2000년 한국정보처리학회 추계 학술발표논문집 제 7 권 제 2 호

RTOS 용 원격 대화형쉘 개발에 관한 연구

김대희*, 남영광*, 이광용**, 김흥남**
*연세대학교 전산학과
**한국전자통신연구원
e-mail : davidkim@magics.yonsei.ac.kr
yknam@dragon.yonsei.ac.kr

Development of a Remote Interactive Shell for RTOS

Dae-Hee Kim*, Young-Kwang Nam*, Kwang-Yong Lee**, Heung-Nam Kim**
*Dept. of Computer Science, Yonsei University
**Electronics and Telecommunication Research Institute

요 약

최근의 삽입 시스템의 실시간 운영체제(RTOS)에서는 메모리와 CPU 파워 등의 제한적인 환경에서 원하는 기능을 최적으로 최단시간에 구현할 수 있는 개방형 개발도구 환경이 거의 필수적이다. 개방형 개발환경은 타겟시스템의 부담을 최소화하면서 원하는 정보를 대화식으로 빠르게 억세스하여 참조, 제어 할 수 있는 원격 대화형쉘이 필수적이다. 본 논문에서는 원격지에서 모듈별 로딩, 태스크의 스폰과 더불어 테스크 상태 등을 확인할 수 있는 원격 대화형쉘 프로그램의 프로토타입과 그에대한 구현방법을 기술하고 원격지에서 타겟의 부담을 최소화하는 방향의 정보참조방법을 비교분석하여 실시간 OS 와 더불어 개발환경의 빠른 상호유지보수를 가능하게 하였다.

1. 서론

근래의 삽입 시스템에서는 빠른 시간내에 양질의 응용프로그램의 개발과 짧은 개발주기에 대한 고객과 제품회사의 요구가 대두되고 있다. 이에 대한 핵심은 개발자와 시스템사이의 인터페이스 즉 개발 환경이 핵심이다. 과거의 텍스트 베이스 도구들은 대화형이 아니며 타겟에 존재하고 테스트 기간이 길어 많은 문제점을 유발하였다. 따라서 최근에는 원격지의 개발 호스트상에 존재하는 여러가지 GUI 대화형 도구들이 개발되어오고 있는데 개발시간상의 문제점과 응용프로그램의 이식에 대한 문제점을 해결하기 위해서는 컴파일러와 디버거 도구들과 더불어 원격 대화형쉘이 필수적으로 추가되어야 한다. 대화형 쉘은 C 인터프리터를 통하여 실행시에 원격지(호스트)에서 타겟을 대화식으로 억세스할 수 있는 방법을 제공하는데 그 목표를 가지고 있다. 쉘은 호스트의 타겟서버와 함께 호스트에서 작동하면서 사용자의 응용 프로그래밍에 대한 타겟보드의 부담을 줄이면서 시스템과 태스크정보의 제어, 참조를 빠르게 하기 위해 사용되게된다.

본 논문에서는 응용프로그램 제작시에 원격지에서 모듈별 로딩, 태스크의 스폰과 더불어 테스크 상태 등을 확인할 수 있는 원격 대화형쉘 프로그램의 프로토타입에 대하여 기술하고 원격지에서 타겟의 부담을 최소화하는 방향의 정보참조방법을 비교함으로써 실시간 OS 와 더불어 개발환경의 빠른 상호유지보수를 가능하게 할 수 있는 방법을 제시한다.

2. 개방형 통합개발도구 시스템에서의 대화형쉘

대화형 쉘은 사용자가 직접적으로 타겟 시스템을 제어할 수 있도록 라인 커맨드 방식으로 운영된다. 이것은 사용자의 커맨드를 받아 호스트 상에서 실행 시키고 타겟 서버에게 심볼 테이블 혹은 프로그램이나 타겟에 상주하고 있는 데이터에 관한 요구사항을 보내어 그 결과를 사용자에게 출력하는 방식으로 이루어져 있다. 통합개발도구 내에서의 대화형 쉘의 조직은 그림 1 과 같다.

