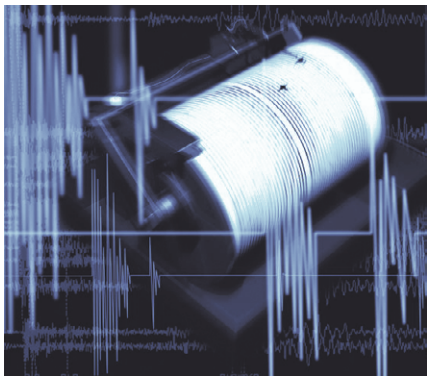

gPS

# Process Overview

Do you tend to lose track with text based process monitors like top? gPS provides a GUI based replacement and is capable of monitoring multiple computers across a network. BY CHRISTIAN PERLE

No, the *gPS* process monitor has nothing to do with the Global Positioning System. In this case the abbreviation stands for "graphical *P*rocess *S*tatus". If you do not feel comfortable with the monitoring tools provided for KDE and GNOME by the major desktop distributions, or simply prefer smaller footprint alternatives such as *XFCE* or *WindowMaker*, *gPS* might be just your choice of tool, to help you search for resource hogs or keep an eye on server processes.

## Building Blocks

In addition to the C/C++ compiler *gcc* or *g++* you will need the *gtk* **library** made famous by the *gimp* **raster graphics** program and the accompanying development package, to compile the gPS sourcecode archive available from *http://gps.seul.org/*.

To be precise gPS comprises two components: the monitor itself and a server, which monitors processes on remote machines. The following steps will unpack, compile and install the process monitor *gPS*:

```
tar xzf gps-1.1.0.tar.gz
cd gps-1.1.0
./configure
make
su  (enter your root password)
make install_gps
exit
```

## Lists and Trees

*gPS* has two ways of displaying processes: the main window displays them in a kind of list (see Figure 1), a display form similar to the output produced by the *ps* command in the **shell**. You can click on a column header (***PID***, *Name*, *Owner* etc.) to sort the output in ascending or descending order by the selected criterion. The four buttons at the bottom are used for refreshing the list (*Refresh*), sending **Signals** to processes (*Hang Up (SIGHUP)* and *Kill (SIGKILL)*), and for quitting the tool (*Close*).

The *View / Tree view…* menu item opens a tree view (see Figure 2). This display mode shows the child processes spawned by other processes. For example, the user *chris* on *camera* is working in the shell (process *#363*) on the console and has used the *startx* script to launch a GUI. This process, *#378*, is a child process of *363* and has a number of child processes itself.

Branches of the tree can be expanded or collapsed, just like the facility provided by most file managers. You can select a process to send signals to that process, or any processes in its branch of the tree, via the *Action* menu.

## More is more

You can select *Action / Details…* to tell *gps* to output more detailed information on the selected process (see Figure 3). The program gathers this information from the *proc* filesystem, which commands like *ps* and *top* also use as a source of information.

The *Action / Watch Process:half second polling* menu item in the main window provides more detail (the *Action* menu in the tree view does not provide access to all sub-items), drawing a histogram of the current memory and processor load for the selected process. You can use the *View / CPU and memory usage…* and *View / CPU and memory usage per user…* menu items to show statistics for the whole system, or individual users. The second of these functions uses a stack diagram to visualize the percentage of *Memory* and processing capacity (*CPU*) used by individual users and user groups.

You can use the *Filter / Set filter…* function to filter the list according to selectable criteria. The filter in Figure 4 tells *gPS* to display only those processes that are hogging more than 20 percent of the total CPU capacity. Unfortunately, there is no way to combine

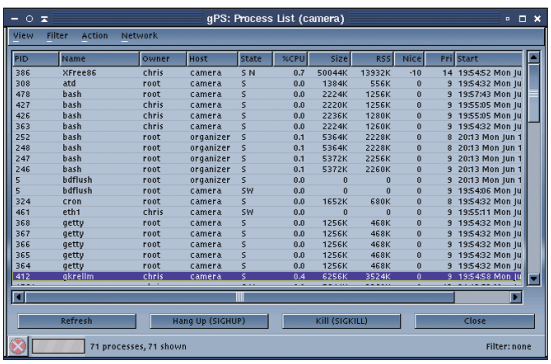


Figure 1: The gps process list looks a lot like output from the ps command, but with enhance legibility

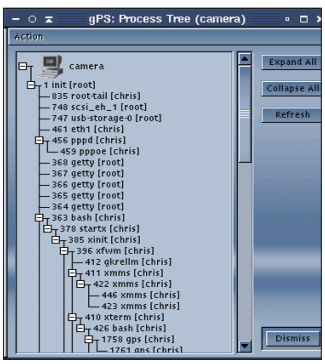


Figure 2: The process tree clarifies the relationships between processes

### Out of the Box

There are thousands of tools and utilities for Linux. "Out of the Box" takes a pick of the bunch and each month suggests a little program, which we feel is either absolutely indispensable or unduly ignored.

