How to get started in hacking OpenJDK?

Vineel Kumar Reddy Kovvuri
http://vineelkumarreddy.com

Contents

1 Introduction

This blog post is not about internals of JDK(Java Development Kit), But a mere documentation for people who want to get started. JDK is open sourced a long time ago. The reference implementation of Java is now based on OpenJDK. So any one interested in it can get the source code and play with it. Here on, when I refer to JDK it means all the components including Java class libraries/JVM(Hotspot)/Java Compiler. In this post, I will walk you in getting and building the latest JDK 9 sources.

2 How do I get the source code?

All the components of JDK 9 are from http://hg.openjdk.java.net/jdk9/ . OpenJDK is under Mercurial. For any one familiar with Git, this is no different. Below is the layout of repositories in JDK 9 OpenJDK forest.

1. dev - top-level umbrella project for all the sub projects
   (a) corba - Not my interest
   (b) hotspot - The Java Virtual Machine implementation. JIT/GC go here
   (c) jaxp - Not my interest
   (d) jaxws - Not my interest
   (e) jdk - The Java class libraries including native JNI implementation go here
   (f) langtools - The Java compiler and other tools go here
   (g) nashorn - Not my interest

There are multiple ways to get the code. The easiest being, first clone the umbrella 'dev' project and run get_source.sh to pull all other repositories.

hg clone http://hg.openjdk.java.net/jdk9/dev 9dev

cd 9dev

sh ./get_source.sh

Once done you should see below directory structure.

```
vineel@Z97X:~/9dev$ tree -L 1
.
├── [1.5K]  ASSEMBLY_EXCEPTION
├── [4.0K]  build
├── [4.0K]  common
├── [1.6K]  configure
├── [4.0K]  corba
├── [3.0K]  get_source.sh
├── [4.0K]  hacking
├── [4.0K]  .hg
├── [ 103]  .hgignore
├── [ 17K]  .hgtags
├── [4.0K]  hotspot
├── [4.0K]  jaxp
├── [4.0K]  jaxws
├── [4.0K]  .jcheck
├── [4.0K]  jdk
├── [4.0K]  langtools
├── [ 19K]  LICENSE
├── [4.0K]  make
├── [2.7K]  Makefile
├── [ 53K]  modules.xml
├── [4.0K]  nashorn
├── [1.5K]  README
├── [ 57K]  README-builds.html
├── [ 50K]  README-builds.md
├── [4.0K]  test
└── [174K]  THIRD_PARTY_README
```

Figure 1

3 How do I build it?

Building OpenJDK is straight forward too, The only precursor is to get the dependencies. The OpenJDKs Adopt OpenJDK project has already documented the dependencies here, Its a matter of sudo apt-get install . Run below commands to build the source code.

