ACCT20

DECSYSTEM-20 Resource-Usage Accounting Program

Version 2.0

Program Logic Manual

Written by
David Todd
Director of University Computing
Wesleyan University
Middletown, CT 06457

2 May 1988

Introduction

ACCT20 analyzes the TOPS-20 `SYSTEM-DATA.BIN` files that contain audit trails of DEC-20 usage and produces summary and detailed reports of usage statistics. The *ACCT20 User Manual* explains the use of the program for usage accounting. This *ACCT20 Program Logic Manual* outlines the internal operation of ACCT20 and explains how to assemble and load the necessary program modules.

Before attempting to understand the operation of program, you will need to have read the Digital *DEC-20 Accounting System Specifications* manual, Order Number DEC-20-UASSA-A-D (part of the TOPS-20 documentation set). This document explains the internal format of the `SYSTEM-DATA.BIN` file that TOPS generates to record system usage. The binary, variable-length records in that file contain information about CPU usage, structure and tape mounts, printer usage, etc. It is that file that ACCT20 processes to summarize system usage.

It will also be helpful to have access to the files that generate and are used by ACCT20. The distribution tape for ACCT20 contains the following files:

`ACCT20.MAC, .REL, .EXE`
: The assembly-language source program, assembled code, and executable code for ACCT20.

`BINTAB.MAC, .REL`
: A package of routines for handling binary tables.

`TREES.MAC, .REL`
: A package of routines used for binary-tree manipulation by ACCT20.

`CLSACT.MAC, .REL, .EXE`
: A program that can be used to close the TOPS-maintained file `SYSTEM-DATA.BIN` daily, giving a unique name to the file that summarizes the accounting data for that day.

`MONTHLY.CTL`
: A prototype for a batch control file that can be used to perform monthly accounting with ACCT20.

`ACCT20.CHG`
: An example of a file that can be used to specify charges to be assessed for resources used.

`ACCT20.PCG`
: An example of a file that can be used to specify charges to be assessed for different types of printer forms used.

`ACCT20.CAT`
: An example file to demonstrate how specific users and departments can be grouped together into categories for summarization of use by broad categories instead of use by individual users.

`ACCT20-USER.MSS, .PS, .LPT`
: The SCRIBE source file, PostScript-printable, and line-printable text files for the user manual.

`ACCT20-PLM.MSS, .PS, .LPT`
: The SCRIBE source file, PostScript-printable, and line-printable text files for this manual.

The latter, data files are used by ACCT20 in processing the accounting information into usage summaries, and much of the program deals with building tables from these files.

Internal Operation

The ACCT20 program is table-driven by various commands. The primary loop of the program is a very simple one beginning at `LOOP`, shortly after the `ACCT20` starting address, and is driven by the processing of a COMAND JSYS call (to DPROMP) that takes a command from the user and dispatches to the appropriate routine to process the command. Most commands set status bits in a flag accumulator to indicate post-processing output routine to be invoked, open and process auxiliary data files to build internal data tables, and open output JFN's.

The main processing routine is the one that processes the system data file (`SYSTEM-DATA.BIN`). This processing is initiated by the `PROCESS` command. The code in that section, starting at address `.PRO`:

- reads in a record and validates it as best it can,
- dispatches to the processing routine appropriate for the type of entry of the accounting record,
- returns to read the next record to process, and
- at the end of the last input file (from a possible sequence of input files), invokes procedures to dump and list summary data tables.

The summary information is maintained in a binary tree that is dynamically created during the `PROCESS`ing of the accounting files. The `DUMP` command dumps the (memory-independent) information in this tree, along with some header information, into the `SUMMARY.BIN` file. This file is can be used for `REPROCESS`ing, to re`CATEGORIZE` the summarized data for various usage analyses, for example. The `LIST` commands process the binary summary tree into text files.

Most of the processing is actually straightforward, and there are, hopefully, few interdependencies between sections of code. The addition of new commands should be relatively easy should new developments be desired.

Assembly and Loading

ACCT20 must be compiled with several standard Digital symbol-definition modules and loaded with two other modules written specifically for ACCT20, BINTAB and TREES. BINTAB is a set of routines to keep one-word-entry binary tables (similar to TBLUK JSYS methods but with binary keys). TREES builds, maintains, and processes the internal binary trees used for data storage for summary purposes (among other things). The LOAD command then takes the form

```
@LOAD ACCT20,BINTAB,TREES
@SAVE
```

The TREES routines use height-balancing algorithms explained by Knuth in Volume 1, *Fundamental Algorithms* in his series on *The Art of Computer Programming*. Version 1 of ACCT20 did not use height-balancing. Height balancing was expected to improve performance, but the net effect was minimal: sometimes an improvement, sometimes not, depending upon the data stream. Version 2 retains the height-balancing algorithms, though.

There are numerous demonstration and support files supplied with the ACCT20 package. Most of these are described in the ACCT20 User Manual. The `MONTHLY.CTL` and `BILLER.CTL` files serve as prototypes for a monthly summary report generating batch control file and for a monthly billing file. Instructions on use of these files are contained in the files themselves.

Deficiencies

The following are the major decificiencies of which I'm aware in ACCT20:

- Error recovery for various commands processing from disk files is poor. For example, if a keyword is mistyped in the `ACCT20.CHG` file, no recovery is attempted and the user must `CNTL-C` and search through the file to attempt to find and fix the problem.
- The binary tree routines do height balancing, but it appears that algorithm does not improve execution time of the program (and may make it worse in some cases). It appears that the efficiency of the summarizing process depends upon the order in which entries appear in the system data file. Use of a height-balancing algorithm for tree maintenance may help in the generation of the tree but not in its processing. Further analyses might be useful (but probably won't effect significant performance improvements).
- The summary listing file does not visually group summary lines by directory level. This should be a relatively easy fix.
- Generation of details listings for bills should be made automatic rather than requiring a separate processing step.
- Rob Austein has pointed out that while reading a data file to establish tables (e.g., *ACCT20.CAT*), the message "too big for internal buffer" can occasionally be generated. He points out that this is because even in processing a file the DPROMP JSYS should be called to set up state pointers. I haven't had a chance to try his suggested patches and so haven't installed them in Version 2, but I trust that these will solve that problem:

  Replace

  ```
  HRROI   A,[ASCIZ "Foo>"]
  TXNE    F,F%TTIN
   CALL DPROMP
  ```

  with code such as

  ```
  HRROI   A,[ASCIZ "Foo>"]
  TXNN    F,F%TTIN
   HRROI  A,[Z]
  CALL    DPROMP
  ```

  This will invoke DPROMP with a null string if not interactive, so the state block is initialized as it should be.

Unfortunately I will be unable to provide ongoing support or future versions of ACCT20. I invite anyone who has additional edits to submit the revised code to the DECUS Library, Marlboro, MA.

Table of Contents