIB
View Disassembly...
Registers
Modules
Outline
An outline is not available.
Console
Tasks
Problems
Executabl
Properties for C/C++ Function Breakpoint
Common
Cancel
OK

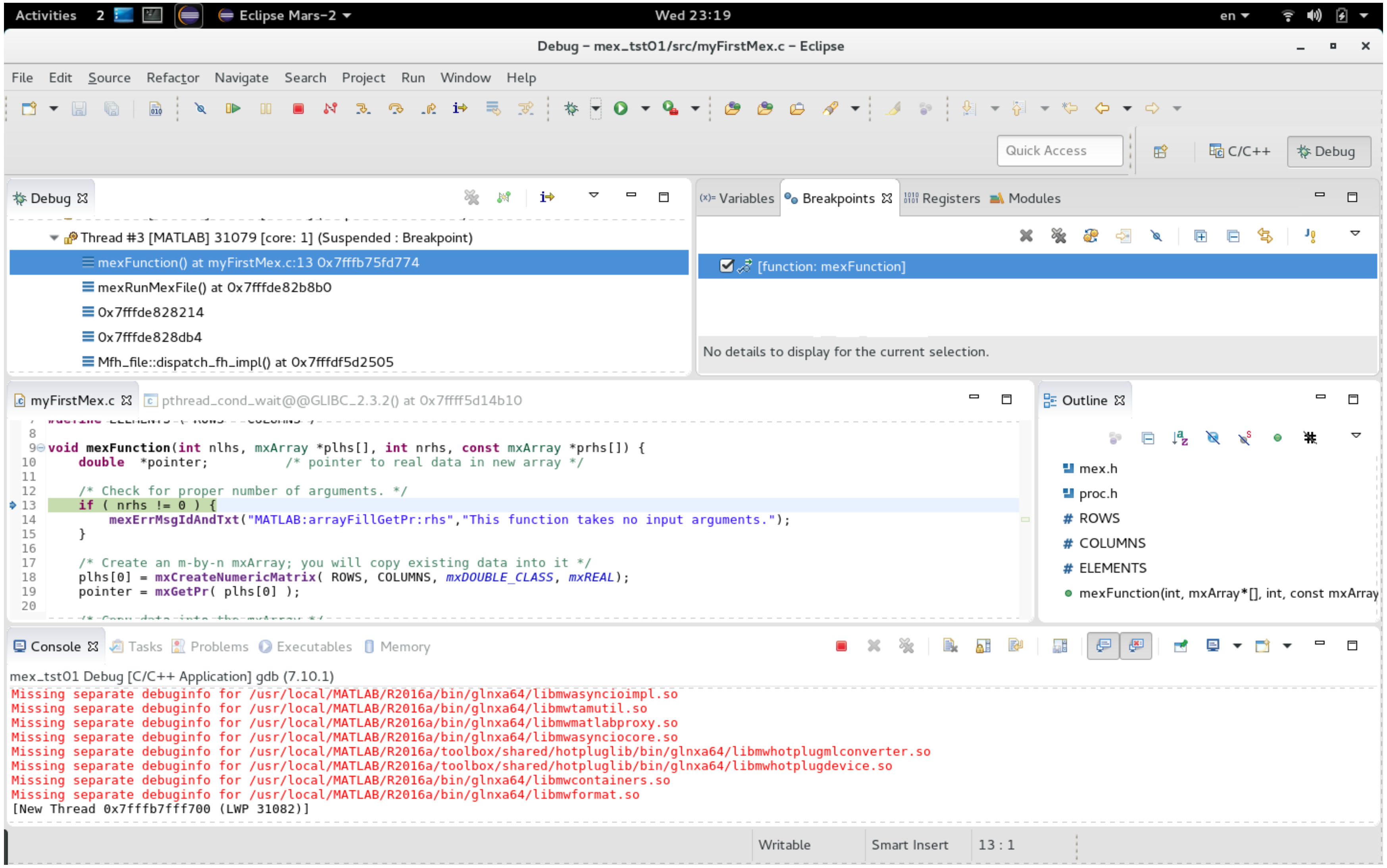
No source available for "pthread_cond_wait@@GLIB

