Instructions

Book Title: My Name is *******, My Job is System Administration

2. Minimum No. of Pages: 30 pages

3. Content and Design: As you wish

4. File Format: PDF

5. Last Date of Submission in Classwork: Wednesday, 31-March-2021

6. Marks Split: 10 marks for on time submission and 10 marks for content

“My Name is Arindam Sharma”

First Edition

Arindam Sharma

About Arthur :

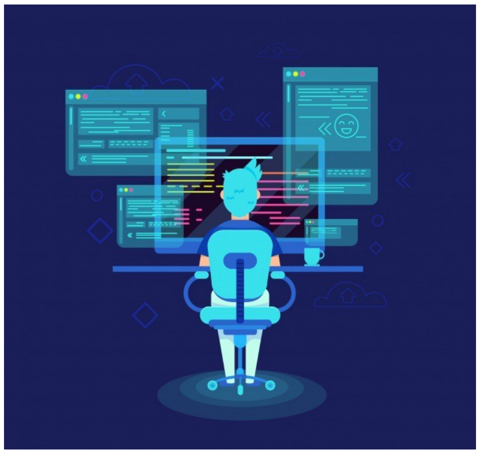
Name: Arindam Sharma
Job: System Administration

Publication :

Arindam Sharma Publication LTD.

Acknowledgements

This book is dedicated to this course which I am doing as one of my college courses (Device Driver) and the professor “**Sankaran Vaidyanathan**” who has given me the opportunity to write this book as one of his assignments.



Life of System Administrator

This book tells the Experience of System administration how the interest got developed. And how the learning progressed from ground level .

What generally people think of system administration as a role that manages in-house computer software systems, servers, storage devices and network connections to ensure high availability and security of the supported business applications. And also participates in the planning and implementation of policies and procedures to ensure system provisioning and maintenance that is consistent with company goals, industry best practices, and regulatory requirements.

When I was hired, I was purely focused on testing Red Hat CloudForms, which is management software for the aforementioned environments. But as one of our previous senior software engineers departed to take on another role within Red Hat, I saw an opportunity that interested me. I was already helping him and learning sysadmin tasks by then, so after looking at my progress and interest, I was a natural successor for the work in my team's perspective. And hence, I ended up becoming a sysadmin who is working partly as a software engineer in testing.

The first experience I had was when I got to know about the SSH protocol . That was the initial step I took to understand the network connection. The motivation was not that ethical but still I got motivated to learn more abt the network around me .
Initially it started with the Login to locally connected system and access files ,believe me it was a lot of fun that i could access some other files system other than mine which was not connected directly like USB ...etc.

Gradually getting to know different skills to handle /manage systems/databases have become really interesting .I got to know a lot of things while working .

I usually start the day by to log into my Red Hat Server .

my RHEV manager was a VM running on a stand-alone Kernel-based Virtual Machine (KVM) host, separate from the cluster it manages. I had been running RHEV since version 3.0, before hosted engines were a thing, and I hadn't gone through the effort of migrating. I was already in the process of building a new set of clusters with a new manager, but this older manager was still controlling most of our production VMs. It had filled its disk again, and the underlying database had stopped itself to avoid corruption.

So, I logged into the KVM host that ran the VM, and started the well-known procedure of creating a new empty disk file, and then attaching it via `virsh`. The procedure goes something like this: Become `root`, use `dd` to write a stream of zeros to a new file, of the proper size, in the proper location, then use `virsh` to attach the new disk to the already running VM. Then, of course, log into the VM and do your disk expansion.

I logged in, ran the following commands

```
sudo -i
```

```
cd /var/lib/libvirt/images
```

`ls -l` to find the existing disk images, and then started carefully crafting my `dd` command:

`dd` ... `bs=1k count=40000000 if=/dev/zero` ... `of=./vmname-disk` ...

Which was the next disk again?

```
<Tab> of=vmname-disk2.img <Back arrow, Back arrow,
Back arrow, Back arrow, Backspace>
```

Don't want to `dd` over the existing disk, that'd be bad. Let's change that `2` to a `2`, and `Enter`. OH CRAP, I CHANGED THE `2` TO A `2` NOT A `3` !
`<Ctrl+C><Ctrl+C><Ctrl+C><Ctrl+C><Ctrl+C><Ctrl+C>`

I still get sick thinking about this. I'd done the stupidest thing I possibly could have done, I started `dd` as `root`, over the top of an EXISTING DISK ON A RUNNING VM. What kind of idiot does that?! (The kind that's at work late, trying to get this one little thing done before he heads off to see his friend. The kind that thinks he knows better, and thought he was careful enough to not make such a newbie mistake. Gah.)

So, how fast does `dd` start writing zeros? Faster than I can move my fingers from the `Enter` key to the `Ctrl+C` keys. I tried a number of things to recover the running disk from memory, but all I did was make things worse, I think. The system was still up, but still broken like it was before I touched it, so it was useless.