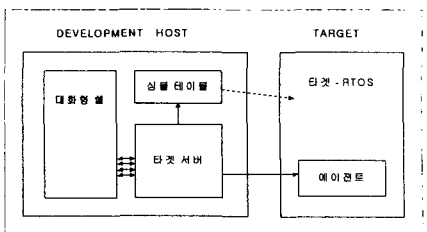

그림 1. 개발도구에서의 대화형 쉘

3. 원격 대화형쉘의 구조

대화형 쉘은 개방형 도구들 중에 하나로 속해 있으면서 개방형 개발 도구들과 함께 타겟서버와 연결되어 있으며 타겟서버와는 서버간 프로토콜을 기반한 API 들의 모듈을 통해 서버와 통신을 할 수 있는 구조를 가진다. 쉘 내부에는 사용자와의 인터페이스를 만들어주는 GUI 라인 에디터 모듈과 C 인터프리터가 인터페이스를 가지고 입력을 처리 하게 되고 C 인터프리터 에는 입력을 파싱하는 파서를 통해 각각의 처리 유닛으로 연결되어 입력을 처리하고 결과를 디스플레이 하게 된다. 이벤트 관리기는 주기적인 이벤트 참조 루틴을 발생시켜 타겟을 감시하고 이에 대한 정보를 출력하게 되며 일괄처리기는 에디터에서 관리하며 쉘 루틴과 연동수행할 수 있도록 파서와 연결된 함수를 제공한다.

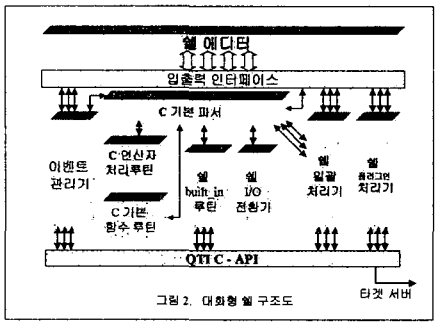

그림 2. 대화형 쉘 구조도

4. 원격 대화형쉘의 기능

4.1 C 언어표현해석기 모듈

C 언어 표현 해석기 모듈은 프롬프트 상에서 명령어를 입력 받아 수행하는데 C 파서는 C 인터프리터 내부에 존재하는 유닛으로 토큰에 대한 파싱과 변수참조에 대한 기능을 수행한다. 개발 도구가 동작하고 있는 호스트상의 프롬프트에서 개발자가 입력한 명령어 라인을 읽어서, 그것이 내장형 쉘 명령어인지, 타겟의 시스템 함수인지, 응용 프로그램내의 함수인지를 판단하고 알맞은 유닛의 루틴을 호출함으로써 결과를 출력한다. 이때, 새로운 식별자가 나타났을 때는 타겟의 심볼 테이블을 참조하여 호출하거나 새로운 심볼을 등록하고 그 주소를 가지고 관련 동작을 수행한다. 모듈의 주요 유닛들은 다음과 같은 것들이 있다.

- C 파서 유닛 – C 수식 파싱기능
- C 오퍼레이터 유닛 – C 연산자(+,-,*,/,등) 수행기능
- 내장형 쉘 루틴 유닛 – 타겟 쉘과 대응하여 호스트상에서 타겟을 직접 제어하는 기능제공

(1) 타스크 관리를 위한 루틴
- sp () 스폰 함수
- sps() 서스펜드형 스폰
- tr(char *taskNameOrId) 타스크 재시작
- ts(char *taskNameOrId) 타스크 멈춤
- td(char *taskNameOrId) 타스크 제거
- period() 주기적인 함수 호출
- repeat() 지정한 수만큼 함수 반복호출
- taskIdDefault() 기본 타스크 아이디 설정

(2) 타스크 정보 디스플레이 루틴
- i([char *taskNameOrId]) 타스크 정보 출력
- ti() 타스크 전체 정보 출력
- checkStack() 타스크 스택정보 출력
- taskIdFigure() 타스크 아이디 출력

