Paper: 194-2007

A Simple Chargeback System For SAS® Applications Running on UNIX and Linux Servers

Michael A. Raithel, Westat, Rockville, MD

Abstract

Organizations that run SAS on UNIX and Linux servers often have a need to measure overall SAS usage and to charge the projects and users who utilize the servers. This allows an organization to pay for the hardware, software, and labor necessary to maintain and run these types of shared servers. However, performance management and accounting software is normally expensive. Purchasing such software, configuring it, and managing it may be prohibitive in terms of cost and in terms of having staff with the right skill sets to do so. Consequently, many organizations with UNIX or Linux servers do not have a reliable way to measure the overall use of SAS software and to charge individuals and projects for its use.

This paper presents a simple methodology for creating a chargeback system on UNIX and Linux servers. It uses basic accounting programs found in the UNIX and Linux operating systems, and exploits them using SAS software. The paper presents an overview of the UNIX/Linux "sa" command and the basic accounting system. Then, it provides a SAS program that utilizes this command to capture and store monthly SAS usage statistics. The paper presents a second SAS program that reads the monthly SAS usage SAS data set and creates both a chargeback report and a general usage report.

After reading this paper, you should be able to easily adapt the sample SAS programs to run on servers in your own environment. Then, you will be able to create monthly chargeback and usage reports of SAS usage on your own UNIX and Linux servers.

Overview

Several years ago, the author was tasked with finding a method of charging users for their use of UNIX servers to run SAS programs. Like many organizations, Westat wanted to recover the cost of hardware, software, and technical support by charging the various projects that used its UNIX SAS servers. An overall monthly charge for the servers had been computed, so there needed to be a way to determine how users would pay their fair share of that monthly cost.

There is a plethora of chargeback schemes in use among the many organizations that charge for compute usage. Each has its own strengths and weaknesses. Some of the things that organizations charge for are:

- CPU time
- I/O events
- Memory usage
- Disk storage
- Service Units (a mixture of CPU time, I/O, and memory usage)

Based on UNIX usage patterns, we decided that the best way to recover costs would be to charge projects by the percentage of the overall CPU time they used for a given month. The general algorithm for this is:

Project A's Monthly Charge = ((Project A's CPU Time) / (All Projects' CPU Time)) * Monthly Server Charge

We employ a small "fudge-factor" to ensure that no project takes a heavy financial hit on those rare months when very few projects use a particular UNIX server. See the *monthly_linux_accounting_report.sas* SAS program (Appendix E) for the fudge-factor.

Once we decided upon the chargeback algorithm, the author needed a methodology to make it work. A survey of the commercial UNIX-based computer chargeback packages revealed that they are fairly expensive. Purchasing any one of the more prominent packages would cost as much as, or more than, the UNIX server and its software. So, the author settled on developing a home-grown system that uses UNIX Process Accounting as its backbone and SAS as the data collection and reporting tool.

There are strengths and weaknesses to using UNIX Process Accounting as the basis for a computer chargeback system. Strengths are: it comes free with UNIX/Linux, it runs in the background and is unobtrusive to users, and it is fairly simple to use. Weaknesses are: it does not provide a lot of detailed information, it has a limited set of metrics, it doesn't scale well for users who are on multiple projects, and it is not well documented.

Since we needed a simple, CPU-based source of accounting information, the UNIX Process Accounting system was fine for our needs. We decided that we would do the following each month:

- Extract CPU usage information from UNIX Process Accounting
- Remove all of the System and the Systems Programmers' CPU time
- Summarize all of the users' CPU time

- Aggregate users' CPU time by Project
- Charge each Project for their portion of the monthly server charge based on the overall percentage that they used.

This methodology is very stable and has worked well over the past several years—even after we migrated from UNIX to Linux. It was originally created for SAS V8.2 running on HP UNIX servers; and now runs for SAS V9.1.3 running on Red Hat Linux servers. Consequently, this paper uses the phrase "UNIX/Linux" when referring to servers, and the example programs mention "Linux", instead of "UNIX".