bash ./configure #Setup the environment
make all #build the entire forest

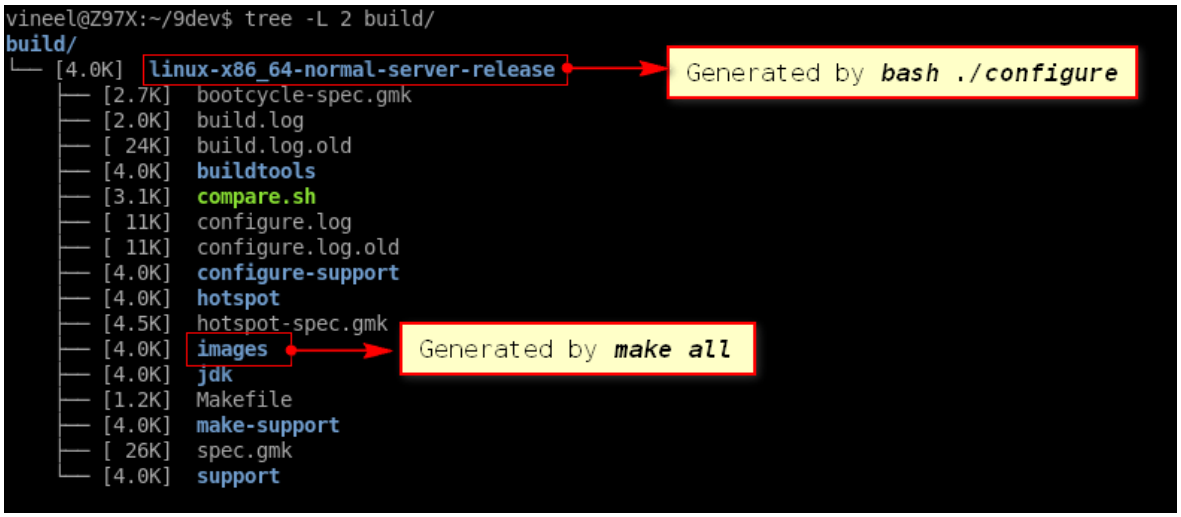

Figure 2: OpenJDK build directory after running bash ./configure

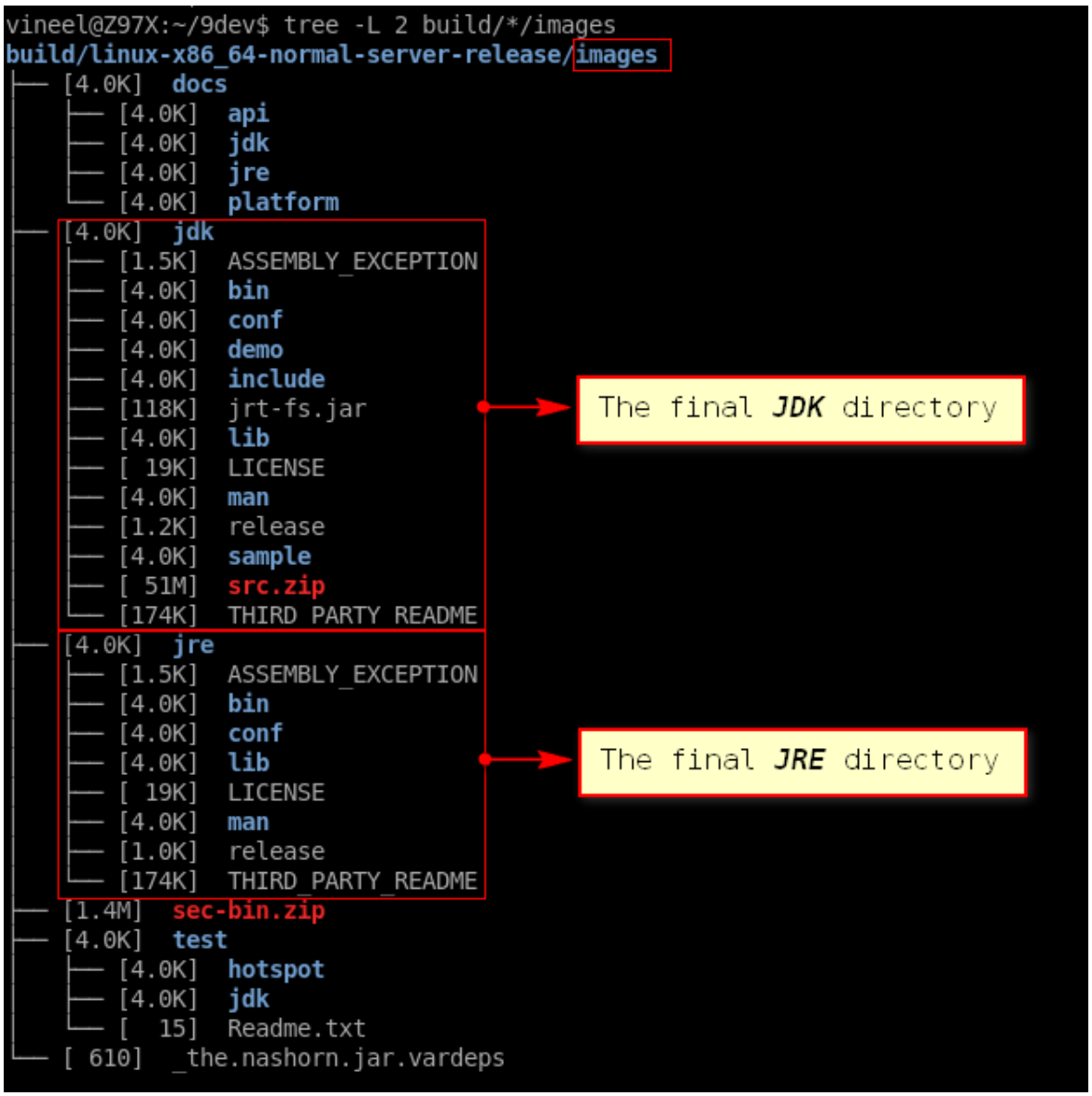

Figure 3: OpenJDK images directory after running make all

4 How do I hack it?

Now its all yours! This is where the fun begins, Even though OpenJDK sources are highly organized, you might need the help of some tools to get around with it. My favorite tool of choice for any source code exploration is OpenGrok, I have my local OpenGrok instance setup.