(3) 시스템 정보 디스플레이 루틴
- lkup() 해당 스트링 포함 심볼의 출력
- lkAddr(int addr) 해당 주소지의 심볼 출력
- d() 해당 주소지의 메모리 내용 출력
- cd() 워킹 디렉토리의 표현 및 바꿈
- ls() 해당 디렉토리의 내용 출력
- pwd(void) 현재의 워킹 디렉토리
- h() 히스토리 출력 및 히스토리 개수 세팅

(4) 시스템 디버깅과 변환 루틴
- ld(char * moduleName) 해당 모듈의 로딩
- mld() 멀티플 로딩
- unld(char * moduleName) 해당 모듈 언로딩
- m(int addr) 해당 주소지의 내용 및 바꿈
- mRegs(int taskNameOrId) 레지스터정보출력
- b(int addr) 브레이크포인터의 출력 및 세팅
- s(char* taskNameOrId) 해당 태스크 스텝핑
- c(char* taskNameOrId) 해당 태스크 계속
- bdall() 모든 브레이크 포인트를 제거한다.
- bd() 해당 주소지의 브레이크 포인터 제거

(5) 오브젝트 디스플레이를 위한 루틴
- show(int objectAddr) 해당 오브젝트 출력
- classShow(int classId) 해당 클래스정보출력
- taskShow() 타스크 정보 출력
- mutexShow(int mutexid) 뮤텍스 정보 출력
- semShow(int semid) 세마포 정보 출력
- msgQShow(int msgQid) 메시지큐 정보 출력
- moduleShow(void) 모듈정보 출력
- moduleIdFigure(char * name) 모듈 ID 확인

(6) 기타 대화형 쉘 루틴
- shEvtOn 주기적인 이벤트폴링 시작명령.
- shEvtOff 진행중인 이벤트폴링 정지명령.
- funcExe() 펑션콜함수 (sp 와 같은 역할)
- directExe()직접호출함수
- exit(void) 타겟과의 연결을 종료하고 쉘을 끝내는 명령

4.2 이벤트관리기 모듈

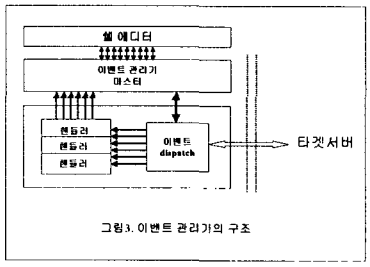

그림 3. 이벤트 관리기의 구조

이벤트 관리기 모듈은 지속적인 타겟 감시기능의 한 종류로서 타겟이벤트 내용을 받아 필요 부분을 출력하도록 되어 있다. 이 모듈은 이벤트 핸들러와 반복 폴링수행유닛과 해당 사항을 출력유닛으로 구성되어 있다. 핸들러들은 사용자가 필요한 데이터 무엇인지 확인하고 구현되었다. 이벤트 관리기는 현재 디폴트로 Off 상태를 유지하고 있으며 사용자의 목적에 의해서 On 혹은 Off 가 가능하도록 명령어와 단축기를 제공하여 유연성을 주도록 하였다.

4.3 일괄처리기 모듈

일괄처리기는 라인 커맨드 방식의 대화형 쉘과 연동하여 사용될 수 있도록 설계 되었으며 계속되는 라인 커맨드의 일괄처리를 파일로부터 입력 받아 실행할 수 있게 하는 기능을 제공한다. 일괄 처리기는 파일 스크립트를 해석하는 파서 부분과 해석된 부분의 실행유닛들로 이루어 지게 되는데 대화형 쉘의 커맨드 실행유닛과 리턴값을 받는 부분이 이에 해당한다. 또한 간단한 컨트롤 구문을 추가하여 결과에 따라 실행을 변화 할 수 있도록 하였다.

