user-setup: A System for Custom Configuration of User Environments, or Helping Users Help Themselves

Richard Elling & Matthew Long – Auburn University

ABSTRACT

Large sites often have the problem of too much software, too many users, and not enough support staff. This paper describes user-setup: an easy to use system which allows users to customize their environment by selecting their preferred applications. The system is easy to administer, scales well, and does not limit advanced users. user-setup generates correct by construction C-shell "dot" files and customized OpenLook Window Manager menus. Thus, a user can have a personally customized environment that works without having to learn shell languages, X window manager menu languages, or text editors. The infrastructure is based upon the Modules package [Furlani91].

Introduction

Typical UNIX shells offer an often bewildering assortment of options and customizations. The shell is the most intimate interface between the system and its users. The shell is also one of the most complicated utilities to configure. Any mistake in the shell configuration files could render the shell completely useless. For the naive user, this is often catastrophic and contributes to the myth that UNIX is difficult to use. From the administrator's point of view, deciding on the perfect shell configuration becomes increasingly difficult as applications are added, upgraded, or phased out. For large sites the concept of editing every user's shell configuration file is ludicrous.

user-setup attempts to solve the problem by allowing the users to select applications for themselves. user-setup uses a simple menu interface with on-line help to guide the user through a list of available software packages. The user can review the packages and select those they wish to try. user-setup generates correct by construction shell configuration files that give users access to the applications they desire. Most importantly, users can run user-setup to safely change their environment at any time without having to consult the computer center support staff.

The Problem

Our problem is really one of resources: how to provide access to the large diversity of applications for a large number of users in a ubiquitous environment with a limited support staff.

Too Much Software

There is a **lot** of software in the universe. It is not unusual for a single site to have several word processing, publishing, math, or spreadsheet packages. In the engineering world, there are dozens of drafting, modeling, and analysis packages. Worse yet, many sites support several different window systems. Some of the packages only work in some of the window systems or perhaps only certain versions of the package under certain window systems. In many cases a new version of the application arrives which cannot immediately replace the existing version due to window system changes, operating system changes, or new features. So there may be several different versions of the same application which require support.

New Users

New users are perhaps the one true test of any system. New users often don't know what a shell is, what an editor is, what a "dot" file is, what a path is, or why they should bother with them. Often users are told to get an account and start using it. A typical new user will not know what applications are available or how to gain access to them. A simple typographical mistake in a shell configuration file could render the user's session unusable. Learning how to recover from a broken shell configuration should not be the second lesson taught to a new user. Frantic calls to the help desk are assured if new users are required to edit shell configuration files. user-setup solves this problem by creating correct shell configuration files for the user.

Guidance Conflicts

Guidance conflicts arise when several people give conflicting instructions to a naive user. For example, one instructor might require the users to run an initialization script to setup their environment. This script will likely conflict with the script that another instructor requires of the same users. All users are strongly encouraged to use user-setup and it has become the only help desk supported method of changing their environment.

Dot File Virus

There is a very nasty virus that can spread among users like wildfire: the "dot file virus." This virus is easily spread from user to user by such words as, "to run application ABC, just copy my dot file." Once a broken dot file virus starts, it can be very difficult to stop. Meanwhile, dealing with users bit by it can consume vast quantities of help desk time. In such cases, the user can always run user-setup to recreate a stable environment. As long as the modulefiles are properly maintained and updated, "dot" file viruses can be virtually eliminated. Now users say, "to run application ABC, just run user-setup."

Requirements

The design goal of user-setup is simple: provide a tool for users which will correctly configure their environment for whatever application they want to use. To reach this goal, several requirements were identified:

User-friendly

First and foremost, user-setup must be user-friendly. Not surprisingly, user-setup is one of the first applications run by new users. Since first impressions are important, we tried to make user-setup as friendly as possible. The idea was to make it so easy to use that users would be willing to run it again at any time to change their environment.

Idiot Proof

Obviously, user-setup must be idiot proof. No user-setup actions can affect the user's current environment. No "dot" files are modified until the user applies the environment changes. All new "dot" files are created in a temporary directory and are not copied into the user's home directory until they are completed constructed. All existing "dot" files are backed up before the new ones are copied. user-setup must not rely on the current user environment setup to run. Thus, if the user's current environment is totally trashed, user-setup must be able to recreate a stable, working environment.

