Using Representation Clauses as an Operating System Interface

Karl A. Nyberg
Grebyn Corporation
P. O. 1144
Vienna, VA 22180

(703)-281-2194
karl@grebyn.com

Abstract

A set of Ada® packages and programs for performing system accounting under the Ultrix™ operating system on the VAX™ using VADS® are described. These programs provide for the billing of timesharing customers based on amount of connect time and quantity of CPU usage. Interactions at the operating system level are accomplished through the use of representation specifications. A description of the changes required when the operating system data structures were modified by the operating system manufacturer during an upgrade are also presented.

1. Timesharing Accounting

In the operation of a timesharing system, it is necessary to be able to allocate charges to users according to their utilization of the system resources. The most common system resources that are charged for include connect time, central processing unit (CPU) utilization, and possibly disk, character input / output, and memory utilization. The amounts and kinds of information maintained and available differ with various operating systems and environments. This paper describes the development of software in Ada for producing billing invoices under the Ultrix operating system, with a cost schedule where charges accrue for connect time and CPU time only.

2. Ultrix Accounting

The Ultrix operating system (as with all UNIX® derived operating systems) maintains two types of accounting data on users - connect time and resource utilization time. These two set of data are maintained separately by the operating system. The description of these data types and maintenance of the information is provided in the following sections.

2.1. Connect Time

The Ultrix operating system maintains a record of all login and logout events for all users of the system. This record is maintained in a file (/usr/adm/wtmp), that also contains data on system crashes, reboots, and date changes. Each record consists of four fields - the line number, the user name, the host name, and the time of the event. Events other than logins / logouts, are differentiated by special user names, such as "reboot". Time in the Ultrix operating system is represented by an integer as the number of seconds since a set time in the past.

There is only one utility program provided with the operating system that displays the amount of connect time from the collected data. This program, called ac, is capable of displaying the data on a per-day, per-user, or summary basis. There is no facility provided for displaying connect time according to any distinction of "prime" or "non-prime" time categories, nor for even specifying such categories. The C specification of the record structure of the file is provided with the operating system (and in the Appendix), so that additional accounting software could have been written in C if it had been so desired.

The records in the accounting file are stored using the internal representation of the C compiler for the record's layout. Once a matching Ada representation had been developed, it was simply a matter of instantiating sequential_io on the newly developed data type to allow the records to be read in directly in Ada without any translation or modification upon input.

Ada is a registered trademark of the U. S. Government (AJPO).

Ultrix, VAX are trademarks of Digital Equipment Corporation.

VADS is a registered trademark of Verdix Corporation.

UNIX is a registered trademark of AT&T Information Systems.

2.2. Resource Utilization Time

The Ultrix operating system maintains per-process accounting data for all activity on the system. This data is maintained in the file (/usr/adm/acct), not only for actual customers of the system, but also for all system software, demons, and other "overhead" programs. The fields of major interest include the command string, amount of user and system time, beginning time of the command, and the user identifier on whose behalf the process was initiated.

There is also a single utility program provided for displaying the data collected on process accounting. This program, called sa, provides information on the number of times each command was invoked, the total amount of CPU time used for all invocations, and other statistics. It does not provide any summary of accounting with respect to the individual users. However, as with the connect time, the C specification of the record structure is provided, so that development of additional accounting software in C that analyzes this data is also possible.

As with the connect time data type, once the matching representation was developed, input from the file could be accomplished with the facilities available in sequential_io, instantiated upon the type.

3. Timesharing Billing

The cost schedule for resource utilization on the computer system for which this software was developed is based on a monthly minimum, and additional charges for additional usage. The first ten hours of connect time and one hour of CPU time are included in the monthly charge of $25.00. After those amounts are exceeded, charges accrue at $2.00 / hour for connect time and at $50.00 / hour for CPU time. Thus, the charge for seven hours connect time and two hours CPU time would be $75.00, with an hour of connect time unused for the month, which would be forfeited. Additional charges are levied for use of special peripherals (*e.g.*, laser printers, outgoing modems) and disk quota allocations.

3.1. Implementation

The connect time data and resource utilization time data are collected nightly and merged into a file of the relevant information for billing purposes. This file is again an instantiation of sequential_io on a record type (billing) that has fields for user identification number, connect time, and CPU time. The records in this file are updated by programs that read each of the data files nightly, and update the relevant fields. A final program is provided that reads the billing data, and generates a billing report. Customers of the system are able to display their own billing data, to keep track of expenses, while the system administrator is able to obtain all data, to generate monthly invoices.

4. Results

4.1. Development

The development time was approximately one man month, consisting of 700 lines of Ada. Development was completed using the MICROVAX II, under Ultrix 1.1 and 1.2, using versions 5.1b and 5.2a of VADS.