일괄처리 파서는 스크립트방식으로 source화일을 직접 읽어가며 처리를 한다. 그러므로서, shell 명령문 한 라인을 읽고 이를 전송하여 실행을 하고 그 다음 문장을 실행한다. 만약 if문장이나 while 문장을 만나면 이들을 각각 스택에 저장하고 이들의 실행이 모두 끝나면 스택에서 빼낸다. if문장의 경우 조건식이 ture 이면 flag를 set하여 이를 바탕으로 then 부분이 실행이 되고 , else 가 있다면, else부분은 그냥 scan만을 하고 어떤 명령어도 실행하지 않는다. 반명에 조건식이 flase이면 이와 반대로 한다. while 문장의 경우, 현재 while문장의 맨 처음 위치를 마크하고 while문의 조건식을 검색을 한 후 조건식이 true일 경우 while문장의 내부를 실행을 하고 false 일 경우 while문장의 내부를 단지 scan만을 할 뿐 명령문들을 실행하지 않는다. while문장의 맨 끝에 왔을 경우, 조건식이 true 이면 마크해 두었던 파일의 포인터 위치인 while문장의 맨 처음 위치로 다시 이동해서 while문을 실행을 한다. 이 같은 방식으로 조건식이 false 가 될 때 까지 반복함으로써 while문장에 해당하는 반복이 가능하게 되어진다.

4.4 사용자명령어 실행 모듈

대화형 쉘을 사용함에 있어서 더 나은 유연성을 가능하게 하기 위하여 구현된 부분으로 사용자가 직접 임의의 명령을 제공되는 API 함수를 이용하여 구현할 수 있게 하고 이러한 부분을 대화형 쉘에서 직접 프로세스 실행 방식의 리다이렉션 기법을 이용하여 출력을 쉘 에디터에 할 수 있게 하였다. 여기에 타겟과의 연결과 리다이렉션을 위한 시그널 발생 함수, 출력함수 등을 라이브러리함수로 제공하여 프로그램 할 수 있게 하였다.

● 사용자 명령어 사용을 위한 라이브러리 함수.
- HQTI hQti : 서버 API 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): 스트링출력함수
- int util_endExit(): 핸들 릴리즈 및 마무리 함수.

5. 타겟제어의 방법적 분류 및 비교

(1) 타겟서버가 지원하는 API 를 이용하여 원하는 작업를 수행하는 방법으로 예를들어 qtiContectCreate() 함수를 이용한 오브젝트모듈의 태스크를 스폰과 qtiSymFind() 함수를 이용하여 타겟심볼을 찾아 제어하는 것이 이에 해당한다.

(2) 타겟 API 를 직접 실행하는 방법으로 타겟의 함수를 직접 호출함으로써 원하는 작업을 수행하게 되며 이는 원격도구가 타겟의 작업이 끝날 때 까지 대기상태가 된다.

(3) 타겟 구조에 따른 offset 설정으로 메모리를 직접 억세스하는 제어방법으로 해당 RTOS 의 구조의 이해가 요구된다.

이번 연구에서는 세가지 방법이 전부 사용되었으나 이중 타겟자체의 부담을 가장 적게 해주는 offset 설정 방법이 많이 사용되었는데 이는 타겟 에이젼트의 API 요구를 줄임은 물론이고 RTOS 에서 지원하는 API 의 의존도를 줄여 조립형 실시간 OS 의 개발에 부담을 주지 않고 개발 환경을 구축하는데 장점을 가지고 있다. 그리고 메모리 읽기/쓰기의 함수로 이루어진 Gopher 형식을 사용하기 때문에 개발단계에 타겟시스템의 부담을 최소화 할 수 있으며 OS 와 개발도구간의 상호유지보수 또한 용이하게 할 수 있다. 삽입시스템에서 한계가 있는 자원요소를 최대한 사용할 수 있을 것이다.

