Creating and Using a JNI Shared Library on Ubuntu (Detailed Guide)

This document explains how to create a Dynamic Link Library (shared object) for mathematical operations
in C, and how to use it from Java via the Java Native Interface (JNI) on Ubuntu Linux.

---

1. Write the Java Program (NativeMath.java)

```
public class NativeMath {
    static {
        System.loadLibrary("mathops"); // Loads libmathops.so
    }

    public static native int addNative(int a, int b);

    public static void main(String[] args) {
        int a = 10, b = 25;
        int result = addNative(a, b);
        System.out.println(a + " + " + b + " = " + result);
    }
}
```

Save this as: NativeMath.java

---

2. Compile and Generate JNI Header

In the terminal, navigate to your file directory and run:

```
javac -h . NativeMath.java
```

This will create:

- NativeMath.class
- NativeMath.h

---

3. Write the C Implementation (mathops.c)

```
#include <jni.h>
#include "NativeMath.h"

JNIEXPORT jint JNICALL Java_NativeMath_addNative(JNIEnv *env, jclass cls, jint a, jint b) {
    return a + b;
}
```

Save this as: mathops.c

---

4. Compile the Shared Library (.so)

Simplest version using JAVA_HOME environment variable:

```
gcc -fPIC -I"$JAVA_HOME/include" -I"$JAVA_HOME/include/linux" -shared -o libmathops.so mathops.c
```

This will produce a file named: libmathops.so

---

5. Run the Java Program

```
java -Djava.library.path=. NativeMath
```

Expected Output:

```
10 + 25 = 35
```

---

6. Verify the Library (optional)

You can inspect the shared object using:

```
nm -D libmathops.so | grep Java_
```

It should show a function like:

```
Java_NativeMath_addNative
```

---

7. Common Troubleshooting Tips

- UnsatisfiedLinkError:
  Java can't find libmathops.so. Make sure it's in the current directory and run with:
  `java -Djava.library.path=. NativeMath`

- Include path errors:
  Ensure correct JDK paths are included (default-java/include and default-java/include/linux).

- 32-bit/64-bit mismatch:
  Use matching versions of the JDK and GCC compiler.

---

8. Adding More Functions

You can easily extend mathops.c:

```
JNIEXPORT jint JNICALL Java_NativeMath_subNative(JNIEnv *env, jclass cls, jint a, jint b) {
    return a - b;
}
```

And add to Java:

```
public static native int subNative(int a, int b);
```

Quick Summary

```
# Build the shared library
gcc -fPIC -I"$JAVA_HOME/include" -I"$JAVA_HOME/include/linux" -shared -o libmathops.so mathops.c

# Run the program
java -Djava.library.path=. NativeMath
```

Output:

```
10 + 25 = 35
```

End of Document
