내장 프로세서 응용 개발을 위한 미들웨어 구현

박 현수* , 한 경숙* , 우 덕균* , 표 창우** , 김 흥남***
*홍익대학교 대학원 전자계산학과
**홍익대학교 컴퓨터공학과
***한국전자통신연구원

Implementing a middleware for development of embedded processor applications

Hyunsoo Park* , Kyungsook Han* , Dukkyun Wu* , Changwoo Pyo** , Heungnam Kim***
*Dept. of Computer Science, Graduate School, Hongik University
**Dept. of Computer Engineering, Hongik University
***Electronics and Telecommunications Research Institute

요 약

내장 프로세서에서 실행되는 응용 프로그램 개발환경에서는 타겟에서 실행될 프로그램을 호스트에서 개발하여 타겟에서 실행되도록 구성된다. 내장 프로세서 응용 개발환경의 중심을 이루는 호스트-타겟 연결 미들웨어(이후 타겟서버라 칭함)를 개발하였다. 타겟서버는 전단부, 후단부 인터페이스를 통하여 호스트에서 개발된 목적모듈을 타겟시스템에 로딩시키는 기능을 한다. 또한, 호스트에서 운용되는 타겟 접근도구의 요구사항을 전송하고 타겟의 정보를 호스트 도구에게 전송하는 역할을 한다. 이러한 역할을 수행하기 위하여 내부모듈에서 리더, 로더, 타겟메모리 관리, 심볼테이블 관리모듈이 구현되었다. 이렇게 구현된 타겟서버는 기존의 호스트 시스템 도구에 연결되어 정확한 수행에 대한 테스팅 과정을 거쳤고 하나의 라이브러리로 지원되었다.

1. 서론

내장 프로세서를 포함한 휴대폰, TV등의 가전제품에서 실행되는 응용 프로그램은 각각의 프로세서에 따라 다르게 개발된다. 이에 따라 내장형시스템 응용 프로그램 개발환경은 타겟시스템에서 실행되는 프로그램을 호스트에서 개발하여 타겟에서 수행되도록 구성된다. 최근 프로세서마다 다르게 개발될 수 있는 프로그램 개발환경의 필요성은 점점 더 높아지고 있다. 이러한 개발환경에서는 타겟시스템에 무관한 동일한 인터페이스를 통하여 호스트에서의 요구사항을 수행할 수 있고, 타겟시스템과의 정보전송을 할 수 있게 하는 미들웨어의 역할이 중요시된다.

타겟서버는 호스트시스템에서 내장 프로세서마다 다르게 컴파일된 코드를 타겟시스템에 보내어 실행시키고, 호스트에서 타겟시스템을 관리할 수 있는 개발환경을 제공한다. 호스트시스템에서 수행되는 타겟서버는 전단부 인터페이스를 통하여 호스트 도구들과 통신하고, 후단부 인터페이스를 통하여 타겟시스템과의 전송을 하게 된다.

우리가 개발한 타겟서버는 전단부, 후단부 인터페이스를 통하여 전송될 요구사항을 처리하는 타겟서버 내부모듈이다. 내부모듈로서 목적모듈을 읽어들이는 리더와 타겟에 목적모듈을 보내는 로더를 개발하였고, 리더와 로더에서 호출되어 사용되는 타겟메모리 관리모듈과 심볼테이블 관리모듈을 개발하였다.

본 논문의 구성은 다음과 같다. 2절에서는 내장형 시스템 개발환경에 대하여 기술하고 3절에서는 타겟서버의 역할과 개발된 내부모듈에 대해 설명한다. 그리고 4절에서는 개발된 내부모듈에 대한 테스팅 환경을 설명하고 5절에서 결론과 향후 발전방향을 제시하였다.

2. 관련 연구 분야

기존의 내장형 시스템 응용 프로그램 개발환경을 살펴보면 다음과 같다.

MULTI 개발환경은 하나의 실행파일 생성을 위하여 Ada, C, Fortran등과 같은 여러 가지 언어가 함께 결합되어 프로그램을 개발할 수 있다. 타겟과의 연결은 호스트에 상주하는 디버거서버에 의하여 이루어진다[2].

Microware사의 FasTrak은 Ultra C/C++ 컴파일러를 통하여 타겟시스템에 적합한 목적모듈을 생성하고, 타겟시스템과는 타겟시스템 툴에 의하여 연결되는 개발환경이다[3].

Mentor Graphics사의 Spectra는 XSH쉘을 통하여 호스트와 타겟사이의 크로스 개발 인터페이스를 제공하고 타겟매니저를 통하여 타겟과 연결된다[4].

