Stabs에서 DIL로의 번역기의 구현

김성엽* 장종현** 김성재* 김승호*
*경북대학교 컴퓨터공학과 **한국전자통신연구원
sykim@borami.knu.ac.kr jangjh@etri.re.kr
sjkim@borami.knu.ac.kr shkim@bh.knu.ac.kr

An Implementation of Stabs to DIL Translator

Seong-Yup Kim* Jong-Hyun Jang** Seong-Jae Kim* Sung-Ho Kim*
*Department of Computer Engineering, Kyungpook National University
**Electronic and Telecommunication Research Institute

요 약

교환기 소프트웨어와 같은 통신 시스템은 분산 병렬적인 수행 특성을 가지며, CHILL의 디버깅 환경인 TECH는 이러한 특성을 반영하고 있다. 최근의 교환기 소프트웨어 개발에 C 언어가 많이 사용되고 있는데, C 디버깅 환경은 이러한 특성을 잘 반영하지 못한다. 따라서, C 언어로 작성된 프로그램을 TECH에서 디버깅할 수 있다면 새로운 환경을 구축하지 않고도 기존의 환경을 이용함으로써 비용과 시간을 줄일 수 있다. 그러나, 두 환경의 디버깅 정보인 Stabs와 DIL은 표현 방식이 다르므로 이들간의 번역을 위한 도구가 필요하다. 본 논문에서는 이를 위해 Stabs 정보와 실행 파일을 분석하는 분석기, 분석된 정보를 각각에 해당하는 DIL에 대응시키는 대응기, 대응된 DIL 정보를 생성하는 생성기로 구성된 Stabs에서 DIL로의 번역기를 구현하였다.

1. 서론

대형 교환기 소프트웨어들은 다른 분야의 소프트웨어에 비해 많은 개발 인력을 필요로 하고, 일반적으로 긴 생명주기를 가진다. 이로 인해 시스템에 관련된 비용 측면에서 볼 때, 초기 시스템 개발보다는 소프트웨어의 유지보수 및 관리에 더 많은 비용이 투자된다. 이에, CCITT(The Consultative Committee for International Telegraph and Telephone) 및 이의 후신인 ITU-T(International Telecommunication Union - Telecommunication Standardization Sector)에서는 보다 효율적으로 시스템을 유지보수하기 위해, 시스템에 대한 목적 코드를 개발하기 전에 목적 시스템을 명세(Specification)하고 기술(Description)할 수 있는 언어로 SDL(Specification and Description Language)을, 목적 시스템을 구현할 수 있는 언어로 CHILL(CCITT High Level Language)을 권고하였다[1-3].

또한, 교환기 소프트웨어 등의 통신용 소프트웨어들은 일반적인 소프트웨어와는 달리 분산 병렬적인 수행 특성을 가진다. 따라서, 이러한 소프트웨어들을 개발하는데 사용할 디버깅 환경은 실시간 수행 시 나타나는 특성들에 대한 고려가 필요하다. CHILL 언어로 생성된 목적 코드의 디버깅 환경인 TECH(Testing Environment for CHILL)는 이러한 특성들을 반영할 수 있는 동시적인 디버깅이 가능하여 널리 사용되고 있다[4].

최근의 소프트웨어 개발 시, 다양한 응용분야에 적용할 수 있는 일반적인 기능에 중점을 두고 설계된 C 언어가 많이 사용되고 있다. 범용 프로그래밍 언어인 C는 구조화된 프로그래밍 및 모듈화된 설계를 쉽게 할 수 있어서 프로그램의 신뢰도가 높고 이해하기 쉽다. 또한, 유연성과 이식성도 매우 좋아, 특정 시스템을 위해 만들어진 C 프로그램을 다른 시스템에서 실행하려 할 때도 그 프로그램의 일부만 수정하거나 또는 전혀 수정하지 않아도 되는 등의 여러 가지 장점을 가진다. 그래서, C 언어는 교환기용 소프트웨어 개발에도 많이 사용되고 있다.

