PACE

Applied Technology, Inc.

White Paper

UNIX Resource Accounting

"A piece of the IS puzzle"

By: Michael A. Richardson

Table of Contents

Introduction

Over the last several years, the use of UNIX servers for mission critical IS applications has risen dramatically. Many companies are using UNIX for new systems or are migrating applications from mainframes to UNIX servers such as Sun Microsystems' SPARC servers or Hewlett-Packard's HP 9000 Series servers. The IS manager is now surrounded by several different types of hardware. The mainframe is still there, although its role is changing; and now UNIX servers are being rapidly acquired and deployed for the latest 'cutting-edge' applications. The management tools available for UNIX are limited in comparison to the robust tools available in Z\os. Furthermore, very few system management tools for UNIX easily integrate with their Z\os equivalents.

One area of systems management where this scenario is readily apparent is in resource accounting. Traditionally, IS managers recover the cost of mainframe hardware and software by charging the users of the system based on how much they use the system. Many would like to do the same for the new UNIX servers. Although UNIX has some facilities for system accounting, they have some limitations. Also, many question the usefulness of this data in the face of emerging technology such as distributed and Client-Server computing.

The purposes of this white paper are:

- To describe the resource data provided by UNIX and the files or commands which provide the data.
- To describe ways to enhance this data to make it more useful for chargeback purposes.
- To describe how this data can be part of a comprehensive chargeback scheme.
- To describe how a software package, SCSU from PACE Applied Technology, Inc., provides the means for UNIX utilization data to be incorporated into enterprise wide chargeback.

Background — A Brief History of UNIX

All of the various flavors of UNIX available today have their roots in two main versions of UNIX: AT&T UNIX (also known as System V) and Berkeley UNIX (also know as BSD). AT&T Bell Labs first developed UNIX in 1969 for internal use. The initial version was written by Ken Thompson in assembly language on a DEC PDP-7. In the early 1970s, Thompson, along with Dennis Ritchie rewrote UNIX in C, a newly invented portable systems programming language. In 1976, Bell Labs released the Sixth Edition of UNIX, also known as V6. Although now available to universities at little or no cost, UNIX did not catch on until Version Seven was released three years later. Bell Labs subsequently released System III and System V before selling off its UNIX operations to Novell in 1992. BSD UNIX began in 1977, when a research group from the University of California, Berkeley licensed V6 from AT&T and started making modifications.

Through the chaos of these different versions of UNIX, workstation vendors such as Sun Microsystems, Silicon Graphics and DEC began making their own ports of UNIX to their hardware. Although most of these ports were based on AT&T UNIX, each has its own peculiarities. It is beyond the scope of this paper to go into detail about the heritage of each of these UNIX variants. To get an idea of the complexity of the UNIX family tree, refer to figure 1. The most widely used UNIX variants are based in System V, but borrow heavily from BSD in certain areas (i.e. printing and networking functions).

