韓國情報敎育學會論文誌
第3卷 第1號 pp.143~152

온라인 실습환경 구축을 위한 클라이언트-서버 모델

이 수 현

suhyun@sarim.changwon.ac.kr

창원대학교 컴퓨터공학과 프로그래밍언어연구실[1]

A Client-Server Model for Online Practicing Environment

Su-Hyun Lee

Department of Computer Science, Changwon National University

요 약

컴퓨터 관련 교육에 있어서 실험과 실습은 교육 내용의 효과적인 전달과 교육의 목적 달성을 위하여 대단히 중요한 부분이다. 현재의 환경에서 소프트웨어의 실험과 실습을 위해서는 실습하고자 하는 프로그램을 실습자가 사용하는 컴퓨터에 모두 설치하여야 한다. 실습 프로그램을 개별적으로 갖추는 것은 설치에 대한 부담, 설치에 의한 자원 낭비, 업그레이드의 필요성 등으로 인해서 실습 자체와는 직접적인 관련이 없는 부분에 대한 부담이 커진다.

본 논문에서는 WWW 환경에서 컴퓨터 관련 교과를 실습할 수 있는 환경을 구축하기 위한 기본적인 모델을 제시하는데 그 목적이 있다. 제안된 모델은 클라이언트-서버 구조를 기반으로 하여 구축되었으며, 실습자는 웹 브라우저를 실행함으로써 실습에 대한 준비가 끝나며, 프로그램의 실행이나 처리는 서버에서 모두 이루어진다. 또한 WWW에서 제공하는 하이퍼링크 기능을 이용하여 실습에 관련된 사항이나 정보를 실습 환경 속에 포함할 수 있어 실습을 위한 통합된 환경을 제공한다.

제안된 모델의 유용성을 검증하기 위하여 본 논문에서는 다양한 프로그래밍 언어를 실습하기 위한 시스템, UNIX 운영체제 실습 시스템, 오라클 데이타베이스 실습 시스템을 제안된 모델을 이용하여 구현하였다.

[1] http://pl.changwon.ac.kr

1. 서 론

컴퓨터 관련 교육에 있어서 실험과 실습은 교육 내용의 효과적인 전달과 교육의 목적 달성을 위하여 대단히 중요한 부분이다. 현재의 환경에서 소프트웨어의 실험과 실습을 위해서는 실습하고자 하는 프로그램을 실습자가 사용하는 컴퓨터에 모두 설치하여야 한다. 그런데, 실습 프로그램을 개인의 시스템에 설치하여 사용하는 것은 다음과 같은 문제점이 있다. 첫째, 새로운 소프트웨어에 대하여 실험을 하는 경우, 학습자는 해당 소프트웨어를 직접 설치하여야 한다. 소프트웨어를 구하고 설치하는 작업은 소프트웨어의 실습과는 직접적인 연관이 없는 일로서, 이는 학습자에게는 소프트웨어를 익힌다는 본래의 목적이 아닌 추가의 부담이 된다. 둘째, 하나의 소프트웨어에 대하여 여러 가지의 구현이 존재하는 것이 일반적이다. 여러 구현의 차이점이나 장단점을 비교/분석하기 위하여 각각의 구현을 설치하고 사용해야 하는데, 이 또한 쉬운 일은 아니다. 셋째, 소프트웨어가 업그레이드(upgrade)되면 새로운 버전을 설치해야 한다. 새로운 버전에 대한 정보를 얻는 것과 다시 설치하는 일은 모두 상당한 부담이 된다. 넷째, 소프트웨어를 각각의 시스템에 유지하는 것은 시스템 자원의 낭비를 가지고 온다. 여러 소프트웨어에 대한 처리기를 모두 갖추는 경우 이러한 문제점은 더욱 부각될 것이다.

소프트웨어를 중앙의 서버에 설치하고 사용자는 터미널을 사용하는 경우에 있어서도 (1) 학습자는 소프트웨어를 사용하기 위하여 서버에 로긴(login)하는 과정이 필요하며, 서버에 계정이 없는 경우나 직접 연결할 수 없는 경우에 사용할 수 없고, (2) 사용할 수 있는 경우라 할지라도 열악한 유저 인터페이스로 인하여 사용상의 어려움이 있다.

WWW은 사용하기 쉬운 인터페이스를 기반으로 현재 가장 많이 사용되고 있는 인터넷 서비스이다. 하이퍼미디어를 이용하여 전세계에 퍼져있는 다양한 정보를 서로 연결할 수 있으므로 교육 매체로서 많이 활용되고 있다. 한편, Java[1]는 인터넷 환경에서 가장 많이 사용되는 언어로서 정적인 웹 페이지를 동적으로 만드는데 주요한 역할을 한다. WWW과 Java를 소프트웨어 실습에 응용하면 사용자의 입장에서는 간단하고 편리한 인터페이스로 학습에 임할 수 있고, 관리자의 입장에서도 관리 및 유지 보수의 편리성을 가져온다.