Common

| | |
|---|---|
| Class: | C/C++ Function Breakpoint |
| Type: | Regular |
| Function name: | mexFunction |
| ☑ Enabled | |
| Condition: | |
| Ignore count: | 0 |

```
mex_tst01 Debug [C/C++ Application] MATLAB
Start @ 08-Jun-2016 23:08:17
***************************************
MEX debug configuration.
MEX Call 01 result:
    1.2000    5.6000
    3.4000    7.8000


MEX FILE: /home/tkelin/sandbox/mex_tst01/mat/../mex/Debug/myFirstMex.mexa64 entry point located at address 0xb75fd75e
Add breakpoints at the debugger prompt and issue a "continue" to resume
execution of MATLAB.
```

GDB stops at the first executable line of out mex-function. Now we have access to all the features of the source-level debugger.



```
Thread #3 [MATLAB] 31079 [core: 1] (Suspended : Breakpoint)
    mexFunction() at myFirstMex.c:13 0x7fffb75fd774
    mexRunMexFile() at 0x7fffde82b8b0
    0x7fffde828214
    0x7fffde828db4
    Mfh_file::dispatch_fh_impl() at 0x7fffdf5d2505
```

[function: mexFunction]

No details to display for the current selection.

myFirstMex.c    pthread_cond_wait@@GLIBC_2.3.2() at 0x7ffff5d14b10

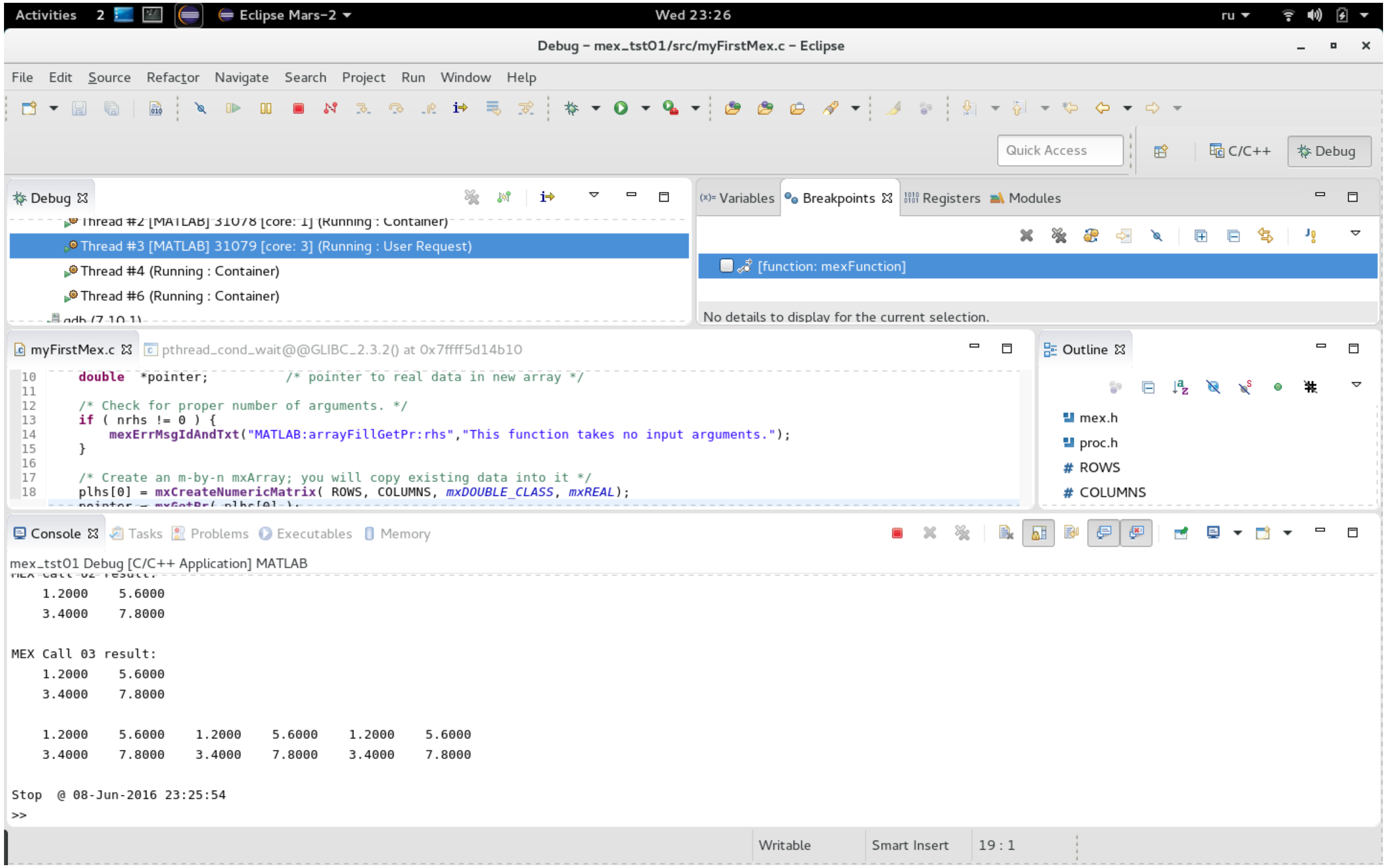
```

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]) {
    double  *pointer;          /* pointer to real data in new array */

    /* Check for proper number of arguments. */
    if ( nrhs != 0 ) {
        mexErrMsgIdAndTxt("MATLAB:arrayFillGetPr:rhs","This function takes no input arguments.");
    }

    /* Create an m-by-n mxArray; you will copy existing data into it */
    plhs[0] = mxCreateNumericMatrix( ROWS, COLUMNS, mxDOUBLE_CLASS, mxREAL);
    pointer = mxGetPr( plhs[0] );

