ENVLANG 파일 포맷 v0.2

정 원교

2006.2.25

지구 남쪽에서

이메일 : weongyo@gmail.org

요 약

이 문서는 ENVLANG Compiler (아직 완전한 컴파일러가 존재하지 않습니다만) 에 의해 생성되는 바이너리이며, ROVM (원격 객체 가상 머신) 에서 사용하기 될 “ENVLANG File Format” 에 대한 전체적인 구조에 대해서 설명한 문서입니다.

차 례

제 1 절 소개

ENVLANG File Format 은 에랑 프로젝트의 한 부분인 ROVM (Remote Object Virtual Machine : 원격 객체 가상 머신, 이하 ROVM) 에서 사용하기 위해서 만들어 졌습니다.

파일 포맷을 구조화하면서 가장 중점에 둔 부분은 '확장성'인데, 그 이유로는 ROVM 에서 필요로 하고 지원하는 opcode 와 Class 에 대한 개념이 매우 소규모이기 때문에, ROVM 이 업데이트되면서 조금씩 파일 포맷 또한 업데이트할 수 있는 구조를 원하였고, 또한 가장 기초적인 부분부터 조금씩 탑을 쌓아올리려는 개발자의 생각이 아래 3 절에서 설명하게 되는 파일 포맷 구조에 많은 영향을 주었습니다.

파일 포맷의 전체적인 구조는 Java File Format 으로 부터 아이디어를 얻었으며, 많은 부분에서 유사할 수도 있습니다.

제 2 절 변경 사항 기록

이 절에서는 그 동안 ENVLANG File Format 의 변경 내용에 대해서 기록한 곳입니다.

2.1 v0.2 (2005 년 2월 12일)

아래와 같은 내용이 추가, 변경되었습니다.

- Field
  클래스에 속하게 되는 field 에 대한 개념이 새롭게 추가되었습니다. 이를 통해서 좀 더 클래스 모습에 가까워지게 되었습니다.
- MethodCode 속성에 max local 개념 추가
  이전 파일 포맷에는 존재하지 않았지만, 사용자 메쏘드에서의 변수 사용을 돕기 위해서 새롭게 추가되었습니다.
  이 변경 부분으로 인해 기존 v0.1 파일 포맷과는 호환이 되지 않습니다.

2.2 v0.1 (2005 년 2월 4일)

처음 파일 포맷에 대한 구조가 생성되었습니다.

- Class
  하나의 파일에 하나의 클래스만 기록할 수 있는 파일 포맷의 한계점이 있습니다. 이것은 'ClassName 속성'의 위치를 명확하게 하지 못한데에서 그 원인이 있습니다.
- Method (혹은 Function) 는 항상 Class 에 포함되어야 하며, Class 밖에서는 구현을 못합니다.
- Field 의 개념은 존재하지 않습니다. 현재 테스트를 진행하고자 하는 부분에서는 Field 개념이 필요하지 않기 때문입니다.
- Class 의 상속 개념 또한 존재하지 않으며, Java 의 인터페이스 개념 또한 존재하지 않습니다.

제 3 절 ENVLANG File Format

각 파일 포맷에 대한 표현을 C 언어의 구조체로 표현을 하였습니다. C 언어의 초보 분들도 매우 쉽게 이해하실 수 있으리라 믿습니다. 그리고 이 절에서는 class 파일 정보를 표현하기 위한 data type 들을 가지고 있는데, Type *u1* 과 *u2*, *u4* 등은 unsigned 1 과 2, 4 바이트 용량의 단위를 나타냅니다.

그리고 'ENVLANG File Format'의 모든 정수형, 실수형 등 숫자와 관련된 'byte order' 는 반드시 '***little-endian***' 여야만 합니다. 이렇게 하는 이유는 ix86 에서 빠르게 실행할 수 있도록 하기 위한 목적이 큽니다.

자세한 파일 parsing 부분에 대해 보시길 원하시는 분들은 ROVM Server 소스의 $prefix/src/file.c 파일을 살펴보시기 바랍니다.