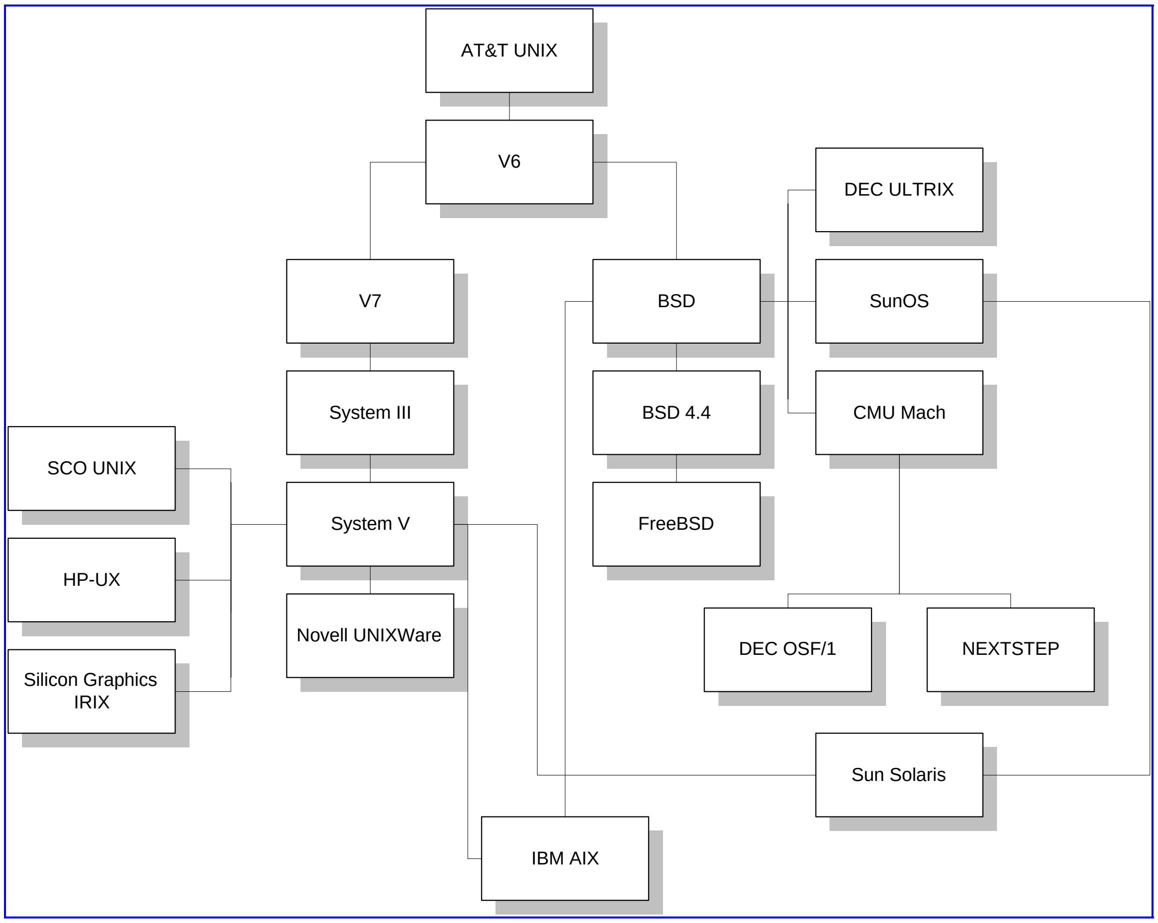

Figure 1

UNIX "Family Tree"

There are many important events in the evolution of UNIX which have made it one of the most popular operating systems in the world. The first was the re-implementation of UNIX in C. This made UNIX not only portable to anything from PCs to mainframes, but also easily extensible. If the version of UNIX you were using did not have a certain desired feature, it could often be implemented with a C program. The second was the virtually free distribution of UNIX to universities. Future systems designers and programmers would become quite familiar with this operating system and remember it after moving into the workplace. These two occurrences directly led to the third, the porting of UNIX to the latest generation of workstations utilizing hot technology such as RISC processing. UNIX continues to be the operating system of choice because of it's scalability, flexibility (via C), and features previously only available with larger systems. These features include: multi-user support, multitasking, job scheduling and control, and advanced networking via TCP/IP.

UNIX Accounting Data

The first step in attempting chargeback for UNIX is to find the data that is available. This data can be divided into four groups: process records, connect records, disk utilization, and printer utilization. In addition, System V UNIX also provides the ability to summarize process, connect and disk utilization into total accounting records.