C 언어로 작성된 교환기용 소프트웨어를 디버깅하기 위한 환경은 분산 병렬적인 수행 특성을 반영할 수 있어야 하는데, C 디버깅 환경은 이러한 특성을 가지지 않는다. 따라서, 기존의 CHILL 디버깅 환경인 TECH를 활용할 수 있다면, 효율이나 비용의 측면에서 새로운 디버깅 환경을 구축하는 것보다 나은 효과를 볼 수 있을 것이다.

그런데, C 언어로 작성된 프로그램의 디버깅을 위한 정보인 Stabs는 TECH에서 필요로 하는 디버깅 정보인 DIL(Debug Information Language)과 일치하지 않기 때문에, 실제로 C 언어로 작성된 프로그램을 TECH에서 그대로 디버깅하는 것은 불가능하다[5, 6]. 따라서, C 언어로 작성된 프로그램을 TECH에서 실시간 병렬 디버깅하기 위해서는 C 프로그램의 디버깅 정보인 Stabs 정보와 실행 파일을 분석하여 TECH에서 사용하

는 디버깅 정보인 DIL 형태로 바꾸어주는 번역기의 개발이 필요하다. 이를 위해, Stabs 정보와 실행 파일을 분석하는 분석기, 분석된 정보를 각각에 해당하는 DIL에 대응시키는 대응기, 대응된 DIL 정보를 생성하는 생성기로 구성된 Stabs에서 DIL로의 번역기를 구현한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 Stabs와 DIL에서 대해 설명하고, 3장에서는 Stabs에서 DIL로의 번역기와 실제 번역 예를 보이며, 4장에서는 결론을 내린다.

2. Stabs 및 DIL

2.1 Stabs

Stabs는 C 언이로 작성된 프로그램으로부터 생성된 목적코드를 디버깅하기 위한 정보를 말한다. Stabs는 어셈블러에서 사용하는 슈도-디렉티브(pseudo-directive) 형태이며 컴파일러가 디버거에서 원시 프로그램의 특성을 기술하는 것이다. Stabs로 기술할 수 있는 원시 프로그램의 특성에는 소스의 라인 번호, 변수의 모드와 범위, 함수의 이름, 함수의 매개변수, 함수의 범위 등이 있으며 각각을 표현하는 형식은 아래와 같다.

- 심벌 정보 표현 : .stabs "string", type, other, desc, value
- 라인 정보 표현 : .stabn type, other, desc, value

이 때, type 필드가 기본적인 정보를 가지며 이에 따라 other, desc, value 필드의 의미가 변한다. 또한 string 필드는 name:symbol-descriptor type-information의 형태로 심벌에 대한 정보를 나타낸다. 표 1은 Stabs 정보의 예이다.

표 1. Stabs 정보의 예

```
.stabs "main:F1", 36, 0, 0, _main
.stabn 192, 0, 0, LBB2
.stabn 224, 0, 0, LBE2
```

표 1에서 '.stabs'의 'main'은 심벌 이름, 'F'는 전역 함수, '1'은 반환형이 정수형, '36'은 함수, '_main'은 어셈블러에서의 레이블로 주소 정보를 나타낸다. '.stabn'의 '192'는 C 함수의 '{', '224'는 C 함수의 '}', 'LBB2'와 'LBE2'는 어셈블러에서의 레이블로 각각의 주소 정보를 나타낸다.

2.2 DIL

DIL은 TECH에서 CHILL 언어로 작성된 프로그램으로부터 생성된 목적코드를 디버깅하기 위한 정보를 말한다. DIL은 LISP 언이에서 사용하는 리스트의 형식으로 ASCII 파일 형태이며 하나의 CHILL 소스에 해당하는 하나의 DIL 파일이 생성되는데 이러한 DIL 파일을 DIL 엔트리라고 한다. 표 2는 DIL의 예이다.

표 2. DIL 정보의 예

```
( DDEFIN "a" DIS ( A ? ) )
```

표 2에서 'DDEFIN'은 변수의 선언, 'a'는 변수명, 'DIS'는 변수의 데이터형, 'A ?'는 주소 정보로 '?' 위치에 실제 주소 정보가 계산되어 들어간다.