Process Accounting and the UNIX/Linux "sa" Command

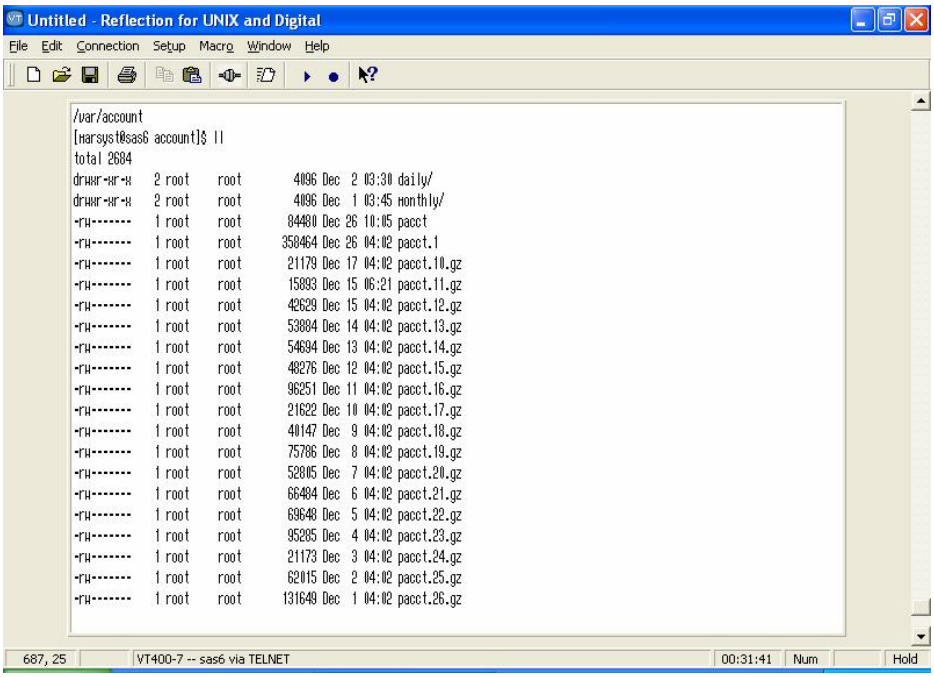
UNIX/Linux Process Accounting runs quietly in the background, recording CPU time, real time, number of processes and other metrics. It makes use of the **/var/account** directory and its subdirectories. That is where Process Accounting stores the data it collects about individual processes and where it stores summarized data.

There are three directories of interest:

1. **/var/account**
   This is the Process Accounting *root* directory. It contains the **/var/account/daily** and **/var/account/monthly** sub-directories, as well as thirty-three daily accounting files. The thirty-three files are composed of the following:

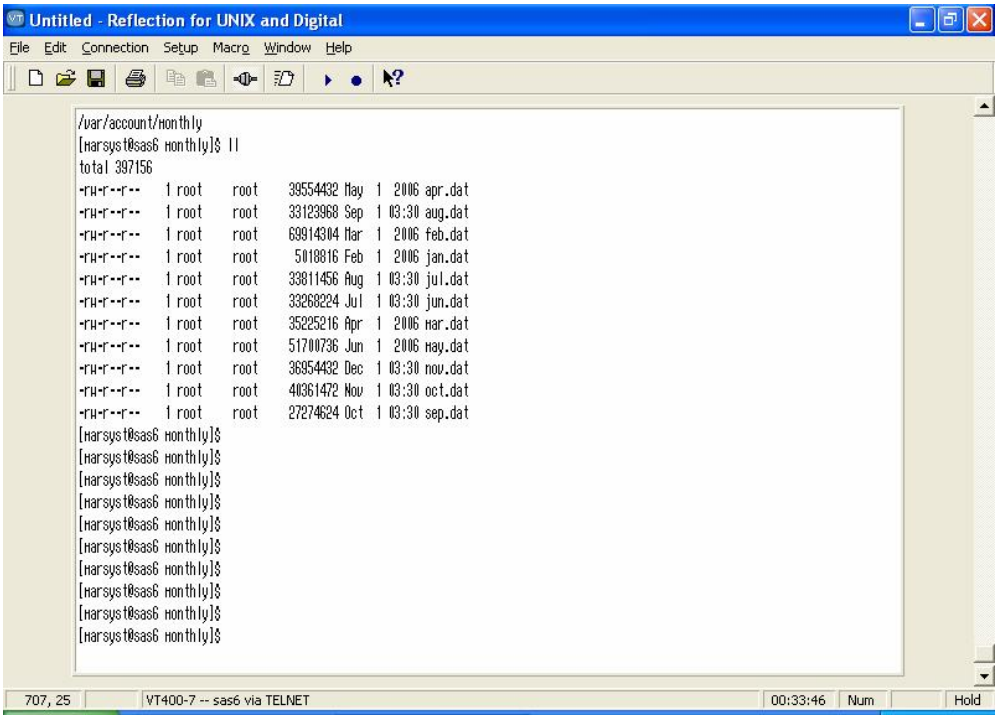
   - **pacct** – The currently open Process Accounting file. Process accounting information from UNIX/Linux tasks that are currently running is written to this file.
   - **pacct.1** – The Process Accounting file from yesterday. This file contains Process Accounting information for all UNIX/Linux users that executed processes "yesterday".
   - **pacct.2.gz** – **pacct.31.gz** – Zipped Process Accounting files for the past 31 days. These files contain detailed information for users' computer resource usage.

   Here is a screen-shot of the **/var/account** directory:



2. **/var/account/daily**
   This sub-directory contains a single file, **pacct.dat**. The **pacct.dat** file contains the Process Accounting information for the current month-to-date for all UNIX/Linux users.

