A deep dive into DEX file format

Rodrigo Chiossi

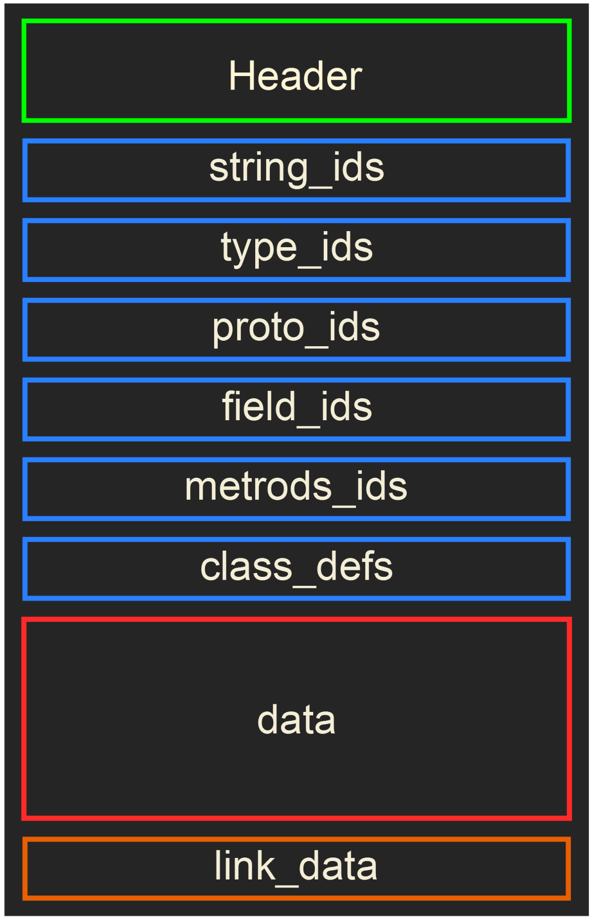
Bio

- Rodrigo Chiossi
  - Android Engineer @ Intel OTC
  - AndroidXRef
    - www.androidxref.com
  - Dexterity
    - https://github.com/rchiossi/dexterity

Overview

- DEX File Structure
  - Characteristics
  - LEB128
  - Relative Indexing
  - MUTF-8
  - The “Big” Header and the data.
- DEX Instrumentation
  - The “String Add” case
- DEX Limitations
  - Bitness restrictions

DEX Structure



Header

string_ids

type_ids

proto_ids

field_ids

metrods_ids

class_defs

data

link_data

Header :

magic	ubyte[8]
checksum	uint
signature	ubyte[20]
file_size	uint
header_size	uint
endian_tag	uint
link_size	uint
link_off	uint
map_off	uint
string_ids_size	uint
string_ids_off	uint
type_ids_size	uint
type_ids_off	uint
proto_ids_size	uint
proto_ids_off	uint
field_ids_size	uint
field_ids_off	uint
method_ids_size	uint
method_ids_off	uint
class_defs_size	uint
class_defs_off	uint
data_size	uint
data_off	uint

map_item:

type	ushort
unused	ushort
size	uint
offset	uint

Type Codes:

HEADER_ITEM	0x0000
STRING_ID_ITEM	0x0001
TYPE_ID_ITEM	0x0002
PROTO_ID_ITEM	0x0003
FIELD_ID_ITEM	0x0004
METHOD_ID_ITEM	0x0005
CLASS_DEF_ITEM	0x0006
MAP_LIST	0x1000
TYPE_LIST	0x1001
ANNOTATION_SET_REF_LIST	0x1002
ANNOTATION_SET_ITEM	0x1003
CLASS_DATA_ITEM	0x2000
CODE_ITEM	0x2001
STRING_DATA_ITEM	0x2002
DEBUG_INFO_ITEM	0x2003
ANNOTATION_ITEM	0x2004
ENCODED_ARRAY_ITEM	0x2005
ANNOTATIONS_DIRECTORY_ITEM	0x2006

DEX Properties

- Reduced Memory Footprint
  - LEB128 encoding
  - Relative Indexing
  - Single file for all classes (vs. 1 file per class in .class format)
  - No duplicate strings
- Modified UTF-8 String Encoding
- Strict requirements for alignment
- Even more strict runtime verifier (DexOpt)

LEB128

- Encoding format from DWARF3.
- Used to encode signed (SLEB128 and ULEB128p1) and unsigned (ULEB128) numbers.
- Used in DEX for encoding 32-bit numbers.
- **Numbers are encoded using 1 to 5 bytes.**
  - Depending on the highest **‘1’**-bit

LEB128 - Example

HEX	BIN	SLEB128	ULEB128	ULEB128p1
00	00000000	0	0	-1
01	00000001	1	1	0
7f	011111111	-1	127	126
80 7f	10000000 011111111	-128	16256	16255

- -1 is used to represent the NO_INDEX value.
- Encoded as ULEB128p1, NO_INDEX requires only one byte to be encoded.

Relative Indexing

- Many DEX objects are represented by its index into a list.
- Encoded object lists use that index value as representation for the first object and diffs for representing the rest of the list.
- Using the delta usually yields smaller numbers with smaller representation in bytes when LEB128 is used.
- Ex:
  - In **class_data_item** structure, **static_fields**, **instance_fields**, **direct_methods** and **virtual_methods** are all represented by the index delta.

