임베디드 리눅스용 응용 소프트웨어 개발을 위한 IDE 구현

우덕균[0], 임채덕, 김흥남
한국전자통신연구원 정보가전연구부
(dkwu[0], cdlim, hnkim)@etri.re.kr
표창우
홍익대학교 컴퓨터공학과
pyo@hongik.ac.kr

Implementation of IDE for Developing Application Software for Embedded Linux

Deok-Kyun Wu[0], Chaedeok Lim, Heung-Nam Kim
Information Appliance Technology Dept., Electronics and Telecommunications Research Institute
Changwoo Pyo
Computer Engineering Dept. Hongik University

요 약

IDE(Integrated Development Environment)는 임베디드 소프트웨어 개발의 생산성과 코드 질을 높이는 데 중요한 역할을 한다[1]. 그러나 임베디드 리눅스용 응용 소프트웨어 개발은 통합된 개발 환경보다는 명령 라인 기반의 개발 도구들을 통하여 개발되고 있다[2]. 이와 같은 개발 환경은 사용자에게 불편함을 주어 프로그램 개발의 생산성을 저하시킬 수 있다. 본 연구에서는 이러한 문제를 해결하기 위한 임베디드 리눅스용 응용 소프트웨어 개발을 위한 IDE를 개발하였다. 본 연구의 IDE는 소스 프로그램 편집, 컴파일, 실행 등의 기능을 프로젝트 기반으로 수행하며, 원격 디버거, 원격 모니터 등의 다른 독립적인 개발 도구들을 관리하는 기능을 포함한다. 본 연구의 IDE는 ETRI에서 개발 중인 정보가전용 응용 소프트웨어 통합 개발 환경인 Esto에[3] 포함되어 구현되었다.

1. 서론

임베디드 소프트웨어 개발의 생산성과 코드 질을 높이기 위하여 소프트웨어 개발 도구들의 통합 개발 환경을 제공하는 IDE의 중요성은 강조되고 있다[1, 4, 5]. WindowsCE[6], VxWorks[7]와 같은 기존의 임베디드 운영체제에 대한 개발 환경은 모두 IDE를 지원하고 있다. 그러나 이와 같은 상업용 임베디드 운영체제는 로열티 등의 문제로 공개 소스인 리눅스를 임베디드 시스템의 운영체제로 사용하기 위한 연구가 활발히 진행되고 있으며, 이미 임베디드 리눅스를 탑재한 PDA와 같은 임베디드 시스템이 상용 제품으로 출시되고 있다.

임베디드 리눅스에서 수행되는 응용 소프트웨어 개발은 주로 데스크탑 PC의 리눅스 환경에서 이루어진다. 소프트웨어 개발자는 리눅스에서 vi 또는 emacs와 같은 소스 편집기를 사용하여 소스 프로그램을 작성하고, 작성된 소스 프로그램은 gcc 크로스 컴파일러를 통하여 컴파일, 링크하여 실행 파일을 생성하고, 실행 파일은 gdb를 통하여 타겟이 되는 임베디드 시스템으로 다운로딩, 디버깅한다[]. 개발자는 이와 같은 과정을 반복하면서 응용 소프트웨어를 개발한다. 이와 같은 개발 과정에서 사용자는 명령 라인에서 명령어를 일일이 입력해야 한다. 이것은 개발자에게 소프트웨어 개발의 불편함을 가져오며, 결국 소프트웨어 개발 시간을 증가시킬 수 있다.

본 연구에서는 이와 같은 문제점을 해결하기 위하여 프로젝트 중심의 GUI 기반의 통합 개발 환경을 개발하였다. 본 연구의 통합 개발 환경은 소스 편집기, 프로젝트 관리자를 기본 도구로 개발되었다.

2. 임베디드 리눅스 개발 환경

임베디드 시스템은 일반적으로 응용 프로그램 개발 환경 및 개발 도구들을 포함하기에는 부족한 자원을 갖는다. 따라서 임베디드 응용 프로그램 개발은 데스크탑 PC와 같은 호스트 시스템에서 수행되고, 개발된 모듈은 테스팅, 디버깅을 위하여 임베디드 시스템으로 다운로딩되어 실행된다. 임베디드 리눅스를 탑재하는 임베디드 시스템의 응용 프로그램 개발 환경도 주로 호스트 시스템에 위치한다.

