RTOS 용 원격 대화형쉘 설계 및 구현

김 대 희[†]·남 영 광[††]·김 흥 남[†††]·이 광 용[††††]

요 약

내장형 시스템의 실시간 운영체제(RTOS)에서는 메모리와 CPU 속도 등의 제한적인 환경에서 원하는 기능을 최적으로 최단시간에 구현할 수 있는 개방형 개발도구 환경이 요구된다. 이러한 개방형 개발환경에는 타겟 시스템에 최소의 부담을 주면서 사용자가 편리하게 원하는 정보를 원격지에서 대화식으로 빠르게 접근, 참조, 제어 할 수 있는 원격 대화형 쉘이 필수적이다. 본 논문에서는 원격지에서 모듈별 로딩, 태스크의 스폰과 디불어 태스크 상태 등을 확인할 수 있는 원격 대화형 쉘 프로그램의 설계와 그에 대한 구현방법을 기술하고 원격지에서 타겟의 부담을 최소화하여 실시간 운영체계에서 정보접근이 용이하고 유지보수가 쉬운 개발환경이 가능하도록 하였다. 이 대화형 쉘은 Q-PLUS RTOS와 연동되어 작동하며 ARM계열의 EBSA285 타겟보드와 NT 호스트상에서 구현, 테스트되었다.

Development of a Remote Interactive Shell for RTOS

Dae-Hee Kim[†]·Young-Kwang Nam[††]·Heung-Nam Kim[†††]·Kwang-Yong Lee[††††]

ABSTRACT

Recently, the Open-Development-Tool-Environment becomes a basic requirement of RTOS (Real Time Operating System) for embedded systems with restricted memory and CPU power in order to develop applications effectively. A remote interactive shell is one of the basic software components which makes users develop, test and control softwares without burdening target systems. In this paper, we have implemented the remote interactive shell with the following functions ; loading object modules, spawning and manipulating tasks facilities thru a remote host. Comparing information reference methods with nonredundant overhead, we have achieved the system with easy maintenance. The shell has been developed with Q-PLUS RTOS under ARM EBSA285 target board and NT host.

키워드 : 실시간 운영체계(RTOS), 내장형 시스템(Embedded System), 개방형통합개발도구(Open-Development-Tool-Environment), 대화형쉘(Interactive Shell)

1. 서 론

실시간 운영체계는 최근 정보가전기기의 급속한 발전에 따라 사용상 편의성과 성능을 극대화한 고기능 저가격의 개발을 필요로 하고 있다. 이에 따라 실시간 운영체계에 적용되는 응용 프로그램의 개발에 있어서 사용자의 편의성을 최대한으로 도모하기 위하여 원격지에서 사용자가 원하는 작업을 쉽게 할 수 있도록 하는 원격지 통합개발도구의 개발이 필수적이며 현재 상용되고 있는 타제품과의 차별화를 위한 연구가 함께 병행되어 수행되고 있다[1, 2].

대화형 쉘은 원격지 통합개발도구의 사용자도구들 중 호스트에서 원격으로 시리얼 또는 이더넷으로 연결되어 타겟보드와 연동시켜 제어명령과 시스템정보 출력명령을 수행하는 기능을 가진다. 내장형 시스템의 실시간 운영체제를 개발하기 위해서는 타겟 시스템이 원격지에 있으므로 호스트 상에서 개발한 모듈을 타겟에 적재하고 이를 수행시키는 기능이 필요하며 또 필요 시 이 모듈만을 대화형 쉘을 통하여 언로딩(unloading)하고 쉘 명령어를 호스트 상에서 수행시키는 기능이 필요하다. 이러한 기능을 가진 대화형 쉘을 호스트/타겟으로 구성된 사용자 개발도구들과 함께 구현함으로써 내장형 시스템의 소프트웨어의 개발 및 디버깅을 용이하게 할 수 있다.

본 연구에서는 타겟 시스템과 연결되어 있는 타겟 서버와 대화형 쉘을 연결하여 사용자가 원하는 시스템 데이터의 정보를 출력하고 타겟 시스템의 쉘 명령어를 호스트에서 실행 가능하게 하며 타겟 시스템을 제어하는 명령을 개발하여 GUI 환경과 접목시켜 사용자가 원하는 작업을 원격

† 준 회 원 : LG 차세대단말연구소
†† 정 회 원 : 연세대학교 전산학과 교수
††† 정 회 원 : ETRI 컴퓨터 소프트웨어기술연구소 인터넷정보가전연구부 책임연구원
†††† 정 회 원 : ETRI 컴퓨터 소프트웨어기술연구소 인터넷정보가전연구부 선임연구원
논문접수 : 2000년 12월 18일, 심사완료 : 2002년 4월 19일

지에서 편리하게 할 수 있는 원격대화형 쉘의 연구개발을 목적으로 한다. 이에 원격 대화형 쉘 프로그램의 설계와 그에 대한 구현방법을 기술하고 원격지에서 타겟의 부담을 최소화하는 방향의 정보참조방법을 비교 분석하여 실시간 운영체계와 더불어 개발환경의 빠른 상호유지보수를 가능하게 하고자 한다.

2. 관련 연구 및 사례와 배경

2.1 기개발된 RTOS와 개발환경의 종류

2.1.1 Tornado(WindRiver System)

Tornado는 세 개의 집적화된 구성요소 집합들로 구성되어 있는데 첫 번째는 Tornado의 도구들의 집합으로 교차개발(cross-development) 도구와 유틸리티들의 집합이고 두 번째는 타겟 프로세서에서 실행되는 실시간 운영체제인 VxWorks 실시간 시스템과 세 번째로 Ethernet, serial line, in circuit emulator (ICE) 그리고 ROM emulator와 같은 호스트와 타겟 간의 전 부분의 통신의 옵션으로 구성되어 있다. WindRiver System사는 제한된 자원과 호스트와 타겟 간의 제한된 통신의 문제점을 해결하기 위하여 과거의 타겟에 의존하던 개발 도구들(예 : shell, loader, symbol table 등)을 호스트로 옮겼다. 따라서 시스템은 더 이상 호스트와 타겟 간의 정보를 전송하고 재수정 하는데 필요했던 부가적인 시간과 높은 대역폭이 필요 없게 됨으로 해서 부수적으로 타겟 시스템은 도구들에 의해 점유되었던 자원들(예 메모리)을 좀더 활용할 수 있게 되었다. 이러한 것을 가능하게 한 기술은 바로 타겟 서버와 타겟 에이전트라는 구성 요소에 있다[3].

