Linux development on the PlayStation 3, Part 2: Working with memory

Identifying memory hogs on PS3 Linux

Skill Level: Intermediate

Peter Seebach (developerworks@seebs.plethora.net)
Freelance author
Plethora.net

31 Mar 2008

The Sony PlayStation 3 (PS3) runs Linux®, but getting it to run well requires some tweaking. In this article, the second in a series, Peter Seebach takes a look at where all the memory goes and how to reclaim it.

In Part 1 of this series, I introduced PlayStation 3 (PS3) Linux and its strengths and weaknesses as a development environment. This second part covers some of the things that can have a significant impact on a PS3 system's performance running Linux.

Not every piece of advice here will work for everyone. If you're doing graphics, you can't just run the system without X. On the other hand, if you're not doing graphics, nothing else makes as much of a difference in the system's memory footprint.

The focus on memory may seem odd if you're used to desktop systems that have never mounted a swap drive in anger, but, in fact, the PS3 doesn't have enough memory for a modern desktop Linux to feel comfortable on it out of the box. With a default Fedora 7 install, the system spends a lot of its time (and yours) swapping. Swapping imposes a large penalty on any system; on a system built around a 2.5-inch disk drive accessed through a hypervisor, with main memory that's substantially faster than what you find in most desktops, the contrast is even sharper than it is on a more traditional desktop system.

One other note: On my test system, rebooting was unreliable under the 2.6.21 kernel

that was originally installed. The 2.6.23 kernel fixed this, using either the version from the PS3 addons CD, or the version from the standard Fedora updater. In general, the kernel from the PS3 addons CD is probably your best bet.

About this series

This series of three articles looks at PS3 Linux as a prospective development environment.

This first article, Part 1, introduces the basic configuration knobs and widgets specific to the PS3, shows you how to use them effectively, and suggests the kind of trickery that might get improved performance or a more usable display.

This article and Part 3 delve into some of the performance and tuning issues that, while they might apply on any system, are particularly useful for turning your PS3 from a proof-of-concept demo into a working system.

Reducing memory usage

This topic comes and goes in importance. For most desktop Linux users, the idea that you'd need to actually do something to reduce memory usage is a distant memory. Furthermore, because processes tend to grow to fill available space, even when a machine with 64MB of main memory was considered a powerful server, there simply wasn't as much software running, and it didn't need as much memory. The PS3 is one of the systems where memory usage matters a lot, though, and Fedora 7, charming though it is, has not been designed around small-memory systems.

To reduce memory usage, start by identifying the largest consumers of memory. One easy way to do this is with `top`, which gives you a live display of processes on the system. By default, `top` shows you processes sorted by CPU consumption, which is useful for other kinds of performance tuning, but isn't the best way to track down memory hogs. Note that `top` gives you a summary of memory usage. For instance, on a PS3 with X running but several services turned off, I got this line:

Listing 1. Am I really using that much memory?

```
Mem:    219192k total,   213692k used,      5500k free,     7232k
buffers
Swap:  4192956k total,        0k used,   4192956k free,    89468k cached
```

Bring up `top` (just run `top` in a shell), then hit **O** (that's a capital O, as in "order by"), then hit **q**, then hit **Return**. In this case, **q** means "resident size" and tells you how much actual memory the process is using. Another likely choice would be "virtual

size" (option **o**).

You should now see a list of processes, sorted by actual physical memory usage. Here's a partial listing, again from a test machine:

Listing 2. I guess I am using that much memory

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3259 root      20   0 65424  36m 4996 S    2 17.2   0:01.39 Xorg
 3422 seebs     20   0 92900  24m  20m S    0 11.6   0:01.22 nautilus
 3439 seebs     20   0 58600  24m  22m S    0 11.5   0:00.36 nm-applet
 3473 seebs     20   0 56620  24m  14m S    0 11.4   0:01.22
/usr/bin/sealer
 3420 seebs     20   0 50248  21m  18m S    0 10.2   0:01.90 gnome-panel
 3476 seebs     20   0 48988  14m  10m S    1  6.9   0:00.64
gnome-terminal
 3445 seebs     20   0 33104  14m 9464 S    0  6.7   0:00.40 puplet
 3453 seebs     20   0 45764  13m  12m S    0  6.4   0:00.22
gnome-power-man
 3414 seebs     20   0 41920 9696 8052 S    0  4.4   0:00.29
gnome-settings-
 3418 seebs     20   0 22200 8996 7316 S    0  4.1   0:00.33 metacity
 3297 seebs     20   0 40544 8384 7088 S    0  3.8   0:00.32
gnome-session
 3432 seebs     20   0 20076 6120 5244 S    0  2.8   0:00.10
bluetooth-apple
 3444 seebs     20   0 14692 6060 3532 S    0  2.8   0:00.24 python
```

The top ten or so consumers of memory are all X-related. This is why, if you really want to free up memory, one of your first choices might be to shut down X. Noticing how many of those applications are GNOME-specific, you might be tempted to try KDE, but I'm afraid that won't get you anywhere. KDE has a comparable memory footprint on the PS3.

In fact, if you absolutely need X, your best option is not to use the X session environments offered by the X login window; instead, log in on the console and start X with a smaller window manager and fewer additional programs. But how to do that? Your first step will be to exit `top` and get your prompt back—just hit **q**.

Making use of runlevels

Runlevels are a feature that many Linux users never have cause to learn about. They're essentially inherited from System V UNIX®, although there are some differences, of course. A *runlevel* is a defined set of system services that are run together. For historical reasons, the usual desktop Linux environment, with a graphical login program that spawns Gnome or KDE, is called runlevel 5. A conventional standalone workstation without X would often run in runlevel 2. In theory, you can do pretty well by just changing the system's runlevel directly with the `init` command, found in /sbin. For instance, running (as root) `/sbin/init 2` tells the system to move to runlevel 2. (Usually this is done by stopping any services not

used in runlevel 2, then starting any services that are used in runlevel 2.)

The default runlevel is set by a line in the /etc/inittab file, which looks like this:

Listing 3. A default runlevel

```
id:5:initdefault:
```

Runlevels

There are actually 7 runlevels, 0 through 6. 0 is "halt," and is the same behavior you get from telling the system to shut down. 1 is "single user mode," which is used mostly for diagnostics and repairs. 2 is multiuser, and 3 is multiuser "plus networking," which mostly means NFS. 4 is unused, 5 is XDM, and 6 is reboot.

As the /etc/inittab file documents, you do not want to ever set the default runlevel of a system to 0 or 6. It can be hard to recover from this, although a friendly boot loader or a recovery CD will let you get in and fix it. The default for nearly all modern desktop Linux systems that use runlevels is 5. Some systems use other startup mechanisms, but runlevel 5 is the one found in Fedora 7.

The format of the line is historical, and all you really need to know to change it effectively is that you can change the 5 to a 2 or a 3. Then the next time the system boots, it will come up to a text console login prompt rather than starting X.

The simple fact is, the text console is a much better choice for a development system with limited memory than the full-bore X environment. This is a good time to do some poking around to see what we can get rid of. Change the `initdefault` value in your /etc/inittab to 3 and reboot. (You can, in fact, use the `init` command to transition, but there are edge cases where this doesn't work.) You'll immediately notice how much more quickly the system comes up. Immediately after logging in, run `top` again. Results will vary, but you might see that memory is a perhaps half full, instead of nearly completely full. Progress!

That's still a lot of memory

While it's certainly good to have a hundred megabytes free, the system could perhaps be slimmed down a bit more. Another run through `top` reveals some more memory hogs:

Listing 4. Excuse me, I don't think I ordered that

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
```

```
 2204 root      39  19 30056  12m 5632 S    0  5.7   0:00.70
yum-updatesd
 1825 root      20   0 19220 5052  972 S    0  2.3   0:00.00 python
 2238 haldaemo  20   0  6648 2728 2180 S    0  1.2   0:00.27 hald
 1909 root      20   0 13480 2236 1564 S    0  1.0   0:00.02 cupsd
 2015 root      20   0 12352 1976 1272 S    0  0.9   0:00.02
console-kit-dae
 1958 root      20   0 12132 1796  748 S    0  0.8   0:00.01 sendmail
 2301 root      20   0  5548 1792 1484 S    0  0.8   0:00.02 login
 2141 xfs       20   0  5388 1776  884 S    0  0.8   0:00.05 xfs
 2425 seebs     20   0  5392 1732 1480 S    0  0.8   0:00.08 bash
 2371 seebs     20   0  5392 1728 1480 S    0  0.8   0:00.08 bash
 1966 smmsp     20   0 10356 1576  696 S    0  0.7   0:00.00 sendmail
 2221 avahi     20   0  3772 1520 1344 S    0  0.7   0:00.07
avahi-daemon
 1751 root      20   0 11332 1348  588 S    0  0.6   0:00.43 pcscd
 1796 root      20   0 29652 1304 1032 S    0  0.6   0:00.02 automount
```

By far, the largest program is yum-updatesd, presumably a daemon related to the system's yum updater. (In the dot com era, that kind of deep insight could have gotten you relocated anywhere in the country you wanted to go.) Luckily, this is also a program we can easily do without; you can run yum by hand when you feel like it.