Modules Infrastructure

user-setup uses the Modules package as its infrastructure. A modulefile is created for each application which contains the configuration information needed to run the application. The configuration information is typically comprised of changing the PATH, MANPATH, and other environment variables. The modulefiles are written in Tcl [Ousterhout90] which allows sophisticated conditionals and control over the environment changes. An added benefit of using the Modules package is that changes can be made to modulefiles which reflect changes in applications or application availability without forcing users to run user-setup again.

Low Maintenance

user-setup must be easy to maintain. The initial configuration of user-setup is relatively simple. The major portion of the maintenance involves the modulefiles. At least one modulefile must be created for each application. By convention, we include a separate file which contains the application description. This file can be displayed from the modulefile. For applications which can be started via olwm menus, two additional files are required which contain the menu information. All of these files are plain ASCII text with revision control by SCCS. Once the modulefile has been added to the modules directory structure it is ready to be accessed by user-setup.

Application Transition

A mechanism must exist to safely transition to new versions of applications as they become available and as the old versions are removed. There are several strategies that can be used by cleverly writing modulefiles. The modules directory structure is constructed so that each application is represented by a directory which has a modulefile for each version. For example, the X11 application directory may have modulefiles for R4 and R5. When R4 is deleted, the R4 modulefile could be rewritten to notify users of the change, run R5 instead, or even run user-setup.

Power Users

user-setup must support power users. Power users are traditionally very leery of anything which will touch their "dot" files. Power users also tend to make rather substantial changes to their "dot" files which they'd like to keep. user-setup provides a personal customization area in which a user may place shell commands. The personal customization area will be copied verbatim into the new .cshrc file. The personal customization area is located near the end of the .cshrc file so that any commands placed there will override the actions of any of the modules. The approach we take is that user-setup should be so good that power users will want to use it.

Lost Server Survival

A reasonable fail-safe for when modulefile servers are down must be included so that a user will not be thrown into a completely useless state.

Heterogeneous OS Support

user-setup must enable us to provide a consistent environment across major OS releases and hardware platforms. Applications which are not available for the current platform must gracefully inform the user without destroying the user's session.

Implementation

user-setup(1) is implemented as csh script. It provides an easy to use, character based, menu driven interface to the Modules package. Some modifications were made to the Modules package version 1.0 in order to support additional file types in the Modules directory structure. Once the user's environment has been created using user-setup, the user may make additional changes using module commands. Most users choose to use user-setup rather than module commands.

Configuration Files

user-setup was designed to be easily maintainable. Besides the modulefiles, only five configuration files are needed: a default .login file, prologue.cshrc, default customization, epilogue.cshrc, and a question order file.

Default .login File

The default .login file is copied into the user's home directory unaltered. The .login file is primarily used to start the appropriate window system. An environment variable is passed to the .login file to provide the command for starting the desired windowing system. A windowing system will only be started if the user is logged into the console. It is not anticipated that a user will modify the .login file so no provision for keeping changes is provided. The user's existing .login file will be moved to .login.bak.

.cshrc File Prologue

The prologue.cshrc file contains: any environment settings (e.g., umask, limit, other shell variables), the Modules package initialization, and the loaded modulefiles which are edited by user-setup. The prologue includes a fail-safe mechanism so that if no modulefiles are found, the user is not left in a helpless state.

Personal Customization Area

A default personal customization file is provided for initial account creation or first time users. If the user's current .cshrc file does not contain a personal customization area, then the default will be inserted. If a user has made a mistake in the personal customization area, the -nc option user-setup will force the default to be used.

The test for interactive shell is made in the personal customization area. This allows users to add shell commands for all shells as well as for interactive shells.

.cshrc File Epilogue

The epilogue.cshrc file contains any commands which will be executed after the personal customization. We place a network message of the day reader here.

Question Order File

The us2.order file is used to define a required order of initial selections. The first required selection chooses which flavor of UNIX is desired: BSD or System-V. The second required selection is the windowing system. This ordering assures that the other applications will have a known base to work from. This also simplifies the modulefile prerequisite list since each windowing system requires a UNIX base. Thus, a modulefile typically needs only to require the necessary windowing system to work properly.

Modules Package

To help ensure fail-safe operation when application servers are down, the Modules package is loaded in the /usr/modules directory on each workstation. The Modules package constructs a MODULEPATH which contains the names of directories which contain modulefiles. To ease maintenance and help ensure coherency, at least two directories will be specified in the MODULEPATH: /usr/modules/modulefiles and /usr/local/modules/*OS/-app-arch/release*. The /usr/modules/modulefiles directory contains the modules required to gain access to the UNIX commands resident on the local disk (e.g., /usr/bin.) The /usr/local/modules/*OS/app-arch/release* directory contains modulefiles which are applicable to the OS, application architecture, and OS release. The /usr/local directory is automounted from several servers and contains the majority of the modulefiles. If no /usr/local server is operational, the /usr/modules modulefiles will be sufficient to allow the user to login in a somewhat working state. Additional MODULEPATH directories may be added by a user, instructor, or workgroup and will be automatically accessible to user-setup.