2.1.2 ITRON (TRON Association)

TRON(The Real-Time Operating system Neucleus)은 일본 도쿄대학에서 1984년에 시작되어 1987년 첫 번째 ITRON 설계서가 1987년에 수립되어 ITRON1이라고 명명되었고 그 응용성을 증명하기 위해 많은 시스템에 적용되었다. 이에 따라 8-bit, 16-bit MCUs(Microcontrol Units)에 적용되는 작은 기능들을 첨가한 ITRON(ver 2.0)을 1989년에 개발했고 이와 함께 32-bit 프로세서를 위한 설계를 마쳤다. 1993년 8-bit에서 32-bit 범위의 MCU에 이용될 수 있는 ITRON3.0이 출시되고 30개 이상의 프로세서와 40개 이상의 제품에 적용되어 현재 이용되고 있다[4].

현재는 5개의 기초 서브프로젝트와 5개의 응용프로그램 서브프로젝트를 진행중인데 이것들은 VLSI 마이크로프로세서의 구조에 관한 TRON-specification CHIP, 내장되는 제어프로그램을 위한 실시간 다중 태스크 OS에 관한 ITRON 연구(Indistrial TRON), 개인용 컴퓨터나 워크스테이션에 이용되는 인간과 기계의 인터페이스에 관한 연구인 BTRON (Business TRON), 정보의 전송이나 처리를 위한 OS 인터페이스인 CTRON(Central / Communication TRON), 프로그램이 가능한 인터페이스 개발에 가장 중요점이 있는 MTRON(Macro TRON)등을 포함하고 있다. 위와 같은 연구와 함께 응용프로그램 디자인 인터페이스인 디버깅 환경과 ITRON-specification 커널과 C++ 언어의 결합에 관한 연구도 진행중이다.

2.1.3 Precise/MQX(Precise Software Technologies)

1984년부터 내장형 다중프로세서의 응용프로그램을 연구하기 시작한 Precise Software는 1989년 포터블 다중프로세서 실시간 시스템을 위해 The National Research Council Canada에서 개발한 Harmony Real-Time Operating System을 발표하였고 이러한 노력으로 인하여 Precise 실시간 운영 시스템 개발기술을 근거로 1991년 Precise/MXQ 실시간 운영 시스템을 소개하였다. Precise/MXQ 시스템의 근간이 되어 있는 Harmony 운영체제는 사실 프로그램 개발 환경을 제공하기 위한 것은 아니었기 때문에 호스트는 자기 자신의 OS에서 프로그램 개발을 완료하고 테스트 실행은 실행 가능한 프로그램만을 타겟 프로세서에 다운로드하여 수행한다. GUI환경으로 시작하였으나 전체적인 통합에 주력하고 있는 실정이며 이 제품은 포인트와 클릭을 기초로 한 GUI 환경을 디자인 도구와 에디터, 시뮬레이터, C/C++ Cross Compilers, C/C++ Cross Debuggers, 태스크 수행 분석 등에 제공한다. 이 제품은 위에서 언급한 Precise/MQX RTOS와 Precise/Design Tool(Cross Compiers), Precise/MQX RTOS Simulator(Cross Debugger), Precise/MTD Task Aware Debugging(Source Code Editor), Embedded I/O Components 그리고 Precise/Performance Analysis Tool로 이루어져 있다[5].

2.1.4 Virtuoso (Eonic System inc.)

1990년 트랜스퓨터용으로 첫 번째 실시간 커널을 소개한 Eonic System은 1992년 Virture Single Processor 프로그램용 모델인 Virtuoso Micro를 개발하고 1993년 어셈블리 레벨 프로그램을 제공하는 유일한 커널이었던 Virtuoso Nano를 발표하였다.

Virtuoso의 구조를 보면 쉬운 프로그래밍과 빠른 속도의 수행을 조합하는 다중레벨 구조를 가지고 있는데 첫 번째 레벨은 Nanokernel 프로세스라 불리는 경량의 태스크를 위한 것이고 마지막 레벨은 C 태스크를 위한 우선순위와 선수행 태스킹을 제공하는 부분으로 구성되어 있고 마지막 레벨에서 쓰여진 코드들은 다른 플랫폼에서 혹은 다른 타겟 프로세서에서 수행될 수 있는 이식성을 가지고 있다[6].

Virtuoso의 버전 4.0과 함께 이들은 SoftSteath라는 이름으로 새 기술을 발표하였는데 이것은 커널로부터 사용되지

않는 코드 부분을 자동적으로 제거하고 더 작은 수행을 가능하게 하는 결과를 내는 것으로 시스템으로부터 사용되지 않는 루틴들을 제거함으로써 메모리를 프로그램의 다른 부분에서 사용할 수 있게 하였다.

2.2 대화형 쉘을 포함하는 통합개발환경

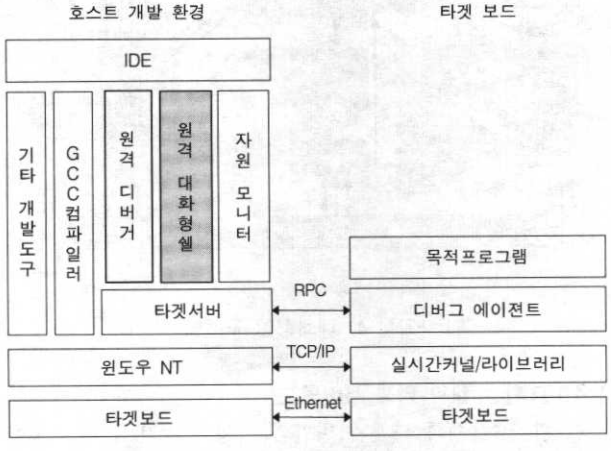
WindRiver 시스템사의 RTOS Tornado의 구조는 개발자와 도구들에게 개방적 환경을 제공하기 위해 설계되어 GUI 인터페이스로부터 연결의 구현까지 몇 가지의 API가 사용가능하며 GUI 레벨에서는 도구들의 확장성과 변경을 제공하기 위하여 Tcl(Tool command language)에 기초한 API가 사용되었고[7, 8] 다음 단계의 API는 호스트에서 사용이 가능한 모든 타겟 정보에 관한 인터페이스를 제공한다. 이러한 모듈방식구조에 대한 접근방식은 개방적인 통합 방식을 가능하게 하였다[9].

