Custom Administration Package

Kevin Foote
CS470

12 - December, 2002

Contents

Abstract

This project combines a variety of technologies to create a customized web server administration package. This package contains a back end, feature-rich, console based set of scripts that can be alternately accessed through an Instant Messenger client using the Jabber protocol. Throughout this project, the goal was to correctly and accurately perform the administration routines, while providing a simple and efficient mechanism for this process.

1 Introduction

This project was developed to assist in the administration tasks of a remote web server. The clientele for this project were Nicholas Kirsch and myself. Current administration activities entail routine additions of web sites and additional user accounts.

Although the administration of this server can be handled by either persons mentioned above, the coordination of their combined efforts will be greatly assisted with the creation of this package. I decided to take on this task of creating a unified administration package to prevent unnecessary overlap of work by either administrator on this server. Since Nicholas is my brother-in-law, the meetings and conversations regarding this project were kept very informal. The requirements for this project were also loosely defined and conversed about regularly.

2 Project Overview

This project was designed to combine administration functions with today's instant messaging technology. The intended deployment of this software package is a remotely located web server. Currently, this machine hosts roughly twenty virtual hosts, requiring twenty-four hour uptime and availability.

The desired outcome of this project was to design and implement a basic administration suite that could be accessed both through the use of a remote UNIX shell and through an instant message client. The scope of this document will be limited to the explanation of the technologies used in this project and the processes used throughout the development of this project.

2.1 Administration Past

Currently, we maintain the server with a system of organizational structure that is very customized to our particular needs. This hierarchy allows us to remain organized and still provide hosting services to almost all top-level domain extentions.

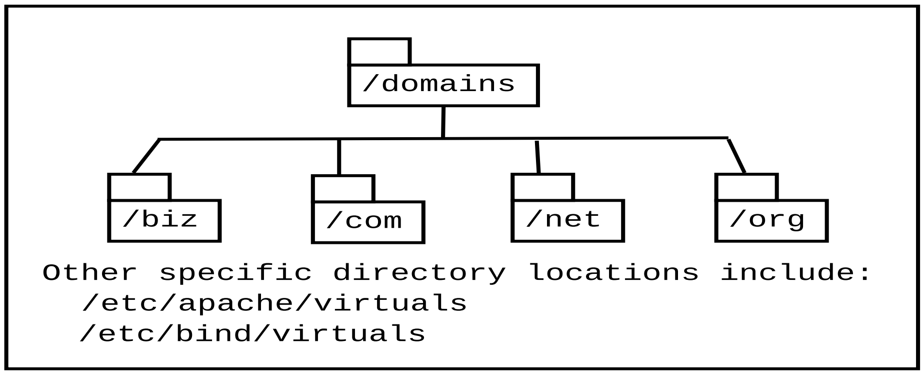


figure 1: (data organization)

This structured web server lends itself easily to organization; however, it requires repetition and attentiveness when performing routine maintenance tasks. A site addition entails adding the correct directory, adding a corresponding apache configuration file, adding a DNS record and restarting both apache and bind on the system.

2.2 Administration Updated

The need for an administrator to manually create each file and restart each system component is time consuming. Under the new system, administration on the web server should be simple and straightforward. The administrator is able to make quick additions to the system without the overhead of trying to remember each file that needs to be created or step that has to be taken.

An example command to create a new user email account on the server looks like the following and results in the addition of user myworld-jjhoward to the system:

```
$> admin.plx –user –add jjhoward@myworld.com
```

3 Requirements

The requirements set forth for this project by Nicholas and myself were to create a collection of solid scripts for administration purposes, with the added feature of being accessible through an instant messaging client.

3.1 Functional Specifications

The functional specifications were divided into two parts. Part one entails the functionality included in the Perl scripts of this package. The second part entails the functionality included in the Jabber component of the project. The functionality included in each component of the system is grouped below. These functions are crucial to the creation of this suite of administration programs.