Process Records

Both System V and BSD variants of UNIX can produce records for process accounting and connect accounting. Process accounting records are defined by the acct struct listed in the file /usr/include/sys/acct.h. When process accounting is activated in the system, the kernel writes a record in the format defined by the acct structure to a log file (usually /usr/adm/pacct) for each process run. The acct structure has the following fields:

Field Name	Field Type	Description
ac_flag	char	accounting flag
ac_stat	char	exit status
ac_uid	ushort	user id
ac_gid	ushort	group id
ac_tty	dev_t	control tty number
ac_btime	time_t	beginning time
ac_utime	comp_t	user CPU time in clock ticks
ac_stime	comp_t	system CPU time in clock ticks
ac_etime	comp_t	elapsed time in clock ticks
ac_mem	comp_t	memory usage
ac_io	comp_t	bytes transferred
ac_rw	comp_t	disk blocks read or written
ac_comm	char[8]	command name

Table 1
acct Structure Fields

The ac_flag field is set to one of a set of values describing special conditions which occurred during the process (i.e. process uses superuser privileges). The user CPU time is the amount of time the CPU spends running the code of the process. The system CPU time is the amount of time the CPU spends running system tasks for the process. These two fields, along with elapsed time, are given in clock ticks. The amount of clock ticks per second can vary from 50 to 100. The ratio for a specific system is defined as a constant and can be accessed via the sysconf system call.

Connect Records

Connect records are defined by the utmp struct listed in the file /usr/include/utmp.h. The kernel writes records in the format defined by utmp to a log file (usually /etc/wtmp) for various system events, including user login, user logout, ftp connections and system boots. The utmp structure has the following fields:

Field Name	Field Type	Description
ut_user	char[8]	user login name
ut_id	char[4]	line id
ut_line	char[12]	terminal device name
ut_pid	short	process id
ut_type	short	type of entry (login, logout, system boot, etc.)
ut_exit.e_termination	short	termination status
ut_exit.e_exit	short	exit status
ut_time	time_t	time stamp

Table 2
utmp Structure Fields

Disk Utilization

There are no disk accounting records per se, but disk utilization can be tracked by processing the output of one of the UNIX disk scan utilities. The two utilities which are most suitable for this task are the quot and ls commands. The quot command can list, for each disk device, the users who own space on the device and the amount of space in disk blocks.

```
/dev/root:
526148    root
121842    bin
44760     mrich
 4354     mmdf
 4250     scsu
 3638     uucp
 3616     games
 3198     sys
 3118     adm
 1142     dos
 1052     audit
  178     auth
   98     sysinfo
   68     lp
    6     debbie
    6     donna
    6     george
    6     guest
    6     network
    6     oscar
    4     cron
    4     ingres
    2     nuucp
    2     skeye
```

Figure 2
Sample quot Output

The ls command can be used to list all files in a file system, along with the size of each file, and the name of the user and group that own the file.

```
total 4786
-rw-------   1 root     root          15 Dec 14  1991 .mailrc
-rw-------   1 root     root         891 Jan 24 13:26 .profile
-rw-r--r--   1 root     other         13 May 25 14:36 .xinitrc
-rw-r--r--   1 bin      other      17982 Sep 17  1993 COPYING
drwx------   4 root     other         64 Oct 24  1994 Mail
-rw-r--r--   1 root     other       3217 Sep 28  1994 Xconfig
drwxr-xr-x   2 bin      bin         2416 May 15 12:08 bin
-r--------   1 bin      bin        78525 Mar 15  1993 boot
drwxr-xr-x   2 root     other         32 Apr 07 11:36 clipdir
drwxr-xr-x   8 bin      bin         4400 Jun 21 16:35 dev
-r--------   1 bin      bin          584 Dec 15  1991 dos
drwxrwxr-x  19 bin      auth        4416 Jun 30 01:30 etc
drwxr-xr-x   2 root     other         32 Jun 09  1992 install
drwxrwxr-x   5 bin      bin         2656 May 03 11:03 lib
-rw-------   1 root     other      23640 Jun 29 10:22 mbox
-rw-------   1 root     other     126769 Jun 05 11:01 mbox.1
drwxrwxrwx   2 root     root          32 Jun 09  1992 mnt
drwxrwxrwx   3 root     root          48 Jun 09  1992 opt
drwxr-xr-x   2 root     other        176 Aug 12  1994 rtr
-rwx------   1 bin      bin         2914 Mar 15  1993 sfmt
drwxr-xr-x   2 bin      bin          128 May 03 11:00 shlib
d--x--x--x   6 bin      bin           96 Jun 09  1992 tcb
drwxrwxrwt   3 sys      sys         8784 Jun 30 01:30 tmp
drwxr-xr-x   2 root     other         32 Aug 12  1994 tmp_mnt
-r--r-----   1 bin      mem       913654 May 25 15:09 unix
-r--r-----   1 bin      mem       913654 May 09 10:44 unix.old
drwxrwxr-x  37 root     auth         592 Jun 07 15:51 usr
drwxr-xr-x   3 root     sys           48 Jun 09  1992 var
```