WindSh 쉘은 대화식의 VxWorks OS의 참조와 프로토타입핑, 대화식 개발 그리고 테스팅을 할 수 있도록 개발되었는데 WindSh는 대부분의 C 언어 수식을 처리할 수 있고 또한 대부분의 C 연산자의 실행과 서브루틴의 활용이 가능하다. 다른 방법으로는 Tcl 인터프리터를 통하여 쉘과 상호활용 할 수 있다. 이 Tcl 인터프리터는 Windows와 Tcl 유틸리티 프로그램 액세스와 함께 전체적인 제어 구조를 제공한다[12, 13].

3. 원격 대화형 쉘의 개요

3.1 통합개발환경의 구조

RTOS용 응용프로그램을 위한 통합환경은 과거 타겟 기반 텍스트형으로부터 점차 원격지 호스트 기반 GUI 형태로 발전하고 있다[15]. 이 방법은 (그림 1)과 같이 호스트의 타겟 서버와 타겟의 디버그 에이전트(debug agent)의 통신에 의해 이루어진다[21]. 타겟 서버에서는 원격지에 존재하는

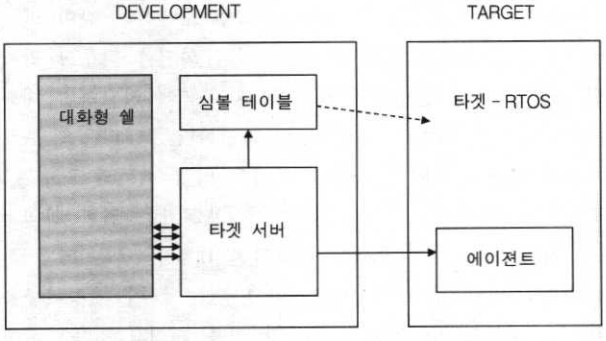


(그림 1) RTOS용 통합개발도구의 구조

개발도구들과 타겟과의 통신을 중간에서 연결하고 디버그 에이전트로부터의 타겟 정보를 제어한다. 타겟 서버는 모듈로더 및 심볼 테이블을 통해 타겟과의 일관성 유지 등을 담당한다.

3.2 통합개발도구 시스템에서의 대화형 쉘

대화형 쉘은 사용자가 직접적으로 타겟 시스템을 제어할 수 있도록 라인명령어 방식으로 운영되는데 이것은 사용자의 명령어를 받아 호스트 상에서 실행시키고 타겟 서버에게 심볼 테이블 혹은 프로그램이나 타겟에 상주하고 있는 데이터에 관한 요구사항을 보내어 그 결과를 사용자에게 출력하는 방법으로 수행된다. 타겟과의 연결을 위해 서버와 에이전트 개념의 구조가 되면서 사용자 도구의 통합의 가능하다. 통합개발도구 내에서의 대화형 쉘의 구조는 (그림 2)와 같다.



(그림 2) 통합개발도구에서의 대화형쉘

3.3 대화형 쉘의 기능

3.3.1 데이터 변환

- 정수나 문자를 십진수나 16진수로 변환하여 표시한다.
- 정수나 문자를 문자 상수로 변환하거나 심볼 주소와 오프셋으로 변환한다.

3.3.2 산술 연산

- 모든 C 언어 산술 연산자에 대한 산술 연산이 가능하다.
- 산술연산에는 선행 연산 기능이 포함되어 있다. 연산 순서지정은 괄호를 이용한다.
- 비트 이동과 비트 연산을 포함한다.

3.3.3 변수를 포함한 산술 연산

- 타겟 내의 변수를 사용하여 모든 C언어 연산자에 대한 산술연산을 수행한다.
- 새로운 변수를 이용하여 어떤 결과 값을 타겟에 설정할 수 있다.

3.3.4 커널 및 라이브러리 함수 호출

- 함수를 타겟 서버와 에이전트를 통하여 호출하고 반환 값을 위한 변수사용이 가능하다.

3.3.5 응용 API 및 응용 프로그램을 호출

- 타겟에 적재한 함수를 호출하고 반환 값을 저장할 수 있는 변수 사용이 가능하다.

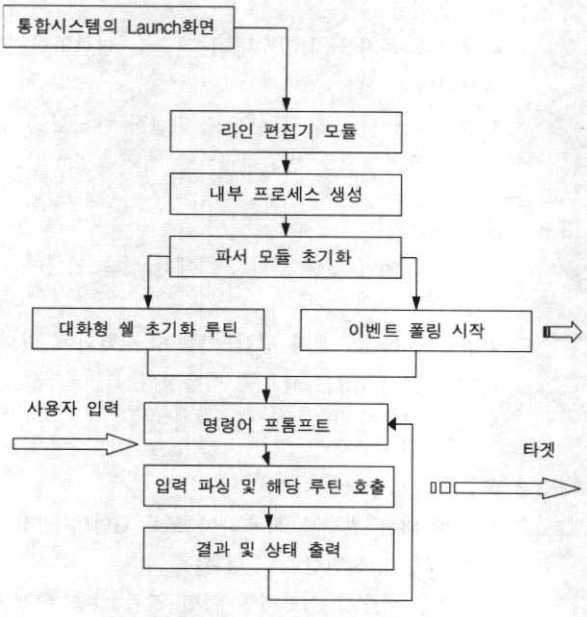
3.3.6 쉘 기본 함수의 호출

- 대화형 쉘 자체에 정의되어 있는 함수를 호출하여 결과를 표시한다.
- 시스템 제어 명령어를 실행하여 타겟 시스템을 변환하고 관리한다.

4. 원격 대화형 쉘의 설계

4.1 대화형 쉘 구동 시나리오

대화형 쉘 프로그램은 시작은 (그림 3)에서와 같이 쉘을 구동시키기 위하여 통합시스템의 시작 화면을 띄우고 연결할 타겟 서버를 설정하여 대화형 쉘 블록에 전달하면 대화형 쉘 블록은 타겟 서버의 ID를 확인한다. 그 다음에 프로토콜의 개방형 API 핸들을 마련하고 타겟 서버와 도구 ID를 이용하여 다른 시스템(타겟)과 정보교환을 위한 인터페이스가 초기화된다. 구동이 시작되면 대화형 쉘은 타겟 접근을 위한 쉘 자체 명령을 가지고 있는데 이 명령어들을 파서가 구분할 수 있도록 파서의 심볼 테이블에 명령어들의 엔트리를 마련하는데 이때 파서는 C 수식을 위한 연산자나 기본함수에 대한 초기화작업을 한다.

