QUACC - A query and reporting system for UNIX system accounting data

Nicole Cooper  
*University of Wollongong*

Raymond Coury  
*University of Wollongong*

Steven Hegyi  
*University of Wollongong*

Neil Hodgson  
*University of Wollongong*

Follow this and additional works at: [https://ro.uow.edu.au/compsciwp](https://ro.uow.edu.au/compsciwp)

**Recommended Citation**
Cooper, Nicole; Coury, Raymond; Hegyi, Steven; and Hodgson, Neil, QUACC - A query and reporting system for UNIX system accounting data, Department of Computing Science, University of Wollongong, Working Paper 85-4, 1985, 67p.  

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au
QUACC – A QUERY AND REPORTING SYSTEM FOR UNIX SYSTEM ACCOUNTING DATA

Nicole Cooper
Raymond Coury
Steven Hegyi
Nell Hodgson

CSCI321 Project 1984
Department of Computing Science
University of Wollongong

Preprint No. 85–4 January 1985
QUACC
A QUERY and REPORTING SYSTEM
for
UNIX SYSTEM ACCOUNTING DATA

Nicole Cooper
Raymond Coury
Steven Hegyi
Neil Hodgson

Submitted in partial fulfillment of the
requirements of CSCI 321 Software Project at the
University of Wollongong in 1984
INDEX


The TECHNICAL REPORT for QUACC

A QUERY and REPORTING SYSTEM

for the UNIX SYSTEM ACCOUNTING DATA

GROUP E

Nicole Cooper
Ray Coury
Steve Hegyi
Neil Hodgson

The University of Wollongong

ABSTRACT

This document is the technical report for an interactive screen-based query and reporting system for accounting data collected by the UNIX system on the Computing Science Perkin-Elmer computers.
INTRODUCTION

DEFINITIONS

OVERVIEW

1. USER INTERFACE
   Overview
   Description
   Data Structures
   Implementation
   Files

2. SYSTEM CO-ORDINATOR
   Overview
   Mainline

3. ACCOUNTING FILES ACCESSOR
   Overview
   Mainline
   Retrieval From Acct File
   Retrieval From Wtmp File
   Connect Information
   Terminals Connected
   User Amusement

4. ADMINISTRATION FILES ACCESSOR
   Overview
   Initialisation
   Convert Names to Ids
   Convert Ids to Names
   Get a Users Group Id

5. REPORT GENERATOR
   Overview
   Report Dimension
   Data Structures
   Initialisation
   Building
   Pruning
   Sorting
   Printing
   Re-Use

6. ARCHIVING

7. WEEKLY SUMMARY REPORTS

8. IMPROVEMENTS
   Retrieval From Login/Logout File
   A NOT Operator for Limitations
   Changing User Ids

9. APPENDICES
INTRODUCTION

QUACC is a highly flexible, interactive screen-based system which queries accounting data. It allows the user to enquire about a wide range and combination of information about the usage and performance of the computer system.

DEFINITIONS

field (on the screen) - an area of the screen containing a string descriptor and a section where the user can enter information.

parameter - a value entered into a field (on the screen)

menu - a collection of fields displayed on the screen

categorize (a report on a field) - a record will be produced in the report for each distinct value of the field in the accounting files. For example, if a report on each group is required, the user must categorize on the group field.
OVERVIEW

This system allows the user to interactively query the accounting data accumulated by the system and produces reports or summaries pertaining to the nature of the enquiry.

The system allows enquiries to cover the following categories:

* UNIX commands
* UNIX users
* UNIX groups
* terminals
* daily reports
* hourly reports
* all calls to commands

The system is broken down into the following five major sub-systems - the User Interface, the System Co-Ordinator, the Accounting Files Accessor, the Administration Files Accessor and the Report Generator.

The User Interface handles all interactions with the user, being responsible for the screen display via which the user enters the requirements for his report. It accepts the data input by the user and checks it for consistency. This checking may require calling the Administration Files Accessor which accesses the system administration files - which contain lists of user, group, and terminal names and their corresponding numerical ids - in order to check that names entered by the user actually exist. When a valid request for a report has been made by the user, the User Interface sets up various structures specifying the content, limitations, and destination of the report. It then calls the System Co-Ordinator, passing it these structures.

The System Co-Ordinator is the link between the User Interface, Accounting Files Accessor, and the Report Generator. It co-ordinates their activities during the creation of a report. The System Co-Ordinator initially passes the content requirements and limitations of the report to the Accounting Files Accessor.

The Accounting Files Accessor uses the limitations that it is passed to retrieve the data required for the report from the system accounting files. To simplify the preparation of the report this sub-system calls the Administration Files Accessor to convert any names that it retrieves such as terminal, command, user, and group names into their corresponding numerical ids. As data is retrieved by the Accounting Files Accessor from the accounting files, it is sent to the Report Generator which builds the report from this data.

Once the Accounting Files Accessor has finished reading the accounting files, control is returned to the System Co-Ordinator.
This sub-system then calls the Report Generator, passing it information on which fields to categorise, print, total, average, and sort. The Report Generator performs the required operations and finally formats the report. In formatting the report, this sub-system calls the Administration Files Accessor to convert the numerical ids of terminals, commands, users, and groups back to names. The formatted report is sent by the Report Generator line by line to the System Coordinator, which outputs it to a VDU, line printer, or to a file.

After the report is generated, control returns to the User Interface.

Following is a diagram showing the overall system control and data flow :-
1. USER INTERFACE

Overview

The User Interface subsystem services all requests from the user. It controls all report tailoring functions, providing the user with a highly flexible and non-procedural specification mechanism. A report format is specified either through an interactive menu system or from a file defining the desired format. This file is created using QUACC interactively. All menus and their selections are read from files and stored dynamically, in such a way providing easy movement between menus and their fields.

If information is to be drawn from accounting files other than the current defaults, those filenames are noted for later use.

The User Interface passes all necessary format information to the System Co-Ordinator, allowing it to expedite the report generation process.

Description

The screen is divided into six areas. These areas are:

* subject menu
* content menu
* output menu
* option list
* report format area / help messages
* function list

Movement is possible between the three menus, with no imposed order. The cursor may be moved forwards and backwards between fields. The cursor will wrap-around at the beginning and end of the menu.

Subject Menu

The subject menu is the primary menu of the subsystem. The subject of the report is selected from any combination of the following:

* date
* time
* command
* user id
* group id
* terminal id
* individual records
Options

The subject fields can have the following options applied to them:

* impose a limitation
* categorise the field

A field may be restricted to particular values by using the Limitation option. All values placed in the subject menu fields are checked for validity. Concise error messages help in rectifying any errors. Whilst entering a limitation, left and right cursor movement is possible within the field’s boundaries. A limitation cannot be placed on the "individual records" field.

The categorization option results in a section of the report being devoted to every distinct value of the categorized field. All categorized fields are printed in the report. Indication of the subject field being categorized appears in the report format area of the screen.

Categorization on the "individual records" field allows the user to extract records for each call of a command from the accounting file. This is useful when trying to analyse small parts of the accounting data files directly.

The selection field the cursor currently points to is the target field on which any selected option is applied.

Content Menu

The content menu contains selections relevant to the subject(s) chosen. The fields available for selection vary accordingly. One of five possible content menus is chosen for display. These include:

i) Time menu
ii) Command menu
iii) User / Terminal menu
iv) Group menu
v) Individual Records menu

See Appendix 1 for details of the fields in each menu.

Options

The following options can be applied to content menu fields:

* impose a limitation
* print field
* graph field
* increasing sort
* decreasing sort
A field may be restricted to particular values by using the limitation option in the same way as has been described in the subject menu. The fields in this menu are numeric, allowing the range operators of "<" and ">" to be applied to restrict the values.

To allow values accumulated for a field to be printed in the report, the print option can be applied to this field. The report format area shows the order in which the headings for the fields will occur in the final report.

One field in the content menu may be graphed by applying the graph option to it. The graph is in the form of a histogram, containing the numerical totals for each bar in the graph. The report format area acknowledges which field is graphed.

Up to four fields in the content menu may be marked for sorting by either increasing magnitude or decreasing magnitude. A sorted field is printed in the report if not already marked for this.

The report format area indicates which fields were marked for sorting, and the direction of the sort.

The order in which fields are selected for sorting will determine the sorting hierarchy, with the first sort field being the primary key.

An option, after being applied to a field, can have its action removed by 'toggling' the command. The option in the option list is dynamically updated to reflect the current state of the toggled command.

Report Format Area

The report format area dynamically shows the user the format of the report which has been tailored before it is generated.

As specific options are applied to fields in the menus, acknowledgement is made in the report format area of the screen. A line showing the headings of the fields being printed in the report is updated dynamically. The order of occurrence of headings in this line is categorized subjects, followed by printed, sorted or graphed content fields.