기존의 개별적으로 소프트웨어를 유지하는데 발생하는 여러 문제점들에 대한 해결 방안으로서, 본 연구에서는 WWW의 일관된 인터페이스로서 쉽고 간편하게 소프트웨어에 대한 실습을 할 수 있는 환경을 구축하기 위한 기본적인 모델을 제시하였다. 제안된 모델은 클라이언트-서버 구조를 기반으로 하여 구축되었으며, 실습자는 웹 브라우저를 실행함으로써 실습에 대한 준비가 끝나며, 프로그램의 실행이나 처리는 서버에서 모두 이루어진다.

제안된 모델을 이용하여 본 논문에서는 다양한 프로그래밍 언어를 실습하기 위한 시스템, UNIX 운영체제 실습 시스템, 오라클 데이타베이스 실습 시스템 등을 구현하였다. 구현된 프로그래밍 언어 실습 시스템에서는 여러 언어에 대하여 언어처리기를 동시에 제공하며, 한 언어에 대하여 여러 버전의 언어처리기가 존재하는 경우 이들을 동시에 제공할 수 있으며, 편리한 사용자 인터페이스의 제공한다. 이 환경을 통하여 학습자는 개인의 시스템에 언어처리기를 설치하지 않고도 프로그래밍 언어를 학습할 수 있으며, WWW을 이용하여 프로그래밍 언어와 관련된 전세계에 퍼져있는 정보를 쉽고 빠르게 이용할 수도 있다.

구현된 UNIX 운영체제 실습 시스템과 오라클 데이타베이스 실습 시스템은 UNIX 또는 오라클을 사용할 수 있는 환경을 갖추지 못한 사용자들에게도 웹 브라우저를 통하여 실습할 수 있는 환경을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 이전의 연구들에 대하여 간략히 살펴보며, 3장에서는 본 논문에서 제안하는 실습 환경 구축용 클라이언트-서버 모델을 설명한다. 4장에서는 제안된 모델을 기초로 실제로 구축한 시스템을 소개하며, 5장에서 결론을 맺는다.

2. 관련 연구

2.1 소프트웨어의 실습 형태

종래의 소프트웨어의 사용이나 실습은 중앙집중식 환경에서 이루어졌으며, 현재도 용량이 많이 필요하거나 빠른 처리 속도를 요구하는 경우에 여전히 사용되는 방법이다. 이 방식에서 소프트웨어는 고가의 대용량 서버에 존재하면서 요구가 있을 시 서버에서 실행되며, 사용자는 일반적으로 텍스트 환경의 단말기를 사용한다. 이러한 환경에서 실습을 위해서는 로긴, 프로그램 실행 방법 등 서버에 대한 기본적인 조작이 필요하며 텍스트 환경의 단조로운 사용자 인터페이스가 단점이다.

개인용 컴퓨터(PC)가 널리 보급되고 그 성능 또한 빠르게 향상되면서 소프트웨어들을 개인용 컴퓨터에서 설치하여 사용하는 것이 보편화되었다. 이런 환경에서는 뛰어난 사용자 인터페이스와 분산처리의 장점은 있지만, 소프트웨어의 관리가 문제가 된다. 소프트웨어의 설치나 변경, 성능향상 시 사용자가 직접 이에 대한 처리를 담당하여야 하므로 부담이 커지게 된다.

최근 컴퓨팅 환경은 인터넷을 중심으로 변화하고 있다. 이 환경에서는 소프트웨어가 중앙의 서버에 존재하고 클라이언트의 요청에 의하여 실행된다. 클라이언트는 인터넷상에 존재하는 어떠한 노드도 가능하며 주로 웹 브라우저가 된다. 이 환경의 장점으로는 클라이언트에 웹 브라우저만 있으면 서버의 응용 프로그램과 데이타를 이용할 수 있고, 소프트웨어의 관리를 서버에 집중시킬 수 있다는 점이다. 잘 알려진 인터페이스와 멀티미디어의 지원 등도 큰 강점이다.

2.2 인터넷을 이용한 프로그래밍 언어 실습

인터넷을 이용한 프로그래밍 언어를 실습하고자 하는 연구는 Java 언어가 등장하면서 활성화되었다. 번역(compile)되는 언어와 해석(interpret)되는 언어에 대하여 현재까지 개발된 대표적인 시스템은 다음과 같다.

(1) 번역 언어

번역 언어 중에서 현재는 Java에 대해서만 개발되어 있다. Java는 번역되면 기계-중립적인 바이트 코드를 생성하는데 반하여, C, Pascal, Ada와 같은 다른 번역 언어들은 목적 기계(target machine)에서만 실행 가능한 코드를 생성하기 때문에 현재는 인터넷을 이용한 서비스는 개발되어 있지 않다.