Figure 3
Sample ls Output

Naturally, the output of the ls command is much more voluminous than that of the quot command due to the added detail of the ls command. Also, the quot command describes the disk space allocated for a certain user. The ls command describes the logical size of each file, as opposed to the space allocated for the file, which is often larger. This is because the logical size of the file is the number of bytes in use by the file. However, the operating system will allocate space for a file in disk blocks, which are usually 1 or 2 kilobyte chunks.

Printer Utilization

There is no consistent record format for print utilization across all versions of UNIX. However, some versions of System V UNIX do log print requests in the file /usr/spool/logs/requests. Each completed print request writes several lines to the log file. Each line begins with a character denoting the type of information which is in the line. The identifier for those lines with accounting information and the relevant fields in those lines are listed in the following table:

Line Identifier	Information Used
=	request date, size in bytes
C	number of copies
P	request priority
s	completion status
U	user name
z	printer name

Table 3
Print Log Record Descriptions

Total Accounting Records

System V UNIX also has a suite of utilities for summarizing connect, process and disk utilization into a single record. This record, known as the total accounting record, relates a user name to connect, process and disk utilization over a collection period. This record has the following fields:

Field Name	Field Type	Description
ta_uid	uid_t	user id
ta_name	char[8]	user name
ta_cpu[0]	float	prime CPU time
ta_cpu[1]	float	non-prime CPU time
ta_kcore[0]	float	prime kcore minutes
ta_kcore[1]	float	non-prime kcore minutes
ta_con[0]	float	prime connect time
ta_con[1]	float	non-prime connect time
ta_du	float	disk blocks owned by user
ta_pc	long	processes run by user
ta_sc	ushort	connect sessions
ta_dc	ushort	number of disk samples
ta_fee	short	fee charged to user

Table 4
Total Accounting Record Fields

The separation of time into prime and non-prime is determined by the file /usr/lib/acct/holidays. This file must be edited to list those days which are holidays and the hours of business days which are prime and non-prime. The value of ta_fee is taken from the amount of fees charged to a user via the chargefee command.

Not only can detail records be summarized into total records, but multiple total records can be summarized into a single total record. This is useful when summarizing daily total records into monthly total records. In this case, each of the daily total records for a specific user would be added together to form the monthly record. For most of the utilization fields in the record this is fine, since the counts are additive. However, in the case of disk usage, adding the fields together does not produce a useful result. This summed disk usage field must be used in conjunction with the ta_dc field to get an accurate calculation of average disk utilization over the month. For example, if you summarize 30 daily records belonging to user root into one record, that summary record will have a ta_dc value equal to 30. You can then divide the ta_dc value into the ta_du (disk usage) value to determine the average disk utilization for root over those 30 days.

Enhancing the Data

The process records identify the user and group which own the process by the numeric user and group id. A more useful identifier would be the actual user and group name. The simplest way of getting the user name is searching the passwd file (usually /etc/passwd). The passwd file lists each user name and the corresponding user id. Similar information on group id can be found in the group file (usually /etc/group). UNIX provides system calls to manipulate both files, in order to encapsulate differences in the file formats across platforms.

