GNU Accounting Utilities

Version 6.3.2
10 March 1998

Noel Cragg (noel@gnu.ai.mit.edu)

Preface

Way back a long time ago, Thompson and Ritchie were sitting opposite one another at the commissary, sipping coffees and discussing their evolving behemoth.

"This behemoth of ours," said Ken, "is becoming rather popular, wouldn't you say?" "Yes," said Dennis. "Every time I want to do a compilation, I have to wait for hours and hours. It's infuriating." They both agreed that the load on their system was too great. Both sighed, picked up their mugs, and went back to the workbench. Little did they know that an upper-management type was sitting just within earshot of their conversation.

"We are AT&T Bell Laboratories, aren't we?" the upper-management type thought to himself. "Well, what is our organization best known for?" The brill-cream in his hair glistened. "Screwing people out of lots of money, of course! If there were some way that we could keep tabs on users and charge them through the nose for their CPU time..."

The accounting utilities were born.

Seriously though, the accouting utilities can provide a system administrator with useful information about system usage—connections, programs executed, and utilization of system resources.

Information about users—their connect time, location, programs executed, and the like—is automatically recored in files by `init` and `login`. Four of them are of interest to us: `wtmp`, which has records for each login and logout; `acct`, which records each command that was run; `usracct` and `savacct`, which contain summaries of the information in `acct` by user and command, respectively. Each of the accounting utilities reports or summarizes information stored in these files.

`ac`
: prints statistics about users' connect time. `ac` can tell you how long a particular user or group of users were connected to your system, printing totals by day or for all of the entries in the `wtmp` file.

`accton`
: turns accounting on or off.

`last`
: lists the logins on the system, most recent first. With `last`, you can search the `wtmp` file for a particular user or terminal name (to which the user was connected). Of special interest are two fake users, '`reboot`' and '`shutdown`', which are recorded when the system is shut down or reboots.

`lastcomm`
: lists the commands executed on the system, most recent first, showing the run state of each command. With `last`, you can search the `acct` file for a particular user, terminal, or command.

`sa`
: summarizes the information in the `acct` file into the `savacct` and `usracct` file. It also generates reports about commands, giving the number of invocations, cpu time used, average core usage, etc.

`dump-acct`
`dump-utmp`
: display `acct` and `utmp` files in a human-readable format.

For more detailed information on any of these programs, check the chapter with the program title.

A Note on File Names and Locations

The `wtmp` and `acct` files seem to live in different places and have different names for every variant of u*x that exists. The name `wtmp` seems to be standard for the login accounting file, but the process accounting file might be `acct` or `pacct` on your system. To find the actual locations and names of these files on your system, specify the `--help` flag to any of the programs in this package and the information will dumped to standard output.

Regardless of the names and locations of files on your system, this manual will refer to the login accounting file as `wtmp` and the process accounting files as `acct`, `savacct`, and `usracct`.

History of the Accounting Utilities

I don't have any idea who originally wrote these utilities. If anybody does, please send some mail to `noel@gnu.ai.mit.edu` and I'll add your information here!

Since the first alpha versions of this software in late 1993, many people have contributed to the package. They are (in alphabetical order):

`Eric Backus <ericb@lsid.hp.com>`
: Suggested fixes for HP-UX 9.05 using /bin/cc: configure assumed you were using `gcc` and tacked on `-Wall` etc. He also noticed that `file_rd.c` was doing pointer arithmetic on a `void *` pointer (non-ANSI).

`Christoph Badura <bad@flatlin.ka.sub.org>`
: Christoph was a BIG HELP in computing statistics, most notably k*sec stuff! He also did Xenix testing and contributed some Makefile fixes and output optimizations.

`Michael Calwas <calwas@ttd.teradyne.com>`
: Fixed bugs in mktime.c.

`Alan Cox <iiitac@pyr.swan.ac.uk>`
: Original Linux kernel accounting patches.

