프로그램 모듈화를 위한 EVM 아카이브 파일 포맷의 설계

고효석[0], 이창환, 오세만
동국대학교 컴퓨터공학과
e-mail : {hsnation[0], yich, smoh}@dongguk.edu

Design of EVM Archive File Format for Program Modulization

Hyo-Seok Ko[0], Chang-Hwan Yi, Se-Man Oh
Dept. of Computer Engineering, Dongguk University

요 약

최근의 프로그램들은 크기가 커지고 기능이 복잡해짐에 따라 모듈화된 구조를 지니고 있다. 프로그램 모듈화는 프로그램의 설계, 제작, 유지보수, 코드의 재사용 부분에서 많은 장점을 가지며 라이브러리라는 개념을 통해 이루어진다. 현재 임베디드 기기를 위한 가상기계인 EVM은 라이브러리 개념의 모듈화를 지원하고 있지 않다. EVM에서 동작하는 프로그램의 모듈화를 도입하기 위해선 라이브러리를 나타내는 아카이브 파일 포맷이 요구되며, 아카이브 파일을 다루기 위한 아카이버와 링커 등의 도구가 필요하다.

본 논문에서는 EVM 프로그램의 모듈화를 위한 아카이브 파일 포맷을 설계하였다. 본 논문의 아카이브 파일 포맷은 유사한 개념의 아카이브 파일 포맷을 분석하여 기본구조를 설계하였으며, EVM 환경에 필요한 특징을 반영하였다. 아카이브 파일 포맷의 설계를 통하여 추후 EVM에서 동작하는 프로그램을 라이브러리화할 수 있는 기반을 만들었다. 이를 바탕으로 프로그램 모듈화를 완성할 것이다.

1. 서론

최근의 프로그램들은 기능이 다양해지고, 크기가 커짐에 따라 모듈화된 구조를 지니고 있다. 프로그램 모듈화란 전체 프로그램을 특정 기능 등의 단위별로 분류하여 작성하는 작업을 의미한다. 이를 통해 프로그램의 설계, 구현, 유지보수, 코드의 재사용, 분담 작업의 가능 등 많은 장점을 얻을 수 있다. 이러한 프로그램 모듈화의 장점은 가상기계 분야에 도입될 수 있다.

프로그램 모듈화를 위해서는 먼저 라이브러리라는 개념이 필요하다. 라이브러리 개념을 개발 과정에 도입하기 위해서는 라이브러리를 나타내는 아카이브 파일이 필요하다. 그리고 아카이브 파일을 다루는 아카이버, 그리고 링커의 도구가 지원됨으로서 프로그램 모듈화를 완성할 수 있다.

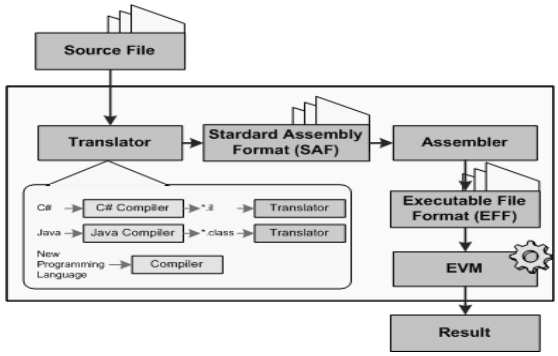
본 논문에서는 EVM(Embedded Virtual Machine)에서 수행되는 프로그램의 모듈화를 위한 첫 번째 단계인 아카이브 파일 포맷을 AFF(Archive File Format)라고 명명하여 설계하였다. 설계된 아카이브 파일 포맷은 생성된 아카이브 파일에서 일정 부분을 추가 및 제거할 수 있는 유연한 구조를 지니고, 기존의 EFF와 호환이 가능해야 한다. 또한, 아카이브 파일의 설계를 통하여 추후 아카이버를 구현하고, 이를 통해 EVM에서 수행되는 프로그램을 라이브러리화 할 수 있는 기반을 만들고자 한다.

2. 관련연구

2.1 EVM

스택 기반의 가상기계인 EVM은 PDA, 디지털 TV 등에 탑재되어 동적으로 응용프로그램을 다운로드하여 실행할 수 있다. (그림 1)은 EVM의 시스템

구성도를 나타낸 것이다[2][4].