None of the utilization information identified so far makes any allowance for the situation of collecting information from more than one UNIX server for use in a centralized accounting system. You may wish to charge different servers with different rates; therefore it would be useful to know the origin of the utilization information. This can be solved by adding a field to each record identifying the host from which the record originated. This host identifier can be taken from the output of the hostname command (on some systems, the uname command could also be used).

The output of the ls command can show the name size and owner of a particular file. However, it would be useful to add the device name where the file resides, since disks differ in price. This information can be extracted from the mnttab file (usually /etc/mnttab). As with the passwd and group files, UNIX provides system calls to access this file.

Records in the utmp format can give information on a particular login or logout, but individual records do not fully describe a connect session. A login record has to be matched with the corresponding logout record to capture all connect session information.

Using the Data

The built-in UNIX accounting data, with the above enhancements, can be used as part of a comprehensive chargeback scheme. The four types of data (connect, disk, process, and print) can be traced back to a user name on a specific host. The utilization information can be summarized by any combination of user name, host name or device name (disk or terminal name). Process and connect utilization can also be summarized by shift by checking the time stamps within the relevant records. Process data can also be summarized by command name to find out which applications are consuming the most resources. Thus, you have the basic data needed for chargeback: accountability (via username), utilization, device identification, and date and time stamps.

Once the data has been collected, the next step is to transfer it to a centralized billing system. In many IS organizations where mainframes and UNIX servers coexist, billing is handled through accounting software located centrally on the mainframe. In this situation, the UNIX data has to be converted to a neutral format, such as ASCII text, and merged into the accounting structure present in the mainframe billing system. This merger of UNIX and mainframe data can be accomplished by relating UNIX user names to projects or departments in the established accounting hierarchy. Also, rates have to be determined for the various utilization fields in the UNIX data. With the rates and accounting structure in place, the charging of UNIX usage as part of an overall resource accounting scheme can be realized.

SCSU Overview

The KOMAND UNIX Charging System consists of two parts: a UNIX component and an Z\os component. The UNIX component consists of shell scripts and C++ programs which harvest utilization data and convert it into a format readable by the Z\os component. These files must then be transferred to the mainframe Z\os component. The UNIX component is run in batch mode and therefore can be run at off hours. The Z\os component consists of costing modules which utilize a UNIX Data Definition Table (supplied by PACE) and control statements for charging resource usage back to an organizational accounting structure. The computer utilization data and their respective charges are passed to the KOMAND Financial Management System (FMS) by Miscellaneous Utilization Records (MURs) or KOMAND Accounting Records (KARs). KOMAND FMS is the KOMAND subsystem used to consolidate all data processing charges on to a customer invoice.

UNIX Component Overview

The UNIX component of SCSU consists of modules, which harvest, convert, and create utilization data into a fixed format for the Z\os component. Bourne shell scripts are used to help automate the conversion process. There are currently nine conversion modules. Some of these modules produce detail records and some produce the same type of information at the summary (within user name) level. Two of these modules produce database level statistics (at the user level) for Oracle and DB2/6000. You would not run the modules to produce all nine records. Select the modules to be run after determining the type of records to be produced. If your installation feels that the data is sufficient at the summary level, the module that produces the summary records would need to be run, otherwise, those modules producing the detail records must be run. The print module would need to be run if your installation wishes to account for print and the UNIX system contains the print file. If you have an IBM AIX system, an equivalent print module exists and would be run in place of the normal module. A detailed description of each module can be found in the Programs section of the Product Reference manual.