Since my VMs were still running, and I'd already done enough damage for one night, I stopped touching things and went home. The next day I owned up to the boss and co-workers pretty much the moment I walked in the door. We started taking an inventory of what we had, and what was lost. I had taken the precaution of setting up backups ages ago. So, we thought we had that to fall back on.

I opened a ticket with my account support and filled them in on

how dumb I'd been. I can only imagine the reaction of the support person when they read my ticket. I worked a help desk for years, I know how this usually goes. They probably gathered their closest coworkers to mourn for my loss, or get some entertainment out of the guy who'd been so foolish. (I say this in jest. Red Hat's support was awesome through this whole ordeal, and I'll tell you how soon.)

So, I figured the next thing I would need from my broken server, which was still running, was the backups I'd diligently been collecting. They were on the VM but on a separate virtual disk, so I figured they were safe. The disk I'd overwritten was the last disk I'd made to expand the volume the database was on, so that logical volume was toast, but I've always set up my servers such that the main mounts—`/`, `/var`, `/home`, `/tmp`, and `/root`—were all separate logical volumes.

In this case, `/backup` was an entirely separate virtual disk. So, I `scp -r`'d the entire `/backup` mount to my laptop. It copied, and I felt a little sigh of relief. All of my production systems were still running, and I had my backup. My hope was that these factors would mean a relatively simple recovery: Build a new VM, install RHEV-M, and restore my backup. Simple right?

By now, my boss had involved the rest of the directors, and let them know that we were looking down the barrel of a possibly bad time. We started organizing a team meeting to discuss how we were going to get through this. I returned to my desk and looked through the backups I had copied from the broken server. All the files were there, but they were tiny. Like, a couple hundred kilobytes each, instead of the hundreds of megabytes or even gigabytes that they should have been.

Happy feeling, gone.

Turns out, my backups were running, but at some point after an RHEV upgrade, the database backup utility had changed. Remember how I said this system had existed since version 3.0? Well, 3.0 didn't have an engine-backup utility, so in my RHEV training, we'd learned how to make our own. Mine broke when the tools changed, and for who knows how long, it had been getting an incomplete backup—just some files from `/etc`.

No database.

I updated my support case with the bad news and started wondering what it would take to break through one of these 4th-floor windows right next to my desk. (Ok, not really.)

At this point, we basically had three RHEV clusters with no manager. One of those was for development work, but the other two were all production. We started using these team meetings to discuss how to recover from this mess. I don’t know what the rest of my team was thinking about me, but I can say that everyone was surprisingly supportive and un-accusatory. I mean, with one typo I’d thrown off the entire department. Projects were put on hold and workflows were disrupted, but at least we had time: We couldn’t reboot machines, we couldn’t change configurations, and couldn’t get to VM consoles, but at least everything was still up and operating.

Red Hat support had escalated my SNAFU to an RHEV engineer, a guy I’d worked with in the past. I don’t know if he remembered me, but I remembered him, and he came through yet again. About a week in, for some unknown reason (we never figured out why), our Windows VMs started dropping offline. They were still running as far as we could tell, but they dropped off the network, Just boom. Offline. In the course of a workday, we lost about a dozen windows systems. All of our RHEL machines were working fine, so it was just some Windows machines, and not even every Windows machine—about a dozen of them.

Well great, how could this get worse? Oh right, add a ticking time bomb. Why were the Windows servers dropping off? Would they all eventually drop off? Would the RHEL systems eventually drop off? I made a panicked call back to support, emailed my account rep, and called in every favor I’d ever collected from contacts I had within Red Hat to get help as quickly as possible.

I ended up on a conference call with two support engineers, and we got to work. After about 30 minutes on the phone, we’d worked out the most insane recovery method. We had the newer RHEV manager I mentioned earlier, that was still up and running, and had two new clusters attached to it. Our recovery goal was to get all of our workloads moved from the broken clusters to these two new clusters.

Want to know how we ended up doing it? Well, as our Windows VMs were dropping like flies, the engineers and I came up with this plan. My clusters used a Fibre Channel Storage Area Network (SAN) as their storage domains. We took a machine that was not in use, but had a Fibre Channel host bus adapter (HBA) in it, and attached the logical unit numbers (LUNs) for both the old cluster's storage domains and the new cluster's storage domains to it. The plan there was to make a new VM on the new clusters, attach blank disks of the proper size to the new VM, and then use `dd` (the irony is not lost on me) to block-for-block copy the old broken VM disk over to the newly created empty VM disk.

I don't know if you've ever delved deeply into an RHEV storage domain, but under the covers it's all Logical Volume Manager (LVM). The problem is, the LV's aren't human-readable. They're just universally-unique identifiers (UUIDs) that the RHEV manager's database links from VM to disk. These VMs are running, but we don't have the database to reference. So how do you get this data?

`virsh` ...

Luckily, I managed KVM and Xen clusters long before RHEV was a thing that was viable. I was no stranger to `libvirt`'s `virsh` utility. With the proper authentication—which the engineers gave to me—I was able to `virsh dumpxml` on a source VM while it was running, get all the info I needed about its memory, disk, CPUs, and even MAC address, and then create an empty clone of it on the new clusters.