3.1.1 Perl Administration Script Portion

The scripts written for this project are administration scripts meant for specific maintenance duties. These scripts provide an administrator with an environment that allows the following to be completed:

1. read in arguments passed on the command line; these arguments consist of actions to take, as well as the actual values to be added or removed
2. parse specific system files and add the desired content to the system in its correct format
3. write specific system files in the correct location and using the correct format
4. start and stop server daemons when needed
5. provide a standardized feedback to the user
6. log all activity to a local log file

3.1.2 Jabber Component Portion

The Jabber component written for this project acts as a transparent access port to the Perl scripts included in this project. This component provides administrators with the following functionality:

1. provide a secure communication connection to "subscribed" users
2. read incoming messages from users that it recognizes
3. parse the incoming messages into script commands
4. call specific administration scripts into action
5. relay script output back to the message originator

3.2 System Specifications

This administration system is made up of a collection of Perl scripts that run on a Debian Linux system using Perl v5.6.1. The Jabber client was created using GNU C/C++ and runs on the same Debian Linux system.

4 System Design

The design of this system is highly customized to the particular layout and system needs that were introduced early on. The scripts and mechanics they incorporate are highly focused for our needs. These scripts may provide little use to others except for reference purposes. The Jabber component is more flexible and can be recompiled to work with any running jabberd server.

The Perl scripting environment was chosen due to its ability to easily handle regular expressions and file manipulation within the UNIX environment. The Jabber instant messaging environment was chosen for its client-server architecture and XML based messaging protocol.

4.1 System Architecture

The building of this project was divided into two focal areas. The first self-sufficient product is a set of solid administration scripts that perform the redundant work relating to the addition of web sites and users to our web server. The second product is a Jabber component that acts as a relay for incoming administration commands.

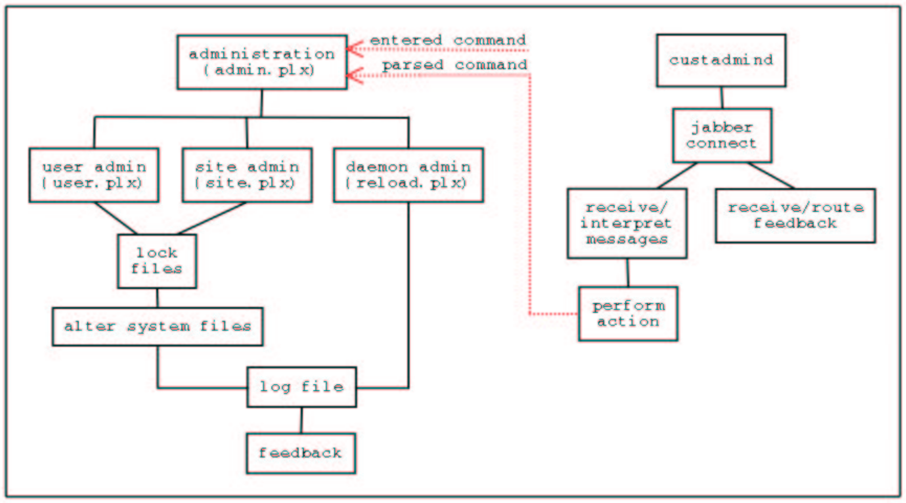


figure 2: (system architecture)

4.1.1 Algorithm

Within the Jabber component, this project implements a very simple looping algorithm that handles the data appropriately. An initial TCP socket is created with the Jabber server. The component then passes data to the server and blocks until data is received. Once data has been received, the component processes the data and returns the result.

4.1.2 Data Structure

Internally, the Jabber component uses simple data constructs to perform all its data manipulation. The basis for all communication between the Jabber server and this component is one, fragmented XML document. The document fragments are received and sent through our established socket connection. All fragments are operated on with string operators. The component populates internal data fields to determine what corresponding action to take. The resulting feedback is packaged into an XML fragment form and sent back to the Jabber server.