A separate line shows which printed fields are marked for sorting, together with the sorting direction.
Output Menu

The output menu contains the following sections:

* report destination
* number of lines in the report

Possible destinations are the screen (default), printer or a file. These and the report length field are entered using the limitation option.

Flexibility

All menus types are displayed on the screen together allowing the user to see the full scope of parameters selected for tailoring the desired report.

Each time the content menu is selected, the type of menu displayed is re-evaluated and printed, removing any inconsistencies between subjects and contents of the final report.

Functions

A static set of functions can be invoked at any time during the report tailoring phase. Their commands appear at the bottom of the screen. These functions are:

* a help facility
* UNIX command execution
* reset all report parameters
* save report parameters in a file
* generate report
* exit the system

The help facility provides the appropriate command or data information at any stage of report tailoring.

The reset function allows the user to clear the screen of his previous entries so that report parameter selection may begin for a new report.

The save report function allows a commonly generated report format to be saved in a file. QUACC may be run with this file as an option, allowing the report described in the file to be generated, thus bypassing the use of the menus. This can be done in the background, and so the speed of generation is not a problem for the user. This also allows QUACC output to be processed by further filters if desired.

When the generate report function is selected, the report format tailored by the user is passed to the system co-ordinator in the structure INPUT_STRUCT, allowing the report to be collated and printed. After the report has been displayed, its format and content are still available, allowing minor modifications to be made if required.
The exit function allows termination of "QUACC" at any time.

**Data Structures**

To allow easy manipulation of each menu on the screen, a data structure, SCRMENU, is generated at initialization to hold all the required menu information. This array of menu structures contains an entry for each possible menu in the system. The information is read from a set of ASCII input files.

A linked list with a node of type FIELDNODE for each selection in the menu is also constructed. The necessary attributes for the selection are placed in the node.

The structure INPUT_TYPE contains the parameters defining the format of the report required by the user. This data structure is passed to the System Co-Ordinator, allowing it to process the requirements.
Implementation

The user interface first determines whether any files have been supplied as options in the command line. If alternate accounting files are specified, their existence is verified and the names are noted for later use. If a report format file is specified, the menus are bypassed, and the System Co-Ordinator is sent this file information for report generation.

If the menus are used, the terminal is set in the correct mode and the menu structure is generated from the input files.

To allow QUACC to process a request as soon as it is entered, the terminal device is set in CBREAK mode. This allows instant processing of each character, providing efficient filtering of characters and more control of the screen in general.

All echoing of characters is program controlled.

The normal functions of CTRL T and CTRL R for starting and stopping output on the terminal are ignored, as other mechanisms are provided for segmented viewing of the report. This also allows these characters to be used as commands in QUACC.

The only exit path from QUACC is via the 'quit' character. This is changed from the UNIX default to a CTRL D, as this is a standard termination character for utilities.

Necessary signals are trapped allowing the terminal to be reset to its normal operating mode if the system is terminated for any reason.

A loop to process each command as it is entered exists, being terminated only upon receipt of the EXIT character. Both the error line and the help area are cleared after messages have been placed in them. The help area is replaced by the current report format.

Any one of the following command types are processed by the loop:

UP - move cursor to the previous field in the menu.
DOWN - move cursor to the next field in the menu.
HELP - print out a help message for available options.
UNIX COMMAND - allows execution of one UNIX command.
LIMITATION - impose a limitation on the current field.
PRINT - print the current field in the report.
INCREASING or DECREASING SORT - sort the current field
in the selected direction.

GRAPH - graph the current field.

CATEGORIZE - categorize the current subject field

RESET - initialize the report format parameters

SAVE REPORT - save the report format in a file

SUBJECT MENU - activate subject menu if not active

OUTPUT MENU - activate output menu if not active

CONTENT MENU - select content menu and display it

GENERATE REPORT - call System Co-Ordinator to generate report

Any changes are made to the option list if necessary.

Files

The following ASCII input files are used:

1) "menu" contains a record for each menu available in the system. Each record contains:

* the name of a file containing all the selections for this menu
* the menu heading
* the screen position the heading is displayed at
* a menu identification

2) Menu content files

For this system the following menu files are used:

* .subjfld  (subject menu)
* .commfld  
* .userfld  |
* .grpfld   | (content menus)
* .timefld  |
* .indivfld v
* .destfld  (output menu)

See Appendix 2 for an example of one of these files.

3) Help Files

These are ASCII files containing the appropriate help message for the selection field(s) it is associated with. These files are up to eighty characters wide and five lines in length.
2. SYSTEM CO-ORDINATOR

Overview

The System Co-Ordinator is the interface between the User-Interface, the Accounting File Accessor and the Report Generator. It co-ordinates the generating of a report given a set of user-requirements by the User-Interface. The sub-system consists of only the procedure sys_cord.

Mainline

Description

This procedure generates and prints a report given a set of user requirements. It first calls the Accounting Files Accessor to read the necessary information generate the basic report. Then the Report Generator routines are called to generate and print the final report.

Definition

sys_cord(user_inputs)
INPUT_STRUCT user_inputs;

"user_inputs" is a summary of the user's inputs.

Data Structures

The structure INPUT_STRUCT contains a summary of all of the user's inputs and is the method by which the User Interface communicates with the System Co-Ordinator. For a detailed description of this structure see Appendix 3.

Implementation

This procedure first determines if the subject of the report has changed from the previous call. This is done because it is possible that the destination, the content or the sorting fields of the report are the only things that have changed, and so the report does not have to be regenerated, but can be reset (by a call to "reset rep") to allow for re-sorting and printing to a new destination.

If the subject of the report has changed then a new report must be generated by calls to "startup" to initiate the building of a report and to "file_accessors" to read the necessary files to build the basic report.

When the basic report has been generated the Report Generator routines are called to build the final report. Firstly, "prune" is called to delete from the report all lines that do not satisfy the content
limitations. Then "sort" is called to sort the report on the field(s) selected for sorting. If no fields have been selected for sorting then sorting should be performed on the field(s) selected for categorising.

Then the main report heading is set-up and the routine "print" called to print the report to the selected output device.

The routine returns to the User Interface a negative number, if a report was not printed, so that the screen does not have to be re-drawn.
3. ACCOUNTING FILES ACCESSOR

Overview

This sub-system accesses the system accounting files and selects from them the data relevant to the requested report.

It consists of the main procedure "file accessor", which is called by the System Co-Ordinator. The file accessor can be broken into the following sections, which handle the different information requirements of the report. These are:

1) Retrieval From the Acct File
2) Retrieval From the Wtmp File
3) Calculating Connect Information
4) Calculating Terminals Connected Per Hour

The information selected for the report is passed to the Report Generator which builds the report.

The file accessor is also responsible for the User Amusement which is described later.

Mainline

Description

This procedure is called by the System Co-Ordinator, which passes to it the restrictions on the data to be extracted from the files. The procedure controls the reading of the accounting files for the information necessary to build the requested report. The procedure returns the limits of the time period within the specified time interval for which data exists in the accounting files.

Definition

```c
file_accessor(limits,acct_flag,conn_flag,
             term_flag,interactive)
SUBJ LIMITS limits;
BOOLEAN acct_flag;
BOOLEAN conn_flag;
BOOLEAN term_flag;
BOOLEAN interactive;
```

The structure SUBJ LIMITS contains the restrictions on the data which is to be selected for the report. For a more detailed description of this structure see Appendix 3.

The acct_flag, conn_flag, and term_flag indicate which files must be accessed to retrieve the data for the report and which calculations must be performed on the retrieved data.
The flag 'interactive' is set if user amusement is required.

Implementation

The mainline of the Accounting Files Accessor checks acct_flag to see if data from a file containing information about each process that terminates is required for the report (the default file of this format is '/usr/adm/acct', but a different file of the same format may have been specified to be used instead when QUACC was invoked). If data from an 'acct' file is required, two binary searches are performed on the file. One finds the first record having a 'process finish time' falling within the user specified time range of the report. The other finds the last occurrence in the file of a record having a 'process finish time' falling within the calculated time range for the report.

These binary searches are made to minimize the number of accesses to the specified 'acct' file. They are possible because records in the file 'acct' are ordered on the finish time of the process.

The mainline then checks conn_flag and term_flag to see if data from a file containing login information is required for the report (the default file of this format is '/usr/adm/wtmp', but a different file of the same format may have been specified to be used instead when QUACC was invoked). If either of these flags is set, a binary search is performed on the file to find the last record in the file having a recorded time within the user specified time range of the report. This binary search reduces the number of accesses to the specified 'wtmp' file.