Once I felt everything was perfect, I would shut down the VM on the broken cluster with either `virsh shutdown`, or by logging into the VM and shutting it down. The catch here is that if I missed something and shut down that VM, there was no way I'd be able to power it back on. Once the data was no longer in memory, the config would be completely lost, since that information is all in the database—and I'd hosed that. Once I had everything, I'd log into my migration host (the one that was connected to both storage domains) and use `dd` to copy, bit-for-bit, the source storage domain disk over to the destination storage domain disk. Talk about nerve-wracking, but it worked! We picked one of the broken windows VMs and followed this process, and within about half an hour we'd completed all of the steps and brought it back online.

We did hit one snag, though. See, we'd used snapshots here and there. RHEV snapshots are `lvm` snapshots. Consolidating them without the RHEV manager was a bit of a chore, and took even more leg work and research before we could `dd` the disks. I had to mimic the snapshot tree by creating symbolic links in the right places, and then start the `dd` process. I worked that one out late that evening after the engineers were off, probably enjoying time with their families. They asked me to write the process up in detail later. I suspect that it turned into some internal Red Hat documentation, never to be given to a customer because of the chance of royally hosting your storage domain.

Somehow, over the course of 3 months and probably a dozen scheduled maintenance windows, I managed to migrate every single VM (of about 100 VMs) from the old zombie clusters to the working clusters. This migration included our Zimbra collaboration system (10 VMs in itself), our file servers (another dozen VMs), our Enterprise Resource Planning (ERP) platform, and even Oracle databases.

We didn't lose a single VM and had no more unplanned outages. The Red Hat Enterprise Linux (RHEL) systems, and even some Windows systems, never fell to the mysterious drop-off that those dozen or so Windows servers did early on. During this ordeal, though, I had trouble sleeping. I was stressed out and felt so guilty for creating all this work for my co-workers, I even had trouble eating.

Similar to this another experience with a previous job company where I forgot abt a bracket . I was a newly-minted sysadmin. I was hired into my first sysadmin job with some Linux skills, but no system administration experience. It was a true junior position, and I was learning on the job while being trained by senior staff. The small team I was on supported the primary web servers for a university and its central IT department, to which my team belonged.
I'd been there no longer than a couple of weeks when I was asked to make a small change to the Apache server configuration hosting the IT department's website. I had been trained by the senior staff in this situation. I knew how to make the change, commit it to our version control, and then roll the change out to the servers. I opened the Apache documentation too, just to be sure, and had it in front of me. I could do this!

I made the change, and double-checked it against the documentation. I committed it to Subversion, and rolled the change out to the servers. Satisfied at a job well done, I added my notes to the ticket, and then closed the request.

Also, I forgot an angle bracket.

A few minutes later, the change propagated to the servers, Apache restarted (or tried to), and the website for our department came crashing down. As I frantically tried to roll back my changes—I didn't know what I'd broken, and could not, in the heat of the moment, remember how to get older versions out of Subversion—I could hear two coworkers talking in a cube nearby.

I sunk lower in my cube, both shame and embarrassment adding to my panic as I finally retrieved the older version, rolled it out to the web servers, and verified that the site had come back up.

Later that afternoon, I was still in my cube, shame and embarrassment still in full effect, but panic replaced by fear. Two or three weeks into my job, I'd taken down one of our highest traffic sites, having been trained and trusted to do the job. Certainly, I was not going to be employed for long. That fear was only compounded when my boss' boss' boss showed up in my cube.

I can only wonder what face of pure terror I made when she came into view.

They were right. Sure, this was a simple typo, but I did not do a syntax check. This lesson has followed me. I never again committed code or configuration to a production service without testing.

There were many days when I got tired of work and seeing the continuous tasks.

Just because you prefer working in a text-mode interface doesn't mean you're not entitled to a little fun here and there.

Last December, I took some time out before the holidays to explore some of my favorite command-line diversions into a series for Opensource.com. It ended up being a bit of an advent calendar for terminal toys, and I got some great suggestions from readers.

There are things like gaming, whether it's board games, console games, or PC games. They help your brain escape and work in different ways than it does as a sysadmin. Personally, I love a good DnD session or a card game with the family. I'm also a pretty avid Doom player.

Now summer has arrived, this means a time of summer breaks, vacations, and generally trying to fit in a little relaxation between committing code and closing tickets. So to that end, I thought I'd revisit five of my favorite command-line games from that series .

All of these games were tested in Fedora, most from the default repositories, but you can likely find them for other distributions as well. Or, grab the source code to compile them yourself.