전체적인 파일 포맷의 구조는 v0.1 현재 아래와 같이 정의되어 있습니다.

```
struct envlang_fileformat
  {
    u4 magic;
    u2 fileformat_version;
    u4 made_time;
    u2 attribute_counts;
    attribute attributes[attribute_counts];
  };
```

위 구조체를 보면 아시겠지만, 이번 파일 포맷의 테마는 '**모든 것을 attribute 로 구성하자**' 입니다.

- magic
  파일 포맷의 magic number 로써 항상 값으로 "\x0e\x0aNG" 을 가집니다. 뜻은 없습니다. 원래는 값을 "ENVLANG" 으로 할려고 했는데, 길이가 6 바이트가 되는 바람에 4 바이트로 만든다고 6 자 중 마음에 드는 글자 4 자를 선택하고 그 값을 조금 변형시킨 것 뿐입니다.
- fileformat_version
  현재 바이너리의 파일 포맷 번호. v0.1 일 경우, 0x0001 로 표현이 됩니다. 만약 버전 1.0 일 경우, 0x0100 으로 표현이 될 것입니다.
- made_time
  현재 바이너리가 생성된 시간. 간단하게 설명해서 python 의 time 모듈 중 time () 함수가 반환하는 값이다. epoch 이루로 UTC 형태로 초로 계산된 값이다.
- attribute_counts
  현재 파일 포맷이 가지고 있는 속성들의 갯수이다.
- attributes
  위 attribute_counts 에 이 속성에 대해서는 3.1 에서 자세하게 다루게 될 것입니다. 현재 v0.1 파일 포맷은 모두 'Attribute (속성)' 들로 이루어져 있다고 할 수 있을 것입니다.

3.1 Attribute (속성)

모든 attribute 들의 구조는 아래와 같은 모습을 가지게 됩니다.

```
struct attribute
  {
    u1 attribute_type;
    u4 attribute_length;
    u1 attribute[attribute_length];
  };
```

하지만 *attribute_type* 에 따라서 *attribute* 에 사용되는 의미가 조금씩 달라지게 되는 것이 특징입니다.

attribute_type 는 u1 크기를 가지고 있기 때문에, 현재 파일 포맷을 이용할 경우, 256 개의 'Attribute (속성)'를 가질 수 있습니다.

현재 'Attribute (속성)'로는 아래와 같은 것들이 정의되어 있으며, 각각 마다 의미를 가지고 있습니다.

- ClassName
  현재 이 파일에서 정의되어 있는 클래스의 이름 정보를 가지고 있습니다. 이 속성의 경우, 이를 포함하는 다른 Attribute 가 아직 존재하지 않습니다. v0.1 버전에서는 항상 가장 윗단에 속하게 되어 있습니다.

  이로 인해서 현재 v0.1 포맷에서는 각 파일당 하나의 클래스만 저장할 수 있도록 구조화되게 된 것입니다.

  이 속성에 대한 정보는 3.1.1 장에서 자세히 설명되어 있습니다.

- Method
  이 속성은 하나의 method 를 표현하기 위한 속성입니다. 3.1.3 장에서 자세히 설명되어 있습니다.

- MethodCode
  이 속성은 "Method 속성"에 포함되는 하위 속성으로써, 해당 Method 의 Opcode 길이, Opcode 정보를 가지고 있습니다. 3.1.4 장에서 자세히 설명되어 있습니다.

3.1.1 ClassName 속성

클래스의 이름을 나타내는 속성입니다.

```
struct ClassName_attribute
  {
    u1 attribute_type;
    u4 attribute_length;
    u1 name_len;
    u1 name[name_len];
  };
```

- attribute_type
  이 속성의 Type 값으며, 항상 `0xFF` 값을 가지게 됩니다.

- attribute_length
  현재 아래로 포함하게 될 요소들의 총 길이를 의미합니다.

- name_len
  클래스 이름의 전체 길이를 의미합니다.

- name
  클래스 이름에 대한 정보가 *name_len* 길이만큼 저장되어 있습니다. 클래스 이름에 대한 encoding 은 'UTF-8' 로 지정되어 있습니다.

