The following paper was originally published
in the Proceedings of the
Tenth USENIX System Administration Conference (LISA X)
Chicago, IL, USA, Sept. 29 - Oct. 4, 1996

UNIX Host Administration in a Heterogeneous Distributed Computing Environment

Gregory S. Thomas, James O. Schroeder, Merrilee E. Orcutt,
Desiree C. Johnson, Jeffrey T. Simmelink, and John P. Moore
Pacific Northwest National Laboratory

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

UNIX Host Administration in a Heterogeneous Distributed Computing Environment

Gregory S. Thomas, James O. Schroeder, Merrilee E. Orcutt, Desiree C. Johnson, Jeffrey T. Simmelink, & John P. Moore
– Pacific Northwest National Laboratory

ABSTRACT

Managing a computing environment consisting of hundreds of systems from multiple vendors and hundreds of users dispersed over several buildings of a research campus is a complex task. We developed a data model for representing pertinent information about our computing environment and implemented this model in a database that tracks equipment, users, and accounts. We created software that automates much of the data collection, maintains the database, and provides easy access to commonly requested information. For example, a user can find out which hosts he or she has an account on, or learn the hardware and software configuration of any host on the local network. We also created software that automates many of the tasks involved in account creation, deletion, and modification, which reduces the burden on support staff by allowing privileged users to manage accounts on their own systems.

Introduction

This paper describes a UNIX Host Administration (UHA) system that is being developed to help manage a scientific computing environment.

The Environmental Molecular Sciences Laboratory (EMSL) is a national collaborative research facility located at the Pacific Northwest National Laboratory (PNNL). The EMSL is currently under construction and is scheduled to become operational in late 1997, but its computing environment is already in production use by staff working in interim facilities.

The research mission supported by the EMSL computing environment is characterized by the frequent introduction of new hardware and operating software. The integration and stabilization of these systems is often a challenge. As the research community supported by EMSL computing continues to extend outward to include more international collaborators, careful management of accounts becomes critical.

There are two primary goals of the EMSL UHA system:

- Simplify common administrative tasks so they can be performed by clerical staff and/or users, thereby focusing limited system administration resources on complex integration issues.
- Improve the efficiency and integrity of processes that track users, accounts, and equipment, as needed to meet security requirements and good business practice standards.

Our approach to meeting these goals has been to implement a simple but effective system that includes a central data repository managed by a relational database and a suite of tools that run on the target hosts and interact with the database to perform tracking and administration services. The simplicity of the system design and extensive use of perl [18] should minimize the effort required to port the code to clients running new operating system releases.

The focus of the initial implementation has been on automated data collection, data reporting, and account administration, but the infrastructure is now in place to rapidly deploy other functions as needed.

The tools implemented to date provide several high-payback benefits:

- Account administration procedures can be delegated to users and clerical staff.
- Account administration procedures are consistent across platforms.
- Policy decisions are codified.
- Tracking of users, accounts, and equipment is centralized.

The following sections of this paper discuss the computing environment, system architecture, database, account management software, related work, current status, and future directions.

Computing Environment

The EMSL's computing environment consists of a wide variety of computers and network equipment, ranging from parallel supercomputers to high performance workstations to Macs and PCs. There are roughly 840 computers on site, including 380 Macs, 260 PCs, 160 UNIX workstations, and 40 X terminals.

UNIX workstations from most of the major vendors are present. From each vendor, there are different models of computers and different versions of operating systems. An effort to consolidate the UNIX environment (i.e., reduce the number of hardware and operating system variants) is underway, but the environment will remain heterogeneous.

Most of the UNIX workstations run AFS [15], which provides a shared file space. Login authentication on many workstations is provided by the Kerberos [9] component of AFS. The goal is to move the passwords for all non-privileged accounts to the Kerberos registry.

EMSL computer users include PNNL staff members associated with the EMSL, post doctoral researchers, other visiting scientists, and remote collaborators from the international scientific community. At this writing there are approximately 300 users of EMSL UNIX systems, evenly divided between PNNL and non-PNNL staff members. When the EMSL becomes operational, it is projected that there will be 300 PNNL users and 2000 non-PNNL users.

EMSL policy is to manage accounts on a per-computer basis in order to meet security requirements. The current UNIX users have about 500 accounts; there are 3400 instances of these accounts altogether.