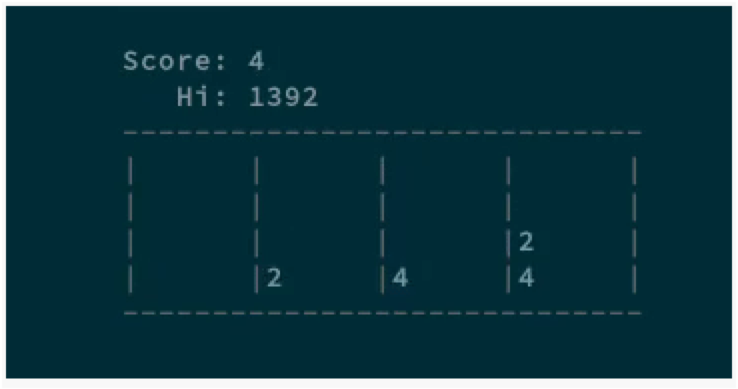
Game 2048

We'll start with one of my all-time favorite casual games, 2048 (which itself is a clone of another clone).

To play, you just slide blocks up, down, left, and right to combine matching pairs and increment numbers, until you've made a block that is 2048 in size. The catch (and the challenge), is that you can't just move one block; instead, you move every block on the screen, provided it has space to slide.

It's simple, fun, and easy to get lost in it for hours. This 2048 clone, 2048-cli, is by Marc Tiehuis and written in C, and made available as open source under an MIT license. You can find the source code on GitHub, where you can also get installation instructions for your platform. Since it was packaged for Fedora, for me, installing it was as simple as:

```
$ sudo dnf install 2048-cli
```



```
Score: 4
   Hi: 1392
-----------------------------
|      |      |      |      |
|      |      |      |      |
|      |      |      |2     |
|      |2     |4     |4     |
-----------------------------
```

MyMan

MyMan is a fun clone of the classic arcade game Pac-Man. (You didn't think this was going to be about the similarly-named Linux package manager, did you?) If you're anything like me, you spent more than your fair share of quarters trying to hit a high score Pac-Man back in the day, and still give it a go whenever you get a chance.

MyMan isn't the only Pac-Man clone for the Linux terminal, but it's the one I chose to include because 1) I like its visual style, which rings true to the original and 2) it's conveniently packaged for my Linux distribution so it was an easy install. But you should check out your other options as well. Here's another one that looks like it may be promising, but I haven't tried it.

Since MyMan was packaged for Fedora, installation was as simple as:

```
$ dnf install myman
```

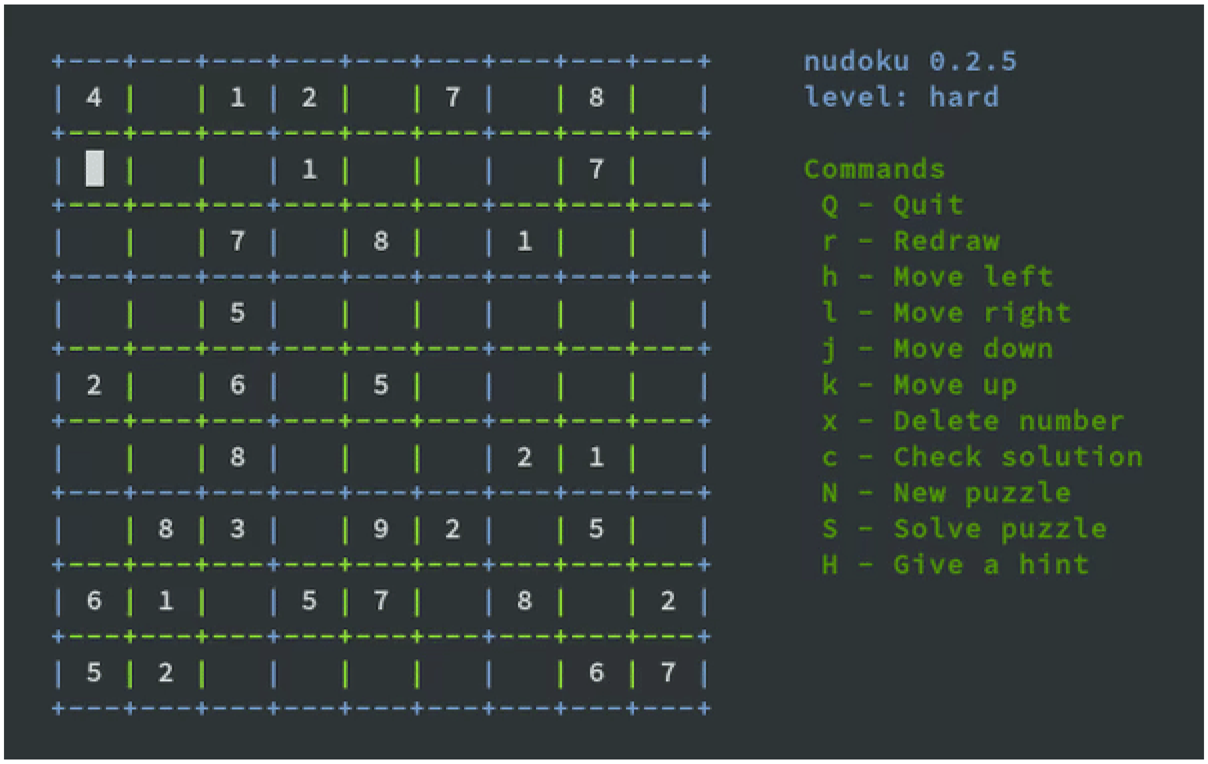
MyMan is made available under an MIT license and you can check out the source code on SourceForge.

```
00

PUSH START BUTTON

1 PLAYER ONLY

BONUS MYMAN FOR 10000 Pts

DEDICATED TO TORU
```

Nudoku

Nudoku is a simple sudoku game that, once installed, can be invoked by name (nudoku) to launch, and should be fairly self-explanatory from there. If you've never played Sudoku before, it's fairly simple: You need to make sure that each row, each column, and each of the nine 3x3 squares that make up the large square each have one of every digit, 1-9.