`Scott Crosby <root@hypercube.res.cmu.edu>`
: Suggested idea behind `--sort-real-time` for `sa`.

`Solar Designer <solar@false.com>`
: Fixed off-by one error in allocations in `sa`.

`Dirk Eddelbuettel <edd@miles.econ.queensu.ca>`
: Managed bug-fixes & etc. for Debian distribution, as well as the architect of merge of GNU + Debian distributions. A big thanks to Dirk for kicking me back into gear again after a long period of no work on this project.

`Jason Grant <jamalcol@pc-5530.bc.rogers.wave.ca>`
: Identified a buffer-overrun bug in `sa`.

`Kaveh R. Ghazi <ghazi@caip.rutgers.edu>`
: Tested the package on many systems with compilers other than gcc. Fixed K&R C support.

`Susan Kleinmann <sgk@sgk.tiac.net>`
: Contributed excellent man pages!

`Marek Michalkiewicz <marekm@i17linuxb.ists.pwr.wroc.pl>`
: Suggested the `--ip-address` flag for `last`.

`David S. Miller <davem@caip.rutgers.edu>`
: Noticed missing GNU-standard makefile rules.

`Walter Mueller <walt@pi4.informatik.uni-mannheim.de>`
: Noticed install target was missing, and corrected a typo for prefix in Makefile.in.

`Ian Murdock <imurdock@gnu.ai.mit.edu>`
: Tracked down miscellaneous bugs in sa.c under Linux. Added Debian package maintenance files.

`Luc I. Suryo <root@patriots.nl.mugnet.org>`
: Suggested the `--user` flag for `lastcomm`.

`Pedro A M Vazquez <vazquez@iqm.unicamp.br>`
: Fixed bugs in sa.c and tested under FreeBSD.

`Marco van Wieringen <Marco.van.Wieringen@mcs.nl.mugnet.org>`
: Modified (wrote?) Linux kernel accounting patches.

1 ac

The `ac` command prints out a report of connect time (in hours) based on the logins/logouts in the current `wtmp` file. A total is also printed out.

The accounting file `wtmp` is maintained by `init` and `login`. Neither of these programs creates the file; if the file is not there, no accounting is done. To begin accounting, create the file with a length of zero. **NOTE:** the `wtmp` file can get really big, really fast. You might want to trim it every once and a while.

GNU `ac` works nearly the same u*x `ac`, though it's a little smarter in its printing out of daily totals—it actually prints *every* day, rather than skipping to the date of the next entry in the `wtmp` file.

1.1 Flags

All of the original `ac`'s options have been implemented, and a few have been added. Normally, when `ac` is invoked, the output looks like this:

```
total 93867.14
```

where total is the number of hours of connect time for every entry in the `wtmp` file. The rest of the flags modify the output in one way or another.

`-d`
`--daily-totals`
: Print totals for each day rather than just one big total at the end. The output looks like this:

```
Jul  3  total     1.17
Jul  4  total     2.10
Jul  5  total     8.23
Jul  6  total     2.10
Jul  7  total     0.30
```

`-p`
`--individual-totals`
: Print time totals for each user in addition to the usual everything-lumped-into-one value. It looks like:

```
bob        8.06
goff       0.60
maley      7.37
root       0.12
total     16.15
```

people
: Print out the sum total of the connect time used by all of the users included in people. Note that people is a space separated list of valid user names; wildcards are not allowed.

`-f` *filename*
`--file` *filename*
: Read from the file *filename* instead of the system's `wtmp` file.

`--complain`
: When the `wtmp` file has a problem (a time-warp, missing record, or whatever), print out an appropriate error.