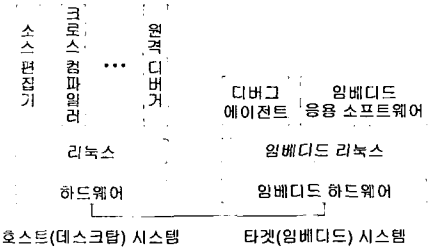

그림 1 임베디드 리눅스 개발 환경

그림 1은 임베디드 리눅스를 타겟으로 하는 리눅스 환경의 임베디드 리눅스 개발 환경을 나타낸다. 임베디드 리눅스용 응용 소프트웨어 개발은 주로 리눅스 데스크탑 PC 환경에서 수행된다[2]. 개발 도구는 소스 편집기, 크로스 컴파일러, 원격 디버거 등으로 구성되고, 타겟에는 원격 디버거와 연동하는 디버그 에이전트가 수행된다.

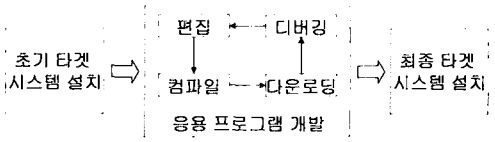

그림 2 임베디드 리눅스용 응용 프로그램 개발 주기

임베디드 리눅스용 응용 프로그램은 일반적으로 그림 2와 같은 단계로 진행된다. 크게 초기 타겟 시스템 설치 단계, 응용 프로그램 개발 단계, 최종 타겟 시스템 설치 단계로 구성된다. 타겟 시스템 설치는 임베디드 리눅스 커널 및 라이브러리, 기타 응용 패키지 설정 및 빌드, 루트 파일 시스템 설치 등의 응용 프로그램이 수행될 수 있는 전반적인 타겟 시스템을 설치하는 과정을 말한다[2].

초기 타겟 시스템 설치 단계는 타겟 응용 프로그램 개발과 실행에 필요한 타겟 시스템 설치 단계를 말하고, 최종 타겟 시스템 설치 단계는 개발된 응용 프로그램을 포함하고 응용 프로그램의 실행에 필요한 최소한의 타겟 시스템 설치 단계를 말한다.

응용 프로그램 개발 단계는 소스 편집기에 의한 소스 프로그램 편집, 크로스 컴파일러를 통한 소스 프로그램의 크로스 컴파일, 원격 디버거를 통한 컴파일된 모듈의 타겟으로 다운로딩, 디버깅을 반복하는 과정을 말한다. 원격 디버거는 타겟의 디버그 에이전트와 연동하여 디버깅 관련 타겟 서비스를 제공한다.

이와 같은 리눅스 개발 환경의 각 도구들은 일반적으로 통합된 환경에서 진행되기 보다는 서로 독립적인 쉘 상의 명령어에 의하여 수행된다[2]. 이와 같은 명령어 기반의 독립적인 개발 환경은 개발자가 하나하나 명령어를 사용하여 개발 과정을 진행해야 하므로 응용 소프트웨어 개발에 불편함을 줄 수 있고, 결국 응용 소프트웨어 개발 시간을 증가시킬 수 있다. 본 연구에서는 이와 같은 문제점을 해결하기 위하여 임베디드 리눅스용 통합 개발 환경을 개발하였다.

본 논문에서는 그림 2의 응용 프로그램 개발 주기에서 첫 번째 단계인 초기 타겟 시스템 설치는 이미 수행되었다고 가정하고, 두 번째 단계인 응용 프로그램 개발에 대하여 논하기로 한다.

3. IDE 설계 및 구현

3.1 IDE 구조

IDE(Integrated Development Environment)는 응용 프로그램 개발을 위한 통합 개발 환경을 말한다. 본 연구의 임베디드 리눅스용 응용 소프트웨어 개발을 위한 IDE는 소스 프로그램 작성, 컴파일, 링킹, 실행 등을 하나의 통합된 환경에서 진행하기 위하여 프로젝트 개념을 도입하였다. 모든 응용 프로그램은 프로젝트 단위로 구성되며, 프로젝트는 소스 파일 정보, 컴파일, 링킹 정보, 실행 정보 등을 포함한다. 이와 같은 프로젝트의 생성, 편집, 빌드, 실행 등을 관리하는 도구를 본 논문에서는 소스 프로젝트 관리자(source project manager)라 한다. 프로젝트 빌드는 프로젝트를 구성하는 소스 파일의 컴파일, 링킹을 통하여 최종 실행 파일을 만드는 것을 말한다.

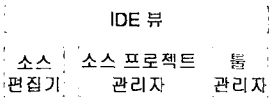

그림 3 IDE 구조도

본 연구의 IDE는 그림 3과 같이 구성된다. IDE 뷰는 IDE의 사용자 인터페이스를 말하며, GUI(Graphical User Interface)로 구성된다. IDE 내부 모듈인 소스 편집기, 소스 프로젝트 관리자, 툴 관리자의 사용자 인터페이스를 제공한다.