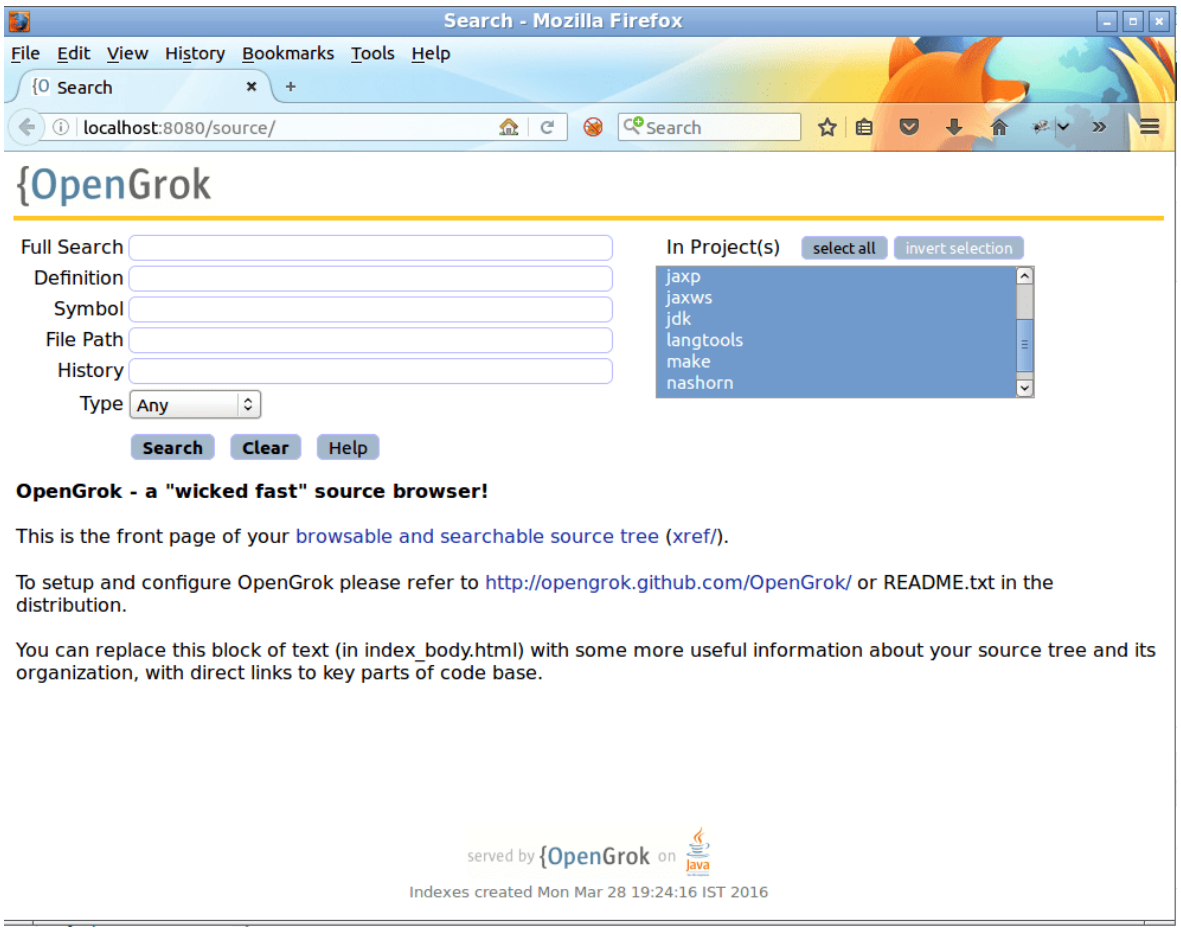

Figure 4

Since images/jdk/bin/java is the binary which executes your class files. I wanted to locate the code for it, so that, I can see where the rabbit hole is leading to(essentially in to hotspot) :), The right way to find out is to decipher the build system to understand how things are getting build and from where they are getting build. I really really! dont want to explore the build system now. So, Its time to wear my black hat and roll up some reverse engineering skills!

With the help of objdump I disassembled the jdk/bin/java binary and looked for symbols, specifically in its main function. Once I know some candidate symbols, I can get to the code from OpenGrok. So below is the disassembled view of main function of the java binary. This function is in turn making calls to JLI_* apis. So I selected JLI_PreprocessArg as my candidate symbol.

```
vineel@Z97X:~/9dev$ objdump -d build/*/images/jdk/bin/java | grep -i -A30 "<main>"
0000000000400880 <main>:
  400880:       55                      push   %rbp
  400881:       48 89 e5                mov    %rsp,%rbp
  400884:       41 57                   push   %r15
  400886:       41 56                   push   %r14
  400888:       41 55                   push   %r13
  40088a:       41 54                   push   %r12
  40088c:       49 89 f6                mov    %rsi,%r14
  40088f:       53                      push   %rbx
  400890:       31 f6                   xor    %esi,%esi
  400892:       89 fb                   mov    %edi,%ebx
