Process Accounting

Standard utilities can help you collect and interpret your system's process accounting data.

by Keith Gilbertson

While on site at a Fortune 500 corporation recently, I overheard a tech support person whispering excitedly to a project manager, ``Don't play any games on your PC! The corporate auditors have a way to find out exactly what programs you use and for how long!"

After loudly assuring the techie that he was all business and didn't intend to play games anyway, the manager smiled. Then in a much quieter tone he said he needn't be concerned; he was using Linux and not Windows, unlike most of the company.

If the tech's tale is true, the manager may indeed have reason for concern. Although the rumoured auditing application at this particular company was developed for Windows, the Linux kernel has a built-in process accounting facility. It allows system administrators to collect detailed information in a log file each time a program is executed on a Linux system. With this capability, our mythical corporate auditor could, in fact, collect information about who has been playing games on a Linux computer and for how long.

Although a company's interest in knowing which employees have been indulging in *Solitaire* on company equipment is of questionable merit, there are good reasons to use process accounting (PA). In this article, I discuss some situations where process accounting is useful, explain where to obtain and how to use the standard process accounting commands, and then demonstrate how to use the process accounting structure and system call in C programs.

Preliminaries

I assume that your system has process accounting support compiled into the kernel. I make this assumption because the kernels on all of the Linux systems I have had access to are configured to allow process accounting, but your distribution may be different. If you compile and run the first code listing in this article as root with no command-line arguments but receive an error message, it is likely that process accounting support is not included in your kernel. You'll need to compile a new kernel and answer yes to CONFIG_BSD_PROCESS_ACCOUNTING, which is the BSD Process Accounting item in the General Setup menu. Recompiling your kernel is beyond the scope of this article, but instructions can be found at the Linux Documentation Project (www.tldp.org/HOWTO/Kernel-HOWTO.html).

On busy systems, keep in mind that turning on process accounting requires significant disk space. On my Pentium III system with Red Hat 7.2, each time a program is executed, 64 bytes of data are written to the process accounting log file. While researching this article and running the process accounting utilities on a test machine with low disk space, I discovered a monitor process that executes every second. The drive on that machine filled up quickly. Some server's dæmons will initiate a separate process for each incoming connection. On a production server that executes nearly 25,000 processes per hour, approximately 1.1GB of process accounting data is generated each month. Utilities, such as the accttrim and handleacct.sh script listed in Table 1, are available to truncate, back up and compress log files at regular intervals. If you plan on doing process accounting on a busy system, it will be important for you to learn about and use these utilities.

Finally, know that you must have root privileges on your Linux system to enable or disable process accounting, whether using the standard

commands or creating your own.

Uses of Process Accounting

One of the earliest uses of process accounting was to calculate the CPU time absorbed by users at computer installations and then bill users accordingly. With the greater abundance and relatively low expense of today's computing resources, this application has fallen by the wayside. If the distributed computing model catches on, however, this application could again become important.

System administrators may wish to use data collected from the PA facilities to monitor which programs are most accessed by users, and then optimize the system configuration for these types of programs. For example, part of the data collected by the PA facilities includes the number of bytes that are input and output by the program and the CPU usage. A system that runs a high percentage of I/O-intensive applications may need to be optimized in ways that a system running a high percentage of CPU-bound applications not.

At some point an administrator might be required to evaluate two products with similar functionality. Let's imagine that before making a selection, the administrator wishes to see which fish forecasting product the people are actually using. To do this, process accounting can be turned on for a week to record the names of all the commands executed in a log file. The administrator can then parse the log file to find out which command was run more often.

The most typical application of process accounting is as a supplement to system security measures. In the case of a break-in on a company server, the log files created by the process accounting facility are useful for collecting forensic evidence. A careful look at the programs an attacker has used on the compromised system can provide useful information

about the damage done, as well as the intruder's methods and possible motivations. Evidence collected from the process accounting logs also may be helpful in court. I know of one criminal case in which this data, when uncontested by the defendant, led to a misdemeanor conviction.