3.1.2 Field 속성

클래스 파일에 소속되는 하나의 field 를 표현할 수 있는 속성입니다.

```
struct Field_attribute
  {
    u1 attribute_type;
    u4 attribute_length;
    u2 RESERVED_SPACE;
    u1 name_len;
    u1 name[name_len];
    u2 type_len;
    u1 type[type_len];
    u2 attribute_counts;
    attribute attributes[attribute_counts];
  };
```

- attribute_type
  이 속성의 Type 값으며, 항상 `0xFC` 값을 가지게 됩니다.

- attribute_length
  현재 아래로 포함하게 될 요소들의 총 길이를 의미합니다.

- RESERVED_SPACE
  나중을 위해 예약된 공간입니다.

- name_len
  Field 의 이름 길이를 저장하고 있습니다.

- name
  Field 의 이름입니다. Field 이름에 대한 encoding 은 'UTF-8' 로 지정되어 있습니다.

- type_len
  Field 의 Type 길이를 저장하고 있습니다.

- type
  Field 의 Type 정보입니다. Field Type 에 대한 encoding 은 'UTF-8' 로 지정되어 있습니다.

- attribute_counts
  Field 가 포함하는 하위 'Attribute 속성'에 대한 갯수 정보입니다.

- attributes
  Field 가 포함하는 하위 'Attribute 속성' 정보입니다. 현재 Field 에 대한 하위 속성으로 등록될 수 있는 것은 없습니다.

3.1.3 Method 속성

클래스 파일에서 하나의 method 를 표현하기 위한 속성입니다.

```
struct Method_attribute
  {
    u1 attribute_type;
    u4 attribute_length;
    u4 RESERVED_SPACE;
    u1 name_len;
    u1 name[name_len];
    u2 type_len;
    u1 type[type_len];
    u2 attribute_counts;
    attribute attributes[attribute_counts];
  };
```

- attribute_type
  이 속성의 Type 값으며, 항상 `0xFE` 값을 가지게 됩니다.

- attribute_length
  현재 아래로 포함하게 될 요소들의 총 길이를 의미합니다.

- RESERVED_SPACE
  나중을 위해 예약된 공간입니다.

- name_len
  Method 의 이름 길이를 저장하고 있습니다.

- name
  Method 의 이름입니다. Method 이름에 대한 encoding 은 'UTF-8' 로 지정되어 있습니다.

- type_len
  Method 의 Type 길이를 저장하고 있습니다.

- type
  Method 의 Type 정보입니다. Method Type 에 대한 encoding 은 'UTF-8' 로 지정되어 있습니다. Type 에 대한 자세한 정보는 3.2 섹션을 살펴보시기 바랍니다.

- attribute_counts
  Method 가 포함하는 하위 'Attribute 속성'에 대한 갯수 정보입니다.

- attributes
  Method 가 포함하는 하위 'Attribute 속성' 정보입니다. 예를 들면, MethodCode 와 같은 속성들이 하위 속성으로 등록되어 있습니다.

3.1.4 MethodCode 속성

'Method 속성'에 대한 하위 속성으로써, 해당 Method 의 Code 정보들을 가지고 있습니다.

```
struct MethodCode_attribute
  {
    u1 attribute_type;
    u4 attribute_length;
    u2 RESERVED_SPACE;
    u2 max_locals;
    u4 code_length;
    u1 code[code_length];
    u2 RESERVED_SPACE;
    u2 attribute_counts;
    attribute attributes[attribute_counts];
  };
```

- attribute_type
  이 속성의 Type 값으며, 항상 `0xFD` 값을 가지게 됩니다.

- attribute_length
  현재 아래로 포함하게 될 요소들의 총 길이를 의미합니다.

- RESERVED_SPACE
  나중을 위해 예약된 공간입니다.

- max_locals
  v0.2 파일 포맷에서 새롭게 추가된 항목으로 메쏘드에서 내부적으로 사용할 로컬 변수들을 위한 공간을 확보할 수 있도록 도와주는 공간입니다.