소스 편집기는 텍스트 문서를 편집할 수 있는 편집기를 말하며, 사용자에게 C 또는 C++ 프로그램 작성에 편의성을 주기 위하여 C 또는 C++ 언어에 대한 구문 하이라이팅(syntax highlighting) 기능과 자동 들여쓰기(auto indentation) 기능 등을 제공한다. 소스 파일 내 또는 소스 파일 전체에 대한 문자열 검색, 치환 기능을 제공한다.

소스 프로젝트 관리자는 소스 프로젝트의 생성, 편집, 빌드, 실행 등을 담당하는 도구이다. 소스 프로젝트는 응용 프로그램 또는 응용 라이브러리 생성을 위한 기본 단위이며, 소스 정보, 빌드 정보, 실행 정보 등을 포함한다. 소스 정보는 소스 파일, 헤더 파일, 라이브러리 파일, 관련 프로젝트 파일의 경로명 등을 포함한다. 빌드 정보는 크로스 컴파일러 이름, 크로스 링커 이름, 헤더 파일 디렉토리, 라이브러리 파일 디렉토리, 컴파일 옵션, 링크 옵션 등을 포함한다. 실행 정보는 타겟 파일 이름과 경로명, 실행 인자, 실행 환경 변수 등을 포함한다. 타겟 파일은 소스 프로젝트로부터 빌드된 최종 파일이 타겟으로 다운로딩되어 타겟 디렉토리로 복사되는 파일을 말한다.

응용 프로그램의 빌드를 담당하는 원격 디버거는 IDE와는 독립적으로 수행되는 개발 도구이다[8]. 즉, 원격 디버거의 사용자 인터페이스는 IDE 뷰에 나타나지 않고, 단지 원격 디버거 실행 메뉴만 IDE 뷰에 나타난다. 이와 같이 IDE와는 독립적으로 수행되는 개발 도구들은 IDE의 툴 관리자에 의하여 관리된다. 툴 관리자는 IDE와는 독립적으로 수행되는 개발 도구들의 실행, 종료 등의 명령을 수행한다.

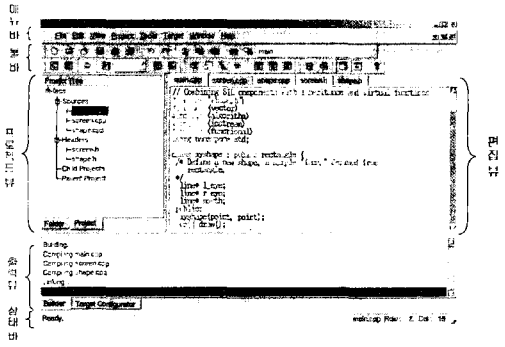

그림 4 IDE 뷰

3.2 IDE 뷰

IDE의 소스 편집기, 소스 프로젝트 관리자, 툴 관리자는 단일한 통합된 모습으로 사용자에게 보여지는 데, 이와 같은 뷰를 본 논문에서는 IDE 뷰라 한다. 그림 4는 IDE의 실행 화면 모습을 보여준다. 위 실행 창은 3개의 소스 파일과 2개의 헤더 파일로 구성되는 소스 프로젝트의 빌드 모습을 보여준다.

창 상단의 메뉴 바는 풀다운 메뉴로 구성되고 통합 개발 환경 관련 모든 명령들이 메뉴 형태로 제공된다. 이와 같은 메뉴 중에 자주 사용되는 메뉴는 메뉴 바 하단의 툴 바에 아이콘 메뉴 형태로 제공된다. 툴 바 아래 좌측의 프로젝트 뷰는 프로젝트의 소스 트리를 나타내고, 우측의 편집 뷰는 소스 프로그램을 편집할 수 있는 편집 공간을 나타내다. 편집 뷰 상단에는 현재 열려진 문서들의 파일 이름들이 탭 형태로 표시된다. 편집 부 아래의 출력 뷰는 프로젝트 빌드, 문자열 검색 등의 결과를 보여주는 출력 공간을 나타낸다. 출력 뷰 아래의 상태 바 좌측은 통합 개발 환경 명령 수행 전후의 메시지들이 나타나고, 우측은 편집 파일에 대한 정보가 나타난다.

3.3 IDE 구현

본 연구의 IDE는 현재 한국전자통신연구원에서 개발되는 정보가전용 응용 소프트웨어 통합 개발 환경인 Esto(Embedded System Tools)의 모듈로 구현되었다. Esto의 구조는 그림 5와 같다[3].