Relative Indexing - Example

Field ID	Field Name
...	
1024	field_1
1025	field_2
...	
1036	field_3
...	

- Field List:
  - Field_1, field_2, field_3
- Encoding:
  - 1024, 1, 11

Modified UTF-8

- Used for encoding all strings in the DEX format.
- Characters may have 1, 2 or 3 bytes.
- Strings are terminated by a single null byte.
- When parsing string_data_item, the uft16_size field cannot be used to calculate the size of the following data as it only represents the number of characters in the MUTF-8 string.
- ASCII strings are MUTF-8 legal strings

The “Big Header”

- Besides the header_item, we have six other structures that describe the DEX file:
  - string_id_item list
  - type_id_item list
  - proto_id_item list
  - field_id_item list
  - method_id_item list
  - class_def_item list
- This structures define all the functional content of the DEX file.

The Map

- The DEX file may contain an optional structure called the Map, composed by map_item structures.
- The Map structure contains information about all the offsets in the file and what is the type of content in that offset.
- **Although optional according to the file format specification, the existence and correctness of the map is enforced by DexOpt.**

The Data

- All the content of the DEX file not in the “big header” goes to the Data area.
- Offsets to structures in the data area must be bigger than the end of the “big header”. This property is enforced by DexOpt.
- It is ok to have gaps in the middle of the data section.
- The map is part of the data area.

The Link Data

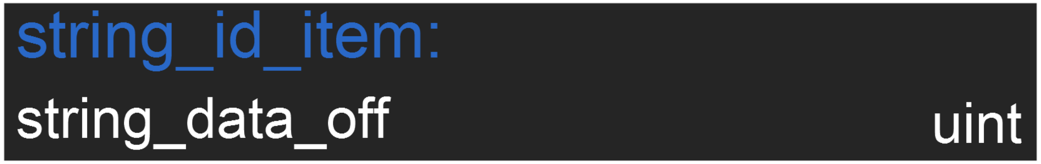
- Optional area at the end of the Data area.
- Format unspecified.
- Never present in “Normal” apks.

DEX Instrumentation

- Case Study: String add
  - String manipulation is required for most obfuscation/deobfuscation techniques.
  - Can be extended for replacing and removing strings.
- Objective:
  - Keep the DEX valid after adding the new string.
  - Pass DexOpt checking.

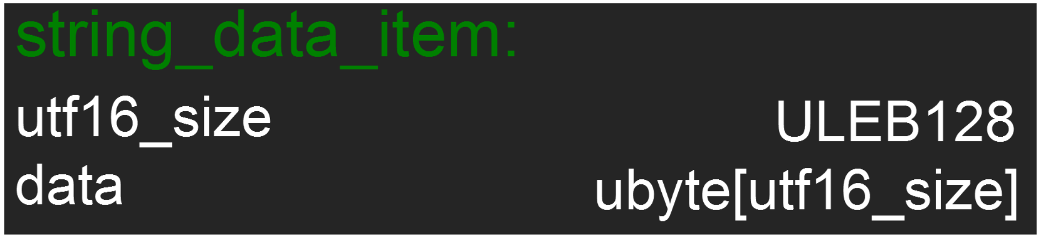
String Structure

- Represented by the pair (**string_id_item**, **string_data_item**)
- **string_id_item** list must be sorted
  - Sorted by the utf16 code points of the string
- Strings are referenced by its index position in the **string_id_item** list.



string_id_item:

string_data_off	uint



string_data_item:

utf16_size	ULEB128
data	ubyte[utf16_size]

Adding a string_id_item

- Must be added in the position of the list that will keep the list sorted.
- Header adjustments:
  - Data offset.
  - File size.
- Maps adjustments:
  - **string_id_item** map size.
- Entire file adjustments:
  - Offsets references in data area must be shifted 4 bytes.
  - String references equal or bigger than the added string must be increased by 1.

LEB128 Expansion

- Some offsets are encoded as ULEB128.
  - E.g. **code_off** inside **encoded_method** object.
- Some string_id_item references are encoded as ULEB128.
  - E.g. **name_idx** inside **annotation_element** object.
- After shifting offsets or increasing **string_id_item** references, the size of the LEB128 in bytes may increase.
- If the expansion occurs, further shifting of offsets is needed in the file.
- Maps size and offset must be updated.

Alignment

- Some structures in the DEX file must be 4-byte aligned.
  - E.g., **code_item**.
- **string_id_item** is 4-byte in size, so adding a new object will not misalign the DEX.
- LEB128 expansion will often add 1 byte shifting, which will break alignment.
- If realignment is required, offset references must be updated.
- Maps size and offset must be updated.

Adding a string_data_item

- Must be inside the data area.
- Header adjustments:
  - Data size.
  - File size.
- Maps adjustments:
  - **string_data_item** map size.
- Entire file adjustments:
  - Offsets references after the offset of the new **string_data_item** must be shifted by the size of the added object.
  - String references equal or bigger than the added string must be increased by 1.
- Check for LEB128 expansion and apply shifting.
- Check for alignment and apply shifting.

DEX Bit Restrictions

- 32 bits encoding
  - Static fields with fixed 32 bit size (E.g. string_id_item).
  - Offsets expected to be within 32 bit range.
- Less than 32 bits encoding
  - Class, type, proto and other lists alike are limited to 16 bits in size.

Rodrigo Chiossi

r.chiossi@androidxref.com

@rchiossi