6. 주요 기능의 흐름도

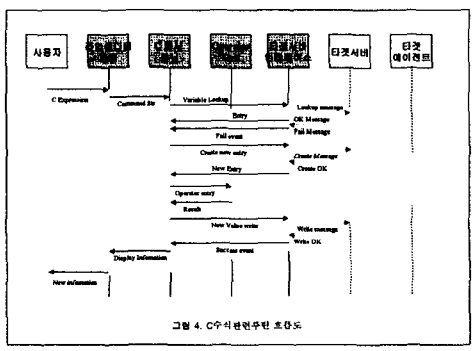

그림 4. C수식관련부분 흐름도

(그림 4)는 C 수식을 계산하는 시나리오를 다이어그램으로 나타낸 것으로 먼저 사용자로부터 C 수식이 입력되면 에디터 모듈은 입력 받은 명령 라인을 파서 유닛에게 전달하게 되고 파서는 해당 라인을 파싱한 후 각 토큰별로 해당하는 작업(심볼에 대한 값 억세스, 새로운 심볼의 등록, 심볼 억세스 에러등)을 타겟서버와 연결하는 통신 함수를 이용하여 수행한후 그 결과를 통해 C 수식을 계산하게 된다. 계산이 끝난 후 파서는 그 결과를 입출력 인터페이스를 통해 전달하고 에디터가 사용자에게 그 결과를 보여 주게 된다.

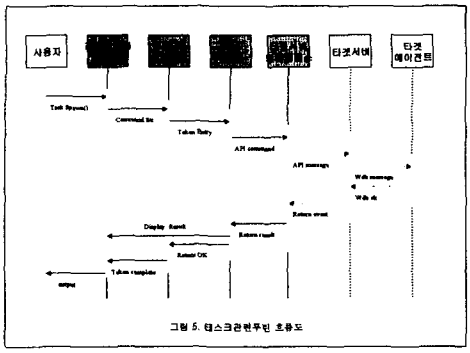

그림 5. 태스크관련루틴 흐름도

(그림 5)는 태스크를 스폰하는 시나리오를 다이어그램으로 나타낸 것으로 먼저 사용자로부터 명령이 들어 오면 에디터 모듈은 입력 받은 명령을 파서 유닛에게 전달하게 되고 파서는 명령문을 파싱한 후 쉘 루틴 엔트리를 가지고 해당 함수를 실행시키면 함수는 통신 API 들을 통해 타겟서버로 필요한 메시지들을 전달 한 후 그 결과를 리턴받아 최종 결과문을 인터페이스를 통해 전달하고 에디터에 출력한다

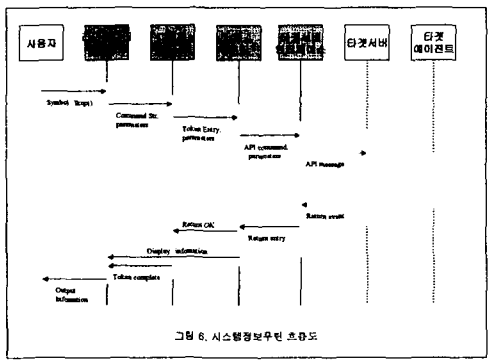

그림 6. 시스템정보루틴 흐름도

(그림 6)은 시스템 정보 중 타겟의 심볼 테이블을 참조하여 해당 심볼을 찾는 루틴의 시나리오를 나타낸 것으로 먼저 사용자로부터 해당 명령이 들어 오면 에디터 모듈은 입력 받은 명령을 파서 유닛에게 전달하게 되고 파서는 명령문을 파싱한 후 해당 함수를 실행시키면 함수는 통신 API 들을 통해 타겟서버로 필요한 메시지들을 전달 한 후 심볼테이블의 엔트리를 리턴받아 심볼에 대한 최종 결과문을 인터페이스를 통해 전달하면 에디터가 그 결과를 출력한다.