A binary search is not performed to find the first occurrence of a record in the 'wtmp' file with a recorded time within the time period of the report. If it was, then the login records of users who logged on before the time range of the report would not be taken into account. However, some of these users might have remained logged on into the time period of interest to the report, meaning that their login record is relevant to the report. Therefore the 'wtmp' file must be searched from the beginning to detect the logins of these users.

If data from both the 'acct' and 'wtmp' files is required for the report, but the data in the 'acct' file doesn't cover the specified time period, then only data within the time period covered by the 'acct' file is collected from the 'wtmp' file and used in the report.

If acct_flag is set, the function to retrieve information from the specified 'acct' file is called,
which passes the relevant records to the Report Generator.

If conn_flag or term_flag are set, indicating that login information is required, a function is called to build a tree of logins from the relevant data in the specified "wtmp" file.

If conn_flag is set, the function to calculate connect information from the tree of logins is called, which passes relevant data to the Report Generator.

Otherwise, if term_flag is set, the function to calculate the number of terminals connected for every hour in the time period of the report from the tree of logins is called, which passes relevant data to the Report Generator.

The Accounting Files Accessor sends the data and calculations from the accounting files to the Report Generator via the structure REP_DATA_STRUCT. For a detailed description of this structure see Appendix 3.

If a tree of logins was built it is now deleted.

If the report format was specified interactively, the Accounting Files Accessor provides the user amusement.

The mainline returns to the System Co-Ordinator the time limits between which data exists in the accounting files within the time period the report was requested to cover. This covers the case where the accounting files did not cover any of the time period of the request.

Retrieval From the Acct File

Description

This function retrieves the data relevant to the report from either the system accounting file "/usr/adm/acct" or the file of the same format that was specified to be used instead when QUACC was invoked. It sends the retrieved data to the Report Generator via the function "save".

Files of the format of "/usr/adm/acct" contain data about each process that terminates, and the format is described in ACCT(5) of the UNIX PROGRAMMER'S MANUAL.
Implementation

The binary searches performed in the mainline on the "acct" file established the position of the first record of interest in the file, and the number of records to read from this position in the file to cover the relevant time period.

For each record read, the function checks if the record passes all of the user's restrictions. If it does, the data on the record (i.e. user id, group id, command id, finish time of command, and user, system and elapsed times) is transferred to the structure REP_DATA_STRUCT which is sent to the Report Generator via the function "save".

Retrieval From the Wtmp File

Description

This function retrieves login information from either the system accounting file "/usr/adm/wtmp" or the file of the same format specified to be used instead when QUACC was invoked. A tree of logins is built from the retrieved data.

Files of the format of "/usr/adm/wtmp" contain data about every login and logout, and their format is described in UTMP(5) of the UNIX PROGRAMMER'S MANUAL.

The login information is collated in an intermediate structure so that corresponding logins and logouts can be matched.

Data Structures

The collating structure is a two dimensional tree such that the first level nodes of the tree each contain a terminal id and a subtree. Each of the nodes of a subtree is a second level node of the tree, each containing a user id and a list. Each node of the list has the login time and the logout time of one login occurrence.

Following is an example of an instance of the tree of logins:
TREE OF LOGINS.

KEY:

- First level nodes, contain terminal ids
- Second level nodes, contain user ids
- List nodes, contain time on / off
The login information is processed so that a terminal can only have one user logged onto it at a time. However, a user may be logged on to any number of terminals at a time. Thus, terminals are stored at the top level of the tree so that the subtree for each terminal can at any time only have one login time stored whose corresponding logout record has not yet been read from the "wtmp" file. As a result the login times for a particular terminal are added to the tree in order of time, and so are stored in lists rather than in a third level subtree, as this would degenerate into a list anyway.

Implementation

The binary search performed in the mainline on the specified "wtmp" file established the number of records that have to be read from the file.

For each record read the type of record must be established. See UTMP(5) in the UNIX PROGRAMMER'S MANUAL. If the record indicates that the system's time has changed, ignore it. If the record indicates a reboot of the system, then the login tree is searched for logins which have no logout time associated with them. The recorded time of the record directly before this reboot record in the "wtmp" file is made the logout time for these logins, and the nodes are updated. This approximates the time that the system crashed.

If the record indicates a login then a check is made to see if it satisfies the report's restrictions on users, groups and terminals. If the record satisfies all these checks then the login is added to the tree.

If the record indicates a logout then two cases are possible.

If the logout time on the record is before the lowest time the report is interested in, then the entire login was performed before the requested time period, and so is irrelevant to the report. Thus, the login entry in the tree corresponding to the logout record must be deleted from the tree of logins.

Otherwise the time of logout is after the minimum time that the report is interested in. Then some or all of the entire login associated with that logout record occurs during the time period of the report and so is relevant to the report. The node containing the login entry corresponding to this logout record is found and the logout time is added to it.

Logout records for which no corresponding login can be found in the login tree are ignored, as a logout record is added to the "wtmp" file whenever a terminal is turned on or off, and so is not necessarily an
indication that a user logged off.

Any logins that continue past the upper limit of the time period of interest to the report will have a logout time of zero in the tree. These logins are interpreted as finishing at the upper time of the interval covered by the report.

**Connect Information**

**Description**

The Connect Information function calculates the time over the time period of interest to the report that each login was logged in for and sends the results to the Report Generator via the function "save". The information is used by the Report Generator to calculate the connect time and number of connects of users, groups, and terminals.

**Implementation**

A tree of logins relevant to the report must have been built from the specified "wtmp" file.

The Connect Information function traverses the tree and retrieves each login entry in the tree. For each of these login entries, the time in seconds that the login lasted over the period covered by the report is calculated. The login information (including the start time of the login) and connect time are transferred to the structure REP_DATASTRUCT which is sent to the Report Generator via the function "save".

**Terminals Connected**

**Description**

This function calculates the number of terminals that were logged on for each hourly interval over the time period covered by the report. The results are sent to the Report Generator via the function "save".

**Implementation**

A tree of relevant logins must have been built from the "wtmp" file.

The Terminals Connected function calculates the beginning and end of each hourly interval over the time period covered by the report. For each of these hourly intervals, the function traverses the tree of logins and counts the number of terminals that were connected at some time during that hourly interval. Each terminal can only be counted as being connected once during that hour. The terminal id, the number of terminals connected for the hour, and the start time of the hour are transferred to the structure REP_DATASTRUCT which is sent to the Report Generator.
via the function "save".

User Amusement

Description

The user amusement gives the user an indication of how the report is progressing. If the report is run interactively (rather than by a command file) the user may have to wait for a short time period before the report is displayed. Irritation at waiting for the report may be reduced if an indication of the waiting time appears.

Implementation

A full line of inverse video spaces is initially sent to the user amusement area of the screen.

The length of the line is proportional to the number of records that still have to be read from the accounting files. It is reduced in length as more records are read. This is achieved using the figure calculated as the number of records that must be read.
4. ADMINISTRATION FILES ACCESSOR

Overview

This section handles the accessing of the administration files of the UNIX operating system that are required by QUACC. It is called by the User Interface, the Accounting Files Accessor, and the Report Generator.

The administration files are accessed in order to map numerical ids of users, groups and terminals into their corresponding names, and vice versa. There are system defined procedures to perform these mappings, however these are too slow for QUACC's purposes.

This mapping is required because the user sees the alphabetic names of users, groups and terminals, while QUACC manipulates the numerical ids of these entities.

This sub-system involves the following sections:

1) Initialisation,
2) Convert Names To Ids,
3) Convert Ids To Names,
4) Get a Users Group Id,

The section Initialisation reads the necessary administration files to create an array of user, group and terminal names. The other three sections access these arrays in order to do the necessary mappings mentioned above.

Initialisation

Description

This procedure is called the first time a mapping is required for a user, group or terminal. It creates an array used to map the numerical ids (of users, groups or terminals) to their names and, vice versa.

Definition

\[
\text{init\_arrays(id)}
\]
\[\text{ID\_TYPE id;}\]

"id" is the type of ids that should be created and may be USER\_ID, COMMAND\_ID or TERMINAL\_ID.
Data Structures

There is an array of names for each of the ID_TYPE's (USER_ID, COMMAND_ID and TERMINAL_ID), and also a corresponding integer for each of these arrays indicating the size of the array. This size is used to dynamically update the size of the array to handle the case of increasing number of user, group or terminal IDs. There is also an array containing the group IDs of each user which is the same size as the array of user names.

Implementation

1) The array of user names indexed by their numerical id is read from the file "/etc/passwd",

2) The array of group names indexed by their numerical id is read from the file "/etc/group",

3) The array of terminal names indexed by their numerical id, is read from the file "/dev",

4) The array of each user's group id is read from the file "/etc/passwd".

These are all used for verification of data specified as part of the report format.

Convert Names to IDs

Description

This procedure converts the name of a command, a user, a group or a terminal into its corresponding numerical id, and returns its value. If the name cannot be found then a negative number is returned.