  400894:       bf 01 00 00 00          mov    $0x1,%edi
  400899:       48 83 ec 18             sub    $0x18,%rsp
  40089d:       e8 3e ff ff ff          callq  4007e0 <JLI_InitArgProcessing@plt>
  4008a2:       8d 7b 01                lea    0x1(%rbx),%edi
  4008a5:       48 63 ff                movslq %edi,%rdi
  4008a8:       e8 63 ff ff ff          callq  400810 <JLI_List_new@plt>
  4008ad:       85 db                   test   %ebx,%ebx
  4008af:       49 89 c5                mov    %rax,%r13
  4008b2:       7e 6e                   jle    400922 <main+0xa2>
  4008b4:       8d 43 ff                lea    -0x1(%rbx),%eax
  4008b7:       49 8d 44 c6 08          lea    0x8(%r14,%rax,8),%rax
  4008bc:       48 89 45 c8             mov    %rax,-0x38(%rbp)
  4008c0:       49 8b 3e                mov    (%r14),%rdi
  4008c3:       e8 28 ff ff ff          callq  4007f0 <JLI_PreprocessArg@plt>
  4008c8:       48 85 c0                test   %rax,%rax
  4008cb:       49 89 c7                mov    %rax,%r15
  4008ce:       0f 84 bd 00 00 00       je     400991 <main+0x111>
  4008d4:       48 8b 50 08             mov    0x8(%rax),%rdx
  4008d8:       85 d2                   test   %edx,%edx
  4008da:       7e 2c                   jle    400908 <main+0x88>
vineel@Z97X:~/9dev$
```

Disassembly of ***main*** function in ***images/jdk/bin/java*** binary

This symbol looks promising!

Figure 5

Now searching for the symbol JLI_PreprocessArg in OpenGrok found the source code for jdk/bin/java!