3. **/var/account/monthly**
   This sub-directory contains twelve Process Accounting files; one for each of the past twelve months. Each of the files contains summarized information about the computer resources used during a previous month. Here is a screen-shot of the **/var/account/monthly** directory:



The UNIX/Linux "sa" command is used to extract computer usage information from the monthly Process Accounting files stored in **/var/account/monthly**. The format of the command is:

/usr/sbin/sa –m /var/account/monthly/xxx.dat

Here is what this command means:

- Execute the **/usr/sbin/sa** command to extract Process Accounting information from a specified file.
- The **–m** option specifies that user summary information is to be extracted from the Process Accounting file.
- /var/account/monthly/xxx.dat is the specific Process Accounting file that is to be processed by the sa command. Substitute "jan", "feb", "mar", "apr", "may", "jun", "jul", "aug", "sep", "oct", "nov" or "dec" for the xxx, depending upon which month you want to have processing information for.

This is what the output from a typical execution of this command looks like:

```
/var/account/monthly
[marsyst@sas6 monthly]$ /usr/sbin/sa -m /var/account/monthly/apr.dat
                         618038 5332248.50re    1525.33cp     1466k
root                     276432 3529553.69re     259.87cp      740k
bonham_j                    368    5843.58re      11.14cp     6290k
page_j                     1534    2498.87re      10.62cp     1077k
plant r                   11552    7550.00re      10.47cp     3469k
jones j                   13431    6749.20re       9.76cp     3146k
waters r                  19157   12473.86re       9.60cp     3144k
mason n                    2001    3088.04re       6.32cp     3257k
gilmore d                  3994    2949.42re       5.36cp     4093k
wright r                   1423      28.83re       4.28cp      816k
gdm                           4   41786.03re       0.04cp    10216k
lp                         1870       2.64re       0.04cp      672k
xfs                           1   50331.65re       0.02cp     1980k
```

In the **sa** command output, above:

- Column 1 contains the userid
- Column 2 contains the total number of processes this user used during the month.
- Column 3 contains the total number of seconds of "real time"—wallclock time—the user used during the month.

- Column 4 contains the total amount of CPU time, in seconds, the user consumed during the month.
- Column 5 contains the average CPU storage in "kilo-core seconds".

Looking at the **sa** command output, you can see how easy it would be to write a SAS program to process each line and collect computer resource usage metrics for each userid.

Implementing a Process Accounting-Based Computer Chargeback System

This section details the steps necessary to implement the chargeback system based on UNIX/Linux Process Accounting information. Those steps are:

- **Enable Process Accounting**. The UNIX or Linux administrator must set up Process Accounting on the server and ensure that it is turned on. The administrator should make the necessary configuration changes so that Process Accounting automatically starts when the server boots up.

- **Maintain the Accounting Project Codes SAS Format**. The Accounting Project Codes SAS format (see Appendix A) links user ID's with the projects that should be charged for the users' CPU time. Whenever the UNIX/Linux administrator creates a new user account, the administrator updates the Accounting Project Codes SAS format with the new user ID and project code. Similarly, when user ID's are eliminated because a project has come to an end, they are removed from the Accounting Project Codes SAS format.

- **Execute the Monthly SAS Accounting Programs**. The UNIX/Linux administrator must make an entry in the system *crontab* to schedule the monthly SAS accounting programs to execute after the monthly Process Accounting file has been created. In our case, a script named *MONTHLY_LINUX_ACCOUNTING_SCRIPT* (See Appendix B) was scheduled for execution. That script invokes a SAS driver program named *monthly_linux_accounting_driver_program.sas*. This program %INCLUDE's and executes the following two accounting programs:

    - **monthly_linux_accounting_routine.sas –** This program reads *last month's* UNIX/Linux Process Accounting file and creates observations in a permanent accounting SAS data set.

    - **monthly_linux_accounting_report.sas –** This program reads the permanent accounting SAS data set and creates a server usage report and a server chargeback report. The reports are then emailed to the author.

As you can see, setting up the monthly chargeback routine is a fairly simple endeavor. As long as careful attention is paid to the second step—maintaining the accounting project codes SAS format—there are very few things that can go wrong with the process.

Getting Information from the Monthly Process Accounting Log File

The program that processes the monthly Process Accounting log file, *monthly_linux_accounting_routine.sas*, can be found in Appendix D. This section provides an overview of the main components of that program.

1. Several formats are created that will be used later in the program.

2. A DATA _NULL_ step determines the three-letter name of *last month* and stores it in the FISCFILE macro variable. This will be used later in the program to allocate *last month's* Process Accounting file.

3. A DATA _NULL_ step determines the SYSID of the UNIX/Linux server and stores it in the SYSID macro variable so that it can be used later on in the program.

4. A FILENAME statement allocates *last month's* Process Accounting file, using the FISCFILE macro variable. The FILENAME statement is configured to execute the /usr/sbin/sa UNIX/Linux command, with the "-m" option against last month's file and pipe the results back to SAS.

5. The SAS data library with the permanent UNIX/Linux accounting SAS data set is allocated via a LIBNAME statement.

6. A DATA step reads in the monthly Process Accounting file information via an INFILE statement that accesses the UNIX/Linux system command described in #4, above. It reads the file, line-by-line as it is piped into the SAS program, and parses the lines to retrieve information to populate the *year, month, sysid, userid, tasks, realtime* and *cputime* variables. Observations with accounting information are stored in the LINUXACT temporary SAS data set.