Java Compiler Service[7]는 클라이언트에서 개발한 프로그램을 번역하여 그 결과를 전달하는 시스템으로서 Java 컴파일러를 갖추지 못한 사용자라 할지라도 Java 언어로 프로그램하고 실습할 수 있는 환경을 제공한다.

(2) 해석 언어

해석 언어가 인터넷에서 서비스될 때는 대부분 Java 애플릿의 형태로 HTML 문서에 포함되어 사용된다. 즉, 클라이언트에서 해당 HTML 문서를 로딩하면 언어처리기가 같이 로딩되어 실행된다.

대표적인 시스템으로는 Forth 프로그래밍 언어를 구현한 Misty Beach Forth[8], Prolog 언어를 구현한 W-Prolog[9], BASIC 언어를 구현한 COCOA[10], Lisp 언어를 구현한 webLISP[11] 등이 있다.

현재까지 개발된 사례들에서 번역 언어의 경우 Java를 제외한 다른 언어들은 제공되지 않으며, 해석 언어의 경우에는 하나의 애플릿은 하나의 언어만을 지원하기 때문에 여러 언어를 사용하기 위해서는 각각을 새로이 연결하고 로딩하여야 하는 단점이 있다.

본 연구에서는 단 한번의 연결로 다양한 종류의 번역 언어와 해석 언어를 모두 지원하는 시스템을 구축하고자 하였으며, 이러한 시스템의 근간이 되는 클라이언트-서버 실습환경의 모델을 설계하고 Java로 구현하였다.

3. 클라이언트-서버 실습환경

제안된 실습환경의 시스템 구조는 그림 1과 같

다. 웹 브라우저로 실습 홈페이지에 접속하면 클라이언트 프로그램인 Java 애플릿을 서버로부터 가져와서 실행한다. 애플릿이 실행되면 웹과는 다른 채널로 서버와 통신하게 된다.

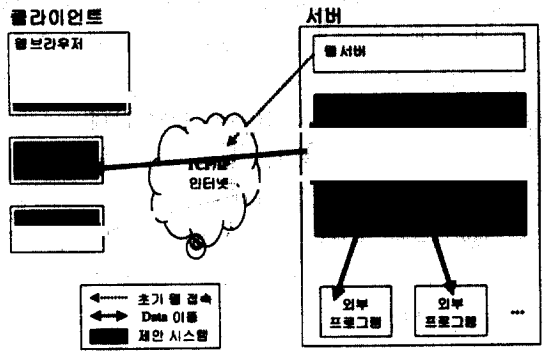

그림 1. 가상실습환경의 구조

그림에서 보듯이, 실습용 클라이언트는 웹 브라우저 속에서 실행되며, 자체적으로 TCP/IP 기능이 있어서 웹 브라우저의 도움 없이 실행된다. 즉, 본 모델에서 웹은 사용자와 실습 서버 사이의 초기 연결을 담당하며, 도움말이나 예제 프로그램을 제공하는 역할을 담당한다. 또한 실습 서버는 웹 서버와는 별도로 실행되는 프로그램으로 실습 클라이언트와 TCP/IP로 직접적으로 통신한다.

3.1 고려사항

본 실습환경에서 클라이언트는 Java를 실행 가능한 웹 브라우저이며, 하드웨어나 운영체제의 종류와는 상관이 없지만 낮은 사양의 컴퓨터를 가진 사용자를 위하여 클라이언트의 부담을 최소화 하는 것이 중요하다. 이를 위하여 본 시스템에서는 서비스되는 소프트웨어 프로그램 및 소프트웨어에 대한 정보를 클라이언트가 아닌 서버가 보관토록 하였다. 따라서 기존 서비스되는 실습 환경에 대한 어떠한 변경이나 새로운 실습환경의 구축하여 서비스하는 경우에도 클라이언트는 전혀 변경될 필요가 없다. 서버가 보관하는 구성(configuration) 정보에는 제공하는 소프트웨어의 종류와 실행정보, 화면 배치 정보, 창과 버튼에 관한 정보 등이다.

학습자가 보다 편리한 환경에서 프로그램을 작성할 수 있는 기능 역시 본 시스템의 중요한 설계 요구 사항이다. 이를 위하여 WWW을 기반으로 설계하였으며, 입력 편의를 위한 편집 창, 매뉴얼이나 예제 프로그램을 위한 링크 등을 포함한다. 특히, 편집 창은 입력의 기록을 위하여 다수의 프로그램을 동시에 보관할 수 있도록 설계되었다.

학습자가 우연 또는 고의로 악성 프로그램을 작성하여 실행하는 것을 방지하기 위해 서버 자체에서 필터(filter) 기능을 포함하였다. 서버의 시스템을 조작하거나 디스크를 접근하는 등의 작업은 필터링을 통하여 거부될 수 있다.