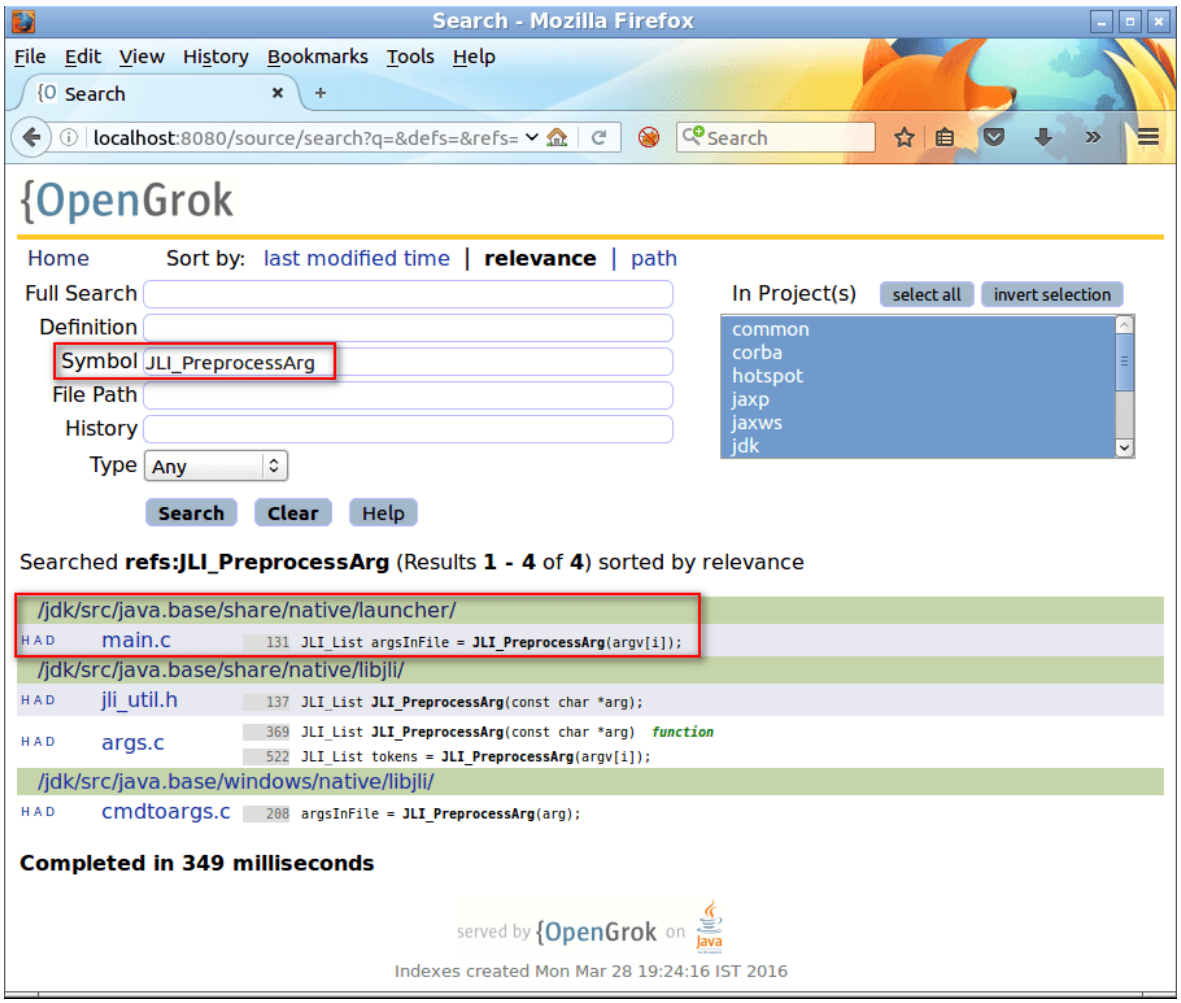
Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Search
localhost:8080/source/search?q=&defs=&refs=
Search
{OpenGrok
Home
Sort by: last modified time | relevance | path
Full Search
Definition
Symbol JLI_PreprocessArg
File Path
History
Type Any
Search Clear Help
In Project(s)
select all
invert selection
common
corba
hotspot
jaxp
jaxws
jdk
Searched refs:JLI_PreprocessArg (Results 1 - 4 of 4) sorted by relevance
/jdk/src/java.base/share/native/launcher/
HAD main.c 131 JLI_List argsInFile = JLI_PreprocessArg(argv[i]);
/jdk/src/java.base/share/native/libjli/
HAD jli_util.h 137 JLI_List JLI_PreprocessArg(const char *arg);
HAD args.c 369 JLI_List JLI_PreprocessArg(const char *arg) function
522 JLI_List tokens = JLI_PreprocessArg(argv[i]);
/jdk/src/java.base/windows/native/libjli/
HAD cmdtoargs.c 208 argsInFile = JLI_PreprocessArg(arg);
Completed in 349 milliseconds
served by {OpenGrok on Java
Indexes created Mon Mar 28 19:24:16 IST 2016