(그림 1) EVM의 시스템 구성도

EVM의 중간 언어인 SIL(Standard Intermediate Language)은 스택 기반의 명령어 집합으로 언어 독립성과 하드웨어 및 플랫폼 독립성을 갖고 있다. SIL은 다양한 프로그래밍 언어를 수용하기 위해서 바이트코드, .NET IL 등 기존의 가상기계 코드들의 분석을 토대로 정의되었다. 또한, 프로그램의 확장성을 위해 순차적 언어와 객체지향 언어를 모두 수용할 수 있도록 설계되었다[1][9].

고급 언어로 작성된 코드는 변환기를 통해서 의사코드와 연산코드로 구성되고 클래스 기반의 구조를 가지는 EVM의 어셈블리 포맷인 SAF(Standard Assembly Format)로 변환된다[8].

SAF는 어셈블러를 통해 EVM 상에서 실행 가능한 EFF(Executable File Format) 파일로 변환된다.

EFF 파일의 전체 구조는 (그림 2)와 같다[3][7].

```
EFF_File {
    U4 magic;            // Header
    U2 majorVersion;
    U2 minorVersion;
    U4 module;
    U4 language;
    U4 entryPoint;
    U4 VMCodesCount;  // Code
    VMCodeInfo VMCode[VMCodesCount];
    U4 metadataCount;  // Metadata
    MetadataInfo Metadata[metadataCount];
}
```

(그림 2) EFF 파일의 구조

EFF는 실행에 필요한 모든 정보를 가진 EVM의 실행 파일 포맷으로 이진 형태의 스트림으로 구성된다. EFF 파일의 아이템은 헤더, 코드, 메타데이터의 3부분으로 구성된다. 헤더 부분은 EFF 파일 버전, 프로그램 이름, 시작점 등의 정보를 갖는다. 코드 부분은 실질적인 명령어들을 포함하고 있으며 메타데이터 부분은 실행에 필요한 모든 정보를 저장한다.

2.2 아카이브 파일

모듈화를 통하여 분리되어 생성된 여러 개의 목적 프로그램을 하나의 파일로 결합하면 목적 프로그램 관리가 용이해지는 장점이 있다. 다수의 목적 프로그램을 하나의 프로그램으로 결합하는 작업을 아카이브(archive)라고 한다. 그리고 아카이브 과정을 통해 생성되는 파일을 아카이브 파일(archive file)이라고 한다. 이런 아카이브 파일 포맷에는 유닉스 ar 프로그램에서 사용하는 아카이브 포맷인 A 파일 포맷[5]과 자바에서 사용하는 JAR 파일 포맷이 있다[6].

2.2.1 유닉스의 아카이브 파일

ar은 라이브러리를 관리하는 GNU 유틸리티로서 목적 프로그램인 *.o 파일들이 결합하여 *.a 파일을 생성한다. *.a와 같은 파일을 유닉스에서 사용하는 아카이브 파일이라고 한다.

생성된 *.a 파일의 구조는 (그림 3)과 같다. *.a 파일의 가장 처음 전역 헤더가 위치하고, 결합된 각각의 파일 앞에 파일 헤더가 위치하게 된다[5].

<table>
<tr><td colspan="2">[Global Header]</td></tr>
<tr><td>[File Header]</td><td>Group ID</td></tr>
<tr><td>File name</td><td>File mode</td></tr>
<tr><td>File modification date</td><td>File size in bytes</td></tr>
<tr><td>Owner ID</td><td>File magic</td></tr>
<tr><td colspan="2">a.o</td></tr>
<tr><td>[File Header]</td><td>Group ID</td></tr>
<tr><td>File name</td><td>File mode</td></tr>
<tr><td>File modification date</td><td>File size in bytes</td></tr>
<tr><td>Owner ID</td><td>File magic</td></tr>
<tr><td colspan="2">b.o</td></tr>
<tr><td colspan="2">...</td></tr>
</table>

(그림 3) *.a 파일의 구조

전역 헤더는 아카이브 파일 당 하나씩 존재하고, 파일 헤더는 아카이브 파일에 결합되어 있는 파일의

개수만큼 존재한다. 전역 헤더는 매직 스트링이라 불리는 유일한 식별자로, "!<arch>\n" 값을 갖는다. 파일 헤더는 <표 1>과 같이 구성된다.