3.2 연결 형태

그림 1에서 보듯이 실습 서버는 웹 서버와는 독립적으로 동작하며, 클라이언트가 요청한 소프트웨어를 실행하여, 실행한 프로세스와 클라이언트를 서로 연결해 주는 역할을 한다. 서버가 연결해 주는 형태에 따라 직접 연결과 간접 연결의 두 가지 방식이 있다.

(1) 직접 연결

클라이언트가 서버에서 실행되는 프로세스와 직접적으로 연결되어 클라이언트가 입력한 내용을 서버가 즉시 처리하는 형태이다. 직접 연결로 처리하는 실습환경에서 서버는 해당 소프트웨어를 실행하여 프로세스를 생성하고 클라이언트와 연결되도록 한다. 이후의 서버의 동작은 이 프로세스와 클라이언트 사이의 데이타 전달과 필터링의 역할만 한다.

(2) 간접 연결

클라이언트가 입력한 내용을 저장하거나 가공하였다가, 추후에 클라이언트의 요청이 있을 때 이를 다시 처리하는 방법이다. 클라이언트가 보내는 정보는 서버에서 실행한 프로그램에게 전달되거나 임시로 저장된다. 간접 연결로 처리하는 실습환경에서 학습자가 실습환경을 요청할 경우 서버는 화면 구성을 위한 정보만을 클라이언트에 보내며, 어떠

한 프로세스도 생성하지 않는다. 이후에 클라이언트가 처리를 요청하면 소프트웨어를 호출하여 실행하고 그 결과를 보여준다. 이러한 형태의 연결에서 서버의 역할은 클라이언트로부터의 정보를 가공, 저장, 재처리의 기능을 가지고 있어야 한다.

3.3 구성 화일

서버와 클라이언트에서 필요로 하는 정보는 구성(configuration) 화일에 저장되어 있으며, 시스템의 전체 동작의 많은 부분이 이곳의 정보들을 중심으로 작동한다.

구성 화일 내에서 정보의 단위는 Section 이다. Section 은 Component 정보와 Action 정보 이루어져 있다. 구성 정보를 확장된 BNF 형태로 정리하면 다음과 같다.

```
Configuration ::=
        Section*
Section ::=
        '[' SectionName ']' Component* ';'
Component ::=
        BackGround | MainArea | EditArea
        | Button | Radio | Exec | Filter
BackGround ::=
        'BG=' Area ',' Path_image
MainArea ::=  'Content=' Area
EditArea ::=  'Edit=' Area
Button ::=  'Button=' Area ',' Label ',' Action
Action ::=  'SCR+' SectionName | 'URL+' URL
        | 'EXEC' | 'COMPILE'
        | 'EDITBOX' | 'EDITMODE'
Radio ::=  'Check=' Area ',' Label
Exec ::=  Run | Interpret | Compile
Run ::='Run=' [ Path_program ]
Interpret ::=  'Interp=' Path_interpreter
Compile ::=  'Comp=' Path_compiler
Filter ::=  'Filter=' String '/' Conv
Conv ::=  'IGNORE' | String
SectionName ::=  'INIT' | String
Area ::=  Num_left ',' Num_top ',' Num_width ',' Num_height
Label ::=  'TXT+' String | 'IMG+' Path_image
String =  문자열
Num =  숫자
Path =  UNIX pathname 표기
URL =  인터넷 URL 표기
```

구성 화일에서 [INIT]는 초기 화면을 구성하는 예약된 Section 이름이다. 화면 배치를 위한 요소는 Component 들로 이루어지는데, 각 요소를 정리하면 다음의 표와 같다.

BG	배경 화면의 그림을 지정
Content	입출력 영역의 위치를 지정
Edit	편집 영역의 위치를 지정
Button	버튼 생성
Check	체크박스 생성

Content 로 지정된 영역이 기본 인터페이스 영역이다. 버튼은 두 종류를 지정할 수 있는데, 버튼 속에 글자를 나타내려면 **TXT** 레이블을 사용하고, 그림 버튼을 나타내려면 **IMG** 레이블을 사용한다.

글자 버튼	TXT+초기화면	
그림 버튼	IMG+first.jpg,	

버튼이 눌러지면 그에 따라 적절한 반응이 일어나야 한다. 버튼이 눌러졌을 때 실행해야 할 Action 에는 다음의 종류가 있다.

SCR	특정 Section 을 요청
COMPILE	클라이언트에서 전송된 입력의 가공을 요청
EXEC	일차로 가공된 입력의 처리를 요청
EDITBOX	직접 연결 형태에서 편집을 위한 창을 요청
EDITMODE	간접 연결 형태에서 편집을 위한 창을 요청
URL	새로운 웹 페이지를 요청

외부 프로그램을 실행하기 위한 항목으로는 **Interp, Comp, Run** 등이 있다. **Interp** 는 직접 연결에 의한 프로그램의 실행시 사용되고, **Comp** 와 **Run** 은 간접 연결로 실행시에 사용된다. 이들을 정리하면 다음의 표와 같다.