Figure 6

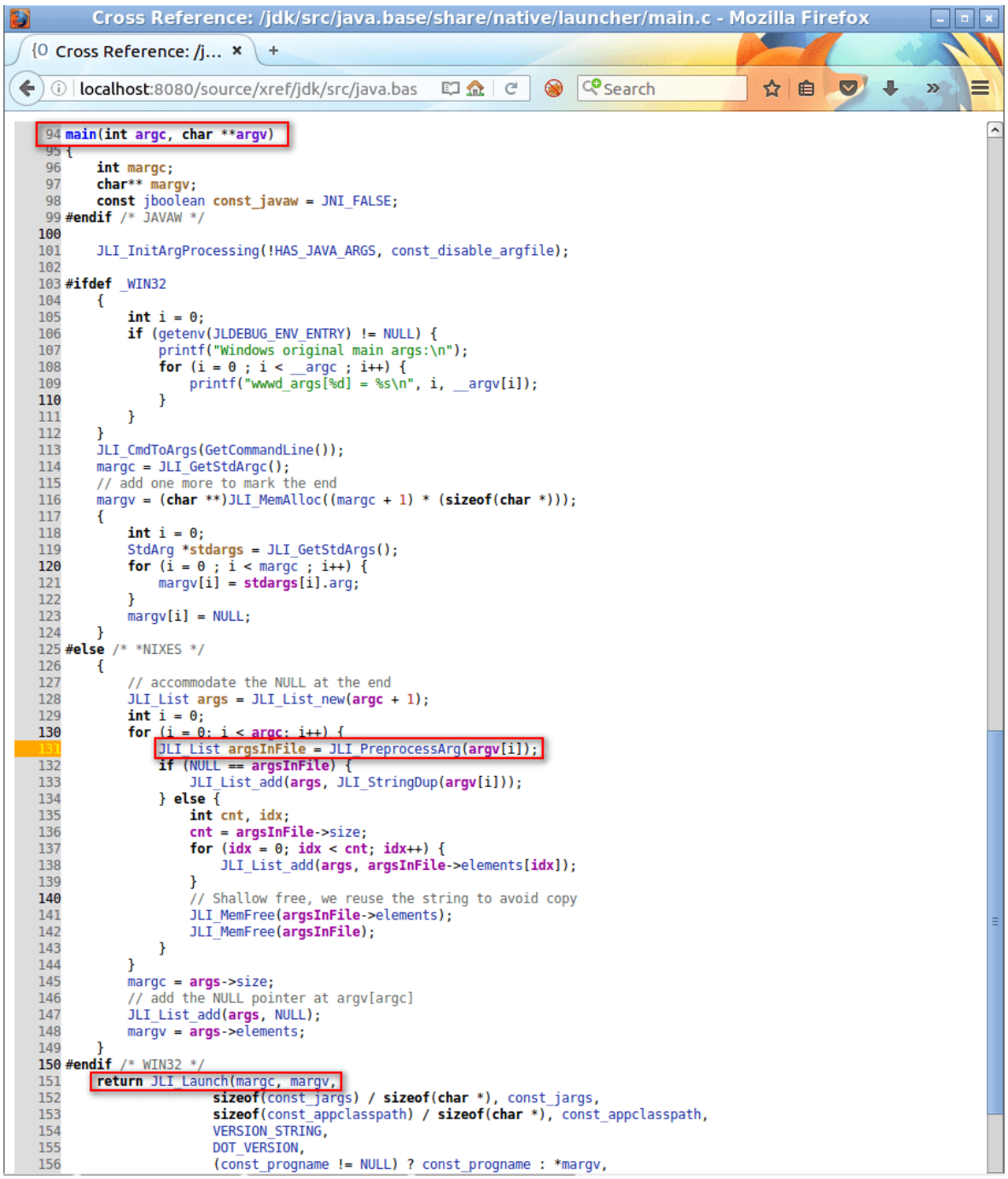

Figure 7: Inside main.c

To double confirm, I have added a print message and did the make all and it indeed worked as expected! You could see my print when