본 논문은 1999년 한국전자통신연구원의 "사용자 개발 도구 연구"의 지원을 받은 논문임.

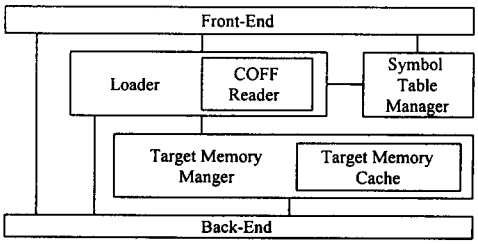
Integrated Systems사의 pRISM+는 pRISM+ 매니저에 의하여 프로그램을 컴파일하고 CORBA 미들웨어를 통하여 타겟과 연결된다[5].

마지막으로 우리가 실험대상으로 삼은 WindRiver사의 Tornado는 GNU C 컴파일러를 사용하고 타겟서버를 통하여 타겟과 연결된다[1]. Tornado는 새로운 목적파일을 Tornado시스템과 연결할 수 있는 장점을 가지고 있어서 새로운 컴파일러도 쉽게 수용할 수 있는 장점을 가지고 있다.

이러한 응용 프로그램 개발환경은 호스트에서 각각의 타겟 시스템에 따라 프로그램을 개발하고, 개발된 프로그램을 타겟에서 수행시키기 위하여 호스트와 타겟을 연결하는 미들웨어가 구축되어 있다.

3. 타겟서버의 역할

타겟서버의 구조는 [그림 1]과 같다.



[그림 1] 타겟서버 구성도

타겟서버는 호스트시스템에서 개발된 프로그램을 타겟으로 전송하는 기능과 호스트 도구에서의 요구사항을 처리하는 기능을 한다. 호스트 도구의 요구사항은 전단부 인터페이스를 통하여 전송되며 후단부 인터페이스를 통하여 타겟 시스템으로 전송되거나, 타겟서버에서 처리된다. 타겟으로 요구사항이 전송된 경우 타겟에서 전달된 결과값을 호스트 도구에게 전송한다.

내부모듈에는 OMF(목적 모듈 형식) 리더, 로더, 심볼테이블 관리모듈, 타겟 메모리 관리모듈이 있다. 이러한 내부모듈은 타겟시스템과의 전송없이 호스트 도구에서의 요구사항을 처리할 수 있다. 또한 목적모듈을 타겟시스템에 로드하여 해당 모듈을 실행하며, 타겟에 로드되어 있는 모듈을 타겟에서 언로드시키는 역할을 한다.

목적모듈 로드/언로드의 수행과정을 살펴보면, 리더에서 목적포맷에 맞는 목적모듈을 분석하고, 타겟메모리 캐시에서 심볼에 대한 재배치를 수행한다. 로더에서는 타겟메모리 관리 모듈을 이용하여 타겟에 대한 메모리 정보를 얻는다. 그리고 타겟에 모듈을 로드시키고 모듈에 대한 메모리 정보를 유지한다. 언로드시에는 심볼테이블 관리모듈에서 언로드된 모듈에 대한 심볼들을 삭제하고 타겟메모리 관리모듈에서는 타겟메모리에 대한 메모리정보 수정과 모듈에 대한 정보를 삭제하게 된다.

그러면 타겟서버의 내부모듈에 대하여 자세히 살펴보겠다.

3.1 Coff 리더 모듈

리더는 타겟에 로딩할 목적모듈의 포맷에 따라 목적모듈을 분석하고 재배치하는 역할을 한다. 여러 목적모듈 포맷에 맞는 리더모듈이 생성되어 타겟서버와 연결될수 있다. 본 논문에서 제시된 리더는 타겟시스템에 내장된 strongARM SA-110 프로세서[8]와 coff 모듈포맷[6,7]에 적합하게 구현되었다. 호스트에서 타겟의 메모리와 타겟에 로딩될 목적모듈을 관리하는 개발환경에서는 타겟프로세서에 대한 재배치 작업을 모듈단위로 행하게 된다.

리더에 의해 목적모듈이 coff 모듈인지 검사하고 심볼들을 심볼테이블에 등록한다. 목적모듈의 각 섹션을 저장하기 위해 타겟메모리 영역을 확보한다. coff 목적모듈에 대한 섹션 정보를 얻어서 섹션들의 집합인 텍스트, 데이터, bss 세그먼트의 크기를 결정하고 목적모듈의 섹션들을 세그먼트 단위로 묶어 타겟메모리에 로딩한다. 이때 모듈의 타입에 따라 호스트 캐시 혹은 타겟메모리로 로딩된다. 재배치 모듈인 경우 세그먼트들을 캐시에 로딩하고 캐시에서 심볼에 대한 재배치를 수행하게 된다. 재배치는 심볼에 대하여 실제 타겟메모리에서의 주소값을 구하여 호스트캐시의 텍스트, 데이터 세그먼트에서 심볼에 대한 새로운 주소값으로 할당한다.