7. The SAS program determines if there are any observations in the LINUXACT temporary SAS data set. If so, the LINUXACT temporary data set is sorted into the proper order. Then it is processed in a DATA step with a MODIFY statement to update the LINUXACT data set in the permanent UNIX/Linux chargeback SAS data library.

The UNIX/Linux Accounting SAS Data Set

The result of processing the monthly UNIX/Linux Process Accounting file is an accounting SAS data set that can be used for chargeback purposes. Here are the variables that are found in that data set:

- **CPUTIME** – The total CPU time used by the userid, in seconds
- **MONTH** – The month of the year that this accounting information is for.
- **REALTIME** – The total wall-clock time used by the userid, in seconds.
- **SYSID** – The system id of the UNIX/Linux server.
- **TASKS** – The total number of tasks executed by this userid.
- **USERID** – The userid that is charged for the consumed computer resources.
- **YEAR** – The year that this accounting information is for.

In the author's organization, the UNIX/Linux accounting SAS data set is named *linuxact.sas7bdat*, and exists in the */home/linuxacct/data* directory on a Linux server. Each SAS Linux server in the organization runs the aforementioned monthly Process Accounting extract SAS program and stores the data in a like-named SAS data set in a like-named directory on the server. Consequently, we know exactly where to look for Linux accounting information on each SAS Linux server in the organization.

Reporting SAS Usage and Chargeback

The program that creates the monthly Linux accounting reports, *monthly_linux_accounting_report.sas*, can be found in Appendix E. Here is an overview of the main components of that program.

1. Several formats are created that are used later in the program. The $EXCLUDE format contains a list of user ID's and tasks that should be excluded from being charged for services. The $SYSTEMS format character lists user ID's and tasks that are systems oriented; not user project oriented. The MONTHFMT correlates the month number with the name of the month.

2. The program %INCLUDES the *monthly_linux_accounting_project_codes.sas* program, which contains the Accounting Project Codes SAS format (see Appendix A). This format links user ID's with the projects that should be charged for the users' CPU time. It is used later in the program.

3. The SAS data library with the permanent UNIX/Linux accounting SAS data set is allocated via a LIBNAME statement.

4. Section I of the program does the following:

    a. A DATA _NULL_ step determines the month and year of *last month* and stores those values in the MONTH and YEAR macro variables, respectively. It also stores the full month name of *last month* in the RTPMONTH macro variable.
    b. A DATA step extracts all observations for *last month* from the *linuxact.sas7bdat* SAS data set. It uses the $EXCLUDE, $CHARGCD, and $SYSTEMS formats to populate the EXCLUDE, CHARGCD, and SYSTEMS variables, depending upon the value of USERID.

5. Section II of the program creates the monthly billing report. It does the following:

    a. It uses the SUMMARY Procedure to summarize the extracted data by CHARGCD. A WHERE data set option is used to include only chargeable observations. The output data set includes an observation with the grand total as well as an observation for the summarization of CPUTIME for each distinct CHARGCD.
    b. A DATA step determines each distinct CHARGCD's percentage of total CPU time and its monthly charge for use of the server. There is a *fudge-factor* built into the algorithm, such that at least 30 minutes of CPU time must be used in a given month. If total CPU time is less than 30 minutes, then the difference between 30 minutes and the total CPU time used is charged to an overhead category. This prevents a project from taking a hit for the full cost of using the server when there has been low server usage.
    c. The PRINTTO Procedure is used to allocate a report file for the chargeback report. Then, PROC PRINT is used to create the report.

6. Section III of the program creates the monthly resource usage report. It does the following:

    a. PROC SUMMARY summarizes all observations in the extract SAS data set by USERID to get the grand total of all CPU usage. It only includes observations where CPUTIME is greater than zero, and USERID is not equal to "TOTAL".
    b. A DATA step reads the summarized data and categorizes the observations into the following categories:
        - Systems
        - Overhead
        - Chargeable

- New Charge

c. The PRINTTO Procedure is used to allocate a report file for the resource usage report. Then, PROC PRINT is used to create the report.

7. Section IV uses the SAS email facility to attach the reports to an email and send them to the interested parties. This is done via a FILENAME statement and a DATA _NULL_ step. Check SAS Online Documentation to verify that you have your UNIX/Linux SAS system configured properly to send email.

The reports created by this program are put to use every month. The first report, "*Total Project Charges on the XXXX Linux Server For the Month of &RPTMONTH, &YEAR*", is used to charge the individual projects for their use of a particular Linux server. The second report, "*Total CPU Usage for All UIDs on the XXXX Linux Server For the Month of &RPTMONTH, &YEAR*" is inspected to see what the CPU usage patterns are. Occasionally, a new userid will pop up on that report that was heretofore unknown. When that happens, research is done to determine whether the user should be associated with an existing project or with a new project. Adjustments to the Accounting Project Codes SAS format are made accordingly and the reports are rerun.