Account administration in the EMSL is a continuous process, characterized by steady activity without the peaks typical of the university environment. It is important that new accounts are opened in a timely manner, and, for foreign national users, that accounts be created only on approved machines. Equally important is that all instances of a user's account(s) be disabled when the user's association with the EMSL concludes.

System Architecture

This section gives a brief overview of the system architecture for the suite of UNIX Host Administration (UHA) tools developed for the EMSL. A simplified view of the UHA system architecture is shown in Figure 1.

The UHA suite provides a semi-automated method for managing accounts on heterogeneous distributed UNIX hosts.

The UHA suite uses a central database as the controlling data structure for users and hosts, and a host-resident UNIX data structure (e.g., `/etc/passwd`) as the source of account data.

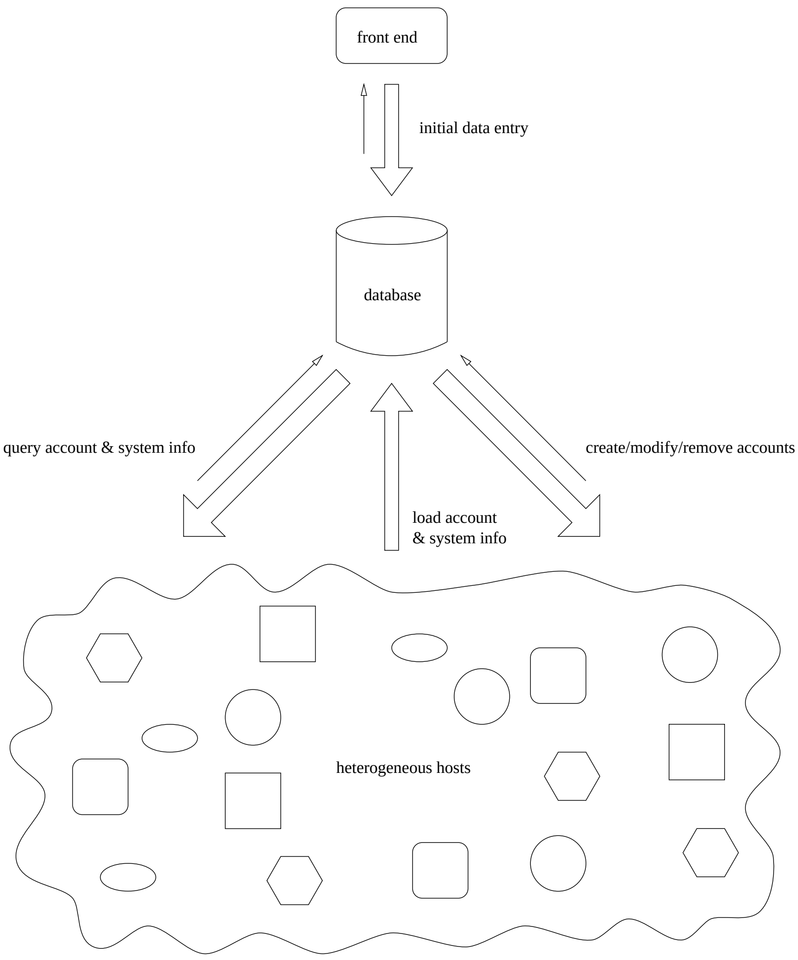

Figure 1: UHA System Architecture

The database is maintained by direct data entry through a spreadsheet interface, and by perl scripts run on distributed hosts to extract system and account information with which to update the database. Each time a new user is registered, a user profile is entered into the database.

Account information on distributed hosts is managed incrementally. The administrator or user invokes a script on the target host to create a new account. The script extracts user information from the database and executes architecture specific commands to add the account to the appropriate host-resident data structure (e.g., `/etc/passwd`).

Each host periodically uploads its system information (e.g., hardware model, operating system version, memory size) and information about active accounts into the database. Accurate system and account information for each host is thus maintained automatically, without manual updating of the database.

Commands are provided that allow users to query the database from the UNIX command line. Users can find out which accounts they own, which hosts the accounts reside on, what the hardware and software configuration of a host is, and so on.

Database

To help manage the EMSL computing environment, the Operations, Administration, and Support Information System (OASIS) database was created. The following sections describe OASIS: the data model, the software architecture, the implementation, server operation, writing data, and reading data.