You can find nudoku's c source code on GitHub under a GPLv3 license.



```
+---+---+---+---+---+---+---+---+---+    nudoku 0.2.5
| 4 |   | 1 | 2 |   | 7 |   | 8 |   |    level: hard
+---+---+---+---+---+---+---+---+---+
|   |   |   | 1 |   |   |   | 7 |   |    Commands
+---+---+---+---+---+---+---+---+---+     Q - Quit
|   |   | 7 |   | 8 |   | 1 |   |   |     r - Redraw
+---+---+---+---+---+---+---+---+---+     h - Move left
|   |   | 5 |   |   |   |   |   |   |     l - Move right
+---+---+---+---+---+---+---+---+---+     j - Move down
| 2 |   | 6 |   | 5 |   |   |   |   |     k - Move up
+---+---+---+---+---+---+---+---+---+     x - Delete number
|   |   | 8 |   |   |   | 2 | 1 |   |     c - Check solution
+---+---+---+---+---+---+---+---+---+     N - New puzzle
|   | 8 | 3 |   | 9 | 2 |   | 5 |   |     S - Solve puzzle
+---+---+---+---+---+---+---+---+---+     H - Give a hint
| 6 | 1 |   | 5 | 7 |   | 8 |   | 2 |
+---+---+---+---+---+---+---+---+---+
| 5 | 2 |   |   |   |   |   | 6 | 7 |
+---+---+---+---+---+---+---+---+---+
```

Snake

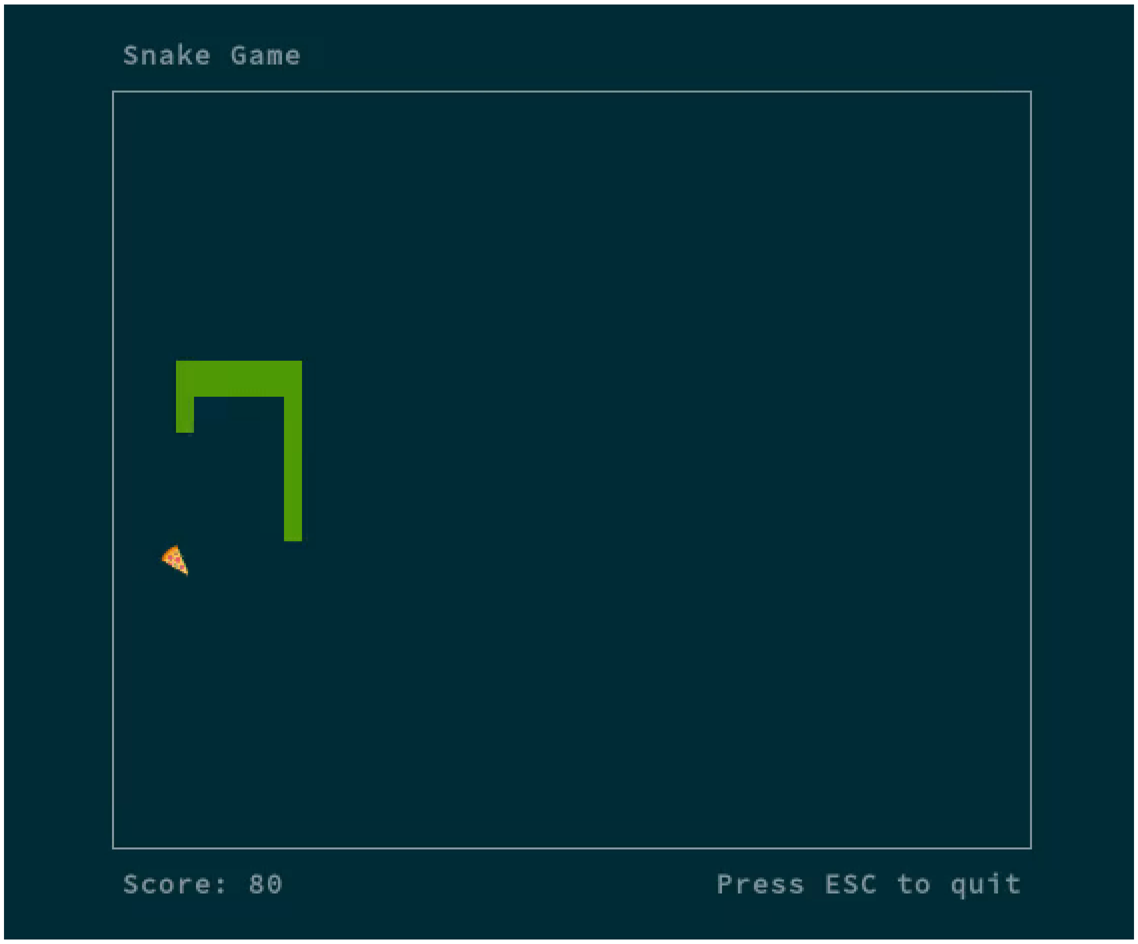
Snake is an oldie but goodie; versions of it have been around seemingly forever. The first version I remember playing was one called Nibbles that came packaged with QBasic in the 1990s, and was probably pretty important to my understanding of what a programming language even was. Here I had the source code to a game that I could modify and just see what happens, and maybe learn something about what all of those funny little words that made up a programming language were all about.