Interp	직접 연결하여 실행할 프로그램을 지정
Comp	간접 연결 시 입력을 일차로 가공할 프로그램을 지정
Run	Comp에 의하여 가공된 처리하는 방법을 지정

Comp는 COMPILE Action을 가진 버튼이 선택되었을 때, 실행될 프로그램을 지정하고, Run은 EXEC Action을 가진 버튼이 눌러졌을 때, 실행될 방법을 지정한다. Run에서 지정된 방법으로는 외부 프로그램을 필요로 하는 경우와 그렇지 않은 경우가 있다. Java 언어 처리기와 같이 외부 프로그램(즉, Java 바이트코드 해석기)의 도움을 받아서 실행하는 경우 그 프로그램의 위치를 지정한다. 또한 프로그램 실행 시에 본 시스템에서 제공하는 필터(filter)가 아닌 다른 종류의 필터링 프로그램 등을 등록할 때 유용하게 사용할 수 있다.

Filter는 서버의 보안상 서버에 전달되면 안되는 상황에서 사용한다. 해석형 언어에서 적용되는 내용으로 어떠한 문자열을 다른 문자열로 바꾸거나 무시할 때 사용한다.

입력 무시	load/IGNORE	클라이언트에서 "load"라는 스트링이 전송되면 서버에서는 이 스트링을 무시한다.
입력 변환	save/FTP	클라이언트에서 "save"라는 스트링이 전송되면 서버에서는 이 스트링을 "FTP"로 바꾸어서 처리한다.

다음은 구성 화일의 예인데, 프로그래밍 언어의 실습을 위한 구성 화일이다.

```
[INIT]
  BG=0,0,700,600, logo.jpg
  Button=710,10,90,30, IMG+f_java.jpg, SCR+Java
  Button=710,50,90,30, IMG+f_scm.jpg, SCR+SCM
  ;
[Java]
  Edit=0,0,600,600
  BG=700,0,100,400,l_java.jpg
  Button=710,10,90,30, IMG+first.jpg, SCR+INIT
  Button=710,50,90,30, IMG+h_java.jpg, URL+http://java.sun.com/
  Button=710,90,90,30, IMG+guide.jpg, URL+http://java.sun.com/docs
  Button=710,130,90,30, IMG+edit.jpg, EDITMODE
  Button=710,170,90,30, IMG+comp.jpg, COMPILE
  Button=710,210,90,30, IMG+exec.jpg, EXEC
  Check=320,10,50,30, TXT+Applet
  Comp=/usr/java/bin/javac -classpath /usr/java/lib/classes.zip
  Run=/usr/java/bin/java -classpath /usr/java/lib/classes.zip
  ;
[SCM]
  Content=0,0,600,600
  BG=700,0,100,400,l_scheme.jpg
  Button=710,10,90,30, IMG+first.jpg, SCR+INIT
  Button=710,50,90,30, IMG+h_scheme.jpg, URL+http://www-swiss.ai.mit.edu/scheme-home.html
  Button=710,90,90,30, IMG+guide.jpg, URL+http://www.yahoo.com/Computers_and_Internet/Programming_Languages/Scheme/
  Button=710,130,90,30, IMG+edit.jpg, EDITBOX
  Interp=/home2/pl/pulgrim/scm/scmlit
  Filter=load/IGNORE, save/FTP
  ;
```

초기화면 [INIT]에서는 배경그림 logo.jpg와 두 개의 언어선택 버튼이 있으며, [Java]에는 편집 창과 6개의 버튼 그리고, 컴파일러의 위치를 알려주기 위한 Comp와 Java 프로그램을 실행하기 위한 Run Component가 포함되어 있다. [SCM]에는 하나의 Content와 4개의 버튼 그리고 인터프리터의 위치를 알려주기 위한 Interp가 있다. Filter에서는 두 개의 명령어를 항상 검사하는데, "load"를 만나면 무시하고, "save"를 만나면 다른 스트링으로 바꾸어서 실행한다. 각 Section에 있는 Button들은 각각 다른 작동을 하도록 지정되어 있다.

3.4 구현

제안된 모델은 Java 언어로 구현되었으며, JDK1.1를 이용하여 컴파일되었다. 서버는 Java Application으로 약 730줄이며, 클라이언트는 Java 애플릿으로 약 1770줄의 크기이다.

서버는 항상 실행되는 데몬 프로그램으로 작성되었으며, 클라이언트로부터의 요청이 있을 시 쓰레

드(thread)를 생성하여 개별적 클라이언트에 대한 처리를 전담하도록 하였다. 서버의 주요한 기능으로는 TCP/IP 연결, 구성 화일을 통한 환경 설정, 쓰레드 생성, 외부 프로그램 호출, 필터링 등이 있다.