`--reboots`
: Reboot records are *not* written at the time of a reboot, but when the system restarts; therefore, it is impossible to know *exactly* when the reboot occurred. Users may have been logged into the system at the time of the reboot, and many `ac`'s automatically count the time between the login and the reboot record against the user (even though all of that time *shouldn't* be, perhaps, if the system is down for a long time, for instance). If you want to count this time, include the flag. **To make `ac` behave like the one that was distributed with your OS, include this flag.**

`--supplants`
: Sometimes a logout record is not written for a specific terminal, so the time that the last user accrued cannot be calculated. If you want to include the time from the user's login to the next login on the terminal (though probably incorrect), include this flag. **To make `ac` behave like the one that was distributed with your OS, include this flag.**

`--timewarps`
: Sometimes, entries in a `wtmp` file will suddenly jump back into the past without a clock change record occurring. It is impossible to know how long a user was logged in when this occurs. If you want to count the time between the login and the time warp against the user, include this flag. **To make `ac` behave like the one that was distributed with your OS, include this flag.**

`--compatibility`
: This is shorthand for typing out the three above options.

`-a`
`--all-days`
: If we're printing daily totals, print a record for every day instead of skipping intervening days where there is no login activity. Without this flag, time accrued during those intervening days gets listed under the next day where there is login activity.

`-y`
`--print-year`
: Print out the year when displaying dates.

`--print-zeros`
: If a total for any category (save the grand total) is zero, print it. The default is to suppress printing.

`--debug`
: Print verbose internal information.

`--tw-leniency` *value*
: Set the time warp leniency value (in seconds). Records in `wtmp` files might be slightly out of order (most notably when two logins occur within a one-second

period – the second one gets written first). By default, this value is set to 1 second. Some `wtmp`'s are really screwed up (Suns) and require a larger value here. If the program notices this problem, time is not assigned to users unless the `--timewarps` flag is used. See the Problems section for more information.

`--tw-suspicious` *value*
: Set the time warp suspicious value (in seconds). If two records in the `wtmp` file are farther than this number of seconds apart, there is a problem with the wtmp file (or your machine hasn't been used in a year). If the program notices this problem, time is not assigned to users unless the `--timewarps` flag is used.

`-V`
`--version`
: Print `ac`'s version number.

`-h`
`--help`
: Print `ac`'s usage string and default locations of system files to standard output.

1.2 Problems

For no fault of `ac`'s, if two logins occur at the same time (within a second of each other), each `login` process will try to write an entry to the `wtmp` file. With file system overhead, it is forseeable that the entries would get written in the wrong order. GNU `ac` automatically compensates for this, but some other `ac`s may not... beware.

The FTP Problem

I've tested the standard `ac` in Ultrix 4.2 (DECstation/DECsystem), SunOS 4.1.1 (Sun3, Sun4, Sparc), Mach 2.5 (Omron/Luna), and DomainOS 10.3 (DN3500). All of these `ac`s have trouble parsing entries in which the line is `ftp`*xxxx* (*xxxx* being some number). Whenever these `ac`s see one of these entries, they log everyone out at the time of the entry.

HOW IT HAPPENS: if there is a user logged into the machine when an ftp connection occurs, (minimally) you'll get a login record for the user, a login record for the ftp connection, and the logouts for both afterwards (in either order).

TANGIBLE RESULT: the user who was logged in gets 'logged out' at the time the ftp connection begins, and none of the time spent during or after the ftp connection. Therefore, when you run GNU `ac`, the totals will most likely be greater than those of your system's `ac` (provided you specify the other flags that will make GNU `ac` behave like the system's).

The Shutdown/Reboot Problem

On Suns, `init` is a little screwed up. For some reason, after a shutdown record is written, a reboot record is written with a time-stamp *before* the shutdown (less than 30 seconds, usually).

TANGIBLE RESULT: GNU `ac` will notice the problem, log everyone out (you can specify if you want the time to be added to the user's total) and begin a new day entry based on the time of the out-of-sync record. If you try to print out daily totals, you'll notice that some days might have two or more entries.