Definition

get_id(name, id_type)
char *name;
ID_TYPE id_type;

"name" is the name of a type "id_type", where "id_type" can be a user, group, command or terminal.

Data Structures

The above mentioned arrays of user, group and terminal names are used by this procedure.

This procedure also uses the following structure to convert a command name to a numerical id.

typedef union {
    int un_id;
    struct field_struct {
        unsigned : 1;
        unsigned char1 : 7;
    } field;
};
unsigned char2 : 7;
unsigned char3 : 7;
unsigned char4 : 7;
unsigned n : 3;
}
} ID_UN_TYPE;

The procedure also uses a hash table for the command name conversion. This process is explained in detail below.

Implementation

If the name passed to the procedure is a user, group or terminal and this is the first reference to a name of this type then "init_arrays" is called to initialise the particular array of names. Only those arrays required for the report format are initialised.

Once the array of names has been initialised then the id can be found by searching through the array.

However, if the name is a command then there is a problem, because the command names do not have numerical ids associated with them and can not be given set numerical ids because there are constantly being more command names created.

To overcome this problem a function was written to convert a ten character command name into a unique numerical id. This function must be as efficient as possible because it will be used by the File Accessor for every selected accounting record so that the Report Generator is only ever given a numerical id for each selected subject of the report.

The chosen solution follows.

**Command Name Hashing**

Assign the value of the first four characters of the name to an integer as follows :-

\[
\begin{align*}
\text{bit 1} & = 0, /* \text{make number id positive */} \\
\text{bits 2-8} & = \text{character 1}, \\
\text{bits 9-15} & = \text{character 2}, \\
\text{bits 16-22} & = \text{character 3}, \\
\text{bits 23-29} & = \text{character 4}, \\
\text{bits 30-32} & = 0. /* \text{length of name } \leq 4 ? */
\end{align*}
\]

If length of name is greater than four then
Hash the integer created into a table of arrays of names.
If the name is not in the array then
Add name to next free array element.
Add position of name in the array to the integer value as follows :-

\[
\begin{align*}
\text{bits 30-32} & = \text{position in the array}.
\end{align*}
\]
This solution was chosen because the majority of commands are four or less characters long, and therefore their conversion will just involve four assignments, instead of the slower methods that involve searching through data structures using string comparisons.

Convert Ids to Names

Description

This procedure converts the numerical id of a command, a user, a group or a terminal into its corresponding name and returns a pointer to this name. If the id is not found a NULL pointer is returned.

Definition

\[
\text{get name}(id, i\_t\_y\_p\_e) \\
\text{int id;} \\
\text{ID\_TYPE id\_type;}
\]

"id" is the numerical id of a "id\_type", where "id\_type" can be a user, group, command or terminal.

Data Structures

The above mentioned arrays of user, group and terminal names are used by this procedure.

This procedure also uses the structure ID\_UN\_TYPE mentioned above and the same hash table that is used by "get\_id".

Implementation

If the id passed to the procedure is a user, group or terminal and this is the first reference to an id of this type then "init\_arrays" is called to initialise the particular array of names. This is done to avoid any unnecessary initialisation.

Once the array of names has been initialised then the name can be found in by indexing into the array with the particular numerical id.

However, if the name is a command then the command name must be obtained from the numerical id by the reverse of the hashing process used by "get\_id".
Get a Users Group Id

Description
This procedure returns the group id of the given user.

Definition
get_group_id(id)
int id;

"id" is a user's numerical id.

Data Structures
This procedure accesses the array of group ids created by "init_arrays".

Implementation
If this is the first reference to the group ids array or the user names array which are both initialised at the same time, then "init_arrays" is called to initialise the two arrays.

Then once the array is initialised the procedure simply looks up the array of group ids to find the group id of the given user.
5. REPORT GENERATOR

Overview

This subsystem collects the data for a report and sorts and prints the report. It is responsible for formatting the report appropriately for the particular output device being used.

The sections involved in this sub-system are :-

1) Initialisation,
2) Building,
3) Pruning,
4) Sorting,
5) Printing,
6) Re-use.

The section Initialisation starts up and defines the subject of the report. The section Building collects information for the report, while Pruning selects only those records that fit the content limitations. Sorting and Printing sort and print the report. The section Re-use redisplays a report without re-building it.

To understand how this sub-system works the concept of report dimension must first be explained.

Report Dimension

The basic type of report is a simple list. An example of this is the following:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>nkh</td>
<td>10</td>
</tr>
<tr>
<td>nlc</td>
<td>53</td>
</tr>
<tr>
<td>skh</td>
<td>29</td>
</tr>
</tbody>
</table>

The leftmost column is the key of the report, and its elements are called ids. All of the ids are members of the categorisation domain of the report. The categorisation domain of this report is user. This can also be described as a one dimensional report.

A two dimensional report, categorised on user and group can be thought of as follows:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cs3</td>
<td>E321</td>
<td>staff</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nkh</td>
<td>32</td>
<td>21</td>
</tr>
<tr>
<td>skh</td>
<td>12</td>
<td>23</td>
</tr>
</tbody>
</table>
The categorisation domains for this report are user and group.

The concept of report dimension can be generalised to that of an n dimensional report. An n dimensional report has a categorisation domain for each dimension. For the above example, the categorisation domains are user name and user group. An example of a three dimensional report is one categorised by date, user and group. This would give a daily report on each user’s groups.

The output format of a report is, thus, dependent on the dimension of the report.

A ternary tree can be used to manipulate an n dimensional report.

Data Structures

This subsystem uses a ternary tree to store the report before printing. The third pointer in each node points to a subtree with a different categorisation domain. Thus the tree has a level for each field categorised in the report. An advantage of this data structure is that the storage of a report is very flexible. There is no limit (apart from the memory available to the program) on the number of records in a report. The tree expands dynamically to store all required information.

A node in the tree contains the fields:

   id, pruned, data, left, right, down, sup.

For the above example, there is a first level tree containing the user ids ‘nkh’ and ‘skh’. Each of these users is represented by a node. These user nodes contain pointers to subtrees, which store the data for the groups.

Calculation of totals and subtotals, and selection of records is simplified as all data is in storage and a simple tree traversal will allow processing of all records. Sorting is simplified as a heapsort can be used without moving any data - only the pointers between them are changed.

A picture of the above example as a tree is:
When a report is categorised on individual records then the tree degenerates into a linked list. If a recursive algorithm is to be used for tree traversal then this may overflow the fixed size stack. A non-recursive algorithm is used to avoid this problem. It is based on the triple order mechanism for tree traversal as described by Dr. Dromey in his book "How to Solve it by Computer". This algorithm uses cyclic pointer modification to traverse a binary tree in any of the three standard tree traversal orders. The left and right pointers of the tree nodes are modified as the algorithm progresses to allow return to higher level nodes. Any null pointers are also set to be reflexive, that is, point to the node that they are fields in.

By introducing a pointer to the immediate superior node in each node (called sup) there is no need to modify the tree to maintain this information during the progress of the traversal. Another point in favour of this method is that a traversal can be aborted part way through without harming the tree's integrity.

This tree traversal algorithm can be implemented in two functions first and next. "First" finds the first node which would be processed if any one of the three traversals were to be performed, given the current node only. As this algorithm is more complex than that of first it is given below:
ORDER = (PREORDER, INORDER, POSTORDER);
next(current:TREE,order:ORDER) returns TREE;
        variable state : ORDER := order;
begin
    while (current <> nil) loop
        (* until state matches order *)
        case state of
            PREORDER:
                if (current->left = nil)
                    state := INORDER;
                else
                    current := current->left;
                end if
            INORDER:
                if (current->right = nil)
                    state := POSTORDER;
                else
                    current := current->right;
                    state := PREORDER;
                end if
            POSTORDER:
                if (current->sup = nil)
                    return nil;
                end if;
                if (current->sup->left = current)
                    state := INORDER;
                end if;
                current := current->sup;
        end case;
        if (state = order)
            (* reached the next node *)
        return current;
        end if;
    end loop;
return nil;
end next;

This allows tree traversals to be simple 'for' loops (in C):

    for (node = first(root,PREORDER), node != NULL;
        node = next(node,PREORDER))
    activity(node);

When the number of users is required, then an additional level of the
tree is built, called the user tree. This subtree contains only user ids. This minimizes storage requirements.

Initialisation

Description

The report generator must be told, before any data is read in, what the subjects of the report are. Calling this function will also remove the old report so that a new one can be generated.
Definition

start rep(subjects,user_used)
VARIABLE subjects[];
BOOLEAN user_used;

"subjects" is a list of categorisation domains. "user_used" is true if number of users, or an average per user is required.