This version of Snake is written in Go, and while it's simple, it's just as much fun as the original. Like most simple old games, there are a ton of versions to choose from. In Snake's case, there's even a version in the classic bsdgames package that's almost certainly packaged for your distribution.

But what I like about this version of Snake is that it's packaged in a container so I can easily run it in one line from my terminal without worrying about anything disto-specific. That, and it makes use of 15 randomized food emojis for the snake to eat. I'm a sucker for food emojis. Anyway, give it a try using:

```
$ docker run -ti dyego/snake-game
```

This Snake is licensed as open source under an MIT license, and you can check out the source code on GitHub.


```
Snake Game

Score: 80                                   Press ESC to quit
```

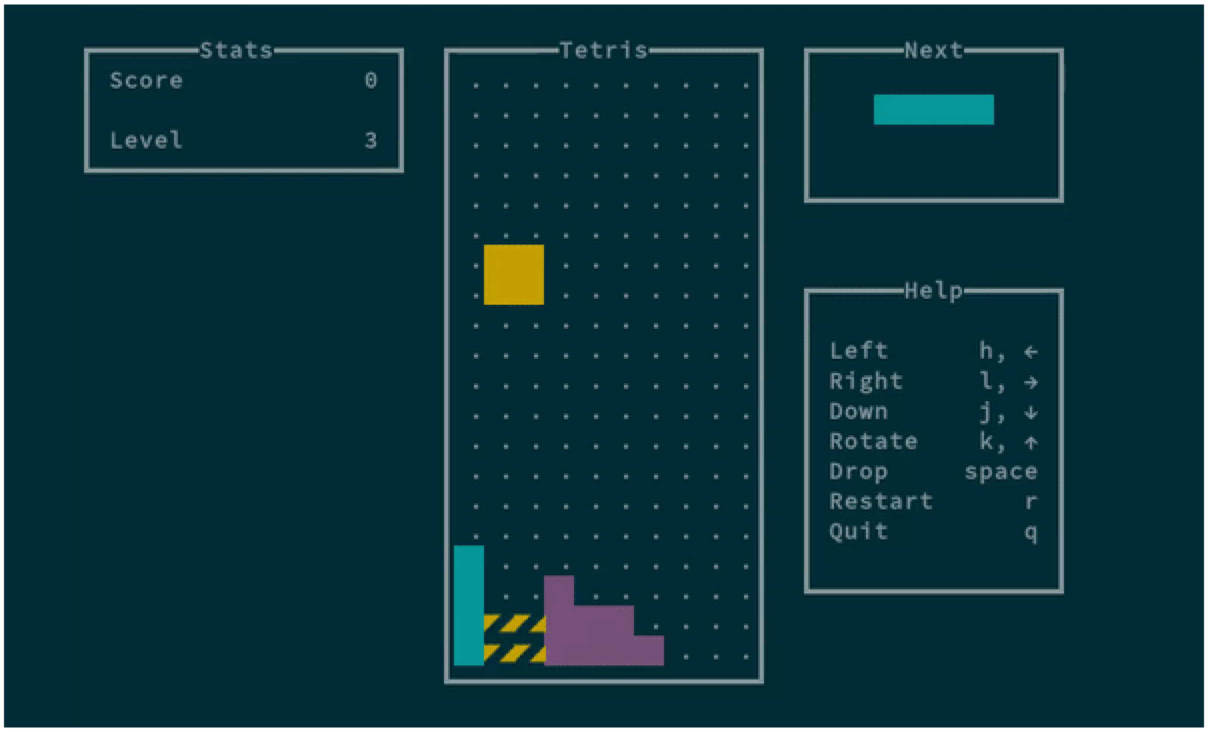
Tetris

After taking the world by storm in the 1980s., Tetris was cloned many, many times. I would suspect you could find a Tetris clone for just about any operating system in any language you looked for. Seriously, go look. There are some fun ones out there.

The version I'm bringing you for today's command-line toy is written in Haskell, and it's one of the better-done versions I've seen, with on-screen preview, score, help, and a clean look.