A few minor modifications to the original Modules package were made. These changes dealt with the view of available modules. A program was written to recursively search a directory and list any files which had the Modules magic cookie. This addition allows arbitrary directory depth as well as the ability to place other files in the same directory structure. This also allowed us to consolidate all of the files relating to an application (e.g., olwm menu generation information) in one place.

The description of each application is kept in a file separate from the modulefile. By convention, the module display command will also display the contents of the description file. The convention allows a simple shell script to be written which generates a list of the descriptions of all available applications. This will also come in handy for future versions of user-setup.

Related Programs

There are several programs which can be used to modify a user's environment which was created by user-setup. *make-olwm-menu*(1) will generate a custom applications menu for the user based upon which modulefiles are selected. *default-printer*(1) will safely add or change the PRINTER environment variable. *prmap*(1) is an X-Window front end to *default-printer*(1) and also provides information on the location and type of network printers available.

Future Work

X-Window Interface

Development is underway on an X-window based version of user-setup. It is anticipated that the X version will be more popular than the terminal based version. The goal of the X version will be to provide a drag and droppable icons for each modulefile. A user will be able to peruse a collection of modulefile icons and select those that are desired.

Performance Enhancements

As the number of applications grow, the modulefile directory structure also grows. This affects the user-setup startup time since user-setup searches all of the directories in the MODULEPATH for modulefiles. A modulefile cache is being considered.

As part of the X-Window interface, a C++ class library for accessing modulefile description and dependency information is being developed. This development may lead to a C++ terminal version of user-setup which will replace the csh script.

More Shells

user-setup currently supports csh. tcsh is easily supported via user-setup since it is backwards compatible with csh. Bourne and Korn shell support will be provided in a future version.

Example Session

In the extended example shown below, the user would like to add a statistics package to their environment.

```
guest% user-setup

Welcome to user-setup!

user-setup is a menu driven program that helps you to create and modify
your environment.  You can run user-setup at any time to make changes in
your environment by adding or removing applications.  You may quit at any
time and the no changes will be made to your account.

To use user-setup, select the applications you wish to add or remove from
your account.  You will be shown a brief description of the application
and asked whether you wish to use it.  When you are finished selecting
applications, select "apply" from the main menu.

Searching for available Modules.....done.

Press [return] to continue.

Application Classes

          1 - apply
          2 - help
          3 - show
          4 - demo
          5 - math
          6 - misc
          7 - office
          8 - unix
          9 - win

Please select a Class of Applications [?,??,q]: math

Applications for math

          1 - done
          2 - mathematica
          3 - matlab
          4 - sas
          5 - splus
          6 - tksolver

Please select an Application [?,??,q]: 4

------- /usr/local/modules/sparc/SunOS/4.1.1/math/sas/6.07 -------

                       SAS version 6.07

The SAS system is an integrated system of software providing complete
control over data management, analysis, and presentation.  There are
```

```
two components of the SAS system available on the College of Engineering
Sun network: SAS/GRAPH and SAS/STAT.

Prequisites are (ORed):  win/openwin/3.0 win/openwin/2.0 win/X11/R4 win/X11/R5
Append /vol/sas to PATH
Append /vol/sas/utilities/man to MANPATH
-------

Do you want to use sas/6.07 [y,n,q] y

Applications for math

          1 - done
          2 - mathematica
          3 - matlab
          4 - sas
          5 - splus
          6 - tksolver

Please select an Application [?,??,q]: done

Application Classes

          1 - apply
          2 - help
          3 - show
          4 - demo
          5 - math
          6 - misc
          7 - office
          8 - unix
          9 - win

Please select a Class of Applications [?,??,q]: 1

Applying environment changes...

math/sas/6.07
unix/sysv
win/openwin/3.0

Do you wish to apply these changes ? [y,n,q] y

Finished making changes to your environment.  You may need to logout and
log back in for the changes to take affect.

You may also wish to create a custom menu based on your environment.  To
do so, run the command "make-olwm-menu" or select the "Make new menu"
entry from the "User services" menu in OpenWindows.

Press [return] to continue.
Cleaning up temporary files.
```

Now the user's environment is properly configured to run SAS. Now the user creates a custom menu for the new environment:

```
guest% make-olwm-menu

Hello.  I am ready to customize your OpenLook menu based upon the
modules you have loaded.  The menu will be custom tailored to the
applications you've selected.  We hope this will help make the
system easier to use.

Feel free to run this program any time to update your menu.

Do you want to update your menu now [y,n,q] y

Creating new menu...
```

```
No menu entry found for application: unix/sysv
Adding menu for application: math/sas/6.07
Adding default window system menu win/openwin/3.0.menu...
Renaming your existing .openwin-menu to /home/guest/.openwin-menu.bak
make-olwm-menu: removing temporary file.
```

The user's environment is now completely configured with a custom menu for running SAS.