Implementation

The subjects are stored away, and the tree is set to be empty. The user_used flag is used to determine whether a user tree must be built for each lowest level node. A user tree holds the users for a node, so for example if a report's categorisation domain is group then a user tree is constructed for each group so that the number of users in each group can be determined.

Building

Description

This function is called from the Accounting File Accessor subsystem to save every record which is to be included in the report. There are three types of record to save:

1) COMMAND: contains information on a call of a command,
2) TIME: contains connect information,
3) TERMINAL: contains login information.

Definition

save(rec)
REP_DATA_STRUCT rec;

REP_DATA_STRUCT is described in the Accounting Files Accessor subsystem.

Implementation

A tree, as described above, is built from records from the accounting files. All three types of save record are built into one tree, so that if a report is required showing both total CPU time, and total connect time, then both quantities can be found in the same tree.

Each record saved contains a number of ids, for example, user id, and date. The save function finds the node in the topmost level, whose id is the same as the appropriate id in the save record. If no node exists with the appropriate id then a new node is created. If this is not the lowest level of the tree then the down pointer is followed to the subtree where
the process is repeated. Saved values are stored in the bottom-most levels of the tree only.

All nodes are initialised to be unpruned. When the tree has been completely built then a function sum() is called to calculate subtotals and totals. The sum of each variable in each tree is found and stored as the value of the parent node of that tree. A zeroeth level node is created to hold the grandtotals of the report.

**Pruning**

**Description**

Pruning is the process of removing records from a report to ensure that the only records in that report are those selected by the user. A record which is pruned is not displayed on a report, and is not included when totals are calculated.

**Definition**

\[
\text{prune(field,condition,value)}
\]

VARIABLE field;
COMP_TYPE condition;
float value;

If the specified "field" does not obey the specified "condition" then the record is pruned. The "condition" can be either MIN or MAX. If MIN is selected then only those records whose field is greater than "value" are selected.

**Implementation**

Records are removed from the report by marking all lowest level nodes of the tree which fail to meet a given condition. The tree is traversed by inorder traversal. If a lowest level node fails to meet the condition then the node is marked by setting pruned to true. If all of a subtree is pruned then the parent node is also pruned. A node which is pruned is not displayed on a report. Totals are recalculated (using sum()) after pruning and nodes pruned are not included in these totals.

**Sorting**

**Description**

The report can be sorted on a list of keys in either ascending or descending order.
Definition

\[
\text{sort(}\text{fields,}\text{field_direction})
\]

VARIABLE fields[];
COMP_TYPE field_direction[];

The report is sorted on "fields", in the direction given in "field_direction". If an entry in "field_direction" is \text{MIN} then the report is sorted with minimum values first, that is, in ascending order.

Implementation

All the subtrees which make up the tree are sorted on a list of fields. A heapsort is used to perform the sort, but the sort is done in place — no nodes are moved, the pointers between them are changed to indicate the new ordering. To improve performance, a check is first done to see if the tree is already in order (a common case, e.g. when a time level is involved), and no sort is carried out if this is the case. This initial checking phase also propagates the sorting to all levels of the tree.

Printing

Description

This function prints out a finished report, which has been saved, pruned, and sorted by the preceding functions. It allows a report to be directed to one of several output devices: to the screen, the printer, or into a file. The report is reformatted to suit the particular media. If required, a graph will accompany the report. The report may have a limit on the number of lines to be printed.

Definition

\[
\text{print(}\text{headers,}\text{out,}\text{linelimit,}\text{fields,}\text{gr})
\]

STRING headers;
DESTINATION out;
int linelimit;
VARIABLE fields[];
BOOLEAN gr;

The "headers" are printed before a report is displayed. The destination of the report, SCREEN, PRINTER, or FILE is given in "out". A report is limited in length by a positive "linelength". The "fields" are the fields to be displayed for each output record. An accompanying graph is produced when "gr" is TRUE.
Implementation

The tree is toured by inorder traversal and an output record is produced for each node in the tree with some exceptions. An output record contains the value of the node's id, and the values of the printing fields for the node.

Headings are printed at the top of a report and when a new page is started and the report is printed on a printer. The headings consist of subject information from the System Co-ordinator and labels for each column.

Nodes not at the lowest level are printed in two parts: firstly the node's id is printed; then all of its subtree is printed; lastly, the values of the fields are printed. Note that the values of a higher level node are the totals of its subtree.

If a graph is being produced then for all records the printing fields are first printed, and then a row of asterisks is printed to indicate the relative magnitude of the first printing field.

When a report of dimension greater than one is produced then a special format is used with offset totals. This is done so that a reader can add up columns with the last line being the total of all above. This is illustrated:

<table>
<thead>
<tr>
<th>User</th>
<th>Group</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>nkh</td>
<td>cs3</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>E321</td>
<td>21</td>
</tr>
<tr>
<td>skh</td>
<td>cs3</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>E321</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>staff</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>107</td>
</tr>
</tbody>
</table>

Re-use

Description

This function is used to avoid reading data twice from the accounting files. It is called when the subjects of a report are the same as the previous report.

Definition

    reset_rep()
Implementation

Limited re-use of the tree is possible when the tree is unchanged. If a report is to be produced which is very similar to the previous report, differing only in the output destination, sort fields, or whether a graph is to be displayed, then the old tree can be re-used. The function reset_rep is called, which sets all nodes to unpruned. The appropriate sort, prune, and print calls can then be used to print the report without reading the data from the files again.
Archiving is the process of storing away the current data for later analysis. This system does not contain any computer procedures to provide archiving. The accounting files are stored on tape every week and truncated. When a period's data is to be analysed then it is read back from tape into a disk file and QUACC used upon that file. This facility would mostly be used at the end of year to provide annual summaries, at this time of year there is enough disk space available to hold all of a year's data (approximately 100 megabytes).

Alternatively, the entire year's file could be permanently stored online to allow analysis at any time. With small changes to the QUACC system the year's data could be stored on tape and processed when desired. Reading from tape may even be faster than reading from disk in this case as the file is read sequentially when all data must be examined.

By archiving all data on tape QUACC improves on many of the schemes used to archive the accounting data such as the -s option of sa (which saves a summary list of commands with data on each command). These schemes rely on storing a summary of the available accounting data. Using s on archived data allows queries such as finding all the commands issued by group cs3. If only summary files are available then this type of information will be lost and the system not be fully general.
7. WEEKLY SUMMARY REPORTS

The current weekly reports can be implemented as command files for quacc. There are however slight differences between the way quacc does things and the way the current systems work.

Note that an asterisk ('*') introduces a comment in a command file.

* cmd.weekly
* This file generates a report which is similar to the current command report.
* There are some small differences between this report and the current report: the current report is sorted on the total of CPU and system time, but "quacc" cannot do this. A program which executed a fork but did not execute an exec is flagged in the current report with a star as a separate command but quacc does not do this (this commonly occurs with the shell).

N LINES 110
CAT COMMAND
PRT NUM_COMMANDS AVG_REL AVG_CPU AVG_SYS T_CPU SRT T_CPU dcr

* user.weekly
* This file generates a report which is similar to the current user report.
CAT USER GROUP
PRT CON_TIME NUM_COMMANDS T_CPU SRT T_CPU max

* group.weekly
* This file generates a report which is similar to the current group report.
CAT GROUP
PRT NUM_USERS CON_TIME NUM_COMMANDS T_CPU UA_CPU

* term.weekly
* This file generates a report which is similar to the current terminal usage report.
CAT DATE TIME
GPH TERMS
8. IMPROVEMENTS

Retrieval From Login/Logout File

The file of login/logout information is read every time a report that requires information from it is generated. As it is read, the tree of login information which covers only the time period of interest to the report is built.

Rather than building this tree each time a report is generated and deleting it once the report has been generated, a tree that covers the entire period that the login/logout file covers could be built when QUACC is initially invoked. Then this tree can be used as the source of login/logout information for every report that is generated from one invocation of QUACC.

This would reduce the time taken to produce each report, as the login/logout file would only be read once at the beginning rather than each time.

A NOT Operator for Limitations

It should be possible to ask for a report on all commands used by people not in group csl, or a report on all commands executed this week excluding the editors ed, edit, and re. The NOT operator could be an initial exclamation mark ('!') as the first character entered in a limitation.

Changing User Ids

When a user's id is deleted from the password file or the numerical id of a given user is changed, problems can occur when reading archived accounting files.

The first problem is that it is possible for a numerical user id not to have a user associated with it in the current password file.

The second problem is that if a user's numerical id has changed then the archived accounting information will be associated with the wrong user.