3.2 로더 모듈

리더에 의해 읽혀진 목적모듈을 타겟시스템에 로드/언로드시키고 타겟에 로딩된 모듈을 관리하는 역할을 한다.

목적모듈을 타겟에 로드할때는 로딩될 모듈의 세그먼트별로 타겟메모리 관리모듈을 호출하여 타겟메모리를 할당한다. 심볼테이블 관리모듈을 호출하여 전역변수나 전역함수에 대한 심볼들을 심볼테이블에 등록한다. 리더를 호출하여 모듈의 분석과 재배치를 수행한 뒤 텍스트, 데이터, bss 세그먼트별로 타겟메모리에 저장한다. 타겟에 로딩된 모듈은 모듈정보 리스트에 추가되어 호스트 도구에서의 모듈에 대한 정보를 요구할 때 또는 재배치 시 활용된다. 모듈정보에 대한 요구사항에 부응하기 위해 모듈을 탐색하고, 모듈정보를 출력하거나 모듈의 세그먼트정보를 출력하는 루틴이 사용된다.

타겟에서 모듈을 언로드할 때는 타겟메모리 관리모듈에서 모듈이 차지하는 타겟메모리 공간을 해제하고, 모듈에서 정의되었던 심볼들을 심볼테이블에서 삭제한다. 그리고, 모듈에 관한 정보를 모듈정보 리스트에서 삭제한다.

3.3 심볼테이블 관리 모듈

타겟서버는 타겟시스템에 로드된 모든 함수와 변수들을 위한 심볼테이블을 유지하고 있다. 이 심볼테이블은 목적모듈 로딩시에 목적모듈에서 정의된 함수와 심볼을 추가하고, 심볼에 대한 정보가 필요할 때 활용된다. 목적모듈을 언로드시에는 모듈에서 사용된 함수와 심볼이름을 삭제하게 된다.

심볼들은 크게 정의된 심볼과 정의되지 않은 심볼로 나뉜다. 정의된 심볼은 텍스트, 데이터, bss 세그먼트에서 참조되고 목적모듈에 정의되어 있다. 반면, 정의되지 않은 심볼은 텍스트, 데이터 세그먼트에서 참조되지만, 목적모듈에는 정의되지 않은 심볼로서 심볼테이블에는 저장되지 않는다.

심볼테이블은 해쉬테이블로 구성되며 각 심볼에 대해 심볼이름, 심볼 주소값, 심볼타입, 심볼이 정의된 모듈을 구분하기 위한 그룹 id의 정보를 가지고 있다. 이러한 심볼테이블에 대한 정보를 이용하여 다양한 방법으로 심볼테이블에 있

는 심볼을 검색할 수 있고, 각각의 심볼에 대하여 필요한 함수를 실행시킬 수 있다.

3.4 타겟메모리 관리 모듈

타겟메모리 관리모듈은 타겟메모리 영역 중 사용되고 있는 메모리 영역과 목적모듈이 로드될 수 있는 메모리 영역에 대한 정보를 유지하여 목적모듈 로드 시 사용한다. 또한 타겟메모리 캐시를 사용하여 재배치를 수행하고 호스트-타겟간 정보전송을 감소시킬 수 있다.

타겟메모리 관리모듈은 타겟 구동시 호스트에서 관리할 수 있는 타겟메모리 풀 영역에 대한 정보를 얻는다. 타겟메모리 풀이란 타겟메모리 영역중 호스트 시스템이 관리하는 영역이다. 타겟메모리 관리모듈에서 이 영역에 대한 정보를 유지하여 모듈 로드시 사용한다. 타겟메모리 관리모듈은 로더가 목적모듈을 로드할 때 타겟메모리 정보를 사용하여 메모리를 할당한다. 만약 타겟메모리 공간이 부족하게 되면 타겟시스템에 메모리를 요구하여 타겟메모리 영역을 늘릴 수 있다. 목적모듈이 언로드될 때 해당된 타겟메모리도 해제시키고, 타겟메모리에 대한 정보도 수정한다.

타겟메모리 관리모듈에서 관리되는 캐시는 리더에서의 재배치를 수행하고, 타겟과의 전송으로 인한 손실을 줄이기 위해 구현되었다. 타겟메모리 관리모듈에서 유지되는 타겟메모리에 대한 정보를 이용하여 호스트 시스템에서 메모리를 할당받아 캐시로 사용하게 된다. 목적모듈 로드시에 캐시에서 모듈의 재배치를 수행하고 타겟에 로딩한다.