If you're willing to run a compiled binary from an untrusted source (I wouldn't recommend it), you can grab that directly, but for a safer approach, it's also easy to use a containerized version with dex, or to install from source with stack.

This particular Tetris clone is by Sam Tay and available under a BSD license. Check it out!



Other than these games I want to highlight that it is important, in my opinion, for any sysadmin to have a healthy interest in technology outside of their job responsibilities. In fact, it's one of the things that I try to gauge when I'm interviewing technology folks. Whether it's for a development job or something more admin-related, I think it's important that folks working in technology have a passion for technology.

Personally, I have a home lab where I try things out. It's currently running oVirt-node and an all-in-one oVirt manager/hypervisor. I run Red Hat Virtualization in my day job, so it's nice to get my hands on the upstream projects for the hardened, stable products that Red Hat provides us.

I use that home lab to tinker with whatever new technology comes along and catches my interest. Sometimes it's OpenShift, sometimes it's containers, and sometimes it's a Minecraft server for my kids. I also tinker in electronics—it’s not uncommon to find me in my workshop holding a soldering iron. Tech outside of your daily work life helps keep you sharp and lets you explore new technology that you might not have the time or resources to tackle at your day job.

However, this practice is a double-edged sword. I recently went through a yearlong process of simplifying my home’s dependence on my skills. It was getting to the point where I'd come home from work, and step from my professional sysadmin role into a volunteer sysadmin role. Sure, I enjoy setting up services. I enjoy running services, too. But sometimes when you get home at night, you just want to sit down and let your brain relax.

I was running my own websites out of my basement *and* running a side business hosting for others. I felt like I never got a break. So, I put a bunch of time into simplifying. I moved my self-hosted email to a free email provider. I moved my websites to a cloud provider and closed up the side business (it wasn't making any money anyway). I even replaced my slick Linux-based firewall with a simple to set up wifi mesh.

Now, when things go sideways, we don't need to be a senior sysadmin to figure out the problem. All in all, it's really helped my sanity. Plus, using technology like an average user helps me understand where they're coming from. Users don't have the level of control that sysadmins have over technology, so they're sort of dependent on us, or on the vendors that make their devices. Sometimes it's good to step into those shoes so you can understand where they're coming from.

The stereotype says that we're all afraid of the outside world; that we hunker down in basements with the windows closed, and nothing but the glow of a monitor to light our workspace. That rumor might actually be true for some of us, but personally, I have always tried to maintain outside hobbies away from a computer.

Firstly, sitting—or even standing—at a computer isn't all that healthy. Getting away from your terminal helps your brain relax, but just being away from your computer, or your office, might not be enough. It's easy to be away from something and still be preoccupied with it, so don't just go for a walk. *Do something*.

You need to move around more, and what better way to do that than to go outside. I'm not about to tell you what you should do out there, but make it something engaging. I'll give you some examples of what works for me.

To keep myself busy I also like to build things, whether it's home renovations or upgrades to the Jeep. Sometimes, I even fabricate things myself. Mostly for the Jeep, but I've applied my skills to my house, the kids' fancy costumes, and projects for friends. It's always a good idea to have a broad range of skills.

I know, you can probably save time and pay someone else to expand your closet or build you a garage, but doing it yourself is so rewarding.

Finally, I'd like to cover creative outlets in general. I touched on this topic slightly when I mentioned teaching and fabricating. I like to teach others what I know, and I like to create things that might help others, like this article!

I find creating something like that to be very rewarding, and believe it or not, it's helped me learn about technologies which benefit me in my day job, ones that I'd never have come across otherwise.

In today’s connected world, it’s likely that you rely on a sysadmin in one way or another for almost everything you’re doing online—from checking in with your friends on social media, to tracking that shipment you’re expecting later today. That supercomputer in your pocket? Somewhere there’s a sysadmin making sure that each service it relies on is running—which gives you that connected lifestyle you enjoy—and also keeps your data safe.

I’m composing this post in a collaborative editing suite that we all take for granted. Knowing what I know, I can tell you that there’s someone, or a team of someones, keeping these lights on. We take technology for granted, and sometimes forget that someone, somewhere, is paid (or sometimes, volunteers) to keep these systems working. I’ve got a T-shirt that I found online somewhere that says, “System Administrator, I solve problems you don’t know you have, in ways you can’t understand.” That statement really sums it up.

Well, Sysadmin Appreciation Day is coming on July 26th. This day is your chance to thank one of the least thanked people in your life. System administration carries a ton of responsibility. We’re expected to be experts in technology, from printers to desktops, smartphones, and even smart toasters. I’ve worked jobs where I was responsible for anything with a CPU, and in some cases, anything even slightly more complicated than a coffee maker. I was once responsible for changing the filter in the building AC unit at a small web host. Talk about tangentially related.

These “extra” duties don’t even take into consideration the complex systems we’re *actually* here to support. Clusters, virtualization, email, messaging, calendaring, web sites—all of these things are services that consumers use daily, and they need to work. Largely, they do, but that’s due to the tireless work of sysadmins.