To overcome this problem a historical password file should be kept to allow correct interpretation of archived data.
APPENDIX 1

Possible Content Menus:

TIME MENU
* total CPU time
* average CPU time per call
* total system time
* average system time per call
* total real time
* average real time per call
* number of terminals in use
* number of commands executed

COMMAND MENU
* total CPU time
* average CPU time per call
* total system time
* average system time per call
* total real time
* average real time per call
* number of commands executed

USER MENU
* total CPU time
* average CPU time per call
* total system time
* average system time per call
* total real time
* average real time per call
* total connect time
* number of times connected
* number of commands executed

GROUP MENU
* total CPU time
* average CPU time per user
* total system time
* average system time per user
* total real time
* average real time per user
* total connect time
* number of times connected
* number of users in group
* number of commands executed

INDIVIDUAL RECORDS MENU
* total CPU time
* total system time
* total real time
### APPENDIX 2

An example of the USER MENU input file:

<table>
<thead>
<tr>
<th>Field string</th>
<th>String</th>
<th>Field</th>
<th>Buffer</th>
<th>Buffer</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Total CPU time...................</td>
<td>3 40</td>
<td>14 3</td>
<td>70 9</td>
<td>h.real</td>
<td></td>
</tr>
<tr>
<td>* Avg CPU time / command..........</td>
<td>4 40</td>
<td>17 4</td>
<td>70 9</td>
<td>h.real</td>
<td></td>
</tr>
<tr>
<td>* Total system time...............</td>
<td>5 40</td>
<td>15 5</td>
<td>70 9</td>
<td>h.real</td>
<td></td>
</tr>
<tr>
<td>* Avg system time / command......</td>
<td>6 40</td>
<td>18 6</td>
<td>70 9</td>
<td>h.real</td>
<td></td>
</tr>
<tr>
<td>* Total real time..................</td>
<td>7 40</td>
<td>16 7</td>
<td>70 9</td>
<td>h.real</td>
<td></td>
</tr>
<tr>
<td>* Avg real time / command.........</td>
<td>8 40</td>
<td>19 8</td>
<td>70 9</td>
<td>h.real</td>
<td></td>
</tr>
<tr>
<td>* Total connect time..............</td>
<td>9 40</td>
<td>12 9</td>
<td>70 9</td>
<td>h.int</td>
<td></td>
</tr>
<tr>
<td>* Number of times connected......</td>
<td>10 40</td>
<td>9 10</td>
<td>70 9</td>
<td>h.int</td>
<td></td>
</tr>
<tr>
<td>* Number commands executed......</td>
<td>11 40</td>
<td>23 11</td>
<td>70 9</td>
<td>h.int</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX 3

Introduction

The sub-systems communicate by passing various structures to one another. These structures describe the requirements of the report and are explained below.

Structures Sent By User Interface

The User Interface sets up a summary of all the restrictions on the report in the structure INPUT_TYPE. It has the form:

```
structure INPUT_TYPE
    - structure SUBJ_LIMITS
    - structure CONT_LIMITS
    - structure OUT_LIMITS
    - list of the fields to be printed
    - list of the fields to be sorted and their ordering
    - the field to be graphed
```

The structure SUBJ_LIMITS contains the restrictions on the subject of the report. It has the form:

```
structure SUBJ_LIMITS
    - minimum date
    - maximum date
    - minimum time
    - maximum time
    - list of user ids
    - list of group ids
    - list of command ids
    - list of terminal ids
```

The structure CONT_LIMITS contains the restrictions on the content of the report. It has the form:

```
structure CONT_LIMITS
    - list of the limitations on the content fields
```

The structure OUT_LIMITS contains the restrictions on the destination of the report. It has the form:

```
structure OUT_LIMITS
    - the type of destination which can be either
      screen or
      printer or
      a file
```

The User Interface sends the structure INPUT_TYPE to the System Co-Ordinator Sub-System.
Structures Sent By System Co-Ordinator

The System Co-Ordinator sends the structure SUBJ LIMITS, received from the User Interface in the structure INPUT_TYPE, to the Accounting Files Accessor Sub-System.

The remaining parts of the structure INPUT_TYPE are divided up and the relevant parts sent to the different Report Generator routines.

Structures Sent By Accounting Files Accessor

The Accounting Files Accessor sends the data retrieved from the accounting files to the Report Generator Sub-System via the structure REP_DATA_STRUCT. This structure has the form:

```
structure REP_DATA_STRUCT
  - the type of data which can be either
    acct file data or
    connect information or
    terminals connected per hour data
  - user id
  - group id
  - command id
  - year
  - month
  - day
  - hour
  - user time
  - system time
  - elapsed time
  - terminal id
  - connect time
  - number of terminals
```
### S = SUBJECT MENU

- DATE
- TIME
- COMMAND
- GROUP
- CPU
- TIME
- USER
- CPU
- TIME
- SYSTEM
- CPU
- TIME
- TOTAL
- CPU
- TIME
- AVG
- CPU
- TIME
- /user
- GROUP
- IN
- REPORT
- DECT
- LINES
- IN
- REPORT
- PRINTING
- SORTING

#### G = CONTENT MENU

- TOTAL
- CPU
- TIME
- AVG
- CPU
- TIME
- /user
- TOTAL
- CPU
- TIME
- AVG
- CPU
- TIME
- /user
- NUMBER
- OF
- COMMAND
- EXECUTED
- NUMBER
- OF
- USERS
- IN
- GROUP
- NUMBER
- OF
- TIMES
- CONNECTED
- TOTAL
- CONNECT
- TIME

#### O = OUTPUT MENU

- ERROR
- MESSAGES
- DISPLAYED
- HERE
- USER
- ASSIGNMENT
- DISPLAYED
- HERE

---

The content menu shown in the above screen layout is that of the 'GROUP MENU'. Other menus for the content of the report will begin at the same position. The option list displayed at line 22 is that for the content menu. The option placed in the auxiliary block shows its position when the subject menu is active.
NAME
quacc - interactively query system accounting data

SYNOPSIS
quacc [ [ option1 file1 ] [ option2 file2 ] ... ]

DESCRIPTION
Quacc creates reports on the usage and performance of the UNIX system with regard to users, groups, commands, and terminals. It supplies an interactive menu through which the contents and format of the report may be specified.

For commonly used report types, the format may be stored in a file, eliminating the need to tailor this format every time it is required.

The following options are recognised.

- c file Build up the report format from file, bypassing the use of the menus. This file must have been created from a previous interactive run of Quacc.
- a file Read all process termination information from file, instead of the default file /usr/adm/acct.
- w file Read all login and logout information from file, instead of the default file /usr/adm/wtmp.

FILES
/usr/adm/acct (default accounting info file)
/usr/adm/wtmp (default login/logout file)
/dev ... 
/etc/passwd
/etc/group

AUTHORS
Nicole Cooper
Ray Coury
Steve Hegyi
Neil Hodgson

SEE ALSO
acct(5)
utmp(5)

BUGS
If information is to be retrieved from both accounting files, then the information from the login/logout file will only be retrieved if the login/logout file covers the same time period as the period for which information is retrieved from the process termination file.
USER MANUAL for QUACC

A QUERY and REPORTING SYSTEM

for the UNIX SYSTEM ACCOUNTING DATA

GROUP E

Nicole Cooper
Ray Coury
Steve Hegyi
Neil Hodgson

The University of Wollongong

ABSTRACT

This document is the user manual for QUACC, an interactive screen-based query and reporting system for accounting data collected by the UNIX system on the Computing Science Perkin-Elmer computers.
CONTENTS

1. INTRODUCTION
2. INVOKING QUACC
3. SCREEN LAYOUT
4. CURSOR MOVEMENT
5. SUBJECT MENU
   Enter Limitation
   Categorise
6. CONTENT MENU
   Enter Limitation
   Print
   Graph
   Sort
7. OUTPUT MENU
   Report Destination
   Number of Lines in Report
8. REPORT FORMAT AREA
   Sorting Format
   Printing Format
9. OPTIONS and FUNCTIONS AREA
   Options
   Functions
     Help
     Reset
     Unix Command
     Exit
     Generate Report
     Save Report Format
10. REPORT OUTPUT
    Screen
    Printer
    File
11. GENERATE REPORT CYCLE
12. FORMAT of SUBJECT LIMITATIONS
1. INTRODUCTION

QUACC (QUery ACCounting) is an interactive screen based system that provides the facilities for answering queries about the performance and usage of the UNIX system on the Computing Science Perkin Elmer computers.

QUACC can be used to generate a wide range of reports about the activities of users and groups, and the usage of commands and terminals on the UNIX system.

2. INVOKING QUACC

QUACC is invoked by entering the command:

\[ \text{quacc} \ [\ -c \ \text{file1}\ ] \ [\ -a \ \text{file2}\ ] \ [\ -w \ \text{file3}\ ] \]

If QUACC is invoked without any parameters then QUACC sets up an interactive screen which allows a report format to be selected and a report to be generated. The reports are created from files that are maintained by the UNIX system. These files are /usr/adm/acct, which contains information about each process that terminates, and /usr/adm/wtmp, which contains information about each login and logout that has occurred.