4. 타겟서버 테스팅 환경

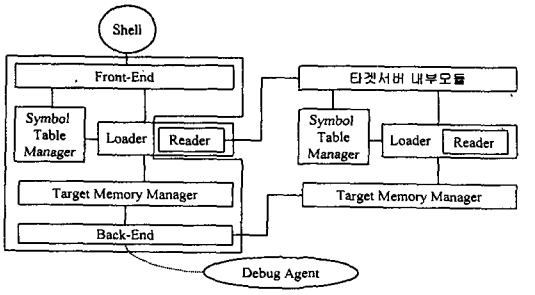
본 논문에서 구현된 타겟서버를 테스트하기 위하여 Tornado 개발도구를 사용하였다[1].

Tornado 호스트 시스템에는 쉘, 디버거, 모니터등의 타겟에 대한 정보나 명령을 요구하는 도구들이 있고, 호스트-타겟 연결 미들웨어 역할을 하는 타겟서버를 통해 타겟시스템의 디버그 에이전트와 연결되어 통신한다. 디버그 에이전트는 타겟시스템에서 호스트의 타겟서버와 연결되어 통신하고 호스트 도구의 요구사항에 대하여 서비스를 제공하는 모듈이다. 그리고 타겟시스템에는 VxWorks라는 운영체제 위에 호스트에서 로딩된 응용프로그램이 수행된다. 또한 타겟서버는 쉘이나 디버거의 요구 중 내부모듈에서 해결이 가능한 요구에 대해서 타겟과의 통신 없이 실행된다.

Tornado는 dll 라이브러리 형태로 타겟서버를 지원하는데 본 논문에서 개발한 타겟서버로 교체후 테스팅하였다.

[그림 2]는 이러한 테스팅 환경을 보여준다.

[그림 2]의 왼쪽과 같이 기존의 타겟서버로 Tornado를 구동시킨다. 타겟서버는 전단부, 후단부 인터페이스를 초기화하고 타겟의 디버그 에이전트와 연결된다. 그리고 나서 쉘에서 임의의 목적화일을 로딩하면, 개발된 타겟서버 내부모듈로 제어가 이동된다. 개발된 타겟서버에서는 로딩될 목적모듈들을 지정하여 로드시키고 타겟에서 실행시킨 후, 언로드하는 루틴을 구현하여 내부모듈의 테스팅을 수행하였다.



[그림 2] 타겟서버 테스팅 환경

5. 결론 및 향후 발전 방향

strongARM SA-110 프로세서가 탑재된 타겟 시스템에 coff 포맷 목적모듈을 로드시키는 타겟서버의 내부모듈로서 리더와 로더, 타겟메모리 관리모듈, 심볼테이블 관리모듈을 구현하였다. 구현된 타겟서버는 기존의 내장형 시스템 응용 프로그램 개발환경을 사용하여 실험하였다.

호스트와 타겟을 연결하는 타겟서버의 구현으로 내장형 시스템 응용 프로그램 개발환경에 대한 기술을 축적할 수 있었다. 호스트에서의 타겟 정보관리 기능을 구현함으로서 내장형 시스템에 대한 응용 프로그램 개발환경 구축에 기여할 것으로 기대된다.

우리가 구현한 타겟서버에서의 캐시는 타겟서버 구동 시 타겟메모리 정보와 같은 크기로 구성되고 타겟메모리 풀의 크기 변화에 따라 캐시도 변하는 가변크기 구조로 구현되어 있다. 캐시크기가 가변적이기 때문에 호스트 메모리의 대부분을 캐시가 차지할 수도 있다. 이러한 문제점을 고려한 고정된 크기의 캐시를 구현하는 것이 향후 연구과제이다.

참고문헌

1. Tornado, WindRiver Systems, http://www.wrs.com
2. MULTI, Green Hills Software, http://www.ghs.com
3. FasTrak, Microware Systems, http://www.microware.com
4. Spectra, Mentor Graphics, http://www.mentor.com/embedded
5. pRISM+, Integrated Systems, http://www.isi.com
6. *Understanding and Using COFF*, Gintaras R.Gircys, O'Reilly & Associates, Inc., Aug, 1998
7. *The Binary File Descriptor Library*, Steve Chamberlain , Cygnus Support, Apr, 1991
8. *ARM Architectural Reference Manual*, Dave Jaggar, Prentice Hall, Jul, 1996
9. *Tornado User's Guide*, WindRiver Systems, Jan, 1996
10. *Tornado API Guide*, WindRiver Systems, Mar, 1997