The latest enhancement to the UNIX Charging System is the ability to track usage within an Oracle or DB2/6000 database by user. **Oracle** usage is tracked by Oracle user name and the records can be produced at either the detail or summary level. Detailed records are produced for each session within Oracle user name and contain resources consumed for that session. Summary records are produced for each Oracle user and contain resources consumed from multiple sessions. Both the detail and summary records contain the UNIX user name as well as the Oracle user name giving one the ability to account for this usage from either name. The resources tracked are the same in both types of record. Some of these tracked resources include: connect time, CPU times, messages sent and received, physical reads and writes, write requests, PGA and UGA memory, user commits, rollbacks, and disk sorts. In order to get these statistics, you must set TIMED_STATISTICS = TRUE and AUDIT_TRAIL= DB. **DB2/6000** records are produced at a detail level and contain key fields such as Application ID, Correlation Token, and Authorization ID. The resources tracked include such fields as User and System CPU Times, Connect Time, reads and writes, updates/deletes/inserts, commits, and rollbacks. For **Sybase**, database activity is summarized by database userid. Resources available for chargeback include CPU time and Input/Output usage.

Z\os Component Overview

The Z\os component reads the converted file(s) produced from the UNIX component and combined with the UNIX Data Definition Table and control statements, charges the resource usage back to an organizational accounting structure. These files contain both resource utilization and accounting information. In addition to these files, Control Statements must be prepared and entered telling SCSU the resources to cost, the costing options, and the accounts to be charged. Miscellaneous Utilization Records (MUR) or KOMAND Accounting Records (KAR) are produced as output and may be read into the KOMAND FMS system. (This file is normally produced at the summary level.) The Charging System History File (IHR) is an optional output from SCSU containing resource utilization and costs at the detail level.
SCSU provides a number of charging options to allow customization of the accounting process to organizational needs, such as:

- Shift accounting provides a means for adjusting basic charges by a discount factor or a
surcharge factor.
- The MINIMUM charging option is used to assign a minimum charge for a service if the resource utilization charges in the record do not accrue to the minimum charge.
- The EXTRA charging option is used to add a fixed surcharge for a service in addition to the basic resource charges.
- The FIXED charge option allows a fixed charge to be applied to a service and disregards resource charges.

Discriminatory charging can be applied at the account level. Different users can be charged for resource usage by special rates or all users charged at the same rate. The variety of charging options available in SCSU provides a flexible means for implementing an accounting strategy.

The SCSU system may be tailored to your installation's specifics by modifying the supplied control statements to indicate site specific processing options, accounts numbers, and resource rates.

SCSU provides a Report Generator that can be used to provide Ad Hoc reports from the output Charging System History File. The Report Generator can be used to produce detail reports to back up the reports already produced or to analyze resource usage/charges, with respect to any control data contained in the Charging System History Record.

Future UNIX Projects

We are currently in the process of developing a UNIX Stand Alone Charging System. It is being designed for the installation that is either a UNIX only shop or is in the process of downsizing to UNIX.

KOMAND UNIX (SCSU) Solution

Early in the development of the KOMAND UNIX Charging System (SCSU), PACE discovered a good deal of information missing from the native UNIX accounting records. In order to enable customers to sufficiently charge for UNIX usage, these records needed to be enhanced. The major deficiencies discovered and the KOMAND/SCSU solutions for each are listed in the table below.

UNIX Process Accounting Record Deficiency	
The standard UNIX process accounting records do not provide a level of detail sufficient to satisfy the chargeback requirements of most installations.	SCSU records provide a full detail record for each process, including start and stop date and time, user and group name, and all available utilization data.
For systems running under UNIX (i.e., Oracle, DB2, etc.) the standard UNIX accounting records do not track the usage within these systems to a level sufficient for chargeback. These records simply group all the activity within these systems together, and charge it to one account.	PACE has developed enhancements to SCSU that track usage within Oracle, Sybase and DB2. Records are created that relate database utilization to database username, allowing the responsible account to be charged. PACE is currently investigating similar solutions for Informix.
The UNIX accounting data does not provide information on the host machine where the activity took place. This does not facilitate charging different rates for different host machines.	PACE has added a field to each of the SCSU records to identify the host where the activity took place, thus allowing for differential charging, not only by UNIX variant (i.e., HP-UX, IBM AIX, Sun Solaris, SCO UNIX, etc.), but also by host machine within each UNIX environment.
Standard print records generated for IBM AIX do not contain printer id information. This means that all printers must be charged the same rate. In addition, the standard records do not track the number of pages printed by job, but roll this information into a totals (summary) record.	PACE has developed a script which adds printer identification information to the standard UNIX print record so that the resulting SCSU IBM AIX print records contain the printer name, thus allowing for charging different rates for different printers. The SCSU IBM AIX print record also includes the number of pages printed by job.