5 Development Process

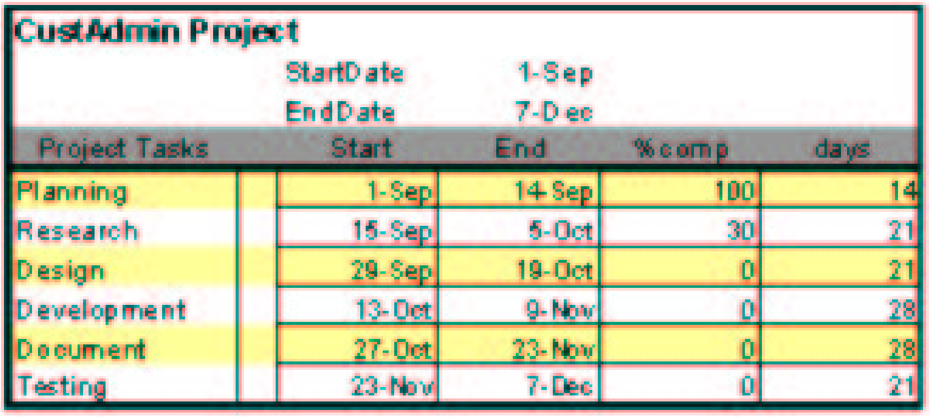
The development model I chose to use for this project was the waterfall model of program development. I chose to use this method because it seems to work well with the way I develop projects. The functionality of the system continuously builds on itself each step of the way until the desired functionality is present within the system.

5.1 Challenges

I encountered many challenges during the development of this project. An initial challenge was to correctly and securely set up my test Jabber server. Other challenges included learning the Perl scripting language and discovering what the Jabber protocol can and can not do. Another challenge during development of this package was keeping my work organized. I used CVS extensively, later in the project, both to keep a log of work and also to track changes to my code. These challenges proved to be great learning experiences and led to further understanding of real world project implementation.

5.2 Timeline Breakdown

The timeline for this project was modified slightly from the original timeline proposed (figure 3). The final schedule for this project shows more emphasis spent in the research phase and the development phase of this project, as well as a condensed overall project length (figure 4).

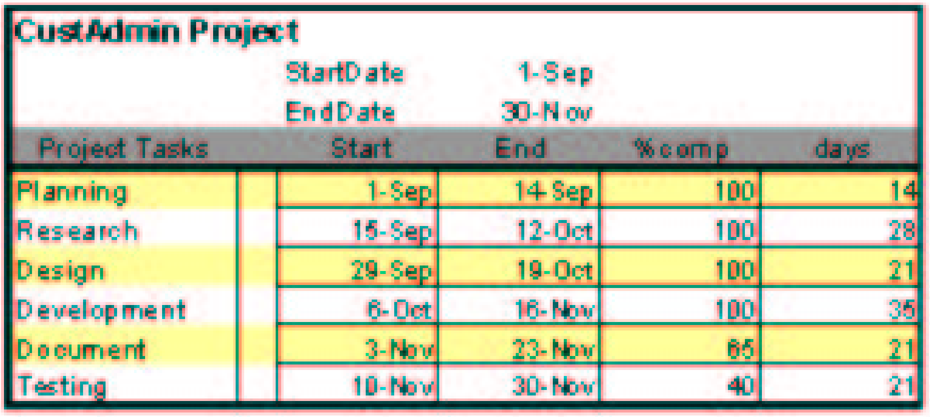


CustAdmin Project

	StartDate	1-Sep	
	EndDate	7-Dec	

Project Tasks		Start	End	%comp	days
Planning		1-Sep	14-Sep	100	14
Research		15-Sep	5-Oct	30	21
Design		29-Sep	19-Oct	0	21
Development		13-Oct	9-Nov	0	28
Document		27-Oct	23-Nov	0	28
Testing		23-Nov	7-Dec	0	21

figure 3: (proposed schedule)