Standard Process Accounting Commands

Even if process accounting facilities have been compiled into your kernel, you might not have the user commands for process accounting installed on your system. If this is the case, and you're looking to get started quickly, first try finding the process accounting commands for your specific Linux distribution.

The package for your distribution likely is configured to place log files in the appropriate location for your system's setup, making installation much simpler. On my Red Hat 7.2 distribution CDs, I found the psacct-6.3.2-9.i386.rpm on the second disk, in the directory. If you use the gnorpm graphical install tool, the package will appear in the Packages/Applications/System hierarchy. On a Debian system, install the acct package.

If you're installing from source, two versions of the utilities are available. One version, under the BSD license, is available at www.ibiblio.org/pub/Linux/system/admin/accounts. The filename will be similar to acct-1.3.73.tar.gz, with small differences depending on the version number. In order to get these utilities to compile on my system, I had to edit the lastcomm.c file and comment out the prototype for the strcpy function.

There is also a process accounting utilities set written by Noel Cragg and licensed under the GNU GPL. It's available at www.gnu.org/directory/System_administration/Monitoring/acct.html.

The exact process accounting commands installed on your system will vary depending on the particular package you've chosen. Table 1 shows a list of the commands you could encounter and the purpose of each.

Table 1. Process Accounting Commands

Command Name	Purpose
accton	Enables or disables process accounting
acctentries	Counts the number of accounting entries in the log file
accttrim	Truncates the accounting file specified
dumpacct	Dumps the contents of the log file
dump-acct	Similar to dumpacct
handleacct.sh	Script to compress and backup logs and delete the oldest
lastcomm	Prints commands executed on the system, most recent first
sa	Summarize accounting information

Installation of the GNU Accounting Utilities

Let's take a quick look at how to install the GNU Accounting Utilities on a system. Use the following commands:

```
 tar zxvf acct_6.3.5.orig.tar.gz cd acct-6.3.5 ./configure
make su make install
```

A few basic process accounting commands have now been installed on your system. You're now ready to turn on the accounting and start using the commands.

Using the Utilities

In this brief introduction to using the process accounting commands, I look at two commands, accton and lastcomm. I've chosen these two commands because they are standard on all process accounting versions.

The accton command switches process accounting on or off. If a filename is specified on the command line, that filename will be used to log the process accounting information. If no argument is specified, process accounting will be switched off.

To start the process accounting facilities on your system, `su` to become root. Make sure that the log file exists by performing a **touch** on the desired location. Example:

```
touch /var/log/pacct
```

Then type the full path to your accton program (usually /usr/sbin/accton or /sbin/accton) followed by the filename. Example:

```
/sbin/accton /var/log/pacct
```

You've just started the process accounting facilities. Note that the data actually is not added to the file when each process begins execution; it is written when a process exits. The aforementioned project manager can play the *xbill* game all day long and not have this information written to the process accounting file, as long as he never exits the program. When he goes home at night, he can choose to leave *xbill* running and minimize the window, or he can simply power off his computer without performing a proper shutdown.

Now that you've switched on the accounting, run a few normal commands as an ordinary user to get some data for the lastcomm command, which you'll use next. When you're finished, `su` to root once more, and run `/usr/sbin/accton` or `/sbin/accton` with no arguments to switch off process accounting.