UNIX accounting records do not include the name of the disk device where a file resides. This precludes charging differentially by device.	SCSU adds a field to the detail disk record to identify the disk device name, thus allowing different rates for different disk devices.

SCSU — The UNIX Charging System

SCSU, from PACE Applied Technology, provides the means of collecting, converting and merging into an accounting structure UNIX utilization for the purposes of chargeback. SCSU will take System V connect, disk, process and print information, add the information described above, and convert it into a flat format for the costing component. SCSU consists of C++ programs and shell scripts running under UNIX to collect and convert the data, and a costing component running under Z\os to cost out the records. The costed records may then be merged with other enterprise charges for billing purposes. The following records can be created by the UNIX component of SCSU:

1. **Connect Record**

The connect record describes information relating to a connect session. One connect record is generated for each completed connect session. This record lists start and end time stamps, the total connect time in seconds, the name of the user connected, and the host and terminal name on which the connection took place.

2. **Disk Detail Record**

The disk detail record contains information on the size and ownership of a file. For each file in the file system, a disk detail record can be generated describing the owner of the file, the size of the file in bytes, and the host and disk device name on which the file resides.

3. **Disk Summary Record**

The disk summary record breaks down disk allocation on each device by user. For each mounted disk, a disk summary record is generated for each user owning blocks on that disk. This record contains the user name, the disk device name, the disk allocation in blocks, and the name of the host on which the disk is mounted.

4. **Print Record**

The print record describes information on a completed print request. A print record is generated for each completed print request to a local printer. It lists the name of the user who made the request, the size of the document in bytes, the number of copies, and the name of the printer and host on which the print took place.

5. **Process Record**

The process record describes information related to a completed process. A process record is generated for each process run. It relates CPU usage, memory usage and I/O of a process back to the name of the user who ran the process.

6. **User Summary Record**

The user summary record summarizes connect, disk, and processor utilization for each user.

7. **Database Utilization Records**

Oracle, DB2, and Sybase database utilization may be charged to the responsible clients.

Conclusion

The UNIX operating system remains on the cutting edge of technology. As UNIX continues to evolve, tools for system management will become more abundant. At the same time, as UNIX systems become larger, cost recovery will become increasingly important. A comprehensive chargeback system will be needed to recover costs on UNIX server investments. There is also a need for those organizations with both UNIX systems and mainframes to consolidate utilization information to a centralized billing system. SCSU, from PACE Applied Technology, can be used to bring UNIX resource data into a central billing system to show what resources are being consumed and who is consuming them. Thus, SCSU gives UNIX chargeback an excellent fit in the IS puzzle.

References

- Frisch, Aeleen, *Essential System Administration*, O'Reilly & Associates, Sebastopol, California, 1993
- Nemeth, E., Snyder, G., Seebass, S., and Hein, T., *UNIX System Administration Handbook, Second Edition*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1995
- Raymond, Eric S., ed., *The New Hackers Dictionary, Second Edition*, The MIT Press, Cambridge, Massachusetts, 1993
- Sobell, Mark G., *A Practical Guide to UNIX System V, Second Edition*, Benjamin/Cummings, Redwood City, California, 1991
- Stevens, W. Richard, *Advanced Programming in the UNIX Environment*, Addison Wesley, Reading, Massachusetts, 1992