4.2. Functionality

The accounting system has been in use for approximately four months now. The system does not yet automatically generate billing invoices Also, given the software does not totally reflect the complete nature of the cost schedule in effect, as costs for partial months, based on prorated usage and for disk space quota allocations are not included.

4.3. Performance

One disappointing result in the development of this software was the performance of the system. After a small amount of analysis, it was found that one major bottleneck occurred in the use of sequential_io under Ada. For the connect time data, the reading of records was approximately twenty times slower in Ada than in C, while for the process accounting data, the reading of records was approximately seventeen times slower. Attempting to use direct_io produced even somewhat slightly more slower times, and use of the current optimizer produced no noticeable change.

4.4. Maintainability

During the development of this software, the underlying data structures used by the operating system to represent accounting information were modified during an operating system upgrade. In particular, the fields for user time, system time, elapsed time, and average memory usage were modified from being 16 bit integers to being 32 bit floating point values. This was done to increase the precision of the accounting data. As a result it was necessary to modify not only

the data types of the fields of the record representing the accounting data, but also to modify the representation of the data to conform to the new layout.

Due to the strong typing in Ada, it was also necessary to make minor adjustments to constants (namely the value of zero) from an integer constant to a floating constant as a result of the modifications of the operating system structures. These modifications were performed with only a minor impact (two hours) on the development schedule of the software.

4.5. Portability

This software was designed for use in operating systems derived from the UNIX operating system. The major obstacle to the portability of this software to such operating systems lies in the type layouts embedded in the record specifications. This obstacle would not be so difficult in situations where representation specifications are supported in Ada implementations on those architectures. Without such features to specify the layout of the data for Ada, porting would be nigh on impossible unless the layouts ''just happenned'' to match those for the default Ada layouts.

5. Appendix

This Appendix provides the specifications for the various data structures that required representation specifications in order to map to the internal representation as provided by the C compiler under the Ultrix operating system. Both data structures for resource utilization as required under the separate versions of the operating system are provided.

5.1. Connect Time Accounting Data Structures

```
type utmp is
  record
    ut_line :    string (1 .. 8);          -- Terminal line
    ut_name :    string (1 .. 8);          -- User name
    ut_host :    string (1 .. 16);         -- Computer name
    ut_time :    integer;                  -- Time of event
  end record;

for utmp use
  record at mod 4;
    ut_line      at 0     range 0 .. 63;
    ut_name      at 8     range 0 .. 63;
    ut_host      at 16    range 0 .. 127;
    ut_time      at 32    range 0 .. 31;
  end record;
```

5.2. Resource Accounting Data Structures

5.2.1. C Version

```
struct   acct
{
        char    ac_comm[10];        /* Accounting command name */
        float   ac_utime;           /* Accounting user time */
        float   ac_stime;           /* Accounting system time */
        float   ac_etime;           /* Accounting elapsed time */
        time_t  ac_btime;           /* Beginning time */
        short   ac_uid;             /* Accounting user ID */
        short   ac_gid;             /* Accounting group ID */
        float   ac_mem;             /* average memory usage */
        float   ac_io;              /* number of disk IO blocks */
        dev_t   ac_tty;             /* control typewriter */
        char    ac_flag;            /* Accounting flag */
};
```

5.2.1.1. Ada Version

```
type acct is
  record
        ac_comm :       string (1 .. 10);   -- Accounting command name
        ac_utime :      comp_t;             -- Accounting user time
        ac_stime :      comp_t;             -- Accounting system time
        ac_etime :      comp_t;             -- Accounting elapsed time
        ac_btime :      time_t;             -- Beginning time
        ac_uid :        short_integer;      -- Accounting user ID
        ac_gid :        short_integer;      -- Accounting group ID
        ac_mem :        comp_t;             -- Average memory usage
        ac_io :         comp_t;             -- Number of disk IO blocks
        ac_tty :        dev_t;              -- Control typewriter
        ac_flag :       character;          -- Accounting flag
  end record;

  for acct use
    record at mod 4;
        ac_comm         at 0    range 0 .. 79;
        ac_utime        at 12   range 0 .. 31;
        ac_stime        at 16   range 0 .. 31;
        ac_etime        at 20   range 0 .. 31;
        ac_btime        at 24   range 0 .. 31;
        ac_uid          at 28   range 0 .. 15;
        ac_gid          at 30   range 0 .. 15;
        ac_mem          at 32   range 0 .. 31;
        ac_io           at 36   range 0 .. 31;
        ac_tty          at 40   range 0 .. 15;
        ac_flag         at 42   range 0 .. 7;
  end record;
```

5.3. Billing Record

```
type billing is
  record
    uid :       integer;        -- integer id for the user
    cpu :       comp_t;         -- cpu time, in seconds
    con :       integer;        -- internal time format
  end record;
```