```

mex.h
proc.h
ROWS
COLUMNS
ELEMENTS
mexFunction(int, mxArray*[], int, const mxArray

mex_tst01 Debug [C/C++ Application] gdb (7.10.1)

```
Missing separate debuginfo for /usr/local/MATLAB/R2016a/bin/glnxa64/libmwasyncioimpl.so
Missing separate debuginfo for /usr/local/MATLAB/R2016a/bin/glnxa64/libmwtamutil.so
Missing separate debuginfo for /usr/local/MATLAB/R2016a/bin/glnxa64/libmwmatlabproxy.so
Missing separate debuginfo for /usr/local/MATLAB/R2016a/bin/glnxa64/libmwasynciocore.so
Missing separate debuginfo for /usr/local/MATLAB/R2016a/toolbox/shared/hotpluglib/bin/glnxa64/libmwhotplugmlconverter.so
Missing separate debuginfo for /usr/local/MATLAB/R2016a/toolbox/shared/hotpluglib/bin/glnxa64/libmwhotplugdevice.so
Missing separate debuginfo for /usr/local/MATLAB/R2016a/bin/glnxa64/libmwcontainers.so
Missing separate debuginfo for /usr/local/MATLAB/R2016a/bin/glnxa64/libmwformat.so
[New Thread 0x7fffb7fff700 (LWP 31082)]
```

After debugging the source we deactivate the breakpoint at the start of mexFunction() and continue: Run → Resume. In the Console output window we can see the result of the script execution.



Thread #2 [MATLAB] 31078 [core: 1] (Running : Container)
Thread #3 [MATLAB] 31079 [core: 3] (Running : User Request)
Thread #4 (Running : Container)
Thread #6 (Running : Container)
gdb (7.10.1)

[function: mexFunction]

No details to display for the current selection.

myFirstMex.c
pthread_cond_wait@@GLIBC_2.3.2() at 0x7ffff5d14b10

```c
    double  *pointer;          /* pointer to real data in new array */

    /* Check for proper number of arguments. */
    if ( nrhs != 0 ) {
        mexErrMsgIdAndTxt("MATLAB:arrayFillGetPr:rhs","This function takes no input arguments.");
    }

    /* Create an m-by-n mxArray; you will copy existing data into it */
    plhs[0] = mxCreateNumericMatrix( ROWS, COLUMNS, mxDOUBLE_CLASS, mxREAL);
```

mex.h
proc.h
ROWS
COLUMNS

mex_tst01 Debug [C/C++ Application] MATLAB

```
    1.2000    5.6000
    3.4000    7.8000

MEX Call 03 result:
    1.2000    5.6000
    3.4000    7.8000

    1.2000    5.6000    1.2000    5.6000    1.2000    5.6000
    3.4000    7.8000    3.4000    7.8000    3.4000    7.8000

Stop  @ 08-Jun-2016 23:25:54
>>
```

Note:
During the debug session MATLAB runs without JVM. This is why the functions which use Java become unavailable, for example plotting functions. Avoid using these functions while debugging by testing MATLAB_DEBUG environment variable in the script.

T.K. June 2016