CustAdmin Project

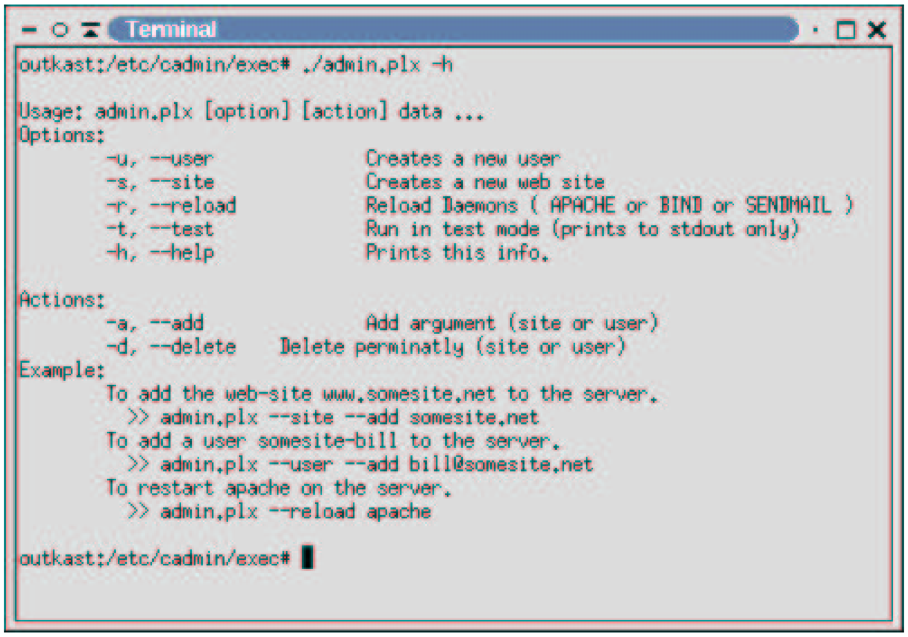
	StartDate	1-Sep	
	EndDate	30-Nov	

Project Tasks		Start	End	%comp	days
Planning		1-Sep	14-Sep	100	14
Research		15-Sep	12-Oct	100	28
Design		29-Sep	19-Oct	100	21
Development		6-Oct	16-Nov	100	35
Document		3-Nov	23-Nov	65	21
Testing		10-Nov	30-Nov	40	21

figure 4: (actual schedule)

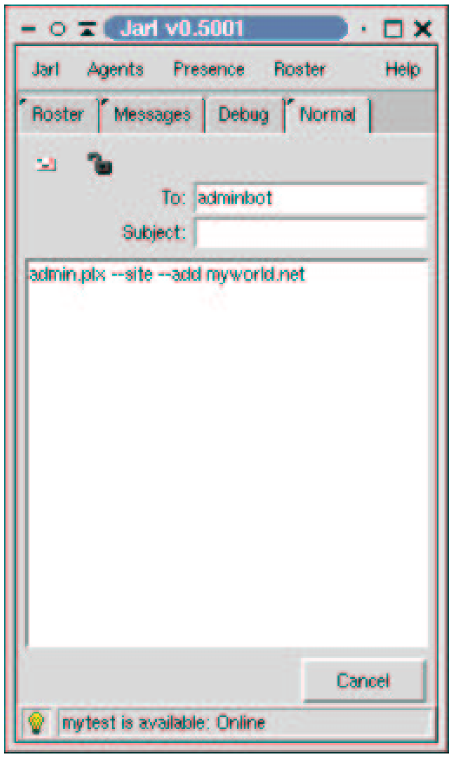
6 Result

The final result that emerged from this project is a complete working system for administration through a UNIX console and through the Jabber instant messaging protocol. The included set of administration scripts sufficiently handles the addition of sites and users to this system through an easy to follow command line interface (figure 5). This functionality can also be accessed through the Jabber component incorporated in this project. The Jabber component listens for administration commands coming from a client and then returns the results upon completion (figures 6 & 7).