The lastcomm command prints information contained in the accounting log files, with the most recent record printed first. You can use the -f

command-line option to specify a filename. Typically, the process accounting log file on a system is set up so that only root can read it. This command is then executed by root, for example:

```
lastcomm -f /var/log/pacct
```

When you type in the above command, the output is similar to this:

```
id     root  stdin 0.00 secs Mon Jul 22 12:41 xauth  S
root  stdin 0.00 secs Mon Jul 22 12:41 xauth  S keithg
stdin 0.00 secs Mon Jul 22 12:41 xauth  S keithg stdin 0.01
secs Mon Jul 22 12:41 bubbles X keithg ??   0.01 secs Mon
Jul 22 12:33 ls     keithg ??   0.01 secs Mon Jul 22 12:26
bash   X keithg ??   0.03 secs Mon Jul 22 08:25
```

lastcomm displays the command name, options, user name, terminal and exit time for each command. A particular command, user or terminal also can be specified on the command line. For example, if you want to find instances only of when the su program was started, you can type:

```
lastcomm -f /var/log/pacct --command su
```

Now you'll see output like this:

```
su   root   ??   0.01 secs Mon Jul 22 10:52 su   keithg
stdout 0.05 secs Mon Jul 22 09:32 su   keithg  stdout 0.00
secs Mon Jul 22 09:17 su   root   ??   0.00 secs Mon Jul 22
03:29 su   keithg  tty1  0.00 secs Sun Jul 21 19:49
```

Notice that on each line, the command listed in the left column is now su. For more details about these commands and the other programs in the table, see the respective man pages.

Programming Details

The acct structure for collecting process accounting details is documented in the header files /usr/include/linux/acct.h and /usr/include/sys/acct.h. Table 2 displays the members available in the acct struct and a brief description of each member.

Table 2. Members in the acct struct

Member Name	Type	Description
ac_flag	char	Special flags showing process behaviors
ac_uid	u_int16_t	User ID of the process
ac_gid	u_int16_t	Group ID of the process
ac_tty	u_int16_t	Controlling terminal of the process
ac_btime	u_int_32_t	Process start time
ac_utime	comp_t	User time
ac_stime	comp_t	System time
ac_etime	comp_t	Elapsed time
ac_mem	comp_t	Average memory usage
ac_io	comp_t	Characters transferred
ac_rw	comp_t	Blocks read or written
ac_minflt	comp_t	Minor pagefaults
ac_majflt	comp_t	Major pagefaults
ac_swaps	comp_t	Number of swaps
ac_exitcode	u_int32_t	Process exitcode
ac_comm	char[]	First 16 characters of command name
ac_pad	char[]	Padding bytes

As you can see from the table, a lot of information is packed into the 64-byte accounting record. If you feel you need more information than is available with standard process accounting, consult the book by Mann and Mitchell listed in Resources at the end of this article.

Example Programs

Listing 1. Enabling and Disabling Accounting to a File

```
/* pa.c
 * Linux demonstration program.
 * Logs process accounting information to a
 * file specified on the command line.
 * If no filename is specified, process
 * accounting is switched off.
 */
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int
main (int argc, char **argv)
{

   int rc;

   if (argc == 1) /* No arguments - switch off */
   {
      printf("Turning off process accounting.\n");
      if ( (rc = acct (NULL)) )
      {
     if (errno == ENOSYS)
     {
       printf
         ("It appears your kernel does not"
          " include accounting support\n");
     }
        perror("Problem turning off accounting");
     return rc;
      }
   }


   else /* cmd line arg - switch accounting on */
```

```
   {
      printf
      ("Attempting to log to file %s.\n",
       argv[1]);
      rc =
      creat (argv[1],
          S_IRWXU | S_IRGRP | S_IROTH);

      if (rc == -1)
      {
      perror("Problem creating log file");
      return rc;
      }

      if ( (rc = acct (argv[1])) )
      {
     perror("Problem in acct() call");
     return rc;
      }

   }

   return 0;
}
```

Listing 1 is a simple demonstration of the use of the acct system call. The acct call takes one argument, the name of the file to which process accounting information is appended. If the argument is NULL, process accounting will be turned off. In addition, the file already must exist when the system call is made, or the call will fail and an error will be returned.

If a program running with the ID of an ordinary user makes a call to acct, the call also will fail and return an error. Programs that attempt to switch process accounting on and off must have root privileges to succeed.