클라이언트의 주요 기능으로는 TCP/IP 접속, GUI 컴퍼넌트 생성, 이벤트의 처리 등이 있다.

3.5 장점과 비교

본 모델의 장점을 5가지 측면에서 살펴보면 다음과 같다.

- 성능: 클라이언트-서버 모델은 클라이언트의 서비스 요청을 서버가 받아서 처리하여 그 결과를 되돌려주는 형태로 이루어져 있다. 따라서 본 모델은 필요할 때 즉시 서버에 접속하여 복잡한 계산을 처리할 수 있기 때문에 클라이언트의 성능이 좋지 않아도 성능이 좋은 서버의 처리 능력을 사용할 수 있다.

- 유지보수성: 전체 시스템에 대한 관리도 서버에만 집중되므로 클라이언트 쪽의 사용자의 부담이 경감된다. 기존의 소프트웨어의 관리, 설치된 소프트웨어의 업그레이드, 새로운 소프트웨어의 설치 등은 모두 서버에서 이루어진다.

- 안정성: 클라이언트-서버 모델에서 프로그램의 실제적인 실행은 서버에서 이루어지며, 클라이언트는 그 결과를 받아서 이용한다. 나쁜 의도를 가진 악성의 프로그램에 대한 방비도 서버만의 책임이다. 따라서 안전한지 않은지 모르는 미지의 프로그램에 의한 어떠한 보안 위협도 클라이언트 측에서는 없다.

- 확장성: 본 모델에 의하여 실습 환경을 구축할 경우 새로운 소프트웨어를 제공하고자 하는 경우에 서버에 소프트웨어를 설치하고 구성 화일만 조절하여 주면 되므로 다양한 소프트웨어를 제공하도록 쉽게 확장할 수 있다.

- 유용성: 본 모델의 클라이언트로서는 Java가 실행 가능한 웹 브라우저만 있으면 충분하므로 누구든지 어디에서나 실행가 능하다.

본 모델을 기존의 인터넷에서 제공되는 번역 언어 시스템과 비교하면 다음의 표와 같다.

기능	번역	번역 및 실행
프로그램 실행	번역 결과를 로딩하여 실행	서버에서 바로 실행
제공 언어	Java	다양한 언어 모두 가능

본 모델을 기존의 인터넷에서 제공되는 해석 언어 시스템과 비교하면 다음의 표와 같다.

제공 언어	각 시스템 당 하나의 언어	다양한 언어를 모두 지원
크기	해석기가 내장되어 크기가 크다	사용자 인터페이스 부분만 있어서 크기가 작다.
실행 위치	클라이언트	서버
실행 속도	클라이언트의 성능에 의존	서버의 성능에 의존
유지 보수 대상	각 클라이언트 모두	서버 한군데

4. 제안된 모델의 응용

제안된 시스템 구조를 바탕으로 프로그래밍 언어 가상실습환경을 제공하는 "풀그림서당"[6]과 UNIX 및 오라클을 실습할 수 있는 시스템을 각각 구현하였다.

4.1 초기화면

풀그림서당은 다양한 플랫폼에서 이용 가능하기 위해 Java 1.1.x를 기준으로 구현하였으며, 이 시스템을 이용하기 위해 학습자는 Java 애플릿을 수행할 수 있는 웹 브라우저를 가지고 있어야 한다. 그림 2는 풀그림서당을 수행하였을 때의 초기화면이다. 왼쪽 아래에 언어 선택을 위한 버튼과 UNIX 실습과 오라클 실습을 위한 버튼이 보인다.

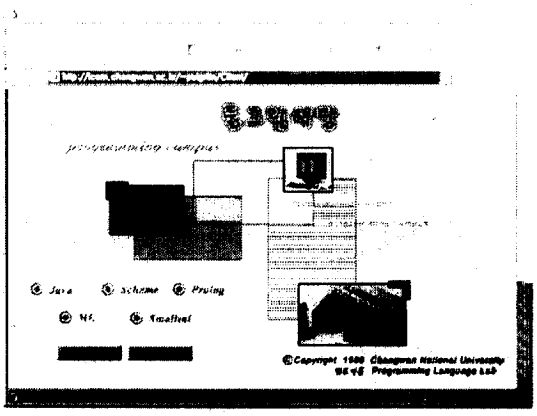

그림 2. 가상실습환경의 초기화면

현재 구현된 언어[1-5]와 처리기는 다음의 표와 같다. 이외에도 언어처리기가 갖추어지면 얼마든지 확장이 가능하다

Java	JDK1.2
Prolog	BinProlog 5.75
ML	SML/NJ 110
Scheme	MIT Scheme 5b1
Smalltalk	GNU Smalltalk 1.1.5

4.2 직접 연결형

그림 3은 직접 연결형으로 작동하는 실습환경에서의 작업환경을 보여주고 있다. 이 그림에서 보듯이 몇 개의 버튼과 실행되는 프로그램과 자료를 주고 받기 위한 실행 창이 제공된다. 화면 왼편은 프로그램이 실행되어 명령어 입력 대기 상태의 모습이다. 이 창에서 직접 입력하면 실행된 결과가 출력된다. 그리고 프로그램 실행 중에 사용하는 프로그램에 대한 정보가 필요할 경우는 홈 페이지 버튼이나 가이드 버튼을 눌러서 관련 정보를 쉽게 이용할 수 있도록 하였다.