Conclusions

This paper described a simple, cost-effective method of charging SAS users for their use of UNIX or Linux servers. The methodology makes use of UNIX/Linux Process Accounting and is based on charging users for the total percentage of CPU time that they consume in a given month. Two SAS programs are employed: The first extracts CPU usage information from the monthly UNIX/Linux Process Accounting file; the second creates monthly reports on server charges per project and CPU usage per user ID. The programs are *cron*-ed to automatically run each month, and the reports are emailed to staff who review them and charge the various projects for their use of UNIX/Linux resources.

This simple chargeback system may fill your own organization's need to recover UNIX/Linux server costs. If so, you will undoubtedly be successful in implementing a fair way of allocating monthly charges to the many users and/or projects that use your organization's UNIX and Linux servers.

Disclaimer

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

References

SAS Institute Inc. 2004. **SAS OnlineDoc® 9.1.2**. Cary, NC: SAS Institute Inc.

Red Hat Linux man page reference.

Acknowledgements

I would like to thank Westat Vice President Mike Rhoads for his review and comments on the draft of this paper.

Contact Information

Please feel free to contact me if you have any questions or comments about this paper. You may reach me at:

Michael A. Raithel
Westat
1650 Research Boulevard
Room RW4521
Rockville, Maryland 20850

michaelraithel@westat.com

Appendix A
Accounting Codes SAS Formats

These formats inhabit their own SAS program, named *monthly_linux_accounting_project_codes.sas.* The Linux administrator updates this program after creating a user ID for a new user. The new user ID and corresponding project number (to which usage will be charged) are added to the $CHARGCD SAS format. If it is a new project, then the project number and the project name are added to the $APPLID SAS format.

```
proc format;
     Value $chargcd
     'bonham_j'     = '1234.56.78'    /* lzep     */
     'page_j'       = '1234.56.78'    /* lzep     */
     'plant_r'      = '1234.56.78'    /* lzep     */
     'jones_j'      = '1234.56.78'    /* lzep     */
     'waters_r'     = '8765.43.21'    /* pfloyd   */
     'mason_'       = '8765.43.21'    /* pfloyd   */
     'gilmour_d'    = '8765.43.21'    /* pfloyd   */
     'wright_r'     = '8765.43.21'    /* pfloyd   */
     'raithel_m'    = '2468.10.12'    /* sasgroup */
     other= 'No Charge '
     ;
     Value $applid
     '1234.56.78'    = 'lzep      '
     '8765.43.21'    = 'pfloyd    '
     '2468.10.12'    = 'sasgroup  '
     other           = 'No Charge '
     ;
run;
```

Appendix B
Monthly Accounting Script

This is the UNIX/Linux script file that executes the SAS programs that collect the monthly Process Accounting metrics and create the monthly reports. The UNIX/Linux administrator must add this script to the system *crontab*, and schedule it to run after the monthly Process Accounting file has been created.

```
#################################################################################
# Script:  MONTHLY_LINUX_ACCOUNTING_DRIVER_SCRIPT                               #
#                                                                               #
# Author:  Michael A. Raithel                                                   #
#                                                                               #
# Created: 05/03/2003                                                           #
#                                                                               #
# Purpose: The purpose of this script is to process the monthly Linux Accounting #
#          files into a SAS data base and create monthly reports.               #
#                                                                               #
#################################################################################

/home/sas913/sas /home/linuxacct/programs/monthly_linux_accounting_driver_program.sas \
 -noterminal -log /home/linuxacct/programs/monthly_linux_accounting_driver_program.log \
 -print /home/linuxacct/programs/monthly_linux_accounting_driver_program.lst
```

Appendix C
Monthly Accounting Driver Program

The monthly accounting driver program, *monthly_linux_accounting_driver_program.sas*, is executed by the monthly accounting script. It is a simple SAS program that contains %INCLUDE's to include and then execute the accounting SAS programs. The advantage to using the driver program is that additional SAS programs can be added as %INCLUDE's without having to change the *cron*-ed monthly accounting script.

```
*************************************************************************;
* Program: monthly_Linux_accounting_driver_program.sas                   *;
*                                                                        *;
* Author:  Michael A. Raithel                                            *;
*                                                                        *;
* Date:    May 3, 2003                                                   *;
*                                                                        *;
* Purpose: This is a driver program that completes the following functions:  *;
*                                                                        *;
*          1. Executes a program that processes /var/account/monthly/xxx.dat  *;
*          2. Executes the program that creates the monthly accounting reports*;
*                                                                        *;
* Changes:                                                               *;
*                                                                        *;
*************************************************************************;
options macrogen symbolgen mlogic mprint source source2;


  /*********************************************************************/
  /*                           Section I                               */
  /*********************************************************************/
  /* This section processes the monthly xxx.dat accounting file.       */
  /*********************************************************************/

*****************************************************************;
* Include and execute the Linux accounting program.             *;
*****************************************************************;
%include '/home/linuxacct/programs/monthly_linux_accounting_routine.sas';


  /*********************************************************************/
  /*                           Section II                              */
  /*********************************************************************/
  /* This section creates the monthly accounting reports.              */
  /*********************************************************************/

*****************************************************************;
* Include and execute the monthly accounting report program.    *;
*****************************************************************;
%include '/home/linuxacct/programs/monthly_linux_accounting_report.sas';
```