<표 1> File Header의 구성

File Offset	Field Name	Field Format
0-15	File Name	Decimal
16-27	File Modification date	Decimal
28-33	Owner Id	Decimal
34-39	Group Id	Decimal
40-47	File mode	Octal
48-57	File size in bytes	Decimal
58-59	File magic	String

2.2.2 자바의 JAR 파일

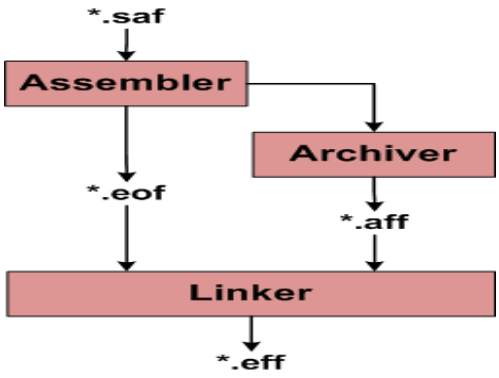
자바에서 사용하는 아카이브 파일인 JAR(Java ARchive)는 zip 압축 파일 포맷을 기반으로 다수의 파일을 하나의 파일로 결합한 파일 포맷이다. JAR 파일은 클래스 파일 뿐만 아니라 이미지, 음성 파일의 결합도 지원한다[6].

JAR 파일은 임의의 META-INF 디렉토리를 포함하여 확장이나 패키지를 구성하는 데이터를 저장한다. 그리고 META-INF 디렉토리는 보안, 버전, 확장, 서비스의 정보를 포함한다. META-INF 디렉토리 안에 존재하는 MANIFEST.MF는 기본적으로 Manifest-Version, Created-By, Signature-Version, Class-Path의 항목으로 구성된다. 그러나 애플릿, 확장 등에 따라 속성들이 추가될 수 있다.

3. 모듈화를 위한 아카이브 파일 포맷

3.1 모듈화를 위한 EVM 시스템

(그림 5)는 EVM에서 동작하는 프로그램의 모듈화를 위한 시스템 구성도이다. 구성도에 따라 각 도구별로 기능을 살펴보면 다음과 같다. 어셈블러는 어셈블리 파일인 *.saf를 입력으로 받아 목적 프로그램 파일인 *.eof를 출력한다. 아카이버는 다수의 *.eof나 아카이브 파일인 *.aff를 입력으로 받아 하나의 *.aff 파일로 출력한다. 링커에서는 *.eof나 *.aff를 입력으로 받아 EVM의 실행 파일인 *.eff를 출력한다. 본 논문에서는 (그림 5)의 시스템을 구현하기 위해 아카이버에서 출력될 파일 포맷인 *.aff를 설계한다.



(그림 5) 모듈화를 위한 EVM 시스템

3.2 설계 목표

본 논문에서 설계하는 AFF 파일은 EVM에서 실행되는 프로그램의 모듈화를 위한 아카이브 파일 포맷이다. AFF는 파일 관리를 위해 둘 이상의 EFF 파일들을 하나의 아카이브 파일로 결합하고, 아카이브 파일에 특정 EFF 파일을 추가 또는 제거하는 기능을 수행할 수 있도록 유연한 구조를 지녀야 한다. 그리고 기존의 EFF와 AFF가 호환 가능하도록 한다.

3.3 AFF의 구조

본 논문에서 설계하는 AFF 파일의 전체적인 구조는 (그림 2)의 EFF와 동일한 구조로 하여 기존 EFF와 호환이 가능하도록 한다. 그리고 메타데이터 테이블에 *MDTArchiveDT*를 추가하여 아카이브에 관련된 정보를 저장할 수 있도록 한다.

AFF 파일 구조와 데이터들은 <표 2>와 같이 설계한다.