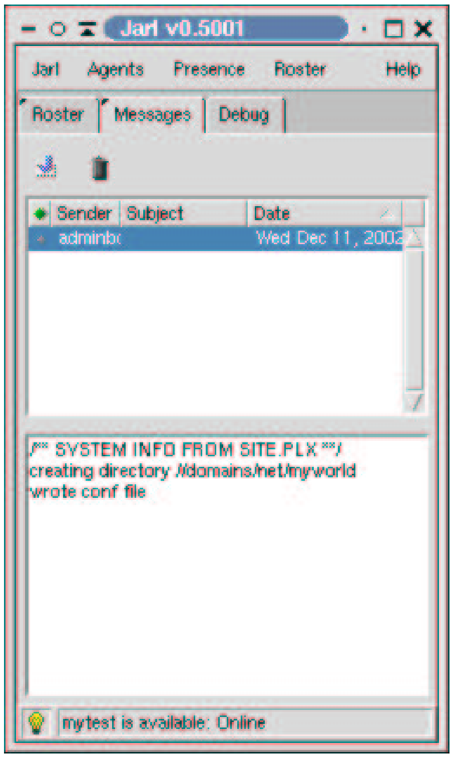


```
outkast:/etc/cadmin/exec# ./admin.plx -h

Usage: admin.plx [option] [action] data ...
Options:
        -u, --user              Creates a new user
        -s, --site              Creates a new web site
        -r, --reload            Reload Daemons ( APACHE or BIND or SENDMAIL )
        -t, --test              Run in test mode (prints to stdout only)
        -h, --help              Prints this info.

Actions:
        -a, --add               Add argument (site or user)
        -d, --delete    Delete perminatly (site or user)
Example:
        To add the web-site www.somesite.net to the server.
          >> admin.plx --site --add somesite.net
        To add a user somesite-bill to the server.
          >> admin.plx --user --add bill@somesite.net
        To restart apache on the server.
          >> admin.plx --reload apache

outkast:/etc/cadmin/exec# 
```

figure 5: (command line interface)



```
admin.plx --site --add myworld.net
```



```
/** SYSTEM INFO FROM SITE.PLX **/
creating directory ./domains/net/myworld
wrote conf file
```

figure 6: (message to component)   figure 7: (message from component)

6.1 Future Development

I plan to continue to work on this project and to expand its capabilities. Additions to the system may include a module that gives periodic updates of the host system to clients. Also, work is underway to make use of public libraries for handling the XML data, both incoming and outgoing. A final advancement I would like to make in this project is to rewrite the socket looping code so that it incorporates threads. This would greatly help the back-grounding of this daemon. These aspects give this project both room to grow and to expand.

7 Conclusion

The CustAdmin package was developed using GNU C/C++ and the Perl scripting language for Nicholas Kirsch and myself with the goal of creating a unified administration package for our web server. The project was completed on time and incorporates the necessary functionality for this system. Currently, further development and functionality is being actively planned by the clientele.

Overall, this project provided a great handson learning experience for developing real-world software. I was able to learn valuable lessons about project organization from beginning to end and the importance of organizing work schedules to produce on a deadline. The added lessons in code design and layout, as well as learning a new programming language, contributed to the overall learning experience.

References

[1] Adams, DJ (2002), Programming Jabber. O'Reilly & Associates, Inc., Sebastopol, CA.

Appendix A: [User Manual]

Custom Administration User Manual

System Requirements

GNU Debian Linux
Perl v5.6.1
An installed & working Jabber server (preferably jabberd)

Install

To install this package you must have the root user permissions. The scripts and Jabber component reside in /etc/cadmin/exec. To install these into this directory run the included install file as root. ex: $> ./install

This package assumes you have a currently working jabberd server running on your system. Some modification will need to be made to the /etc/jabber/jabber.xml file.