Appendix D
Monthly_linux_accounting_routine.sas

```
*************************************************************************;
* Program: Linux_accounting_routine.sas                                 *;
*                                                                       *;
* Author:  Michael A. Raithel                                           *;
*                                                                       *;
* Date:    May 17, 2003                                                 *;
*                                                                       *;
* Purpose: This program reads the /var/account/monthly/xxx.dat file for last  *;
*          month and updates a SAS data base with Linux User CPU usage, and   *;
*          with some other Linux usage metrics.                         *;
*                                                                       *;
* Changes:                                                              *;
*************************************************************************;

*****************************************************************;
* Options, Formats, Includes, etc.                              *;
*****************************************************************;
options macrogen symbolgen mprint mlogic;
options nodate nonumber linesize=80 pagesize=60;

proc format;
     value mnthnum
     1  = 'jan'
     2  = 'feb'
     3  = 'mar'
     4  = 'apr'
     5  = 'may'
     6  = 'jun'
     7  = 'jul'
     8  = 'aug'
     9  = 'sep'
     10 = 'oct'
     11 = 'nov'
     12 = 'dec'
     ;
     value $nummnth
     'jan' = 1
     'feb' = 2
     'mar' = 3
     'apr' = 4
     'may' = 5
     'jun' = 6
     'jul' = 7
     'aug' = 8
     'sep' = 9
     'oct' = 10
     'nov' = 11
     'dec' = 12
     ;
run;

*****************************************************************;
* Determine Last month and store in a Macro variable.           *;
*****************************************************************;
data _null_;

    length fiscfile $3;

     monthnum = month(today()) - 1;
       if monthnum = 0 then monthnum = 12;

    fiscfile = put(monthnum,mnthnum.);
```

```
     call symput('FISCFILE',fiscfile);

run;

*******************************************************************;
* Determine the Linux host system name and store in a Macro variable*;
*******************************************************************;
filename hostname pipe 'hostname';

data null;
infile hostname;

input sysid $4.;

call symput('SYSID',trim(left(sysid)));

run;

*******************************************************************;
* Allocate the Linux Monthly Accounting File.                     *;
*******************************************************************;
filename fiscfile pipe "/usr/sbin/sa -m /var/account/monthly/&FISCFILE..dat";

*******************************************************************;
* Allocate the SAS data library for Linux fiscal data.            *;
*******************************************************************;
libname  prodfile '/home/linuxacct/data';

*******************************************************************;
* Process the fiscal file to get the Prime and Non-Prime time CPU  *;
* usage of all users.                                             *;
*******************************************************************;
data linuxact(keep=year month sysid userid tasks realtime cputime);

label year     = 'Year'
      month    = 'Month'
      sysid    = 'System ID'
      userid   = 'Userid'
      tasks    = 'Number of Tasks'
      realtime = 'Real Time (Seconds)'
      cputime  = 'CPU Time(Seconds)'
      ;

infile fiscfile missover length=length;

input @;

length bigline $200;

input bigline $varying200. length;

retain year 0 month 0 sysid "&SYSID";

if length  = 0    then delete;

   /*********************************************************/
   /* Determine the Month and Year of this Linux Fiscal Report. */
   /*********************************************************/
if _n_ = 1 then do;

   curmon  = "&FISCFILE";
   month = input(put(curmon,$nummnth.),3.);

  year = year(today());
   if month = 12 then  year  = year - 1;

end;
```

```
   /*********************************************************/
   /* Get the important accounting information.             */
   /*********************************************************/
if _n_ > 1 then do;

   userid = input(scan(bigline,1,' '),$20.);
   tasks = input(scan(bigline,2,' '),5.);
   realstr  = input(scan(bigline,3,' '),$10.);
   cputstr = input(scan(bigline,4,' '),$8.);
end;
else do;
   userid = 'TOTAL';
   tasks = input(scan(bigline,1,' '),5.);
   realstr  = input(scan(bigline,2,' '),$10.);
   cputstr = input(scan(bigline,3,' '),$8.);
end;

 realstr  = translate(realstr,'  ','re');
 realtime = input(trim(left(realstr)),10.2);

 cputstr  = translate(cputstr,'  ','cp');
 cputime  = input(trim(left(cputstr)),8.2);

run;

   /*********************************************************/
   /* This Macro determines if there are observations in the    */
   /* LINUXACT data set and updates the master file accordingly.*/
   /*********************************************************/

%MACRO UPDATEFL;

*************************************************************;
* Determine if there are any obs in LINUXACT.               *;
*************************************************************;
proc sql noprint;
select nobs - delobs into :numbrobs
       from dictionary.tables
        where libname = "WORK" and
              memname = "LINUXACT"
              ;
quit;
  /*******************************************************/
  /* Only do the following if there are obs in LINUXACT. */
  /*******************************************************/
%IF &NUMBROBS NE 0 %THEN %DO;

*************************************************************************;
* Sort the data by product date/time/userid/pid/product.                *;
*************************************************************************;
proc sort data=linuxact nodupkey;
        by year month sysid userid;
run;

*************************************************************;
* Update accounting production SAS data set.                *;
*************************************************************;
data prodfile.linuxact;
modify prodfile.linuxact linuxact;
       by year month sysid userid;

       if _iorc_= 0 then replace;
          else output;
run;
  /*******************************************************/
  /* End of DO Loop to process obs in LINUXACT.          */
```

```
  /*****************************************************/
%END;

%MEND UPDATEFL;

%UPDATEFL;
```