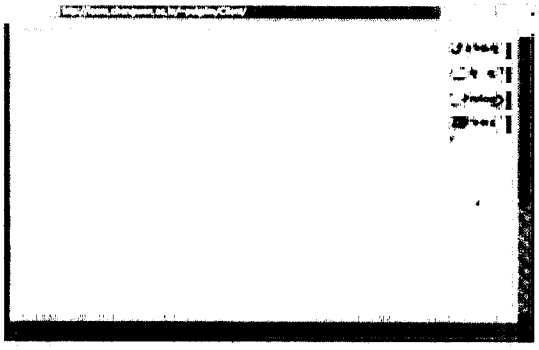

그림 3. 직접 연결형 인터페이스

그림 4는 편집 버튼에 의해 생성되며 여러 줄의 입력을 작성/수정을 쉽게 하는 편집상자 창이다.

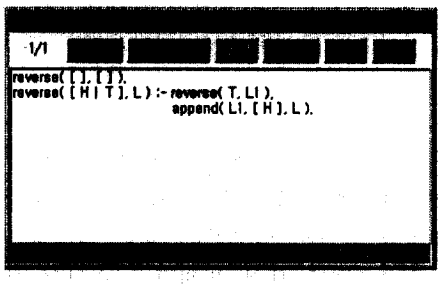

그림 4. 입력을 위한 편집 창

4.3 간접 연결형

그림 5는 간접 연결형으로 작동하는 실습환경에서의 작업환경을 보여주고 있다.

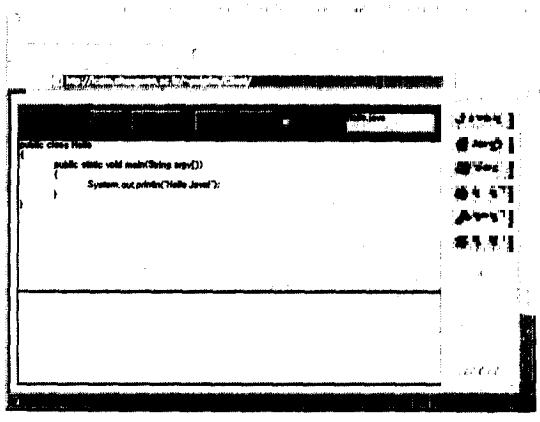

그림 5. 간접 연결형 인터페이스

간접 연결형의 프로그램에서는 입력할 내용을 먼저 작성하고, 실행을 위해 적당한 형태의 변형 과정을 거친다. 풀그림서당에서는 간접 연결 형태를 위해서 편집 창과 실행 창을 각각 제공하였다. 편집 창은 입력을 작성하고 가공할 수 있으며, 실행 창은 직접 연결형의 프로그램의 수행과 동일하다. 편집 버튼과 실행 버튼으로 편집 창과 실행 창을 서로 옮겨 다닐 수 있도록 하였다.

4.4 WWW 기반 UNIX 실습 시스템

그림 6은 UNIX 실습 시스템의 작동을 보여주고 있다.

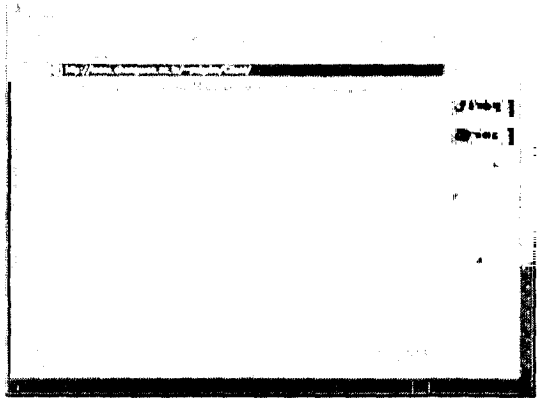

그림 6. UNIX 실습 시스템

본 UNIX 실습 시스템의 기능으로는 도움말 기능으로 연결되는 버튼과 자주 사용되는 용어를 자동 입력하는 버튼들이 제공된다. 화면의 오른쪽의 버튼들은 UNIX 시스템에 대한 정보를 위한 것이며, 아래쪽의 버튼은 입력을 대신하는 버튼들이다.