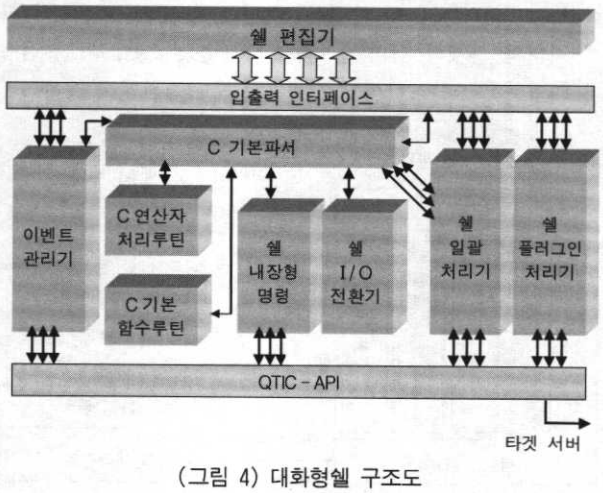


(그림 3) 대화형쉘 시나리오

4.2 원격 대화형 쉘의 구조

대화형 쉘은 개방형 도구에 속해 있으면서 개방형 개발 도구들과 함께 타겟 서버와 서버간 프로토콜을 기반한 API들의 모듈을 통해 서버와 통신할 수 있는 구조를 가진다. 쉘 내부에는 GUI 라인 편집기 모듈과 C 인터프리터가 있다. 이 두 모듈은 GUI형식으로서 로그(log) 파일 생성과 같은 기타의 작업 없이 사용자가 원하는 부분만 저장할 수 있도록 한다. C 인터프리터는 편집기 모듈로부터 사용자의 입력을 라인명령어 방식으로 넘겨받아 C 기본 파서를 통해 파싱하고 각각의 토큰의 요구에 필요한 처리루틴을 호출하여 처리한 후 그 결과를 돌려 받아 사용자 인터페이스인 편집기에 출력한다[14, 18]. (그림 4)는 대화형 쉘의 시스템 구성도로서 각 모듈의 기능은 다음과 같다.

① 쉘 편집기 : 편집과 히스토리(history) 기능을 포함하여 인터프리터 모듈에 명령어를 제공한다.
② C 파서 : 입력된 명령어 라인을 파싱하고 필요한 함수를 호출한다.
③ 쉘 내장형 명령어 : 타겟 시스템의 정보참조와 제어함수를 포함한다.
④ 이벤트 관리기 : 타겟 시스템으로부터의 이벤트를 사용자에게 출력한다.
⑤ 쉘 I/O 전환기 : 파일로부터 입력과 출력을 전환하는 기능을 갖는다.
⑥ 일괄처리기 : 간단한 제어구문을 포함한 일괄처리파일로부터 일괄처리를 가능케 한다.
⑦ 쉘 플러그인 처리기 : 사용자정의명령을 프로세스호출 방식으로 사용 가능하게 한다.

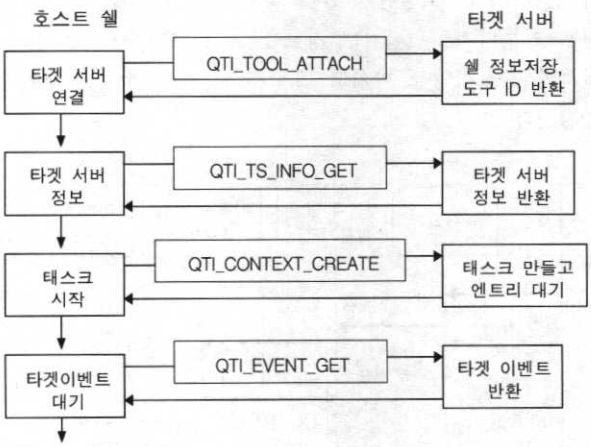


(그림 4) 대화형쉘 구조도

4.3 대화형 쉘과 타겟 서버의 연결

(그림 5)는 쉘과 타겟 서버간의 설정과 정보 교환 관계를 보여준다[7, 8]. 쉘은 QTI_TOOL_ATTACH 메시지를 이용

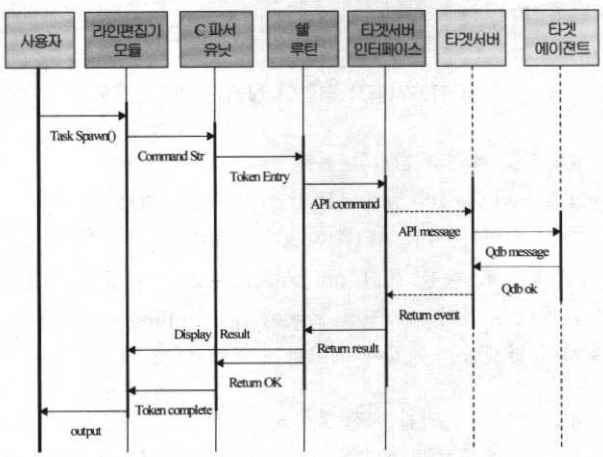
하여 서버와의 연결을 시도하고 정상적인 연결 상태가 되면 쉘 내부 모듈을 초기화하여 그 때부터 정보 교환이 가능하다. 원하는 모듈을 타겟 서버에 적재한 후 QTI_CONTEXT_CREATE 메시지를 통하여 태스크를 타겟에 설정하여 실행시키고 그에 따른 리턴 이벤트를 QTI_EVE NT_GET 메시지를 통하여 전달받아 사용자에게 태스크에 대한 정보를 출력한다.



(그림 5) 타겟서버와 쉘간의 연결

4.4 대화형 쉘의 주요 시나리오

4.4.1 태스크 관련 명령 처리 흐름도

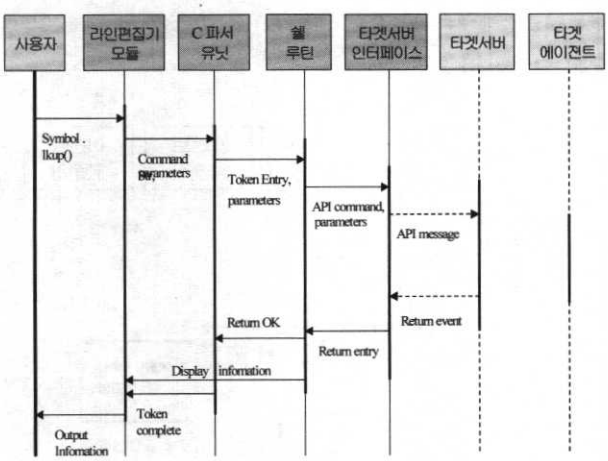


(그림 6) 태스크 관련 명령 처리 흐름도

(그림 6)은 태스크를 스폰(spawn)하는 시나리오의 흐름도로 나타낸다. 사용자로부터 명령이 들어오면 편집기 모듈은 입력받은 명령을 파서 유닛에게 전달하고 파서는 명령문을 파싱한 후 쉘 루틴 엔트리를 가지고 해당 함수를 실행시키면 함수는 통신 API들을 통해 타겟 서버로 필요한 메시지들을 전달한 후 그 결과를 전달받아 최종 결과문을 인터페이스를 통해 사용자에게 출력한다.