Appendix E
Monthly_linux_accounting_report.sas

```
*************************************************************************;
* Program: monthly_linux_accounting_report.sas                          *;
*                                                                       *;
* Author:  Michael A. Raithel                                           *;
*                                                                       *;
* Date:    May 19, 2003                                                 *;
*                                                                       *;
* Purpose: This program reads the Linux Accounting SAS data base and creates  *;
*          a report of the charges for last month. It mails the report out    *;
*          to specified users.                                          *;
*                                                                       *;
*          This program is divided into a number of sections that do specific *;
*          tasks:                                                       *;
*                                                                       *;
*          I   - Extracts last months data.                             *;
*          II  - Creates a report and CSV file for last month.          *;
*          III - Creates additional reports that characterize utilization of  *;
*                the server over the last month.                        *;
*          IV  - Email reports to interested parties.                   *;
*                                                                       *;
* Changes:                                                              *;
*************************************************************************;

*****************************************************************;
* Options, Formats, Includes, etc.                              *;
*****************************************************************;
options macrogen symbolgen mprint mlogic source2;
options nodate nonumber linesize=80 pagesize=60;

proc format;
     value $exclude
     'sysprog1'  = 'Y'
     'sysprog2'  = 'Y'
     'root'      = 'Y'
     'daemon'    = 'Y'
     'timss'     = 'Y'
     'lasradom'  = 'Y'
     'desktop'   = 'Y'
     'uagent'    = 'Y'
     'bin'       = 'Y'
     'www'       = 'Y'
     'adm'       = 'Y'
     'gdm'       = 'Y'
     'uucp'      = 'Y'
     'lp'        = 'Y'
     'TOTAL'     = 'Y'
     'oracle'    = 'Y'
     'transfer'  = 'Y'
     'smmsp'     = 'Y'
     'sshd'      = 'Y'
```

```
      other       = 'N'
      ;

      value $systems
      'root'      = 'Y'
      'daemon'    = 'Y'
      'timss'     = 'Y'
      'lasradom'  = 'Y'
      'desktop'   = 'Y'
      'uagent'    = 'Y'
      'bin'       = 'Y'
      'www'       = 'Y'
      'adm'       = 'Y'
      'gdm'       = 'Y'
      'uucp'      = 'Y'
      'lp'        = 'Y'
      'TOTAL'     = 'Y'
      'oracle'    = 'Y'
      'transfer'  = 'Y'
      'smmsp'     = 'Y'
      'sshd'      = 'Y'
      other       = 'N'
      ;
      value monthfmt
      1  = 'January'
      2  = 'February'
      3  = 'March'
      4  = 'April'
      5  = 'May'
      6  = 'June'
      7  = 'July'
      8  = 'August'
      9  = 'September'
      10 = 'October'
      11 = 'November'
      12 = 'December'
      ;
run;

%include "/home/linuxacct/programs/monthly_linux_accounting_project_codes.sas";

*********************************************************************;
* Determine the Linux host system name and store in a Macro variable*;
*********************************************************************;
filename hostname pipe 'hostname';

data null;
infile hostname;

input sysid $4.;

call symput('SYSID',trim(left(sysid)));

run;

*********************************************************************;
* Allocate the SAS data library for Linux fiscal data.              *;
*********************************************************************;
libname  prodfile '/home/linuxacct/data';
```

```
 /*************************************************************************/
 /*                          Section I                                    */
 /*************************************************************************/
 /* This section gets last months data from prodfile.fiscdata and creates  */
 /* several variables used in the reports.                                 */
 /*************************************************************************/


*************************************************************************;
* Determine report month/year and set SAS Macro variables.              *;
*************************************************************************;
data _null_;

date = intnx('month',today(),-1);

call symput('MONTH',trim(left(month(date))));

call symput('RPTMONTH',trim(left(put(month(date),monthfmt.))));

call symput('YEAR',trim(left(year(date))));

run;

*************************************************************************;
* Get the data for last month.                                          *;
*************************************************************************;
data fiscdata;
set  prodfile.linuxact(where=(month = &MONTH and year = &YEAR and sysid = "&SYSID"));

length exclude $1  chargcd $13   systems $1;

label chargcd = 'Charge Code'
      cputime  = 'Total CPU Time (Minutes)'
      ;

exclude = put(trim(left(userid)),$exclude.);
chargcd = put(trim(left(userid)),$chargcd.);
systems = put(trim(left(userid)),$systems.);

cputime = cputime/60;

run;

 /*************************************************************************/
 /*                          Section II                                   */
 /*************************************************************************/
 /* This section creates the chargeback report for last months charges     */