However, if the '-a' option is specified when QUACC is invoked, for example:

\[ \text{quacc} \ -a \ /\text{pub/adm/acct.weekly} \]

then QUACC creates the report from the information in the given file, rather than that in the default file.

If the '-w' option is specified, for example:

\[ \text{quacc} \ -w \ /\text{usr/wtmp.month} \]

then QUACC creates the report from the information in the given file, rather than that in the default file.

If the '-c' option is specified, for example:

\[ \text{quacc} \ -c \ /\text{reps/fortnightly} \]

then the given file is treated as a command file. The report format is read from this file, instead of interactively, and the report generated automatically.

The command file must have previously been created by an interactive run of QUACC.

Note that the options in the call to QUACC may be specified in any order.
3. SCREEN LAYOUT

When QUACC is invoked interactively the user-interface screen is displayed. This consists of the following sections:

1. Subject menu,
2. Content menu,
3. Output menu,
4. Report Format Area,
5. Options and Functions Area.

The layout of the user-interface screen follows:

\[
\begin{array}{c|c}
\text{s = SUBJECT MENU} & \text{c = CONTENT MENU} \\
\hline
\text{DATE} & \text{Total CPU time} \\
\text{TIME} & \text{Avg CPU time / command} \\
\text{COMMAND} & \text{Total system time} \\
\text{USER-ID} & \text{Avg system time / command} \\
\text{GROUP-ID} & \text{Total real time} \\
\text{TERM-ID} & \text{Avg real time / command} \\
\text{RECORDS} & \text{Total connect time} \\
\hline
\text{o = OUTPUT MENU} & \text{Number of times connected} \\
\text{REPORT DEST'N} & \text{Number commands executed} \\
\text{LINES IN REPORT} & \\
\end{array}
\]

Report Format Area

Options and Functions Area

The top area of the screen is the Menu Area, and is where the query requirements are entered.

The next area of the screen is the Report Format Area, where information about the final report format is displayed.

The bottom area of the screen is the Options and Functions Area, and contains a list of the selections available in the current menu field.
4. CURSOR MOVEMENT

It is possible to move between the different menus of the system by selecting the name of the menu from one of the following:

\[ s = \text{SUBJECT MENU} \]
\[ c = \text{CONTENT MENU} \]
\[ o = \text{OUTPUT MENU} \]

Selecting a menu name will make that menu the "active" menu. The active menu is the menu that is currently available for the entering of inputs. Initially the Subject menu is the active menu.

The current field of a menu is the field the cursor is on. When in a menu the cursor is placed at the left of the current field.

Within each menu it is possible to move down the fields by typing a down arrow or a carriage return character. To move up the fields the up arrow character is typed. The fields of each menu wrap around, so that from the top field the bottom field may be moved to by typing the up arrow, and vice versa.
5. SUBJECT MENU

The Subject Menu is the primary menu of the system and is displayed at all times during the tailoring of a report. It is where the subject of the report is entered.

The Subject Menu consists of the following fields:-

* DATE......
* TIME......
* COMMAND...
* USER-ID...
* GROUP-ID..
* TERM-ID...
* RECORDS...

There are two options available for selection in this menu. They are :-

1 = enter a limitation
* = categorise

Enter a Limitation

To restrict a field to a particular value "1" is typed, which QUACC acknowledges by moving the cursor to the right of the field ready for the input of a limitation. The limitation can then be entered and terminated by a carriage return.

Examples of limitations are the date "12/11/84" in the DATE field and the users "skh nkh nlc rpc" in the USER field. For a detailed explanation of the possible limitations for each field see section 12, Format of Subject Limitations.

Categorise

Categorisation is a very important concept to grasp in order to fully understand how to specify a query using QUACC. If the user wants each unique value described in a field to be printed in the report with any associated information beside it, then the field must be categorised on.

For example, categorising on users (with the limitation "nlc rpc" in the USER field) would produce a report of the following form :-

<table>
<thead>
<tr>
<th>User</th>
<th># Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlc</td>
<td>8</td>
</tr>
<tr>
<td>rpc</td>
<td>19</td>
</tr>
<tr>
<td>Totals</td>
<td>27</td>
</tr>
</tbody>
</table>

However, if the user field is not categorised on, then the
report will only contain total information for the selected users:

\[ \text{# Calls} \]
\[ 27 \]

Categorisation may be done on any number of these fields. So if user and command fields were selected for categorisation, in that order, a report of the following form is generated:

<table>
<thead>
<tr>
<th>User</th>
<th>Command</th>
<th># Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlc</td>
<td>cat</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>ls</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>opr</td>
<td>1</td>
</tr>
<tr>
<td>rpc</td>
<td>cat</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>cc</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>opr</td>
<td>2</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>27</td>
</tr>
</tbody>
</table>

The order of categorisation is important! The order in which the fields are selected determines the printing order. For example, if the previous example were categorised by command and then by user the report would have this form:

<table>
<thead>
<tr>
<th>Command</th>
<th>User</th>
<th># Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>nlc</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>rpc</td>
<td>12</td>
</tr>
<tr>
<td>cc</td>
<td>rpc</td>
<td>5</td>
</tr>
<tr>
<td>ls</td>
<td>nlc</td>
<td>5</td>
</tr>
<tr>
<td>opr</td>
<td>nlc</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>rpc</td>
<td>2</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>27</td>
</tr>
</tbody>
</table>

When categorising on date is selected, QUACC will generate a daily report, and for categorisation on time an hourly report is produced.

For categorisation on command, group and terminal the report will be of a similar form as shown for categorisation by the user field above.

If categorisation by the records field is selected then the report will contain information on every individual record from the accounting file that fits the rest of the subject requirements.
Categorisation on records can be used for analysing data in more detail than the summarised reports give. For example, it can be used for observing the activities of a particular student by printing every command that he or she executed within a given time.

An example of a report categorised by records then command is:

<table>
<thead>
<tr>
<th>Record</th>
<th>Command</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:03.1</td>
<td>cat</td>
<td>0.3</td>
</tr>
<tr>
<td>12:05.8</td>
<td>ls</td>
<td>0.2</td>
</tr>
<tr>
<td>12:10.1</td>
<td>cat</td>
<td>0.1</td>
</tr>
<tr>
<td>12:23.1</td>
<td>cat</td>
<td>0.1</td>
</tr>
<tr>
<td>12:48.1</td>
<td>ls</td>
<td>0.1</td>
</tr>
<tr>
<td>13:06.0</td>
<td>ls</td>
<td>0.2</td>
</tr>
<tr>
<td>13:11.3</td>
<td>ls</td>
<td>1.4</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>2.4</td>
</tr>
</tbody>
</table>

This is different from not categorising on records which only produces a summary of the selected data.

For example, the above report not categorised on records would give:

<table>
<thead>
<tr>
<th>Command</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>0.5</td>
</tr>
<tr>
<td>ls</td>
<td>1.9</td>
</tr>
<tr>
<td>Totals</td>
<td>2.4</td>
</tr>
</tbody>
</table>
6. CONTENT MENU

Once the subject of the query has been entered, the Content Menu can be moved to. Here, the contents of the report can be selected.

The fields displayed in this menu vary, depending on which fields are selected in the Subject Menu. This is because some content fields such as "total connect time" apply to some subjects (eg. users and groups), but not to other subjects (eg. commands).

The Content Menu that is displayed for a user report is:

* Total CPU time
* Avg CPU time / command
* Total system time
* Avg system time / command
* Total real time
* Avg real time / command
* Total connect time
* Number of times connected
* Number commands executed

There are a number of options available for selection in this menu. They are:

1 = enter a limitation
p = print
g = graph
d = decreasing sort
i = increasing sort

**Enter a Limitation**

The same principles of this option apply to the Content Menu as for the Subject Menu. The difference is that since all the fields in the Content Menu only have numerical values associated with them, the only limitations available are "<d" or ">d" where "d" is an integer or floating point value.

**Print**

If "p" is selected then the current field of the Content Menu is marked for printing, and the field name is entered in the Report Format Area of the screen which is explained below.

If "p" is selected in a field that is already being printed then the field will no longer be printed.
Graph

It is possible to graph any field in the Content Menu by entering the option 'g'. If 'g' is selected in a field that is already being graphed then the field will no longer be graphed.

It is only possible to generate a report with one field graphed, so that if graphing is selected, the only option available on the other fields of the Content Menu is "enter a limitation".