3. Stabs에서 DIL로의 번역기

Stabs에서 DIL로의 번역기는 C 언어로 작성된 프로그램의 디버깅 정보를 TECH에서 사용할 수 있도록 CHILL 언어로 작성된 프로그램의 디버깅 정보 형태로 번역하여 TECH에서 C 프로그램을 디버깅할 수 있도록 해준다. 그림 1은 C 프로그램을 TECH에서 디버깅하기 위한 전체적인 개요이다.

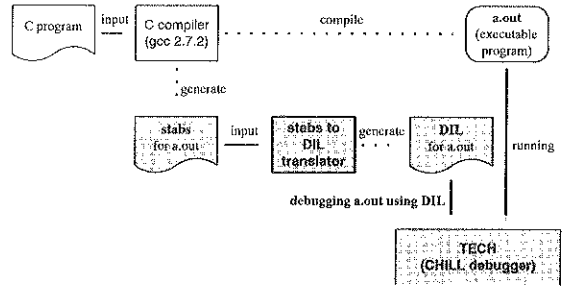

그림 1. C 프로그램을 TECH에서 디버깅하기 위한 전체적인 개요

3.1 설계 및 구현

Stabs에서 DIL로의 번역기는 크게 Stabs 정보와 실행 프로그램을 분석하는 분석기, 분석된 정보를 각각에 해당하는 DIL에 대응시키는 대응기, 대응된 DIL 정보를 생성하는 생성기로 나뉜다. 그림 2는 Stabs에서 DIL로의 번역기 구성을 나타낸다.

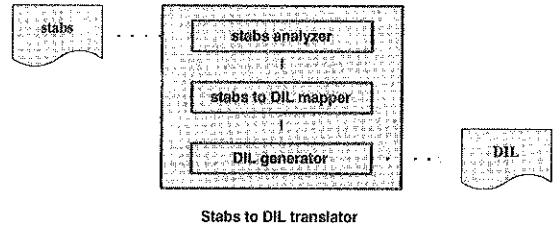

그림 2. Stabs에서 DIL로의 번역기 구성

3.1.1 Stabs 분석기

이 모듈은 입력으로 들어오는 C 언어로 작성된 프로그램의 디버깅 정보인 Stabs와 실행 파일을 분석하는 역할을 한다. Stabs의 심벌 정보와 라인 정보에 해당하는 .stabs 및 .stabn을

분석하여 DIL로의 번역을 위한 정보를 생성하고, 실행 파일에서 추출한 부가적인 주소 정보도 대응기로 전달한다. 또한, 파일의 이름이나 경로, 생성 시간, 컴파일러 관련 정보 등도 전달하는 정보에 포함된다.

3.1.2 Stabs to DIL 대응기

Stabs 분석기에서 분석된 정보를 각각에 해당하는 DIL 정보로 대응시키는 모듈이다. 분석기에서 넘어오는 정보를 Stabs 문법과 동일한 의미를 가지는 DIL 문법으로 대응시키고, 추가적인 주소 정보를 통합하여 생성기에 전달하며, 그 외의 정보들도 전달한다.

3.1.3 DIL 생성기

대응된 각각의 심벌 정보와 주소 정보를 포함하는 DIL의 실제적인 코드를 생성한다. 심벌 정보와 주소 정보 이외의 정보를 바탕으로 DIL 문법에서 필요로 하는 헤더 정보들도 추가한다. 이 모듈에서 생성된 DIL 코드를 이용하여 C 언어로 작성된 프로그램을 TECH에서 디버깅할 수 있다.

3.2 예제

표 3에는 C 언어로 작성된 간단한 프로그램과 해당 프로그램의 Stabs 정보 및 이를 번역한 DIL 정보가 나타나있다.