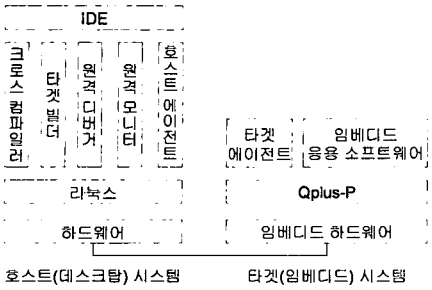

그림 5 Esto 구조도

Esto는 그림 1의 임베디드 리눅스 개발 환경과 마찬가지로 호스트-타겟으로 구성되는 개발 환경 모습을 갖는다. 호스트 쪽의 개발 환경을 살펴보면, 상단의 본 연구의 IDE와 하단에는 독립적인 개발 도구들인 크로스 컴파일러, 원격 디버거, 원격 모니터, 호스트 에이전트가 위치한다. 타겟 쪽을 살펴보면, 하드웨어 위에 Qplus-P가 위치하고, 디버그 에이전트 자리에 타겟 에이전트가 위치하였다. Qplus-P는 한국전자통신연구원에서 개발한 임베디드 리눅스를 말한다.

타겟 에이전트는 기존 디버그 에이전트가 하던 디버깅 관련 서비스 뿐만 아니라 프로그램 실행 등의 다양한 타겟 서비스를 제공한다[9]. 타겟 에이전트는 호스트 시스템의 호스트 에이전트와 연동하여 동작한다. 호스트 에이전트는 소스 프로젝트 관리자, 원격 디버거 등의 호스트 상의 도구들에게 타겟 서비스를 제공하는 일종의 미들웨어 역할을 수행하는 도구이다[10].

크로스 컴파일러는 IDE의 소스 프로젝트 관리자가 소스 프로젝트를 빌드하는 데 사용하는 도구이며, 원격 모니터는 타겟 시스템의 자원 상태, 실행 상태 등을 모니터링하는 도구를 말한다. 타겟 빌더는 커널, 커널 모듈, 패키지 등의 설정, 빌드, 설치와 루트 파일 시스템의 설치 등을 담당하는 도구를 말한다.

본 연구의 IDE를 포함하는 Esto는 RedHat 리눅스 7.1에서 개발되었으며, GUI 라이브러리는 QT 3.0을[11] 사용하였다 타겟 시스템은 x86 기반의 홈 서버 셋탑을 사용하였고, Qplus-P는 리눅스 커널 버전 2.4.3 기반을 바탕으로 한다. 표준 C 라이브러리는 GNU의 glibc 2.2.3 버전을 사용하였고, 크로스 컴파일러는 GNU의 gcc 2.95.3을 사용하였다.

4. 결론 및 향후 연구방향

임베디드 리눅스용 응용 소프트웨어 개발 도구들의 사용 편의성을 높이기 위하여 소스 프로젝트 중심의 GUI 기반 IDE를 개발하였다. 본 연구의 IDE는 GUI 기반으로 메인 창에서 단일한 인터페이스를 통하여 손쉽게 소스 프로그램 작성, 컴파일, 실행 등을 제공하여 사용자에게 임베디드 리눅스용 응용 프로그램 개발 시간을 단축시켜 임베디드 소프트웨어 개발 생산성을 증가시킨다.

현재 리눅스에서 수행되고 있는 통합 개발 환경을 윈도우로 확장하는 중이다. 그리고 통합 개발 환경에 새로운 개발 도구가 포함될 때, 새로운 도구를 추가하는 오버헤드를 최소화하는 방향으로 연구 중에 있다.

참고문헌

[1] P. Varhol, "Integrated software tools improve productivity and code quality", Electronic Design, pp. 62-70, Oct. 1999.

[2] J. Epplin, "A developer's review of the leading embedded linux toolkits", http://www.linuxdevices.com, Sep. 2001.

[3] 임채덕, "Esto : Qplus 통합개발환경", 한국정보과학회컴퓨터시스템연구회 주관 2002년 동계 워크샵 발표집, pp. 116-143, 2002년 1월.

[4] L.P. Maguire, T.M. McGinnity, L.J. McDaid, "Issues in the development of an integrated environment for embedded system design", Microprocessor and Microsystems, pp. 191-206, Vol 23, 1999.

[5] S.V. Tyle, "Engineering software tools meet demands", Electronic Design, pp. 71-80, Jun. 1994.

[6] MicroSoft, http://www.microsoft.com/windows/embedded.

[7] WindRiver Systems, "VxWorks Programmer's Guide", http://www.wrs.com.

[8] K.Y. Lee, etc. "A design and implementation of a remote debugging environment for embedded internet software", ACM SIGPLAN 2000 Workshop on LCTES, Jun. 2000.

[9] 임형택외 4인, "Q+P Esto에서 원격 개발을 지원하는 타겟에이전트", 2001년 정보처리학회추계학술발표논문집, pp. 671-674, 2001년 10월.

[10] C.D. Lim, etc. "Middleware for Platform Independent Toolset to Develop Real-Time Embedded Applications", Proceedings of RTCSA, Mar. 2002.

[11] Trolltech, "Qt 3.0 Whitepaper", Available in http://www.trolltech.com/products/qt/whitepaper/whitepaper.html.