Observations

user-setup version 1 was installed in September 1991. Version 1 was not menu based nor based upon the Modules package and resulted in a system which worked, but was not pleasurable to use nor maintain. user-setup version 2 has been in use since May 1992. Version 2 is menu driven, easy to use and maintain, and based on the Modules package. Approximately 2,500 users have used user-setup to gain access to more than 100 major applications in a ubiquitous environment. During this time we have observed a number of interesting phenomena.

Help Desk

First and foremost, users stop bugging the help desk asking how to get to applications. The standard help desk response is "run user-setup." This has significantly reduced the workload on the help desk.

Guidance

Instructors and workgroup leaders no longer need to provide elaborate instructions on how to modify user environments to gain access to required applications. Their standard response is also "run user-setup." This has proven to be an enormous benefit since it is typical that a user may be taking several classes and often the instructors would not take this into account with their notorious instructions.

Explorers

Users are able to easily discover and use available software. This has helped reduce the number of requests from the user community for new software which is functionally equivalent to existing software. This has also reduced the tendency for users to reinvent or download software which is already installed.

Users can try out new software. We are constantly installing new software and demonstration software. With a few edits we can create new modulefiles which will automatically be available via user-setup. Often creating the new modulefiles is easier than installing the software.

Training Gurus?

Many users don't know UNIX commands, what a "dot" file is, or even how to start applications from the command line. All they need to know is how to login, run user-setup, and run their applications from the custom menu. We believe this is a good thing.

OLWM Menu Madness

Some users are disappointed when their application can't be started from the custom OLWM menu. Unfortunately, some applications are just impossible or impractical to start from menus. For these, we don't provide any custom menu entries. Perhaps in the future we could pop up a brief "how to get started with the application" help screen.

New User Dilemma

There is a dilemma facing administrators when creating new accounts: should the user be forced to run user-setup, or should it be considered an option? If it is forced upon new users, they often are scared and confused and may easily give up. If it is always optional, what initial collection of modules is needed? Specifically, should a windowing system be part of the initial configuration? If so, which one?

Conclusion

user-setup enables users to change their environment easily. Applications can be added or deleted from the user's environment without the user having to learn editors or shell languages. New applications can be installed and made accessible to the users via user-setup by creating a simple modulefile for the application. This helps solve the problem of having too much software, too many users, and not enough support staff.

Availability

user-setup and its associated programs are available for anonymous ftp from ftp.eng.auburn.edu [131.204.10.91]. If you don't have internet access, contact the authors to arrange other media. All code is freely distributable.

Acknowledgments

We would especially like to thank John Furlani who wrote the Modules package. John's timely LISA-V paper allowed us to stop our reinvention of Modules early in the design cycle.

We would also like to thank the faculty, staff, and students of Auburn University who used user-setup version 1 and provided input to the Modules based version 2.

Bibliography

[Furlani91] Furlani, John L., *Modules: Providing a Flexible User Environment*, 1991 USENIX Large Installation System Administration V Conference Proceedings.

[Ousterhout90] Ousterhout, John K., *Tcl: An Embeddable Command Language*, 1990 Winter USENIX Conference Proceedings.

Author Information

Richard Elling is the Manager of Network Support for the College of Engineering at Auburn University. He graduated from Mississippi State University with a BSEE in 1986. He has 11 years of experience in writing and maintaining computer aided engineering tools. His current mission is to develop an awesome engineering environment at Auburn. Reach him via U.S. Mail at Auburn University, Engineering Network Services, 103 L-Building, Auburn University, AL 36849. Reach him via telephone at (205)844-2280. Electronic mail sent to Richard.Elling@eng.auburn.edu is preferred.

Matthew Long is graduating senior at Auburn University. He is available for employment 1/1/93. Reach him via U.S. Mail at Matthew Long, 110 Cedar Crest Circle, Auburn, AL 36830. Reach him via email at Matthew.Long@eng.auburn.edu.