SOLUTION: To fix this, a timewarp leniency value has been implemented. If any record is out of order by this number of seconds (defaults to 60) it gets ignored. If you need to change this value (if you think the totals are off because the value is too high), you can change it using the '`--timewarp-value`' flag. The rationale for the 60 second default is that of all of the machines with this problem, the largest timewarp was 45.

Stupid System V Machines

Some `ac`'s on System V machines (I've tried SGI Indigo & SGI Indy) forget to pay attention to the `ut_type` field in a `struct utmp`. As such, they chalk up a lot of time to non-existant processes called `LOGIN` or `runlevel`.

TANGIBLE RESULT: The amount of total time reported by the system's `ac` is **really** off. Often, it's several times greater than what it should be.

SOLUTION: GNU `ac` always pays attention to the `ut_type` record, so there's no possibility of chalking up time to anything but user processes.

2 accton

`accton` turns process accounting on or off. To save process accounting information in *accountingfile*, use:

`accton` *accountingfile*

If called with no arguments, it will, by default, stop process accounting.

2.1 Flags

`-V`
`--version`
: Print `accton`'s version number.

`-h`
`--help`
: Print `accton`'s usage string and default locations of system files to standard output.

3 last

`last` looks through the `wtmp` file (which records all logins/logouts) and prints information about connect times of users. Records are printed from most recent to least recent. Records can be specified by tty and username. tty names can be abbreviated: ‘`last 0`’ is equivalent to ‘`last tty0`’.

Multiple arguments can be specified: ‘`last root console`’ will print all of the entries for the user `root` and all entries logged in on the `console` tty.

The special users `reboot` and `shutdown` log in when the system reboots or (surprise) shuts down. ‘`last reboot`’ will produce a record of reboot times.

If `last` is interrupted by a quit signal, it prints out how far its search in the `wtmp` file had reached and then quits:

```
weerapan  ttyq6    132.162.32.37    Mon Feb 15 19:07 - 19:21  (00:13)
weerapan  ttyq6    132.162.32.37    Mon Feb 15 19:07 - 19:21  (00:13)

interrupted at Mon Feb 15 19:07:52 1993
```

3.1 Flags

This program implements the features of regular u*x `last` with a few extra flags. When `last` is invoked with no arguments, the output looks like this:

```
gr151     ttyp2    ray.cs.oberlin.e Tue Feb 16 17:40   still logged in
jhoggard  ttyp2    csts.cs.oberlin. Tue Feb 16 17:39 - 17:39  (00:00)
jstarr    ttyp1    UNIX5.ANDREW.CMU Tue Feb 16 17:38   still logged in
jberman   ttypb    132.162.32.25    Tue Feb 16 17:34   still logged in
alee      ttyp7    csts.cs.oberlin. Tue Feb 16 17:34   still logged in
jbrick    ttyp2    ocvaxa.cc.oberli Tue Feb 16 17:33 - 17:36  (00:03)
mbastedo  ttypc    ocvaxa.cc.oberli Tue Feb 16 17:25 - 17:26  (00:01)
rgoodste  ttypb    ocvaxa.cc.oberli Tue Feb 16 17:22 - 17:26  (00:03)
huttar    ttyp9    lobby.ti.com     Tue Feb 16 17:19   still logged in
klutz     ttyp3    132.162.32.25    Tue Feb 16 17:14   still logged in
```

`--no-truncate-ftp-entries`
: When printing out the information, don’t chop the number part off of `ftp`*xxxx* entries.

`-`*number*
`-n` *number*
`--lines` *number*
: Limit the number of lines that `last` prints.

`-f` *filename*
`--file` *filename*
: Read from the file *filename* instead of the system’s `wtmp` file.

`-y`
`--print-year`
: Print out the year when displaying dates.

`-s`
`--print-seconds`
Print out seconds when displaying dates and durations.

`--complain`
When the `wtmp` file has a problem (a time-warp, missing record, or whatever), print out an appropriate error.

`-x`
`--more-records`
Print out run level changes, shutdowns, and time changes in addition to the normal records.