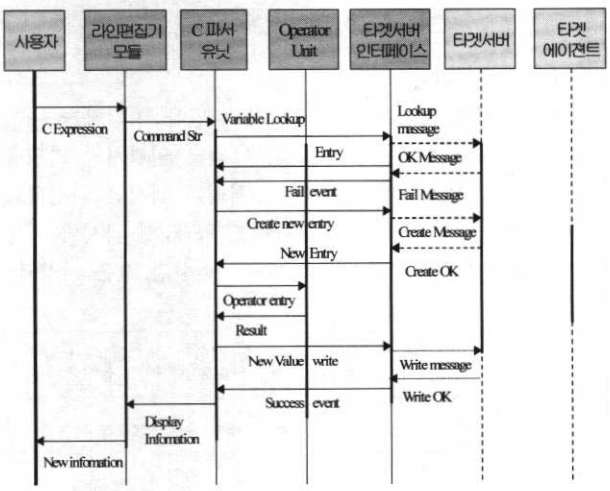
4.4.2 시스템 정보 관련 명령 흐름도

(그림 7)은 시스템 정보 중 심볼테이블을 참조하여 해당 심볼을 찾는 명령의 시나리오를 흐름도를 나타낸다. 사용자로부터 명령이 들어오면 편집기 모듈은 입력받은 명령을 파서에게 전달한다. 파서는 명령문을 파싱한 후 해당 함수를 실행시키면 함수는 통신 API들을 통해 타겟 서버와의 통신으로 심볼테이블의 엔트리를 전달받아 심볼에 대한 최종 결과문을 인터페이스를 통해 사용자에게 출력한다.



(그림 7) 시스템정보 관련 명령 처리 흐름도

4.4.3 C 수식 관련 명령 처리 흐름도



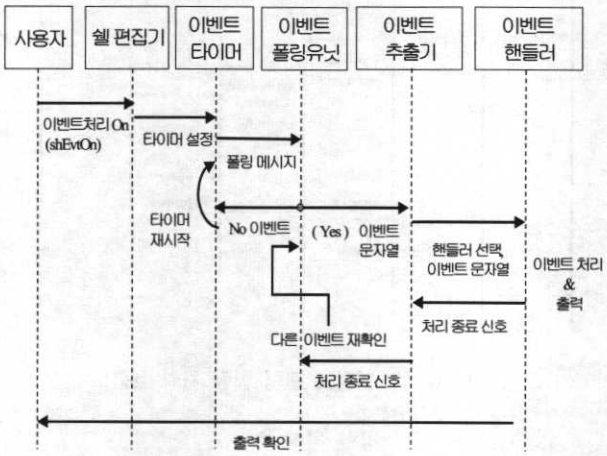
(그림 8) C수식 관련 명령 처리 흐름도

(그림 8)은 C 수식을 계산하는 시나리오를 흐름도로 나타낸 것으로 먼저 사용자로부터 C 수식이 입력되면 편집기 모듈은 입력받은 명령어를 파서에게 전달한다. 파서는 해당 명령 어을 파싱한 후 각 토큰별로 해당작업 (심볼에 대한 값 접근, 새로운 심볼의 등록, 심볼접근오류등)을 타겟 서버와의 통신 함수를 이용하여 수행하고 그 결과를 통해 C 수식을 계산한다. 계산이 끝난 후 파서는 그 결과를 입출력

인터페이스를 통해 사용자에게 출력한다.

4.4.4 이벤트 처리 흐름도

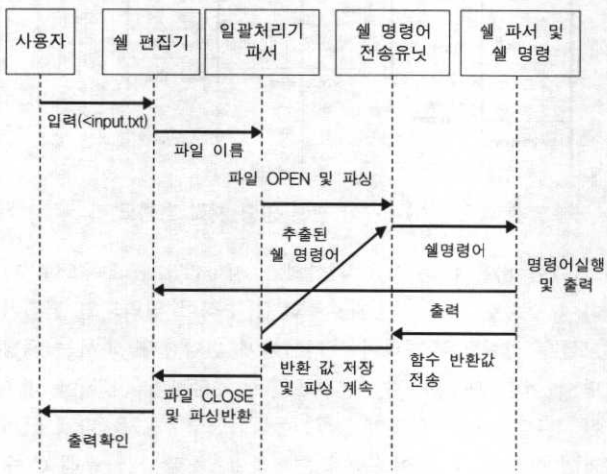
이벤트 관리기능은 사용자가 이벤트 폴링 옵션을 on 상태로 지정하였을 경우에 수행된다. 사용자가 이벤트를 on으로 하게되면 이벤트를 위한 타이머가 설정되고 정해진 주기마다 이벤트 폴링을 한다. 폴링 메시지에 따라 이벤트 추출기는 이벤트를 확인하고 이벤트가 존재하지 않으면 반환하고 이벤트가 존재하면 이를 구분하여 이벤트 처리기를 호출한다. 이벤트 처리는 이벤트가 없을 때까지 계속 수행되고 그렇지 않으면 반환한다.



(그림 9) 이벤트 처리의 흐름도

4.4.5 일괄처리기 흐름도

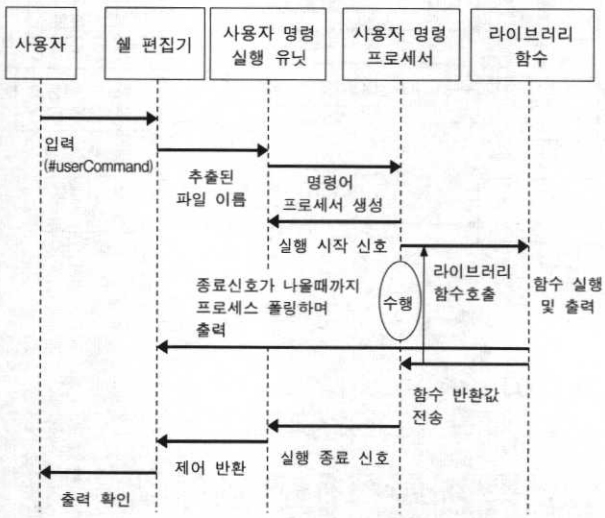
일괄처리기는 사용자의 일괄처리 파일을 처리하는 기능으로 먼저 해당 명령어가 입력되면 해당 일괄처리 파일을 열고 처리기 파서가 파싱을 한다. 처리기 파서는 파싱하는 동안 쉘 명령어를 만나면 전송단위를 통해 쉘 파서에 실행명령을 보내고 실행결과 값은 받는다. 반환된 값은 저장되어 다시 파싱하는데 사용하며 이를 반복한다. 파일처리가 끝나거나 에러가 발생하면 일괄처리기 파서는 파일을 닫고 제어를 반환한다.