Data Model

The OASIS data model contains approximately fifty tables. In this paper, the following four tables are of primary interest:

- `equipment`: hardware configuration, software configuration, owner, location, and inventory information for a computer
- `users`: name, identification information, location, and organizational affiliation for a person
- `user_accounts`: account name, account information (e.g., user identification (UID) number), and identification information for a person's UNIX account
- account : account information for an instance of a person's account on a particular host

A simplified version of an entity-relationship diagram for the subset of the OASIS schema corresponding to these four tables is shown in Figure 2. A single arrowhead denotes a one-to-one relationship, e.g., there is one `equipment` record associated with a given `account` record. A double arrowhead denotes a one-to-many relationship, e.g., there are many `account` records associated with a given `equipment` record.

There is one `equipment` record per computer, and one `users` record per person. Often there is only one `user_accounts` record per `users` record, but there can be more than one. Multiple `account` records may be associated with a single `user_accounts` record, one for each host on which the account exists.

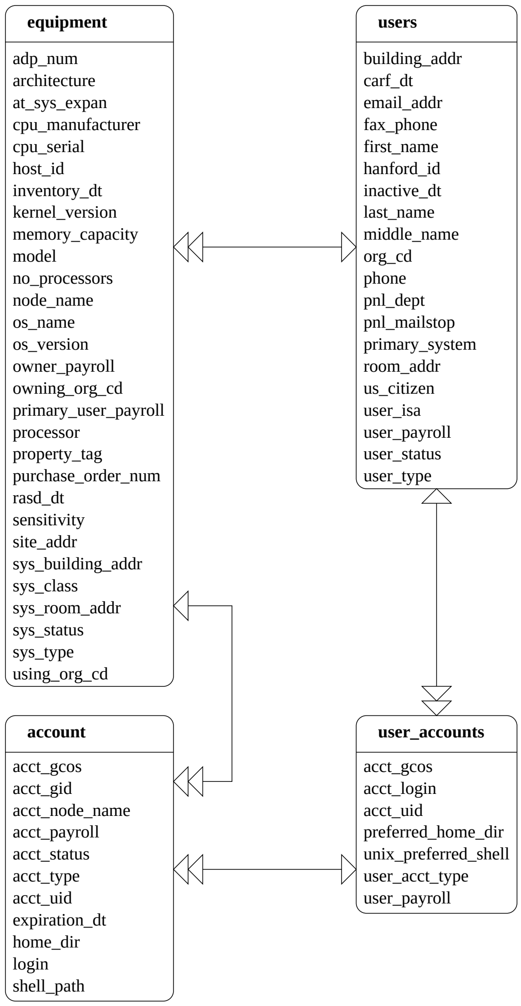

Figure 2: Partial OASIS Schema

As an example, a user named Robert might have an account named **bob** for personal use and a different account named **tex** for doing TeX administration. If the **bob** account can access hosts **earth** and **mars** and the **tex** account can access hosts **mars** and **pluto**, there are two `account` records associated with the **bob** `user_accounts` record and two `account` records associated with the **tex** `user_accounts` record. Figure 3 shows graphically how records in OASIS tables model this situation.

Software Architecture

The database server runs on an EMSL workstation (the *database host*). Access to the database from other hosts is provided by the database vendor's client-server software libraries. The client libraries are installed in the shared AFS file space for the other hosts to access. Most of the UHA software runs on the hosts and accesses the database via the client libraries. Some of the UHA software for database maintenance runs directly on the database host and uses an alternate interface (*isql* [13], the database vendor's command line SQL [14] utility) to the database.