 /*************************************************************************/

*************************************************************************;
* Summarize all chargables to get grand total.                          *;
*************************************************************************;
proc summary nway data=fiscdata(where=(exclude='N' and systems = 'N' and
                                       chargcd ne 'No Charge ' and cputime > 0));
     class chargcd;
     var   cputime;
     types () chargcd;
output out= billdata (drop=_freq_) sum=;
run;

*************************************************************************;
* Calculate charges for each individual userid/chargcd.                 *;
*************************************************************************;
data billdata(keep=chargcd applid cputime pctcpu charges);
set billdata;

retain syscpu 0;
```

```
format pctcpu percent8.2 charges  dollar12.2;

length applid $10;
label  applid = 'Application';

label pctcpu  = 'Percentage of Total CPU Time'
      charges = 'Total Charges'
      ;

if _type_ = 0 then do;
       if cputime >= 30 then do;
              syscpu = cputime;
              delete;
       end;
       else do;
              cputime = 30 - cputime;
              syscpu = 30;
              chargcd = '1171.13';
       end;
end;

pctcpu  = cputime/syscpu;
charges = pctcpu * 25000;

applid = put(trim(left(chargcd)),$applid.);

run;

*****************************************************************;
* Create Chargeback report.                                     *;
*****************************************************************;
proc printto print="/home/linuxacct/programs/Monthly_Chargeback_Report_&SYSID..txt" new;
run;

proc print data=billdata noobs label;
var chargcd applid cputime pctcpu charges;
sum                cputime pctcpu charges;
title1 "Total Project Charges on the &SYSID Linux Server";
title2 "For the Month of &RPTMONTH, &YEAR";
run;

proc printto;
run;

 /*************************************************************************/
 /*                          Section III                                  */
 /*************************************************************************/
 /* This section creates additional reports that characterize CPU usage on */
 /* the server for last month.                                            */
 /*************************************************************************/

*****************************************************************;
* Summarize all userids to get grand total of _ALL_ CPU usage.  *;
*****************************************************************;
proc summary data=fiscdata(where=(cputime > 0 and userid ne 'TOTAL'));
     class userid;
     id year month;
     var   cputime;
output out= report2(drop=_freq_) sum=;
run;
```

```
*****************************************************************;
* Create a report of resource usage by categories of users.     *;
*****************************************************************;
data report2;
set  report2;

retain syscpu  0
       numdays 0
       nummins 0
       ;

format pctcpu percent8.2;

length category $10;

label category = 'Processing Category'
      pctcpu  = 'Percentage of Total CPU Used'
      pctotal = 'Percentage of Possible Monthly CPU'
      ;

if _type_ = 0 then do;

   syscpu = cputime;

   fromdate = mdy(month,01,year);
    todate  = intnx('month',fromdate,+1);
   numdays  = intck('day',fromdate,todate);
   nummins  = numdays * 24 * 60;
   call symput('NUMMINS',trim(left(nummins)));

   delete;

end;

pctcpu   = cputime/syscpu;
pctotal  = cputime/nummins;

if put(userid,$systems.) = 'Y' then category = 'Systems   ';
   else if put(userid,$exclude.) = 'Y' then category = 'Overhead  ';
        else if put(userid,$chargcd.) ne 'No Charge' then category = 'Chargeable';
        else category = 'New Charge';

run;

proc sort data=report2;
by category userid;
run;

*****************************************************************;
* Create Resource Usage report                                  *;
*****************************************************************;
proc printto print="/home/linuxacct/programs/Monthly_CPU_Usage_Report_&SYSID..txt" new;
run;

proc print data=report2 noobs label;
by category;
id category;
var userid cputime pctcpu;
sum        cputime pctcpu;
title1 ”Total CPU Usage for All UIDs on the &SYSID Linux Server”;
title2 "For the Month of &RPTMONTH, &YEAR";
run;

proc printto;
run;
```

```
 /*************************************************************************/
 /*                          Section IV                                   */
 /*************************************************************************/
 /* Email the monthly Linux accounting and resource usage reports to      */
 /* various people for review.                                            */
 /*************************************************************************/

*************************************************************************;
* Mail the reports to the various users.                                 *;
*************************************************************************;
filename outbox email;

data _null_;
file outbox
        to=('michaelraithel@westat.com')
     subject="&SYSID Linux System Accounting Reports For &RPTMONTH, &YEAR"
         attach=("/home/linuxacct/programs/Monthly_Chargeback_Report_&SYSID..txt"
                 "/home/linuxacct/programs/Monthly_CPU_Usage_Report_&SYSID..txt")
     ;

put 'Good morning,';
put ;
put 'Attached, are the Monthly Linux Accounting Reports.';
put 'If you have any questions, do not hesitate to contact me at X - 3976.';
put ;
put ;
put 'Sincerely,';
put ;
put 'Michael A. Raithel';

run;
```