An example of a graphed report is one in which the terminal field is categorised upon to produce a report on each terminal, and connect time is graphed. The form of this report is:

<table>
<thead>
<tr>
<th>Terminal</th>
<th>ConTime</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>tty74a</td>
<td>35m24</td>
<td></td>
</tr>
<tr>
<td>tty81a</td>
<td>18m51</td>
<td></td>
</tr>
<tr>
<td>tty83c</td>
<td>1h00m01</td>
<td></td>
</tr>
<tr>
<td>tty91c</td>
<td>31m11</td>
<td></td>
</tr>
<tr>
<td>tty51a</td>
<td>5m16</td>
<td></td>
</tr>
<tr>
<td>Totals 2h30m43</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This type of report is useful for checking terminal usage for terminals, particularly those terminals not in the computing laboratories.

Sort

It is possible to sort the report on any of the Content Menu fields, and to sort on up to four of these fields in either increasing or decreasing order.

If no Content Menu fields are selected for sorting then the report will be sorted on the Subject Menu fields selected for categorising.

If sorting is selected and the field is not already being printed then the field will be marked for printing. This is because it is meaningless to sort on a field that is not printed.

A list of the fields that are sorted are displayed in the Report Format Area in a similar fashion to the printed fields.

If 'd' or 'i' are selected in a field that is already being sorted then:

1) if the field was already being sorted in the selected direction then the field will no longer be sorted,

2) if the field was being sorted in the opposite direction to the selected direction then the sorting direction will be changed to the selected direction.
7. OUTPUT MENU

The Output Menu is displayed at all times of report tailoring, and can be moved to at any time. Information on the destination and size of the final report can be entered in this menu.

The fields in the Output Menu are:

* REPORT DEST'N...
* LINES IN REPORT...

The only option available in this section is:

1 = enter a limitation

Report Destination

There are three possible limitations that may be entered for the report destination. They are:

1) the screen ("s"),
2) the printer ("p"),
3) a specified file.

If no report destination is entered then the report will be output to the screen.

Number of Lines in Report

An integer value specifying the maximum number of lines in the report can be entered in the LINES IN REPORT field. This is useful when combined with sorting, for obtaining such reports as only the top ten CPU users.
8. REPORT FORMAT AREA

An area of the screen below the menus contains the current format of the requested report. This includes a list of the fields that will be printed in the final report and a list of the fields that will be sorted in the final report.

**Printing Format**

A list of the content fields that have been selected for printing is displayed in the Report Format Area, along side any fields that are categorised. This shows the headings that will appear in the final report. This allows fields of the report to be printed in any order, by selecting them in the desirable order.

For example, if Subject Menu fields command and user are selected for categorisation and the Content Menu fields:

- Total CPU Time
- Total Real Time
- Number of Calls to Commands

are selected for printing (in that order) then the Report Format Area will appear as follows:

```
PRINTING
Command User CPU Real # Calls
```

**Sorting Format**

A list of the fields that are sorted are displayed in the Report Format Area in a similar fashion to the printed fields. The order of selection of the sorting fields is important, with the first selected fields having the highest priority. This is reflected by the order in which they are displayed.

For example, if Content Menu fields:

- Number of Calls to Commands
- Total CPU Time

are selected for increasing sorting and decreasing sorting, respectively, then the Report Format Area will appear as follows:

```
SORTING
# Calls (i) CPU (d)
```

In this case the report will be sorted on "Number of Calls to Commands" and for any lines in the report that have the same number of calls, the lines will be sorted on their "Total CPU Time".
9. OPTIONS and FUNCTIONS AREA

The bottom part of the screen contains a list of all of the possible selections that may be requested. These selections include a list of all of the options that can be applied to the active menu, and a list of all the general functions that can be invoked at any time during a session with QUACC.

Options

All of the different options have been described above, in the sections on the three different menus.

However, it should be noted that options can only be selected when the cursor is on the left of the required field. So that when a limitation is being entered (on the right of the field name), no other options may be entered until entry of the limitation is completed.

Functions

At all times during report tailoring there are a number of functions that may be invoked at any time. They are :-

? = Help
^R = Reset
^C = Unix Command
^D = Exit
^G = Generate Report
^S = Save Report Format

Help

Help may be requested at any time. When requested, a help message will be displayed in the Report Format Area giving either a summary of all of the possible inputs that may be entered in the current field, or an explanation of each option which can be applied to the current field.

Reset

If '^R' is selected then the current report format will be cleared and a new query specification may be entered.

Unix Command

It is possible to enter a Unix command by entering '^C'. If you wish to enter more than one Unix command then the Unix command '^sh' should be entered, which can be terminated by '^D'.
Exit

It is possible to exit from QUACC at any time, even during the generation of a report, by entering "~D".

Generate Report

Once the report format has been entered, "~G" may be selected to generate the report. The system reads the necessary information from the accounting files and an indication of the progress being made is displayed on the screen between the menus and the Report Format Area. This indication consists of a horizontal bar that decreases in magnitude in proportion to the time left to generate the report.

Save Report Format

The user can save the format of a desired query into a "command" file using the "~S" Function. Then QUACC can be called at a later date, with this file as a parameter, using the -c option. When this is done the interaction with the user is by-passed and the user's inputs are read from the command file. The report is generated and sent to the destination specified in the command file.

This function allows standard report formats such as weekly reports and daily reports to be generated from shell.

It is also useful for running QUACC in the background.
10. REPORT OUTPUT

The report output is easily readable, with the only explanation required being that time unit entries are in one of the following forms:

1) 9999h  (9999 hours)
2) 9h99m99  (9 hours, 99 minutes, 99 seconds)
3) 99m99  (99 minutes, 99 seconds)
4) 99.9  (99.9 seconds)

An example of a report showing the form of the output is the following report, categorised by user:

<table>
<thead>
<tr>
<th>User</th>
<th>CPU</th>
<th>System</th>
<th>Real</th>
<th>ConTime</th>
<th>Conncts</th>
<th># Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-----</td>
<td>--------</td>
<td>------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>nkh</td>
<td>23.0</td>
<td>34.3</td>
<td>1m05</td>
<td>4m24</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>nlc</td>
<td>2m21</td>
<td>2.7</td>
<td>20m14</td>
<td>22m00</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>rpc</td>
<td>5.4</td>
<td>2m29</td>
<td>5m12</td>
<td>50m21</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>skh</td>
<td>1h23m33</td>
<td>45m55</td>
<td>3h00m10</td>
<td>3h20m30</td>
<td>3</td>
<td>104</td>
</tr>
</tbody>
</table>

Totals: 1h26m22 49m01 3h26m41 4h37m15 5 129

The report output varies slightly depending on the report destination.

Screen

If the report is output to the screen the report is paginated. That is, a return character must be typed to obtain the next page of the report. When viewing the output an escape character may be entered, in place of a return character, to terminate the displaying of a report, and return to the main screen.

Printer

Reports output to a printer will contain headings on each printer page, as opposed to screen output which has only the initial heading.

File

Reports sent to a file will be of the same format as the reports sent to the screen. This allows other Unix commands to operate on the file without having to avoid headings throughout a report.
11. GENERATE REPORT CYCLE

When a report has been generated by the Generate Report function, the cycle of report tailoring and generation is repeated until the Exit function is entered.

When entering a query, the data collected for the previous query is still available. So, if the only changes to the query are one or more of the following:

1) different sorting fields,
2) different limitations in the Content Menu,
3) adding or subtracting print fields whose data was already collected by the previous query,
4) changing from printing to graphing a field, or vice versa,
5) changing the output destination,
6) changing the number of lines in the report,

the accounting files do not have to be re-read, but the report format of the data previously collected is changed, or the output destination is changed.

An advantage of this is that sorting fields can be changed and the new report will be generated almost immediately.

Another advantage, is that it allows a report to be first observed on the screen, before being sent to the printer or a file.
12. FORMAT of SUBJECT LIMITATIONS

Following is a list of all of the fields of the subject menu and the limitations that can be entered within these fields:

DATE :

- 'w' = this week,
- 'l' = last week,
- 't' = today's date,
- 'y' = yesterday's date,
- d = a given date,
- <d = before a date,
- d1-d2 = a date range,
- >d = after a date,
- '-' = all valid dates

where d is a date of any of the following forms:

1) DD/MM/YY
2) DD/MM (the year is the current year)
3) DD (the year and month are the current year and month)

TIME :

- t1-t2 = a time range,
- <t = before a time,
- >t = after a time,
- '-' = 00:00 - 23:59

where t is a time in 24-hour notation of the form HH:MM or HH.

COMMAND (USER-ID, GROUP-ID, TERM-ID) :

A list of one or more commands (user ids, group ids, terminal ids) separated by commas(,`) or blanks(` `).

A blank field implies all available commands (users, groups, terminals).

RECORDS :

No limitation can be entered in this field.

Note: If a date range of more than one day is entered along with a time range, then the report will be restricted to the entered time range for each of the selected dates.