(그림 10) 일괄처리 기능의 흐름도

4.4.6 사용자정의 명령어 실행 흐름도

사용자 명령어 실행은 프로세스 생성 방식으로 이루어지는데 사용자가 미리 정의하고 지정한 명령어 콘솔 파일을 정해진 디렉토리에 넣고 쉘에서 호출한다. 그러면 사용자명령 실행단위는 호출된 파일을 찾아 실행을 시도한다. 시작신호 이후의 쉘 편집기는 정보를 출력하고 종료신호가 도달하면 종료작업을 마치고 프로세스를 종료한 후 제어를 반환한다.



(그림 11) 사용자 명령어 실행기능의 흐름도

4.5 타겟 제어의 방법적 분류

4.5.1 타겟 서버 지원 API만을 이용하는 방법

타겟 서버가 지원 API를 이용하여 원하는 작업를 수행하는 방법으로 예를 들어 qtiContectCreate() 함수를 이용한 오브젝트 모듈의 태스크를 스폰과 qtiSymFind() 함수를 이용하여 타겟심볼을 찾아 제어하는 것이 이에 해당한다.

4.5.2 타겟 API를 직접 호출하는 방법

타겟에 존재하는 함수를 직접 실행시켜 결과를 얻는 방법으로 타겟의 함수를 직접 호출함으로써 원하는 작업을 수행하게 되며 이는 원격도구가 타겟의 작업이 끝날 때까지 대기 상태가 된다[16].

4.5.3 타겟 구조에 따른 오프셋(offset) 설정으로 메모리 액세스 제어 방법

타겟 구조에 따른 오프셋 설정으로 메모리를 직접 액세스하는 제어 방법이다. 본 연구에서는 세가지 방법이 전부 사용되었으나 그 중에서 타겟 자체의 부담을 가장 적게 해

〈표 1〉 쉘 루틴 기능표

명 령 어	기 능	명 령 어	기 능
sp	스폰 함수	lkup	해당 문자열 포함 심볼의 출력
sps	정지형 스폰	lkAddr	해당 주소지의 심볼 출력
tr	태스크 재시작	d	해당 주소지 메모리 내용 출력
td	태스크 제거	cd	작업 디렉토리의 출력 및 바꿈
period	주기적인 함수 호출	ls	해당 디렉토리의 내용 출력
repeat	함수 반복호출	pwd	현재의 작업 디렉토리
taskIdDefault	기본 태스크 ID 설정	h	히스토리 출력
i	태스크 정보 출력	ld	해당 모듈의 로딩(loading)
ti	태스크 전체 정보 출력	mld	멀티플(multiple) 로딩
checkStack	태스크 스택정보 출력	unld	해당 모듈 언로딩(unloading)
taskIdFigure	태스크 ID 출력	m	해당 주소지의 내용 및 바꿈
show	해당 오브젝트 출력	mRegs	레지스터정보출력
classShow	해당 클래스정보출력	b	브레이크포인터의 출력 및 설정
taskShow	태스크 정보 출력	s	해당 태스크 전진
mutexShow	뮤텍스 정보 출력	c	해당 태스크 계속
semShow	세마포 정보 출력	bdAll	모든 브레이크 포인터 제거
msgQShow	메시지큐 정보 출력	bd	해당 브레이크 포인터 제거
moduleShow	모듈정보 출력	shEvtOn	주기적인 이벤트폴링 시작명령.
moduleIdFigure	모듈 ID 확인	shEvtOff	진행중인 이벤트폴링 정지명령
directExe	직접호출함수	funcExe	함수호출명령 (sp와 같은 역할)
		exit, quit	타겟연결 종료후 쉘을 끝내기

주는 오프셋 설정방법이 많이 사용되었는데 이는 타겟 에이전트의 API 요구를 줄이고 RTOS에서 지원하는 API의 의존도를 줄여 조립형 실시간 OS의 개발에 부담을 주지 않고 개발 환경을 구축할 수 있는 장점이 있다. 또한 메모리 읽기/쓰기의 함수로 이루어진 Gopher형식을 사용하기 때문에 개발 단계에 타겟 시스템의 부담을 최소화하였으며 OS와 개발도구간의 상호유지보수 또한 쉽게 할 수 있다. 그렇게 하여 내장형 시스템에서 자원 요소를 최대한 사용할 수 있도록 하였다.

5. 원격 대화형 쉘의 구현

5.1 C 수식 파서

파서가 입력받은 수식을 처리하는 방법은 다음과 같이 네 가지 방법으로 나눈다.

① 입력받은 수식에 식별자 즉, 변수명이나 함수명이 포함되어 있지 않을 경우, 수식을 계산하여 결과를 출력한다.

② 수식에 변수가 포함되어 있을 경우, 해당 변수에 대한 참조 요청을 타겟 서버로 보내고 필요시 해당 변수를 타겟 서버의 심볼 테이블에 등록시키거나 해당 변수에 값을 수정하고 그 결과를 타겟 서버로부터 돌려 받아 수식을 계산하여 결과 값을 사용자에게 출력한다.

③ 수식에 포함되어 있는 함수가 쉘 내장형 명령인 경우, 해당 함수를 실행시키고 함수에 대한 결과 값을 반환받아 수식을 계산한다.

④ 수식에 포함되어 있는 함수가 쉘 내장형 명령이 아닐 경우, 해당 함수에 대한 접근 요청을 타겟 서버로 보내어 해당 함수의 엔트리 주소를 파악하고 다이렉트 함수호출을 이용하여 실행시킨 후 그 결과를 타겟 서버로부터 돌려 받아 수식을 계산한다.