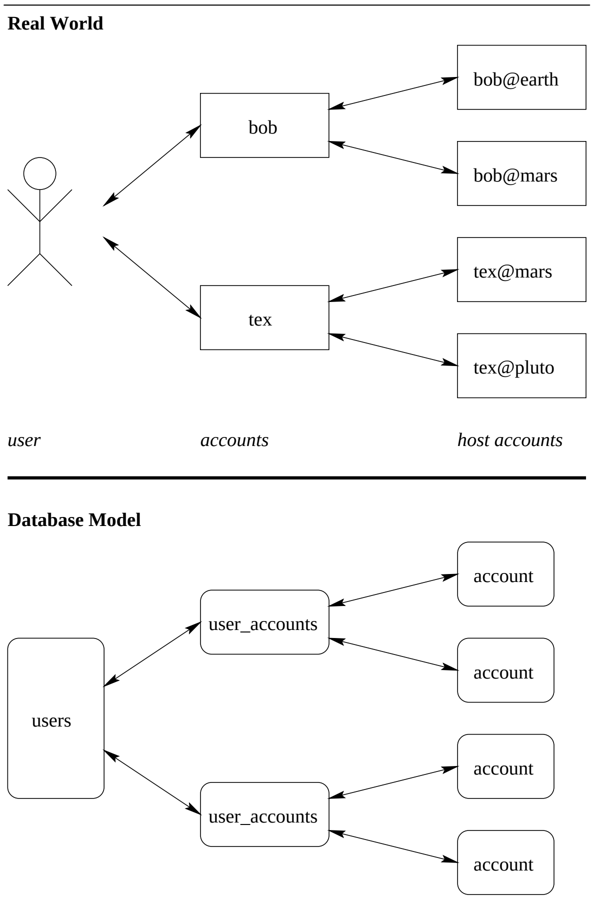

Figure 3: Account Representation in OASIS

A typical UHA application is written in perl. The application reads data from or writes data to the database via the sybperl [6] software package; sybperl provides extension modules that allow perl applications to call Sybase client-server library functions. This three-tiered layering of software is depicted in the client host boxes in Figure 4.

Most of the UHA applications do not invoke sybperl functions directly. Instead, they call UHA database library subroutines, which are part of a larger group of UHA library modules that perform tasks such as the following:

- Encode arrays into strings, decode strings back into arrays
- Compare, compute differences, search, merge, slice, sort, and perform other operations on one-dimensional, two-dimensional, and associative arrays
- Convert between byte prefixes
- Mimic the UNIX crypt program
- Authenticate jobs and users
- Parse flat files into database structures
- Transparently write messages to multiple destinations, e.g., a log file and the standard error output
- Extract host information
- Generate standardized log file entries
- Generate standardized input prompts
- Generate unique names for temporary files
- Manage file locks

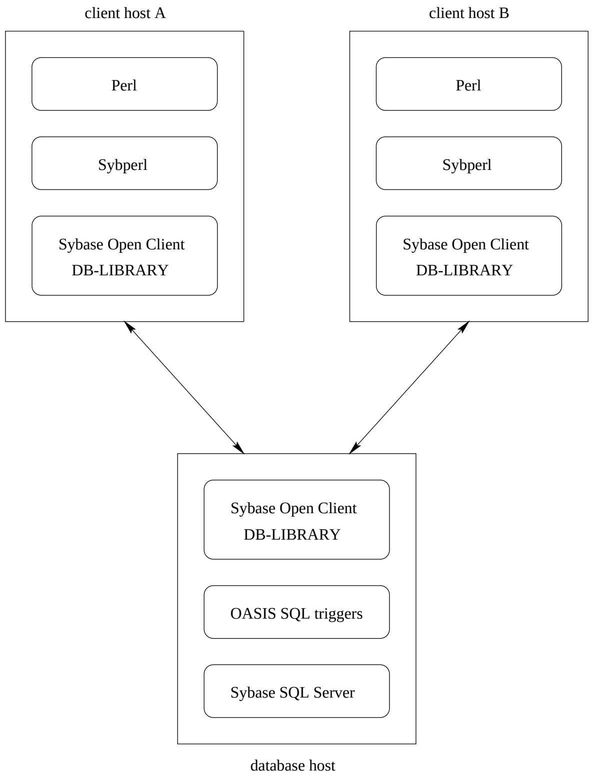

Figure 4: UHA Software Architecture

These perl modules facilitate code reuse across UHA applications and make it easier to extend existing applications and develop new ones.

The UHA library module that provides database access offers two avenues. The first allows access from local or remote hosts and employs sybperl and the Sybase client-server library functions. The second allows access only from the local host (the database host) and employs *isql*. Code implementing the first avenue is separated into three layers. The lowest layer is a wrapper around the sybperl functions (e.g., login to the database). The middle layer adds authentication and database connection opening/closing to support individual queries or sequences of queries (e.g., execute an SQL statement and return the result). The highest layer performs complete operations ranging from the simple (e.g., return the record matching a given key) to the complex (e.g., dynamically query the database for its field definitions and truncate candidate data values to the lengths defined for the corresponding fields).