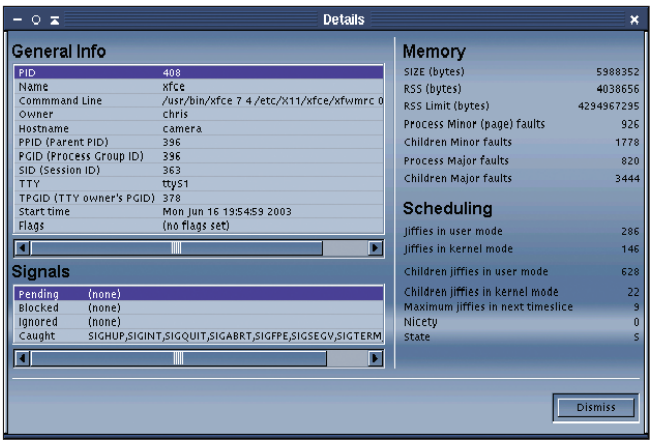


**Figure 3: A process under the magnifying glass**

filters. *Filter / Clear filter* will remove the current filter.

## Remote Viewing

In contrast to *top* or *ps*, gPS can display processes running on remote machines along with the local processes. To do so, the program uses its own server, *rgpsp*. To install the server, assume *root* privileges in the shell, then change to the source directory, *gps-1.1.0*, and enter *make install_net_all*. This installs the server program, and places the **init script**, */etc/rc.d/init.d/rgpsp*, and configuration file, */etc/rgpsp.conf*, on your machine.

They should be located in the correct place for Red Hat, Mandrake, Connectiva and SuSE installations.


**Figure 4: Filters allow you to focus the display on misbehaving processes**

Debian users will need to move the script to the right place by entering *mv /etc/rc.d/init.d/rgpsp /etc/init.d/rgpsp*. The next step is to discover your system's default **runlevel**, to allow the *rgpsp* to launch automatically when you boot your machine.

To do so, look for the line containing the *initdefault* in your */etc/inittab* file. Debian typically uses runlevel 2 as default; in this case, you would need to create a symbolic link in the */etc/rc2.d* directory. You will also want to halt the service gracefully when you shutdown your system, so place a symbolic link in the */etc/rc0.d* and */etc/rc6.d* directories to do so:

```
cd /etc/rc2.d
ln -s ../init.d/rgpsp S90rgpsp
```

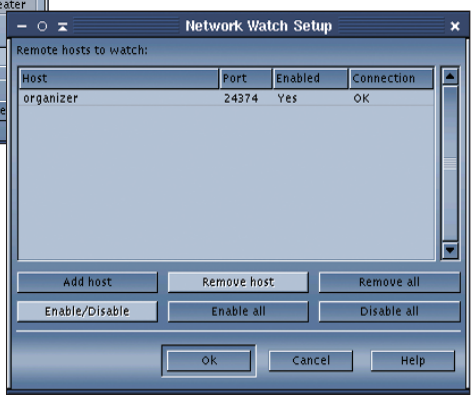


**Figure 5: Monitoring processes across the network**

## INFO

[1] Marc André Selig: "Ready – Steady – Go!", Linux Magazine Issue 27, February 2003, p50

```
cd /etc/rc0.d
ln -s ../init.d/rgpsp K10rgpsp
cd /etc/rc6.d
ln -s ../init.d/rgpsp K10rgpsp
```

The */etc/rgpsp.conf* configuration file allows you to control network access to the server. Use the file to specify the IP addresses of the computers that will be assigned access, typing one address per line. You can allow access to all hosts, by typing a single asterisk * in a line, but this setting is definitely not recommended on grounds of security.

To monitor a computer running *rgpsp* remotely, select the *Network / Network watch…* item in the *gPS* main menu. The window that then appears, *Network Watch Setup* (see Figure 5), allows you to create a list of computers to access. The results are displayed by *gps* as shown in Figure 1, where you can see the processes of the local host *camera* and the remote computer, *organizer*, in a single list.

You can request more detailed information for remote hosts, just as if you were monitoring the local host. But you are not permitted to send signals for security reasons.

If you are really serious about the security aspects, you will want to modify the init script to prevent the *rgpsp* server from running with *root* privileges. To do so, replace the *$RGPSP* line with *su nobody -c $RGPSP*.

## GLOSSARY

**Raster graphics:** *Also known as bitmap: generic term for graphic formats such as TIFF, JPG, PNG or GIF, where information is stored in the form of individual dots on screen (pixels). This contrasts with the various vector graphic formats that use mathematical formulae to describe geometrical forms, and thus scale without compromising quality.*

**Libraries:** *Files containing collections of useful C or C++ functions for specific purposes. For example, libm, which provides mathematical functions, or libXt, which contains functions for programming the X window system. Often multiple programs use a common or "shared" library.*

**Shell:** *One of the most vital components of any UNIX type system – the command-line based user interface. The shell can also execute text files that contain shell commands, so-called scripts.*

**PID:** *To distinguish between programs running on a Linux system, each process is assigned an identification number, or Process ID.*

**Signals:** *Signals tell processes to perform specific actions – this assumes the existence of a signal handler. Thus, many tools re-read their configuration when they receive a hangup signal, SIGHUP. The SIGINT, SIGTERM, and SIGKILL are used to terminate processes – the latter is used to kill the process immediately.*

**Init script:** *A shell script that launches or stops a service when a machine boots or shuts down (or, to be more precise, when the runlevel changes). For additional information on runlevels and system startup refer to [1] and the next definition.*

**Runlevel:** *A runlevel describes a specific system state where pre-defined services will be active. Runlevels 2 through 5 are typical working levels (for normal multi-user operations with or without X), S or 1 (single-user operations for troubleshooting), 0 (halt system) and 6 (restart system).*

**..:** *In the shell the two dots stand for the parent directory.*