- code_length
  현재 Method 의 Opcode 길이를 의미합니다.
- code
  현재 Method 의 Opcode 가 들어가 있는 공간입니다.
- RESERVED_SPACE
  나중의 파일 포맷을 위해 미리 예약해 놓았습니다.
- attribute_counts
  MethodCode 가 포함하는 하위 'Attribute 속성'에 대한 갯수 정보입니다.
- attributes
  MethodCode 가 포함하는 하위 'Attribute 속성' 정보입니다. 현재 파일 포맷 버전에서는 MethodCode 에 대한 하위 속성이 존재하지 않습니다.

3.2 Method 와 Field Type 에 대한 설명

Method 와 Field 의 Type 은 ENVLANG File Format 과 직접적으로 관련이 있기 보다는 이를 실제로 실행할 ROVM Server 와 관련이 깊다고 할 수 있겠습니다. 하지만 Method 와 Field 의 Type 또한 ENVLANG File Format 의 일원이기 때문에 문서의 한 공간을 차지하게 되었습니다.

현재 ROVM Server 의 경우, 공식적으로 10 개의 type 을 지원하고 있습니다. 아래의 표를 살펴보시기 바랍니다.

번호	이름	크기	문자
0	Void	없음	V
1	ArrayRef	주소형	[
2	Boolean	1 비트 boolean 형	B
3	Char	1 바이트 정수형	C
4	Short	2 바이트 정수형	H
5	Integer	4 바이트 정수형	I
6	Float	4 바이트 실수형	F
7	Double	8 바이트 실수형	D
8	ObjectRef	주소형	T
9	StringRef	주소형	S

위의 표에서 '번호'는 각 type 에 붙은 고유 번호입니다. '크기'의 경우, 실제 메모리상에서 차지하는 바이트 크기를 가르키며, '문자'의 경우 확장 모듈이나, Method 타입을 지정할 때 사용되는 character 를 의미합니다.

위의 각 항목에 대한 부연 설명을 한다면 아래와 같습니다.

- StringRef
  ObjectRef 와 flag 하나만 다를 뿐, 완전히 같은 type 입니다. 똑같은 ObjectRef 를 사용하지만 Type 의 번호는 다릅니다. 중심 클래스로는 '`/core/str`'[1] 를 사용합니다.

[1] ROVM Library 상에서 상대 위치를 의미합니다.

- ArrayRef
  ArrayRef 또한 ObjectRef 와 flag 하나만 다르지만, 번호는 다른 Type 입니다. '`/core/array`' 클래스가 그 중심이 되는 클래스이며, 배열 내에는 여러 개의 다른 type 들이 삽입될 수 있는 구조입니다.

Field 의 경우, 단순히 위의 '문자' 들 중 하나를 선택하시면 됩니다.

Method 의 형식은 위의 문자들의 조합으로 이루어질 수 있습니다. 하지만 아래와 같은 형식을 유지하셔야 합니다.

```
(<args>)<ret>
```

`<args>` 의 경우, 반드시 '`(`' 와 '`)`' 로 둘러쌓여져 있어야 하며, 이 곳은 메쏘드의 argument 를 넣는 곳입니다. `<ret>` 의 경우 메쏘드의 반환값을 넣는 곳입니다. 단 Void 타입은 argument 내에 위치할 수 없습니다.

위의 형식을 통해서 method 의 예를 들어 보겠습니다.

- `(TII)I`
  Argument 로써 3 개 항목을 넘기며, 반환값이 Integer 타입임을 의미합니다.
- `(T)V`
  Argument 로써 1 개 항목을 넘기며, 반환값은 '없음'을 의미하는 Void 타입입니다.
- `(TT)I`
  Argument 로써 2 개 항목을 넘기며, 반환값이 Integer 타입임을 의미합니다.
- `(TSI[)I`
  Argument 로써 4 개 항목을 넘기며, 반환값으로 Integer 를 넘기게 됩니다. 첫번째 항목은 this 이며, 두번째 항목은 StringRef, 세번째는 Integer, 네번째는 ArrayRef 입니다.