In addition to the UHA perl code, a separate body of SQL code is stored in the database itself. This SQL code implements the OASIS triggers and stored procedures. A stored procedure is a named collection of SQL statements stored in the database. A trigger is a type of stored procedure that is automatically invoked by the database when a command tries to modify a specified table or column. As shown in Figure 4, the OASIS triggers appear as a layer immediately above the database server.

Implementation

The OASIS database currently runs on a Sybase SQL Server 10.0.2 [12], hosted on a Sun SPARCstation 20 running Solaris 2.4.

Access to the database is provided by command line scripts and a spreadsheet front end. The scripts are written to run on perl 5.001m [17], sybperl 2.0, and Sybase Open Client DB-Library 10.0.*x* [11].

A prototype graphical front end was written using a cross-platform Mac and UNIX tool, but has been discarded in favor of a (future) web-based solution. As an interim measure, Microsoft Excel is employed to provide a spreadsheet style interface (via Microsoft Query [5]) to the database that is familiar to clerical staff who maintain the user and equipment data. Use of this interface requires knowledge of the database structure, but it was chosen because it was easy to implement using readily available software.

Server Operation

A database server requires frequent attention. To reduce the operational burden, scripts were written to perform the following tasks:

- Back up the database
- Perform consistency checks on the database
- Start the server
- Stop the server
- Query the database for various information
- Compare records in the database against records in a file

Currently, the transaction logs of the database are dumped twice daily, a full database dump is performed nightly, and a suite of consistency checks is run nightly.

Automated filtering is performed on the output of the database back up and consistency check scripts. These scripts are normally run periodically via *cron*, which produces electronic mail messages containing output from the scripts. Rather than having a person scan several messages each day for problems, procmail [16] was used to implement a smart mail filter that screens normal messages and forwards only messages that indicate problems.

Writing Data

Many of the mechanisms for maintaining data integrity are incorporated into the database in the form of triggers and stored procedures. OASIS will not allow the deletion of a `users` record, for example, if that `users` record is still the owner of an `equipment` record, because one of the OASIS triggers forbids it. Another example is the stored procedure that generates a unique UID when a new user is registered in the database.

Implementing the integrity mechanisms once in the database saves the effort of reimplementing them for each different interface (e.g., perl scripts and a spreadsheet front end) and minimizes the potential for corruption by centralizing the integrity mechanisms. Furthermore, security is increased because the triggers cannot be bypassed; they can only be disabled by a database user who has owner or root privileges.

The following scripts load data into the database:

- *load_accounts* – upload information about the accounts on a host
- *load_sysinfo* – upload information about the hardware and software configuration of a host, using the *sysinfo* program [3] if it is available for the platform

They are run periodically on each host via a *cron*-based Automated Host Management System (AHMS), which is a PNNL-developed system complementary to UHA.

There is also a *bulk_load* script that imports data from a file into the database, which is useful for initially loading tables with data from a spreadsheet or other database.

Reading Data

To avoid forcing users to learn a database query language (e.g., SQL) or the details of the database schema, the following scripts retrieve commonly requested data from the database:

- *show_acct* – list all instances of an account
- *show_profile* – list the parameters of an account
- *show_sysinfo* – list information about the hardware and software configuration of a host
- *show_user* – list demographic information about a user

Account Management Software

The following scripts simplify common account management tasks:

- *host_acct_create* – create a new account on a host
- *host_acct_modify* – modify an existing account on a host
- *host_acct_remove* – remove an account from a host

These scripts are used by system administrators and privileged users, and offer several benefits. Enabling privileged users to manage accounts on their own systems is a win-win situation because it offloads work from the often overworked support staff and it eliminates the delay between when a privileged user (who is often the owner or primary user of a machine) wants an account change and when that change occurs. Having scripts that encapsulate the sequences of commands necessary to manage accounts is advantageous for several reasons:

- Administration procedures are consistent across platforms, no matter who performs them.
- Administration procedures are easier to perform because the scripts hide platform differences and execute procedures completely and correctly, thus saving time for experienced administrators who already know the underlying commands, and enabling novice administrators who are unfamiliar with the underlying commands to easily perform procedures they might otherwise struggle with.
- Numerous policy decisions are codified, e.g., foreign nationals must have an authorization record in the database before being granted an account on a particular host.