5.2 쉘 내장형 루틴

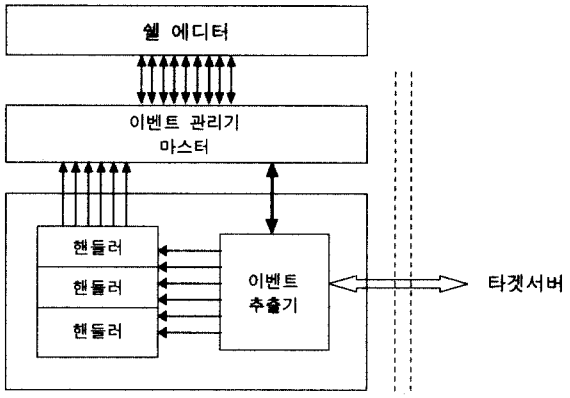
쉘 내부에 가지고 있는 것으로서 사용자는 <표 1>과 같은 루틴을 사용할 수 있다.

5.3 이벤트 관리기

5.3.1 이벤트 관리기의 구조

이벤트 관리기 모듈은 지속적인 타겟 감시 기능의 한 종류로서 타겟 이벤트 내용을 받아 필요 부분을 출력하는 기능을 가진다. 이 모듈은 필요 이벤트에 해당하는 이벤트 처리기와 주기적으로 이벤트 폴링을 하는 단위와 해당 사항을 출력하는 부분으로 구성된다. 이벤트 관리기는 기본으로 off 상태를 유지하고 있으며 사용자가 이벤트 디버깅을 원할 때 on 혹은 off가 가능하도록 명령어와 단축키(CTRL +

F12)를 제공한다.



(그림 12) 이벤트 관리기의 구조

이벤트관리기는 (그림 12)의 구조도와 같이 3가지 주요 단위로 구성되어 있다.

- 이벤트 관리기 마스터 단위 : 주기적 이벤트 폴링을 주관하고 이벤트 출력을 전담한다.
- 이벤트 추출 단위 : 이벤트 폴링 주기마다 서버로부터 이벤트를 추출하는 부분이다.
- 이벤트 처리기 : 이벤트의 종류에 따라 처리하고 적절한 정보를 만든다.

5.3.2 이벤트 처리기

이벤트는 처리기는 타겟에 이벤트가 발생한 것을 이벤트 관리기가 인식하였을 때 해당 이벤트의 종류를 따라 이벤트를 관리하는 루틴으로 이벤트 추출기와 인터페이스를 가지고 있다. 해당 처리기는 이벤트 정보로부터 필요한 정보를 추출하여 사용자에게 출력한다.

5.3.3 주기적 이벤트 폴링 기법

이벤트는 타이머를 이용하여 주기적으로 폴링 상태가 되도록 구현되었으며[19] 타이머를 설정하거나 재설정함으로써 이벤트 관리기를 on/off 할 수 있도록 구현되었다.

5.4 일괄처리기

일괄처리기는 라인명령어 방식의 대화형 쉘과 연동하여 사용될 수 있도록 설계되었으며 일련의 라인명령어와 제어문으로 구성된 일괄처리 파일을 실행할 수 있게 하는 기능을 제공한다. 일괄 처리기는 일괄처리 파일을 해석하는 파서 부분과 해석된 부분의 실행 단위들로 구성되어 있다. 실행 단위에는 대화형 쉘 명령어의 실행부분과 그에 대한 반환값을 받는 부분이 포함되어 있다. 또한 일괄처리 파일을 간단한 제어구문이나 for문과 같은 반복구문이 포함하여 구성할 수 있으며 반환결과에 따라 실행을 변화 할 수 있다.

일괄처리 파서는 스크립트방식으로 파일을 직접 읽어가며 처리를 한다. 쉘 명령문 한 라인을 읽고 이를 전송하여 실행을 하고 그 다음 문장을 실행한다. While 문이나 if 문에 대해서는 스택을 사용하여 그 실행을 저장하는 방식을 사용하였으며 조건문은 조건식에 대한 플래그(flag)를 가지고 실행과 스캔을 결정할 수 있도록 하였다. 그리고 while 문의 경우 파일 포인터를 사용하여 다시 내부함수를 수행할 수 있도록 하였다[14].

5.5 사용자정의 명령어 기능

대화형 쉘에서 사용자에게 편리함을 제공하기 위해서 사용자가 정의한 함수를 만들어서 이를 명령어 혹은 루틴의 일부분으로 사용이 가능하도록 한 것이 사용자 정의 명령어 기능이다. 이를 위해 타겟과의 연결과 출력전환을 위한 신호 발생 함수, 출력함수 등을 라이브러리함수로 제공하여 프로그램이 가능하도록 하였다. 라이브러리 함수를 사용하기 위한 헤더파일은 "UtilApi.h"에 정의되어있다. 다음은 사용자 명령어를 위한 라이브러리 함수 이름이다.

- HQTI hQti : 서버API 핸들을 제공한다.
- util_startInit(toolName) : 타겟연결 초기화 함수
- void util_beginSignal() : 시작신호를 발생하는 함수.
- void util_finishSignal() : 끝신호를 발생하는 함수.
- void util_inputSignal() : 입력대기신호 발생함수.
- void util_printOut(char * outstr) : 문자열출력함수
- int util_endExit() : 핸들 릴리즈 및 마무리 함수.

6. 결론 및 향후 연구

최근의 정보가전기기의 급속한 발전은 여러 가지 가전기기에 쉽게 이용이 가능한 경쟁력 있는 OS의 개발이 필수적이고 이에 따라 이식성 있는 대화형 쉘 또한 경쟁력 향상을 위한 빠른 개발 주기와 쉬운 프로그램방식의 이용에 필수이다[21]. 특히 내장형 시스템 프로그래밍의 경우 경쟁력 제고를 위해 신제품에 신속하게 기능을 추가하고 확장할 수 있는 조립형 실시간 OS의 개발과 도구들의 통합이 필수적이며 통합된 제품을 각 응용분야에 제공함으로써 경쟁력 강화에 필요하다. 정보기기의 응용 분야가 디지털 TV, PDA, 인터넷폰, 사무자동화기기, 디지털 비디오나 오디오등과 같은 가전기기의 종류가 방대해짐에 따라 앞으로 실시간 OS의 사용은 계속적으로 높아질 전망이고 각 응용분야별 실시간 OS와 함께 대화형 쉘 또한 필수적인 요소로 될 것이다.

본 연구는 내장형 시스템의 RTOS 프로그래밍 도구들 중 프로그래머가 타겟 시스템의 정보를 원하는 방향으로

조작, 참조, 디버깅을 지원하는 대화형 쉘 도구에 관한 것으로서 ETRI에서 개발된 RTOS Q+와 함께 개발되었고 통합개발환경의 한 도구로서 개발, 사용되고 있다.

