TypesMethodsAndFields

Jump to bottom

Ben Gruver edited this page on Apr 1, 2015 · 1 revision

Types

dalvik's bytecode has two major classes of types, primitive types and reference types. Reference types are objects and arrays, everything else is a primitive.

Primitives are represented by a single letter. I didn't come up with these abbreviations - they are what is actually stored in the dex file, in string form. They are specified in the dex-format.html document (dalvik/docs/dex-format.html in the AOSP repository)

V	**void - can only be used for return types**
Z	boolean
B	byte
S	short
C	char
I	int
J	long (64 bits)
F	float
D	double (64 bits)

Objects take the form `Lpackage/name/ObjectName;` - where the leading L indicates that it is an object type, package/name/ is the package that the object is in, `ObjectName` is the name of the object, and ; denotes the end of the object name. This would be equivalent to `package.name.ObjectName` in java. Or for a more concrete example, `Ljava/lang/String;` is equivalent to `java.lang.String`

Arrays take the form `[I` - this would be an array of ints with a single dimension. i.e. `int[]` in java. For arrays with multiple dimensions, you simply add more `[` characters. `[[I` = `int[][]`, `[[[I` = `int[][][]`, etc. (Note: The maximum number of dimensions you can have is 255).

You can also have arrays of objects, `[Ljava/lang/String;` would be an array of Strings.

Methods

Methods are always specified in a very verbose form that includes the type that contains the method, the method name, the types of the parameters and the return type. All this information is required for the virtual machine to be able to find the correct method, and to be able to perform static analysis on the bytecode (for verification/optimization purposes)

They take the form

```
Lpackage/name/ObjectName;->MethodName(III)Z
```

In this example, you should recognize `Lpackage/name/ObjectName;` as a type. `MethodName` is obviously the name of the method. `(III)Z` is the method's signature. `III` are the parameters (in this case, 3 ints), and `Z` is the return type (bool).

The method parameters are listed one right after another, with no separators between them.

Here's a more complex example:

```
method(I[[IILjava/lang/String;
[Ljava/lang/Object;)Ljava/lang/String;
```

In java, this would be

```
String method(int, int[][], int, String, Object[])
```

Fields

Fields are likewise always specified in verbose form that includes the type that contains the field, the name of the field, and the type of the field. Again, this is to allow the virtual machine to be able to find the correct field, as well as to perform static analysis on the bytecode.

They take the form

```
Lpackage/name/ObjectName;->FieldName:Ljava/lang/String;
```

This should be pretty self-explanatory - it is the package name, the field name and the type of the field respectively.

DeodexInstructions

Registers

SmaliBaksmali2.2

SmaliBaksmali20

smalidea

TypesMethodsAndFields

UnresolvableOdexInstruction

Clone this wiki locally

https://github.com/JesusFreke/smali.wiki.git