Before invoking a UHA account management script, a user must obtain root privileges, which is accomplished by running *sudo*. The *sudo* program, which is a modified version of the standard UNIX *su* program, uses access control lists to grant privileges based on user name, command name, and host name.

Related Work and Products

The UHA architecture is powerful but inherently simple. It is built on the concepts developed in similar packages and products. Several such systems have been described in the literature.

Moira [8], from MIT Project Athena [2], is a service management system that feeds information from a central database to other Athena servers. Moira is the architectural genesis for UHA and possibly other systems that use a centralized database as the definitive source of configuration and control data. However, redeployment of Moira without other Athena services did not appear to be a pragmatic approach, and a full Athena implementation was more pervasive than desired.

Systems that perform comprehensive management of host configuration and accounts include GeNUAdmin [4] and lcfg [1]. These systems use information maintained in a central data repository to build control structures that are then loaded on the client. In contrast, UHA's account management commands are interactive and incremental, targeted at the primary user(s) of a host who uses *sudo* to execute the commands with root permission. UHA tracking functions work in the opposite direction, periodically collecting host configuration and account information into the database.

UHA account management commands are intended for low volume transactions where immediate results are more important than the ability to manage a large number of accounts automatically. This difference in purpose results in a simpler architecture than that of other systems such as UAMS [10] and AGUS [7] that are capable of the high volume automated account processing required in a university environment.

High equipment turnover through vendor upgrade programs is typical in the EMSL in order to maintain a competitive research environment. The result is that many of our critical resources are running new (often beta) operating system releases. When cross-platform administration tools are provided by vendors as part of the operating system release, investment in commercial software that performs administration services on distributed systems will be practical in our environment. Considering products that can help administer the EMSL Distributed Computing Environment (DCE) cell now being deployed, maturing standards and the focus on cell management rather than host management bodes well for the use of comprehensive and supported commercial cell administration software.

Current Status

Components of the software described herein are in various stages of development. The OASIS schema, triggers, stored procedures, and server maintenance software have been in production use for about a year. The UHA load and show scripts have been deployed on SunOS and Solaris platforms for about six months; ports to IRIX and AIX are underway. The UHA account management scripts are currently being tested on SunOS and Solaris platforms; ports to IRIX and AIX will follow.

Future Work

Work is now underway to augment the spreadsheet front end to the database with a web-based interface. The UHA perl library modules for database access developed in the first release of the system make this task relatively easy.

The database contains the information necessary to create a useful set of electronic mail aliases for the EMSL mail hub. A similar service is provided by Moira in the Athena environment. Again, the UHA infrastructure now in place will simplify this task.

The OASIS database is being extended to manage more of the information needed for EMSL operations, including staff demographics, operating procedures, training and safety programs, publications data, network cable schedules, and so on. As this integration effort advances, data duplication problems diminish and direct access to the database for host administration functions becomes an even greater advantage.

In general, future UHA work will focus on development of interactive tracking and administration tools that can be placed in the hands of users and clerical staff. Administration services that control the configuration of multiple hosts are the purview of the experienced system administrator. These will be handled using the EMSL AHMS software, possibly extended to draw configuration information from the OASIS database in a manner similar to other systems referenced in the Related Work and Products section.

Summary

This paper described software and a database developed at the EMSL to help manage its computing environment of heterogeneous UNIX workstations distributed across the PNNL campus.

The architecture of the UHA system is very simple. A relational database serves as the central information repository. Initial data entry is done through a spreadsheet interface or by importing flat files. Host and account data is kept current by perl scripts running periodically on the distributed hosts, automatically scheduled by a *cron*-based mechanism. Account management is assisted by scripts that retrieve account template information from the database and execute platform-specific commands to perform the desired operation.

Availability

The UHA system is not currently packaged for distribution. Contact the first author (address given below) if you wish to inquire about gaining access to the source code.

Acknowledgements

This work was funded by the Environmental Molecular Science Laboratory construction project at the Pacific Northwest National Laboratory (PNNL). PNNL is operated for the U.S. Department of Energy by Battelle under Contract DE-AC06-76RLO 1830.