I execute java from the command line in the right side window.

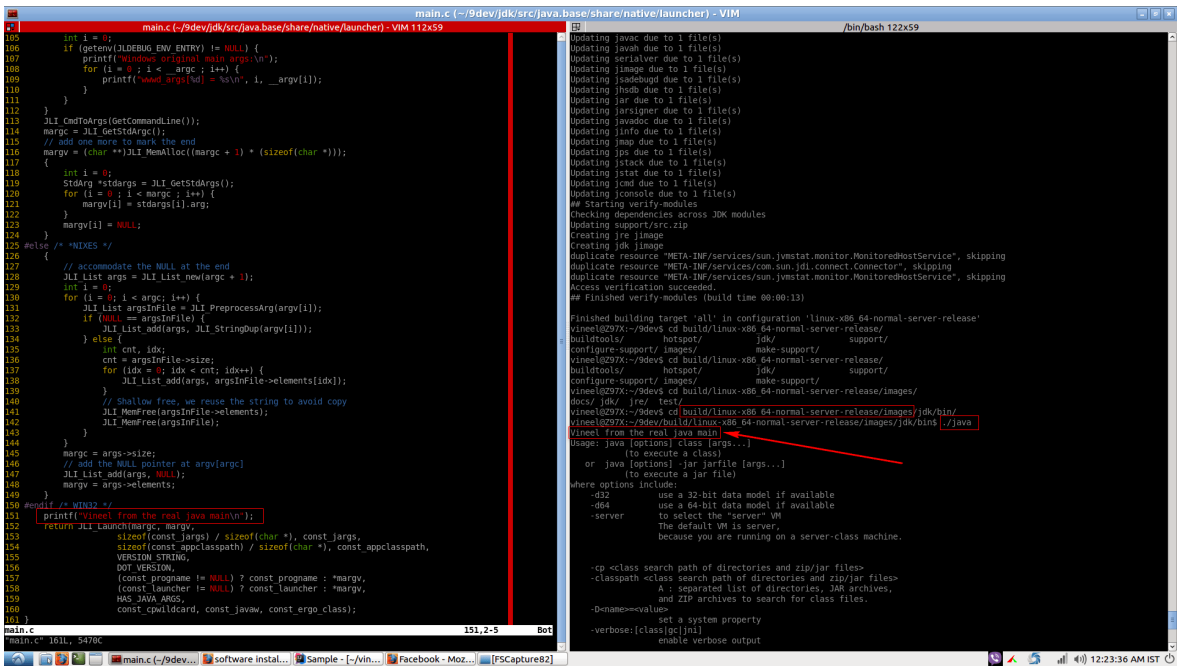

Figure 8: Inside main.c

5 References

1. The OpenJDK Developers' Guide - http://openjdk.java.net/guide/