<표 2> AFF 파일의 구조와 데이터

AFF의 항목들	기본값
U4 magic	0x054C4CAB
U2 majorVersion	0x0001
U2 minorVersion	0x0000
U4 module	0x00000000
U4 language	0x00000000
U4 entryPoint	0x00000000
U4 VMCodesCount	0x00000000
VMCodeInfo VMCode[VMCodesCount]	없음
U4 metadataCount	0x00000001
MetadataInfo Metadata[MetadataCount]	MDTArchiveDT 테이블에 아카이브 파일 정보 저장

3.3.1 *MDTArchiveDT* (0xa000)

아카이브되는 파일들의 정보를 저장하기 위해 추가된 *MDTArchiveDT* 메타데이터의 구조는 (그림 6)과 같다.

```
MDTArchiveDT {
      U4 FileCount;
      FileInfo ArchiveList[FilecCount];
      U1 ArchiveData[Total_Filesize];
}
```

(그림 6) MDTArchiveDT의 구조

*MDTArchiveDT*는 AFF에 결합되어지는 파일들의 개수인 *FileCount*, 결합된 각 파일들의 속성에 관련된 정보를 저장하는 *FileInfo*, 결합된 파일들의 실제 데이터를 순차적으로 저장하는 *ArchiveData*로 구성된다. *ArchiveData*의 *Total_Filesize* 값은 *FileInfo*의 항목인 *FileLength*의 값을 통하여 결합되는 전체 파일의 크기를 계산하여 설정한다.

3.3.2 *FileInfo*

MDTArchiveDT 메타데이터의 항목 중 하나인 *FileInfo* 구조체는 (그림 7)과 같은 구조를 갖는다.

```
FileInfo {
      U2 FileNameLength;
      U1 FileName[FileNameLength];
      U4 FileLength;
      U4 FileOffset;
      U2 FileOptionLength;
      U1 FileOption[FileOptionLength];
}
```

(그림 7) FileInfo의 구조

FileInfo 구조체는 결합되는 각 파일들의 속성에 관련된 정보들을 저장한다. *FileInfo*의 결합되는 파일 이름의 길이를 저장하는 *FileNameLength*, 파일명을 저장하는 *FileName*, 파일의 길이를 저장하는 *FileLength*, 해당 아카이브 파일 내에서 결합되는 파일의 위치를 나타내는 *FileOffset*, 파일 옵션의 길이를 저장하는 *FileOptionLength*, 파일의 암호화 여부 등 옵션을 저장하는 *FileOption*으로 구성된다.

4. 결론 및 향후 연구

본 논문에서는 EVM의 실행 파일 포맷과 기존의 유사한 개념의 파일 포맷을 분석하여, EVM에서 사용할 아카이브 파일 포맷인 AFF를 정의하였다. AFF는 다수의 파일을 결합하여 보관할 수 있고, 결합된 파일에 특정 파일을 추가 및 삭제시킬 수 있도록 유연한 구조를 가지고 있다. 그리고 기존 EFF와 호환 가능하도록 하기 위해 EFF의 구조를 반영하여 AFF를 설계하였다. 아카이브 파일 포맷의 설계로 추후 아카이버를 구현할 수 있는 기반을 만들었다.

향후에는 본 논문에서 설계한 아카이브 파일 포맷을 기반으로 프로그램 모듈화를 지원하는 도구인 아카이버와 링커를 구현할 것이다. 아카이버는 다수의 파일이 결합된 아카이브 파일을 생성하고 관리하는 기능을 지원하는 도구이다. 링커는 목적 파일과 아카이브 파일에 존재하는 미정의 심볼을 해결하는 기능을 수행하는 도구이다. 이러한 도구들의 구현으로 EVM 프로그램의 모듈화를 완성할 것이다.

참고문헌

[1] 남동근, “가상기계를 위한 어셈블리 언어의 설계”, 동국대학교 석사학위논문, 2003.

[2] 고광만, 이양선, 이양선, “임베디드 시스템을 위한 가상기계의 설계 및 구현”, 한국멀티미디어학회 논문지, 제 8권, 제 9호, pp.1283~1291, 2005.

[3] 정한종, “임베디드 가상기계를 위한 실행 파일 포맷”, 동국대학교 석사학위논문, 2004.

[4] 손윤식, 오세만, “실행 파일 포맷 생성기의 설계 및 구현”, 한국정보처리학회 추계학술발표대회 논문집, 제 11권, 제 2호, pp.623~626, 2004.

[5] The Open Group Base Specification Issue 6, http://www.opengroup.org/onlinepubs/000095399/utilities/ar.html

[6] JAR File Specification, Sun Microsystems

[7] EFF Specification, 동국대학교 프로그래밍 언어 연구실, 2006.

[8] SAF Specification, 동국대학교 프로그래밍 언어 연구실, 2006.

[9] SIL Specification, 동국대학교 프로그래밍 언어 연구실, 2006.