7. 결론 및 향후계획

대화형 쉘 개발 연구는 개발 도구 통합 시스템에서 다른 도구들과 함께 원격지에서 프로그래밍 할 수 있는 환경을 제공 하는데 있어서 호스트에서 원격으로 시리얼 또는 이더넷으로 연결된 타겟 보드위에서 시스템 제어 명령을 수행하는 인터렉티브 환경을 제공하는 것을 목적으로 한다. 프로그래머가 호스트에서 개발한 모듈을 타겟상에 로드하고 이를 수행시키는 기능이 필요하며 또한 필요 시 원하는 모듈을 언로딩하고 타겟보드상의 쉘 명령어를 호스트 상에서 수행시키는 기능들을 포함하고 있다. 따라서 이러한 기능들을 타겟의 부가적인 오버헤드를 최소화 하고 호스트의 매개체인 타겟서버의 부담을 최소화 하면서 프로그래머가 원하는 정보에 관한 모든 기능을 제공 할 수 있도록 코드의 최적화와 메모리에 대한 부담을 줄이는 연구를 포함하고 있다. 또한 정보의 변환과 저장을 용이하게 할 수 있도록 C 수식해석기를 포함 시켜 최대한 정보조작을 용이하게 만드는 연구와 함께 GUI 환경에 편리한 에디팅커맨드 방식을 도입하여 사용자의 적응력과 편의성을 최대한 제공할 수 있게 한다.

현재는 타겟서버와 함께 원격대화형쉘이 호스트상에서 구동됨에도 불구하고 타겟OS의 구조에 의존하는 오프셋설정을 기초로 하기 때문에 완전한 이식성의 구현이 되지 못하고 있다. 이에 여러 RTOS의 필수구조를 분석하고 적절한 오프셋 설정 방법이나 중요 부분의 타겟서버 지원형태로 발전시켜 타겟의 부담이 추가로 발생하지 않는 범위에서 더 나은 중간단계 구현법을 연구개발함으로써 완전한 이식성의 방향으로 나아가야 할 것이며 사용자 요구분석을 토대로 더 효율적인 정보출력 및 제어방법을 제공하는데 또한 계속적인 연구 노력이 계속되어야 할 것이다.

본 연구가 추구하는 사용자 편의적인 대화형 쉘 개발은 앞으로 모든 응용 분야에서 내장형 RTOS 시스템의 프로토타입핑, 인터렉티브 개발 및 테스팅에 필수적인 부분으로 자리 잡을 것이다.

참고문헌

[1] ETRI, "조립형 RTOS 사용자요구사항정의서 1.0"
[2] http://www.wrs.com/windword/html/writing-1.html, 1997
[3] http://www.wrs.com/windword/html/writing-2.html, 1997
[4] http://www.dasan.co.kr/papers/whitepapers/index.html.
[5] 박상서,"Unix 커널디버거", 정보과학회지 제 12 권 10 호, 1994. 11.
[6] Jonathan Rosenberg, "How Debuggers Work", John Wiley & Sons, 1996.
[7] WindRiver, "Tornado API Guide 1.01", 1997. 3
[8] WindRiver, "VxWorks 5.3.1 Reference manual", 1997. 4.
[9] ABRAXAS software "PCYACC OBJECT ORIENTED TOOLKIT", Y.Jenny Luo, 1995.10
[10] WindRiver, "VxWorks Training Workshop", 1996. 3.
[11] WindRiver, "VxWorks 5.3.1 Programmer's Guide", 1997. 4.
[12] WindRiver, "Tornado 1.0 User's Guide (Windows Version) ", 1997. 4.
[13] O'Reilly "lex & yacc", John R. Levine & Doug Brown, 1995.2
[14] AT&T Bell Lab. , "PROGRAMING LANGUAGE (ANSI C)", Dennis M.Ritche 1990
[15] Prentice-Hall International Inc. , "UNIX Network Programming", W.Richard Steven 1999
[16] MicroSoft , "Programming WINDOWS" , Charles Petzold 1999
[17] "Leaning Real-Time Programming Concept through VxWorks lab experience" , Endrew J.Koneeki, Zanusz Zalewski, Daniel Eyassu, 1999
[18] E. Sorton, A. Korneeki "Hands-on Software Design" , IEEE Potentilas, vol 17(2), 1996.5 , pp 42-44