Now, most of the sysadmins in the world, you can’t identify. They’re working for some huge tech firm keeping the Googles and Facebooks of the world running. I’m not suggesting you hunt all of those folks down, but if everyone makes an effort to find the people that keep their office running, it will go a long way to help the sysadmins of the world feel better. While you’re at it, find your local network engineer and thank them, too. They deserve it!

If you’re managing sysadmins, it will go a long way toward team building if you make it a point to thank your sysadmins directly—and not just at their yearly review. I’ve worked for managers that didn’t get that, and others who did. I can tell you which ones I was happier to work for.

If you *really* want to appreciate your sysadmin, I can recommend one thing: Read *The Phoenix Project*, by Gene Kim, Kevin Behr, and George Spafford. This book is considered to be the DevOps bible, but I found that it resonated with me as a description of what it means to be an IT worker today. Pay attention to the character “Brent.” That’s your sysadmin. Trying to understand the stresses of a sysadmin is probably the best gift you can give them on Sysadmin Appreciation Day.

All of these things, while not really IT-related, apply to the skills you need as a good sysadmin. Get away from that terminal. Get outside and do something. Give your brain a break, and try to avoid burnout!

Now i wanted to explain my workflow for a day and how it feels to be system administrator .

Every day when I come in, I check our monitoring system's dashboard to see if any of our 50 most important hosts are complaining regarding the over 490 checks that are running. If anything is wrong, I try to fix it, or I delegate the task to someone who can fix it for me. When things are not complaining, I try to improve our current setup, write more comprehensive automation to monitor the infrastructure, and think about how to automate remediation of things when they are broken or complaining.

What I mean by complaining is really an alert sent by our monitoring tool about the state of a given host or service. Since we live on the bleeding edge of Red Hat's technology, we always need to have our systems up-to-date with the latest (mostly internal) builds of Red Hat Virtualization and Red Hat's cloud software.

Some of that process has already been automated, and I try to leverage automation to redeploy things. But as I stated in a previous article on infrastructure-as-code, it is possible that my code breaks when running within new builds, and I need to debug and fix the automation when that happens. Some of my time, despite our automation, is wasted by the fact that people from my team create resources and forget to clean up. We have different cleanup scripts to remove things, but if resources don't match conditions for cleanup, we end up having to remove them manually.

When none of the above is happening, we sometimes run into bugs or issues that were previously unknown and require a fix from the engineering team to get systems working again. In that case, I need to engage with relevant people to:

- Prove the problem is reproducible and indeed is a bug.
- Keep in touch with engineering to get a fix ready and apply the updates.
- Use any workaround, if one exists, to keep the systems from going insane.

From time to time, as you would guess, we run into a hardware malfunction or limitations that we must carefully understand and remediate. This issue includes, but is not limited to:

- Designing and understanding what hardware resources we have.
- Determining our requirements.
- Coming up with a design.
- Checking the design with the datacenter team for feasibility.
- Talking to vendors to procure new hardware.
- Talking to support when things are not looking good.

Every now and then, I also need to educate people on what to do and what *not* to do with our systems, as not everyone necessarily knows what each does and why they are designed and used the way they are. If I still have time, after all of that stuff, I try to spend that on automating tests for CloudForms using Python and Selenium.

The thing I love about my job is two-fold. One reason is that our infrastructure is internal and does not require pager duty, as people can often wait for me to come online to fix things. The other reason is that my team is globally distributed, so I can rely on others (although I am the only sysadmin) to try and fix problems or create workarounds in the meantime.

In Red Hat's CloudForms quality engineering, we maintain a lot of infrastructure assembled from different internal products. To my knowledge, no other team has so much diversity in its infrastructure. Many teams only need to have their own product deployed, or maybe one or two others. This distinction has given me a lot of opportunities to learn.

All in all, if things are not working, people will surely know your name and find you. Also, any system malfunction can cause my team to miss targets, slip release dates, and cause a loss of revenue for Red Hat. And when we do release stuff on time, I can see that I made an impact by making sure our systems are available 24x7.

System administration is a position where you have great power with great responsibility. I like to think I use it wisely.

Humans make mistakes. Failures will occur. In a safe, blame-free culture, team members can learn from their mistakes, and as a result, services can be hardened against the same problems, and the team and organization as a whole can grow.

Reference

[1]. https://classroom.google.com/c/MjU2NjgxNjcxNzE5/a/MjczOTkyMTAyOTQx/details
[2]. https://www.redhat.com/sysadmin/stories-trenches
[2]. https://www.redhat.com/sysadmin/command-line-games
[3]. https://www.redhat.com/sysadmin/virtual-machine-recovery
[4], https://www.redhat.com/sysadmin/forgotten-bracket
[5]. https://www.redhat.com/sysadmin/stories-trenches
[6]. https://www.redhat.com/sysadmin/geeking-outside-office
[7]. https://www.redhat.com/sysadmin/appreciating-your-sysadmin