package com.twitter.ann.faiss;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

import java.nio.file.Files;

import java.nio.file.StandardCopyOption;

import java.util.Locale;

public final class NativeUtils {

private static final int MIN\_PREFIX\_LENGTH = 3;

public static final String NATIVE\_FOLDER\_PATH\_PREFIX = "nativeutils";

public static File temporaryDir;

private NativeUtils() {

}

private static File unpackLibraryFromJarInternal(String path) throws IOException {

if (null == path || !path.startsWith("/")) {

throw new IllegalArgumentException("The path has to be absolute (start with '/').");

}

String[] parts = path.split("/");

String filename = (parts.length > 1) ? parts[parts.length - 1] : null;

if (filename == null || filename.length() < MIN\_PREFIX\_LENGTH) {

throw new IllegalArgumentException("The filename has to be at least 3 characters long.");

}

if (temporaryDir == null) {

temporaryDir = createTempDirectory(NATIVE\_FOLDER\_PATH\_PREFIX);

temporaryDir.deleteOnExit();

}

File temp = new File(temporaryDir, filename);

try (InputStream is = NativeUtils.class.getResourceAsStream(path)) {

Files.copy(is, temp.toPath(), StandardCopyOption.REPLACE\_EXISTING);

} catch (IOException e) {

temp.delete();

throw e;

} catch (NullPointerException e) {

temp.delete();

throw new FileNotFoundException("File " + path + " was not found inside JAR.");

}

return temp;

}

/\*\*

\* Unpack library from JAR into temporary path

\*

\* @param path The path of file inside JAR as absolute path (beginning with

\* '/'), e.g. /package/File.ext

\* @throws IOException If temporary file creation or read/write

\* operation fails

\* @throws IllegalArgumentException If source file (param path) does not exist

\* @throws IllegalArgumentException If the path is not absolute or if the

\* filename is shorter than three characters

\* (restriction of

\* {@link File#createTempFile(java.lang.String, java.lang.String)}).

\* @throws FileNotFoundException If the file could not be found inside the

\* JAR.

\*/

public static void unpackLibraryFromJar(String path) throws IOException {

unpackLibraryFromJarInternal(path);

}

/\*\*

\* Loads library from current JAR archive

\* <p>

\* The file from JAR is copied into system temporary directory and then loaded.

\* The temporary file is deleted after

\* exiting.

\* Method uses String as filename because the pathname is "abstract", not

\* system-dependent.

\*

\* @param path The path of file inside JAR as absolute path (beginning with

\* '/'), e.g. /package/File.ext

\* @throws IOException If temporary file creation or read/write

\* operation fails

\* @throws IllegalArgumentException If source file (param path) does not exist

\* @throws IllegalArgumentException If the path is not absolute or if the

\* filename is shorter than three characters

\* (restriction of

\* {@link File#createTempFile(java.lang.String, java.lang.String)}).

\* @throws FileNotFoundException If the file could not be found inside the

\* JAR.

\*/

public static void loadLibraryFromJar(String path) throws IOException {

File temp = unpackLibraryFromJarInternal(path);

try (InputStream is = NativeUtils.class.getResourceAsStream(path)) {

Files.copy(is, temp.toPath(), StandardCopyOption.REPLACE\_EXISTING);

} catch (IOException e) {

temp.delete();

throw e;

} catch (NullPointerException e) {

temp.delete();

throw new FileNotFoundException("File " + path + " was not found inside JAR.");

}

try {

System.load(temp.getAbsolutePath());

} finally {

temp.deleteOnExit();

}

}

private static File createTempDirectory(String prefix) throws IOException {

String tempDir = System.getProperty("java.io.tmpdir");

File generatedDir = new File(tempDir, prefix + System.nanoTime());

if (!generatedDir.mkdir()) {

throw new IOException("Failed to create temp directory " + generatedDir.getName());

}

return generatedDir;

}

public enum OSType {

Windows, MacOS, Linux, Other

}

protected static OSType detectedOS;

/\*\*

\* detect the operating system from the os.name System property and cache

\* the result

\*

\* @returns - the operating system detected

\*/

public static OSType getOperatingSystemType() {

if (detectedOS == null) {

String osname = System.getProperty("os.name", "generic").toLowerCase(Locale.ENGLISH);

if ((osname.contains("mac")) || (osname.contains("darwin"))) {

detectedOS = OSType.MacOS;

} else if (osname.contains("win")) {

detectedOS = OSType.Windows;

} else if (osname.contains("nux")) {

detectedOS = OSType.Linux;

} else {

detectedOS = OSType.Other;

}

}

return detectedOS;

}

}