Modifying jabber.xml

In the <browse> section of jabber.xml insert the following lines:

```
<service type=”adminbot” jid=”adminbot.yourserver” name=AdminBot Tool”>
<ns>jabber::iq::agent</ns>
<ns>jabber::iq::auth</ns>
</service>
```

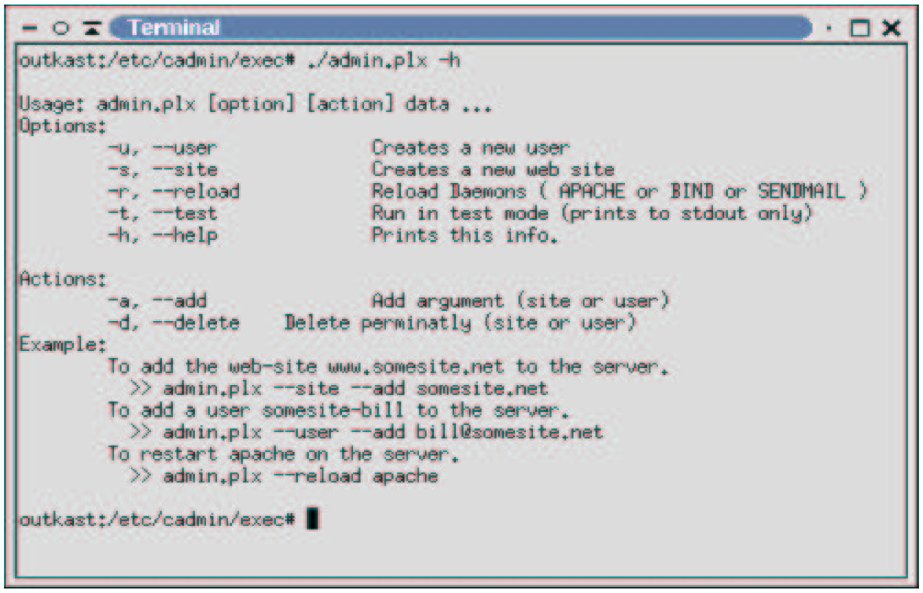
Also you will need to create a <service> tag for adminbot such as the following:

```
<service id=”adminbot”>
<host>adminbot.yourserver”</host>
<accept>
<ip>localhost</ip><port>5999</port><secret>secret</secret>
</accept>
</service>
```

Using CustAdmin

Scripts

To use the custom administration package you must have root privileges. All administration routines are accessed through the admin.plx script. A help menu (figure 1) is provided with the –help argument.



```
outkast:/etc/cadmin/exec# ./admin.plx -h

Usage: admin.plx [option] [action] data ...
Options:
        -u, --user              Creates a new user
        -s, --site              Creates a new web site
        -r, --reload            Reload Daemons ( APACHE or BIND or SENDMAIL )
        -t, --test              Run in test mode (prints to stdout only)
        -h, --help              Prints this info.

Actions:
        -a, --add               Add argument (site or user)
        -d, --delete    Delete perminatly (site or user)
Example:
        To add the web-site www.somesite.net to the server.
          >> admin.plx --site --add somesite.net
        To add a user somesite-bill to the server.
          >> admin.plx --user --add bill@somesite.net
        To restart apache on the server.
          >> admin.plx --reload apache

outkast:/etc/cadmin/exec# 
```

figure 1: (help menu)

Jabber component

To access the administration routines through a Jabber client start the /etc/cadmin/exec/custadmin.sh script in daemon mode (-d). Then connect to your jabberd server with the Jabber client of your choice. When sending messages send them to adminbot (or whatever you named your daemon in the service section of the jabber.xml file). You should now be able to send messages to and from the Jabber component.

Log

An activity log of all actions is kept for reference of changes and additions. The log resides in /etc/cadmin/log directory.

Notes

It is a good idea to periodically restart the cadmind using the /etc/cadmin/exec/custadmin.sh script. Also jabberd is periodically updated and new versions are released. It is also preferable to run the latest version of the jabberd server.