Editing runlevels

Sadly, there's not a specific runlevel for "runlevel 3, only without yum-updatesd". That means it's time to start manually removing services. There are a couple of ways to do this. Each runlevel is defined by a corresponding directory in /etc/rc.d, named rcN.d; for instance, runlevel 3 is defined by the files in /etc/rc.d/rc3.d. (There are also symlinks to these directories in /etc, on a Fedora 7 system, but it's a good habit to use the full path.)

Each such directory contains a number of files, with slightly cryptic names like "K74nscd" or "S88nasd." The naming convention is simple: names starting with a K are services to be stopped when entering this runlevel (presumably from a higher-numbered runlevel, which might have been using them), and names starting with an S are services to be started when entering this runlevel. The two-digit numbers are used to sort services; S13rpcbind is started before S14nfslock, which is in turn started before S25netfs. Simple, and effective.

In fact, these are usually not files, but symlinks to scripts stored in /etc/rc.d/init.d. Typically, there's a single script that can either start or stop a given service, and then links to it are made appropriately. When `init` changes levels, it calls the scripts with "start" or "stop" arguments, as indicated.

If you're feeling like just diving in, guns blazing, you can simply remove unwanted S or K links from a runlevel directory. Similarly, you can add new links. Another option is to use the `chkconfig` utility; this is a very flexible and powerful utility that can maintain these symlinks for you. A caveat: if you remove something crucial, you may have to go in with a rescue CD of some sort (such as the Fedora install disc; see Resources) to get your system booting cleanly again. Be sure you understand what

things are, and what depends on them, before you remove them!

As an example, if you wanted to remove the yum-updatesd program from runlevel 2, you could simply remove the link /etc/rc.d/rc2.d/S97yum-updatesd. To remove it from runlevel 2 with `chkconfig`, the command would be `/sbin/chkconfig yum-updatesd off`.

Tracking down more space

With `chkconfig` and `top`, it's possible to track down a fair number of large space users, figure out what they provide, and remove them if you don't need them. But what about the Python process? There's no Python service. The `ps` command reveals more:

Listing 5. Spying in Python

```
$ ps ax | grep python
 1825 ?        S      0:00 python ./hpssd.py
```

A quick `grep` among the startup scripts reveals that this is part of the hplip service, which provides "HP Linux Imaging and Printing." This can be used with some HP printers and scanners, but is otherwise unneeded. So, if you haven't turned that off already, do it now.

On a fairly typical dedicated development system, the final total was 49,896KB used, down from an initial start of 213,692KB. Free memory went from 5,500KB to 169,296KB—a noticeable improvement in room to run the compiler in. The difference this makes will vary depending on your workload; many of the background daemons, once swapped out, will stay swapped out and leave your system nearly as responsive as it would be without them. Over a long period of time, though, even a smallish difference in compile times adds up.

Next: Getting a usable X environment

As you see, if you're willing to sacrifice a number of unnecessary or unused features, you can reclaim a huge amount of system memory, leaving you with plenty of memory to run compilers and start developing code. However, many users will find the complete loss of X too high a price to pay. Part 3 in this series looks at what you can do to get a usable X environment for doing simple graphical work, without losing the ability to run the compiler.

Resources

Learn

- See all parts in the *Linux development on the PlayStation 3* series.
- The Cell Broadband Engine Resource Center is the definitive resource for all things Cell/B.E.
- Learn more about top, ps, and other fine UNIX utilities from Wikipedia. For `chkconfig`, see man chkconfig.
- Wikipedia also has the skinny on swap.
- Your Fedora install CD has a rescue mode which may be able to help you out of a jam.
- In the developerWorks Linux zone, find more resources for Linux developers, and scan our most popular articles and tutorials.
- See all Linux tips and Linux tutorials on developerWorks.
- Stay current with developerWorks technical events and Webcasts.

Get products and technologies

- Order the SEK for Linux, a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With IBM trial software, available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Get involved in the developerWorks community through blogs, forums, podcasts, and community topics in our new developerWorks spaces.

About the author

Peter Seebach
Peter Seebach has been collecting video game consoles for years, but has only been running Linux on them recently. He is still not sure whether this is a Linux machine that plays video games, or a game machine that runs Linux.

Trademarks