표 3. C 프로그램 및 해당 Stabs와 DIL의 예

C	1 : void x() 2 : { 3 : int a; 4 : }
Stabs	1 : .stabs "x:F19", 36, 0, 2, x 2 : .stabs "a:1", 128, 0, 3, -20 3 : .stabn 192, 0, 0, .LLBB2-x 4 : .stabn 224, 0, 0, .LLBE2-x
DIL	1 : (DFU "x" "" 2 : (DFUN DVOID ( ) ( ) ((r 14 64) 4) ( )) 3 : (1 ((A 562800) (A 562812))) 4 : (1 ((A 562804) (A 562812))) 5 : (A 562800) 6 : (1 DDEFIN "a" DIS (r 30 -20))) 7 :)

표 3에서 C 코드의 1번 행은 C 컴파일러를 통해 Stabs 정보의 1번 행과 같은 디버깅 정보를 가진다. 해당 행의 'F'는 전역함수를 의미하며, '19'는 함수의 반환형으로 'void'를 의미하고, 행의 마지막에 나오는 레이블은 후에 실제 주소 정보로 바뀐다. 이렇게 표현된 Stabs 정보는 Stabs 분석기, Stabs to DIL 대응기와 DIL 생성기를 거쳐 DIL 정보의 1, 2번 행으로 번역된다. C 코드의 2, 4번 행은 함수의 영역을 나타내는 것으로 Stabs 정보의 3, 4번 행과 대응이 되며, DIL 정보의 3, 4, 5, 7번 행에서 실제 주소 정보와 함께 표현된다. C 코드의 3번 행은 Stabs 정보의 2번 행을 디버깅 정보로 가지는데, "a:1"의 '1'은 변수의 데이터형으로 정수형을 의미한다. 이 Stabs 정보는 DIL 정보의 6번 행으로 대응되며, 'DIS'가 마찬가지로 변수의 데이터형을 나타낸다.

표 3에 나타난 Stabs 정보 및 DIL 정보는 해당 C 코드에 대한 완전한 정보는 아니며, 파일명이나 컴파일러 관련 정보 등의 헤더 정보들이 추가되어야 한다.

4. 결론

본 논문에서는 대규모의 개발 인력과 긴 생명주기를 가지는 대형 교환기용 소프트웨어의 효율적인 개발을 위해 TECH에서 C 프로그램을 디버깅할 수 있도록 C 프로그램의 디버깅 정보인 Stabs를 TECH에서 사용하는 디버깅 정보인 DIL로 변환하는 번역기를 구현하였다.

Stabs에서 DIL로의 번역기는 C 프로그램의 디버깅 정보인 Stabs와 실행 파일을 분석하는 Stabs 분석기, 분석된 정보를 TECH 환경을 위한 디버깅 정보인 DIL에 대응시키는 Stabs에서 DIL로의 대응기, 대응된 DIL 코드를 생성하기 위한 DIL 생성기 모듈로 구성되어 있다.

구현된 번역기를 통해 TECH에서 C 프로그램을 디버깅할 수 있게 되어 프로그램의 수정이 용이해짐에 따라 효율적인 유지보수 및 관리가 가능하게 되었다. 또한, 기존의 SDL-CHILL 개발환경을 활용하여 개발시간의 단축 및 비용의 절감 효과를 얻을 수 있었다.

참고 문헌

[1] A. Olsen, O. færgemend, B. Pedersen, R. Reed, and J. Smith, *Systems Engineering Using SDL-92*, ELSEVIER SCIENCE B.V., 1995.

[2] 최완, 박항구, 홍진표, 강석열, 송영기, 최고봉, 김환철, 신영승, 조준희, 이근구, 정찬진, *SDL 환경 : TDX-10 총서 제 6 권*, 한국전자통신연구소, 1994.

[3] *CCITT Specification and Description Language*, ITU Recommendation Z. 100, p.220, 1992.

[4] D. Lee, J. Lee, W. Choi, B. Lee, and C. Han, *"A New Integrated Software Development Environment Based on SDL, MSC, and CHILL for Large-scale Switching Systems,"* ETRI journal, Vol. 18, No. 4, 1997.

[5] *http://sources.redhat.com/cygwin/stabs.html*

[6] S. Deraas, P. Moe, and T. Børsting, *DIL - Debug Information Language*, Kvatro Telecom AS, 1998.