`-a`
`--all-records`
Print out all records in the `wtmp` file.

`-i`
`--ip-address`
Some machines store the IP address of a connection in a utmp record. Enabling this option makes `last` print the IP address instead of the hostname.

`--tw-leniency` *value*
Set the time warp leniency value (in seconds). See the `ac` chapter for information.

`--tw-suspicious` *value*
Set the time warp suspicious value (in seconds). See the `ac` chapter for information.

`--debug` Print verbose internal information.

`-V`
`--version`
Print `last`'s version number.

`-h`
`--help` Print `last`'s usage string and default locations of system files to standard output.

3.2 Problems

The Clock Change Problem

Of the `last`s I've tried, all of them have had problems parsing a system clock change. Instead of modifying the entries that have been read, they just ignore the change and give you incorrect values. GNU `last` knows about clock changes and prints the correct times.

TANGIBLE RESULT: if you `diff` the output of your `last` and GNU `last`, entries after (before, rather) a clock change will be off by the amount of the clock change.

The Ftp Problem

Most `last`s that I've examined have the same problem here as `ac` does—they log everyone out as soon as they see an ftp entry.

TANGIBLE RESULT: GNU `last` will reflect the correct time spent in an ftp session, so the totals that it gives will most likely be greater than those given by the system `last`.

4 lastcomm

`lastcomm` prints out information about previously executed commands. If no arguments are specified, `lastcomm` will print info about all of the commands in the `acct` file (the record file). If called with a command name, user name, or tty name, only records containing those items will be displayed. For example, to find out which users used command '`a.out`' and which users were logged into '`tty0`', type:

```
lastcomm a.out tty0
```

This will print any entry for which '`a.out`' or '`tty0`' matches in any of the record's fields (command, name, or tty). If you want to find only items that match ALL of the arguments on the command line, you must use the '–strict-match' option. For example, to list all of the executions of command '`a.out`' by user '`root`' on terminal '`tty0`', type:

```
lastcomm --strict-match a.out root tty0
```

The order of the arguments is not important.

For each entry the following information is printed:

- command name of the process
- flags, as recorded by the system accounting routines:
  - **S** command executed by super-user
  - **F** command executed after a fork but without a following exec
  - **C** command run in PDP-11 compatibility mode (VAX only)
  - **D** command terminated with the generation of a core file
  - **X** command was terminated with the signal SIGTERM
- the name of the user who ran the process
- time the process exited

4.1 Flags

This program implements the features of regular u*x `lastcomm` with a few extra flags. When `lastcomm` is invoked without arguments, the output looks like this:

```
nslookup          jberman  ttypb      0.03 secs Tue Feb 16 19:23
comsat            root     __         0.03 secs Tue Feb 16 19:19
uptime            ctilburg __         0.11 secs Tue Feb 16 19:23
sh          F     ctilburg __         0.02 secs Tue Feb 16 19:23
sleep             ctilburg __         0.02 secs Tue Feb 16 19:22
ls                noel     ttyp4      0.19 secs Tue Feb 16 19:23
```

`--strict-match`
: Print only entries that match *all* of the arguments on the command line.

`--user` *name*
: List records for user with *name*. This is useful if you're trying to match a username that happens to be the same as a command (e.g., `ed`).

`--command` *name*
: List records for command *name*.

`--tty` *name*
: List records for tty *name*.

`-f` *filename*
`--file` *filename*
: Read from the file *filename* instead of the system's `acct` file.

`--debug`
: Print verbose internal information.

`--version`
: Print `lastcomm`'s version number.

`--help`
: Print `lastcomm`'s usage string and default locations of system files to standard output.

5 sa

`sa` summarizes information about previously executed commands as recorded in the `acct` file. In addition, it condenses this data into the `savacct` summary file, which contains the number of times the command was called and the system resources used. The information can also be summarized on a per-user basis; `sa` will save this information into `usracct`. Usage:

`sa [`*opts*`] [`*file*`]`

If no arguments are specified, `sa` will print information about all of the commands in the `acct` file. If command names have unprintable characters, or are only called once, `sa` will sort them into a group called `***other`.