대화형 쉘은 개발 도구 통합 시스템에서 다른 도구들과 함께 원격지에서 프로그래밍 할 수 있는 환경을 제공하는 것으로서 호스트에서 원격으로 시리얼 또는 이더넷으로 연결된 타겟 보드위에서 시스템 제어 명령을 수행하는 대화형으로 수행한다. 프로그래머가 호스트에서 개발한 모듈을 타겟에 적재하고 이를 수행시키는 기능과 원하는 모듈을 언로딩하고 타겟 보드상의 쉘 명령어를 호스트 상에서 수행시키는 기능들을 포함하고 있다. 이러한 기능들은 타겟 시스템의 부가적인 오버헤드를 최소화하고 호스트의 매개체인 타겟 서버의 부담을 최소화하면서 프로그래머가 원하는 정보에 관한 모든 기능을 제공 할 수 있도록 코드의 최적화와 메모리에 대한 부담을 줄일 수 있도록 하였다.

또한 정보의 변환과 저장을 용이하게 할 수 있도록 C 수식해석기를 포함시켜 최대한 정보조작을 용이하게 만드는 연구와 함께 GUI 환경에 윈도우 에디팅 기능을 포함하여 사용자의 적응력과 편의성을 최대한 제공할 수 있게 하였다.

향후과제로서는 현재 구현된 대화형 쉘의 기능과 이벤트 관리기능을 강화하고 응용프로그램 제작자의 요구분석을 통한 기능추가 및 루틴 수정이 지속적으로 진행되어야 하며 UNIX 환경의 GCC를 활용하는 bash 환경의 쉘 내 통합 등의 기능적인 부분 또한 추가개발 되어야 할 것이다. 쉘의 이식성에 있어서 또한 실시간 OS 의존적인 부분의 해결 방안을 모색하여 더 나은 활용성을 보장하는 연구가 필요하다.

본 연구가 추구하는 사용자 편의적인 대화형 쉘 개발은 앞으로 모든 응용 분야에서 내장형 RTOS 응용프로그램 개발시스템의 설계, 대화형식 개발 및 테스팅에 필수적인 부분으로 자리 잡을 것으로 예상된다.

참 고 문 헌

[1] 한국전자통신연구원, "조립형 실시간 OS 사용자요구사항정의서 1.0", 1998.

[2] 한국전자통신연구원, "실시간 OS 커널 상세 설계서 1.0", 1999.

[3] WindRiver, "VxWorks 5.3.1 Programmer's Guide", 1997.

[4] TRON ASSOCIATION, ITRON Specification Ver. 2.02.01.00, Ken Sakamura, 1998.

[5] Precise Software Technologies, "New RTOS Capabilities that Support Development of High-End Embedded Applications," Embedded Systems Conference Chicago, 2000.

[6] Eonic System, "Technical White Paper," version 1.0, 1999.

[7] http://www.wrs.com/windword/html/writing-1.html, 1997.

[8] http://www.wrs.com/windword/html/writing-2.html, 1997.

[9] http://www.dasan.co.kr/papers/whitepapers/FutureOfESD/index.html.

[10] 박상서, "Unix 커널 디버거", 정보과학회지, 제12권 제10호, 1994.

[11] Jonathan Rosenberg, "How Debuggers Work," John Wiley & Sons, 1996.

[12] WindRiver, "Tornado API Guide 1.01," 1997.

[13] WindRiver, "VxWorks 5.3.1 Reference manual," 1997.

[14] Y. Jenny Luo, "PCYACC OBJECT ORIENTED TOOLKIT," ABRAXAS software, 1995.

[15] WindRiver, "VxWorks Training Workshop," 1996.

[16] WindRiver, "Tornado 1.0 User's Guide (Windows Version)," 1997.

[17] O'Reilly "lex & yacc," John R. Levine & Doug Brown, 1995.

[18] Dennis M.Ritche, "PROGRAMING LANGUAGE (ANSI C)," AT&T Bell Lab., 1990.

[19] W. Richard Steven, "UNIX Network Programming," Prentice-Hall, 1999.

[20] Charles Petzold, "Programming WINDOWS," MicroSoft, 1999.

[21] Endrew J.Koneeki, Zanusz Zalewski, Daniel Eyassu, "Leaning Real-Time Programming Concept through VxWorks lab experience," 1999.

[22] 김대희, 남영광, 김홍남, 이광용, "RTOS용 원격대화형쉘 개발에 관한 연구", 정보처리학회 추계학술발표논문집, 2000.

김 대 희

e-mail : davidong@lge.com

1999년 연세대학교 문리대학 전산학과 (학사)

2001년 연세대학교 전산학과(석사)

2001년~현재 LG 차세대단말연구소

관심분야 : 실시간운영체제, 소프트웨어공학

남 영 광

e-mail : yknam@dragon.yonsei.ac.kr

1982년 연세대학교 수학과 졸업(학사)

1985년 한국과학기술원 전산학과 졸업 (석사)

1992년 노스웨스턴대학교 전산학과 졸업 (박사)

1992년~1994년 시스템공학연구소 선임연구원

1995년~현재 연세대학교 전산학과 부교수

관심분야 : 소프트웨어공학, 프로그래밍언어, 정보검색

김 흥 남

e-mail : hnkim@etri.re.kr

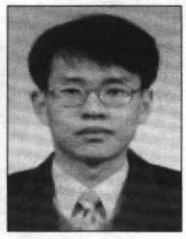
1980년 서울대학교 전자공학과 학사

1989년 미국 Ball State University 전산학 석사

1996년 미국 Pennsylvania State University 전산학 박사

1983년~현재 ETRI 컴퓨터 소프트웨어기술연구소 인터넷정보가전연구부 책임연구원(내장형 S/W 연구팀장)

관심분야 : 실시간운영체제, 비디오압축알고리즘, 분산멀티미디어시스템



이 광 용

e-mail : kylee@etri.re.kr

1991년 숭실대학교 전자계산학과(학사)

1993년 숭실대학교대학원 전자계산학과 (공학석사)

1997년 숭실대학교대학원 전자계산학과 (공학박사)

1996년~1998년 숭실대학교 생산기술연구소 연구원

1997년~1998년 ETRI 컴퓨터 소프트웨어기술연구소 소프트웨어공학연구부 박사후연수연구원(Post-Doc.)

1999년~현재 ETRI 컴퓨터 소프트웨어기술연구소 인터넷정보가전연구부 선임연구원

관심분야 : 소프트웨어공학, 실시간운영체제, 인터넷정보가전, 내장형S/W, 객체지향 모델링/시뮬레이션/디버깅 도구, 분산멀티미디어시스템, 정형기법