Author Information

Greg Thomas holds an M.S. in computer science from the University of Washington and a dual B.A. in computer science and statistics from the University of California, Berkeley. While in school, he held internship positions with Bell Communications Research, GTE Laboratories, Lockheed Missiles and Space Company, and Sun Microsystems. He joined the Pacific Northwest National Laboratory as a research scientist in 1991, where he has developed software for projects ranging from a graphical interface for human genome databases to a portable multi-language dynamic memory allocation library for computational chemistry applications. He can be reached at Pacific Northwest National Laboratory, Mail Stop K7-22, P.O. Box 999, Richland, WA 99352; or electronically at gs_thomas@pnl.gov.

Jim Schroeder has been involved with computer systems integration projects at the Pacific Northwest National Laboratory for over twenty years as a technical contributor and project manager. He can be reached at jo_schroeder@pnl.gov.

Merrilee Orcutt holds an M.S. in information sciences from Hawaii Pacific University and a B.A. in computer science from the University of California, San Diego. She joined the Pacific Northwest National Laboratory as a research scientist in 1993, and is currently the software engineering task leader for the Extensible Computational Chemistry Environment project. She can be reached at me_orcutt@pnl.gov.

Desiree Johnson graduated in 1991 from the University of Minnesota with a B.S. in computer science. She spent three years at the Pacific Northwest National Laboratory in the Applied Physics group working with heterogeneous database systems. She currently works in the Information Resource Management group at Norwest Mortgage in Des Moines, Iowa, supporting VLDB systems. She can be reached at Desiree.Johnson@nmb.norwest.com.

Jeff Simmelink graduated in 1990 from Eastern Washington University with a B.A. in computer science and mathematics. He spent five years at the Pacific Northwest National Laboratory as a UNIX systems administrator. He is currently a UNIX systems analyst at Itron, Inc. in Spokane, Washington. He can be reached at jeff.simmelink@itron.com.

Jack Moore left the Pacific Northwest National Laboratory in 1995 to join Itron, Inc. in Spokane, Washington. Reach him at jack.moore@itron.com .

Bibliography

[1] Paul Anderson. Towards a High-Level Machine Configuration System. In *Proceedings of the Eighth Systems Administration Conference*, pages 19-26. USENIX Association, September 1994.

[2] George A. Champine. *MIT Project Athena: A Model for Distributed Campus Computing*. Digital Press, Bedford, MA, 1991.

[3] Michael A. Cooper. *Sysinfo Reference Manual*. University of Southern California, Los Angeles, CA, version 3.1.3 edition, March 1996.

[4] Magnus Harlander. Central System Administration in a Heterogeneous Unix Environment: GeNUAdmin. In *Proceedings of the Eighth Systems Administration Conference*, pages 1-8. USENIX Association, September 1994.

[5] Microsoft Corporation. *Microsoft Query User's Guide*.

[6] Michael Peppler. *Sybperl Reference Manual*. ITF Management SA, Geneva, Switzerland, version 2.0 edition, October 1995.

[7] Paul Riddle, Paul Danckaert, and Matt Metaferia. AGUS: An Automatic Multi-Platform Account Generation System. In *Proceedings of the Ninth Systems Administration Conference*, pages 171-180. USENIX Association, September 1995.

[8] M. Rosenstein, D. Geer, and P. Levine. The Athena Service Management System. In *Proceedings of the USENIX Winter Conference*. USENIX Association, February 1988.

[9] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the USENIX Winter Conference*. USENIX Association, February 1988.

[10] Roland J. Stolfa. Simplifying System Administration Tasks: The UAMS Approach. In *Proceedings of the Seventh Systems Administration Conference*, pages 203-208. USENIX Association, November 1993.

[11] Sybase, Incorporated, Emeryville, CA. *Open Client DB-Library/C Reference Manual*, 1994.

[12] Sybase, Incorporated, Emeryville, CA. *SQL Server Reference Manual*, 1994.

[13] Sybase, Incorporated, Emeryville, CA. *SQL Server Utility Programs for UNIX*, 1994.

[14] Sybase, Incorporated, Emeryville, CA. *Transact-SQL User's Guide*, 1994.

[15] Transarc Corporation, Pittsburgh, PA. *AFS User's Guide*.

[16] Stephen R. van den Berg. *Procmail Reference Manual*, version 2.81 edition, February 1993.

[17] Larry Wall. *Perl Reference Manual*, version 5.001m edition.

[18] Larry Wall and Randal L. Schwartz. *Programming perl*. O'Reilly and Associates, Inc., Sebastopol, CA, 1991.