If called with a file name as the last argument, `sa` will use that file instead of `acct`.

By default, `sa` will sort the output by sum of user and system time.

The output fields are labeled as follows:

`cpu` sum of system and user time in cpu seconds

`re` “real time” in cpu seconds

`k` cpu-time averaged core usage, in 1k units

`avio` average number of I/O operations per execution

`tio` total number of I/O operations

`k*sec` cpu storage integral (kilo-core seconds)

`u` user cpu time in cpu seconds

`s` system time in cpu seconds

An asterisk will appear after the name of commands that forked but didn't call `exec`.

5.1 Flags

`-a`
`--list-all-names`
: Force `sa` not to sort those command names with unprintable characters and those used only once into the ‘`***other`’ group.

`-b`
`--sort-sys-user-div-calls`
: Sort the output by the sum of user and system time divided by the number of calls.

`-c`
`--percentages`
: Print percentages of total time for the command's user, system, and real time values.

`-d`
`--sort-avio`
: Sort the output by the average number of disk I/O operations.

`-D`
`--sort-tio`
: Print and sort the output by the total number of disk I/O operations.

`-f`
`--not-interactive`
: When using the `--threshold` option, assume that all answers to interactive queries will be affirmative.

`-i`
`--dont-read-summary-file`
: Don't read the information in `savacct`.

`-j`
`--print-seconds`
: Instead of printing total minutes for each category, print seconds per call.

`-k`
`--sort-cpu-avmem`
: Sort the output by cpu time average memory usage.

`-K`
`--sort-ksec`
: Print and sort the output by the cpu-storage integral.

`-l`
`--separate-times`
: Print separate columns for system and user time; usually the two are added together and listed as `cpu`.

`-m`
`--user-summary`
: Print the number of processes and number of CPU minutes on a per-user basis.

`-n`
`--sort-num-calls`
: Sort the output by the number of calls. This is the default sorting method.

`-r`
`--reverse-sort`
: Sort output items in reverse order.

`-s`
`--merge`
: Merge the summarized accounting data into the summary files `savacct` and `usracct`.

`-t`
`--print-ratio`
: For each entry, print the ratio of real time to the sum of system and user times. If the sum of system and user times is too small to report—the sum is zero—`*ignore*` will appear in this field.

`-u`
`--print-users`
: For each command in the accounting file, print the userid and command name. After printing all entries, quit. **Note**: this flag supersedes all others.

`-v` *num*
`--threshold` *num*
: Print commands which were executed *num* times or fewer and await a reply from the terminal. If the response begins with `y`, add the command to the `**junk**` group.

`--separate-forks`
: It really doesn't make any sense to me that the stock version of `sa` separates statistics for a particular executable depending on whether or not that command forked. Therefore, GNU `sa` lumps this information together unless this option is specified.

`--sort-real-time`
: Sort the output by the "real time" (elapsed time) for each command.

`--debug`
: Print verbose internal information.

`-V`
`--version`
: Print `sa`'s version number.

`-h`
`--help`
: Print `sa`'s usage string and default locations of system files to standard output.

Note: if more than one sorting option is specified, the list will be sorted by the one specified last on the command line.

5.2 Problems

I haven't been able to test this on many different machines because the data files grow so big in a short time; our sysadmin would rather save the disk space.

Most versions of `sa` that I've tested don't pay attention to flags like `--print-seconds` and `--sort-num-calls` when printing out commands when combined with the `--user-summary` or `--print-users` flags. GNU `sa` pays attention to these flags if they are applicable.

5.2.1 mips sa

The average memory use is stored as a short rather than a double, so we suffer from round-off errors. GNU `sa` uses double the whole way through.

Table of Contents