The code in Listing 1 is similar to a typical implementation of the accton command, but there are two main differences. The first is that this code will report its actions in messages to standard output. The second is that

if the file specified on the command line does not exist, it will be created.

The file includes the <unistd.h> header file. All programs that make use of the acct call should include this file. The program checks to see if argc is equal to one, meaning no arguments were passed on the command line. If this is so, the program attempts to turn off process accounting by calling acct with a NULL argument.

If the command is run with an argument, the program will assume that the first argument is the filename. If the file does not exist, the program will attempt to create it with the creat system call. Then, the program will call acct with the filename as an argument to turn on process accounting. If an error code is returned from a system call, a message will be printed and the program will exit.

Listing 2. Parsing the Accounting File

```
/*
 * paread.c: Linux program to demonstrate reading a
 * process accounting record into memory.
 */

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/acct.h>

int
main (int argc, char *argv[])
{

   int fd;
```

```
   int bytesread;

   struct acct a; /* accounting record */

   if (argc == 1)
   {
      printf("You must supply a filename"
          " on the cmd line\n");
      return -1;
   }


   fd = open (argv[1], O_RDONLY);
   if (fd == -1)
   {
      perror ("Problem opening specified file");
      return -1;
   }

   /*
    * Read and print command name from each record
    * in the file
    */
   while ((bytesread =
          read (fd, &a, sizeof (struct acct))) > 0)
   {
      printf ("%s\n", a.ac_comm);
   }
   return 0;
}
```

Listing 2 demonstrates how to read records from the log file into an acct structure in memory so that the information can be printed out or operated upon. This program includes the <sys/acct.h> header file. All programs that need to work with the acct structure should include this file. Local variables in the main function include a file descriptor, a variable to hold the number of bytes read from the file and an acct struct.

The user of the program must specify a filename on the command line. The program attempts to open this file for read-only access. If the open was successful, the program will read() a record from the file directly

into the local acct structure, a. Due to space constraints for the article, I've made the assumption that a read() always will return exactly the number of bytes requested, until the end of file is reached. The program continues to read and print the command name from the records until a zero is returned from the read() call, signalling the end of file condition.

The Listings in this article are intended to be simple introductions to the system accounting structures. Robust programs would create a buffer to read multiple accounting records at once, and they would check for issues such as fewer bytes read from the file than were requested. To see examples of robust programs, look at the source for the process accounting utilities that you've installed.

Conclusion

You now have enough information to enable process accounting and use the standard commands to retrieve information about programs executed on a Linux system. If you're so inclined, you also can learn to make custom tools that parse the process accounting log files.

If you're using process accounting for system security, keep in mind that it is not by any means a comprehensive solution, but only one small tool. In fact, as Mann and Mitchell point out, you should be careful about trusting the information in the process accounting log files; the logs may have been modified by a technically savvy attacker.

With a basic understanding of the process accounting tools in Linux and some experimentation, you can set up these utilities on your own computer. If you're fortunate enough to have root access to the systems at work, you'll also be prepared to remove all traces of the *Sokoban* game from the accounting log files--in case that evil corporate auditor really does show up in your department one day.

Resources

GNU Accounting Utilities, Noel Cragg: www.gnu.org/directory/System_administration/Monitoring/acct.html

Linux Documentation Project Kernel-HOWTO, Brian Ward, Al Dev (Alavoor Vasudevan): www.tldp.org/HOWTO/Kernel-HOWTO.html

Linux System Security by Scott Mann and Ellen L. Mitchell, Prentice Hall PTR, 2002.

The Process Accounting Mini-HOWTO, Albert M. C. Tam: www.tldp.org/HOWTO/mini/Process-Accounting/index.html

Other Process Accounting Utilities: www.ibiblio.org/pub/Linux/system/admin/accounts

Keith Gilbertson *(keith.gilbertson@vt.edu) is currently working as a librarian at Virginia Tech.*