본 시스템은 UNIX가 설치되어 있지 않은 어떠한 컴퓨터에서도 웹 브라우저만 있으면 UNIX 시스템을 실습할 수 있다는데 의의가 있다.

4.5 온라인 오라클 실습 시스템

그림 7은 오라클 데이타베이스를 실습하기 위한 작업환경을 보여주고 있다.

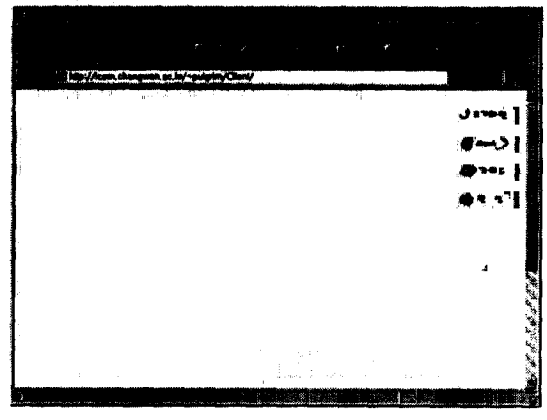

그림 7. 오라클 실습 시스템

본 오라클 실습 시스템에서는 다양한 예제를 통하여 데이타베이스를 실습할 수 있는 것에 중점을 두고 구현되었다. 화면의 아래쪽의 버튼들이 여러 예제를 참고할 수 있는 곳으로 링크되어 있다.

UNIX 실습 시스템과 마찬가지로 오라클 데이타베이스 실습 시스템에서도, 오라클이 설치되어 있지 않은 컴퓨터에서도 웹 브라우저를 통하여 실습이 가능하도록 하였다.

5. 결론

본 연구에서는 인터넷을 이용하여 소프트웨어의 실습을 지원하는 모델을 설계하고 구현하였다. 또

한 이 모델을 기반으로 실제로 여러 가지의 다양한 실습환경을 구축하여 본 모델의 유용성을 증명하였다.

본 시스템의 가장 큰 특징은 웹을 통해서 다양한 프로그램을 실습해 볼 수 있다는 것이다. 각각의 클라이언트에는 단지 웹 브라우저가 존재할 뿐이며, 실습하고자 하는 프로그램을 설치하지 않아도 되기 때문에 자원과 시간을 절약할 수 있다. 동시에 학습자에게는 쉽게 접하지 못하거나 설치가 어려운 프로그램을 경험을 해 볼 수 있는 환경을 제공할 수 있다.

서버에는 사용자들 수만큼의 계정을 생성해 줄 필요가 없으며, 서비스의 추가나 변경 시 서버 관리자는 간단히 구성 화일을 바꾸어 주면 된다.

본 논문은 클라이언트에게는 사용의 편리성과 접근의 용이성을 제공하고, 서버에게는 전체 시스템의 동작 제어를 구성 화일에 집중되도록 하여 시스템의 변경이나 추가에 유연히 대처한다는 점에서 그 효용성이 높다.

현재의 환경은 텍스트 위주로 처리되어, 프로그램의 입력이나 실행결과의 출력은 모두 텍스트의 형태로 나타난다. 비쥬얼(visual) 환경을 가상실습환경으로 구축하는 연구도 흥미로울 것이다.

참 고 문 헌

[1] K. Arnold and J. Gosling, *The Java Programming Language* (2/ed), Addison-Wesley, 1998.

[2] W. F. Clocksin and C. S. Mellish, *Programming in Prolog* (3/ed), Springer-Verlag, 1987.

[3] A. Goldberg and D. Robson, *Smalltalk-80: The Language*, Addison-Wesley, 1989.

[4] R. Milner, M. Tofte, and R. Harper, *The Definition of Standard ML*, MIT Press, 1990.

[5] J. Rees and W. Clinger(eds.), "The Revised^3 report on the algorithmic language Scheme," *ACM SIGPLAN Notices*, V21 N12, p.37-79, 1986.

[6] 풀그림서당, http://icom.changwon.ac.kr/~pulgrim/Client/

[7] Java Compiler Service , http://javaboutique.webdeveloper.com/compiler.html

[8] Misty Beach Forth , http://www.mistybeach.com/Forth/JavaForth.html

[9] W-Prolog, http://members.xoom.com/ winikoff/wp/

[10] COCOA, http://professionals.com/~cmc manis/java/javaworld/examples/BASIC.html

[11] webLISP, http://www.techno.net/pcl/tm/ plisp/pr-java/index.htm

이수현
1987년 광운대학교 전자계산학과 (학사)
1989년 한국과학기술원 전산학과 (석사)
1994년 한국과학기술원 전산학과 (박사)
1994년~1996년 한국전자통신연구원 선임연구원
1996년 창원대학교 컴퓨터공학과 조교수
연구분야: 프로그래밍 언어, 컴파일러, 웹의 교육적 활용